Class notes for Randomized Algorithms

Sariel Har-Peled[①]

April 2, 2024

[①]Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; sariel@illinois.edu; http://sarielhp.org/. Work on this paper was partially supported by a NSF CAREER award CCR-0132901.

Contents

Contents 3

1 Introduction to Randomized Algorithms 17
- 1.1 What are randomized algorithms? 17
  - 1.1.1 The benefits of unpredictability 17
  - 1.1.2 Back to randomized algorithms 18
  - 1.1.3 Randomized vs average-case analysis 18
- 1.2 Examples of randomized algorithms 19
  - 1.2.1 2SAT 19
    - 1.2.1.1 The algorithm 19
    - 1.2.1.2 Intuition 19
  - 1.2.2 Walk on the grid 19
    - 1.2.2.1 Walk on the line 19
    - 1.2.2.2 Walk on the two dimensional grid 20
    - 1.2.2.3 Walk on the two dimensional grid 20
  - 1.2.3 RSA and primality testing 20
  - 1.2.4 Min cut 21

2 Probability and Expectation 23
- 2.1 Basic probability 23
  - 2.1.1 Formal basic definitions: Sample space, σ-algebra, and probability 23
  - 2.1.2 Expectation and conditional probability 24
  - 2.1.3 Variance and standard deviation 25
- 2.2 Some distributions and their moments 25
  - 2.2.1 Bernoulli distribution 25
  - 2.2.2 Geometric distribution 26
- 2.3 Application of expectation: Approximating 3SAT 27
- 2.4 Markov's inequality 28
  - 2.4.1 Markov's inequality 28
  - 2.4.2 Example: A good approximation to kSAT with good probability 29
  - 2.4.3 Example: Coloring a graph 30
    - 2.4.3.1 Getting a valid coloring 31

3 Analyzing QuickSort and QuickSelect via Expectation 33
- 3.1 **QuickSort** 33
- 3.2 **QuickSelect**: Median selection in linear time 34
  - 3.2.1 Analysis via expectation and indicator variables 34

3.2.2 Analysis of QuickSelect via conditional expectations . . . 35

4 Chebychev, Sampling and Selection **37**
4.1 Chebyshev's inequality . . . 37
4.1.1 Example: A better inequality via moments . . . 37
4.1.2 Chebychev's inequality . . . 37
4.2 Estimation via sampling . . . 38
4.3 Randomized selection – Using sampling to learn the world . . . 39
4.3.1 Inverse estimation . . . 39
4.3.1.1 Inverse estimation – intuition . . . 40
4.3.2 Randomized selection . . . 40
4.3.2.1 The algorithm . . . 40
4.3.2.2 Analysis . . . 41

5 Verifying Identities, and Some Complexity **43**
5.1 Verifying equality . . . 43
5.1.1 Vectors . . . 43
5.1.1.1 Amplification . . . 44
5.1.2 Matrices . . . 44
5.1.3 Checking identity for polynomials . . . 45
5.1.3.1 The SchwartzZippel lemma . . . 45
5.1.3.2 Applications . . . 46
5.1.4 Checking if a bipartite graph has a perfect matching . . . 46
5.2 Las Vegas and Monte Carlo algorithms . . . 47
5.2.1 Complexity Classes . . . 47
5.3 Bibliographical notes . . . 48

6 The Birthday Paradox, Occupancy and the Coupon Collector Problem **49**
6.1 Some needed math . . . 49
6.2 The birthday paradox . . . 50
6.3 Occupancy Problems . . . 51
6.3.1 The Probability of all bins to have exactly one ball . . . 52
6.4 The Coupon Collector's Problem . . . 53
6.5 Notes . . . 54

7 On k-wise independence **55**
7.1 Pairwise independence . . . 55
7.1.1 Pairwise independence . . . 55
7.1.2 A pairwise independent set of bits . . . 55
7.1.3 An application: Max cut . . . 56
7.1.4 Analysis . . . 56
7.2 On k-wise independence . . . 57
7.2.1 Definition . . . 57
7.2.2 On working modulo prime . . . 57
7.2.3 Construction of k-wise independence variables . . . 58
7.2.4 Construction . . . 58
7.2.5 Applications of k-wide independent variables . . . 59

7.2.5.1 Product of expectations . . . 59
7.2.5.2 Application: Using less randomization for a randomized algorithm . . . 60
7.3 Higher moment inequalities . . . 60

8 Hashing **63**
8.1 Introduction . . . 63
8.2 Universal Hashing . . . 65
8.2.1 How to build a 2-universal family . . . 66
8.2.1.1 A quick reminder on working modulo prime . . . 66
8.2.1.2 Constructing a family of 2-universal hash functions . . . 66
8.2.1.3 Analysis . . . 66
8.2.1.4 Explanation via pictures . . . 67
8.3 Perfect hashing . . . 68
8.3.1 Some easy calculations . . . 68
8.3.2 Construction of perfect hashing . . . 69
8.3.2.1 Analysis . . . 69
8.4 Bloom filters . . . 69
8.5 Bibliographical notes . . . 70

9 Closest Pair **73**
9.1 How many times can a minimum change? . . . 73
9.2 Closest Pair . . . 73
9.3 Bibliographical notes . . . 76

10 Coupon's Collector Problems II **77**
10.1 The Coupon Collector's Problem Revisited . . . 77
10.1.1 Some technical lemmas . . . 77
10.1.2 Back to the coupon collector's problem . . . 78
10.1.3 An asymptotically tight bound . . . 78
10.2 Bibliographical notes . . . 79

11 Conditional Expectation and Concentration **81**
11.1 Conditional expectation . . . 81
11.1.1 Concentration from conditional expectation . . . 82

12 Quick Sort with High Probability **83**
12.1 **QuickSort** runs in $O(n \log n)$ time with high probability . . . 83
12.2 Treaps . . . 83
12.2.1 Construction . . . 84
12.2.2 Operations . . . 84
12.2.2.1 Insertion . . . 85
12.2.2.2 Deletion . . . 85
12.2.2.3 Split . . . 86
12.2.2.4 Meld . . . 86
12.2.3 Summery . . . 86
12.3 Extra: Sorting Nuts and Bolts . . . 86
12.3.1 Running time analysis . . . 87

12.4 Bibliographical Notes . . . 87

13 Concentration of Random Variables – Chernoff's Inequality **89**
13.1 Concentration of mass and Chernoff's inequality . . . 89
13.1.1 Example: Binomial distribution . . . 89
13.1.2 A restricted case of Chernoff inequality via games . . . 89
13.1.2.1 Chernoff games . . . 89
13.1.2.2 Chernoff's inequality . . . 93
13.1.2.3 Some low level boring calculations . . . 93
13.1.3 A proof for $-1/+1$ case . . . 94
13.2 The Chernoff Bound — General Case . . . 95
13.2.1 The lower tail . . . 96
13.2.2 A more convenient form of Chernoff's inequality . . . 97
13.2.2.1 Bound when the expectation is small . . . 98
13.3 A special case of Hoeffding's inequality . . . 99
13.3.1 Some technical lemmas . . . 101
13.4 Hoeffding's inequality . . . 101
13.5 Bibliographical notes . . . 103
13.6 Exercises . . . 103

14 Applications of Chernoff's Inequality **105**
14.1 **QuickSort** is Quick . . . 105
14.2 How many times can the minimum change? . . . 106
14.3 Routing in a parallel computer . . . 106
14.3.1 Analysis . . . 107
14.4 Faraway Strings . . . 109
14.5 Bibliographical notes . . . 109
14.6 Exercises . . . 110

15 Min Cut **111**
15.1 Branching processes – Galton-Watson Process . . . 111
15.1.1 The problem . . . 111
15.1.2 On coloring trees . . . 112
15.2 Min Cut . . . 113
15.2.1 Problem Definition . . . 113
15.2.2 Some Definitions . . . 113
15.3 The Algorithm . . . 114
15.3.1 Analysis . . . 116
15.3.1.1 The probability of success . . . 116
15.3.1.2 Running time analysis. . . . 117
15.3.2 An alternative implementation using MST . . . 117
15.4 A faster algorithm . . . 118
15.5 Bibliographical Notes . . . 119

16 Discrepancy and Derandomization **121**
16.1 Discrepancy . . . 121
16.2 The Method of Conditional Probabilities . . . 122
16.3 Bibliographical Notes . . . 123

17 Independent set – Turán's theorem **125**
17.1 Turán's theorem . . . 125
17.1.1 Some silly helper lemmas . . . 125
17.1.2 Statement and proof . . . 126
17.1.3 An alternative proof of Turán's theorem . . . 126
17.1.4 An algorithm for the weighted case . . . 127

18 Derandomization using Conditional Expectations **129**
18.1 Method of conditional expectations . . . 129
18.1.1 Applications . . . 130
18.1.1.1 Max kSAT . . . 130
18.1.1.2 Max cut . . . 130
18.1.1.3 Turán theorem . . . 130

19 Martingales **131**
19.1 Martingales . . . 131
19.1.1 Preliminaries . . . 131
19.1.2 Martingales . . . 131
19.1.2.1 Examples of martingales . . . 132
19.1.2.2 Azuma's inequality . . . 133
19.2 Bibliographical notes . . . 134

20 Martingales II **135**
20.1 Filters and Martingales . . . 135
20.2 Martingales . . . 136
20.2.1 Martingales – an alternative definition . . . 137
20.3 Occupancy Revisited . . . 138
20.3.1 Lets verify this is indeed an improvement . . . 139

21 The power of two choices **141**
21.1 Balls and bins with many rows . . . 141
21.1.1 The game . . . 141
21.1.2 Analysis . . . 141
21.1.3 With only d rows . . . 143
21.2 The power of two choices . . . 144
21.2.1 Upper bound . . . 144
21.2.2 Applications . . . 146
21.2.3 The power of restricted d choices: Always go left . . . 146
21.3 Avoiding terrible choices . . . 147
21.4 Escalated choices . . . 147
21.5 Bibliographical notes . . . 148

22 Evaluating And/Or Trees **149**
22.1 Evaluating an And/Or Tree . . . 149
22.1.1 Randomized evaluation algorithm for T_{2k} . . . 150
22.1.2 Analysis . . . 150
22.2 Bibliographical notes . . . 150

23 The Probabilistic Method II **151**
23.1 Expanding Graphs . . . 151
23.1.1 An alternative construction . . . 152
23.1.2 An expander . . . 152
23.2 Probability Amplification . . . 152
23.3 Oblivious routing revisited . . . 154

24 Dimension Reduction **155**
24.1 Introduction to dimension reduction . . . 155
24.2 Normal distribution . . . 155
24.2.1 The standard multi-dimensional normal distribution . . . 157
24.3 Dimension reduction . . . 157
24.3.1 The construction . . . 157
24.3.2 Analysis . . . 158
24.3.2.1 A single unit vector is preserved . . . 158
24.3.3 All pairwise distances are preserved . . . 159
24.4 Even more on the normal distribution . . . 160
24.5 Bibliographical notes . . . 161

25 Streaming and the Multipass Model **163**
25.1 The secretary problem . . . 163
25.2 Reservoir sampling: Fishing a sample from a stream . . . 164
25.3 Sampling and median selection revisited . . . 164
25.3.1 A median selection with few comparisons . . . 166
25.4 Big data and the streaming model . . . 167
25.5 Heavy hitters . . . 167
25.5.1 A randomized algorithm . . . 167
25.5.2 A deterministic algorithm . . . 167

26 Frequency Estimation over a Stream **169**
26.1 The art of estimation . . . 169
26.1.1 The problem . . . 169
26.1.2 Averaging estimator: Success with constant probability . . . 169
26.1.2.1 The challenge . . . 169
26.1.2.2 Taming of the variance . . . 170
26.1.3 Median estimator: Success with high probability . . . 170
26.2 Frequency estimation over a stream for the kth moment . . . 171
26.2.1 An estimator for the kth moment . . . 171
26.2.1.1 Basic estimator . . . 171
26.2.1.2 Analysis . . . 172
26.2.2 An improved estimator: Plugin . . . 173

26.3 Better estimation for F_2 . . . 174
26.3.1 Pseudo-random k-wide independent sequence of signed bits . . . 174
26.3.2 Estimator construction for F_2 . . . 174
26.3.2.1 The basic estimator . . . 174
26.3.3 Improving the estimator . . . 175
26.4 Bibliographical notes . . . 176

27 Approximating the Number of Distinct Elements in a Stream **177**
27.1 Counting number of distinct elements . . . 177
27.1.1 First order statistic . . . 177
27.1.2 The algorithm . . . 178
27.2 Sampling from a stream with "low quality" randomness . . . 179
27.3 Bibliographical notes . . . 180

28 Approximate Nearest Neighbor (ANN) Search in High Dimensions **181**
28.1 ANN on the hypercube . . . 181
28.1.1 ANN for the hypercube and the Hamming distance . . . 181
28.1.2 Preliminaries . . . 182
28.1.2.1 Algorithm . . . 182
28.1.2.2 Analysis . . . 182
28.1.2.3 Running time . . . 183
28.2 Testing for good items . . . 184
28.3 LSH for the hypercube: An elaborate construction . . . 184
28.3.0.1 On sense and sensitivity . . . 184
28.3.1 A simple sensitive family . . . 185
28.3.2 A family with a large sensitivity gap . . . 185
28.3.3 Amplifying sensitivity . . . 186
28.3.4 The near neighbor data-structure and handling a query . . . 186
28.3.4.1 Setting the parameters . . . 187
28.3.5 The result . . . 188
28.4 LSH and ANN in Euclidean space . . . 189
28.4.1 Preliminaries . . . 189
28.4.2 Locality sensitive hashing (LSH) . . . 189
28.4.3 ANN in high-dimensional euclidean space . . . 190
28.4.3.1 The result . . . 191
28.5 Bibliographical notes . . . 191

29 Random Walks I **193**
29.1 Definitions . . . 193
29.1.1 Walking on grids and lines . . . 193
29.1.1.1 Walking on the line . . . 193
29.1.1.2 Walking on two dimensional grid . . . 194
29.1.1.3 Walking on three dimensional grid . . . 194
29.2 Bibliographical notes . . . 195

30 Random Walks II **197**
30.1 Catalan numbers . . . 197
30.2 Walking on the integer line revisited . . . 198
30.2.1 Estimating the middle binomial coefficient . . . 198
30.3 Solving 2SAT using random walk . . . 199
30.3.1 Solving 2SAT . . . 199
30.4 Markov chains . . . 200

31 Random Walks III **205**
31.1 Random walks on graphs . . . 205
31.2 Electrical networks and random walks . . . 207
31.2.1 A tangent on parallel and series resistors . . . 208
31.2.2 Back to random walks . . . 208
31.3 Bibliographical Notes . . . 210

32 Random Walks IV **211**
32.1 Cover times . . . 211
32.1.1 Rayleigh's Short-cut Principle. . . . 212
32.2 Graph Connectivity . . . 212
32.2.1 Directed graphs . . . 213

33 A Bit on Algebraic Graph Theory **215**
33.1 Graphs and Eigenvalues . . . 215
33.1.1 Eigenvalues and eigenvectors . . . 215
33.1.2 Eigenvalues and eigenvectors of a graph . . . 216
33.2 Bibliographical Notes . . . 216

34 Random Walks V **217**
34.1 Explicit expander construction . . . 217
34.2 Rapid mixing for expanders . . . 218
34.2.1 Bounding the mixing time . . . 218
34.3 Probability amplification by random walks on expanders . . . 219
34.3.1 The analysis . . . 220
34.3.2 Some standard inequalities . . . 222

35 Complexity classes **223**
35.1 Las Vegas and Monte Carlo algorithms . . . 223
35.2 Complexity classes . . . 223
35.3 Bibliographical notes . . . 224

36 Backwards analysis **225**
36.1 How many times can the minimum change? . . . 225
36.2 Computing a good ordering of the vertices of a graph . . . 226
36.2.1 The algorithm . . . 226
36.2.2 Analysis . . . 226
36.3 Computing nets . . . 227
36.3.1 Basic definitions . . . 227
36.3.1.1 Nets . . . 227

36.3.2 Computing an r-net in a sparse graph . . . 227
36.3.2.1 The algorithm . . . 227
36.3.2.2 Analysis . . . 227
36.4 Bibliographical notes . . . 229

37 Multiplicative Weight Update: Expert Selection **231**
37.1 The problem: Expert selection . . . 231
37.2 Majority vote . . . 231
37.3 Randomized weighted majority . . . 232
37.4 Bibliographical notes . . . 233

38 On Complexity, Sampling, and ε-Nets and ε-Samples **235**
38.1 VC dimension . . . 235
38.1.1 Examples . . . 236
38.1.1.1 Halfspaces . . . 237
38.2 Shattering dimension and the dual shattering dimension . . . 238
38.2.1 Mixing range spaces . . . 239
38.3 On ε-nets and ε-sampling . . . 240
38.3.1 ε-nets and ε-samples . . . 240
38.3.2 Some applications . . . 241
38.3.2.1 Range searching . . . 241
38.3.2.2 Learning a concept . . . 242
38.4 A better bound on the growth function . . . 242
38.5 Some required definitions . . . 243

39 Double sampling **245**
39.1 Double sampling . . . 245
39.1.1 Disagreement between samples on a specific set . . . 246
39.1.2 Exponential decay for a single set . . . 246
39.1.3 Moments of the sample size . . . 247
39.1.4 Growth function . . . 247
39.2 Proof of the ε-net theorem . . . 248
39.2.1 The proof . . . 248
39.2.1.1 Reduction to double sampling . . . 248
39.2.1.2 Using double sampling to finish the proof . . . 249

40 Finite Metric Spaces and Partitions **251**
40.1 Finite Metric Spaces . . . 251
40.2 Examples . . . 252
40.2.1 Hierarchical Tree Metrics . . . 253
40.2.2 Clustering . . . 253
40.3 Random Partitions . . . 253
40.3.1 Constructing the partition . . . 254
40.3.2 Properties . . . 254
40.4 Probabilistic embedding into trees . . . 255
40.4.1 Application: approximation algorithm for k-median clustering . . . 256
40.5 Embedding any metric space into Euclidean space . . . 257

40.5.1 The bounded spread case . . . 257
40.5.2 The unbounded spread case . . . 259
40.6 Bibliographical notes . . . 260
40.7 Exercises . . . 260

41 Entropy, Randomness, and Information **263**
41.1 The entropy function . . . 263
41.2 Extracting randomness . . . 265
41.3 Bibliographical Notes . . . 268

42 Entropy II **269**
42.1 Huffman coding . . . 269
42.1.1 The algorithm to build Hoffman's code . . . 270
42.1.2 Analysis . . . 270
42.1.3 A formula for the average size of a code word . . . 271
42.2 Compression . . . 273
42.3 Bibliographical Notes . . . 274

43 Entropy III - Shannon's Theorem **275**
43.1 Coding: Shannon's Theorem . . . 275
43.2 Proof of Shannon's theorem . . . 276
43.2.1 How to encode and decode efficiently . . . 276
43.2.1.1 The scheme . . . 276
43.2.1.2 The proof . . . 276
43.2.2 Lower bound on the message size . . . 280
43.3 Bibliographical Notes . . . 280

44 Approximate Max Cut **281**
44.1 Problem Statement . . . 281
44.1.1 Analysis . . . 282
44.2 Semi-definite programming . . . 284
44.3 Bibliographical Notes . . . 284

45 Expanders I **287**
45.1 Preliminaries on expanders . . . 287
45.1.1 Definitions . . . 287
45.2 Tension and expansion . . . 288

46 Expanders II **291**
46.1 Bi-tension . . . 291
46.2 Explicit construction . . . 292
46.2.1 Explicit construction of a small expander . . . 293
46.2.1.1 A quicky reminder of fields . . . 293
46.2.1.2 The construction . . . 294

47 Expanders III - The Zig Zag Product **297**
47.1 Building a large expander with constant degree . . . 297
47.1.1 Notations . . . 297
47.1.2 The Zig-Zag product . . . 297
47.1.3 Squaring . . . 299
47.1.4 The construction . . . 300
47.2 Bibliographical notes . . . 301
47.3 Exercises . . . 301

48 The Probabilistic Method **303**
48.1 Introduction . . . 303
48.1.1 Examples . . . 303
48.1.1.1 Max cut . . . 303
48.2 Maximum Satisfiability . . . 304

49 The Probabilistic Method III **307**
49.1 The Lovász Local Lemma . . . 307
49.2 Application to k-SAT . . . 309
49.2.1 An efficient algorithm . . . 309
49.2.1.1 Analysis . . . 310

50 The Probabilistic Method IV **311**
50.1 The Method of Conditional Probabilities . . . 311
50.2 Independent set in a graph . . . 312
50.3 A Short Excursion into Combinatorics via the Probabilistic Method . . . 313
50.3.1 High Girth and High Chromatic Number . . . 313
50.3.2 Crossing Numbers and Incidences . . . 314
50.3.3 Bounding the at most k-level . . . 315

51 Sampling and the Moments Technique **317**
51.1 Vertical decomposition . . . 317
51.1.1 Randomized incremental construction (RIC) . . . 318
51.1.1.1 Analysis . . . 319
51.1.2 Backward analysis . . . 320
51.2 General settings . . . 322
51.2.1 Notation . . . 322
51.2.1.1 Examples of the general framework . . . 323
51.2.2 Analysis . . . 324
51.2.2.1 On the probability of a region to be created . . . 324
51.2.2.2 On exponential decay . . . 325
51.2.2.3 Bounding the moments . . . 326
51.3 Applications . . . 327
51.3.1 Analyzing the RIC algorithm for vertical decomposition . . . 327
51.3.2 Cuttings . . . 327
51.4 Bounds on the probability of a region to be created . . . 329
51.5 Bibliographical notes . . . 330
51.6 Exercises . . . 332

52 Primality testing **335**
52.1 Number theory background . . . 335
52.1.1 Modulo arithmetic . . . 335
52.1.1.1 Prime and coprime . . . 335
52.1.1.2 Computing gcd . . . 336
52.1.1.3 The Chinese remainder theorem . . . 337
52.1.1.4 Euler totient function . . . 337
52.1.2 Structure of the modulo group $\mathbb{Z}_n$ . . . 338
52.1.2.1 Some basic group theory . . . 338
52.1.2.2 Subgroups . . . 338
52.1.2.3 Cyclic groups . . . 339
52.1.2.4 Modulo group . . . 339
52.1.2.5 Fields . . . 340
52.1.2.6 $\mathbb{Z}_p^*$ is cyclic for prime numbers . . . 340
52.1.2.7 $\mathbb{Z}_n^*$ is cyclic for powers of a prime . . . 341
52.1.3 Quadratic residues . . . 341
52.1.3.1 Quadratic residue . . . 341
52.1.3.2 Legendre symbol . . . 342
52.1.3.3 Jacobi symbol . . . 343
52.1.3.4 **Jacobi**(a, n): Computing the Jacobi symbol . . . 345
52.1.3.5 Subgroups induced by the Jacobi symbol . . . 346
52.2 Primality testing . . . 347
52.2.1 Distribution of primes . . . 347
52.3 Bibliographical notes . . . 348

53 Talagrand's Inequality **351**
53.1 Introduction . . . 351
53.1.1 Talangrand's inequality, and the T-distance . . . 351
53.1.2 On the way to proving Talagrand's inequality . . . 353
53.1.2.1 The low level details used in the above proof . . . 354
53.1.3 Proving Talagrand's inequality . . . 355
53.2 Concentration via certification . . . 355
53.3 Some examples . . . 357
53.3.1 Longest increasing subsequence . . . 357
53.3.2 Largest convex subset . . . 358
53.3.3 Balls into bins revisited . . . 359
53.4 Bibliographical notes . . . 361
53.5 Problems . . . 361

54 Low Dimensional Linear Programming **363**
54.1 Linear programming in constant dimension ($d > 2$) . . . 363
54.2 Handling Infeasible Linear Programs . . . 367
54.3 References . . . 367

55 Algorithmic Version of Lovász Local Lemma **369**
55.1 Introduction . . . 369
55.1.1 Analysis . . . 369

56 Some math stuff **371**
56.1 Some useful estimates 371

Index **378**

Chapter 1

Introduction to Randomized Algorithms

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

People tell me it's a sin
To know and feel too much within.
I still believe she was my twin, but I lost the ring.
She was born in spring, but I was born too late.
Blame it on a simple twist of fate.

A little twist of fate, Bob Dylan

1.1. What are randomized algorithms?

Randomized algorithms are algorithms that makes random decision during their execution. Specifically, they are allowed to use variables, such that their value is taken from some random distribution. It is not immediately clear why adding the ability to use randomness helps an algorithm. But it turns out that the benefits are quite substantial. Before listing them, let start with an example.

1.1.1. The benefits of unpredictability

Consider the following game. The adversary has a equilateral triangle, with three coins on the vertices of the triangle (which are, numbered by, I don't known, 1,2,3). Initially, the adversary set each of the three coins to be either heads or tails, as she sees fit.

At each round of the game, the player can ask to flip certain coins (say, flip coins at vertex 1 and 3). If after the flips all three coins have the same side up, then the game stop. Otherwise, the adversary is allowed to rotate the board by $0, 120$ or -120 degrees, as she seems fit. And the game continues from this point on. To make things interesting, the player does not see the board at all, and does not know the initial configuration of the coins.

A randomized algorithm. The randomized algorithm in this case is easy – the player randomly chooses a number among $1, 2, 3$ at every round. Since, at every point in time, there are two coins that have the same side up, and the other coin is the other side up, a random choice hits the lonely coin, and thus finishes the game, with probability $1/3$ at each step. In particular, the number of iterations of the game till it terminates behaves like a geometric variable with geometric distribution with probability $1/3$ (and thus the expected number of rounds is 3). Clearly, the probability that the game continues for more than i rounds, when the player uses this random algorithm, is $(2/3)^i$. In particular, it vanishes to zero relatively quickly.

A deterministic algorithm. The surprise here is that there is no deterministic algorithm that can generate a winning sequence. Indeed, if the player uses a deterministic algorithm, then the adversary can simulate the algorithm herself, and know at every stage what coin the player would ask to flip (it is easy to verify that

flipping two coins in a step is equivalent to flipping the other coin – so we can restrict ourselves to a single coin flip at each step). In particular, the adversary can rotate the board in the end of the round, such that the player (in the next round) flips one of the two coins that are in the same state. Namely, the player never wins.

The shocker. One can play the same game with a board of size 4 (i.e., a square), where at each stage the player can flip one or two coins, and the adversary can rotate the board by $0, 90, 180, 270$ degrees after each round. Surprisingly, there is a deterministic winning strategy for this case. The interested reader can think what it is (this is one of these brain teasers that are not immediate, and might take you 15 minutes to solve, or longer [or much longer]).

The unfair game of the analysis of algorithms. The underlying problem with analyzing algorithm is the inherent unfairness of worst case analysis. We are given a problem, we propose an algorithm, then an all-powerful adversary chooses the worst input for our algorithm. Using randomness gives the player (i.e., the algorithm designer) some power to fight the adversary by being unpredictable.

1.1.2. Back to randomized algorithms

(A) **Best.** There are cases where only randomized algorithms are known or possible, especially for games. For example, consider the 3 coins example given above.
(B) **Speed.** In some cases randomized algorithms are considerably faster than any deterministic algorithm.
(C) **Simplicity.** Even if a randomized algorithm is not faster, often it is considerably simpler than its deterministic counterpart.
(D) **Derandomization.** Some deterministic algorithms arises from derandomizing the randomized algorithms, and this are the only algorithm we know for these problems (i.e., discrepancy).
(E) **Adversary arguments and lower bounds.** The standard worst case analysis relies on the idea that the adversary can select the input on which the algorithm performs worst. Inherently, the adversary is more powerful than the algorithm, since the algorithm is completely predictable. By using a randomized algorithm, we can make the algorithm unpredictable and break the adversary lower bound.

Namely, randomness makes the algorithm vs. adversary game a more balanced game, by giving the algorithm additional power against the adversary.

1.1.3. Randomized vs average-case analysis

Randomized algorithms are not the same as ***average-case analysis***. In average case analysis, one assumes that is given some distribution on the input, and one tries to analyze an algorithm execution on such an input.

On the other hand, randomized algorithms do not assume random inputs – inputs can be arbitrary. As such, randomized algorithm analysis is more widely applicable, and more general.

While there is a lot of average case analysis in the literature, the problem that it is hard to find distribution on inputs that are meaningful in comparison to real world inputs. In particular, for numerous cases, the average case analysis exposes structure that does not exist in real world input.

1.2. Examples of randomized algorithms

1.2.1. 2SAT

The input is a 2SAT formula. That is a 2CNF boolean formula – that is, the formula is a conjunction of clauses, where each clause is made out of two literals, which are ored together. A literal here is either a variable or its negation. For example, the input formula might be

$$F = (x_1 \vee \overline{x_2}) \wedge (\overline{x_2} \vee x_3) \wedge \cdots \wedge (\overline{x_1} \vee x_{17}).$$

(Here, $\vee$ is a boolean or, and $\wedge$ is a boolean and.) Assume that F is using n variables (say $x_1, \ldots, x_n \in \{0, 1\}$), and m clauses. The task at hand is to compute a satisfying assignment for F. That is, determine what values has to be assigned to $x_1, \ldots, x_n$.

This problem can be solved in linear time (i.e., $O(n+m)$) by a somewhat careful and somewhat clever usage of directed graphs and strong connected components. Here, we present a much simpler randomized algorithm – we will present some intuition why this algorithm works. We hopefully will provide a more detailed formal proof later in the course.

1.2.1.1. The algorithm

The algorithm starts with an arbitrary assignment to the variables of F. If F evaluates to TRUE, then the algorithm is done. Otherwise, there must be a clause, say $C_i = \ell_i \vee \ell_i'$, that is not satisfied. The algorithm randomly chooses (with equal probability) one of the literals of C_i, and flip the value assigned to variable in this literal. Thus, if the algorithm chosen $\ell_i' = \overline{x_{17}}$, then the algorithm would flip the value of x_{17}. The algorithm continues to the next iteration.

Claim 1.2.1 (Proof later in the course). *If F has a satisfying assignment, then the above algorithm performs $O(n^2)$ iterations in expectation, till it finds a satisfying assignment. Thus, the expected running time of this algorithm is $O(n^2 m)$.*

1.2.1.2. Intuition

Fix a specific satisfying assignment Ξ to F. Assume X_i is the number of variables in the assignment in the beginning of the ith iteration that agree with Ξ. If $X_i = n$, then the algorithm found Ξ, and it is done. Otherwise, X_i changes by exactly one at each iteration. That is $X_{i+1} = X_i + 1$ or $X_{i+1} = X_i - 1$. If both variables of C_i are assigned the "wrong" value (i.e., the negation of their value in Ξ), then $X_{i+1} = X_i + 1$. The other option is that one of the variables of C_i is assigned the wrong value. The probability that the algorithm guess the right variable to flip is $1/2$. Thus, we have $X_{i+1} = X_i + 1$ with probability $1/2$, and $X_{i+1} = X_i - 1$ with probability $1/2$.

Thus, the execution of the algorithm is a random process. Starting with X_1 being some value in the range $[\![0 : n]\!] = \{0, \ldots, n\}$, the question is how long do we have to run this process till $X_i = n$? It turns out that the answer is $O(n^2)$, because essentially this process is related to the random walk on the line, described next.

1.2.2. Walk on the grid

1.2.2.1. Walk on the line

Let $\mathbb{Z}$ denote the set of all integer numbers. Consider the random process, that starts at time zero, with the "player" being at position $X_0 = 0$. In the ith step of the game, the player randomly choose with probability half to go left – that is, to move to $X_{i-1} - 1$, or with equal probability to the right (i.e., $X_i = X_{-1} + 1$). The sequence

$\mathcal{X} = X_0, X_1, \ldots$ is a *random walk* on the integers. A natural question is how many times would the walk visit the origin, in the infinite walk $\mathcal{X}$?

Well, the probability of the random walk at time $2n$ to be in the origin is exactly $\alpha_n = \binom{2n}{n}/2^{2n}$. Indeed, there are 2^{2n} random walks of length $2n$. For the walk to be in the origin at time $2n$, the walk has to be balanced – equal number of steps have to be taken to the left and to the right. The number of binary sequences of length $2n$ that have exactly n 0s and n 1s is $\binom{2n}{n}$.

Exercise 1.2.2. Prove that $\binom{2n}{n} = \Theta(2^{2n}/\sqrt{n})$. (An easy proof follows from using Stirling's formula, but there is also a not too difficult direct elementary proof).

As such, we have that $c_-/\sqrt{n} \leq \alpha_n \leq c_+/\sqrt{n}$, where c_-, c_+ are two constants. Thus, the expected number of times the random walk visits the origin is

$$\sum_{n=1}^{\infty} \alpha_n \geq c_- \sum_{n=1}^{\infty} \frac{1}{\sqrt{n}} = +\infty.$$

(Why the last argument is valid would be explained in following lectures.)

Namely, the random walk visits the origin infinite number of times.

1.2.2.2. Walk on the two dimensional grid

The same question can be asked when the underlying set is $\mathbb{Z} \times \mathbb{Z}$ – that is the two dimensional integer grid. Here, the walk starts at the origin $X_0 = (0, 0)$, and in the ith step, the walk moves with (equal) probability to one of the four adjacent locations. That is, if $X_{i-1} = (x_{i-1}, y_{i-1})$, then X_i is one of the following four locations with equal probability:

$$(x_i - 1, y_i), \quad (x_i + 1, y_i), \quad (x_i, y_i - 1), \quad \text{and} \quad (x_i, y_i + 1).$$

As before, one can ask what is the number of times this random walk visits the origin. Let β_n be the probability of being in the origin at time $2n$.

Exercise 1.2.3. Prove that $\beta_n = \alpha_n^2 = \Theta(1/n)$. (There is a nifty trick to prove this. See if you can figure it out.)

Arguing as above, we have that the expected number of times the walk visits the origin is $\sum_{n=1}^{\infty} \Theta(1/n) = +\infty$. Namely, the walk visit the origin infinite number of times.

1.2.2.3. Walk on the two dimensional grid

The same question can be asked when the underlying set is $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$ – as before the walk starts at the origin, and at each step the walk goes to one of the six adjacent cells. It turns out that the probability of being in the origin at time (say) $6n$ is $\Theta(1/n^{3/2})$ (the proof is not clean or easy in this case), and as such, the expected number of times this walk visits the origin $\sum_{n=1}^{\infty} \Theta(1/n^{3/2}) = O(1)$. Surprise!

1.2.3. RSA and primality testing

Oversimplifying the basic idea, RSA works as follows. Compute two huge random primes p and q, release $n = pq$ as the public key. Given n one can encrypt a message, but to decrypt it, one needs both p and q. So, we rely here on the computational hardness of factoring.

Using RSA thus boils down to computing large prime numbers. Fortunately, the following is known (and somewhat surprisingly, is not difficult to prove):

Theorem 1.2.4. *The range* $[\![n]\!] = \{1, \ldots, n\}$ *contains* $\Theta(n/\log n)$ *prime numbers.*

As a number n can be written using $O(\log n)$ digits, that essentially means that a random number with t digits has probability $\approx 1/t$ to be a prime number. Namely, primes are quite common.

Fortunately, one can test quickly whether or not a random number is a prime.

Theorem 1.2.5. *Given a positive integer n, it can be written using* $T = \lceil \log_{10} n \rceil$ *digits. Furthermore, one can decide in* $O(T^4) = O(\log^4 n)$ *randomized time if n is prime. More precisely, if n is not prime, the algorithm would return "not prime" with probability half, if it is prime, it would return "prime".*

A natural way to decide if a number n with t bits is prime, is to run the above algorithm (say), $10t$ times. If any of the runs returns that the number is not prime, then we return "not prime". Otherwise, we return the number of is a prime. The probability that a composite number would be reported as prime is $1/2^{10t} \leq 1/10^t$, which is a tiny number, for t, say, larger than 512.

This gives us an efficient way to pick random prime numbers – the time to compute such a number is polynomial in the number of bits its uses. Now, we can deploy RSA as computing large random prime numbers is the main technical difficulty in computing it.

1.2.4. Min cut

In the most basic version of the min-cut problem, you are given an undirected graph G with n vertices and m edges, and the task is to compute the minimum number of edges one has to delete so the graph becomes disconnected.

Consider the following algorithm – it randomly assigns the edges of G weights from the range $[0, 1]$. It then computes the MST T of this graph (according to the random weights on the edges). Let e be the heaviest edge in T. Removing it breaks T into two subtrees, with two disjoint sets of vertices S and T. Let (S, T) denote the set of all the edges in G that have one end point in S and one in T. The algorithm outputs the edges of (S, T) as the candidate to be the minimum cut.

The following result is quite surprising.

Theorem 1.2.6. *The above algorithm always outputs a cut, and it outputs a min-cut with probability* $\geq 2/n^2$.

In particular, it turns out that if you run the above algorithm $O(n^2 \log n)$ times, and returns the smallest cut computed, then with probability $\geq 1 - 1/n^{O(1)}$, the returned cut is the minimum cut! This algorithm has running time (roughly) $O(n^4)$ – it can be made faster, but this is already pretty good.

Chapter 2

Probability and Expectation

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

Everybody knows that the dice are loaded
Everybody rolls with their fingers crossed
Everybody knows the war is over
Everybody knows the good guys lost
Everybody knows the fight was fixed
The poor stay poor, the rich get rich
That's how it goes
Everybody knows

Everybody knows, Leonard Cohen

2.1. Basic probability

Here we recall some definitions about probability. The reader already familiar with these definition can happily skip this section.

2.1.1. Formal basic definitions: Sample space, σ-algebra, and probability

A ***sample space*** Ω is a set of all possible outcomes of an experiment. We also have a set of events $\mathcal{F}$, where every member of $\mathcal{F}$ is a subset of Ω. Formally, we require that $\mathcal{F}$ is a σ-algebra.

Definition 2.1.1. A single element of Ω is an ***elementary event*** or an ***atomic event***.

Definition 2.1.2. A set $\mathcal{F}$ of subsets of Ω is a ***σ-algebra*** if:

(i) $\mathcal{F}$ is not empty,
(ii) if $X \in \mathcal{F}$ then $\overline{X} = (\Omega \setminus X) \in \mathcal{F}$, and
(iii) if $X, Y \in \mathcal{F}$ then $X \cup Y \in \mathcal{F}$.

More generally, we require that if $X_i \in \mathcal{F}$, for $i \in \mathbb{Z}$, then $\cup_i X_i \in \mathcal{F}$. A member of $\mathcal{F}$ is an ***event***.

As a concrete example, if we are rolling a dice, then $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $\mathcal{F}$ would be the power set of all possible subsets of Ω.

Definition 2.1.3. A ***probability measure*** is a mapping $\mathbb{P} : \mathcal{F} \to [0, 1]$ assigning ***probabilities*** to events. The function $\mathbb{P}$ needs to have the following properties:

(i) Additive: for $X, Y \in \mathcal{F}$ disjoint sets, we have that $\mathbb{P}[X \cup Y] = \mathbb{P}[X] + \mathbb{P}[Y]$, and
(ii) $\mathbb{P}[\Omega] = 1$.

Observation 2.1.4. *Let $C_1, \ldots, C_n$ be random events (not necessarily independent). Than*

$$\mathbb{P}[\cup_{i=1}^{n} C_i] \leq \sum_{i=1}^{n} \mathbb{P}[C_i].$$

(This is usually referred to as the **union bound**.*) If $C_1, \ldots, C_n$ are* disjoint *events then*

$$\mathbb{P}[\cup_{i=1}^{n} C_i] = \sum_{i=1}^{n} \mathbb{P}[C_i].$$

Definition 2.1.5. A ***probability space*** is a triple $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is a sample space, $\mathcal{F}$ is a σ-algebra defined over Ω, and $\mathbb{P}$ is a probability measure.

Definition 2.1.6. A ***random variable*** f is a mapping from Ω into some set $\mathcal{G}$. We require that the probability of the random variable to take on any value in a given subset of values is well defined. Formally, for any subset $U \subseteq \mathcal{G}$, we have that $f^{-1}(U) \in \mathcal{F}$. That is, $\mathbb{P}[f \in U] = \mathbb{P}\left[f^{-1}(U)\right]$ is defined.

Going back to the dice example, the number on the top of the dice when we roll it is a random variable. Similarly, let X be one if the number rolled is larger than 3, and zero otherwise. Clearly X is a random variable.

We denote the ***probability*** of a random variable X to get the value x, by $\mathbb{P}[X = x]$ (or sometime $\mathbb{P}[x]$, if we are lazy).

2.1.2. Expectation and conditional probability

Definition 2.1.7 (Expectation). The expectation of a random variable X, is its average. Formally, the ***expectation*** of X is

$$\mathbb{E}[X] = \sum_{x} x \, \mathbb{P}[X = x].$$

Definition 2.1.8 (Conditional Probability.). The ***conditional probability*** of X given Y, is the probability that $X = x$ given that $Y = y$. We denote this quantity by $\mathbb{P}[X = x \mid Y = y]$.

One useful way to think about the conditional probability $\mathbb{P}[X \mid Y]$ is as a function, between the given value of Y (i.e., y), and the probability of X (to be equal to x) in this case. Since in many cases x and y are omitted in the notation, it is somewhat confusing.

The conditional probability can be computed using the formula

$$\mathbb{P}[X = x \mid Y = y] = \frac{\mathbb{P}[(X = x) \cap (Y = y)]}{\mathbb{P}[Y = y]}.$$

For example, let us roll a dice and let X be the number we got. Let Y be the random variable that is true if the number we get is even. Then, we have that

$$\mathbb{P}\left[X = 2 \;\middle|\; Y = true\right] = \frac{1}{3}.$$

Definition 2.1.9. Two random variables X and Y are ***independent*** if $\mathbb{P}\left[X = x \;\middle|\; Y = y\right] = \mathbb{P}[X = x]$, for all x and y.

Observation 2.1.10. *If X and Y are independent then $\mathbb{P}\Big[X = x \,\Big|\, Y = y\Big] = \mathbb{P}[X = x]$ which is equivalent to $\dfrac{\mathbb{P}[X = x \cap Y = y]}{\mathbb{P}[Y = y]} = \mathbb{P}[X = x]$. That is, X and Y are* independent*, if for all x and y, we have that*

$$\mathbb{P}[X = x \cap Y = y] = \mathbb{P}[X = x]\,\mathbb{P}[Y = y].$$

Remark. Informally, and not quite correctly, one possible way to think about the conditional probability $\mathbb{P}[X = x \mid Y = y]$ is that it measure the benefit of having more information. If we know that $Y = y$, do we have any change in the probability of $X = x$?

Lemma 2.1.11 (Linearity of expectation). **Linearity of expectation** *is the property that for any two random variables X and Y, we have that $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$.*

Proof: $\mathbb{E}[X + Y] = \sum_{\omega\in\Omega} \mathbb{P}[\omega](X(\omega) + Y(\omega)) = \sum_{\omega\in\Omega} \mathbb{P}[\omega]X(\omega) + \sum_{\omega\in\Omega} \mathbb{P}[\omega]Y(\omega) = \mathbb{E}[X] + \mathbb{E}[Y]$. ■

Lemma 2.1.12. *If X and Y are two random independent variables, then $\mathbb{E}[XY] = \mathbb{E}[X]\,\mathbb{E}[Y]$.*

Proof: Let $U(X)$ the sets of all the values that X might have. We have that

$$\begin{aligned}
\mathbb{E}[XY] &= \sum_{x\in U(X), y\in U(Y)} xy\,\mathbb{P}[X = x \text{ and } Y = y] = \sum_{x\in U(X), y\in U(Y)} xy\,\mathbb{P}[X = x]\,\mathbb{P}[Y = y] \\
&= \sum_{x\in U(X)} \sum_{y\in U(Y)} xy\,\mathbb{P}[X = x]\,\mathbb{P}[Y = y] = \sum_{x\in U(X)} x\,\mathbb{P}[X = x] \sum_{y\in U(Y)} y\,\mathbb{P}[Y = y] \\
&= \mathbb{E}[X]\,\mathbb{E}[Y].
\end{aligned}$$

■

2.1.3. Variance and standard deviation

Definition 2.1.13 (Variance and Standard Deviation). For a random variable X, let

$$\mathbb{V}[X] = \mathbb{E}\Big[(X - \mu_X)^2\Big] = \mathbb{E}\Big[X^2\Big] - \mu_X^2$$

denote the ***variance*** of X, where $\mu_X = \mathbb{E}[X]$. Intuitively, this tells us how concentrated is the distribution of X. The ***standard deviation*** of X, denoted by σ_X is the quantity $\sqrt{\mathbb{V}[X]}$.

Observation 2.1.14. *(i) For any constant $c \geq 0$, we have $\mathbb{V}[cX] = c^2\,\mathbb{V}[X]$.*
(ii) For X and Y independent variables, we have $\mathbb{V}[X + Y] = \mathbb{V}[X] + \mathbb{V}[Y]$.

2.2. Some distributions and their moments

2.2.1. Bernoulli distribution

Definition 2.2.1 (Bernoulli distribution). Assume, that one flips a coin and get 1 (heads) with probability p, and 0 (i.e., tail) with probability $q = 1 - p$. Let X be this random variable. The variable X is has ***Bernoulli distribution*** with parameter p.

We have that $\mathbb{E}[X] = 1 \cdot p + 0 \cdot (1 - p) = p$, and

$$\mathbb{V}[X] = \mathbb{E}\Big[X^2\Big] - \mu_X^2 = \mathbb{E}\Big[X^2\Big] - p^2 = p - p^2 = p(1 - p) = pq.$$

Definition 2.2.2 (Binomial distribution). Assume that we repeat a Bernoulli experiment n times (independently!). Let $X_1, \ldots, X_n$ be the resulting random variables, and let $X = X_1 + \cdots + X_n$. The variable X has the ***binomial distribution*** with parameters n and p. We denote this fact by $X \sim \mathrm{Bin}(n, p)$. We have

$$b(k; n, p) = \mathbb{P}[X = k] = \binom{n}{k} p^k q^{n-k}.$$

Also, $\mathbb{E}[X] = np$, and $\mathbb{V}[X] = \mathbb{V}[\sum_{i=1}^n X_i] = \sum_{i=1}^n \mathbb{V}[X_i] = npq$.

2.2.2. Geometric distribution

Definition 2.2.3. Consider a sequence $X_1, X_2, \ldots$ of independent Bernoulli trials with probability p for success. Let X be the number of trials one has to perform till encountering the first success. The distribution of X is a ***geometric distribution*** with parameter p. We denote this by $X \sim \mathrm{Geom}(p)$.

Lemma 2.2.4. *For a variable $X \sim \mathrm{Geom}(p)$, we have, for all i, that $\mathbb{P}[X = i] = (1-p)^{i-1}p$. Furthermore, $\mathbb{E}[X] = 1/p$ and $\mathbb{V}[X] = (1-p)/p^2$.*

Proof: The proof of the expectation and variance is included for the sake of completeness, and the reader is of course encouraged to skip (reading) this proof. So, let $f(x) = \sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$, and observe that $f'(x) = \sum_{i=1}^{\infty} i x^{i-1} = (1-x)^{-2}$. As such, we have

$$\mathbb{E}[X] = \sum_{i=1}^{\infty} i\,(1-p)^{i-1}p = pf'(1-p) = \frac{p}{(1-(1-p))^2} = \frac{1}{p},$$

$$\text{and } \mathbb{V}[X] = \mathbb{E}\left[X^2\right] - \frac{1}{p^2} = \sum_{i=1}^{\infty} i^2\,(1-p)^{i-1}p - \frac{1}{p^2}. = p + p(1-p)\sum_{i=2}^{\infty} i^2\,(1-p)^{i-2} - \frac{1}{p^2}.$$

Observe that

$$f''(x) = \sum_{i=2}^{\infty} i(i-1)x^{i-2} = \left((1-x)^{-1}\right)'' = \frac{2}{(1-x)^3}.$$

As such, we have that

$$\begin{aligned}\Delta(x) &= \sum_{i=2}^{\infty} i^2 x^{i-2} = \sum_{i=2}^{\infty} i(i-1)x^{i-2} + \sum_{i=2}^{\infty} i x^{i-2} = f''(x) + \frac{1}{x}\sum_{i=2}^{\infty} i x^{i-1} = f''(x) + \frac{1}{x}(f'(x) - 1) \\ &= \frac{2}{(1-x)^3} + \frac{1}{x}\left(\frac{1}{(1-x)^2} - 1\right) = \frac{2}{(1-x)^3} + \frac{1}{x}\left(\frac{1-(1-x)^2}{(1-x)^2}\right) = \frac{2}{(1-x)^3} + \frac{1}{x}\cdot\frac{x(2-x)}{(1-x)^2} \\ &= \frac{2}{(1-x)^3} + \frac{2-x}{(1-x)^2}.\end{aligned}$$

As such, we have that

$$\begin{aligned}\mathbb{V}[X] &= p + p(1-p)\Delta(1-p) - \frac{1}{p^2} = p + p(1-p)\left(\frac{2}{p^3} + \frac{1+p}{p^2}\right) - \frac{1}{p^2} = p + \frac{2(1-p)}{p^2} + \frac{1-p^2}{p} - \frac{1}{p^2} \\ &= \frac{p^3 + 2(1-p) + p - p^3 - 1}{p^2} = \frac{1-p}{p^2}.\end{aligned}$$

■

2.3. Application of expectation: Approximating 3SAT

Let F be a boolean formula with n variables in CNF form, with m clauses, where each clause has exactly k literals. We claim that a random assignment for F, where value 0 or 1 is picked with probability $1/2$, satisfies in expectation $(1 - 2^{-k})m$ of the clauses.

We remind the reader that an instance of 3SAT is a boolean formula, for example $F = (x_1 + x_2 + x_3)(x_4 + \overline{x_1} + x_2)$, and the decision problem is to decide if the formula has a satisfiable assignment. Interestingly, we can turn this into an optimization problem.

Max 3SAT

Instance: A collection of clauses: $C_1, \ldots, C_m$.
Question: Find the assignment to $x_1, ..., x_n$ that satisfies the maximum number of clauses.

Clearly, since 3SAT is NP-Complete it implies that Max 3SAT is NP-Hard. In particular, the formula F becomes the following set of two clauses:

$$x_1 + x_2 + x_3 \quad \text{and} \quad x_4 + \overline{x_1} + x_2.$$

Note, that Max 3SAT is a ***maximization problem***.

Definition 2.3.1. Algorithm **Alg** for a maximization problem achieves an approximation factor α if for all inputs, we have:

$$\frac{\mathbf{Alg}(G)}{\mathrm{Opt}(G)} \geq \alpha.$$

In the following, we present a ***randomized algorithm*** – it is allowed to consult with a source of random numbers in making decisions. A key property we need about random variables, is the linearity of expectation property defined above.

Definition 2.3.2. For an event $\mathcal{E}$, let X be a random variable which is 1 if $\mathcal{E}$ occurred, and 0 otherwise. The random variable X is an ***indicator variable***.

Observation 2.3.3. *For an indicator variable X of an event $\mathcal{E}$, we have*

$$\mathbb{E}[X] = 0 \cdot \mathbb{P}[X = 0] + 1 \cdot \mathbb{P}[X = 1] = \mathbb{P}[X = 1] = \mathbb{P}[\mathcal{E}].$$

Theorem 2.3.4. *One can achieve (in expectation) $(7/8)$-approximation to Max 3SAT in polynomial time. Specifically, consider a 3SAT formula F with n variables and m clauses, and consider the randomized algorithm that assigns each variable value 0 or 1 with equal probability (independently to each variable) . Then this assignment satisfies $(7/8)m$ clauses in* expectation.

Proof: Let $x_1, \ldots, x_n$ be the n variables used in the given instance. The algorithm works by randomly assigning values to $x_1, \ldots, x_n$, independently, and equal probability, to 0 or 1, for each one of the variables.

Let Y_i be the indicator variables which is 1 if (and only if) the ith clause is satisfied by the generated random assignment, and 0 otherwise, for $i = 1, \ldots, m$. Formally, we have

$$Y_i = \begin{cases} 1 & C_i \text{ is satisfied by the generated assignment,} \\ 0 & \text{otherwise.} \end{cases}$$

Now, the number of clauses satisfied by the given assignment is $Y = \sum_{i=1}^{m} Y_i$. We claim that $\mathbb{E}[Y] = (7/8)m$, where m is the number of clauses in the input. Indeed, we have

$$\mathbb{E}[Y] = \mathbb{E}\Big[\sum_{i=1}^{m} Y_i\Big] = \sum_{i=1}^{m} \mathbb{E}[Y_i],$$

by linearity of expectation. The probability that $Y_i = 0$ is exactly the probability that all three literals appearing in the clause C_i are evaluated to FALSE. Since the three literals, Say ℓ_1, ℓ_2, ℓ_3, are instance of three distinct variable these three events are independent, and as such the probability for this happening is

$$\mathbb{P}[Y_i = 0] = \mathbb{P}[(\ell_1 = 0) \wedge (\ell_2 = 0) \wedge (\ell_3 = 0)] = \mathbb{P}[\ell_1 = 0]\,\mathbb{P}[\ell_2 = 0]\,\mathbb{P}[\ell_3 = 0] = \frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{1}{8}.$$

(Another way to see this, is to observe that since C_i has exactly three literals, there is only one possible assignment to the three variables appearing in it, such that the clause evaluates to FALSE. Now, there are eight (8) possible assignments to this clause, and thus the probability of picking a FALSE assignment is $1/8$.) Thus,

$$\mathbb{P}[Y_i = 1] = 1 - \mathbb{P}[Y_i = 0] = \frac{7}{8},$$

and

$$\mathbb{E}[Y_i] = \mathbb{P}[Y_i = 0] * 0 + \mathbb{P}[Y_i = 1] * 1 = \frac{7}{8}.$$

Namely, $\mathbb{E}[\#\text{ of clauses sat}] = \mathbb{E}[Y] = \sum_{i=1}^{m} \mathbb{E}[Y_i] = (7/8)m$. Since the optimal solution satisfies at most m clauses, the claim follows. ■

Curiously, Theorem 2.3.4 is stronger than what one usually would be able to get for an approximation algorithm. Here, the approximation quality is independent of how well the optimal solution does (the optimal can satisfy at most m clauses, as such we get a $(7/8)$-approximation. Curiouser and curiouser[①], the algorithm does not even look on the input when generating the random assignment.

Håstad [Hås01a] proved that one can do no better; that is, for any constant $\varepsilon > 0$, one can not approximate 3SAT in polynomial time (unless P = NP) to within a factor of $7/8 + \varepsilon$. It is pretty amazing that a trivial algorithm like the above is essentially optimal.

Remark 2.3.5. For $k \geq 3$, the above implies $(1 - 2^{-k})$-approximation algorithm, for k-SAT, as long as the instances are each of length at least k.

2.4. Markov's inequality

2.4.1. Markov's inequality

We remind the reader that for a random variable X assuming real values, its expectation is $\mathbb{E}[Y] = \sum_y y \cdot \mathbb{P}[Y = y]$. Similarly, for a function $f(\cdot)$, we have $\mathbb{E}[f(Y)] = \sum_y f(y) \cdot \mathbb{P}[Y = y]$.

Theorem 2.4.1 (Markov's Inequality). *Let Y be a random variable assuming only non-negative values. Then for all $t > 0$, we have*

$$\mathbb{P}[Y \geq t] \leq \frac{\mathbb{E}[Y]}{t}.$$

[①] "Curiouser and curiouser!" Cried Alice (she was so much surprised, that for the moment she quite forgot how to speak good English). – Alice in wonderland, Lewis Carol

Proof: Indeed,

$$\begin{aligned}\mathbb{E}[Y] &= \sum_{y \geq t} y\,\mathbb{P}[Y = y] + \sum_{y<t} y\,\mathbb{P}[Y = y] \geq \sum_{y \geq t} y\,\mathbb{P}[Y = y] \\ &\geq \sum_{y \geq t} t\,\mathbb{P}[Y = y] = t\,\mathbb{P}[Y \geq t].\end{aligned}$$

■

Markov inequality is tight, as the following exercise testifies.

Exercise 2.4.2. For any (integer) $k > 1$, define a random positive variable X_k such that $\mathbb{P}[X_k \geq k\,\mathbb{E}[X_k]] = 1/k$.

2.4.2. Example: A good approximation to kSAT with good probability

In Section 2.3 we saw a surprisingly simple algorithm that, for a formula F that is 3SAT with n variables and m clauses, in expectation (in linear time) it finds an assignment that satisfies $(7/8)m$ of the clauses (for simplicity, here we set $k = 3$).

The problem is that the guarantee is only in expectation – and the assignment being output by the algorithm might satisfy much fewer clauses. Namely, we would like to convert a guarantee that is in expectation into, a good probability guarantee. So, let $\varepsilon, \varphi \in (0, 1/2)$ be two parameters. We would like an algorithm that outputs an assignment that satisfies (say) $(1 - \varepsilon)(7/8)m$ clauses, with probability $\geq 1 - \varphi$.

To this end, the new algorithm runs the previous algorithm

$$u = \left\lceil \frac{1}{\varepsilon} \ln \frac{1}{\varphi} \right\rceil$$

times, and returns the assignment satisfying the largest number of clauses.

Lemma 2.4.3. *Given a 3SAT formula with n variables and m clauses, and parameters $\varepsilon, \varphi \in (0, 1/2)$, the above algorithm returns an assignment that satisfies $\geq (1 - \varepsilon)(7/8)m$ clauses of F, with probability $\geq 1 - \varphi$. The running time of the algorithm is $O(\varepsilon^{-1}(n + m) \log \varphi^{-1})$.*

Proof: Let Z_i be the number of clauses *not* satisfied by the ith random assignment considered by the algorithm. Observe that $\mathbb{E}[Z_i] = m/8$,as the probability of a clause not to be satisfied is $1/2^3$. The ith iteration *fails* if

$$m - Z_i < (1 - \varepsilon)(7/8)m \quad \Longrightarrow \quad Z_i > m(1 - (1 - \varepsilon)7/8) = (1 + 7\varepsilon)\frac{m}{8} = (1 + 7\varepsilon)\,\mathbb{E}[Z_i]\,.$$

Thus, by Markov's inequality, the ith iteration fails with probability

$$p = \mathbb{P}[m - Z_i < (1 - \varepsilon)(7/8)m] = \mathbb{P}[Z_i > (1 + 7\varepsilon)\,\mathbb{E}[Z_i]] < \frac{\mathbb{E}[Z_i]}{(1 + 7\varepsilon)\,\mathbb{E}[Z_i]} = \frac{1}{1 + 7\varepsilon} < 1 - \varepsilon,$$

since $(1 + 7\varepsilon)(1 - \varepsilon) = 1 + 6\varepsilon - 7\varepsilon^2 > 1$, for $\varepsilon < 1/2$.

For the algorithm to fail, all u iterations must fail. Since $1 - x \leq \exp(-x)$, we have that

$$p^u \leq (1 - \varepsilon)^u \leq \exp(-\varepsilon)^u \leq \exp(-\varepsilon u) \leq \exp\left(-\varepsilon \left\lceil \frac{1}{\varepsilon} \ln \frac{1}{\varphi} \right\rceil\right) \leq \varphi.$$

■

2.4.3. Example: Coloring a graph

Consider a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with n vertices and m edges. We would like to color it with k colors. As a reminder, a ***coloring*** of a graph by k colors, is an assignment $\chi : \mathsf{V} \to [\![k]\!]$ of a color to each vertex of G, out of the k possible colors $[\![k]\!] = \{1, 2, \ldots, k\}$. A coloring of an edge $uv \in \mathsf{E}$ is *valid* if $\chi(u) \neq \chi(v)$.

Lemma 2.4.4. *Consider a random coloring χ of the vertices of a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, where each vertex is assigned a color randomly and uniformly from $[\![k]\!]$. Then, the expected number of edges with invalid coloring is m/k, where $m = |\mathsf{E}(\mathsf{G})|$ is the number of edges in G.*

Proof: Let $\mathsf{E} = \{e_1, \ldots, e_m\}$. Let X_i be an indicator variable that is 1 $\iff$ e_i is invalid colored by χ. Let $e_i = u_i v_i$. We have that

$$\mathbb{P}[X_i = 1] = \mathbb{P}[\chi(u_i) = \chi(v_i)] = \frac{1}{k}.$$

Indeed, conceptually color u_i first, and v_i later. The probability that v_i would be assigned the same color as u_i is $1/k$. Let Z be the random variable that is the number of edges that are invalid for χ. We have that $Z = \sum_i X_i$. By linearity of expectations, and the expectation of an indicator variable, we have

$$\mathbb{E}[Z] = \sum_{i=1}^{m} \mathbb{E}[X_i] = \sum_{i=1}^{m} \mathbb{P}[X_i = 1] = \sum_{i=1}^{m} \frac{1}{k} = \frac{m}{k}.$$

■

That is pretty good, but what about an algorithm that always succeeds? The above algorithm might always somehow gives us a bad coloring. Well, not to worry.

Lemma 2.4.5. *The above random coloring of G with k colors, has at most $2m/k$ invalid edges, with probability $\geq 1/2$.*

Proof: We have that $\mathbb{E}[Z] = m/k$. As such, by Markov's inequality, we have that

$$\mathbb{P}[Z > 2m/k] \leq \mathbb{P}[Z \geq 2m/k] \leq \frac{\mathbb{E}[Z]}{2m/k} = \frac{m/k}{2m/k} = \frac{1}{2}.$$

Thus

$$\mathbb{P}[Z \leq 2m/k] = 1 - \mathbb{P}[Z > 2m/k] \geq 1 - \frac{1}{2} = \frac{1}{2}.$$

■

In particular, consider he modified algorithm – it randomly colors the graph G. If there are at most $2m/k$ invalid edges, it output the coloring, and stops. Otherwise, it retries. The probability of every iteration to succeeds is $p \geq 1/2$, and as such, the number of iterations behaves like a geometric random variable. It follows, that in expectation, the number of iterations is at most $1/p \leq 2$. Thus, the expected running time of this algorithm is $O(m)$. Indeed, let R be the number iterations performed by the algorithm. We have that the expected running time is proportional to

$$\mathbb{E}[Rm] = m\,\mathbb{E}[R] = 2m.$$

Note, that this is not the full picture – $\mathbb{P}[R = i] \leq 1/2^{i-1}$. So the probability of this algorithm tor for long decreases quickly.

2.4.3.1. Getting a valid coloring

2.4.3.1.1. A fun algorithm. A natural approach is to run the above algorithm for $k = \sqrt{m}$ (assume it is an integer). Then identify all the invalid edges, and invalidate the color of all the vertices involved. We now repeat the coloring algorithm on these invalid vertices and invalid edges, again using random coloring, but now using colors $\{k+1, \ldots, 2k\}$. If after this, there is a single invalid edge, we color one of its vertices by the color $2k+1$, and output this coloring. Otherwise, it fails.

Lemma 2.4.6. *The above algorithm succeeds with probability at least* $1/2$.

Proof: Let Y be the number of invalid edges in the end of the second round. For an edge to be invalid, its coloring must have failed in both rounds, and the probability for that is exactly $(1/k) \cdot (1/k) = 1/m$ since the two events are independent. As such, arguing as above, we have $\mathbb{E}[Y] = 1$. By Markov's inequality, we have that

$$\mathbb{P}[\text{algorithm fails}] = \mathbb{P}[Y > 1] = \mathbb{P}[Y \geq 2] \leq \frac{\mathbb{E}[Y]}{2} = \frac{1}{2}.$$ ■

Remark 2.4.7. This is a toy example - it is not hard to come up with a deterministic algorithm that uses (say) $\sqrt{2m} + 2$ colors (how? think about it). However, this algorithm is a nice distributed algorithm - after three rounds of communications, it colors the graph in a valid way, with probability at least half.

References

[Hås01a] J. Håstad. Some optimal inapproximability results. *J. Assoc. Comput. Mach.*, 48(4): 798–859, 2001.

Chapter 3

Analyzing QuickSort and QuickSelect via Expectation

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> NOBODY expects the Spanish Inquisition! Our chief weapon is surprise...surprise and fear...fear and surprise.... Our two weapons are fear and surprise...and ruthless efficiency.... Our *three* weapons are fear, surprise, and ruthless efficiency...and an almost fanatical devotion to the Pope.... Our *four*...no... *Amongst* our weapons.... Amongst our weaponry...are such elements as fear, surprise....
>
> The Spanish Inquisition, Monty Python

3.1. QuickSort

Let the input be a set $T = \{t_1, \ldots, t_n\}$ of n items to be sorted. We remind the reader, that the **QuickSort** algorithm randomly pick a pivot element (uniformly), splits the input into two subarrays of all the elements smaller than the pivot, and all the elements larger than the pivot, and then it recurses on these two subarrays (the pivot is not included in these two subproblems). Here we will show that the expected running time of **QuickSort** is $O(n \log n)$.

Let $S_1, \ldots, S_n$ be the elements in their sorted order (i.e., the output order). Let $X_{ij} = 1$ be the indicator variable which is one $\iff$ **QuickSort** compares S_i to S_j, and let p_{ij} denote the probability that this happens. Clearly, the number of comparisons performed by the algorithm is $C = \sum_{i<j} X_{ij}$. By linearity of expectations, we have

$$\mathbb{E}[C] = \mathbb{E}\Big[\sum_{i<j} X_{ij}\Big] = \sum_{i<j} \mathbb{E}\Big[X_{ij}\Big] = \sum_{i<j} p_{ij}.$$

We want to bound p_{ij}, the probability that the S_i is compared to S_j. Consider the last recursive call involving both S_i and S_j. Clearly, the pivot at this step must be one of $S_i, \ldots, S_j$, all equally likely. Indeed, S_i and S_j were separated in the next recursive call.

Observe, that S_i and S_j get compared if and only if pivot is S_i or S_j. Thus, the probability for that is $2/(j - i + 1)$. Indeed,

$$p_{ij} = \mathbb{P}\Big[S_i \text{ or } S_j \text{ picked} \;\Big|\; \text{picked pivot from } S_i, \ldots, S_j\Big] = \frac{2}{j - i + 1}.$$

Thus,

$$\sum_{i=1}^{n}\sum_{j>i} p_{ij} = \sum_{i=1}^{n}\sum_{j>i} 2/(j-i+1) = \sum_{i=1}^{n}\sum_{k=1}^{n-i+1}\frac{2}{k} \leq 2\sum_{i=1}^{n}\sum_{k=1}^{n}\frac{1}{k} \leq 2nH_n \leq n + 2n\ln n,$$

where H_n is the ***harmonic number***[1] $H_n = \sum_{i=1}^{n} 1/i$. We thus proved the following result.

Lemma 3.1.1. **QuickSort** *performs in expectation at most* $n + 2n\ln n$ *comparisons, when sorting* n *elements.*

Note, that this holds for all inputs. No assumption on the input is made. Similar bounds holds not only in expectation, but also with high probability.

This raises the question, of how does the algorithm pick a random element? We assume we have access to a random source that can get us number between 1 and n uniformly.

Note, that the algorithm always works, but it might take quadratic time in the worst case.

Remark 3.1.2 (Wait, wait, wait). Let us do the key argument in the above more slowly, and more carefully. Imagine, that before running **QuickSort** we choose for every element a random priority, which is a real number in the range $[0, 1]$. Now, we re-implement **QuickSort** such that it always pick the element with the lowest random priority (in the given subproblem) to be the pivot. One can verify that this variant and the standard implementation have the same running time. Now, a_i gets compares to a_j if and only if all the elements $a_{i+1}, \ldots, a_{j-1}$ have random priority larger than both the random priority of a_i and the random priority of a_j. But the probability that one of two elements would have the lowest random-priority out of $j - i + 1$ elements is $2 * 1/(j - i + 1)$, as claimed.

3.2. QuickSelect: Median selection in linear time

3.2.1. Analysis via expectation and indicator variables

We remind the reader that **QuickSelect** receives an array $\mathcal{T}[1 \ldots n]$ of n real numbers, and a number k, and returns the element of rank k in the sorted order of the elements of $\mathcal{T}$, see Figure 3.1. We can of course, use **QuickSort**, and just return the kth element in the sorted array, but a more efficient algorithm, would be to modify **QuickSelect**, so that it recurses on the subproblem that contains the element we are interested in. Formally, **QuickSelect** chooses a random pivot, splits the array according to the pivot. This implies that we now know the rank of the pivot, and if its equal to $\overline{\mathsf{m}}$, we return it. Otherwise, we recurse on the subproblem containing the required element (modifying $\overline{\mathsf{m}}$ as we go down the recursion. Namely, **QuickSelect** is a modification of **QuickSort** performing only a single recursive call (instead of two).

As before, to bound the expected running time, we will bound the expected number of comparisons. As before, let $S_1, \ldots, S_n$ be the elements of t in their sorted order. Now, for $i < j$, let X_{ij} be the indicator variable that is one if S_i is being compared to S_j during the execution of **QuickSelect**. There are several possibilities to consider:

(i) If $i < j < \overline{\mathsf{m}}$: Here, S_i is being compared to S_j, if and only if the first pivot in the range $S_i, \ldots, S_k$ is either S_i or S_j. The probability for that is $2/(k - i + 1)$. As such, we have that

$$\alpha_1 = \mathbb{E}\Big[\sum_{i<j<\overline{\mathsf{m}}} X_{ij}\Big] = \mathbb{E}\Big[\sum_{i=1}^{\overline{\mathsf{m}}-2}\sum_{j=i+1}^{\overline{\mathsf{m}}-1} X_{ij}\Big] = \sum_{i=1}^{\overline{\mathsf{m}}-2}\sum_{j=i+1}^{\overline{\mathsf{m}}-1}\frac{2}{\overline{\mathsf{m}}-i+1} = \sum_{i=1}^{med-2}\frac{2(\overline{\mathsf{m}}-i-1)}{\overline{\mathsf{m}}-i+1} \leq 2(\overline{\mathsf{m}}-2).$$

[1] Using integration to bound summation, we have $H_n \leq 1 + \int_{x=1}^{n}\frac{1}{x}dx \leq 1 + \ln n$. Similarly, $H_n \geq \int_{x=1}^{n}\frac{1}{x}dx = \ln n$.

(ii) If $\overline{\mathsf{m}} < i < j$: Using the same analysis as above, we have that $\mathbb{P}\Big[X_{ij} = 1\Big] = 2/(j - \overline{\mathsf{m}} + 1)$. As such,

$$\alpha_2 = \mathbb{E}\left[\sum_{j=\overline{\mathsf{m}}+1}^{n}\sum_{i=\overline{\mathsf{m}}+1}^{j-1} X_{ij}\right] = \sum_{j=\overline{\mathsf{m}}+1}^{n}\sum_{i=\overline{\mathsf{m}}+1}^{j-1}\frac{2}{j-\overline{\mathsf{m}}+1} = \sum_{j=\overline{\mathsf{m}}+1}^{n}\frac{2(j-\overline{\mathsf{m}}-1)}{j-\overline{\mathsf{m}}+1} \leq 2\Big(n-\overline{\mathsf{m}}\Big).$$

(iii) $i < \overline{\mathsf{m}} < j$: Here, we compare S_i to S_j if and only if the first indicator in the range $S_i, \ldots, S_j$ is either S_i or S_j. As such, $\mathbb{E}\Big[X_{ij}\Big] = \mathbb{P}\Big[X_{ij} = 1\Big] = 2/(j - i + 1)$. As such, we have

$$\alpha_3 = \mathbb{E}\left[\sum_{i=1}^{\overline{\mathsf{m}}-1}\sum_{j=\overline{\mathsf{m}}+1}^{n} X_{ij}\right] = \sum_{i=1}^{\overline{\mathsf{m}}-1}\sum_{j=\overline{\mathsf{m}}+1}^{n}\frac{2}{j-i+1}.$$

Observe, that for a fixed $\Delta = j - i + 1$, we are going to handle the gap Δ in the above summation, at most $\Delta - 2$ times. As such, $\alpha_3 \leq \sum_{\Delta=3}^{n} 2(\Delta-2)/\Delta \leq 2n$.

(iv) $i = \overline{\mathsf{m}}$. We have $\alpha_4 = \sum_{j=\overline{\mathsf{m}}+1}^{n}\mathbb{E}\Big[X_{ij}\Big] = \sum_{j=\overline{\mathsf{m}}+1}^{n}\frac{2}{j-\overline{\mathsf{m}}+1} = \ln n + 1.$

(v) $j = \overline{\mathsf{m}}$. We have $\alpha_5 = \sum_{i=1}^{\overline{\mathsf{m}}-1}\mathbb{E}\Big[X_{ij}\Big] = \sum_{i=1}^{\overline{\mathsf{m}}-1}\frac{2}{\overline{\mathsf{m}}-i+1} \leq \ln\overline{\mathsf{m}} + 1.$

Thus, the expected number of comparisons performed by **QuickSelect** is bounded by

$$\sum_i \alpha_i \leq 2(\overline{\mathsf{m}}-2) + 2(n-\overline{\mathsf{m}}) + 2n + \ln n + 1 + \ln\overline{\mathsf{m}} = 4n - 2 + \ln n + \ln\overline{\mathsf{m}}.$$

Theorem 3.2.1. *In expectation,* **QuickSelect** *performs at most* $4n - 2 + \ln n + \ln\overline{\mathsf{m}}$ *comparisons, when selecting the* $\overline{\mathsf{m}}$*th element out of* n *elements.*

A different approach can reduce the number of comparisons (in expectation) to $1.5n + o(n)$. More on that later in the course.

3.2.2. Analysis of QuickSelect via conditional expectations

Consider the problem of given a set X of n numbers, and a parameter k, to output the kth smallest number (which is the number with ***rank*** k in X). This can be easily be done by modifying **QuickSort** only to perform one recursive call. See Figure 3.1 for a pseud-code of the resulting algorithm.

Intuitively, at each iteration of **QuickSelect** the input size shrinks by a constant factor, leading to a linear time algorithm.

Theorem 3.2.2. *Given a set* X *of* n *numbers, and any integer* k*, the expected running time of* **QuickSelect**(X, n) *is* $O(n)$*.*

Proof: Let $X_1 = X$, and X_i be the set of numbers in the ith level of the recursion. Let y_i and r_i be the random element and its rank in X_i, respectively, in the ith iteration of the algorithm. Finally, let $n_i = |X_i|$. Observe that the probability that the pivot y_i is in the "middle" of its subproblem is

$$\alpha = \mathbb{P}\Big[\frac{n_i}{4} \leq r_i \leq \frac{3}{4}n_i\Big] \geq \frac{1}{2},$$

```
QuickSelect(𝒯 ⟦1 : n⟧, k)
  // Input:  𝒯 ⟦1 : n⟧ array with n numbers, parameter k.
  // Assume all numbers in t are distinct.
  // Task:  Return kth smallest number in 𝒯.
  y ← random element of 𝒯.
  r ← rank of y in 𝒯.
  if r = k then return y
  𝒯_< = array with all elements in 𝒯 < than y
  𝒯_> = all elements in 𝒯 > than y
  // By assumption |𝒯_<| + |𝒯_>| + 1 = |𝒯|.
  if r < k then
    return QuickSelect( 𝒯_>, k − r )
  else
    return QuickSelect( 𝒯_<, k )
```

Figure 3.1: **QuickSelect** pseudo-code.

and if this happens then

$$n_{i+1} \leq \max(r_i - 1, n_i - r_i) \leq \frac{3}{4} n_i.$$

We conclude that

$$\begin{aligned}\mathbb{E}[n_{i+1} \mid n_i] &\leq \mathbb{P}[y_i \text{ in the middle}]\frac{3}{4}n_i + \mathbb{P}[y_i \text{ not in the middle}]n_i \\ &\leq \alpha\frac{3}{4}n_i + (1-\alpha)n_i = n_i(1-\alpha/4) \leq n_i(1-(1/2)/4) = (7/8)n_i.\end{aligned}$$

Now, we have that

$$\begin{aligned}m_{i+1} &= \mathbb{E}[n_{i+1}] = \mathbb{E}[\mathbb{E}[n_{i+1} \mid n_i]] \leq \mathbb{E}[(7/8)n_i] = (7/8)\,\mathbb{E}[n_i] = (7/8)m_i \\ &= (7/8)^i m_0 = (7/8)^i n,\end{aligned}$$

since for any two random variables we have that $\mathbb{E}[X] = \mathbb{E}\Big[\mathbb{E}\Big[X \;\Big|\; Y\Big]\Big]$. In particular, the expected running time of **QuickSelect** is proportional to

$$\mathbb{E}\left[\sum_i n_i\right] = \sum_i \mathbb{E}[n_i] \leq \sum_i m_i = \sum_i (7/8)^i n = O(n),$$

as desired. ■

Chapter 4

Chebychev, Sampling and Selection

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

During a native rebellion in German East Africa, the Imperial Ministry in Berlin issued the following order to its representatives on the ground: The natives are to be instructed that on pain of harsh penalties, every rebellion must be announced, in writing, six weeks before it breaks out.

Dead Funny: Humor in Hitler's Germany, Rudolph Herzog

4.1. Chebyshev's inequality

4.1.1. Example: A better inequality via moments

Let $X_i \in \{-1, +1\}$ with probability half for each value, for $i = 1, \ldots, n$ (all picked independently). Let $Y = \sum_i X_i$. We have that

$$\mathbb{E}[Y] = \mathbb{E}\Big[\sum_i X_i\Big] = \sum_i \mathbb{E}[X_i] = n \cdot 0 = 0.$$

A more interesting quantity is

$$\begin{aligned}\mathbb{E}\Big[Y^2\Big] &= \mathbb{E}\Big[\Big(\sum_i X_i\Big)^2\Big] = \mathbb{E}\Big[\sum_i X_i^2 + 2\sum_{i<j} X_i X_j\Big] = \sum_i \mathbb{E}\Big[X_i^2\Big] + 2\,\mathbb{E}\Big[\sum_{i<j} X_i X_j\Big] = n + 2\sum_{i<j} \mathbb{E}\Big[X_i X_j\Big] \\ &= n + 2\sum_{i<j} \mathbb{E}[X_i]\,\mathbb{E}\Big[X_j\Big] = n.\end{aligned}$$

Lemma 4.1.1. *Let $X_i \in \{-1, +1\}$ with probability half for each value, for $i = 1, \ldots, n$ (all picked independently). We have that $\mathbb{P}\Big[|\sum_i X_i| > t\sqrt{n}\Big] < 1/t^2$.*

Proof: Let $Y = \sum_i X_i$ and $Z = Y^2$. We have

$$\mathbb{P}\Big[\Big|\sum_i X_i\Big| > t\sqrt{n}\Big] = \mathbb{P}\Big[\Big(\sum_i X_i\Big)^2 > t^2 n\Big] = \mathbb{P}\Big[Y^2 > t^2\,\mathbb{E}\Big[Y^2\Big]\Big] = \mathbb{P}\Big[Z > t^2\,\mathbb{E}[Z]\Big] \le 1/t^2,$$

by Markov's inequality. ■

4.1.2. Chebychev's inequality

As a reminder, the variance of a random variable X is $\mathbb{V}[X] = \mathbb{E}\Big[(X - \mu_X)^2\Big] = \mathbb{E}\Big[X^2\Big] - \mu_X^2$.

Theorem 4.1.2 (Chebyshev's inequality). *Let X be a real random variable, with $\mu_X = \mathbb{E}[X]$, and $\sigma_X = \sqrt{\mathbb{V}[X]}$. Then, for any $t > 0$, we have $\mathbb{P}[|X - \mu_X| \geq t\sigma_X] \leq 1/t^2$.*

Proof: Set $Y = (X - \mu_X)^2$, and observe that

$$\sigma_X^2 = \mathbb{V}[X] = \mathbb{E}[Y] = \mathbb{E}\Big[(X - \mu_X)^2\Big] = \mathbb{E}\Big[X^2\Big] - \mu_X^2.$$

As such, we have that

$$\mathbb{P}[|X - \mu_X| \geq t\sigma_X] = \mathbb{P}\Big[\, (X - \mu_X)^2 \geq t^2\sigma_X^2\Big] = \mathbb{P}\Big[Y \geq t^2\, \mathbb{E}[Y]\Big] \leq \frac{1}{t^2},$$

by Markov's inequality. ■

4.2. Estimation via sampling

One of the big advantages of randomized algorithms, is that they sample the world; that is, learn how the input looks like without reading all the input. For example, consider the following problem: We are given a set of U of n objects $u_1, \ldots, u_n$. and we want to compute the number of elements of U that have some property. Assume, that one can check if this property holds, in constant time, for a single object, and let $\psi(u)$ be the function that returns 1 if the property holds for the element u. and zero otherwise. Now, let Γ be the number of objects in U that have this property. We want to reliably estimate Γ without computing the property for all the elements of U.

A natural approach, would be to pick a random sample R of m objects, $r_1, \ldots, r_m$ from U (with replacement), and compute $Y = \sum_{i=1}^{m} \psi(r_1)$. The estimate for Γ is $\beta = (n/m)Y$. It is natural to ask how far is β from the true value Γ.

Lemma 4.2.1. *Let U be a set of n elements, with Γ of them having a certain property ψ. Let R be a uniform random sample from U (with repetition) of size m, and let Y be the number of elements in R that have the property ψ, and let $Z = (n/m)Y$ be the estimate for Γ. Then, for any $t \geq 1$, we have that*

$$\mathbb{P}\Big[\Gamma - t\frac{n}{2\sqrt{m}} \leq Z \leq \Gamma + t\frac{n}{2\sqrt{m}}\Big] \geq 1 - \frac{1}{t^2}.$$

Similarly, we have that $\mathbb{P}\Big[\mathbb{E}[Y] - t\sqrt{m}/2 \leq Y \leq \mathbb{E}[Y] + t\sqrt{m}/2\Big] \geq 1 - 1/t^2$.

Proof: Let $Y_i = \psi(r_i)$ be an indicator variable that is 1 if the ith sample r_i has the property ψ, for $i = 1, \ldots, m$. Observe that

$$p = \mathbb{E}[Y_i] = \frac{\Gamma}{n}.$$

Consider the random variable $Y = \sum_i Y_i$.

Variance of a binomial distribution. (I am including the following here as a way to remember this formula.) The variable Y is a binomial distribution with probability $p = \Gamma/n$, and m samples; that is, $Y \sim \mathrm{Bin}(m, p)$. Thus, Y is the sum of m random variables $Y_1, \ldots, Y_m$ that are independent indicator variables (i.e., Bernoulli distribution), with $\mathbb{E}[Y_i] = p$, and $\mathbb{V}[Y_i] = \mathbb{E}\Big[Y_i^2\Big] - \mathbb{E}[Y_i]^2 = p - p^2 = p(1-p)$. Since the variance is additive for independent variables, we have $\mathbb{V}[Y] = \mathbb{V}\Big[\sum_i Y_i\Big] = \sum_{i=1}^{m} \mathbb{V}[T_i] = mp(1-p)$.

Thus, we have

$$\mathbb{E}[Y] = mp = m \cdot \frac{\Gamma}{n} = \frac{m}{n}\Gamma, \qquad \text{and} \qquad \mathbb{V}[Y] = mp(1-p).$$

The standard deviation of Y is

$$\sigma_Y = \sqrt{mp(1-p)} \leq \sqrt{m}/2,$$

as $\sqrt{p(1-p)}$ is maximized for $p = 1/2$.

Consider the estimate $Z = (n/m)Y$ for Γ, and observe that

$$\mathbb{E}[Z] = \mathbb{E}[(n/m)Y] = \frac{n}{m}\mathbb{E}[Y] = \frac{n}{m}\frac{m}{n}\Gamma = \Gamma.$$

By Chebychev's inequality, we have that $\mathbb{P}[|Y - \mathbb{E}[Y]| \geq t\sigma_Y] \leq 1/t^2$. Since $(n/m)\,\mathbb{E}[Y] = \mathbb{E}[Z] = \Gamma$, this implies that

$$\begin{aligned}\mathbb{P}\left[|Z-\Gamma| \geq t\frac{n}{2\sqrt{m}}\right] &= \mathbb{P}\left[|Z-\Gamma| \geq \frac{n}{m}t\cdot\frac{\sqrt{m}}{2}\right] \leq \mathbb{P}\left[|Z-\Gamma| \geq \frac{n}{m}t\sigma_Y\right] = \mathbb{P}\left[\left|\frac{n}{m}Y - \frac{n}{m}\mathbb{E}[Y]\right| \geq \frac{n}{m}t\sigma_Y\right] \\ &= \mathbb{P}[|Y-\mathbb{E}[Y]| \geq t\sigma_Y] \leq \frac{1}{t^2}\end{aligned}$$

■

4.3. Randomized selection – Using sampling to learn the world

4.3.1. Inverse estimation

We are given a set $U = \{u_1, \ldots, u_n\}$ of n distinct numbers. Let $U_{\langle i\rangle}$ denote the ith smallest number in U – that is $U_{\langle i\rangle}$ is the number of ***rank*** i in U.

Lemma 4.3.1. *Given a set U of n numbers, a number k, and parameters $t \geq 1$ and $m \geq 1$, one can compute, in $O(m \log m)$ time, two numbers $r_-, r_+ \in U$, such that:*

(A) The number of rank k in U is in the interval $\mathcal{I} = [r_-, r_+]$.

(B) There are at most $8tn/\sqrt{m}$ numbers of U in $\mathcal{I}$.

The above two properties hold with probability $\geq 1 - 3/t^2$.

(Namely, as t increases, the interval $\mathcal{I}$ becomes bigger, and the probability it contains the desired element increases.)

Proof: **(A)** Compute a random sample R of U of size m in $O(m)$ time (assuming the input numbers are given in an array, say). Next sort the numbers of R in $O(m \log m)$ time. Let

$$\ell_- = \left\lfloor m\frac{k}{n} - t\sqrt{m}/2 \right\rfloor - 1 \qquad \text{and} \qquad \ell_+ = \left\lceil m\frac{k}{n} + t\sqrt{m}/2 \right\rceil + 1.$$

Set $r_- = \mathsf{R}[\ell_-]$ and $r_+ = \mathsf{R}[\ell_+]$.

Let Y be the number of elements in the sample R that are $\leq U_{\langle k\rangle}$. By Lemma 4.2.1, we have

$$\mathbb{P}\left[\mathbb{E}[Y] - t\sqrt{m}/2 \leq Y \leq \mathbb{E}[Y] + t\sqrt{m}/2\right] \geq 1 - 1/t^2.$$

In particular, if this happens, then $r_- \leq U_{\langle k\rangle} \leq r_+$.

(B) Let $g = k - t\frac{n}{\sqrt{m}} - 3\frac{n}{m}$, and let g_{R} be the number of elements in R that are smaller than $U_{\langle g\rangle}$. Arguing as above, we have that $\mathbb{P}\left[g_{\mathsf{R}} \leq \frac{g}{n}m + t\sqrt{m}/2\right] \geq 1 - 1/t^2$. Now

$$\frac{g}{n}m + t\sqrt{m}/2 = \frac{m}{n}\left(k - t\frac{n}{\sqrt{m}} - 3\frac{n}{m}\right) + t\sqrt{m}/2 = k\frac{m}{n} - t\sqrt{m} - 3 + t\sqrt{m}/2 = k\frac{m}{n} - t\sqrt{m}/2 - 3 < \ell_-.$$

This implies that the g smallest numbers in U are outside the interval $[r_-, r_+]$ with probability $\geq 1 - 1/t^2$.

Next, let $h = k + t\frac{n}{\sqrt{m}} + 3\frac{n}{m}$. A similar argument, shows that all the $n - h$ largest numbers in U are too large to be in $[r_-, r_+]$. This implies that

$$|[r_-, r_+] \cap U| \leq h - g + 1 = 6\frac{n}{m} + 2t\frac{n}{\sqrt{m}} \leq 8\frac{tn}{\sqrt{m}}.$$ ■

4.3.1.1. Inverse estimation – intuition

Here we are trying to give some intuition to the proof of the previous lemma. Feel free to skip this part if you feel you already understand what is going on.

Given k, we are interested in estimating $s_k = U_{\langle k \rangle}$ quickly. So, let us take a sample R of size m. Let $\mathsf{R}_{\leq s_k}$ be the set of all the numbers in R that are $\leq s_k$. For $Y = \left|\mathsf{R}_{\leq s_k}\right|$, we have that $\mu = \mathbb{E}[Y] = m\frac{k}{n}$. Furthermore, for any $t \geq 1$, Lemma 4.2.1 implies that $\mathbb{P}\left[\mu - t\sqrt{m}/2 \leq Y \leq \mu + t\sqrt{m}/2\right] \geq 1 - 1/t^2$. In particular, with probability $\geq 1 - 1/t^2$ the number $r_- = \mathsf{R}_{\langle \ell_- \rangle}$, for $\ell_- = \left\lfloor \mu - t\sqrt{m}/2 \right\rfloor - 1$, is smaller than s_k, and similarly, the number $r_+ = \mathsf{R}_{\langle \ell_+ \rangle}$ of rank $\ell_+ = \left\lceil \mu + t\sqrt{m}/2 \right\rceil + 1$ in R is larger than s_k.

One can conceptually think about the interval $\mathcal{I}(k) = [r_-, r_+]$ as a *confidence interval* – we know that $s_k \in \mathcal{I}(k)$ with probability $\geq 1 - 1/t^2$. But how heavy is this interval? Namely, how many elements are there in $\mathcal{I}(k) \cap U$?

To this end, consider the interval of ranks, *in the sample*, that might contain the kth element. By the above, this is $\mathcal{I}(k,t) = k\frac{m}{n} + \left[-t\sqrt{m}/2 - 1, t\sqrt{m}/2 + 1\right]$. In particular, consider the maximum $\nu \leq k$, such that $\mathcal{I}(\nu, t)$ and $\mathcal{I}(k,t)$ are disjoint. We have the condition that $\nu\frac{m}{n} + t\sqrt{m}/2 + 1 \leq k\frac{m}{n} - t\sqrt{m}/2 - 1 \implies \nu \leq k - t\frac{n}{\sqrt{m}} - 2\frac{n}{m}$. Let $g = k - t\frac{n}{\sqrt{m}} - 2\frac{n}{m}$ and $h = k + t\frac{n}{\sqrt{m}} + 2\frac{n}{m}$. We have that $\mathcal{I}(g,t)$, $\mathcal{I}(k,t)$ and $\mathcal{I}(h,t)$ are all disjoint with probability $\geq 1 - 3/t^2$.

To this end, let $g = k - \left\lceil 2\left(t\frac{n}{2\sqrt{m}}\right)\right\rceil$ and $h = k + \left\lceil 2\left(t\frac{n}{2\sqrt{m}}\right)\right\rceil$. It is easy to verify (using the same argumentation as above) that with probability at least $1 - 3/t^2$, the three confidence $\mathcal{I}(g), \mathcal{I}(k)$ and $\mathcal{I}(h)$ do not intersect. As such, we have $\left|\mathcal{I}(k) \cap U\right| \leq h - g \leq 4\left(t\frac{n}{2\sqrt{m}}\right)$.

4.3.2. Randomized selection

4.3.2.1. The algorithm

Given an array S of n numbers, and the rank k. The algorithm needs to compute $S_{\langle k \rangle}$. To this end, set $t = \left\lceil n^{1/8} \right\rceil$, and $m = \left\lceil n^{3/4} \right\rceil$.

Using the algorithm of Lemma 4.3.1, in $O(m \log m)$ time, we get two numbers r_- and r_+, such that $S_{\langle k \rangle} \in [r_i, r_+]$, and

$$|\underbrace{S \cap (r_i, r_+)}_{S_m}| = O(tn/\sqrt{m}) = O\left(n^{1/8} n/m^{3/8}\right) = O(n^{3/4}).$$

To this end, we break S into three sets:

(i) $S_< = \{s \in S \mid s \leq r_-\}$,
(ii) $S_m = \{s \in S \mid r_- < s < r_+\}$,
(iii) $S_> = \{s \in S \mid r_+ \leq s\}$.

This three way partition can be done using $2n$ comparisons and in linear time. We now can readily compute the rank of r_- in S (it is $|S_<|$) and the rank of r_+ in S (it is $|S_<| + |S_m| + 1$). If $r_{-\langle S \rangle} > k$ or $r_{+\langle S \rangle} < k$ then the

algorithm failed. The other possibility for failure is that S_m is too large. That is, larger than $8tn/\sqrt{m} = O(n^{3/4})$. If any of these failures happen, then we rerun this algorithm from scratch.

Otherwise, the algorithm need to compute the element of rank $k - |S_<|$ in the set S_m, and this can be done in $O(|S_m| \log |S_m|) = O(n^{3/4} \log n)$ time by using sorting.

4.3.2.2. Analysis

The correctness is easy – the algorithm clearly returns the desired element. As for running time, observe that by Lemma 4.3.1, by probability $\geq 1 - 1/n^{1/4}$, we succeeded in the first try, and then the running time is $O(n + (m \log m)) = O(n)$. More generally, the probability that the algorithm failed in the first α tries to get a good interval $[r_-, r_+]$ is at most $1/n^{\alpha/4}$.

One can slightly improve the number of comparisons performed by the algorithm using the following modifications.

Lemma 4.3.2. *Given the numbers r_-, r_+, one can compute the sets $S_<, S_m, S_>$ using in expectation (only!) $1.5n + O(n^{3/4})$ comparisons.*

Proof: We need to compute the sets $S_<, S_m, S_>$. Namely, we need to compare all the numbers of S to r_- and r_+. Since only $O(n^{3/4})$ numbers fall in S_m, almost all of the numbers are in either $S_<$ or $S_>$. If a number of is in $S_<$ (resp. $S_>$), then comparing it r_- (resp. r_+) is enough to verify that this is indeed the case. Otherwise, perform the other comparison and put the element in its proper set (in this case we had to perform two comparisons to handle the element).

So let us guess, by a coin flip, for each element of S whether they are in $S_<$ or $S_>$. If we are right, then the algorithm would require only one comparison to put them into the right set. Otherwise, it would need two comparisons. Let X_s be the random variable that is the number of comparisons used by this algorithm for an element $s \in S$. We have that if $s \in S_< \cup S_>$ then $\mathbb{E}[X_s] = 1(1/2) + 2(1/2) = 3/2$. If $s \in S_m$ then both comparisons will be performed, and thus $\mathbb{E}[X_s] = 2$ in this case.

Thus, the expected numbers of comparisons for all the elements of S, by linearity of expectations, is $\frac{3}{2}(n - |S_m|) + 2|S_m| = (3/2)n + |S_m|/2$. ■

Theorem 4.3.3. *Given an array S with n numbers and a rank k, one can compute the element of rank k in S in expected linear time. Formally, the resulting algorithm performs in expectation $1.5n + O(n^{3/4} \log n)$ comparisons.*

Proof: Let X be the random variable that is the number of iteration till the interval is good. We have that X is a geometric variable with probability of success $\geq 1 - 1/n^{1/4}$. As such, the expected number of rounds till success is $\leq 1/p \leq 1 + 2/n^{1/4}$. As such, the expected number of comparisons performed by the algorithm is $\mathbb{E}\Big[X \cdot \Big(1.5n + O(n^{3/4} \log n)\Big)\Big] = 1.5n + O(n^{3/4} \log n)$. ■

Chapter 5

Verifying Identities, and Some Complexity

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> The events of September 8 prompted Foch to draft the later legendary signal: "My centre is giving way, my right is in retreat, situation excellent. I attack." It was probably never sent.
>
> John Keegan, The first world war

5.1. Verifying equality

5.1.1. Vectors

You are given two binary vectors $\mathbf{v} = (v_1, \ldots, v_n), \mathbf{u} = (u_1, \ldots, u_n) \in \{0,1\}^n$ and you would like to decide if they are equal or not. Unfortunately, the only access you have to the two vectors is via a black-box that enables you to compute the dot-product of two binary vectors over $\mathbb{Z}_2$. Formally, given two binary vectors as above, their dot-product is $\langle \mathbf{v}, \mathbf{u} \rangle = \sum_{i=1}^{n} v_i u_i$ (which is a non-negative integer number). Their dot product modulo 2, is $\langle \mathbf{v}, \mathbf{u} \rangle \bmod 2$ (i.e., it is 1 if $\langle \mathbf{v}, \mathbf{u} \rangle$ is odd and 0 otherwise).

Naturally, we could the use the black-box to read the vectors (using $2n$ calls), but since we are interested only in deciding if they are equal or not, this should require less calls to the black-box (which is expensive).

Lemma 5.1.1. *Given two binary vectors $\mathbf{v}, \mathbf{u} \in \{0,1\}^n$, a randomized algorithm can, using two computations of dot-product modulo 2, decide if $\mathbf{v}$ is equal to $\mathbf{u}$ or not. The algorithm may return one of the following two values:*

≠: *Then $\mathbf{v} \neq \mathbf{u}$.*

=: *Then the probability that the algorithm made a mistake (i.e., the vectors are different) is at most $1/2$.*

The running time of the algorithm is $O(n + B(n))$, where $B(n)$ is the time to compute a single dot-product of vectors of length n.

Proof: Pick a random vector $\mathbf{r} = (r_1, \ldots, r_n) \in \{0,1\}^n$ by picking each coordinate independently with probability $1/2$. Compute the two dot-products $\langle \mathbf{v}, \mathbf{r} \rangle$ and $\langle \mathbf{u}, \mathbf{r} \rangle$.

(A) If $\langle \mathbf{v}, \mathbf{r} \rangle \equiv \langle \mathbf{v}, \mathbf{r} \rangle \bmod 2 \quad \Rightarrow \quad$ the algorithm returns '**=**'.

(B) If $\langle \mathbf{v}, \mathbf{r} \rangle \not\equiv \langle \mathbf{v}, \mathbf{r} \rangle \bmod 2 \quad \Rightarrow \quad$ the algorithm returns '**≠**'.

Clearly, if the '**≠**' is returned then $\mathbf{v} \neq \mathbf{u}$.

So, assume that the algorithm returned '**=**' but $\mathbf{v} \neq \mathbf{u}$. For the sake of simplicity of exposition, assume that they differ on the nth bit: $u_n \neq v_n$. We then have that

$$\alpha = \langle \mathbf{v}, \mathbf{r} \rangle = \overbrace{\sum_{i=1}^{n-1} v_i r_i}^{=\alpha'} + v_n r_n \qquad \text{and} \qquad \beta = \langle \mathbf{u}, \mathbf{r} \rangle = \overbrace{\sum_{i=1}^{n-1} u_i r_i}^{=\beta'} + u_n r_n.$$

Now, there are two possibilities:

(A) If $\alpha' \not\equiv \beta' \bmod 2$, then, with probability half, we have $r_i = 0$, and as such $\alpha \not\equiv \beta \bmod 2$.

(B) If $\alpha' \equiv \beta' \bmod 2$, then, with probability half, we have $r_i = 1$, and as such $\alpha \not\equiv \beta \bmod 2$.

As such, with probability at most half, the algorithm would fail to discover that the two vectors are different. ■

5.1.1.1. Amplification

Of course, this is not a satisfying algorithm – it returns the correct answer only with probability half if the vectors are different. So, let us run the algorithm t times. Let $T_1, \ldots, T_t$ be the returned values from all these executions. If any of the t executions returns that the vectors are different, then we know that they are different.

$$\begin{aligned}
\mathbb{P}[\text{Algorithm fails}] &= \mathbb{P}[\mathbf{v} \neq \mathbf{u}, \text{ but all } t \text{ executions return `='}] \\
&= \mathbb{P}[(T_1 = \text{`='}) \cap (T_2 = \text{`='}) \cap \cdots \cap (T_t = \text{`='})] \\
&= \mathbb{P}[T_1 = \text{`='}]\, \mathbb{P}[T_2 = \text{`='}] \cdots \mathbb{P}[T_t = \text{`='}] \leq \prod_{i=1}^{t} \frac{1}{2} = \frac{1}{2^t}.
\end{aligned}$$

We thus get the following result.

Lemma 5.1.2. *Given two binary vectors $\mathbf{v}, \mathbf{u} \in \{0,1\}^n$ and a* **confidence** *parameter $\delta > 0$, a randomized algorithm can decide if $\mathbf{v}$ is equal to $\mathbf{u}$ or not. More precisely, the algorithm may return one of the two following results:*

≠: *Then $\mathbf{v} \neq \mathbf{u}$.*

=: *Then, with probability $\geq 1 - \delta$, we have $\mathbf{v} \neq \mathbf{u}$.*

The running time of the algorithm is $O\Big((n + B(n)) \ln \delta^{-1}\Big)$, where $B(n)$ is the time to compute a single dot-product of two vectors of length n.

Proof: Follows from the above by setting $t = \lceil \lg(1/\delta) \rceil$. ■

5.1.2. Matrices

Given three binary matrices $\mathsf{B}, \mathsf{C}, \mathsf{D}$ of size $n \times n$, we are interested in deciding if $\mathsf{BC} = \mathsf{D}$. Computing BC is expensive – the fastest known (theoretical!) algorithm has running time (roughly) $O\Big(n^{2.37}\Big)$. On the other hand, multiplying such a matrix with a vector $\mathbf{r}$ (modulo 2, as usual) takes only $O(n^2)$ time (and this algorithm is simple).

Lemma 5.1.3. *Given three binary matrices $\mathsf{B}, \mathsf{C}, \mathsf{D} \in \{0,1\}^{n \times n}$ and a confidence parameter $\delta > 0$, a randomized algorithm can decide if $\mathsf{BC} = \mathsf{D}$ or not. More precisely the algorithm can return one of the following two results:*

≠: *Then $\mathsf{BC} \neq \mathsf{D}$.*

=: *Then $\mathsf{BC} = \mathsf{D}$ with probability $\geq 1 - \delta$.*

The running time of the algorithm is $O\Big(n^2 \log \delta^{-1}\Big)$.

Proof: Compute a random vector $\mathbf{r} = (r_1, \ldots, r_n)$, and compute the quantity $\mathbf{x} = \mathsf{BC}\mathbf{r} = \mathsf{B}(\mathsf{C}\mathbf{r})$ in $O(n^2)$ time, using the associative property of matrix multiplication. Similarly, compute $\mathbf{y} = \mathsf{D}\mathbf{r}$. Now, if $\mathbf{x} \neq \mathbf{y}$ then return '='.

Now, we execute this algorithm $t = \left\lceil \lg \delta^{-1} \right\rceil$ times. If all of these independent runs return that the matrices are equal then return '='.

The algorithm fails only if $\mathsf{BC} \neq \mathsf{D}$, but then, assume the ith row in two matrices BC and D are different. The probability that the algorithm would not detect that these rows are different is at most $1/2$, by Lemma 5.1.1. As such, the probability that all t runs failed is at most $1/2^t \leq \delta$, as desired. ■

5.1.3. Checking identity for polynomials

5.1.3.1. The SchwartzZippel lemma

Let $\mathbb{F}$ be a field (i.e., real numbers). Let $\mathbb{F}[x_1, \ldots, X_n]$ denote the set of polynomials over the n variables $x_1, \ldots, x_n$ over $\mathbb{F}$. Such a polynomial is a sum of monomial, where a ***monomial*** has the form $c \cdot x_1^{i_1} \cdot x_2^{i_2} \cdots x_n^{i_n}$, where $c \in \mathbb{F}$. The ***degree*** of this monomial is $\text{degree}\left(c \cdot x_1^{i_1} \cdot x_2^{i_2} \cdots x_n^{i_n}\right) = i_1 + i_2 + \cdots + i_n$. Thus, a polynomial of degree d over n variables has potentially up to dn^d monomials (the exact bound is messier, but an easy lower bound on this quantity is $\binom{n}{d}$).

For a polynomial $f \in \mathbb{F}[x_1, \ldots, X_n]$, the ***zero set*** of f, is the set $Z_f = \{(x_1, \ldots, x_n) \mid f(x_1, \ldots, x_n) = 0\}$. Intuitively, the zero set $Z_f = \mathbb{F}^n$ only if $f(x_1, \ldots, x_n) = 0$ (and then it is the ***zero*** polynomial), but otherwise (i.e., $f \neq 0$) Z_f should be much "smaller".

Specifically, a polynomial in a single variable of degree d is either zero everywhere, or has at most d roots (i.e., $|Z_f| \leq d$). This is known as the *fundamental theorem of algebra*[①]. The picture gets much messier once one deals with multi-variate polynomials, but fortunately there is a simple and elegant lemma that bounds the number of zeros if we pick the values from the right set of values.

Lemma 5.1.4 (SchwartzZippel). *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a non-zero polynomial of total degree $d \geq 0$, over a field $\mathbb{F}$. Let $S \subseteq \mathbb{F}$ be finite. Let $\mathsf{r} = (r_1, \ldots, r_n)$ be randomly and uniformly chosen from S^n. Then*

$$\mathbb{P}[f(\mathsf{r}) = 0] \leq \frac{d}{|S|}.$$

Equivalently, we have $|Z_f \cap S^n| \leq d|S|^{n-1}$.

Proof: The proof is by induction on n. For $n = 1$, by the theorem, formally known as the fundamental theorem of algebra, $|Z_f \cap S| \leq |Z_f| \leq d$. So assume the theorem holds for $n - 1$. Since f is non-zero, it can be written as a sum of d polynomials in $n - 1$ variables. That is, f can be written as

$$f(x_1, \ldots, x_n) = \sum_{i=0}^{d} x_1^i f_i(x_2, \ldots, x_n),$$

where $\text{degree}(f_i) \leq d - i$. Since f is not zero, one of the fs must be non-zero, and let i the maximum value such that $f_i \neq 0$.

Now, we randomly choose the values $r_2, \ldots, r_n \in S$ (independently and uniformly). And consider the polynomial in the single variable x, which is

$$g(x) = \sum_{j=0}^{d} f_j(r_2, \ldots, r_n) x^i.$$

Let $\mathcal{F}$ be the event that $f_i(r_2, \ldots, r_n) = 0$. Let $\mathcal{G}$ be the event that $g(x) = 0$. By induction, we have $\mathbb{P}[\mathcal{F}] \leq (d - i)/|S|$. More interestingly if $\mathcal{F}$ does not happen, then $\text{degree}(g) = i$. As such, by induction, we have that

$$\mathbb{P}[\mathcal{G} \mid \overline{\mathcal{F}}] = \mathbb{P}[g(x) = 0 \mid \overline{\mathcal{F}}] \leq \frac{i}{|S|}.$$

[①] Wikipedia notes that the proof is not algebraic, and it is definitely not fundamental to modern algebra. So maybe it should be cited as "the theorem formerly known as the fundamental theorem of algebra".

We conclude that

$$\mathbb{P}[f(\mathsf{r}) = 0] = \mathbb{P}[\mathcal{G} \cap \mathcal{F}] + \mathbb{P}[\mathcal{G} \cap \overline{\mathcal{F}}] \leq \mathbb{P}[\mathcal{F}] + \mathbb{P}[\mathcal{G} \mid \overline{\mathcal{F}}] \leq \frac{d-i}{|S|} + \frac{i}{|S|} \leq \frac{d}{|S|}.$$ ■

Remark 5.1.5. Consider the polynomial $f(x, y) = (x-1)^2 + (y-1)^2 - 1$. The zero set of this polynomial is the unit circle. So the zero set Z_f is infinite in this case. However, note that for any choice of S, the set S^2 is a grid. The Schwartz-Zippel lemma, tells us that there relatively few grid points that are in the zero set.

5.1.3.2. Applications

5.1.3.2.1. Checking if a polynomial is the zero polynomial. Let f be a polynomial of degree d, with n variables, over the reals that can be evaluated in $O(T)$ time. One can check if f zero, by picking randomly a n numbers from $S = [\![d^3]\!]$. By Lemma 5.1.4, we have that the probability of f to be zero over the chosen values is $\leq d/d^3$, which is a small number. As above, we can do amplification to get a high confidence answer.

5.1.3.2.2. Checking if two polynomials are equal. Given two polynomials f and g, one can now check if they are equal by checking if $f(r) = g(r)$, for some random input. The proof of correctness follows from the above, as one interpret the algorithm as checking if $f - g$ is the zero polynomial.

5.1.3.2.3. Verifying polynomials product. Given three polynomials f, g, and h, one can now check if $fg = h$. Again, one randomly pick a value r, and check if $f(r)g(r) = h(r)$. The proof of correctness follows from the above, as one interprets the algorithm as checking if $fg - h$ is the zero polynomial.

5.1.4. Checking if a bipartite graph has a perfect matching

Let $\mathsf{G} = (L \cup R, \mathsf{E})$ be a bipartite graph. Let $L = \{u_1, \ldots, u_n\}$ and $R = \{v_1, \ldots, v_n\}$. Consider the set of variables

$$\mathcal{V} = \left\{ x_{i,j} \;\middle|\; u_i v_j \in \mathsf{E} \right\}.$$

Let M be an $n \times n$ matrix, where $M[i, j] = 0$ if $u_i v_j \notin \mathsf{E}$, and $M[i, j] = x_{i,j}$ otherwise. Let Π be the set of all permutations of $[\![n]\!]$.

A ***perfect matching*** is a permutation $\pi : [\![n]\!] \to [\![n]\!]$, such that for all i, we have $u_i v_{\pi(i)} \in \mathsf{E}$. For such a permutation π, consider the monomial

$$f_\pi = \operatorname{sign}(\pi) \prod_{i=1}^{n} M[i, j],$$

where sign is the ***sign*** of the permutation (it is either -1 or $+1$ – for our purpose here we do not care about the exact definition of this quantity). It is either a polynomial of degree exactly n, or it is zero. Furthermore, observe that for any two different permutation $\pi, \sigma \in \Pi$, we have that if f_π and f_σ are both non-zero, then $f_\pi \neq f_\sigma$ and $f_\pi \neq -f_\sigma$.

Consider the following "crazy" polynomial over the set of variables $\mathcal{V}$:

$$\psi = \psi(\mathcal{V}) = \det(M) = \sum_{\pi \in \Pi} \operatorname{sign}(\pi) f_\pi.$$

If there is perfect matching in G, then there is a permutation π such that $f_\pi \neq 0$. But this implies that $\psi \neq 0$ (since it has a non-zero monomials, and the monomials can not cancel each other).

In the other direction, if there is no perfect matching in G, then $f_\pi = 0$ for all permutation π. This implies that $\psi = 0$. Thus, deciding if G has a perfect matching is equivalent to deciding if $\psi \neq 0$. The polynomial ψ is defined via a determinant of a matrix that variables as some of the entries (and zeros otherwise). By the above, all we need to do is to evaluate ψ over some random values. If we use exact arithmetic, we would just pick a random number in $[0, 1]$ for each variable, and evaluate ψ for these values of the variable. Namely, we filled the matrix M with values (so it is all numbers now), and we need to computes its determinant. Via Gaussian elimination, the determinant can be computed in cubic time. Thus, we can evaluate ψ in cubic time, which implies that with high probability we can check if G has a perfect matching.

If we do not want to be prisoners of the impreciseness of floating point arithmetic, then one can perform the above calculations over some finite field (usually, the field is simply working modulo a prime number).

5.2. Las Vegas and Monte Carlo algorithms

Definition 5.2.1. A ***Las Vegas algorithm*** is a randomized algorithms that *always* return the correct result. The only variant is that it's running time might change between executions.

An example for a Las Vegas algorithm is the **QuickSort** algorithm.

Definition 5.2.2. A ***Monte Carlo algorithm*** is a randomized algorithm that might output an incorrect result. However, the probability of error can be diminished by repeated executions of the algorithm.

The matrix multiplication algorithm is an example of a Monte Carlo algorithm.

5.2.1. Complexity Classes

I assume people know what are Turing machines, **NP**, **NPC**, RAM machines, uniform model, logarithmic model. **PSPACE**, and **EXP**. If you do now know what are those things, you should read about them. Some of that is covered in the randomized algorithms book, and some other stuff is covered in any basic text on complexity theory[②].

Definition 5.2.3. The class **P** consists of all languages L that have a polynomial time algorithm **Alg**, such that for any input Σ^*, we have

(A) $x \in L \Rightarrow$ **Alg**(x) accepts,
(B) $x \notin L \Rightarrow$ **Alg**(x) rejects.

Definition 5.2.4. The class **NP** consists of all languages L that have a polynomial time algorithm **Alg**, such that for any input Σ^*, we have:

(i) If $x \in L \Rightarrow$ then $\exists y \in \Sigma^*$, **Alg**(x, y) accepts, where $|y|$ (i.e. the length of y) is bounded by a polynomial in $|x|$.
(ii) If $x \notin L \Rightarrow$ then $\forall y \in \Sigma^*$**Alg**(x, y) rejects.

Definition 5.2.5. For a complexity class $\mathcal{C}$, we define the complementary class co-$\mathcal{C}$ as the set of languages whose complement is in the class $\mathcal{C}$. That is

$$\text{co}-\mathcal{C} = \{L \mid \overline{L} \in \mathcal{C}\},$$

4 where $\overline{L} = \Sigma^* \setminus L$.

[②]There is also the internet.

It is obvious that $\mathbf{P} = \text{co}-\mathbf{P}$ and $\mathbf{P} \subseteq \mathbf{NP} \cap \text{co}-\mathbf{NP}$. (It is currently unknown if $\mathbf{P} = \mathbf{NP} \cap \text{co}-\mathbf{NP}$ or whether $\mathbf{NP} = \text{co}-\mathbf{NP}$, although both statements are believed to be false.)

Definition 5.2.6. The class **RP** (for Randomized Polynomial time) consists of all languages L that have a randomized algorithm **Alg** with worst case polynomial running time such that for any input $x \in \Sigma^*$, we have

(i) If $x \in L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] \geq 1/2$.
(ii) $x \notin L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] = 0$.

An **RP** algorithm is a Monte Carlo algorithm, but this algorithm can make a mistake only if $x \in L$. As such, $\text{co}-\mathbf{RP}$ is all the languages that have a Monte Carlo algorithm that make a mistake only if $x \notin L$. A problem which is in $\mathbf{RP} \cap \text{co}-\mathbf{RP}$ has an algorithm that does not make a mistake, namely a Las Vegas algorithm.

Definition 5.2.7. The class **ZPP** (for Zero-error Probabilistic Polynomial time) is the class of languages that have a Las Vegas algorithm that runs in expected polynomial time.

Definition 5.2.8. The class **PP** (for Probabilistic Polynomial time) is the class of languages that have a randomized algorithm **Alg**, with worst case polynomial running time, such that for any input $x \in \Sigma^*$, we have

(i) If $x \in L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] > 1/2$.
(ii) If $x \notin L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] < 1/2$.

The class **PP** is not very useful. Why?

Exercise 5.2.9. Provide a **PP** algorithm for 3SAT.

Consider the mind-boggling stupid randomized algorithm that returns either yes or no with probability half. This algorithm is almost in **PP**, as it return the correct answer with probability half. An algorithm is in **PP** needs to be *slightly* better, and be correct with probability better than half. However, how much better can be made to be arbitrarily close to $1/2$. In particular, there is no way to do effective amplification with such an algorithm.

Definition 5.2.10. The class **BPP** (for Bounded-error Probabilistic Polynomial time) is the class of languages that have a randomized algorithm **Alg** with worst case polynomial running time such that for any input $x \in \Sigma^*$, we have

(i) If $x \in L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] \geq 3/4$.
(ii) If $x \notin L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] \leq 1/4$.

5.3. Bibliographical notes

Section 35.1 follows [MR95, Section 1.5].

References

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

Chapter 6

The Birthday Paradox, Occupancy and the Coupon Collector Problem

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

I built on the sand
And it tumbled down,
I built on a rock
And it tumbled down.
Now when I build, I shall begin
With the smoke from the chimney

Leopold Staff, Foundations

6.1. Some needed math

Lemma 6.1.1. *For any positive integer n, we have:*

(i) $1 + x \le e^x$ *and* $1 - x \le e^{-x}$, *for all* x.
(ii) $(1 + 1/n)^n \le e \le (1 + 1/n)^{n+1}$.
(iii) $(1 - 1/n)^n \le \frac{1}{e} \le (1 - 1/n)^{n-1}$.
(iv) $(n/e)^n \le n! \le (n + 1)^{n+1}/e^n$.
(v) For any $k \le n$, *we have:* $\left(\frac{n}{k}\right)^k \le \binom{n}{k} \le \left(\frac{ne}{k}\right)^k$.

Proof: **(i)** Let $h(x) = e^x - 1 - x$. Observe that $h'(x) = e^x - 1$, and $h''(x) = e^x > 0$, for all x. That is $h(x)$ is a convex function. It achieves its minimum at $h'(x) = 0 \implies e^x = 1$, which is true for $x = 0$. For $x = 0$, we have that $h(0) = e^0 - 1 - 0 = 0$. That is, $h(x) \ge 0$ for all x, which implies that $e^x \ge 1 + x$, see Figure 6.1.

(ii, iii) Indeed, $1 + 1/n \le \exp(1/n)$ and $(1 - 1/n)^n \le \exp(-1/n)$, by (i). As such

$$(1 + 1/n)^n \le \exp(n(1/n)) = e \qquad \text{and} \qquad (1 - 1/n)^n \le \exp(n(-1/n)) = \frac{1}{e},$$

which implies the left sides of (ii) and (iii). These are equivalent to

$$\frac{1}{e} \le \left(\frac{n}{n+1}\right)^n = \left(1 - \frac{1}{n+1}\right)^n \qquad \text{and} \qquad e \le \left(1 + \frac{1}{n-1}\right)^n,$$

which are the right side of (iii) [by replacing $n + 1$ by n], and the right side of (ii) [by replacing n by $n + 1$].

(iv) Indeed,

$$\frac{n^n}{n!} \le \sum_{i=0}^{\infty} \frac{n^i}{i!} = e^n,$$

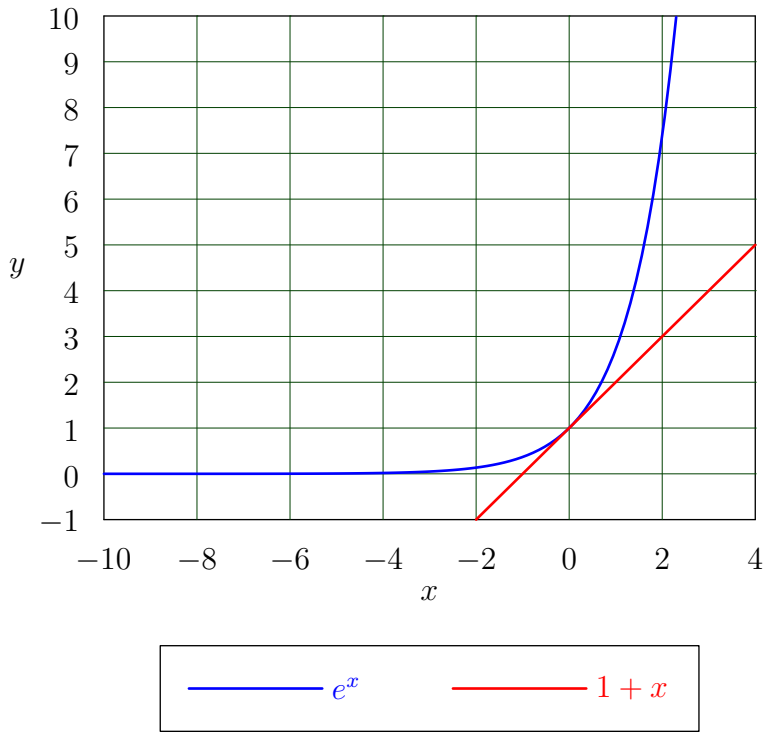


Figure 6.1

by the Taylor expansion of $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$. This implies that $(n/e)^n \leq n!$, as required.

As for the righthand side. The claim holds for $n = 0$ and $n = 1$. Let $f(n) = (n+1)^{n+1}/e^n$, and observe that by (ii), we have

$$\frac{f(n)}{f(n-1)} = \frac{(n+1)^{n+1}/e^n}{n^n/e^{n-1}} = \frac{n}{e}\left(\frac{n+1}{n}\right)^{n+1} = \frac{n}{e}\left(1+\frac{1}{n}\right)^{n+1} \geq n\frac{e}{e} = n.$$

Thus, we have

$$\frac{(n+1)^{n+1}}{e^n} = f(n) = \frac{f(n)}{f(n-1)} \cdot \frac{f(n-1)}{f(n-2)} \cdots \frac{f(1)}{f(0)} \geq n \cdot n - 1 \cdots 1 = n!$$

(v) For any $k \leq n$, we have $\frac{n}{k} \leq \frac{n-1}{k-1}$ since $kn - n = n(k-1) \leq k(n-1) = kn - k$. As such, $\frac{n}{k} \leq \frac{n-i}{k-i}$, for $1 \leq i \leq k-1$. As such,

$$\left(\frac{n}{k}\right)^k \leq \frac{n}{k} \cdot \frac{n-1}{k-1} \cdots \frac{n-i}{k-i} \cdots \frac{n-k+1}{1} = \frac{n!}{(n-k)!k!} = \binom{n}{k}.$$

As for the other direction, we have $\binom{n}{k} \leq \frac{n^k}{k!} \leq \frac{n^k}{(k/e)^k} = \left(\frac{ne}{k}\right)^k$, by (iii). ■

6.2. The birthday paradox

Consider a group of n people, and assume their birthdays are uniformly distributed no the dates in the year (this assumption is not quite true, but close enough). We are interested in the question of how large n has to be till we get a collision – that is, two people with the same birthday. Intuitively, since the year has $m = 364$ days, the probability of person to land on a specific birthday is $p = 1/364$. So the natural guess would be that n needs to be approximately 364. Surprisingly, the answer is much smaller.

Lemma 6.2.1. *Let $X_1, \ldots, X_n$ be n variables picked uniformly, randomly and independently from $[\![m]\!] = \{1, \ldots, m\}$. Then, the expected number of collisions is $\binom{n}{2}/m$.*

Proof: Let $Y_{i,j} = 1 \iff X_i = X_j$. We have that $\mathbb{E}\Big[Y_{i,j}\Big] = \mathbb{P}\Big[Y_{i,j} = 1\Big] = 1/m$. Thus, the expected number of collisions is

$$\mathbb{E}\left[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} Y_{i,j}\right] = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\mathbb{E}\Big[Y_{i,j}\Big] = \binom{n}{2}\frac{1}{m}.$$ ■

As such, for birthdays, for $m = 364$, and $n = 28$, we have that the expected number of collisions is

$$\binom{28}{2}\frac{1}{364} = \frac{378}{364} > 1.$$

This seems weird, but is it the truth?

Lemma 6.2.2. *Let $X_1, \ldots, X_n$ be n variables picked uniformly, randomly and independently from $[\![m]\!] = \{1, \ldots, m\}$. Then, the probability that no collision happened is at most $\exp\Big(-\binom{n}{2}/m\Big)$.*

Proof: Let $\mathcal{E}_i$ be the event that X_i is distinct from all the values in $X_1, \ldots, X_{i-1}$. Let $\mathcal{B}_i = \cap_{k=1}^{i}\mathcal{E}_k = \mathcal{B}_{i-1} \cap \mathcal{E}_i$ be the event that all of $X_1, \ldots, X_i$ are distinct. Clearly, we have

$$\mathbb{P}[\mathcal{E}_i \mid \mathcal{B}_{i-1}] = \mathbb{P}[\mathcal{E}_i \mid \mathcal{E}_1 \cap \cdots \cap \mathcal{E}_{i-1}] = \frac{m-(i-1)}{m} = 1 - \frac{i-1}{m} \le \exp\left(-\frac{i-1}{m}\right).$$

Observe that

$$\mathbb{P}[\mathcal{B}_i] = \mathbb{P}[\mathcal{B}_{i-1}]\frac{\mathbb{P}[\mathcal{E}_i \cap \mathcal{B}_{i-1}]}{\mathbb{P}[\mathcal{B}_{i-1}]} = \mathbb{P}[\mathcal{B}_{i-1}]\,\mathbb{P}[\mathcal{E}_i \mid \mathcal{B}_{i-1}] \le \exp\left(-\frac{i-1}{m}\right)\mathbb{P}[\mathcal{B}_{i-1}] \le \prod_{k=1}^{i}\exp\left(-\frac{k-1}{m}\right).$$

$$= \exp\left(-\sum_{k=1}^{i}\frac{k-1}{m}\right) = \exp\left(-\frac{i(i-1)}{2}\frac{1}{m}\right) = \exp\left(-\binom{i}{2}/m\right).$$

Which implies the desired claim for $i = n$. ■

6.3. Occupancy Problems

Problem 6.3.1. We are throwing m balls into n bins randomly (i.e., for every ball we randomly and uniformly pick a bin from the n available bins, and place the ball in the bin picked). There are many natural questions one can ask here:

(A) What is the maximum number of balls in any bin?
(B) What is the number of bins which are empty?
(C) How many balls do we have to throw, such that all the bins are non-empty, with reasonable probability?

Theorem 6.3.2. *With probability at least $1 - 1/n$, no bin has more than $k^* = \left\lceil \frac{3\ln n}{\ln\ln n} \right\rceil$ balls in it.*

Proof: Let X_i be the number of balls in the ith bins, when we throw n balls into n bins (i.e., $m = n$). Clearly,

$$\mathbb{E}[X_i] = \sum_{j=1}^{n} \mathbb{P}[\text{The } j\text{th ball fall in } i\text{th bin}] = n \cdot \frac{1}{n} = 1,$$

by linearity of expectation. The probability that the first bin has exactly i balls is

$$\binom{n}{i}\left(\frac{1}{n}\right)^i\left(1-\frac{1}{n}\right)^{n-i} \le \binom{n}{i}\left(\frac{1}{n}\right)^i \le \left(\frac{ne}{i}\right)^i\left(\frac{1}{n}\right)^i = \left(\frac{e}{i}\right)^i$$

This follows by Lemma 6.1.1 (iv).

Let $C_j(k)$ be the event that the jth bin has k or more balls in it. Then,

$$\mathbb{P}[C_1(k)] \le \sum_{i=k}^{n}\left(\frac{e}{i}\right)^i \le \left(\frac{e}{k}\right)^k\left(1+\frac{e}{k}+\frac{e^2}{k^2}+\ldots\right) = \left(\frac{e}{k}\right)^k \frac{1}{1-e/k}.$$

For $k^* = c\ln n/\ln\ln n$, we have

$$\begin{aligned}\mathbb{P}[C_1(k^*)] &\le \left(\frac{e}{k^*}\right)^{k^*}\frac{1}{1-e/k^*} \le 2\exp\Big(k^*(1-\ln k^*)\Big) \le 2\exp\left(-\frac{k^*\ln k^*}{2}\right)\\ &\le 2\exp\left(-\frac{c\ln n}{2\ln\ln n}\underbrace{\ln\frac{c\ln n}{\ln\ln n}}_{\approx \ln\ln n}\right) \le 2\exp\left(-\frac{c\ln n}{4}\right) \le \frac{1}{n^2},\end{aligned}$$

for n and c sufficiently large.

Let us redo this calculation more carefully (yuk!). For $k^* = \lceil(3\ln n)/\ln\ln n\rceil$, we have

$$\begin{aligned}\mathbb{P}[C_1(k^*)] &\le \left(\frac{e}{k^*}\right)^{k^*}\frac{1}{1-e/k^*} \le 2\left(\frac{e}{(3\ln n)/\ln\ln n}\right)^{k^*} = 2\exp\Big(\underbrace{1-\ln 3}_{<0} - \ln\ln n + \ln\ln\ln n\Big)^{k^*}\\ &\le 2\exp((-\ln\ln n + \ln\ln\ln n)k^*)\\ &\le 2\exp\left(-3\ln n + 6\ln n\frac{\ln\ln\ln n}{\ln\ln n}\right) \le 2\exp(-2.5\ln n) \le \frac{1}{n^2},\end{aligned}$$

for n large enough. We conclude, that since there are n bins and they have identical distributions that

$$\mathbb{P}[\text{any bin contains more than } k^* \text{ balls}] \le \sum_{i=1}^{n} C_i(k^*) \le \frac{1}{n}.$$ ■

Exercise 6.3.3. Show that when throwing $m = n\ln n$ balls into n bins, with probability $1 - o(1)$, every bin has $O(\log n)$ balls.

6.3.1. The Probability of all bins to have exactly one ball

Next, we are interested in the probability that all m balls fall in distinct bins. Let X_i be the event that the ith ball fell in a distinct bin from the first $i-1$ balls. We have:

$$\begin{aligned}\mathbb{P}[\cap_{i=2}^m X_i] &= \mathbb{P}[X_2]\prod_{i=3}^{m}\mathbb{P}\Big[X_i \;\Big|\; \cap_{j=2}^{i-1}X_j\Big] \le \prod_{i=2}^{m}\left(\frac{n-i+1}{n}\right) \le \prod_{i=2}^{m}\left(1-\frac{i-1}{n}\right)\\ &\le \prod_{i=2}^{m} e^{-(i-1)/n} \le \exp\left(-\frac{m(m-1)}{2n}\right),\end{aligned}$$

thus for $m = \lceil \sqrt{2n} + 1 \rceil$, the probability that all the m balls fall in different bins is smaller than $1/e$.

This is sometime referred to as the *birthday paradox*, which was already mentioned above. You have $m = 30$ people in the room, and you ask them for the date (day and month) of their birthday (i.e., $n = 365$). The above shows that the probability of all birthdays to be distinct is $\exp(-30 \cdot 29/730) \leq 1/e$. Namely, there is more than 50% chance for a birthday collision, a simple but counter-intuitive phenomena.

6.4. The Coupon Collector's Problem

There are n types of coupons, and at each trial one coupon is picked in random. How many trials one has to perform before picking all coupons? Let m be the number of trials performed. We would like to bound the probability that m exceeds a certain number, and we still did not pick all coupons.

Let $C_i \in \{1, \ldots, n\}$ be the coupon picked in the ith trial. The jth trial is a success, if C_j was not picked before in the first $j-1$ trials. Let X_i denote the number of trials from the ith success, till after the $(i+1)$th success. Clearly, the number of trials performed is

$$X = \sum_{i=0}^{n-1} X_i.$$

Furthermore, X_i has a geometric distribution with parameter p_i, that is $X_i \sim \text{Geom}(p_i)$, with $p_i = (n-i)/n$. The expectation and variance of X_i are

$$\mathbb{E}[X_i] = \frac{1}{p_i} \qquad \text{and} \qquad \mathbb{V}[X_i] = \frac{1 - p_i}{p_i^2}.$$

Lemma 6.4.1. *Let X be the number of rounds till we collection all n coupons. Then, $\mathbb{V}[X] \approx \left(\pi^2/6\right)n^2$ and its standard deviation is $\sigma_X \approx (\pi/\sqrt{6})n$.*

Proof: The probability of X_i to succeed in a trial is $p_i = (n-i)/n$, and X_i has the geometric distribution with probability p_i. As such $\mathbb{E}[X_i] = 1/p_i$, and $\mathbb{V}[X_i] = q/p^2 = (1-p_i)/p_i^2$.

Thus,

$$\mathbb{E}[X] = \sum_{i=0}^{n-1} \mathbb{E}[X_i] = \sum_{i=0}^{n-1} \frac{n}{n-i} = nH_n = n(\ln n + \Theta(1)) = n \ln n + O(n),$$

where $H_n = \sum_{i=1}^{n} 1/i$ is the nth Harmonic number.

As for variance, using the independence of $X_0, \ldots, X_{n-1}$, we have

$$\begin{aligned}
\mathbb{V}[X] &= \sum_{i=0}^{n-1} \mathbb{V}[X_i] = \sum_{i=0}^{n-1} \frac{1-p_i}{p_i^2} = \sum_{i=0}^{n-1} \frac{1-(n-i)/n}{\left(\frac{n-i}{n}\right)^2} = \sum_{i=0}^{n-1} \frac{i/n}{\left(\frac{n-i}{n}\right)^2} = \sum_{i=0}^{n-1} \frac{i}{n}\left(\frac{n}{n-i}\right)^2 \\
&= n \sum_{i=0}^{n-1} \frac{i}{(n-i)^2} = n \sum_{i=1}^{n} \frac{n-i}{i^2} = n\left(\sum_{i=1}^{n} \frac{n}{i^2} - \sum_{i=1}^{n} \frac{1}{i}\right) = n^2 \sum_{i=1}^{n} \frac{1}{i^2} - nH_n \approx \frac{\pi^2}{6} n^2,
\end{aligned}$$

since $\lim_{n\to\infty} \sum_{i=1}^{n} \frac{1}{i^2} = \pi^2/6$, we have $\lim_{n\to\infty} \dfrac{\mathbb{V}[X]}{n^2} = \dfrac{\pi^2}{6}$. ■

This implies a weak bound on the concentration of X, using Chebyshev inequality, we have

$$\mathbb{P}\left[X \geq n \ln n + n + t \cdot n \frac{\pi}{\sqrt{6}}\right] \leq \mathbb{P}\left[\left|X - \mathbb{E}[X]\right| \geq t\sigma_X\right] \leq \frac{1}{t^2},$$

Note, that this is somewhat approximate, and hold for n sufficiently large.

6.5. Notes

The material in this note covers parts of [MR95, sections 3.1, 3.2, 3.6]

References

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

Chapter 7

On k-wise independence

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

7.1. Pairwise independence

7.1.1. Pairwise independence

Definition 7.1.1. A set of random variables $X_1, \ldots, X_n$ is ***pairwise independent***, if for any pair of values α, β, and any two indices i, j, we have that

$$\mathbb{P}\Big[X_i = \alpha \text{ and } Y_j = \beta\Big] = \mathbb{P}[X_i = \alpha]\, \mathbb{P}\Big[Y_j = \beta\Big].$$

Namely, the variables are independent if you look at pairs of variables. Compare this to the much stronger property of independence.

Definition 7.1.2. A set of random variables $X_1, \ldots, X_n$ is ***independent***, if for any t, and any t values $\alpha_1, \ldots, \alpha_t$, and any t indices $i_1, \ldots, i_t$, we have that

$$\mathbb{P}[X_{i_1} = \alpha_1, X_{i_2} = \alpha_2, \ldots, \text{ and } Y_{i_t} = \alpha_{i_t}] = \prod_{j=1}^{t} \mathbb{P}\Big[X_{i_j} = \alpha_j\Big].$$

7.1.2. A pairwise independent set of bits

Let n be a number which is a power of two. As such, $t = \log_2 n = \lg n$ is an integer. Let $X_0, \ldots, X_{t-1}$ be truly independent random bits, each one of them is 1 with probability $1/2$.

For a non-negative integer number x, let $\mathrm{bit}(x, j) \in \{0, 1\}$ be the jth bit in the binary representation of x. That is, we have $x = \sum_j \mathrm{bit}(x, j) 2^j$.

For an index $i = 1, \ldots, 2^t - 1$, we define $Y_i = \bigotimes_{j:\mathrm{bit}(i,j)=1} X_j$, where $\otimes$ is the *xor* operator.

Lemma 7.1.3. *The random variables $Y_1, Y_2, \ldots, Y_{n-1}$ are pairwise independent.*

Proof: We claim that, for any i, we have $\mathbb{P}[Y_i = 1] = \mathbb{P}[Y_i = 0] = 1/2$. So fix i, and let α be an index such that $\mathrm{bit}(i, \alpha) = 1$, and observe that this follows readily if pick all the true random variables $X_0, \ldots, X_{t-1}$ in such an order such that X_α is the last one to be set.

Next, consider two distinct indices i, i', and two arbitrary values v, v'. We need to prove that

$$\mathbb{P}[Y_i = v \text{ and } Y_{i'} = v'] = \mathbb{P}[Y_i = v]\, \mathbb{P}[Y_{i'} = v'] = \frac{1}{4}.$$

To this end, let $B = \{j \mid \mathrm{bit}(i,j) = 1\}$ and $B' = \{j \mid \mathrm{bit}(i',j) = 1\}$. If there is an index $\beta \in B \setminus B'$, then we have

$$\mathbb{P}\Big[Y_i = v \mid Y_{i'} = v'\Big] = \mathbb{P}\Big[\bigotimes\nolimits_{j:\mathrm{bit}(i,j)=1} X_j = v \;\Big|\; Y_{i'} = v'\Big] = \mathbb{P}\Big[X_\beta \otimes \bigotimes_{j:\mathrm{bit}(i,j)=1} X_j = v \;\Big|\; Y_{i'} = v'\Big]$$
$$= \mathbb{P}\Big[X_\beta = \Big(v \otimes \bigotimes_{j:\mathrm{bit}(i,j)=1} X_j\Big) \;\Big|\; Y_{i'} = v'\Big] = \frac{1}{2}.$$

This implies that $\mathbb{P}[Y_i = v \text{ and } Y_{i'} = v'] = \mathbb{P}[Y_i = v \mid Y_{i'} = v']\,\mathbb{P}[Y_{i'} = v'] = (1/2)(1/2) = 1/4$, as claimed.

A similar argument implies that if there is an index $\beta \in B' \setminus B$, then $\mathbb{P}[Y_{i'} = v' \mid Y_i = v] = 1/2$, which implies the claim in this case.

Since $i \neq i'$, one of the two scenarios must happen, implying the claim. ■

7.1.3. An application: Max cut

Given a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with n vertices and m edges, consider the problem of computing the max-cut. That is, computing the set of vertices S, such that the cut

$$(S, \overline{S}) = (S, \mathsf{V} \setminus S) = \{uv \in \mathsf{E} \mid u \in S, v \in \mathsf{V} \setminus S\}.$$

is of maximum cardinality.

7.1.3.0.1. Algorithm. To this end, let $Y_1, \ldots, Y_n$ be the pairwise independent bits of Section 7.1.2. Here, let S be the set of all vertices $v_i \in \mathsf{V}$, such that $Y_i = 1$. The algorithm outputs $(S, \overline{S})$ as the candidate cut.

7.1.4. Analysis

Lemma 7.1.4. *The expected size of the cut computed by the above algorithm is $m/2$, where $m = |\mathsf{E}(\mathsf{G})|$.*

Proof: Let Z_{uv} be an indicator variable for the event that the edge $uv \in \mathsf{E}$ is in the cut $(S, \overline{S})$.

We have that

$$\mathbb{E}\Big[\big|(S, \overline{S})\big|\Big] = \mathbb{E}\Big[\sum_{uv \in \mathsf{E}} Z_{uv}\Big] = \sum_{uv \in \mathsf{E}} \mathbb{E}[Z_{uv}] = \sum_{uv \in \mathsf{E}} \mathbb{P}[Y_u \neq Y_v] = |\mathsf{E}|/2,$$

using linearity of expectation and pairwise independence. ■

Lemma 7.1.5. *Given a graph G with n vertices and m edges, say stored in a read only memory, one can compute a max-cut of G, and the edges in it, using $O(\log n)$ random bits, and $O(\log n)$ RAM bits. Furthermore, the expected size of the cut is $\geq m/2$.*

Proof: The algorithm description is above. The pairwise independence is also described above, and requires only $O(\log n)$ random bits, which needs to be stored. Otherwise, all we need is to scan the edges of the graph, and for each one to decide if it is, or not in the cut. Clearly, this can be done using $O(\log n)$ RAM bits. ■

Compare this to the natural randomized algorithm of computing a random subset S. This would require using n random bits, and n bits of space to store it.

Max cut in the streaming model. Imagine that the edges of the graph are given to you via streaming: You are told the number of vertices in advance, but then edges arrive one by one. The above enables you to compute the cut in a streaming fashion using $O(\log n)$ bits. Alternatively, you can output the edges in a streaming fashion.

Another way of thinking about it, is that given a set $S = \{s_1, \ldots, s_n\}$ of n elements, we can use the above to select a random sample where every element is selected with probability half, and the samples are pairwise independent. The kicker is that to specify the sample, or decide if an element is in the sample, we can do it using $O(\log n)$ bits. This is a huge save compared to the regular n bits required as storage to remember the sample.

It is clear however that we want a stronger concept – where things are k-wise independent.

7.2. On k-wise independence

7.2.1. Definition

Definition 7.2.1. A set of variables $X_1, \ldots, X_n$ are ***k-wise independent*** if for any set $I = \{i_1, i_2, \ldots, i_t\}$ of indices, for $t \leq k$, and any set of values $v_1, \ldots, v_t$, we have that

$$\mathbb{P}[X_{i_1} = v_1 \text{ and } X_{i_2} = v_2 \text{ and } \cdots \text{ and } X_{i_t} = v_t] = \prod_{j=1}^{t} \mathbb{P}\Big[X_{i_j} = v_j\Big].$$

Observe, that verifying the above property needs to be done only for $t = k$.

7.2.2. On working modulo prime

Definition 7.2.2. For a number p, let $\mathbb{Z}_n = \{0, \ldots, n-1\}$.

For two integer numbers x and y, the ***quotient*** of x/y is $x \operatorname{div} y = \lfloor x/y \rfloor$. The ***remainder*** of x/y is $x \bmod y = x - y\lfloor x/y \rfloor$. If the $x \bmod y = 0$, than y ***divides*** x, denoted by $y \mid x$. We use $\alpha \equiv \beta \pmod p$ or $\alpha \equiv_p \beta$ to denote that α and β are ***congruent modulo*** p; that is $\alpha \bmod p = \beta \bmod p$ – equivalently, $p \mid (\alpha - \beta)$.

Lemma 7.2.3. *Let p be a prime number.*

(A) For any $\alpha, \beta \in \{1, \ldots, p-1\}$, we have that $\alpha\beta \not\equiv 0 \pmod p$.

(B) For any $\alpha, \beta, i \in \{1, \ldots, p-1\}$, such that $\alpha \neq \beta$, we have that $\alpha i \not\equiv \beta i \pmod p$.

(C) For any $x \in \{1, \ldots, p-1\}$ there exists a unique y such that $xy \equiv 1 \pmod p$. The number y is the **inverse** *of x, and is denoted by x^{-1} or $1/x$.*

Proof: (A) If $\alpha\beta \equiv 0 \pmod p$, then p must divide $\alpha\beta$, as it divides 0. But α, β are smaller than p, and p is prime. This implies that either $p \mid \alpha$ or $p \mid \beta$, which is impossible.

(B) Assume that $\alpha > \beta$. Furthermore, for the sake of contradiction, assume that $\alpha i \equiv \beta i \pmod p$. But then, $(\alpha - \beta)i \equiv 0 \pmod p$, which is impossible, by (A).

(C) For any $\alpha \in \{1, \ldots, p-1\}$, consider the set $L_\alpha = \{\alpha * 1 \bmod p, \alpha * 2 \bmod p, \ldots, \alpha * (p-1) \bmod p\}$. By (A), zero is not in L_α, and by (B), L_α must contain $p-1$ distinct values. It follows that $L_\alpha = \{1, 2, \ldots, p-1\}$. As such, there exists exactly one number $y \in \{1, \ldots, p-1\}$, such that $\alpha y \equiv 1 \pmod p$. ■

Lemma 7.2.4. *Consider a prime p, and any numbers $x, y \in \mathbb{Z}_p$. If $x \neq y$ then, for any $a, b \in \mathbb{Z}_p$, such that $a \neq 0$, we have $ax + b \not\equiv ay + b \pmod p$.*

Proof: Assume $y > x$ (the other case is handled similarly). If $ax+b \equiv ay+b \pmod p$ then $a(x-y) \pmod p = 0$ and $a \neq 0$ and $(x-y) \neq 0$. However, a and $x-y$ cannot divide p since p is prime and $a < p$ and $0 < x-y < p$. ■

Lemma 7.2.5. *Consider a prime p, and any numbers $x, y \in \mathbb{Z}_p$. If $x \neq y$ then, for each pair of numbers $r, s \in \mathbb{Z}_p = \{0, 1, \ldots, p-1\}$, such that $r \neq s$, there is exactly one unique choice of numbers $a, b \in \mathbb{Z}_p$ such that $ax + b \pmod p = r$ and $ay + b \pmod p = s$.*

Proof: Solve the system of equations

$$ax + b \equiv r \pmod p \qquad \text{and} \qquad ay + b \equiv s \pmod p.$$

We get $a = \frac{r-s}{x-y} \pmod p$ and $b = r - ax \pmod p$. ■

7.2.3. Construction of k-wise independence variables

7.2.4. Construction

Consider the following matrix, aka the ***Vandermonde matrix***, defined by n variables:

$$V = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \ldots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^{n-1} \end{bmatrix}.$$

Claim 7.2.6. $\det(V) = \prod_{1 \leq i < j \leq n} (x_j - x_i)$.

Proof: One can prove this in several ways, and we include a proof via properties of polynomials. The determinant $\det(V)$ is a polynomial in the variables $x_1, x_2, \ldots, x_n$. Formally, let Π be the set of all permutations of $[\![n]\!] = \{1, \ldots, n\}$. For a permutation $\pi \in \Pi$, let $\text{sign}(\pi) \in \{-1, +1\}$ denote the sign of this permutation. We have that

$$f(x_1, x_2, \ldots, x_n) = \det(V) = \sum_{\pi \in \Pi} \text{sign}(\pi) x_i^{\pi(i)}.$$

Every monomial in this polynomial has total degree $\sum_{i=1}^{n} \pi(i) = 1 + 2 + \cdots + n = n(n-1)/2$. Observe, that if we replace x_j by x_i, then we have $f(x_1, \ldots, x_i, \ldots, x_{j-1}, x_i, x_{j+1}, \ldots, x_n)$ is the determinant of a matrix with two identical rows, and such a matrix has a zero determinate. Namely, the polynomial f is zero if $x_i = x_j$. This implies that $x_j - x_i$ divides f. We conclude that the polynomial $g \equiv \prod_{1 \leq i < j \leq n} (x_j - x_i)$ divides f. Namely, we can write $f = g * h$, where h is some polynomial.

Consider the monomial $x_2 x_3^2 \cdots x_n^{n-1}$. It appears in f with coefficient 1. Similarly, it generated in g by selecting the first term in each sub-polynomial, that is $\prod_{1 \leq i < j \leq n} \left(x_j - x_i \right)$. It is to verify that this is the only time this monomial appears in g. This implies that $h = 1$. We conclude that $f = g$, as claimed. ■

Claim 7.2.7. *If $x_1, \ldots, x_n$ are distinct, then the Vandermonde matrix V is invertible.*

Proof: By Claim 7.2.6, the determinant of V is $\det(V) = \prod_{1 \leq i < j \leq n} (x_j - x_i)$. This quantity is non-zero if the xs are distinct, and a matrix is invertible in such a case. ■

Lemma 7.2.8. *For a vector $\mathsf{b} = (b_0, \ldots, b_{k-1}) \in \mathbb{Z}_p^k$, consider the associated polynomial $f(x, \mathsf{b}) = \sum_{i=0}^{k-1} b_i x^i \bmod p$. For any k distinct values $\alpha_1, \ldots, \alpha_k \in \mathbb{Z}_p$, and k values $v_1, \ldots, v_k \in \mathbb{Z}_p$, there is a unique choice of b, such that $f(\alpha_i) = v_i \bmod p$, for $i = 1, \ldots, k$.*

Proof: Let $\alpha_i = \left(1, \alpha_i, \alpha_i^2, \cdots, \alpha_i^{k-1}\right)$. We have that $f(\alpha_i, \mathsf{b}) = \langle \alpha_i, \mathsf{b} \rangle \bmod p$. This translates into the linear system

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{pmatrix} \mathsf{b}^T = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{pmatrix} \iff \mathsf{M}\,\mathsf{b}^T = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{pmatrix} \qquad \text{where} \qquad \mathsf{M} = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \dots & \alpha_2^{n-1} \\ 1 & \alpha_3 & \alpha_3^2 & \dots & \alpha_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_n & \alpha_n^2 & \dots & \alpha_n^{n-1} \end{bmatrix}.$$

The matrix M is the Vandermonde matrix, and by Claim 7.2.7 it is invertible. We thus get there exists a unique solution to this system of linear equations (modulo p). ■

The construction. So, let us pick independently and uniformly k values $b_0.b_1, \ldots, b_{k-1} \in \mathbb{Z}_p$, let $\mathsf{b} = (b_0, b_1, \ldots, b_{k-1})$. $g(x) = \sum_{i=0}^{k-1} b_i x^i \bmod p$, and consider the random variables

$$Y_i = g(i), \qquad \forall i \in \mathbb{Z}_p.$$

Lemma 7.2.9. *The variables $Y_0, \ldots, Y_{p-1}$ are uniformly distributed and k-wise independent.*

Proof: The uniform distribution for each Y_i follows readily by picking b_0 last, and observing that each such choice corresponds to a different value of Y_i.

As for the k-independence, observe that for any set $I = \{i_1, i_2, \ldots, i_k\}$ of indices, for $t \le k$, and any set of values $v_1, \ldots, v_k \in \mathbb{Z}_p$, we have that the event

$$Y_{i_1} = v_1 \text{ and } Y_{i_2} = v_2 \text{ and } \cdots \text{ and } Y_{i_k} = v_k$$

happens only for a unique choice of b, by Lemma 7.2.8. But there are p^k such choices. We conclude that the probability of the above event is $1/p^k = \prod_{j=1}^{k} \mathbb{P}\left[Y_{i_j} = v_j\right]$, as desired. ■

We summarize the result for later use.

Theorem 7.2.10. *let p be a prime number, and pick independently and uniformly k values $b_0.b_1, \ldots, b_{k-1} \in \mathbb{Z}_p$, and let $g(x) = \sum_{i=0}^{k-1} b_i x^i \bmod p$. Then the random variables*

$$Y_0 = g(0), \ldots, Y_{p-1} = g(p-1).$$

are uniformly distributed in $\mathbb{Z}_p$ and are k-wise independent.

7.2.5. Applications of k-wide independent variables

7.2.5.1. Product of expectations

Lemma 7.2.11. *If $X_1, \ldots, X_k$ are k-wise independent, then $\mathbb{E}[X_1 \cdots X_k] = \mathbb{E}[X_1] \cdots \mathbb{E}[X_k]$.*

Proof: Immediate. ■

7.2.5.2. Application: Using less randomization for a randomized algorithm

We can consider a randomized algorithm, to be a deterministic algorithm $\mathbf{Alg}(x, r)$ that receives together with the input x, a random string r of bits, that it uses to read random bits from. Let us redefine **RP**:

Definition 7.2.12. The class **RP** (for Randomized Polynomial time) consists of all languages L that have a deterministic algorithm $\mathbf{Alg}(x, r)$ with worst case polynomial running time such that for any input $x \in \Sigma^*$,

- $x \in L \implies \mathbf{Alg}(x, r) = 1$ for half the possible values of r.
- $x \notin L \implies \mathbf{Alg}(x, r) = 0$ for all values of r.

Let assume that we now want to minimize the number of random bits we use in the execution of the algorithm (Why?). If we run the algorithm t times, we have confidence 2^{-t} in our result, while using $t \log n$ random bits (assuming our random algorithm needs only $\log n$ bits in each execution). Similarly, let us choose two random numbers from $\mathbb{Z}_n$, and run $\mathbf{Alg}(x, a)$ and $\mathbf{Alg}(x, b)$, gaining us only confidence $1/4$ in the correctness of our results, while requiring $2 \log n$ bits.

Can we do better? Let us define $r_i = ai + b \mod n$, where a, b are random values as above (note, that we assume that n is prime), for $i = 1, \ldots, t$. Thus $Y = \sum_{i=1}^{t} \mathbf{Alg}(x, r_i)$ is a sum of random variables which are pairwise independent, as the r_i are pairwise independent. Assume, that $x \in L$, then we have $\mathbb{E}[Y] = t/2$, and $\sigma_Y^2 = \mathbb{V}[Y] = \sum_{i=1}^{t} \mathbb{V}[\mathbf{Alg}(x, r_i)] \leq t/4$, and $\sigma_Y \leq \sqrt{t}/2$. The probability that all those executions failed, corresponds to the event that $Y = 0$, and

$$\mathbb{P}\Big[Y = 0\Big] \leq \mathbb{P}\left[\Big|Y - \mathbb{E}[Y]\Big| \geq \frac{t}{2}\right] = \mathbb{P}\left[\Big|Y - \mathbb{E}[Y]\Big| \geq \frac{\sqrt{t}}{2} \cdot \sqrt{t}\right] \leq \frac{1}{t},$$

by the Chebyshev inequality. Thus we were able to "extract" from our random bits, much more than one would naturally suspect is possible. We thus get the following result.

Lemma 7.2.13. *Given an algorithm* $\mathbf{Alg}$ *in* **RP** *that uses* $\lg n$ *random bits, one can run it* t *times, such that the runs results in a new algorithm that fails with probability at most* $1/t$*, and uses only* $2 \lg n$ *random bits.*

7.3. Higher moment inequalities

The following is the higher moment variant of Chebychev inequality.

Lemma 7.3.1. *For a random variable* X*, we have that* $\mathbb{P}\left[|X - \mathbb{E}[X]| \geq t\mathbb{E}\Big[|X - \mathbb{E}[X]|^k\Big]^{1/k}\right] \leq \frac{1}{t^k}$

Proof: Setting $Z = |X - \mathbb{E}[X]|^k$, and raising the inequality by a power of k, we have

$$\mathbb{P}\Big[|X - \mathbb{E}[X]| \geq t\mathbb{E}[|X - \mathbb{E}[X]|^k]^{1/k}\Big] = \mathbb{P}\Big[Z^{1/k} \geq t\,\mathbb{E}[Z]^{1/k}\Big] = \mathbb{P}\Big[Z \geq t^k\,\mathbb{E}[Z]\Big] \leq \frac{1}{t^k},$$

by Markov's inequality. ■

The problem is that computing (or even bounding) the kth moment $M_k(X) = \mathbb{E}\Big[|X - \mathbb{E}[X]\,|^k\Big]$ is usually not easy. Let us do it for one interesting example.

Lemma 7.3.2. *Consider* k *be an even integer and let* $X_1, \ldots, X_n$ *be* n *random independent variables such that* $\mathbb{P}[X_i = -1] = \mathbb{P}[X_i = +1] = 1/2$*. Let* $X = \sum_{i=1}^{n} X_i$*. Then, we have*

$$\mathbb{P}\left[|X| \geq \frac{tk}{2}\sqrt{n}\right] \leq \frac{1}{t^k}.$$

Proof: Observe that $\mathbb{E}[X] = n\,\mathbb{E}[X_1] = 0$. We are interested in computing

$$M_k(X) = \mathbb{E}\Big[X^k\Big] = \mathbb{E}\Big[\Big(\sum_i X_i\Big)^k\Big] = \mathbb{E}\Big[\sum_{i_1=1}^{n} \cdots \sum_{i_k=1}^{n} X_{i_1} X_{i_2} \cdots X_{i_k}\Big] = \sum_{i_1=1}^{n} \cdots \sum_{i_k=1}^{n} \mathbb{E}[X_{i_1} X_{i_2} \cdots X_{i_k}] \tag{7.1}$$

Consider a term in the above summation, where one of the indices (say i_1) has a unique value among $i_1, i_2, \ldots, i_k$. By independence, we have

$$\mathbb{E}[X_{i_1} X_{i_2} \cdots X_{i_k}] = \mathbb{E}[X_{i_1}]\,\mathbb{E}[X_{i_2} \cdots X_{i_k}] = 0,$$

since $\mathbb{E}[X_{i_1}] = 0$. As such, in the above all terms that have a unique index disappear. A term that does not disappear is going to be of the form

$$\mathbb{E}\Big[X_{i_1}^{\alpha_1} X_{i_2}^{\alpha_2} \ldots X_{i_\ell}^{\alpha_\ell}\Big] = \mathbb{E}\Big[X_{i_1}^{\alpha_1}\Big]\,\mathbb{E}\Big[X_{i_2}^{\alpha_2}\Big] \ldots \mathbb{E}\Big[X_{i_\ell}^{\alpha_\ell}\Big]$$

where $\alpha_i \geq 2$, and $\sum_i \alpha_i = k$. Observe that

$$\mathbb{E}[X_1^t] = \begin{cases} 0 & t \text{ is odd} \\ 1 & t \text{ is even.} \end{cases}$$

As such, all the terms in the summation of Eq. (7.1) that have value that is not zero, have value one. These terms corresponds to tuples $T = (i_1, i_2, \ldots, i_k)$, such that the set of values $I(T) = \{i_1, \ldots, i_k\}$ has at most $k/2$ values, and furthermore, each such value appears an even number of times in T (here $k/2$ is an integer as k is even by assumption). We conclude that the total number of such tuples is at most

$$n^{k/2}(k/2)^k.$$

Note, that this is a naive bound – indeed, we choose the $k/2$ values that are in $I(T)$, and then we generate the tuple T, by choosing values for each coordinate separately. We thus conclude that

$$M_k(X) = \mathbb{E}\Big[X^k\Big] \leq n^{k/2}(k/2)^k.$$

Since k is even, we have $\mathbb{E}\Big[X^k\Big] = \mathbb{E}\Big[|X|^k\Big]$, and by Lemma 7.3.1, we have

$$\mathbb{P}\left[|X| \geq \frac{tk}{2}\sqrt{n}\right] = \mathbb{P}\left[|X| \geq t\Big(n^{k/2}(k/2)^k\Big)^{1/k}\right] \leq \mathbb{P}\left[|X| \geq t\mathbb{E}\Big[|X|^k\Big]^{1/k}\right] \leq 1/t^k. \qquad \blacksquare$$

Corollary 7.3.3. *Consider k be an even integer and let $X_1, \ldots, X_n$ be n random independent variables such that $\mathbb{P}[X_i = -1] = \mathbb{P}[X_i = +1] = 1/2$. For $X = \sum_{i=1}^n X_i$, and any k, we have $\mathbb{P}\Big[|X| \geq k\sqrt{n}\Big] \leq 1/2^k$.*

Observe, that the above proof did not require all the variables to be purely independent – it was enough that they are k-wise independent. We readily get the following.

Definition 7.3.4. Given n random variables $X_1, \ldots, X_n$ they are ***k-wise independent***, if for any k of them (i.e., $i_1 < i_2, \ldots, i_k$), and any k values $x_1, \ldots, x_k$, we have

$$\mathbb{P}\Big[\bigcap_{\ell=1}^{k}(X_{i_\ell} = v_\ell)\Big] = \prod_{\ell=1}^{k} \mathbb{P}[X_{i_\ell} = v_\ell].$$

Informally, variables are k-wise independent, if any k of them (on their own) looks totally random.

Lemma 7.3.5. *Let $k > 0$ be an even integer, and let $X_1, \ldots, X_n$ be n random independent variables, that are k-wise independent, such that $\mathbb{P}[X_i = -1] = \mathbb{P}[X_i = +1] = 1/2$. Let $X = \sum_{i=1}^n X_i$. Then, we have*

$$\mathbb{P}\Big[|X| \geq \frac{tk}{2}\sqrt{n}\Big] \leq \frac{1}{t^k}.$$

References

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

Chapter 8

Hashing

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "I tried to read this book, Huckleberry Finn, to my grandchildren, but I couldnt get past page six because the book is fraught with the n-word. And although they are the deepest-thinking, combat-ready eight- and ten-year-olds I know, I knew my babies werent ready to comprehend Huckleberry Finn on its own merits. Thats why I took the liberty to rewrite Mark Twains masterpiece. Where the repugnant n-word occurs, I replaced it with warrior and the word slave with dark-skinned volunteer."
>
> Paul Beatty, The Sellout

8.1. Introduction

We are interested here in dictionary data structure. The settings for such a data-structure:

(A) $\mathcal{U}$: universe of keys with total order: numbers, strings, etc.
(B) Data structure to store a subset $S \subseteq \mathcal{U}$
(C) **Operations:**
  (A) **search/lookup**: given $x \in \mathcal{U}$ is $x \in S$?
  (B) **insert**: given $x \notin S$ add x to S.
  (C) **delete**: given $x \in S$ delete x from S
(D) ***Static*** structure: S given in advance or changes very infrequently, main operations are lookups.
(E) ***Dynamic*** structure: S changes rapidly so inserts and deletes as important as lookups.

Common constructions for such data-structures, include using a static sorted array, where the lookup is a binary search. Alternatively, one might use a *balanced* search tree (i.e., red-black tree). The time to perform an operation like lookup, insert, delete take $O(\log |S|)$ time (comparisons).

Naturally, the above are potently an "overkill", in the sense that sorting is unnecessary. In particular, the universe $\mathcal{U}$ may not be (naturally) totally ordered. The keys correspond to large objects (images, graphs etc) for which comparisons are expensive. Finally, we would like to improve "average" performance of lookups to $O(1)$ time, even at cost of extra space or errors with small probability: many applications for fast lookups in networking, security, etc.

Hashing and Hash Tables. The hash-table data structure has an associated (hash) table/array T of size m (the table ***size***). A hash function $h : \mathcal{U} \to \{0, \ldots, m-1\}$. An item $x \in \mathcal{U}$ hashes to slot $h(x)$ in T.

Given a set $S \subseteq \mathcal{U}$, in a perfect ideal situation, each element $x \in S$ hashes to a distinct slot in T, and we store x in the slot $h(x)$. The **Lookup** for an item $y \in \mathcal{U}$, is to check if $T[h(y)] = y$. This takes constant time.

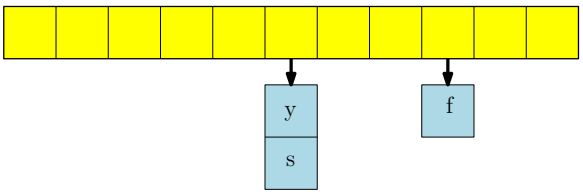


Figure 8.1: Open hashing.

Unfortunately, *collisions* are unavoidable, and several different techniques to handle them. Formally, two items $x \neq y$ ***collide*** if $h(x) = h(y)$.

A standard technique to handle collisions is to use ***chaining*** (aka *open hashing*). Here, we handle collisions as follows:

(A) For each slot i store all items hashed to slot i in a linked list. $T[i]$ points to the linked list.

(B) **Lookup**: to find if $y \in \mathcal{U}$ is in T, check the linked list at $T[h(y)]$. Time proportion to size of linked list.

Other techniques for handling collisions include associating a list of locations where an element can be (in certain order), and check these locations in this order. Another useful technique is ***cuckoo hashing*** which we will discuss later on: Every value has two possible locations. When inserting, insert in one of the locations, otherwise, kick out the stored value to its other location. Repeat till stable. if no stability then rebuild table.

The relevant questions when designing a hashing scheme, include: (I) Does hashing give $O(1)$ time per operation for dictionaries? (II) Complexity of evaluating h on a given element? (III) Relative sizes of the universe $\mathcal{U}$ and the set to be stored S. (IV) Size of table relative to size of S. (V) Worst-case vs average-case vs randomized (expected) time? (VI) How do we choose h?

The ***load factor*** of the array T is the ratio n/t where $n = |S|$ is the number of elements being stored and $m = |T|$ is the size of the array being used. Typically n/t is a small constant smaller than 1.

In the following, we assume that $\mathcal{U}$ (the universe the keys are taken from) is large – specifically, $N = |\mathcal{U}| \gg m^2$, where m is the size of the table. Consider a hash function $h : \mathcal{U} \to \{0, \ldots, m-1\}$. If hash N items to the m slots, then by the pigeon hole principle, there is some $i \in \{0, \ldots, m-1\}$ such that $N/m \geq m$ elements of $\mathcal{U}$ get hashed to i. In particular, this implies that there is set $S \subseteq \mathcal{U}$, where $|S| = m$ such that all of S hashes to same slot. Oops.

Namely, for every hash function there is a *bad set* with many collisions.

Observation 8.1.1. *Let $\mathcal{H}$ be the set of all functions from $\mathcal{U} = \{1, \ldots, U\}$ to $\{1, \ldots, m\}$. The number of functions in $\mathcal{H}$ is m^U. As such, specifying a function in $\mathcal{H}$ would require $\log_2 |\mathcal{H}| = O(U \log m)$.*

As such, picking a truely random hash function requires many random bits, and furthermore, it is not even clear how to evaluate it efficiently (which is the whole point of hashing).

Picking a hash function. Picking a good hash function in practice is a dark art involving many non-trivial considerations and ideas. For parameters $N = |\mathcal{U}|$, $m = |T|$, and $n = |S|$, we require the following:

(A) $\mathcal{H}$ is a ***family*** of hash functions: each function $h \in \mathcal{H}$ should be efficient to evaluate (that is, to compute $h(x)$).

(B) h is chosen *randomly* from $\mathcal{H}$ (typically uniformly at random). Implicitly assumes that $\mathcal{H}$ allows an efficient sampling.

(C) Require that for any *fixed* set $S \subseteq \mathcal{U}$, of size m, the expected number of collisions for a function chosen from $\mathcal{H}$ should be "small". Here the expectation is over the randomness in choice of h.

8.2. Universal Hashing

We would like the hash function to have the following property – For any element $x \in \mathcal{U}$, and a random $h \in \mathcal{H}$, then $h(x)$ should have a uniform distribution. That is $\Pr[h(x) = i] = 1/m$, for every $0 \le i < m$. A somewhat stronger property is that for any two distinct elements $x, y \in \mathcal{U}$, for a random $h \in \mathcal{H}$, the probability of a collision between x and y should be at most $1/m$. $\mathbb{P}[h(x) = h(y)] = 1/m$.

Definition 8.2.1. A family $\mathcal{H}$ of hash functions is ***2-universal*** if for all distinct $x, y \in \mathcal{U}$, we have $\mathbb{P}[h(x) = h(y)] \le 1/m$.

Applying a 2-universal family hash function to a set of distinct numbers, results in a 2-wise independent sequence of numbers.

Lemma 8.2.2. *Let S be a set of n elements stored using open hashing in a hash table of size m, using open hashing, where the hash function is picked from a 2-universal family. Then, the expected lookup time, for any element $x \in \mathcal{U}$ is $O(n/m)$.*

Proof: The number of elements colliding with x is $\ell(x) = \sum_{y \in S} D_y$, where $D_y = 1 \iff x$ and y collide under the hash function h. As such, we have

$$\mathbb{E}[\ell(x)] = \sum_{y \in S} \mathbb{E}\Big[D_y\Big] = \sum_{y \in S} \mathbb{P}[h(x) = h(y)] = \sum_{y \in S} \frac{1}{m} = |S|/m = n/m. \qquad \blacksquare$$

Remark 8.2.3. The above analysis holds even if we perform a sequence of $O(n)$ insertions/deletions operations. Indeed, just repeat the analysis with the set of elements being all elements encountered during these operations.

The worst-case bound is of course much worse – it is not hard to show that in the worst case, the load of a single hash table entry might be $\Omega(\log n / \log \log n)$ (as we seen in the occupancy problem).

Rehashing, amortization, etc. The above assumed that the set S is fixed. If items are inserted and deleted, then the hash table might become much worse. In particular, $|S|$ grows to more than cm, for some constant c, then hash table performance start degrading. Furthermore, if many insertions and deletions happen then the initial random hash function is no longer random enough, and the above analysis no longer holds.

A standard solution is to rebuild the hash table periodically. We choose a new table size based on current number of elements in table, and a new random hash function, and rehash the elements. And then discard the old table and hash function. In particular, if $|S|$ grows to more than twice current table size, then rebuild new hash table (choose a new random hash function) with double the current number of elements. One can do a similar shrinking operation if the set size falls below quarter the current hash table size.

If the working $|S|$ stays roughly the same but more than $c|S|$ operations on table for some chosen constant c (say 10), rebuild.

The ***amortize*** cost of rebuilding to previously performed operations. Rebuilding ensures $O(1)$ expected analysis holds even when S changes. Hence $O(1)$ expected look up/insert/delete time *dynamic* data dictionary data structure!

8.2.1. How to build a 2-universal family

8.2.1.1. A quick reminder on working modulo prime

Definition 8.2.4. For a number p, let $\mathbb{Z}_n = \{0, \ldots, n-1\}$.

For two integer numbers x and y, the ***quotient*** of x/y is $x \operatorname{div} y = \lfloor x/y \rfloor$. The ***remainder*** of x/y is $x \bmod y = x - y\lfloor x/y \rfloor$. If the $x \bmod y = 0$, than y ***divides*** x, denoted by $y \mid x$. We use $\alpha \equiv \beta \pmod{p}$ or $\alpha \equiv_p \beta$ to denote that α and β are ***congruent modulo*** p; that is $\alpha \bmod p = \beta \bmod p$ – equivalently, $p \mid (\alpha - \beta)$.

Remark 8.2.5. A quick review of what we already know. Let p be a prime number.

(A) Lemma 7.2.3: For any $\alpha, \beta \in \{1, \ldots, p-1\}$, we have that $\alpha\beta \not\equiv 0 \pmod{p}$.
(B) Lemma 7.2.3: For any $\alpha, \beta, i \in \{1, \ldots, p-1\}$, such that $\alpha \neq \beta$, we have that $\alpha i \not\equiv \beta i \pmod{p}$.
(C) Lemma 7.2.3: For any $x \in \{1, \ldots, p-1\}$ there exists a unique y such that $xy \equiv 1 \pmod{p}$. The number y is the ***inverse*** of x, and is denoted by x^{-1} or $1/x$.
(D) Lemma 7.2.4: For any numbers $x, y \in \mathbb{Z}_p$. If $x \neq y$ then, for any $a, b \in \mathbb{Z}_p$, such that $a \neq 0$, we have $ax + b \not\equiv ay + b \pmod{p}$.
(E) Lemma 7.2.5: For any numbers $x, y \in \mathbb{Z}_p$. If $x \neq y$ then, for each pair of numbers $r, s \in \mathbb{Z}_p = \{0, 1, \ldots, p-1\}$, such that $r \neq s$, there is exactly one unique choice of numbers $a, b \in \mathbb{Z}_p$ such that $ax + b \pmod{p} = r$ and $ay + b \pmod{p} = s$.

8.2.1.2. Constructing a family of 2-universal hash functions

For parameters $N = |\mathcal{U}|$, $m = |T|$, $n = |S|$. Choose a ***prime*** number $p \geq N$. Let

$$\mathcal{H} = \left\{ h_{a,b} \;\middle|\; a, b \in \mathbb{Z}_p \text{ and } a \neq 0 \right\},$$

where $h_{a,b}(x) = ((ax + b) \pmod{p}) \pmod{m}$. Note that $|\mathcal{H}| = p(p-1)$.

8.2.1.3. Analysis

Once we fix a and b, and we are given a value x, we compute the hash value of x in two stages:
(A) ***Compute***: $r \leftarrow (ax + b) \pmod{p}$.
(B) ***Fold***: $r' \leftarrow r \pmod{m}$

Lemma 8.2.6. *Assume that p is a prime, and $1 < m < p$. The number of pairs $(r, s) \in \mathbb{Z}_p \times \mathbb{Z}_p$, such that $r \neq s$, that are folded to the same number is $\leq p(p-1)/m$. Formally, the set of bad pairs*

$$B = \left\{ (r, s) \in \mathbb{Z}_p \times \mathbb{Z}_p \;\middle|\; r \equiv_m s \right\}$$

is of size at most $p(p-1)/m$.

Proof: Consider a pair $(x, y) \in \{0, 1, \ldots, p-1\}^2$, such that $x \neq y$. For a fixed x, there are at most $\lceil p/m \rceil$ values of y that fold into x. Indeed, $x \equiv_m y$ if and only if

$$y \in L(x) = \{x + im \;|\; i \text{ is an integer}\} \cap \mathbb{Z}_p.$$

The size of $L(x)$ is maximized when $x = 0$, The number of such elements is at most $\lceil p/m \rceil$ (note, that since p is a prime, p/m is fractional). One of the numbers in $O(x)$ is x itself. As such, we have that

$$|B| \leq p(|L(x)| - 1) \leq p(\lceil p/m \rceil - 1) \leq p(p-1)/m,$$

since $\lceil p/m \rceil - 1 \leq (p-1)/m \iff m\lceil p/m \rceil - m \leq p - 1 \iff m\lfloor p/m \rfloor \leq p - 1 \iff m\lfloor p/m \rfloor < p$, which is true since p is a prime, and $1 < m < p$. ■

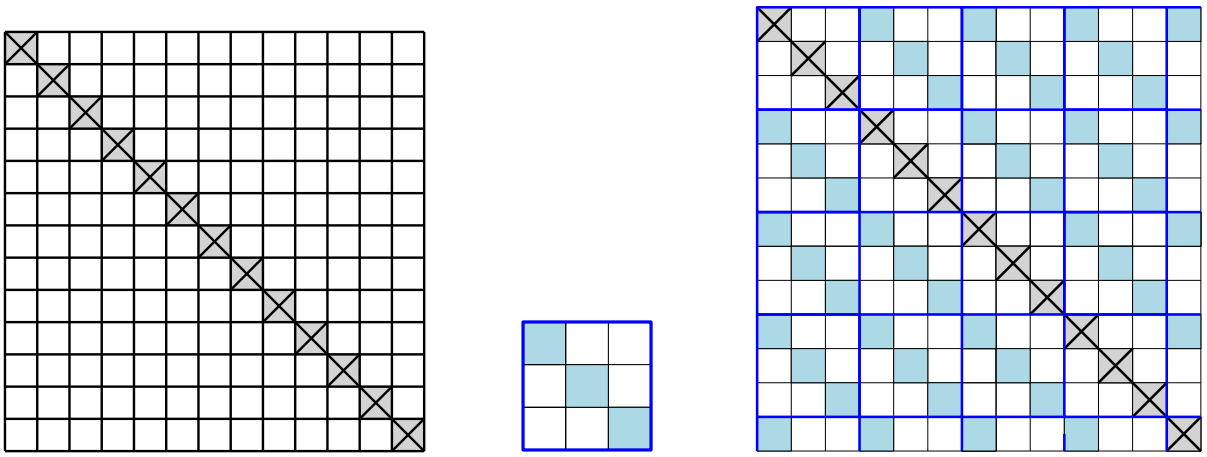


Figure 8.2: Explanation of the hashing scheme via figures.

Claim 8.2.7. *For two distinct numbers $x, y \in \mathcal{U}$, a pair a, b is* **bad** *if $h_{a,b}(x) = h_{a,b}(y)$. The number of bad pairs is $\leq p(p-1)/m$.*

Proof: Let $a, b \in \mathbb{Z}_p$ such that $a \neq 0$ and $h_{a,b}(x) = h_{a,b}(y)$. Let

$$r = (ax + b) \bmod p \qquad \text{and} \qquad s = (ay + b) \bmod p.$$

By Lemma 7.2.4, we have that $r \neq s$. As such, a collision happens if $r \equiv s \pmod m$. By Lemma 8.2.6, the number of such pairs (r, s) is at most $p(p-1)/m$. By Lemma 7.2.5, for each such pair (r, s), there is a unique choice of a, b that maps x and y to r and s, respectively. As such, there are at most $p(p-1)/m$ bad pairs. ■

Theorem 8.2.8. *The hash family $\mathcal{H}$ is a 2-universal hash family.*

Proof: Fix two distinct numbers $x, y \in \mathcal{U}$. We are interested in the probability they collide if h is picked randomly from $\mathcal{H}$. By Claim 8.2.7 there are $M \leq p(p-1)/m$ bad pairs that causes such a collision, and since $\mathcal{H}$ contains $N = p(p-1)$ functions, it follows the probability for collision is $M/N \leq 1/m$, which implies that $\mathcal{H}$ is 2-universal. ■

8.2.1.4. Explanation via pictures

Consider a pair $(x, y) \in \mathbb{Z}_p^2$, such that $x \neq y$. This pair (x, y) corresponds to a cell in the natural "grid" $\mathbb{Z}_p^2$ that is off the main diagonal. See Figure 8.2

The mapping $f_{a,b}(x) = (ax + b) \bmod p$, takes the pair (x, y), and maps it randomly and uniformly, to some other pair $x' = f_{a,b}(x)$ and $y' = f_{a,b}(y)$ (where x', y' are again off the main diagonal).

Now consider the smaller grid $\mathbb{Z}_m \times \mathbb{Z}_m$. The main diagonal of this subgrid is bad – it corresponds to a collision. One can think about the last step, of computing $h_{a,b}(x) = f_{a,b}(x) \bmod m$, as tiling the larger grid, by the smaller grid. in the natural way. Any diagonal that is in distance mi from the main diagonal get marked as bad. At most $1/m$ fraction of the off diagonal cells get marked as bad. See Figure 8.2.

As such, the random mapping of (x, y) to (x', y') causes a collision only if we map the pair to a badly marked pair, and the probability for that $\leq 1/m$.

8.3. Perfect hashing

An interesting special case of hashing is the static case – given a set S of elements, we want to hash S so that we can answer membership queries efficiently (i.e., dictionary data-structures with no insertions). it is easy to come up with a hashing scheme that is optimal as far as space.

8.3.1. Some easy calculations

The first observation is that if the hash table is quadraticly large, then there is a good (constant) probability to have no collisions (this is also the threshold for the birthday paradox).

Lemma 8.3.1. *Let $S \subseteq \mathcal{U}$ be a set of n elements, and let $\mathcal{H}$ be a 2-universal family of hash functions, into a table of size $m \geq n^2$. Then with probability $\leq 1/2$, there is a pair of elements of S that collide under a random hash function $h \in \mathcal{H}$.*

Proof: For a pair $x, y \in S$, the probability they collide is at most $\leq 1/m$, by definition. As such, by the union bound, the probability of any collusion is $\binom{n}{2}/m = n(n-1)/2m \leq 1/2$. ■

We now need a second moment bound on the sizes of the buckets.

Lemma 8.3.2. *Let $S \subseteq \mathcal{U}$ be a set of n elements, and let $\mathcal{H}$ be a 2-universal family of hash functions, into a table of size $m \geq cn$, where c is an arbitrary constant. Let $h \in \mathcal{H}$ be a random hash function, and let X_i be the number of elements of S mapped to the ith bucket by h, for $i = 0, \ldots, m-1$. Then, we have $\mathbb{E}\left[\sum_{j=0}^{m-1} X_j^2\right] \leq (1 + 1/c)n$.*

Proof: Let $s_1, \ldots, s_n$ be the n items in S, and let $Z_{i,j} = 1$ if $h(s_i) = h(s_j)$, for $i < j$. Observe that $\mathbb{E}\left[Z_{i,j}\right] = \mathbb{P}\left[h(s_i) = h(s_j)\right] \leq 1/m$ (this is the only place we use the property that $\mathcal{H}$ is 2-universal). In particular, let $\mathcal{Z}(\alpha)$ be all the variables $Z_{i,j}$, for $i < j$, such that $Z_{i,j} = 1$ and $h(s_i) = h(s_j) = \alpha$.

If for some α we have that $X_\alpha = k$, then there are k indices $\ell_1 < \ell_2 < \ldots < \ell_k$, such that $h(s_{\ell_1}) = \cdots = h(s_{\ell_k}) = i$. As such, $z(\alpha) = |\mathcal{Z}(\alpha)| = \binom{k}{2}$. In particular, we have

$$X_\alpha^2 = k^2 = 2\binom{k}{2} + k = 2z(\alpha) + X_\alpha$$

This implies that

$$\sum_{\alpha=0}^{m-1} X_\alpha^2 = \sum_{\alpha=0}^{m-1} (2z(\alpha) + X_\alpha) = 2\sum_{\alpha=0}^{m-1} z(\alpha) + \sum_{\alpha=0}^{m-1} X_\alpha = n + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} Z_{ij}$$

Now, by linearity of expectations, we have

$$\begin{aligned}\mathbb{E}\left[\sum_{\alpha=0}^{m-1} X_\alpha^2\right] &= \mathbb{E}\left[n + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} Z_{ij}\right] = n + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \mathbb{E}\left[Z_{ij}\right] \leq n + 2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{1}{m} \\ &= n + \frac{2}{m}\binom{n}{2} = n + \frac{2n(n-1)}{2m} \leq n\left(1 + \frac{n-1}{m}\right) \leq n\left(1 + \frac{1}{c}\right)\end{aligned}$$

since $m \geq cn$. ■

8.3.2. Construction of perfect hashing

Given a set S of n elements, we build a open hash table T of size, say, $2n$. We use a random hash function h that is 2-universal for this hash table, see Theorem 8.2.8. Next, we map the elements of S into the hash table. Let S_j be the list of all the elements of S mapped to the jth bucket, and let $X_j = \left|L_j\right|$, for $j = 0, \ldots, n-1$.

We compute $Y = \sum_{i=1} X_j^2$. If $Y > 6n$, then we reject h, and resample a hash function h. We repeat this process till success.

In the second stage, we build secondary hash tables for each bucket. Specifically, for $j = 0, \ldots, 2n-1$, if the jth bucket contains $X_j > 0$ elements, then we construct a secondary hash table H_j to store the elements of S_j, and this secondary hash table has size X_j^2, and again we use a random 2-universal hash function h_j for the hashing of S_j into H_j. If any pair of elements of S_j collide under h_j, then we resample the hash function h_j, and try again till success.

8.3.2.1. Analysis

Theorem 8.3.3. *Given a (static) set $S \subseteq \mathcal{U}$ of n elements, the above scheme, constructs, in expected linear time, a two level hash-table that can perform search queries in $O(1)$ time. The resulting data-structure uses $O(n)$ space.*

Proof: Given an element $x \in \mathcal{U}$, we first compute $j = h(x)$, and then $k = h_j(x)$, and we can check whether the element stored in the secondary hash table H_j at the entry k is indeed x. As such, the search time is $O(1)$.

The more interesting issue is the construction time. Let X_j be the number of elements mapped to the jth bucket, and let $Y = \sum_{i=1}^{n} X_i^2$. Observe, that $\mathbb{E}[Y] \leq (1 + 1/2)n = (3/2)n$, by Lemma 8.3.2 (here, $m = 2n$ and as such $c = 2$). As such, by Markov's inequality, $\mathbb{P}[X > 6n] = \frac{(3/2)n}{6n} \leq 1/4$. In particular, picking a good top level hash function requires in expectation at most $1/(3/4) = 4/3 \leq 2$ iterations. Thus the first stage takes $O(n)$ time, in expectation.

For the jth bucket, with X_j entries, by Lemma 8.3.1, the construction succeeds with probability $\geq 1/2$. As before, the expected number of iterations till success is at most 2. As such, the expected construction time of the secondary hash table for the jth bucket is $O(X_j^2)$.

We conclude that the overall expected construction time is $O(n + \sum_j X_j^2) = O(n)$.

As for the space used, observe that it is $O(n + \sum_j X_j^2) = O(n)$. ■

8.4. Bloom filters

Consider an application where we have a set $S \subseteq \mathcal{U}$ of n elements, and we want to be able to decide for a query $x \in \mathcal{U}$, whether or not $x \in S$. Naturally, we can use hashing. However, here we are interested in more efficient data-structure as far as space. We allow the data-structure to make a mistake (i.e., say that an element is in, when it is not in).

First try. So, let start silly. Let $B[0 \ldots, m]$ be an array of bits, and pick a random hash function $h : \mathcal{U} \to \mathbb{Z}_m$. Initialize B to 0. Next, for every element $s \in S$, set $B[h(s)]$ to 1. Now, given a query, return $B[h(x)]$ as an answer whether or not $x \in S$. Note, that B is an array of bits, and as such it can be bit-packed and stored efficiently.

For the sake of simplicity of exposition, assume that the hash functions picked is truly random. As such, we have that the probability for a false positive (i.e., a mistake) for a fixed $x \in \mathcal{U}$ is n/m. Since we want the size of the table m to be close to n, this is not satisfying.

Using k hash functions. Instead of using a single hash function, let us use k independent hash functions $h_1, \ldots h_k$. For an element $s \in S$, we set $B[h_i(s)]$ to 1, for $i = 1, \ldots, k$. Given an query $x \in \mathcal{U}$, if $B[h_i(x)]$ is zero, for any $i = 1, \ldots, k$, then $x \notin S$. Otherwise, if all these k bits are on, the data-structure returns that x is in S.

Clearly, if the data-structure returns that x is not in S, then it is correct. The data-structure might make a mistake (i.e., a false positive), if it returns that x is in S (when is not in S).

We interpret the storing of the elements of S in B, as an experiment of throwing kn balls into m bins. The probability of a bin to be empty is

$$p = p(m, n) = (1 - 1/m)^{kn} \approx \exp(-k(n/m)).$$

Since the number of empty bins is a martingale, we know the number of empty bins is strongly concentrated around the expectation pm, and we can treat p as the true probability of a bin to be empty.

The probability of a mistake is

$$f(k, m, n) = (1 - p)^k.$$

In particular, for $k = (m/n) \ln n$, we have that $p = p(m, n) \approx 1/2$, and $f(k, m, n) \approx 1/2^{(m/n) \ln 2} \approx 0.618^{m/n}$.

Example 8.4.1. Of course, the above is fictional, as k has to be an integer. But motivated by these calculations, let $m = 3n$, and $k = 4$. We get that $p(m, n) = \exp(-4/3) \approx 0.26359$, and $f(4, 3n, n) \approx (1 - 0.265)^4 \approx 0.294078$. This is better than the naive $k = 1$ scheme, where the probability of false positive is $1/3$.

Note, that this scheme gets exponentially better over the naive scheme as m/n grows.

Example 8.4.2. Consider the setting $m = 8n$ – this is when we allocate a byte for each element stored (the element of course might be significantly bigger). The above implies we should take $k = \lceil (m/n) \ln 2 \rceil = 6$. We then get $p(8n, n) = \exp(-6/8) \approx 0.5352$, and $f(6, 8n, n) \approx 0.0215$. Here, the naive scheme with $k = 1$, would give probability of false positive of $1/8 = 0.125$. So this is a significant improvement.

Remark 8.4.3. It is important to remember that Bloom filters are competing with direct hashing of the whole elements. Even if one allocates 8 bits per item, as in the example above, the space it uses is significantly smaller than regular hashing. A situation when such a Bloom filter makes sense is for a cache – we might want to decide if an element is in a slow external cache (say SSD drive). Retrieving item from the cache is slow, but not so slow we are not willing to have a small overhead because of false positives.

8.5. Bibliographical notes

Practical Issues Hashing used typically for integers, vectors, strings etc.

- Universal hashing is defined for integers. To implement it for other objects, one needs to map objects in some fashion to integers.
- Practical methods for various important cases such as vectors, strings are studied extensively. See http://en.wikipedia.org/wiki/Universal_hashing for some pointers.
- Recent important paper bridging theory and practice of hashing. "The power of simple tabulation hashing" by Mikkel Thorup and Mihai Patrascu, 2011. See http://en.wikipedia.org/wiki/Tabulation_hashing

References

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

Chapter 9

Closest Pair

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

The events of September 8 prompted Foch to draft the later legendary signal: "My centre is giving way, my right is in retreat, situation excellent. I attack." It was probably never sent.

John Keegan, The first world war

9.1. How many times can a minimum change?

Let $a_1, \ldots, a_n$ be a set of n numbers, and let us randomly permute them into the sequence $b_1, \ldots, b_n$. Next, let $c_i = \min_{k=1}^{i} b_i$, and let X be the random variable which is the number of distinct values that appears in the sequence $c_1, \ldots, c_n$. What is the expectation of X?

Lemma 9.1.1. *In expectation, the number of times the minimum of a prefix of n randomly permuted numbers change, is $O(\log n)$. That is $\mathbb{E}[X] = O(\log n)$.*

Proof: Consider the indicator variable X_i, such that $X_i = 1$ if $c_i \neq c_{i-1}$. The probability for that is $\leq 1/i$, since this is the probability that the smallest number of $b_1, \ldots, b_i$ is b_i. (Why is this probability not simply equal to $1/i$?) As such, we have $X = \sum_i X_i$, and $\mathbb{E}[X] = \sum_i \mathbb{E}[X_i] = \sum_{i=1}^{n} \frac{1}{i} = O(\log n)$. ■

9.2. Closest Pair

Assumption 9.2.1. Throughout the discourse, we are going to assume that every hashing operation takes (worst case) constant time. This is quite a reasonable assumption when true randomness is available (using for example perfect hashing [CLRS01]). We will revisit this issue later in the course.

For a real positive number r and a point $\mathsf{p} = (x, y)$ in $\mathbb{R}^2$, define

$$\mathsf{G}_r(\mathsf{p}) := \left(\left\lfloor \frac{x}{r} \right\rfloor r \ , \ \left\lfloor \frac{y}{r} \right\rfloor r \right) \in \mathbb{R}^2.$$

The number r is the ***width*** of the ***grid*** G_r. Observe that G_r partitions the plane into square regions, which are *grid cells*. Formally, for any $i, j \in \mathbb{Z}$, the intersection of the half-planes $x \geq ri$, $x < r(i+1)$, $y \geq rj$ and $y < r(j+1)$ is a ***grid cell***. Further a ***grid cluster*** is a block of 3×3 contiguous grid cells.

For a point set P, and a parameter r, the partition of P into subsets by the grid G_r, is denoted by $\mathsf{G}_r(\mathsf{P})$. More formally, two points $\mathsf{p}, \mathsf{u} \in \mathsf{P}$ belong to the same set in the partition $\mathsf{G}_r(\mathsf{P})$, if both points are being mapped to the same grid point or equivalently belong to the same grid cell.

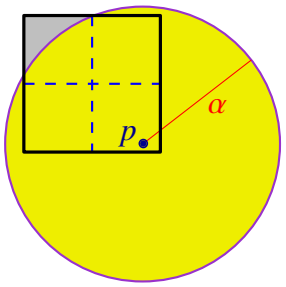
Note, that every grid cell C of G_r, has a unique ID; indeed, let $\mathsf{p} = (x, y)$ be any point in C, and consider the pair of integer numbers $\mathrm{id}_C = \mathrm{id}(\mathsf{p}) = (\lfloor x/r \rfloor, \lfloor y/r \rfloor)$. Clearly, only points inside C are going to be mapped to id_C. This is useful, as one can store a set P of points inside a grid efficiently. Indeed, given a point p, compute its $\mathrm{id}(\mathsf{p})$. We associate with each unique id a data-structure that stores all the points falling into this grid cell (of course, we do not maintain such data-structures for grid cells which are empty). For our purposes here, the grid-cell data-structure can simply be a linked list of points. So, once we computed $\mathrm{id}(\mathsf{p})$, we fetch the data structure for this cell, by using hashing. Namely, we store pointers to all those data-structures in a hash table, where each such data-structure is indexed by its unique id. Since the ids are integer numbers, we can do the hashing in constant time.

We are interested in solving the following problem.

Problem 9.2.2. Given a set P of n points in the plane, find the pair of points closest to each other. Formally, return the pair of points realizing $\mathcal{CP}(\mathsf{P}) = \min_{\mathsf{p},\mathsf{u} \in P} \|\mathsf{p} - \mathsf{u}\|$.

We need the following easy packing lemma.

Lemma 9.2.3. *Let P be a set of points contained inside a square $\square$, such that the sidelength of $\square$ is $\alpha = \mathcal{CP}(\mathsf{P})$. Then $|\mathsf{P}| \leq 4$.*



Proof: Partition $\square$ into four equal squares $\square_1, \ldots, \square_4$, and observe that each of these squares has diameter $\sqrt{2}\alpha/2 < \alpha$, and as such each can contain at most one point of P; that is, the disk of radius α centered at a point $\mathsf{p} \in \mathsf{P}$ completely covers the subsquare containing it; see the figure on the right.

Note that the set P can have four points if it is the four corners of $\square$. ■

Lemma 9.2.4. *Given a set P of n points in the plane, and a distance r, one can verify in linear time, whether or not $\mathcal{CP}(\mathsf{P}) < r$ or $\mathcal{CP}(\mathsf{P}) \geq r$.*

Proof: Indeed, store the points of P in the grid G_r. For every non-empty grid cell, we maintain a linked list of the points inside it. Thus, adding a new point p takes constant time. Indeed, compute $\mathrm{id}(p)$, check if $\mathrm{id}(p)$ already appears in the hash table, if not, create a new linked list for the cell with this ID number, and store p in it. If a data-structure already exist for $\mathrm{id}(p)$, just add p to it.

This takes $O(n)$ time. Now, if any grid cell in $\mathrm{G}_r(\mathsf{P})$ contains more than four points of P, then, by Lemma 9.2.3, it must be that the $\mathcal{CP}(\mathsf{P}) < r$.

Thus, when inserting a point p, the algorithm fetch all the points of P that were already inserted, for the cell of p, and the 8 adjacent cells. All those cells must contain at most 4 points of P (otherwise, we would already have stopped since the $\mathcal{CP}(\cdot)$ of the inserted points is smaller than r). Let S be the set of all those points, and observe that $|S| \leq 4 \cdot 9 = O(1)$. Thus, we can compute by brute force the closest point to p in S. This takes $O(1)$ time. If $\mathsf{d}(p, S) < r$, we stop and return this distance (together with the two points realizing $\mathsf{d}(p, S)$ as a proof that the distance is too short). Otherwise, we continue to the next point, where $\mathsf{d}(p, S) = \min_{s \in S} \|p - s\|$.

Overall, this takes $O(n)$ time. As for correctness, first observe that if $\mathcal{CP}(\mathsf{P}) > r$ then the algorithm would never make a mistake, since it returns '$\mathcal{CP}(\mathsf{P}) < r$' only after finding a pair of points of P with distance smaller than r. Thus, assume that p, q are the pair of points of P realizing the closest pair, and $\|p - q\| = \mathcal{CP}(\mathsf{P}) < r$. Clearly, when the later of them, say p, is being inserted, the set S would contain q, and as such the algorithm would stop and return "$\mathcal{CP}(\mathsf{P}) < r$". ■

Lemma 9.2.4 hints to a natural way to compute $\mathcal{CP}(\mathsf{P})$. Indeed, permute the points of P, in an arbitrary fashion, and let $P = \langle p_1, \ldots, p_n \rangle$. Next, let $r_i = \mathcal{CP}(\{p_1, \ldots, p_i\})$. We can check if $r_{i+1} < r_i$, by just calling the

algorithm for Lemma 9.2.4 on P_{i+1} and r_i. If $r_{i+1} < r_i$, the algorithm of Lemma 9.2.4, would give us back the distance r_{i+1} (with the other point realizing this distance).

So, consider the "good" case where $r_{i+1} = r_i = r_{i-1}$. Namely, the length of the shortest pair does not change. In this case we do not need to rebuild the data structure of Lemma 9.2.4 for each point. We can just reuse it from the previous iteration. Thus, inserting a single point takes constant time as long as the closest pair (distance) does not change.

Things become bad, when $r_i < r_{i-1}$. Because then we need to rebuild the grid, and reinsert all the points of $\mathsf{P}_i = \langle p_1, \ldots, p_i \rangle$ into the new grid $\mathsf{G}_{r_i}(\mathsf{P}_i)$. This takes $O(i)$ time.

So, if the closest pair radius, in the sequence $r_1, \ldots, r_n$, changes only k times, then the running time of the algorithm would be $O(nk)$. But we can do even better!

Theorem 9.2.5. *Let P be a set of n points in the plane. One can compute the closest pair of points of P in expected linear time.*

Proof: Pick a random permutation of the points of P, and let $\langle p_1, \ldots, p_n \rangle$ be this permutation. Let $r_2 = \|p_1 - p_2\|$, and start inserting the points into the data structure of Lemma 9.2.4. In the ith iteration, if $r_i = r_{i-1}$, then this insertion takes constant time. If $r_i < r_{i-1}$, then we rebuild the grid and reinsert the points. Namely, we recompute $\mathsf{G}_{r_i}(\mathsf{P}_i)$.

To analyze the running time of this algorithm, let X_i be the indicator variable which is 1 if $r_i \neq r_{i-1}$, and 0 otherwise. Clearly, the running time is proportional to

$$R = 1 + \sum_{i=2}^{n} (1 + X_i \cdot i).$$

Thus, the expected running time is

$$\mathbb{E}[R] = 1 + \mathbb{E}\Big[1 + \sum\nolimits_{i=2}^{n} (1 + X_i \cdot i)\Big] = n + \sum_{i=2}^{n} (\mathbb{E}[X_i] \cdot i) = n + \sum_{i=2}^{n} i \cdot \mathbb{P}[X_1 = 1],$$

by linearity of expectation and since for an indicator variable X_i, we have that $\mathbb{E}[X_i] = \mathbb{P}[X_i = 1]$.

Thus, we need to bound $\mathbb{P}[X_i = 1] = \mathbb{P}[r_i < r_{i-1}]$. To bound this quantity, fix the points of P_i, and randomly permute them. A point $\mathsf{u} \in \mathsf{P}_i$ is ***critical*** if $\mathcal{CP}(\mathsf{P}_i \setminus \{\mathsf{u}\}) > \mathcal{CP}(\mathsf{P}_i)$.

(A) If there are no critical points, then $r_{i-1} = r_i$ and then $\mathbb{P}[X_i = 1] = 0$.

(B) If there is one critical point, than $\mathbb{P}[X_i = 1] = 1/i$, as this is the probability that this critical point would be the last point in a random permutation of P_i.

(C) If there are two critical points, and let p, u be this unique pair of points of P_i realizing $\mathcal{CP}(\mathsf{P}_i)$. The quantity r_i is smaller than r_{i-1}, if either p or u are p_i. But the probability for that is $2/i$ (i.e., the probability in a random permutation of i objects, that one of two marked objects would be the last element in the permutation).

Observe, that there can not be more than two critical points. Indeed, if p and u are two points that realize the closest distance, than if there is a third critical point v, then $\mathcal{CP}(\mathsf{P}_i \setminus \{\mathsf{v}\}) = \|\mathsf{p} - \mathsf{u}\|$, and v is not critical.

We conclude that

$$\mathbb{E}[R] = n + \sum_{i=2}^{n} i \cdot \mathbb{P}[X_1 = 1] \leq n + \sum_{i=2}^{n} i \cdot \frac{2}{i} \leq 3n.$$

As such, the expected running time of this algorithm is $O(\mathbb{E}[R]) = O(n)$. ■

Theorem 9.2.5 is a surprising result, since it implies that ***uniqueness*** (i.e., deciding if n real numbers are all distinct) can be solved in linear time. However, there is a lower bound of $\Omega(n \log n)$ on uniqueness, using the comparison tree model. This reality dysfunction, can be easily explained, once one realizes that the model of computation of Theorem 9.2.5 is considerably stronger, using hashing, randomization, and the floor function.

9.3. Bibliographical notes

The closest-pair algorithm follows Golin *et al.* [GRSS95]. This is in turn a simplification of a result of the celebrated result of Rabin [Rab76]. Smid provides a survey of such algorithms [Smi00]. A generalization of the closest pair algorithm was provided by Har-Peled and Raichel [HR15].

Surprisingly, Schönhage [Sch79] showed that assuming that the floor function is allowed, and the standard arithmetic operation can be done in constant time, then every problem in PSPACE can be solved in polynomial time. Since PSPACE includes NPC, this is bad news, as it implies that one can solve NPC problem in polynomial time (finally!). The basic idea is that one can pack huge number of bits into a single number, and the floor function enables one to read a single bit of this number. As such, a real RAM model that allows certain operations, and put no limit on the bit complexity of numbers, and assume that each operation can take constant time, is not a reasonable model of computation (but we already knew that).

References

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press / McGraw-Hill, 2001.

[GRSS95] M. Golin, R. Raman, C. Schwarz, and M. Smid. Simple randomized algorithms for closest pair problems. *Nordic J. Comput.*, 2: 3–27, 1995.

[HR15] S. Har-Peled and B. Raichel. Net and prune: A linear time algorithm for Euclidean distance problems. *J. Assoc. Comput. Mach.*, 62(6): 44:1–44:35, 2015.

[Rab76] M. O. Rabin. Probabilistic algorithms. *Algorithms and Complexity: New Directions and Recent Results*. Ed. by J. F. Traub. Orlando, FL, USA: Academic Press, 1976, pp. 21–39.

[Sch79] A. Schönhage. On the power of random access machines. *Proc. 6th Int. Colloq. Automata Lang. Prog.* (ICALP), vol. 71. 520–529, 1979.

[Smi00] M. Smid. Closest-point problems in computational geometry. *Handbook of Computational Geometry*. Ed. by J.-R. Sack and J. Urrutia. Amsterdam, The Netherlands: Elsevier, 2000, pp. 877–935.

Chapter 10

Coupon's Collector Problems II

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> There is not much talking now. A silence falls upon them all. This is no time to talk of hedges and fields, or the beauties of any country. Sadness and fear and hate, how they well up in the heart and mind, whenever one opens the pages of these messengers of doom. Cry for the broken tribe, for the law and custom that is gone. Aye, and cry aloud for the man who is dead, for the woman and children bereaved. Cry, the beloved country, these things are not yet at an end. The sun pours down on the earth, on the lovely land that man cannot enjoy. He knows only the fear of his heart.
>
> Alan Paton, Cry, the beloved country

10.1. The Coupon Collector's Problem Revisited

10.1.1. Some technical lemmas

Unfortunately, in Randomized Algorithms, many of the calculations are awful[①]. As such, one has to be dexterous in approximating such calculations. We present quickly a few of these estimates.

Lemma 10.1.1. *For $x \geq 0$, we have $1 - x \leq \exp(-x)$ and $1 + x \leq e^x$. Namely, for all x, we have $1 + x \leq e^x$.*

Proof: For $x = 0$ we have equality. Next, computing the derivative on both sides, we have that we need to prove that $-1 \leq -\exp(-x) \iff 1 \geq \exp(-x) \iff e^x \geq 1$, which clearly holds for $x \geq 0$.

A similar argument works for the second inequality. ■

Lemma 10.1.2. *For any $y \geq 1$, and $|x| \leq 1$, we have $\left(1 - x^2\right)^y \geq 1 - yx^2$.*

Proof: Observe that the inequality holds with equality for $x = 0$. So compute the derivative of x of both sides of the inequality. We need to prove that

$$y(-2x)\left(1 - x^2\right)^{y-1} \geq -2yx \iff \left(1 - x^2\right)^{y-1} \leq 1,$$

which holds since $1 - x^2 \leq 1$, and $y - 1 \geq 0$. ■

Lemma 10.1.3. *For any $y \geq 1$, and $|x| \leq 1$, we have $\left(1 - x^2 y\right)e^{xy} \leq (1 + x)^y \leq e^{xy}$.*

[①] "In space travel," repeated Slartibartfast, "all the numbers are awful." – Life, the Universe, and Everything Else, Douglas Adams.

Proof: The right side of the inequality is standard by now. As for the left side. Observe that

$$(1 - x^2)e^x \leq 1 + x,$$

since dividing both sides by $(1 + x)e^x$, we get $1 - x \leq e^{-x}$, which we know holds for any x. By Lemma 10.1.2, we have

$$\Big(1 - x^2 y\Big)e^{xy} \leq \Big(1 - x^2\Big)^y e^{xy} = \Big(\Big(1 - x^2\Big)e^x\Big)^y \leq \Big(1 + x\Big)^y \leq e^{xy}.$$ ■

10.1.2. Back to the coupon collector's problem

There are n types of coupons, and at each trial one coupon is picked in random. How many trials one has to perform before picking all coupons? Let m be the number of trials performed. We would like to bound the probability that m exceeds a certain number, and we still did not pick all coupons.

In the previous lecture, we showed that

$$\mathbb{P}\left[\#\text{ of trials} \geq n \log n + n + t \cdot n\frac{\pi}{\sqrt{6}}\right] \leq \frac{1}{t^2},$$

for any t.

A stronger bound, follows from the following observation. Let Z_i^r denote the event that the ith coupon was not picked in the first r trials. Clearly,

$$\mathbb{P}\Big[Z_i^r\Big] = \left(1 - \frac{1}{n}\right)^r \leq \exp\Big(-\frac{r}{n}\Big).$$

Thus, for $r = \beta n \log n$, we have $\mathbb{P}[Z_i^r] \leq \exp\Big(-\dfrac{\beta n \log n}{n}\Big) = n^{-\beta}$. Thus,

$$\mathbb{P}\Big[X > \beta n \log n\Big] \leq \mathbb{P}\Big[\bigcup_i Z_i^{\beta n \log n}\Big] \leq n \cdot \mathbb{P}[Z_1] \leq n^{-\beta+1}.$$

Lemma 10.1.4. *Let the random variable X denote the number of trials for collecting each of the n types of coupons. Then, we have $\mathbb{P}[X > n \ln n + cn] \leq e^{-c}$.*

Proof: The probability we fail to pick the first type of coupon is $\alpha = (1 - 1/n)^m \leq \exp\Big(-\frac{n \ln n + cn}{n}\Big) = \exp(-c)/n$. As such, using the union bound, the probability we fail to pick all n types of coupons is bounded by $n\alpha = \exp(-c)$, as claimed. ■

In the following, we show a slightly stronger bound on the probability, which is $1 - \exp(-e^{-c})$. To see that it is indeed stronger, observe that $e^{-c} \geq 1 - \exp(-e^{-c})$.

10.1.3. An asymptotically tight bound

Lemma 10.1.5. *Let $c > 0$ be a constant, $m = n \ln n + cn$ for a positive integer n. Then for any constant k, we have* $\displaystyle\lim_{n\to\infty} \binom{n}{k}\left(1 - \frac{k}{n}\right)^m = \frac{\exp(-ck)}{k!}.$

Proof: By Lemma 10.1.3, we have

$$\left(1-\frac{k^2m}{n^2}\right)\exp\left(-\frac{km}{n}\right) \leq \left(1-\frac{k}{n}\right)^m \leq \exp\left(-\frac{km}{n}\right).$$

Observe also that $\lim_{n\to\infty}\left(1-\frac{k^2m}{n^2}\right) = 1$, and $\exp\left(-\frac{km}{n}\right) = n^{-k}\exp(-ck)$. Also,

$$\lim_{n\to\infty}\binom{n}{k}\frac{k!}{n^k} = \lim_{n\to\infty}\frac{n(n-1)\cdots(n-k+1)}{n^k} = 1.$$

Thus, $\lim_{n\to\infty}\binom{n}{k}\left(1-\frac{k}{n}\right)^m = \lim_{n\to\infty}\frac{n^k}{k!}\exp\left(-\frac{km}{n}\right) = \lim_{n\to\infty}\frac{n^k}{k!}n^{-k}\exp(-ck) = \frac{\exp(-ck)}{k!}$. ■

Theorem 10.1.6. *Let the random variable X denote the number of trials for collecting each of the n types of coupons. Then, for any constant $c \in \mathbb{R}$, and $m = n\ln n + cn$, we have $\lim_{n\to\infty}\mathbb{P}[X > m] = 1 - \exp(-e^{-c})$.*

Before dwelling into the proof, observe that $1-\exp(-e^{-c}) \approx 1-(1-e^{-c}) = e^{-c}$. Namely, in the limit, the upper bound of Lemma 10.1.4 is tight.

Proof: We have $\mathbb{P}[X > m] = \mathbb{P}\left[\cup_i Z_i^m\right]$. By inclusion-exclusion, we have

$$\mathbb{P}\left[\bigcup_i Z_i^m\right] = \sum_{i=1}^{n}(-1)^{i+1}P_i^n,$$

where $P_j^n = \sum_{1\leq i_1<i_2<\ldots<i_j\leq n}\mathbb{P}\left[\cap_{v=1}^{j}Z_{i_v}^m\right]$. Let $S_k^n = \sum_{i=1}^{k}(-1)^{i+1}P_i^n$. We know that $S_{2k}^n \leq \mathbb{P}\left[\bigcup_i Z_i^m\right] \leq S_{2k+1}^n$.

By symmetry,

$$P_k^n = \binom{n}{k}\mathbb{P}\left[\bigcap_{v=1}^{k}Z_v^m\right] = \binom{n}{k}\left(1-\frac{k}{n}\right)^m,$$

Thus, $P_k = \lim_{n\to\infty}P_k^n = \exp(-ck)/k!$, by Lemma 10.1.5. Thus, we have

$$S_k = \sum_{j=1}^{k}(-1)^{j+1}P_j = \sum_{j=1}^{k}(-1)^{j+1}\cdot\frac{\exp(-cj)}{j!}.$$

Observe that $\lim_{k\to\infty}S_k = 1-\exp(-e^{-c})$ by the Taylor expansion of $\exp(x)$ (for $x = -e^{-c}$). Indeed,

$$\exp(x) = \sum_{j=0}^{\infty}\frac{x^j}{j!} = \sum_{j=0}^{\infty}\frac{(-e^{-c})^j}{j!} = 1+\sum_{j=1}^{\infty}\frac{(-1)^j\exp(-cj)}{j!}.$$

Clearly, $\lim_{n\to\infty}S_k^n = S_k$ and $\lim_{k\to\infty}S_k = 1-\exp(-e^{-c})$. Thus, (using fluffy math), we have

$$\lim_{n\to\infty}\mathbb{P}[X > m] = \lim_{n\to\infty}\mathbb{P}[\cup_{i=1}^{n}Z_i^m] = \lim_{n\to\infty}\lim_{k\to\infty}S_k^n = \lim_{k\to\infty}S_k = 1-\exp(-e^{-c}).$$ ■

10.2. Bibliographical notes

Are presentation follows, as usual, Motwani and Raghavan [MR95].

References

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

Chapter 11

Conditional Expectation and Concentration

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "You see, dogs aren't enough any more. People feel so damned lonely, they need company, they need something bigger, stronger, to lean on, something that can really stand up to it all. Dogs aren't enough, what we need is elephants..."
>
> The roots of heaven, Romain Gary

11.1. Conditional expectation

Definition 11.1.1. For two random variables X and Y, let $\mathbb{E}[X \mid Y]$ denote the expected value of X, if the value of Y is specified. Formally, we have

$$\mathbb{E}[X \mid Y = y] = \sum_{x \in \Omega} x \, \mathbb{P}[X = x \mid Y = y] .$$

The expression $\mathbb{E}[X \mid Y]$, which is a shorthand for $\mathbb{E}[X \mid Y = y]$, is the ***conditional expectation*** of X given Y.

As such, the conditional expectation is a function from the value of y, to the average value of X. As such, one can think of conditional expectation as a function $f(y) = \mathbb{E}[X \mid Y = y]$.

Lemma 11.1.2. *For any two random variables X and Y, we have $\mathbb{E}[\mathbb{E}[X \mid Y]] = \mathbb{E}[X]$.*

Proof: $\mathbb{E}[\mathbb{E}[X \mid Y]] = \mathbb{E}_Y[\mathbb{E}[X \mid Y = y]] = \sum_y \mathbb{P}[Y = y] \, \mathbb{E}[X \mid Y = y]$

$$\begin{aligned}
&= \sum_y \mathbb{P}[Y = y] \frac{\sum_x x \, \mathbb{P}[X = x \cap Y = y]}{\mathbb{P}[Y = y]} \\
&= \sum_y \sum_x x \, \mathbb{P}[X = x \cap Y = y] = \sum_x x \sum_y \mathbb{P}[X = x \cap Y = y] \\
&= \sum_x x \, \mathbb{P}[X = x] = \mathbb{E}[X] .
\end{aligned}$$

■

Lemma 11.1.3. *For any two random variables X and Y, we have $\mathbb{E}[Y \cdot \mathbb{E}[X \mid Y]] = \mathbb{E}[XY]$.*

Proof: We have that $\mathbb{E}\Big[Y \cdot \mathbb{E}\Big[X \,\Big|\, Y\Big]\Big] = \sum_y \mathbb{P}[Y = y] \cdot y \cdot \mathbb{E}\Big[X \,\Big|\, Y = y\Big]$

$$= \sum_y \mathbb{P}[Y = y] \cdot y \cdot \frac{\sum_x x \, \mathbb{P}[X = x \cap Y = y]}{\mathbb{P}[Y = y]} = \sum_x \sum_y xy \cdot \mathbb{P}[X = x \cap Y = y] = \mathbb{E}[XY] .$$

■

11.1.1. Concentration from conditional expectation

Lemma 11.1.4. *Let $X_1, \ldots, X_n$ be independent random variables, that with equal probability are* 0 *or* 1*. We have that $\mathbb{P}[\sum_i X_i < n/4] < 0.9^n$ and $\mathbb{P}[\sum_i X_i > (3/4)n] < 0.9^n$.*

Proof: Let $Y_0 = 1$. If $X_i = 1$, then we set $Y_i = Y_{i-1}$, and if $X_i = 0$, then we set $Y_i = Y_{i-1}/2$. We thus have that

$$\mathbb{E}[Y_i \mid Y_{i-1}] = \frac{1}{2}\frac{Y_{i-1}}{2} + \frac{1}{2}Y_{i-1} = \frac{3}{4}Y_{i-1}.$$

As such, by Lemma 11.1.2 we have

$$\mathbb{E}[Y_i] = \mathbb{E}[\mathbb{E}[Y_i \mid Y_{i-1}]] = \mathbb{E}\left[\frac{3}{4}Y_{i-1}\right] = \frac{3}{4}\mathbb{E}[Y_{i-1}] = \left(\frac{3}{4}\right)^i.$$

In particular, $\mathbb{E}[Y_n] = (3/4)^n$. Now, if $\sum_i X_i > (3/4)n$, then we have

$$Y_n \geq (1/2)^{n/4}.$$

We are now ready for our conclusions:

$$\mathbb{P}\Big[\sum\nolimits_i X_i > (3/4)n\Big] = \mathbb{P}\Big[Y_n \geq (1/2)^{n/4}\Big] \leq \frac{\mathbb{E}[Y_n]}{(1/2)^{n/4}} \leq \frac{(3/4)^n}{(1/2)^{n/4}} = \left(\frac{2^{1/4}3}{4}\right)^n \leq 0.9^n.$$

By symmetry, we have $\mathbb{P}[\sum_i X_i < (1/4)n] = \mathbb{P}[\sum_i X_i > (3/4)n] < 0.9^n$. ■

Chapter 12

Quick Sort with High Probability

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

12.1. QuickSort runs in $O(n \log n)$ time with high probability

Consider a set T of the n items to be sorted, and consider a specific element $t \in T$. Let X_i be the size of the input in the ith level of recursion that contains t. We know that $X_0 = n$, and

$$\mathbb{E}\Big[X_i \;\Big|\; X_{i-1}\Big] \leq \frac{1}{2}\frac{3}{4}X_{i-1} + \frac{1}{2}X_{i-1} \leq \frac{7}{8}X_{i-1}.$$

Indeed, with probability $1/2$ the pivot is the middle of the subproblem; that is, its rank is between $X_{i-1}/4$ and $(3/4)X_{i-1}$ (and then the subproblem has size $\leq X_{i-1}(3/4)$), and with probability $1/2$ the subproblem might has not shrank significantly (i.e., we pretend it did not shrink at all).

Now, observe that for any two random variables we have that $\mathbb{E}[X] = \mathbb{E}_y[\mathbb{E}[X \,|Y = y]]$, see Lemma 11.1.2_{p81}.. As such, we have that

$$\mathbb{E}[X_i] = \mathop{\mathbb{E}}_{y}\Big[\mathbb{E}\Big[X_i \;\Big|\; X_{i-1} = y\Big]\Big] \leq \mathop{\mathbb{E}}_{X_{i-1}=y}\left[\frac{7}{8}y\right] = \frac{7}{8}\,\mathbb{E}[X_{i-1}] \leq \left(\frac{7}{8}\right)^i \mathbb{E}[X_0] = \left(\frac{7}{8}\right)^i n.$$

In particular, consider $M = 8\log_{8/7} n$. We have that

$$\mu = \mathbb{E}[X_M] \leq \left(\frac{7}{8}\right)^M n \leq \frac{1}{n^8}n = \frac{1}{n^7}.$$

Of course, t participates in more than M recursive calls, if and only if $X_M \geq 1$. However, by Markov's inequality (Theorem 2.4.1), we have that

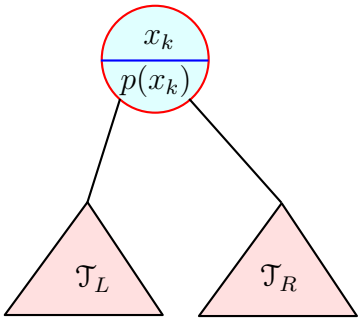
$$\mathbb{P}\left[\begin{array}{c}\text{element } t \text{ participates}\\ \text{in more than } M \text{ recursive calls}\end{array}\right] \leq \mathbb{P}[X_M \geq 1] \leq \frac{\mathbb{E}[X_M]}{1} \leq \frac{1}{n^7},$$

as desired. That is, we proved that the probability that any element of the input T participates in more than M recursive calls is at most $n(1/n^7) \leq 1/n^6$.

Theorem 12.1.1. *For n elements, QuickSort runs in $O(n \log n)$ time, with high probability.*

12.2. Treaps

Anybody that ever implemented a balanced binary tree, knows that it can be very painful. A natural question, is whether we can use randomization to get a simpler data-structure with good performance.



12.2.1. Construction

The key observation is that many of data-structures that offer good performance for balanced binary search trees, do so by storing additional information to help in how to balance the tree. As such, the key Idea is that for every element x inserted into the data-structure, randomly choose a priority $p(x)$; that is, $p(x)$ is chosen uniformly and randomly in the range $[0, 1]$.

So, for the set of elements $X = \{x_1, \ldots, x_n\}$, with (random) priorities $p(x_1), \ldots, p(x_n)$, our purpose is to build a binary tree which is "balanced". So, let us pick the element x_k with the lowest priority in X, and make it the root of the tree. Now, we partition X in the natural way:

(A) L: set of all the numbers smaller than x_k in X, and
(B) R: set of all the numbers larger than x_k in X.

We can now build recursively the trees for L and R, and let denote them by T_L and T_R. We build the natural tree, by creating a node for x_k, having T_L its left child, and T_R as its right child.

We call the resulting tree a ***treap***. As it is a tree over the elements, and a heap over the priorities; that is, TREAP = TREE + HEAP.

Lemma 12.2.1. *Given n elements, the expected depth of a treap T defined over those elements is $O(\log(n))$. Furthermore, this holds with high probability; namely, the probability that the depth of the treap would exceed $c \log n$ is smaller than $\delta = n^{-d}$, where d is an arbitrary constant, and c is a constant that depends on d.*[①]

Furthermore, the probability that T has depth larger than $ct \log(n)$, for any $t \geq 1$, is smaller than n^{-dt}.

Proof: Observe, that every element has equal probability to be in the root of the treap. Thus, the structure of a treap, is identical to the recursive tree of **QuickSort**. Indeed, imagine that instead of picking the pivot uniformly at random, we instead pick the pivot to be the element with the lowest (random) priority. Clearly, these two ways of choosing pivots are equivalent. As such, the claim follows immediately from our analysis of the depth of the recursion tree of **QuickSort**, see Theorem 12.1.1. ■

12.2.2. Operations

The following innocent observation is going to be the key insight in implementing operations on treaps:

Observation 12.2.2. *Given n distinct elements, and their (distinct) priorities, the treap storing them is uniquely defined.*

[①] That is, if we want to decrease the probability of failure, that is δ, we need to increase c.

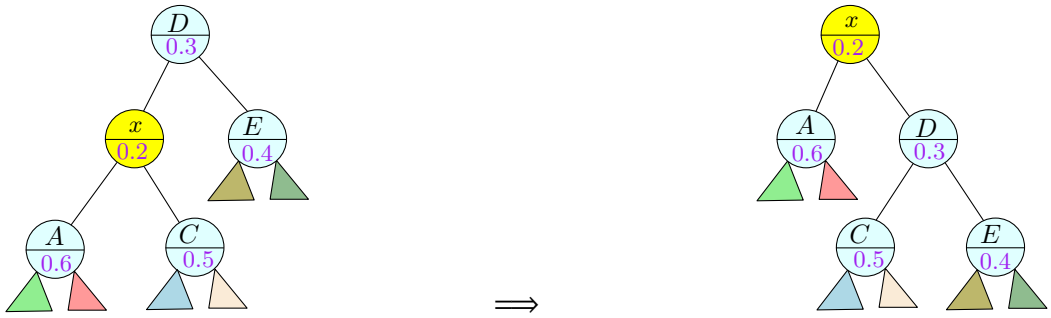


Figure 12.1: **RotateRight**: Rotate right in action. Importantly, after the rotation the priorities are ordered correctly (at least locally for this subtree.

12.2.2.1. Insertion

Given an element x to be inserted into an existing treap T, insert it in the usual way into T (i.e., treat it a regular search binary tree). This takes $O(\text{height}(T))$. Now, x is a leaf in the treap. Set x priority $p(x)$ to some random number $[0, 1]$. Now, while the new tree is a valid search tree, it is not necessarily still a valid treap, as x's priority might be smaller than its parent. So, we need to fix the tree around x, so that the priority property holds.

```
RotateUp(x)
    y ← parent(x)
    while p(y) > p(x) do
        if y.left_child = x then
            RotateRight(y)
        else
            RotateLeft(y)
        y ← parent(x)
```

We call **RotateUp**(x) to do so. Specifically, if x parent is y, and $p(x) < p(y)$, we will rotate x up so that it becomes the parent of y. We repeatedly do it till x has a larger priority than its parent. The rotation operation takes constant time and plays around with priorities, and importantly, it preserves the binary search tree order. A rotate right operation **RotateRight**(D) is depicted in Figure 12.1. **RotateLeft** is the same tree rewriting operation done in the other direction.

Observe that as x is being rotated upwards, the priority properties are being fixed – in particular, as demonstrated in Figure 12.1, nodes are being hanged on nodes that were previously their ancestors, so priorities are still monotonically decreasing along a path.

In the end of this process, both the ordering property and the priority property holds. That is, we have a valid treap that includes all the old elements, and the new element. By Observation 12.2.2, since the treap is uniquely defined, we have updated the treap correctly. Since every time we do a rotation the distance of x from the root decrease by one, it follows that insertions takes $O(\text{height}(T))$.

12.2.2.2. Deletion

Deletion is just an insertion done in reverse. Specifically, to delete an element x from a treap T, set its priority to $+\infty$, and rotate it down it becomes a leaf. The only tricky observation is that you should rotate always so that the child with the lower priority becomes the new parent. Once x becomes a leaf deleting it is trivial - just set the pointer pointing to it in the tree to null.

12.2.2.3. Split

Given an element x stored in a treap T, we would like to split T into two treaps – one treap $T_{\leq}$ for all the elements smaller or equal to x, and the other treap $T_{>}$ for all the elements larger than x. To this end, we set x priority to $-\infty$, fix the priorities by rotating x up so it becomes the root of the treap. The right child of x is the treap $T_{>}$, and we disconnect it from T by setting x right child pointer to null. Next, we restore x to its real priority, and rotate it down to its natural location. The resulting treap is $T_{\leq}$. This again takes time that is proportional to the depth of the treap.

12.2.2.4. Meld

Given two treaps T_L and T_R such that all the elements in T_L are smaller than all the elements in T_R, we would like to merge them into a single treap. Find the largest element x stored in T_L (this is just the element stored in the path going only right from the root of the tree). Set x priority to $-\infty$, and rotate it up the treap so that it becomes the root. Now, x being the largest element in T_L has no right child. Attach T_R as the right child of x. Now, restore x priority to its original priority, and rotate it back so the priorities properties hold.

12.2.3. Summery

Theorem 12.2.3. *Let T be a treap, initialized to an empty treap, and undergoing a sequence of $m = n^c$ insertions, where c is some constant. The probability that the depth of the treap in any point in time would exceed $d \log n$ is $\leq 1/n^f$, where d is an arbitrary constant, and f is a constant that depends only c and d.*

In particular, a treap can handle insertion/deletion in $O(\log n)$ time with high probability.

Proof: Since the first part of the theorem implies that with high probability all these treaps have logarithmic depth, then this implies that all operations takes logarithmic time, as an operation on a treap takes at most the depth of the treap.

As for the first part, let $T_1, \ldots, T_m$ be the sequence of treaps, where T_i is the treap after the ith operation. Similarly, let X_i be the set of elements stored in T_i. By Lemma 12.2.1, the probability that T_i has large depth is tiny. Specifically, we have that

$$\alpha_i = \mathbb{P}[\operatorname{depth}(T_i) > tc' \log n^c] = \mathbb{P}\left[\operatorname{depth}(T_i) > c' t \left(\frac{\log n^c}{\log |T_i|}\right) \cdot \log |T_i|\right] \leq \frac{1}{n^{t \cdot c}},$$

as a tedious and boring but straightforward calculation shows. Picking t to be sufficiently large, we have that the probability that the ith treap is too deep is smaller than $1/n^{f+c}$. By the union bound, since there are n^c treaps in this sequence of operations, it follows that the probability of any of these treaps to be too deep is at most $1/n^f$, as desired. ■

12.3. Extra: Sorting Nuts and Bolts

Problem 12.3.1 (Sorting Nuts and Bolts). You are given a set of n nuts and n bolts. Every nut have a matching bolt, and all the n pairs of nuts and bolts have different sizes. Unfortunately, you get the nuts and bolts separated from each other and you have to match the nuts to the bolts. Furthermore, given a nut and a bolt, all you can do is to try and match one bolt against a nut (i.e., you can not compare two nuts to each other, or two bolts to each other).

When comparing a nut to a bolt, either they match, or one is smaller than other (and you known the relationship after the comparison).

How to match the n nuts to the n bolts quickly? Namely, while performing a small number of comparisons.

The naive algorithm is of course to compare each nut to each bolt, and match them together. This would require a quadratic number of comparisons. Another option is to sort the nuts by size, and the bolts by size and then "merge" the two ordered sets, matching them by size. The only problem is that we can not sorts only the nuts, or only the bolts, since we can not compare them to each other. Indeed, we sort the two sets simultaneously, by simulating **QuickSort**. The resulting algorithm is depicted on the right.

MatchNuts&Bolts (N: nuts, B: bolts)
  Pick a random nut n_{pivot} from N
  Find its matching bolt b_{pivot} in B
  $B_L \leftarrow$ All bolts in B smaller than n_{pivot}
  $N_L \leftarrow$ All nuts in N smaller than b_{pivot}
  $B_R \leftarrow$ All bolts in B larger than n_{pivot}
  $N_R \leftarrow$ All nuts in N larger than b_{pivot}
  MatchNuts&Bolts(N_R,B_R)
  MatchNuts&Bolts(N_L,B_L)

12.3.1. Running time analysis

Definition 12.3.2. Let $\mathcal{RT}$ denote the random variable which is the running time of the algorithm. Note, that the running time is a random variable as it might be different between different executions on the *same input*.

Definition 12.3.3. For a randomized algorithm, we can speak about the expected running time. Namely, we are interested in bounding the quantity $\mathbb{E}[\mathcal{RT}]$ for the worst input.

Definition 12.3.4. The ***expected running-time*** of a randomized algorithm for input of size n is

$$T(n) = \max_{U \text{ is an input of size } n} \mathbb{E}[\mathcal{RT}(U)],$$

where $\mathcal{RT}(U)$ is the running time of the algorithm for the input U.

Definition 12.3.5. The ***rank*** of an element x in a set S, denoted by $\text{rank}(x)$, is the number of elements in S of size smaller or equal to x. Namely, it is the location of x in the sorted list of the elements of S.

Theorem 12.3.6. *The expected running time of* **MatchNuts&Bolts** *(and thus also of* **QuickSort***) is $T(n) = O(n \log n)$, where n is the number of nuts and bolts. The worst case running time of this algorithm is $O(n^2)$.*

Proof: Clearly, we have that $\mathbb{P}\left[\text{rank}(n_{pivot}) = k\right] = \frac{1}{n}$. Furthermore, if the rank of the pivot is k then

$$\begin{aligned} T(n) &= \mathop{\mathbb{E}}_{k=\text{rank}(n_{pivot})}[O(n) + T(k-1) + T(n-k)] = O(n) + \mathop{\mathbb{E}}_{k}[T(k-1) + T(n-k)] \\ &= O(n) + \sum_{k=1}^{n} \mathbb{P}[Rank(Pivot) = k] * (T(k-1) + T(n-k)) \\ &= O(n) + \sum_{k=1}^{n} \frac{1}{n} \cdot (T(k-1) + T(n-k)), \end{aligned}$$

by the definition of expectation. It is not easy to verify that the solution to the recurrence $T(n) = O(n) + \sum_{k=1}^{n} \frac{1}{n} \cdot (T(k-1) + T(n-k))$ is $O(n \log n)$. ■

12.4. Bibliographical Notes

Treaps were invented by Siedel and Aragon [SA96]. Experimental evidence suggests that Treaps performs reasonably well in practice, despite their simplicity, see for example the comparison carried out by Cho and Sahni [CS00]. Implementations of treaps are readily available. An old implementation I wrote in `C` is available here: `http://valis.cs.uiuc.edu/blog/?p=6060`.

References

[CS00] S. Cho and S. Sahni. A new weight balanced binary search tree. *Int. J. Found. Comput. Sci.*, 11(3): 485–513, 2000.

[SA96] R. Seidel and C. R. Aragon. Randomized search trees. *Algorithmica*, 16: 464–497, 1996.

Chapter 13

Concentration of Random Variables – Chernoff's Inequality

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

13.1. Concentration of mass and Chernoff's inequality

13.1.1. Example: Binomial distribution

Consider the binomial distribution $\mathrm{Bin}(n, 1/2)$ for various values of n as depicted in Figure 13.1 – here we think about the value of the variable as the number of heads in flipping a fair coin n times. Clearly, as the value of n increases the probability of getting a number of heads that is significantly smaller or larger than $n/2$ is tiny. Here we are interested in quantifying exactly how far can we divert from this expected value. Specifically, if $X \sim \mathrm{Bin}(n, 1/2)$, then we would be interested in bounding the probability $\mathbb{P}[X > n/2 + \Delta]$, where $\Delta = t\sigma_X = t\sqrt{n}/2$ (i.e., we are t standard deviations away from the expectation). For $t > 2$, this probability is roughly 2^{-t}, which is what we prove here.

More surprisingly, if you look only on the middle of the distribution, it looks the same after clipping away the uninteresting tails, see Figure 13.2; that is, it looks more and more like the normal distribution. This is a universal phenomena known the ***central limit theorem*** – every sum of nicely behaved random variables behaves like the normal distribution. We unfortunately need a more precise quantification of this behavior, thus the following.

13.1.2. A restricted case of Chernoff inequality via games

13.1.2.1. Chernoff games

The game. Consider the game where a player starts with $Y_0 = 1$ dollars. At every round, the player can bet a certain amount x (fractions are fine). With probability half she loses her bet, and with probability half she gains an amount equal to her bet. The player is not allowed to go all in – because if she looses then the game is over. So it is natural to ask what her optimal betting strategy is, such that in the end of the game she has as much money as possible.

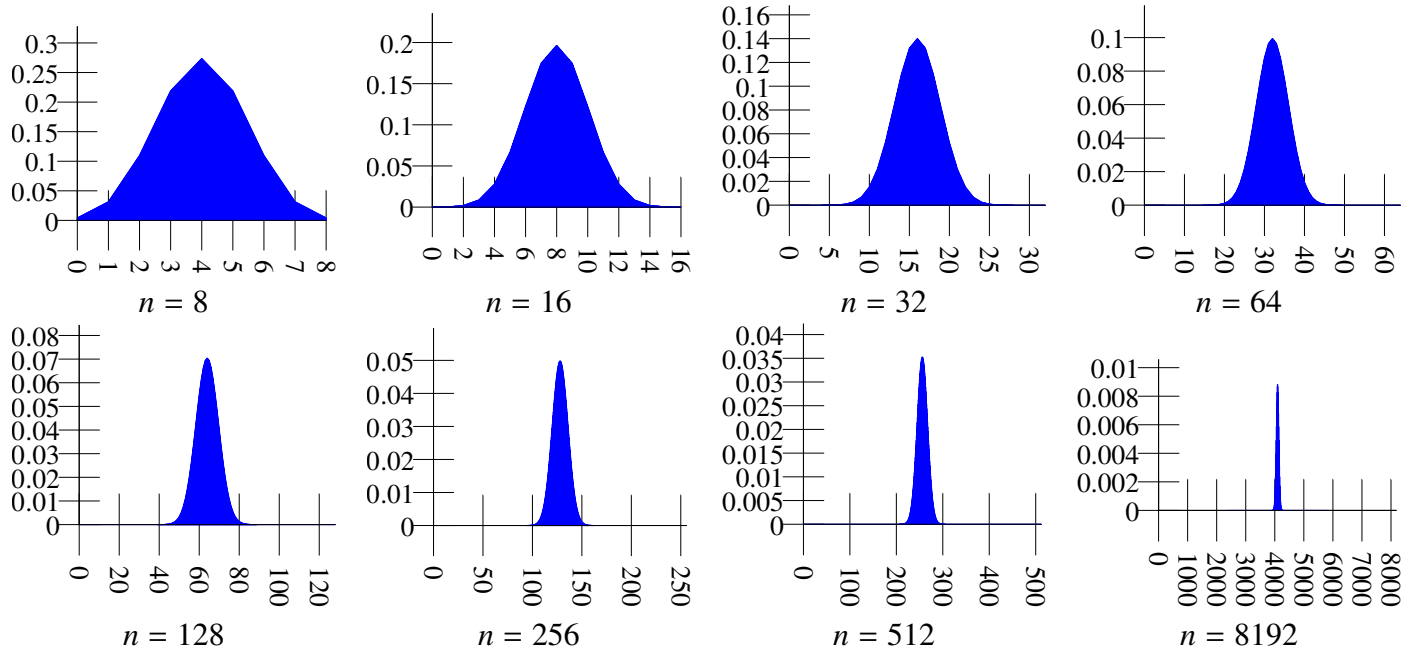


Figure 13.1: The binomial distribution for different values of n. It pretty quickly concentrates around its expectation.

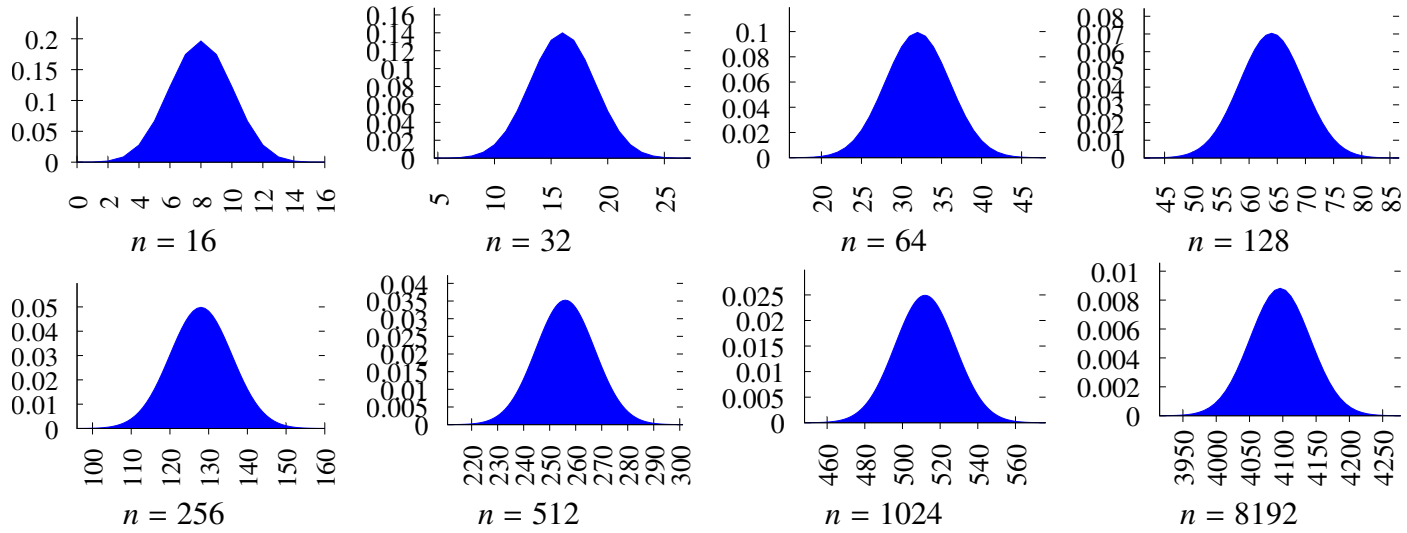


Figure 13.2: The "middle" of the binomial distribution for different values of n. It very quickly converges to the normal distribution (under appropriate rescaling and translation.

$X_i \in \{-1,+1\}$ $\mathbb{P}[X_i=-1]=\mathbb{P}[X_i=1]=1/2$	
$\mathbb{P}[Y \geq \Delta] \leq \exp\left(-\Delta^2/2n\right)$	Theorem 13.1.7
$\mathbb{P}[Y \leq -\Delta] \leq \exp\left(-\Delta^2/2n\right)$	Theorem 13.1.7

$X_i \in \{0,1\}$ $\mathbb{P}[X_i=0]=\mathbb{P}[X_i=1]=1/2$
$\mathbb{P}[\lvert Y-n/2\rvert \geq \Delta] \leq 2\exp\left(-2\Delta^2/n\right)$ Corollary 13.1.9

$X_i \in \{0,1\}$	$\mathbb{P}[X_i=1]=p_i \qquad \mathbb{P}[X_i=0]=1-p_i$	
$\delta \geq 0$	$P=\mathbb{P}[Y>(1+\delta)\mu] < \left(e^{\delta}/(1+\delta)^{1+\delta}\right)^{\mu}$	Theorem 13.2.1
$\delta \in (0,1)$	$P < \exp\left(-\mu\delta^2/3\right)$	Lemma 13.2.5
$\delta \in (0,4)$	$P < \exp\left(-\mu\delta^2/4\right)$	Lemma 13.2.6
$\delta \in (0,6)$	$P < \exp\left(-\mu\delta^2/5\right)$	Lemma 13.2.7
$\delta \geq 2e-1$	$P < 2^{-\mu(1+\delta)}$	Lemma 13.2.8
$\delta \geq e^2$	$P < \exp(-(\mu\delta/2)\ln\delta)$	Lemma 13.2.9
$\delta \geq 0, \varphi \in (0,1]$	$\mathbb{P}[Y>(1+\delta)\mu+\frac{3\ln\varphi^{-1}}{\delta^2}] < \varphi.$	Lemma 13.2.10
$\delta \geq 0$	$\mathbb{P}[Y<(1-\delta)\mu] < \left(e^{-\delta}/(1-\delta)^{1-\delta}\right)^{\mu}$	Theorem 13.2.3
	$\mathbb{P}[Y<(1-\delta)\mu] < \exp\left(-\mu\delta^2/2\right)$	Lemma 13.2.4
$\Delta \geq 0$	$\mathbb{P}[\,Y-\mu \geq \Delta\,] \leq \exp\left(-2\Delta^2/n\right)$ $\mathbb{P}[\,Y-\mu \leq -\Delta\,] \leq \exp\left(-2\Delta^2/n\right).$	Corollary 13.3.5
$\tau \geq 1$	$\mathbb{P}[Y<\mu/\tau] < \exp\left(-[1-\frac{1+\ln\tau}{\tau}]\mu\right)$	Theorem 13.2.3

$X_i \in [0,1]$	Arbitrary independent distributions	
$\delta \in [0,1]$	$\mathbb{P}[Y \geq (1+\delta)\mu] \leq \exp\left(-\delta^2\mu/4\right)$ $\mathbb{P}[Y \leq (1-\delta)\mu] \leq \exp\left(-\delta^2\mu/2\right).$	Theorem 13.3.6
$\Delta \geq 0$	$\mathbb{P}[\,Y-\mu \geq \Delta\,] \leq \exp\left(-2\Delta^2/n\right)$ $\mathbb{P}[\,Y-\mu \leq -\Delta\,] \leq \exp\left(-2\Delta^2/n\right).$	Corollary 13.3.5

$X_i \in [a_i,b_i]$	Arbitrary independent distributions	
$\Delta \geq 0$	$\mathbb{P}[\lvert Y-\mu\rvert \geq \Delta] \leq 2\exp\left(-\frac{2\Delta^2}{\sum_{i=1}^{n}(b_i-a_i)^2}\right)$	Theorem 13.4.3

Table 13.1: Summary of Chernoff type inequalities covered. Here we have n independent random variables $X_1,\ldots,X_n$, $Y=\sum_i X_i$ and $\mu=\mathbb{E}[Y]$.

Is the game pointless? So, let Y_{i-1} be the money the player has in the end of the $(i-1)$th round, and she bets an amount $\psi_i \le Y_{i-1}$ in the ith round. As such, in the end of the ith round, she has

$$Y_i = \begin{cases} Y_{i-1} - \psi_i & \text{LOSE: probability half} \\ Y_{i-1} + \psi_i & \text{WIN: probability half} \end{cases}$$

dollars. This game, in expectation, does not change the amount of money the player has. Indeed, we have

$$\mathbb{E}\Big[Y_i \;\Big|\; Y_{i-1}\Big] = \frac{1}{2}(Y_{i-1} - \psi_i) + \frac{1}{2}(Y_{i-1} + \psi_i) = Y_{i-1}.$$

And as such, we have that $\mathbb{E}[Y_i] = \mathbb{E}\Big[\mathbb{E}\Big[Y_i \Big| Y_{i-1}\Big]\Big] = \mathbb{E}[Y_{i-1}] = \cdots = \mathbb{E}[Y_0] = 1$. In particular, $\mathbb{E}[Y_n] = 1$ – namely, on average, independent of the player strategy she is not going to make any money in this game (and she is allowed to change her bets after every round). Unless, she is lucky[①]...

What about a lucky player? The player believes she will get lucky and wants to develop a strategy to take advantage of it. Formally, she believes that she can win, say, at least $(1+\delta)/2$ fraction of her bets (instead of the predicted $1/2$) – for example, if the bets are in the stock market, she can improve her chances by doing more research on the companies she is investing in[②]. Unfortunately, the player does not know which rounds she is going to be lucky in – so she still needs to be careful.

In a search of a good strategy. Of course, there are many safe strategies the player can use, from not playing at all, to risking only a tiny fraction of her money at each round. In other words, our quest here is to find the best strategy that extracts the maximum benefit for the player out of her inherent luck.

Here, we restrict ourselves to a simple strategy – at every round, the player would bet β fraction of her money, where β is a parameter to be determined. Specifically, in the end of the ith round, the player would have

$$Y_i = \begin{cases} (1-\beta)Y_{i-1} & \text{LOSE} \\ (1+\beta)Y_{i-1} & \text{WIN.} \end{cases}$$

By our assumption, the player is going to win in at least $M = (1+\delta)n/2$ rounds. Our purpose here is to figure out what the value of β should be so that player gets as rich as possible[③]. Now, if the player is successful in $\ge M$ rounds, out of the n rounds of the game, then the amount of money the player has, in the end of the game, is

$$\begin{aligned} Y_n &\ge (1-\beta)^{n-M}(1+\beta)^M = (1-\beta)^{n/2-(\delta/2)n}(1+\beta)^{n/2+(\delta/2)n} = \Big((1-\beta)(1+\beta)\Big)^{n/2-(\delta/2)n}(1+\beta)^{\delta n} \\ &= \Big(1-\beta^2\Big)^{n/2-(\delta/2)n}(1+\beta)^{\delta n} \ge \exp\Big(-2\beta^2\Big)^{n/2-(\delta/2)n}\exp(\beta/2)^{\delta n} = \exp\Big(\Big(-\beta^2+\beta^2\delta+\beta\delta/2\Big)n\Big). \end{aligned}$$

To maximize this quantity, we choose $\beta = \delta/4$ (there is a better choice, see Lemma 13.1.6, but we use this value for the simplicity of exposition). Thus, we have that $Y_n \ge \exp\left(\left(-\frac{\delta^2}{16} + \frac{\delta^3}{16} + \frac{\delta^2}{8}\right)n\right) \ge \exp\left(\frac{\delta^2}{16}n\right)$, proving the following.

Lemma 13.1.1. *Consider a Chernoff game with n rounds, starting with one dollar, where the player wins in $\ge (1+\delta)n/2$ of the rounds. If the player bets $\delta/4$ fraction of her current money, at all rounds, then in the end of the game the player would have at least $\exp\big(n\delta^2/16\big)$ dollars.*

[①] "I would rather have a general who was lucky than one who was good." – Napoleon Bonaparte.

[②] "I am a great believer in luck, and I find the harder I work, the more I have of it." – Thomas Jefferson.

[③] This optimal choice is known as Kelly criterion, see Remark 13.1.3.

Remark 13.1.2. Note, that Lemma 13.1.1 holds if the player wins any $\geq (1+\delta)n/2$ rounds. In particular, the statement does not require randomness by itself – for our application, however, it is more natural and interesting to think about the player wins as being randomly distributed.

Remark 13.1.3. Interestingly, the idea of choosing the best fraction to bet is an old and natural question arising in investments strategies, and the right fraction to use is known as ***Kelly criterion***, going back to Kelly's work from 1956 [Kel56].

13.1.2.2. Chernoff's inequality

The above implies that if a player is lucky, then she is going to become filthy rich[④]. Intuitively, this should be a pretty rare event – because if the player is rich, then (on average) many other people have to be poor. We are thus ready for the kill.

Theorem 13.1.4 (Chernoff's inequality). *Let $X_1, \ldots, X_n$ be n independent random variables, where $X_i = 0$ or $X_i = 1$ with equal probability. Then, for any $\delta \in (0, 1/2)$, we have that*

$$\mathbb{P}\Big[\sum_i X_i \geq (1+\delta)\frac{n}{2}\Big] \leq \exp\Big(-\frac{\delta^2}{16}n\Big).$$

Proof: Imagine that we are playing the Chernoff game above, with $\beta = \delta/4$, starting with 1 dollar, and let Y_i be the amount of money in the end of the ith round. Here $X_i = 1$ indicates that the player won the ith round. We have, by Lemma 13.1.1 and Markov's inequality, that

$$\mathbb{P}\Big[\sum_i X_i \geq (1+\delta)\frac{n}{2}\Big] \leq \mathbb{P}\Big[Y_n \geq \exp\Big(\frac{n\delta^2}{16}\Big)\Big] \leq \frac{\mathbb{E}[Y_n]}{\exp(n\delta^2/16)} = \frac{1}{\exp(n\delta^2/16)} = \exp\Big(-\frac{\delta^2}{16}n\Big). \qquad \blacksquare$$

This is crazy – so intuition maybe? If the player is $(1+\delta)/2$-lucky then she can make a lot of money; specifically, at least $f(\delta) = \exp\big(n\delta^2/16\big)$ dollars by the end of the game. Namely, beating the odds has significant monetary value, and this value grows quickly with δ. Since we are in a "zero-sum" game settings, this event should be very rare indeed. Under this interpretation, of course, the player needs to know in advance the value of δ – so imagine that she guesses it somehow in advance, or she plays the game in parallel with all the possible values of δ, and she settles on the instance that maximizes her profit.

Can one do better? No, not really. Chernoff inequality is tight (this is a challenging homework exercise) up to the constant in the exponent. The best bound I know for this version of the inequality has $1/2$ instead of $1/16$ in the exponent. Note, however, that no real effort was taken to optimize the constants – this is not the purpose of this write-up.

13.1.2.3. Some low level boring calculations

Above, we used the following well known facts.

Lemma 13.1.5. *(A) Markov's inequality. For any positive random variable X and $t > 0$, we have $\mathbb{P}[X \geq t] \leq \mathbb{E}[X]/t$. (B) For any two random variables X and Y, we have that $\mathbb{E}\big[X\big] = \mathbb{E}\Big[\mathbb{E}\big[X\,\big|Y\big]\Big]$. (C) For $x \in (0,1)$, $1 + x \geq e^{x/2}$. (D) For $x \in (0, 1/2)$, $1 - x \geq e^{-2x}$.*

Lemma 13.1.6. *The quantity $\exp\Big(\big(-\beta^2 + \beta^2\delta + \beta\delta/2\big)n\Big)$ is maximal for $\beta = \frac{\delta}{4(1-\delta)}$.*

Proof: We have to maximize $f(\beta) = -\beta^2 + \beta^2\delta + \beta\delta/2$ by choosing the correct value of β (as a function of δ, naturally). $f'(\beta) = -2\beta + 2\beta\delta + \delta/2 = 0 \iff 2(\delta-1)\beta = -\delta/2 \iff \beta = \frac{\delta}{4(1-\delta)}$. $\blacksquare$

[④] Not that there is anything wrong with that – many of my friends are filthy,

13.1.3. A proof for $-1/+1$ case

Theorem 13.1.7. *Let $X_1, \ldots, X_n$ be n independent random variables, such that $\mathbb{P}[X_i = 1] = \mathbb{P}[X_i = -1] = \frac{1}{2}$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^{n} X_i$. Then, for any $\Delta > 0$, we have*

$$\mathbb{P}\Big[Y \geq \Delta\Big] \leq \exp\Big(-\Delta^2/2n\Big).$$

Proof: Clearly, for an arbitrary t, to specified shortly, we have

$$\mathbb{P}[Y \geq \Delta] = \mathbb{P}[\exp(tY) \geq \exp(t\Delta)] \leq \frac{\mathbb{E}[\exp(tY)]}{\exp(t\Delta)},$$

the first part follows by the fact that $\exp(\cdot)$ preserve ordering, and the second part follows by the Markov inequality.

Observe that

$$\begin{aligned}
\mathbb{E}[\exp(tX_i)] &= \frac{1}{2}e^{t} + \frac{1}{2}e^{-t} = \frac{e^t + e^{-t}}{2} \\
&= \frac{1}{2}\left(1 + \frac{t}{1!} + \frac{t^2}{2!} + \frac{t^3}{3!} + \cdots\right) \\
&+ \frac{1}{2}\left(1 - \frac{t}{1!} + \frac{t^2}{2!} - \frac{t^3}{3!} + \cdots\right) \\
&= \left(1 + \quad + \frac{t^2}{2!} + \quad + \cdots + \frac{t^{2k}}{(2k)!} + \cdots\right),
\end{aligned}$$

by the Taylor expansion of $\exp(\cdot)$. Note, that $(2k)! \geq (k!)2^k$, and thus

$$\mathbb{E}[\exp(tX_i)] = \sum_{i=0}^{\infty} \frac{t^{2i}}{(2i)!} \leq \sum_{i=0}^{\infty} \frac{t^{2i}}{2^i(i!)} = \sum_{i=0}^{\infty} \frac{1}{i!}\left(\frac{t^2}{2}\right)^i = \exp\Big(t^2/2\Big),$$

again, by the Taylor expansion of $\exp(\cdot)$. Next, by the independence of the X_is, we have

$$\mathbb{E}[\exp(tY)] = \mathbb{E}\left[\exp\left(\sum_i tX_i\right)\right] = \mathbb{E}\left[\prod_i \exp(tX_i)\right] = \prod_{i=1}^{n} \mathbb{E}[\exp(tX_i)] \leq \prod_{i=1}^{n} e^{t^2/2} = e^{nt^2/2}.$$

We have $\mathbb{P}[Y \geq \Delta] \leq \dfrac{\exp\Big(nt^2/2\Big)}{\exp(t\Delta)} = \exp\Big(nt^2/2 - t\Delta\Big)$.

Next, by minimizing the above quantity for t, we set $t = \Delta/n$. We conclude,

$$\mathbb{P}[Y \geq \Delta] \leq \exp\left(\frac{n}{2}\left(\frac{\Delta}{n}\right)^2 - \frac{\Delta}{n}\Delta\right) = \exp\left(-\frac{\Delta^2}{2n}\right). \qquad \blacksquare$$

By the symmetry of Y, we get the following:

Corollary 13.1.8. *Let $X_1, \ldots, X_n$ be n independent random variables, such that $\mathbb{P}[X_i = 1] = \mathbb{P}[X_i = -1] = \frac{1}{2}$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^{n} X_i$. Then, for any $\Delta > 0$, we have $\mathbb{P}[|Y| \geq \Delta] \leq 2\exp\Big(-\Delta^2/2n\Big)$.*

Corollary 13.1.9. *Let $X_1, \ldots, X_n$ be n independent* coin flips*, such that $\mathbb{P}[X_i = 0] = \mathbb{P}[X_i = 1] = \frac{1}{2}$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^{n} X_i$. Then, for any $\Delta > 0$, we have $\mathbb{P}[|Y - n/2| \geq \Delta] \leq 2\exp\Big(-2\Delta^2/n\Big)$.*

Proof: Consider the random variables $Z_i = 2X_i - 1 \in \{-1, +1\}$. We have that

$$\mathbb{P}[|Y - n/2| \geq \Delta] = \mathbb{P}[|2Y - n| \geq 2\Delta] = \mathbb{P}\left[\left|\sum_{i=1}^{n}(2X_i - 1)\right| \geq 2\Delta\right] = \mathbb{P}\left[\left|\sum_{i=1}^{n} Z_i\right| \geq 2\Delta\right] \leq 2\exp\left(-2\Delta^2/n\right),$$

by Corollary 13.1.8 applied to the independent random variables $Z_1, \ldots, Z_n$. ■

Remark 13.1.10. Before going any further, it is might be instrumental to understand what this inequalities imply. Consider then case where X_i is either zero or one with probability half. In this case $\mu = \mathbb{E}[Y] = n/2$. Set $\delta = t\sqrt{n}$ ($\sqrt{\mu}$ is approximately the standard deviation of X if $p_i = 1/2$). We have by

$$\mathbb{P}\left[\left|Y - \frac{n}{2}\right| \geq \Delta\right] \leq 2\exp\left(-2\Delta^2/n\right) = 2\exp\left(-2(t\sqrt{n})^2/n\right) = 2\exp\left(-2t^2\right).$$

Thus, Chernoff inequality implies exponential decay (i.e., $\leq 2^{-t}$) with t standard deviations, instead of just polynomial (i.e., $\leq 1/t^2$) by the Chebychev's inequality.

13.2. The Chernoff Bound — General Case

Here we present the Chernoff bound in a more general settings.

Theorem 13.2.1. *Let $X_1, \ldots, X_n$ be n independent variables, where $\mathbb{P}[X_i = 1] = p_i$ and $\mathbb{P}[X_i = 0] = q_i = 1 - p_i$, for all i. Let $X = \sum_{i=1}^{b} X_i$. $\mu = \mathbb{E}[X] = \sum_i p_i$. For any $\delta > 0$, we have*

$$\mathbb{P}[X > (1 + \delta)\mu] < \left(e^{\delta}/(1 + \delta)^{1+\delta}\right)^{\mu}.$$

Proof: We have $\mathbb{P}[X > (1 + \delta)\mu] = \mathbb{P}\left[e^{tX} > e^{t(1+\delta)\mu}\right]$. By the Markov inequality, we have:

$$\mathbb{P}\left[X > (1 + \delta)\mu\right] < \frac{\mathbb{E}\left[e^{tX}\right]}{e^{t(1+\delta)\mu}}$$

On the other hand,

$$\mathbb{E}\left[e^{tX}\right] = \mathbb{E}\left[e^{t(X_1+X_2\ldots+X_n)}\right] = \mathbb{E}\left[e^{tX_1}\right]\cdots\mathbb{E}\left[e^{tX_n}\right].$$

Namely,

$$\mathbb{P}[X > (1 + \delta)\mu] < \frac{\prod_{i=1}^{n}\mathbb{E}\left[e^{tX_i}\right]}{e^{t(1+\delta)\mu}} = \frac{\prod_{i=1}^{n}\left((1 - p_i)e^0 + p_i e^t\right)}{e^{t(1+\delta)\mu}} = \frac{\prod_{i=1}^{n}(1 + p_i(e^t - 1))}{e^{t(1+\delta)\mu}}.$$

Let $y = p_i(e^t - 1)$. We know that $1 + y < e^y$ (since $y > 0$). Thus,

$$\begin{aligned}
\mathbb{P}[X > (1 + \delta)\mu] &< \frac{\prod_{i=1}^{n}\exp(p_i(e^t - 1))}{e^{t(1+\delta)\mu}} = \frac{\exp(\sum_{i=1}^{n} p_i(e^t - 1))}{e^{t(1+\delta)\mu}} \\
&= \frac{\exp((e^t - 1)\sum_{i=1}^{n} p_i)}{e^{t(1+\delta)\mu}} = \frac{\exp((e^t - 1)\mu)}{e^{t(1+\delta)\mu}} = \left(\frac{\exp(e^t - 1)}{e^{t(1+\delta)}}\right)^{\mu} \\
&= \left(\frac{\exp(\delta)}{(1 + \delta)^{(1+\delta)}}\right)^{\mu},
\end{aligned}$$

if we set $t = \log(1 + \delta)$. ■

13.2.1. The lower tail

We need the following low level lemma.

Lemma 13.2.2. *For $x \in [0,1)$, we have $(1-x)^{1-x} \geq \exp(-x + x^2/2)$.*

Proof: For $x \in [0,1)$, we have, by the Taylor expansion, that $\ln(1-x) = -\sum_{i=1}^{\infty}(x^i/i)$. As such, we have

$$(1-x)\ln(1-x) = -(1-x)\sum_{i=1}^{\infty}\frac{x^i}{i} = -\sum_{i=1}^{\infty}\frac{x^i}{i} + \sum_{i=1}^{\infty}\frac{x^{i+1}}{i} = -x + \sum_{i=2}^{\infty}\left(\frac{x^i}{i-1} - \frac{x^i}{i}\right) = -x + \sum_{i=2}^{\infty}\frac{x^i}{i(i-1)}.$$

This implies that $(1-x)\ln(1-x) \geq -x + x^2/2$, which implies the claim by exponentiation. ■

Theorem 13.2.3. *Let $X_1, \ldots, X_n$ be n independent random variables, where $\mathbb{P}[X_i = 1] = p_i$, $\mathbb{P}[X_i = 0] = q_i = 1 - p_i$, for all i. For $X = \sum_{i=1}^n X_i$, its expectation is $\mu = \mathbb{E}[X] = \sum_i p_i$. We have that*

$$\mathbb{P}[X < (1-\delta)\mu] < \left[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right]^{\mu}.$$

For any positive $\tau > 1$, we have that $\mathbb{P}[X < \mu/\tau] \leq \exp\left(-\left(1 - \frac{1+\ln\tau}{\tau}\right)\mu\right)$.

Proof: We follow the same proof template seen already. For $t = -\ln(1-\delta) > 0$, we have $\mathbb{E}[\exp(-tX_i)] = (1-p_i)e^0 + p_i e^{-t} = 1 - p_i + p_i(1-\delta) = 1 - p_i\delta \leq \exp(-p_i\delta)$. As such, we have

$$\begin{aligned}\mathbb{P}[X < (1-\delta)\mu] &= \mathbb{P}[-X > -(1-\delta)\mu] = \mathbb{P}[\exp(-tX) > \exp(-t(1-\delta)\mu)] \leq \frac{\prod_{i=1}^n \mathbb{E}[\exp(-tX_i)]}{\exp(-t(1-\delta)\mu)} \\ &\leq \frac{\exp(-\sum_{i=1}^n p_i\delta)}{\exp(-t(1-\delta)\mu)} = \left[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right]^{\mu}.\end{aligned}$$

For the last inequality, set $\delta = 1 - 1/\tau$, and observe that

$$\mathbb{P}[X < (1-\delta)\mu] \leq \left[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right]^{\mu} = \left[\frac{\exp(-1+1/\tau)}{(1/\tau)^{1/\tau}}\right]^{\mu} = \exp\left(-\left(1 - \frac{1+\ln\tau}{\tau}\right)\mu\right).$$ ■

Lemma 13.2.4. *Let $X_1, \ldots, X_n \in \{0,1\}$ be n independent random variables, with $p_i = \mathbb{P}[X_i = 1]$, for all i. For $X = \sum_{i=1}^n X_i$, and $\mu = \mathbb{E}[X] = \sum_i p_i$, we have that $\mathbb{P}[X < (1-\delta)\mu] < \operatorname{Exp}\left(-\mu\delta^2/2\right)$.*

Proof: This alternative simplified form of Theorem 13.2.3, follows readily from Lemma 13.2.2, since

$$\mathbb{P}[X < (1-\delta)\mu] \leq \left[\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right]^{\mu} \leq \left[\frac{e^{-\delta}}{\operatorname{Exp}(-\delta + \delta^2/2)}\right]^{\mu} \leq \operatorname{Exp}(-\mu\delta^2/2).$$ ■

13.2.2. A more convenient form of Chernoff's inequality

Lemma 13.2.5. *Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}[X] = \sum_i p_i$. For $\delta, \in (0, 1)$, we have*

$$\mathbb{P}[X > (1+\delta)\mu] < \exp\Big(-\mu\delta^2/3\Big).$$

Proof: By Theorem 13.2.1, it is sufficient to prove, for $\delta \in [0, 1]$, that

$$\left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \leq \exp\left(-\frac{\mu\delta^2}{c}\right) \iff \mu(\delta - (1+\delta)\ln(1+\delta)) \leq -\mu\delta^2/c$$

$$\iff f(\delta) = \delta^2/c + \delta - (1+\delta)\ln(1+\delta) \leq 0.$$

We have

$$f'(\delta) = 2\delta/c - \ln(1+\delta). \qquad \text{and} \qquad f''(\delta) = 2/c - \frac{1}{1+\delta}.$$

For $c = 3$, we have $f''(\delta) \leq 0$ for $\delta \in [0, 1/2]$, and $f''(\delta) \geq 0$ for $\delta \in [1/2, 1]$. Namely, $f'(\delta)$ achieves its maximum either at 0 or 1. As $f'(0) = 0$ and $f'(1) = 2/3 - \ln 2 \approx -0.02 < 0$, we conclude that $f'(\delta) \leq 0$. Namely, f is a monotonically decreasing function in $[0, 1]$, which implies that $f(\delta) \leq 0$, for all δ in this range, thus implying the claim. ■

Lemma 13.2.6. *Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}[X] = \sum_i p_i$. For $\delta \in (0, 4)$, we have*

$$\mathbb{P}[X > (1+\delta)\mu] < \exp\Big(-\mu\delta^2/4\Big),$$

Proof: Lemma 13.2.5 implies a stronger bound, so we need to prove the claim only for $\delta \in (1, 4]$. Continuing as in the proof of Lemma 13.2.5, for case $c = 4$, we have to prove that

$$f(\delta) = \delta^2/4 + \delta - (1+\delta)\ln(1+\delta) \leq 0,$$

where $f''(\delta) = 1/2 - \frac{1}{1+\delta}$.

For $\delta > 1$, we have $f''(\delta) > 0$. Namely $f(\cdot)$ is convex for $\delta \geq 1$, and it achieves its maximum on the interval $[1, 4]$ on the endpoints. In particular, $f(1) \approx -0.13$, and $f(4) \approx -0.047$, which implies the claim. ■

Lemma 13.2.7. *Let $X_1, \ldots, X_n$ be n independent random variables, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}[X] = \sum_i p_i$. For $\delta \in (0, 6)$, we have*

$$\mathbb{P}[X > (1+\delta)\mu] < \exp\Big(-\mu\delta^2/5\Big),$$

Proof: Lemma 13.2.6 implies a stronger bound, so we need to prove the claim only for $\delta \in (4, 5]$. Continuing as in the proof of Lemma 13.2.5, for case $c = 5$, we have to prove that

$$f(\delta) = \delta^2/5 + \delta - (1+\delta)\ln(1+\delta) \leq 0,$$

where $f''(\delta) = 2/5 - \frac{1}{1+\delta}$. For $\delta \geq 4$, we have $f''(\delta) > 0$. Namely $f(\cdot)$ is convex for $\delta \geq 4$, and it achieves its maximum on the interval $[4, 6]$ on the endpoints. In particular, $f(4) \approx -0.84$, and $f(6) \approx -0.42$, which implies the claim. ■

Lemma 13.2.8. *Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}[X] = \sum_i p_i$. For $\delta > 2e - 1$, we have $\mathbb{P}[X > (1+\delta)\mu] < 2^{-\mu(1+\delta)}$.*

Proof: By Theorem 13.2.1, we have

$$\left(\frac{e}{1+\delta}\right)^{(1+\delta)\mu} \leq \left(\frac{e}{1+2e-1}\right)^{(1+\delta)\mu} \leq 2^{-(1+\delta)\mu},$$

since $\delta > 2e - 1$. ■

Lemma 13.2.9. *Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $X = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}[X] = \sum_i p_i$. For $\delta > e^2$, we have $\mathbb{P}[X > (1+\delta)\mu] < \exp\left(-\frac{\mu\delta \ln \delta}{2}\right)$.*

Proof: Observe that

$$\mathbb{P}\Big[X > (1+\delta)\mu\Big] < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} = \exp\Big(\mu\delta - \mu(1+\delta)\ln(1+\delta)\Big). \tag{13.1}$$

As such, we have

$$\mathbb{P}\Big[X > (1+\delta)\mu\Big] < \exp\Big(-\mu(1+\delta)(\ln(1+\delta) - 1)\Big) \leq \exp\left(-\mu\delta \ln \frac{1+\delta}{e}\right) \leq \exp\left(-\frac{\mu\delta \ln \delta}{2}\right),$$

since for $x \geq e^2$ we have that $\dfrac{1+x}{e} \geq \sqrt{x} \iff \ln \dfrac{1+x}{e} \geq \dfrac{\ln x}{2}$. ■

13.2.2.1. Bound when the expectation is small

Lemma 13.2.10. *Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}[Y] = \sum_i p_i$. For $\delta \in (0, 1]$, and $\varphi \in (0, 1]$, we have*

$$\mathbb{P}\left[Y > (1+\delta)\mu + \frac{3\ln \varphi^{-1}}{\delta^2}\right] < \varphi.$$

Proof: Let $\xi = \delta + \frac{3\ln \varphi^{-1}}{\mu\delta^2}$. If $\xi \geq 2e - 1 \approx 4.43$, by Lemma 13.2.8, we have

$$\alpha = \mathbb{P}\left[Y > (1+\delta)\mu + \frac{3\ln \varphi^{-1}}{\delta^2}\right] = \mathbb{P}[Y > (1+\xi)\mu] \leq 2^{-\mu(1+\xi)} < \varphi,$$

since $-\mu(1+\xi) > -\mu\xi > \mu\frac{3\ln \varphi^{-1}}{\mu\delta^2} > \log_2 \varphi^{-1}$, since $\delta \in (0, 1]$.

If $\xi \leq 6$, then by Lemma 13.2.7, we have

$$\alpha = \mathbb{P}[Y > (1+\xi)\mu] \leq \exp\Big(-\mu\xi^2/5\Big) \leq \varphi,$$

since

$$-\frac{\mu}{5}\xi^2 = -\frac{\mu}{5}\left(\delta + \frac{3\ln \varphi^{-1}}{\mu\delta^2}\right)^2 > -\frac{\mu}{5}\left(2 \cdot \delta \cdot \frac{3\ln \varphi^{-1}}{\mu\delta^2}\right) = -\frac{6}{5} \cdot \frac{\ln \varphi}{\delta} > -\ln \varphi.$$ ■

Example 13.2.11. Let $X_1, \ldots, X_n$ be n independent Bernoulli trials, where $\mathbb{P}[X_i = 1] = p_i$, and $\mathbb{P}[X_i = 0] = 1 - p_i$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^{b} X_i$, and $\mu = \mathbb{E}[Y] = \sum_i p_i$. Assume that $\mu \leq 1/2$. Setting $\delta = 1$, We have, for $t > 6$, that

$$\mathbb{P}[Y > 1 + t] \leq \mathbb{P}\left[Y > (1+\delta)\mu + \frac{3\ln \exp(t/3)}{\delta^2}\right] \leq \exp(-t/3),$$

by Lemma 13.2.10.

13.3. A special case of Hoeffding's inequality

In this section, we prove yet another version of Chernoff inequality, where each variable is randomly picked according to its own distribution in the range $[0, 1]$. We prove a more general version of this inequality in Section 13.4, but the version presented here does not follow from this generalization.

Theorem 13.3.1. *Let $X_1, \ldots, X_n \in [0, 1]$ be n independent random variables, let $X = \sum_{i=1}^{n} X_i$, and let $\mu = \mathbb{E}[X]$. We have that $\mathbb{P}\Big[X - \mu \geq \eta\Big] \leq \left(\frac{\mu}{\mu+\eta}\right)^{\mu+\eta}\left(\frac{n-\mu}{n-\mu-\eta}\right)^{n-\mu-\eta}$.*

Proof: Let $s \geq 1$ be some arbitrary parameter. By the standard arguments, we have

$$\gamma = \mathbb{P}\Big[X \geq \mu + \eta\Big] = \mathbb{P}\Big[s^X \geq s^{\mu+\eta}\Big] \leq \frac{\mathbb{E}\Big[s^X\Big]}{s^{\mu+\eta}} = s^{-\mu-\eta}\prod_{i=1}^{n}\mathbb{E}\Big[s^{X_i}\Big].$$

By calculations, see Lemma 13.3.7 below, one can show that $\mathbb{E}\Big[s^{X_1}\Big] \leq 1 + (s-1)\,\mathbb{E}[X_i]$. As such, by the AM-GM inequality[5], we have that

$$\prod_{i=1}^{n}\mathbb{E}\Big[s^{X_i}\Big] \leq \prod_{i=1}^{n}\Big(1 + (s-1)\,\mathbb{E}[X_i]\Big) \leq \left(\frac{1}{n}\sum_{i=1}^{n}\Big(1 + (s-1)\,\mathbb{E}[X_i]\Big)\right)^n = \left(1 + (s-1)\frac{\mu}{n}\right)^n.$$

Setting $s = \dfrac{(\mu+\eta)(n-\mu)}{\mu(n-\mu-\eta)} = \dfrac{\mu n - \mu^2 + \eta n - \eta\mu}{\mu n - \mu^2 - \eta\mu}$ we have that

$$1 + (s-1)\frac{\mu}{n} = 1 + \frac{\eta n}{\mu n - \mu^2 - \eta\mu}\cdot\frac{\mu}{n} = 1 + \frac{\eta}{n-\mu-\eta} = \frac{n-\mu}{n-\mu-\eta}.$$

As such, we have that

$$\gamma \leq s^{-\mu-\eta}\prod_{i=1}^{n}\mathbb{E}\Big[s^{X_i}\Big] = \left(\frac{\mu(n-\mu-\eta)}{(\mu+\eta)(n-\mu)}\right)^{\mu+\eta}\left(\frac{n-\mu}{n-\mu-\eta}\right)^{n} = \left(\frac{\mu}{(\mu+\eta)}\right)^{\mu+\eta}\left(\frac{n-\mu}{n-\mu-\eta}\right)^{n-\mu-\eta}. \qquad \blacksquare$$

Remark 13.3.2. Setting $s = (\mu+\eta)/\mu$ in the proof of Theorem 13.3.1, we have

$$\mathbb{P}\Big[X - \mu \geq \eta\Big] \leq \left(\tfrac{\mu}{\mu+\eta}\right)^{\mu+\eta}\left(1 + \left(\tfrac{\mu+\eta}{\mu} - 1\right)\tfrac{\mu}{n}\right)^n = \left(\tfrac{\mu}{\mu+\eta}\right)^{\mu+\eta}\left(1 + \tfrac{\eta}{n}\right)^n.$$

Corollary 13.3.3. *Let $X_1, \ldots, X_n \in [0, 1]$ be n independent random variables, let $\overline{X} = \sum_{i=1}^{n} X_i/n$, $p = \mathbb{E}\Big[\overline{X}\Big] = \mu/n$ and $q = 1 - p$. Then, we have that $\mathbb{P}\Big[\,\overline{X} - p \geq t\,\Big] \leq \exp(nf(t))$, for*

$$f(t) = (p+t)\ln\frac{p}{p+t} + (q-t)\ln\frac{q}{q-t}. \tag{13.2}$$

Theorem 13.3.4. *Let $X_1, \ldots, X_n \in [0, 1]$ be n independent random variables, let $\overline{X} = (\sum_{i=1}^{n} X_i)/n$, and let $p = \mathbb{E}[X]$. We have that $\mathbb{P}\Big[\,\overline{X} - p \geq t\,\Big] \leq \exp\Big(-2nt^2\Big)$ and $\mathbb{P}\Big[\,\overline{X} - p \leq -t\,\Big] \leq \exp\Big(-2nt^2\Big)$.*

[5] The inequality between arithmetic and geometric means: $(\sum_{i=1}^{n} x_i)/n \geq \sqrt[n]{x_1 \cdots x_n}$.

Proof: Let $p = \mu/n$, $q = 1 - p$, and let $f(t)$ be the function from Eq. (13.2), for $t \in (-p, q)$. Now, we have that

$$f'(t) = \ln \frac{p}{p+t} + (p+t)\frac{p+t}{p}\left(-\frac{p}{(p+t)^2}\right) - \ln \frac{q}{q-t} - (q-t)\frac{q-t}{q}\frac{q}{(q-t)^2} = \ln \frac{p}{p+t} - \ln \frac{q}{q-t}$$
$$= \ln \frac{p(q-t)}{q(p+t)}.$$

As for the second derivative, we have

$$f''(t) = \frac{\cancel{q(p+t)}}{p(q-t)} \cdot \frac{p}{q} \cdot \frac{(p+t)(-1) - (q-t)}{\cancel{(p+t)^2}} = \frac{-p-t-q+t}{(q-t)(p+t)} = -\frac{1}{(q-t)(p+t)} \leq -4.$$

Indeed, $t \in (-p, q)$ and the denominator is minimized for $t = (q-p)/2$, and as such $(q-t)(p+t) \leq \Big(2q - (q-p)\Big)\Big(2p + (q-p)\Big)/4 = (p+q)^2/4 = 1/4$.

Now, $f(0) = 0$ and $f'(0) = 0$, and by Taylor's expansion, we have that $f(t) = f(0) + f'(0)t + \dfrac{f''(x)}{2}t^2 \leq -2t^2$, where x is between 0 and t.

The first bound now readily follows from plugging this bound into Corollary 13.3.3. The second bound follows by considering the random variants $Y_i = 1 - X_i$, for all i, and plugging this into the first bound. Indeed, for $\overline{Y} = 1 - \overline{X}$, we have that $q = \mathbb{E}\Big[\overline{Y}\Big]$, and then $\overline{X} - p \leq -t \iff t \leq p - \overline{X} \iff t \leq 1 - q - (1 - \overline{Y}) = \overline{Y} - q$. Thus, $\mathbb{P}\Big[\overline{X} - p \leq -t\Big] = \mathbb{P}\Big[\overline{Y} - q \geq t\Big] \leq \exp\Big(-2nt^2\Big)$. ■

Corollary 13.3.5. *Let $X_1, \ldots, X_n \in [0,1]$ be n independent random variables, let $Y = \sum_{i=1}^n X_i$, and let $\mu = \mathbb{E}[X]$. For any $\Delta > 0$, we have $\mathbb{P}[\, Y - \mu \geq \Delta\,] \leq \exp\Big(-2\Delta^2/n\Big)$ and $\mathbb{P}[\, Y - \mu \leq -\Delta\,] \leq \exp\Big(-2\Delta^2/n\Big)$.*

Proof: For $\overline{X} = Y/n$, $p = \mu/n$, and $t = \Delta/n$, by Theorem 13.3.4, we have

$$\mathbb{P}[Y - \mu \geq \Delta] = \mathbb{P}[\overline{X} - p \geq t] \leq \exp\Big(-2nt^2\Big) = \exp\Big(-2\Delta^2/n\Big).$$ ■

Theorem 13.3.6. *Let $X_1, \ldots, X_n \in [0,1]$ be n independent random variables, let $X = (\sum_{i=1}^n X_i)$, and let $\mu = \mathbb{E}[X]$. We have that $\mathbb{P}[X - \mu \geq \varepsilon\mu] \leq \exp\Big(-\varepsilon^2\mu/4\Big)$ and $\mathbb{P}[X - \mu \leq -\varepsilon\mu] \leq \exp\Big(-\varepsilon^2\mu/2\Big)$.*

Proof: Let $p = \mu/n$, and let $g(x) = f(px)$, for $x \in [0,1]$ and $xp < q$. As before, computing the derivative of g, we have

$$g'(x) = pf'(xp) = p \ln \frac{p(q-xp)}{q(p+xp)} = p \ln \frac{q-xp}{q(1+x)} \leq p \ln \frac{1}{1+x} \leq -\frac{px}{2},$$

since $(q - xp)/q$ is maximized for $x = 0$, and $\ln \frac{1}{1+x} \leq -x/2$, for $x \in [0,1]$, as can be easily verified[6]. Now, $g(0) = f(0) = 0$, and by integration, we have that $g(x) = \int_{y=0}^{x} g'(y)\mathrm{d}y \leq \int_{y=0}^{x}(-py/2)\mathrm{d}y = -px^2/4$. Now, plugging into Corollary 13.3.3, we get that the desired probability $\mathbb{P}[X - \mu \geq \varepsilon\mu]$ is

$$\mathbb{P}\Big[\overline{X} - p \geq \varepsilon p\Big] \leq \exp(nf(\varepsilon p)) = \exp\Big(ng(\varepsilon)\Big) \leq \exp\Big(-pn\varepsilon^2/4\Big) = \exp\Big(-\mu\varepsilon^2/4\Big).$$

As for the other inequality, set $h(x) = g(-x) = f(-xp)$. Then

$$h'(x) = -pf'(-xp) = -p \ln \frac{p(q+xp)}{q(p-xp)} = p \ln \frac{q(1-x)}{q+xp} = p \ln \frac{q-xq}{q+xp} = p \ln\left(1 - x\frac{p+q}{q+xp}\right)$$
$$= p \ln\left(1 - x\frac{1}{q+xp}\right) \leq p \ln(1-x) \leq -px,$$

[6] Indeed, this is equivalent to $\frac{1}{1+x} \leq e^{-x/2} \iff e^{x/2} \leq 1 + x$, which readily holds for $x \in [0,1]$.

since $1 - x \leq e^{-x}$. By integration, as before, we conclude that $h(x) \leq -px^2/2$. Now, plugging into Corollary 13.3.3, we get $\mathbb{P}[X - \mu \leq -\varepsilon\mu] = \mathbb{P}\Big[\overline{X} - p \leq -\varepsilon p\Big] \leq \exp(nf(-\varepsilon p)) \leq \exp(nh(\varepsilon)) \leq \exp\Big(-np\varepsilon^2/2\Big) \leq \exp\Big(-\mu\varepsilon^2/2\Big)$. ■

13.3.1. Some technical lemmas

Lemma 13.3.7. *Let $X \in [0,1]$ be a random variable, and let $s \geq 1$. Then $\mathbb{E}\Big[s^X\Big] \leq 1 + (s-1)\,\mathbb{E}[X]$.*

Proof: For the sake of simplicity of exposition, assume that X is a discrete random variable, and that there is a value $\alpha \in (0, 1/2)$, such that $\beta = \mathbb{P}[X = \alpha] > 0$. Consider the modified random variable X', such that $\mathbb{P}[X' = 0] = \mathbb{P}[X = 0] + \beta/2$, and $\mathbb{P}[X' = 2\alpha] = \mathbb{P}[X = \alpha] + \beta/2$. Clearly, $\mathbb{E}[X] = \mathbb{E}[X']$. Next, observe that $\mathbb{E}\Big[s^{X'}\Big] - \mathbb{E}\Big[s^X\Big] = (\beta/2)(s^{2\alpha} + s^0) - \beta s^\alpha \geq 0$, by the convexity of s^x. We conclude that $\mathbb{E}\Big[s^X\Big]$ achieves its maximum if takes only the values 0 and 1. But then, we have that $\mathbb{E}\Big[s^X\Big] = \mathbb{P}[X = 0]s^0 + \mathbb{P}[X = 1]s^1 = (1 - \mathbb{E}[X]) + \mathbb{E}[X]\, s = 1 + (s-1)\,\mathbb{E}[X]$, as claimed. ■

13.4. Hoeffding's inequality

In this section, we prove a generalization of Chernoff's inequality. The proof is considerably more tedious, and it is included here for the sake of completeness.

Lemma 13.4.1. *Let X be a random variable. If $\mathbb{E}[X] = 0$ and $a \leq X \leq b$, then for any $s > 0$, we have $\mathbb{E}\Big[e^{sX}\Big] \leq \exp\Big(s^2(b-a)^2/8\Big)$.*

Proof: Let $a \leq x \leq b$ and observe that x can be written as a convex combination of a and b. In particular, we have

$$x = \lambda a + (1-\lambda)b \quad \text{for} \quad \lambda = \frac{b-x}{b-a} \in [0,1]\,.$$

Since $s > 0$, the function $\exp(sx)$ is convex, and as such

$$e^{sx} \leq \frac{b-x}{b-a}e^{sa} + \frac{x-a}{b-a}e^{sb},$$

since we have that $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$ if $f(\cdot)$ is a convex function. Thus, for a random variable X, by linearity of expectation, we have

$$\begin{aligned}\mathbb{E}\Big[e^{sX}\Big] &\leq \mathbb{E}\left[\frac{b-X}{b-a}e^{sa} + \frac{X-a}{b-a}e^{sb}\right] = \frac{b-\mathbb{E}[X]}{b-a}e^{sa} + \frac{\mathbb{E}[X]-a}{b-a}e^{sb} \\ &= \frac{b}{b-a}e^{sa} - \frac{a}{b-a}e^{sb},\end{aligned}$$

since $\mathbb{E}[X] = 0$.

Next, set $p = -\dfrac{a}{b-a}$ and observe that $1 - p = 1 + \dfrac{a}{b-a} = \dfrac{b}{b-a}$ and

$$-ps(b-a) = -\left(-\frac{a}{b-a}\right)s(b-a) = sa.$$

As such, we have

$$\begin{aligned}\mathbb{E}\left[e^{sX}\right] &\leq (1-p)e^{sa} + pe^{sb} = (1-p+pe^{s(b-a)})e^{sa} \\ &= (1-p+pe^{s(b-a)})e^{-ps(b-a)} \\ &= \exp\left(-ps(b-a) + \ln\left(1-p+pe^{s(b-a)}\right)\right) = \exp(-pu + \ln(1-p+pe^{u})),\end{aligned}$$

for $u = s(b-a)$. Setting

$$\phi(u) = -pu + \ln(1-p+pe^{u}),$$

we thus have $\mathbb{E}\left[e^{sX}\right] \leq \exp(\phi(u))$. To prove the claim, we will show that $\phi(u) \leq u^2/8 = s^2(b-a)^2/8$.

To see that, expand $\phi(u)$ about zero using Taylor's expansion. We have

$$\phi(u) = \phi(0) + u\phi'(0) + \frac{1}{2}u^2\phi''(\theta) \tag{13.3}$$

where $\theta \in [0, u]$, and notice that $\phi(0) = 0$. Furthermore, we have

$$\phi'(u) = -p + \frac{pe^{u}}{1-p+pe^{u}},$$

and as such $\phi'(0) = -p + \frac{p}{1-p+p} = 0$. Now,

$$\phi''(u) = \frac{(1-p+pe^{u})pe^{u} - (pe^{u})^2}{(1-p+pe^{u})^2} = \frac{(1-p)pe^{u}}{(1-p+pe^{u})^2}.$$

For any $x, y \geq 0$, we have $(x+y)^2 \geq 4xy$ as this is equivalent to $(x-y)^2 \geq 0$. Setting $x = 1-p$ and $y = pe^{u}$, we have that

$$\phi''(u) = \frac{(1-p)pe^{u}}{(1-p+pe^{u})^2} \leq \frac{(1-p)pe^{u}}{4(1-p)pe^{u}} = \frac{1}{4}.$$

Plugging this into Eq. (13.3), we get that

$$\phi(u) \leq \frac{1}{8}u^2 = \frac{1}{8}(s(b-a))^2 \quad \text{and} \quad \mathbb{E}\left[e^{sX}\right] \leq \exp(\phi(u)) \leq \exp\left(\frac{1}{8}(s(b-a))^2\right),$$

as claimed. ■

Lemma 13.4.2. *Let X be a random variable. If $\mathbb{E}[X] = 0$ and $a \leq X \leq b$, then for any $s > 0$, we have*

$$\mathbb{P}[X > t] \leq \frac{\exp\left(\frac{s^2(b-a)^2}{8}\right)}{e^{st}}.$$

Proof: Using the same technique we used in proving Chernoff's inequality, we have that

$$\mathbb{P}[X > t] = \mathbb{P}\left[e^{sX} > e^{st}\right] \leq \frac{\mathbb{E}\left[e^{sX}\right]}{e^{st}} \leq \frac{\exp\left(\frac{s^2(b-a)^2}{8}\right)}{e^{st}}.$$ ■

Theorem 13.4.3 (Hoeffding's inequality). *Let $X_1, \ldots, X_n$ be independent random variables, where $X_i \in [a_i, b_i]$, for $i = 1, \ldots, n$. Then, for the random variable $S = X_1 + \cdots + X_n$ and any $\eta > 0$, we have*

$$\mathbb{P}\Big[\Big|S - \mathbb{E}[S]\Big| \geq \eta\Big] \leq 2\exp\left(-\frac{2\eta^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

Proof: Let $Z_i = X_i - \mathbb{E}[X_i]$, for $i = 1, \ldots, n$. Set $Z = \sum_{i=1}^n Z_i$, and observe that

$$\mathbb{P}[Z \geq \eta] = \mathbb{P}\Big[e^{sZ} \geq e^{s\eta}\Big] \leq \frac{\mathbb{E}[\exp(sZ)]}{\exp(s\eta)},$$

by Markov's inequality. Arguing as in the proof of Chernoff's inequality, we have

$$\mathbb{E}[\exp(sZ)] = \mathbb{E}\left[\prod_{i=1}^n \exp(sZ_i)\right] = \prod_{i=1}^n \mathbb{E}[\exp(sZ_i)] \leq \prod_{i=1}^n \exp\left(\frac{s^2(b_i - a_i)^2}{8}\right),$$

since the Z_is are independent and by Lemma 13.4.1. This implies that

$$\mathbb{P}[Z \geq \eta] \leq \exp(-s\eta) \prod_{i=1}^n e^{s^2(b_i - a_i)^2/8} = \exp\left(\frac{s^2}{8}\sum_{i=1}^n (b_i - a_i)^2 - s\eta\right).$$

The upper bound is minimized for $s = 4\eta/\left(\sum_i (b_i - a_i)^2\right)$, implying

$$\mathbb{P}[Z \geq \eta] \leq \exp\left(-\frac{2\eta^2}{\sum (b_i - a_i)^2}\right).$$

The claim now follows by the symmetry of the upper bound (i.e., apply the same proof to $-Z$). ■

13.5. Bibliographical notes

Some of the exposition here follows more or less the exposition in [MR95]. Exercise 13.6.1 (without the hint) is from [Mat99]. McDiarmid [McD89] provides a survey of Chernoff type inequalities, and Theorem 13.3.6 and Section 13.3 is taken from there (our proof has somewhat weaker constants).

A more general treatment of such inequalities and tools is provided by Dubhashi and Panconesi [DP09].

13.6. Exercises

Exercise 13.6.1 (Chernoff inequality is tight.). Let $S = \sum_{i=1}^n S_i$ be a sum of n independent random variables each attaining values $+1$ and -1 with equal probability. Let $P(n, \Delta) = \mathbb{P}[S > \Delta]$. Prove that for $\Delta \leq n/C$,

$$P(n, \Delta) \geq \frac{1}{C}\exp\left(-\frac{\Delta^2}{Cn}\right),$$

where C is a suitable constant. That is, the well-known Chernoff bound $P(n, \Delta) \leq \exp(-\Delta^2/2n))$ is close to the truth.

Exercise 13.6.2 (Chernoff inequality is tight by direct calculations.). For this question use only basic argumentation – do not use Stirling's formula, Chernoff inequality or any similar "heavy" machinery.

(A) Prove that $\sum_{i=0}^{n-k} \binom{2n}{i} \leq \frac{n}{4k^2} 2^{2n}$.

Hint: Consider flipping a coin $2n$ times. Write down explicitly the probability of this coin to have at most $n-k$ heads, and use Chebyshev inequality.

(B) Using (A), prove that $\binom{2n}{n} \geq 2^{2n}/4\sqrt{n}$ (which is a pretty good estimate).

(C) Prove that $\binom{2n}{n+i+1} = \left(1 - \frac{2i+1}{n+i+1}\right)\binom{2n}{n+i}$.

(D) Prove that $\binom{2n}{n+i} \leq \exp\left(\frac{-i(i-1)}{2n}\right)\binom{2n}{n}$.

(E) Prove that $\binom{2n}{n+i} \geq \exp\left(-\frac{8i^2}{n}\right)\binom{2n}{n}$.

(F) Using the above, prove that $\binom{2n}{n} \leq c\frac{2^{2n}}{\sqrt{n}}$ for some constant c (I got $c = 0.824...$ but any reasonable constant will do).

(G) Using the above, prove that

$$\sum_{i=t\sqrt{n}+1}^{(t+1)\sqrt{n}} \binom{2n}{n-i} \leq c2^{2n} \exp\left(-t^2/2\right).$$

In particular, conclude that when flipping fair coin $2n$ times, the probability to get less than $n - t\sqrt{n}$ heads (for t an integer) is smaller than $c' \exp\left(-t^2/2\right)$, for some constant c'.

(H) Let X be the number of heads in $2n$ coin flips. Prove that for any integer $t > 0$ and any $\delta > 0$ sufficiently small, it holds that $\mathbb{P}[X < (1-\delta)n] \geq \exp\left(-c''\delta^2 n\right)$, where c'' is some constant. Namely, the Chernoff inequality is tight in the worst case.

Exercise 13.6.3 (Tail inequality for geometric variables). Let $X_1, \ldots, X_m$ be m independent random variables with geometric distribution with probability p (i.e., $\mathbb{P}[X_i = j] = (1-p)^{j-1}p$). Let $Y = \sum_i X_i$, and let $\mu = \mathbb{E}[Y] = m/p$. Prove that $\mathbb{P}[Y \geq (1+\delta)\mu] \leq \exp\left(-m\delta^2/8\right)$.

References

[DP09] D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[Kel56] J. L. Kelly. A new interpretation of information rate. *Bell Sys. Tech. J.*, 35(4): 917–926, 1956.

[Mat99] J. Matoušek. *Geometric Discrepancy*. Vol. 18. Algorithms and Combinatorics. Springer, 1999.

[McD89] C. McDiarmid. *Surveys in Combinatorics*. Ed. by J. Siemons. Cambridge University Press, 1989. Chap. On the method of bounded differences.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

Chapter 14

Applications of Chernoff's Inequality

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

14.1. QuickSort is Quick

We revisit **QuickSort**. We remind the reader that the running time of **QuickSort** is proportional to the number of comparisons performed by the algorithm. Next, consider an arbitrary element u being sorted. Consider the ith level recursive subproblem that contains u, and let S_i be the set of elements in this subproblems. We consider u to be *successful* in the ith level, if $|S_{i+1}| \leq |S_i|/2$. Namely, if u is successful, then the next level in the recursion involving u would include a considerably smaller subproblem. Let X_i be the indicator variable which is 1 if u is successful.

We first observe that if **QuickSort** is applied to an array with n elements, then u can be successful at most $T = \lceil \lg n \rceil$ times, before the subproblem it participates in is of size one, and the recursion stops. Thus, consider the indicator variable X_i which is 1 if u is successful in the ith level, and zero otherwise. Note that the X_is are independent, and $\mathbb{P}[X_i = 1] = 1/2$.

If u participates in v levels, then we have the random variables $X_1, X_2, \ldots, X_v$. To make things simpler, we will extend this series by adding independent random variables, such that $\mathbb{P}[`]X_i = 1 = 1/2$, for $i \geq v$. Thus, we have an infinite sequence of independent random variables, that are 0/1 and get 1 with probability $1/2$. The question is how many elements in the sequence we need to read, till we get T ones.

Lemma 14.1.1. *Let $X_1, X_2, \ldots$ be an infinite sequence of independent random 0/1 variables. Let M be an arbitrary parameter. Then the probability that we need to read more than $2M + 4t\sqrt{M}$ variables of this sequence till we collect M ones is at most $2\exp\left(-t^2\right)$, for $t \leq \sqrt{M}$. If $t \geq \sqrt{M}$ then this probability is at most $2\exp\left(-t\sqrt{M}\right)$.*

Proof: Consider the random variable $Y = \sum_{i=1}^{L} X_i$, where $L = 2M + 4t\sqrt{M}$. Its expectation is $L/2$, and using the Chernoff inequality, we get

$$\begin{aligned}\alpha = \mathbb{P}[Y \leq M] &\leq \mathbb{P}[|Y - L/2| \geq L/2 - M] \leq 2\exp\left(-\tfrac{2}{L}(L/2 - M)^2\right) \\ &\leq 2\exp\left(-2(M + 2t\sqrt{M} - M)^2/L\right) \leq 2\exp\left(-2(2t\sqrt{M})^2/L\right) = 2\exp\left(-\frac{8t^2M}{L}\right),\end{aligned}$$

by Corollary 13.1.9. For $t \leq \sqrt{M}$ we have that $L = 2M + 4t\sqrt{M} \leq 8M$, as such in this case $\mathbb{P}[Y \leq M] \leq 2\exp\left(-t^2\right)$.

If $t \geq \sqrt{M}$, then $\alpha = 2\exp\left(-\dfrac{8t^2M}{2M + 4t\sqrt{M}}\right) \leq 2\exp\left(-\dfrac{8t^2M}{6t\sqrt{M}}\right) \leq 2\exp\left(-t\sqrt{M}\right)$. ■

Going back to the **QuickSort** problem, we have that if we sort n elements, the probability that u will participate in more than $L = (4+c)\lceil \lg n \rceil = 2\lceil \lg n \rceil + 4c\sqrt{\lg n}\sqrt{\lg n}$, is smaller than $2\exp\left(-c\sqrt{\lg n}\sqrt{\lg n}\right) \leq 1/n^c$, by Lemma 14.1.1. There are n elements being sorted, and as such the probability that any element would participate in more than $(4+c+1)\lceil \lg n \rceil$ recursive calls is smaller than $1/n^c$.

Lemma 14.1.2. *For any $c > 0$, the probability that* **QuickSort** *performs more than $(6+c)n \lg n$, is smaller than $1/n^c$.*

14.2. How many times can the minimum change?

Let $\Pi = \pi_1 \ldots \pi_n$ be a random permutation of $\{1, \ldots, n\}$. Let $\mathcal{E}_i$ be the event that π_i is the minimum number seen so far as we read Π; that is, $\mathcal{E}_i$ is the event that $\pi_i = \min_{k=1}^{i} \pi_k$. Let X_i be the indicator variable that is one if $\mathcal{E}_i$ happens. We already seen, and it is easy to verify, that $\mathbb{E}[X_i] = 1/i$. We are interested in how many times the minimum might change[1]; that is $Z = \sum_i X_i$, and how concentrated is the distribution of Z. The following is maybe surprising.

Lemma 14.2.1. *The events $\mathcal{E}_1, \ldots, \mathcal{E}_n$ are independent (as such, the variables $X_1, \ldots, X_n$ are independent).*

Proof: Exercise. ■

Theorem 14.2.2. *Let $\Pi = \pi_1 \ldots \pi_n$ be a random permutation of $1, \ldots, n$, and let Z be the number of times, that π_i is the smallest number among $\pi_1, \ldots, \pi_i$, for $i = 1, \ldots, n$. Then, we have that for $t \geq 2e$ that $\mathbb{P}[Z > t \ln n] \leq 1/n^{t \ln 2}$, and for $t \in [1, 2e]$, we have that $\mathbb{P}[Z > t \ln n] \leq 1/n^{(t-1)^2/4}$.*

Proof: Follows readily from Chernoff's inequality, as $Z = \sum_i X_i$ is a sum of independent indicator variables, and, since by linearity of expectations, we have

$$\mu = \mathbb{E}[Z] = \sum_i \mathbb{E}[X_i] = \sum_{i=1}^{n} \frac{1}{i} \geq \int_{x=1}^{n+1} \frac{1}{x} \mathrm{d}x = \ln(n+1) \geq \ln n.$$

Next, we set $\delta = t - 1$, and use Theorem 13.2.1. ■

14.3. Routing in a parallel computer

Let G be a graph of a network, where every node is a processor. The processor communicate by sending packets on the edges. Let $[0, \ldots, N-1]$ denote be vertices (i.e., processors) of G, where $N = 2^n$, and G is the hypercube. As such, each processor is identified by a binary string $b_1 b_2 \ldots b_n \in \{0,1\}^n$. Two nodes are connected if their binary string differs only in a single bit. Namely, G is the binary ***hypercube*** over n bits.

We want to investigate the best routing strategy for this network topology. We assume that every processor need to send a message to a single other processor. This is represented by a permutation π, and we would like to figure out how to send the messages encoded by the permutation while creating minimum delay/congestion.

Specifically, in our model, every edge has a FIFO queue[2] of the packets it has to transmit. At every clock tick, the message in front of the queue get sent. All the processors start sending their packets (to the destination specified by the permutation) in the same time.

[1] The answer, my friend, is blowing in the permutation.

[2] First in, first out queue. I sure hope you already knew that.

RandomRoute($v_0, \ldots, v_{N-1}$)

// v_i: Packet at node i to be routed to node $d(i)$.

(i) Pick a *random* intermediate destination $\sigma(i)$ from $[1, \ldots, N]$. Packet v_i travels to $\sigma(i)$.

// Here random sampling is done with ***replacement***.

// Several packets might travel to the same destination.

(ii) Wait till all the packets arrive to their intermediate destination.

(iii) Packet v_i travels from $\sigma(i)$ to its destination $d(i)$.

Figure 14.1: The routing algorithm

A routing scheme is ***oblivious*** if every node that has to forward a packet, inspect the packet, and depending only on the content of the packet decides how to forward it. That is, such a routing scheme is local in nature, and does not take into account other considerations. Oblivious routing is of course a bad idea – it ignores congestion in the network, and might insist routing things through regions of the hypercube that are "gridlocked".

Theorem 14.3.1 ([KKT91]). *For any deterministic oblivious permutation routing algorithm on a network of N nodes each of out-degree n, there is a permutation for which the routing of the permutation takes $\Omega\left(\sqrt{N/n}\right)$ units of time (i.e., ticks).*

Proof: (Sketch.) The above is implied by a nice averaging argument – construct, for every possible destination, the routing tree of all packets to this specific node. Argue that there must be many edges in this tree that are highly congested in this tree (which is NOT the permutation routing we are looking for!). Now, by averaging, there must be a single edge that is congested in "many" of these trees. Pick a source-destination pair from each one of these trees that uses this edge, and complete it into a full permutation in the natural way. Clearly, the congestion of the resulting permutation is high. For the exact details see [KKT91]. ■

How do we send a packet? We use ***bit fixing***. Namely, the packet from the i node, always go to the current adjacent node that have the first different bit as we scan the destination string $d(i)$. For example, packet from (0000) going to (1101), would pass through (1000), (1100), (1101).

The routing algorithm. We assume each edge have a FIFO queue. The routing algorithm is depicted in Figure 14.1.

14.3.1. Analysis

We analyze only step (i) in the algorithm, as (iii) follows from the same analysis. In the following, let ρ_i denote the route taken by v_i in (i).

Exercise 14.3.2. Once a packet v_j that travel along a path ρ_j can not leave a path ρ_i, and then join it again later. Namely, $\rho_i \cap \rho_j$ is (maybe an empty) path.

Lemma 14.3.3. *Let the route of a message* **c** *follow the sequence of edges $\pi = (e_1, e_2, \ldots, e_k)$. Let S be the set of packets whose routes pass through at least one of $(e_1, \ldots, e_k)$. Then, the delay incurred by* **c** *is at most $|S|$.*

Proof: A packet in S is said to leave π at that time step at which it traverses an edge of π for the last time. If a packet is ready to follow edge e_j at time t, we define its *lag* at time t to be $t - j$. The lag of **c** is initially zero, and

the delay incurred by **c** is its lag when it traverse e_k. We will show that each step at which the lag of **c** increases by one can be charged to a distinct member of S.

We argue that if the lag of **c** reaches $\ell+1$, some packet in S leaves π with lag ℓ. When the lag of **c** increases from ℓ to $\ell+1$, there must be at least one packet (from S) that wishes to traverse the same edge as **c** at that time step, since otherwise **c** would be permitted to traverse this edge and its lag would not increase. Thus, S contains at least one packet whose lag reach the value ℓ.

Let τ be the last time step at which any packet in S has lag ℓ. Thus there is a packet **d** ready to follow edge e_μ at τ, such that $\tau-\mu=\ell$. We argue that some packet of S leaves π at time τ – this establishes the lemma since once a packet leaves π, it would never join it again and as such will never again delay **c**.

Since **d** is ready to follow e_μ at time τ, some packet ω (which may be **d** itself) in S traverses e_μ at time τ. Now ω must leave π at time τ – if not, some packet will follow $e_{\mu+1}$ at step $\mu+1$ with lag ℓ. But this violates the maximality of τ. We charge to ω the increase in the lag of **c** from ℓ to $\ell+1$. Since ω leaves π, it will never be charged again. Thus, each member of S whose route intersects π is charge for at most one delay, establishing the lemma. ■

Let H_{ij} be an indicator variable that is 1 if ρ_i and ρ_j share an edge, and 0 otherwise. The total delay for v_i is at most $\leq \sum_j H_{ij}$.

Crucially, for a fixed i, the variables $H_{i1},\ldots,H_{iN}$ are independent. Indeed, imagine first picking the destination of v_i, and let the associated path be ρ_i. Now, pick the destinations of all the other packets in the network. Since the sampling of destinations is done with replacements, whether or not the path ρ_j of v_j intersects ρ_i, is independent of whether ρ_k intersects ρ_i. Of course, the probabilities $\mathbb{P}[H_{ij}=1]$ and $\mathbb{P}[H_{ik}=1]$ are probably different. Confusingly, however, $H_{11},\ldots,H_{NN}$ are not independent. Indeed, imagine k and j being close vertices on the hypercube. If $H_{ij}=1$ then intuitively it means that ρ_i is traveling close to the vertex v_j, and as such there is a higher probability that $H_{ik}=1$.

Let

$$\rho_i=(e_1,\ldots,e_k),$$

and let $T(e)$ be the number of packets (i.e., paths) that pass through e. We have that

$$\sum_{j=1}^{N} H_{ij} \leq \sum_{j=1}^{k} T(e_j) \quad \text{and thus} \quad \mathbb{E}\left[\sum_{j=1}^{N} H_{ij}\right] \leq \mathbb{E}\left[\sum_{j=1}^{k} T(e_j)\right].$$

Because of symmetry, the variables $T(e)$ have the same distribution for all the edges of G. On the other hand, the expected length of a path is $n/2$, there are N packets, and there are $Nn/2$ edges[3]. We conclude

$$\mathbb{E}[T(e)] = \frac{\text{Total length of paths}}{\#\text{ of edges in graph}} = \frac{N(n/2)}{N(n/2)} = 1$$

$=1$. Thus, for $k=|\rho_i|$, we have

$$\mu = \mathbb{E}\Big[\sum_{j=1}^{N} H_{ij}\Big] \leq \mathbb{E}\Big[\sum_{j=1}^{k} T(e_j)\Big] = \mathbb{E}\Big[\mathbb{E}\Big[\sum_{j=1}^{k} T(e_j) \;\Big|\; \rho_i\Big]\Big] = \mathbb{E}\Big[\sum_{j=1}^{|\rho_i|} \mathbb{E}\Big[T(e_j) \;\Big|\; \rho_i\Big]\Big] = \mathbb{E}\Big[\sum_{j=1}^{|\rho_i|} 1\Big] = \mathbb{E}\Big[|\rho_i|\Big] = \frac{n}{2}.$$

By the Chernoff inequality, specifically Lemma 13.2.8, we have

$$\mathbb{P}\left[\sum_j H_{ij} > 7n\right] \leq \mathbb{P}\left[\sum_j H_{ij} > (1+13)\mu\right] < 2^{-13\mu} \leq 2^{-6n}.$$

Since there are $N=2^n$ packets, we know that with probability $\leq 2^{-5n}$ all packets arrive to their temporary destination in a delay of most $7n$.

[3] Indeed, the hypercube has N vertices, all of degree n. As such, the number of edges is $Nn/2$.

Theorem 14.3.4. *Each packet arrives to its destination in $\leq 14n$ stages, in probability at least $1 - 1/N$ (note that this is very conservative).*

14.4. Faraway Strings

Consider the Hamming distance between binary strings. It is natural to ask how many strings of length n can one have, such that any pair of them, is of Hamming distance at least t from each other. Consider two random strings, generated by picking at each bit randomly and independently. Thus, $\mathbb{E}[d_H(x, y)] = n/2$, where $d_H(x, y)$ denote the hamming distance between x and y. In particular, using the Chernoff inequality, specifically Corollary 13.1.9, we have that

$$\mathbb{P}[d_H(x, y) \leq n/2 - \Delta] \leq \exp\Big(-2\Delta^2/n\Big).$$

Next, consider generating M such string, where the value of M would be determined shortly. Clearly, the probability that any pair of strings are at distance at most $n/2 - \Delta$, is

$$\alpha \leq \binom{M}{2} \exp\Big(-2\Delta^2/n\Big) < M^2 \exp\Big(-2\Delta^2/n\Big).$$

If this probability is smaller than one, then there is some probability that all the M strings are of distance at least $n/2 - \Delta$ from each other. Namely, there exists a set of M strings such that every pair of them is far. We used here the fact that if an event has probability larger than zero, then it exists. Thus, set $\Delta = n/4$, and observe that

$$\alpha < M^2 \exp\Big(-2n^2/16n\Big) = M^2 \exp(-n/8).$$

Thus, for $M = \exp(n/16)$, we have that $\alpha < 1$. We conclude:

Lemma 14.4.1. *There exists a set of $\exp(n/16)$ binary strings of length n, such that any pair of them is at Hamming distance at least $n/4$ from each other.*

This is our first introduction to the beautiful technique known as the probabilistic method — we will hear more about it later in the course.

This result has also interesting interpretation in the Euclidean setting. Indeed, consider the sphere $\mathbb{S}$ of radius $\sqrt{n}/2$ centered at $(1/2, 1/2, \ldots, 1/2) \in \mathbb{R}^n$. Clearly, all the vertices of the binary hypercube $\{0, 1\}^n$ lie on this sphere. As such, let P be the set of points on $\mathbb{S}$ that exists according to Lemma 14.4.1. A pair p, q of points of P have *Euclidean* distance at least $\sqrt{d_H(p, q)} = \sqrt{n/4} = \sqrt{n}/2$ from each other. We conclude:

Lemma 14.4.2. *Consider the unit hypersphere $\mathbb{S}$ in $\mathbb{R}^n$. The sphere $\mathbb{S}$ contains a set Q of points, such that each pair of points is at (Euclidean) distance at least one from each other, and $|Q| \geq \exp(n/16)$.*

Proof: Take the above point set, and scale it down by a factor of $\sqrt{n}/2$. ■

14.5. Bibliographical notes

Section 14.3 is based on Section 4.2 in [MR95]. A similar result to Theorem 14.3.4 is known for the case of the wrapped butterfly topology (which is similar to the hypercube topology but every node has a constant degree, and there is no clear symmetry). The interested reader is referred to [MU05].

14.6. Exercises

Exercise 14.6.1 (More binary strings. More!). To some extent, Lemma 14.4.1 is somewhat silly, as one can prove a better bound by direct argumentation. Indeed, for a fixed binary string x of length n, show a bound on the number of strings in the Hamming ball around x of radius $n/4$ (i.e., binary strings of distance at most $n/4$ from x). (Hint: interpret the special case of the Chernoff inequality as an inequality over binomial coefficients.)

Next, argue that the greedy algorithm which repeatedly pick a string which is in distance $\geq n/4$ from all strings picked so far, stops after picking at least $\exp(n/8)$ strings.

References

[KKT91] C. Kaklamanis, D. Krizanc, and T. Tsantilas. Tight bounds for oblivious routing in the hypercube. *Math. sys. theory*, 24(1): 223–232, 1991.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

[MU05] M. Mitzenmacher and U. Upfal. *Probability and Computing – randomized algorithms and probabilistic analysis*. Cambridge, 2005.

Chapter 15

Min Cut

> **To acknowledge the corn** - This purely American expression means to admit the losing of an argument, especially in regard to a detail; to retract; to admit defeat. It is over a hundred years old. Andrew Stewart, a member of Congress, is said to have mentioned it in a speech in 1828. He said that haystacks and cornfields were sent by Indiana, Ohio and Kentucky to Philadelphia and New York. Charles A. Wickliffe, a member from Kentucky questioned the statement by commenting that haystacks and cornfields could not walk. Stewart then pointed out that he did not mean literal haystacks and cornfields, but the horses, mules, and hogs for which the hay and corn were raised. Wickliffe then rose to his feet, and said, "Mr. Speaker, I acknowledge the corn".
>
> Funk, Earle, A Hog on Ice and Other Curious Expressions

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

15.1. Branching processes – Galton-Watson Process

15.1.1. The problem

In the 19th century, Victorians were worried that aristocratic surnames were disappearing, as family names passed on only through the male children. As such, a family with no male children had its family name disappear. So, imagine the number of male children of a person is an independent random variable $X \in \{0, 1, 2, \ldots\}$. Starting with a single person, its family (as far as male children are concerned) is a random tree with the degree of a node being distributed according to X. We continue recursively in constructing this tree, again, sampling the number of children for each current leaf according to the distribution of X. It is not hard to see that a family disappears if $\mathbb{E}[X] \leq 1$, and it has a constant probability of surviving if $\mathbb{E}[X] > 1$.

Francis Galton asked the question of what is the probability of such a blue-blood family name to survive, and this question was answered by Henry William Watson [WG75]. The Victorians were worried about strange things, see [Gre69] for a provocatively titled article from the period, and [Ste12] for a more recent take on this issue.

Of course, since infant mortality is dramatically down (as is the number of aristocrat males dying to maintain the British empire), the probability of family names to disappear is now much lower than it was in the 19th century. Interestingly, countries with family names that were introduced long time ago have very few surnames (i.e., Korean have 250 surnames, and three surnames form 45% of the population). On the other hand, countries that introduced surnames more recently have dramatically more surnames (for example, the Dutch have surnames only for the last 200 years, and there are $68,000$ different family names).

Here we are going to look on a very specific variant of this problem. Imagine that starting with a single male. A male has exactly two children, and one of them is a male with probability half (i.e., the Y-chromosome is being passed only to its male children). As such, the natural question is what is the probability that h generations down, there is a male decedent that all his ancestors are male (i.e., it caries the original family name, and the original Y-chromosome).

15.1.2. On coloring trees

Let T_h be a complete binary tree of height h. We randomly color its edges by black and white. Namely, for each edge we independently choose its color to be either black or white, with equal probability (say, black indicates the child is male). We are interested in the event that there exists a path from the root of T_h to one of its leafs, that is all black. Let $\mathcal{E}_h$ denote this event, and let $\rho_h = \mathbb{P}[\mathcal{E}_h]$. Observe that $\rho_0 = 1$ and $\rho_1 = 3/4$ (see below).

To bound this probability, consider the root u of T_h and its two children u_l and u_r. The probability that there is a black path from u_l to one of its children is ρ_{h-1}, and as such, the probability that there is a black path from u through u_l to a leaf of the subtree of u_l is $\mathbb{P}[\text{the edge } uu_l \text{ is colored black}] \cdot \rho_{h-1} = \rho_{h-1}/2$. As such, the probability that there is no black path through u_l is $1 - \rho_{h-1}/2$. As such, the probability of not having a black path from u to a leaf (through either children) is $(1 - \rho_{h-1}/2)^2$. In particular, there desired probability, is the complement; that is

$$\rho_h = 1 - \left(1 - \frac{\rho_{h-1}}{2}\right)^2 = \frac{\rho_{h-1}}{2}\left(2 - \frac{\rho_{h-1}}{2}\right) = \rho_{h-1} - \frac{\rho_{h-1}^2}{4} = f(\rho_{h-1}) \qquad \text{for} \qquad f(x) = x - x^2/4.$$

The starting values are $\rho_0 = 1$, and $\rho_1 = 3/4$.

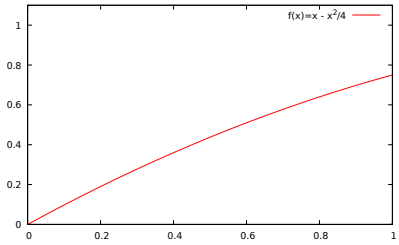


Figure 15.1: A graph of the function $f(x) = x - x^2/4$.

Lemma 15.1.1. *We have that $\rho_h \geq 1/(h+1)$.*

Proof: (Feel free to skip reading.) The proof is by induction. For $h = 1$, we have $\rho_1 = 3/4 \geq 1/(1+1)$.

Observe that $\rho_h = f(\rho_{h-1})$ for $f(x) = x - x^2/4$, and $f'(x) = 1 - x/2$. As such, $f'(x) > 0$ for $x \in [0, 1]$ and $f(x)$ is increasing in the range $[0, 1]$. As such, by induction, we have that

$$\rho_h = f(\rho_{h-1}) \geq f\left(\frac{1}{(h-1)+1}\right) = \frac{1}{h} - \frac{1}{4h^2}.$$

We need to prove that $\rho_h \geq 1/(h+1)$, which is implied by the above if

$$\frac{1}{h} - \frac{1}{4h^2} \geq \frac{1}{h+1} \quad \Leftrightarrow \quad 4h(h+1) - (h+1) \geq 4h^2 \quad \Leftrightarrow \quad 4h^2 + 4h - h - 1 \geq 4h^2 \quad \Leftrightarrow \quad 3h \geq 1,$$

which trivially holds. ■

Lemma 15.1.2. *We have that* $\rho_h = O(1/h)$.

Proof: (Feel free to skip reading.) We prove the claim for infinite number of values of h – the claim then follows for all h by fiddling with the constants. The claim trivially holds for small values of h. For any $j > 0$, let h_j be the minimal index such that $\rho_{h_j} \leq 1/2^j$. It is easy to verify that $\rho_{h_j} \geq 1/2^{j+1}$. We claim (mysteriously) that

$$h_{j+1} - h_j \leq \frac{\rho_{h_j} - \rho_{h_{j+1}}}{(\rho_{h_{j+1}})^2/4}.$$

Indeed, ρ_{k+1} is the number resulting from removing $\rho_k^2/4$ from ρ_k. Namely, the sequence $\rho_1, \rho_2, \ldots$ is a monotonically decreasing sequence of numbers in the interval $[0, 1]$, where the gaps between consecutive numbers decreases. In particular, to get from ρ_{h_j} to $\rho_{h_{j+1}}$, the gaps used were of size at least $\Delta = (\rho_{h_{j+1}})^2$, which means that there are at least $(\rho_{h_j} - \rho_{h_{j+1}})/\Delta - 1$ numbers in the series between these two elements. As such, since $\rho_{h_j} \leq 1/2^j$ and $\rho_{h_{j+1}} \geq 1/2^{j+2}$, we have

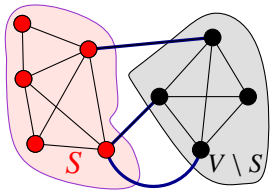
$$h_{j+1} - h_j \leq \frac{\rho_{h_j} - \rho_{h_{j+1}}}{(\rho_{h_{j+1}})^2/4} \leq \frac{1/2^j - 1/2^{j+2}}{1/2^{2(j+2)+2}} \leq 2^{2j+6}/2^j = 2^{j+6}.$$

This implies that $h_j \leq (h_j - h_{j-1}) + (h_{j-1} - h_{j-2}) + \cdots + (h_1 - h_0) \leq 2^{j+6}$. As such, we have $\rho_{h_j} \leq 1/2^j \leq 2^6/2^{j+6} \leq 2^6/h_j$, which implies the claim. ■

15.2. Min Cut

15.2.1. Problem Definition

Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be an undirected graph with n vertices and m edges. We are interested in ***cuts*** in G.



Definition 15.2.1. A ***cut*** in G is a partition of the vertices of V into two sets S and $\mathsf{V} \setminus S$, where the edges of the cut are

$$(S, \mathsf{V} \setminus S) = \left\{ uv \;\middle|\; u \in S, v \in \mathsf{V} \setminus S, \text{ and } uv \in E \right\},$$

where $S \neq \emptyset$ and $\mathsf{V} \setminus S \neq \emptyset$. We will refer to the number of edges in the cut $(S, \mathsf{V} \setminus S)$ as the *size of the cut*. For an example of a cut, see figure on the right.

We are interested in the problem of computing the ***minimum cut*** (i.e., ***mincut***), that is, the cut in the graph with minimum cardinality. Specifically, we would like to find the set $S \subseteq \mathsf{V}$ such that $(S, \mathsf{V} \setminus S)$ is as small as possible, and S is neither empty nor $\mathsf{V} \setminus S$ is empty.

15.2.2. Some Definitions

We remind the reader of the following concepts. The ***conditional probability*** of X given Y is $\mathbb{P}\left[X = x \;\middle|\; Y = y\right] = \mathbb{P}[(X = x) \cap (Y = y)]/\mathbb{P}[Y = y]$. An equivalent, useful restatement of this is that

$$\mathbb{P}[(X = x) \cap (Y = y)] = \mathbb{P}\left[X = x \;\middle|\; Y = y\right] \cdot \mathbb{P}[Y = y]. \tag{15.1}$$

The following is easy to prove by induction using Eq. (15.1).

Lemma 15.2.2. *Let* $\mathcal{E}_1, \ldots, \mathcal{E}_n$ *be* n *events which are not necessarily independent. Then,*

$$\mathbb{P}[\cap_{i=1}^n \mathcal{E}_i] = \mathbb{P}[\mathcal{E}_1] * \mathbb{P}\left[\mathcal{E}_2 \;\middle|\; \mathcal{E}_1\right] * \mathbb{P}\left[\mathcal{E}_3 \;\middle|\; \mathcal{E}_1 \cap \mathcal{E}_2\right] * \ldots * \mathbb{P}\left[\mathcal{E}_n \;\middle|\; \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{n-1}\right].$$

15.3. The Algorithm

The basic operation used by the algorithm is ***edge contraction***, depicted in Figure 15.2. We take an edge $e = xy$ in G and merge the two vertices into a single vertex. The new resulting graph is denoted by G/xy. Note, that we remove self loops created by the contraction. However, since the resulting graph is no longer a regular graph, it has parallel edges – namely, it is a multi-graph. We represent a multi-graph, as a regular graph with multiplicities on the edges. See Figure 15.3.

The edge contraction operation can be implemented in $O(n)$ time for a graph with n vertices. This is done by merging the adjacency lists of the two vertices being contracted, and then using hashing to do the fix-ups (i.e., we need to fix the adjacency list of the vertices that are connected to the two vertices).

Note, that the cut is now computed counting multiplicities (i.e., if e is in the cut and it has weight w, then the contribution of e to the cut weight is w).

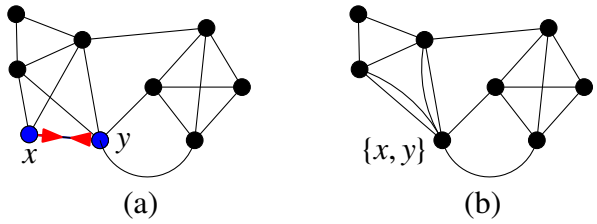


Figure 15.2: (a) A contraction of the edge xy. (b) The resulting graph.

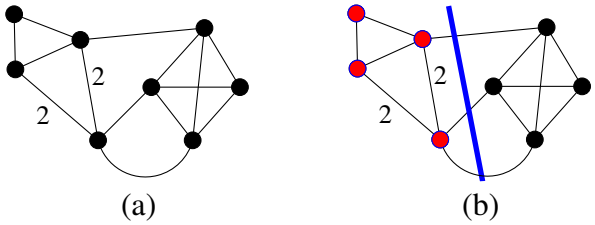


Figure 15.3: (a) A multi-graph. (b) A minimum cut in the resulting multi-graph.

Observation 15.3.1. *A set of vertices in G/xy corresponds to a set of vertices in the graph G. Thus a cut in G/xy always corresponds to a valid cut in G. However, there are cuts in G that do not exist in G/xy. For example, the cut $S = \{x\}$, does not exist in G/xy. As such, the size of the minimum cut in G/xy is at least as large as the minimum cut in G (as long as G/xy has at least one edge). Since any cut in G/xy has a corresponding cut of the same cardinality in G.*

Our algorithm works by repeatedly performing edge contractions. This is beneficial as this shrinks the underlying graph, and we would compute the cut in the resulting (smaller) graph. An "extreme" example of this, is shown in Figure 15.4, where we contract the graph into a single edge, which (in turn) corresponds to a cut in the original graph. (It might help the reader to think about each vertex in the contracted graph, as corresponding to a connected component in the original graph.)

Figure 15.4 also demonstrates the problem with taking this approach. Indeed, the resulting cut is not the minimum cut in the graph.

So, why did the algorithm fail to find the minimum cut in this case?[①] The failure occurs because of the contraction at Figure 15.4 (e), as we had contracted an edge in the minimum cut. In the new graph, depicted in Figure 15.4 (f), there is no longer a cut of size 3, and all cuts are of size 4 or more. Specifically, the algorithm succeeds only if it does not contract an edge in the minimum cut.

Observation 15.3.2. *Let $e_1, \ldots, e_{n-2}$ be a sequence of edges in G, such that none of them is in the minimum cut, and such that $\mathsf{G}' = G/\{e_1, \ldots, e_{n-2}\}$ is a single multi-edge. Then, this multi-edge corresponds to a minimum cut in G.*

Note, that the claim in the above observation is only in one direction. We might be able to still compute a minimum cut, even if we contract an edge in a minimum cut, the reason being that a minimum cut is not

[①] Naturally, if the algorithm had succeeded in finding the minimum cut, this would have been **our** success.

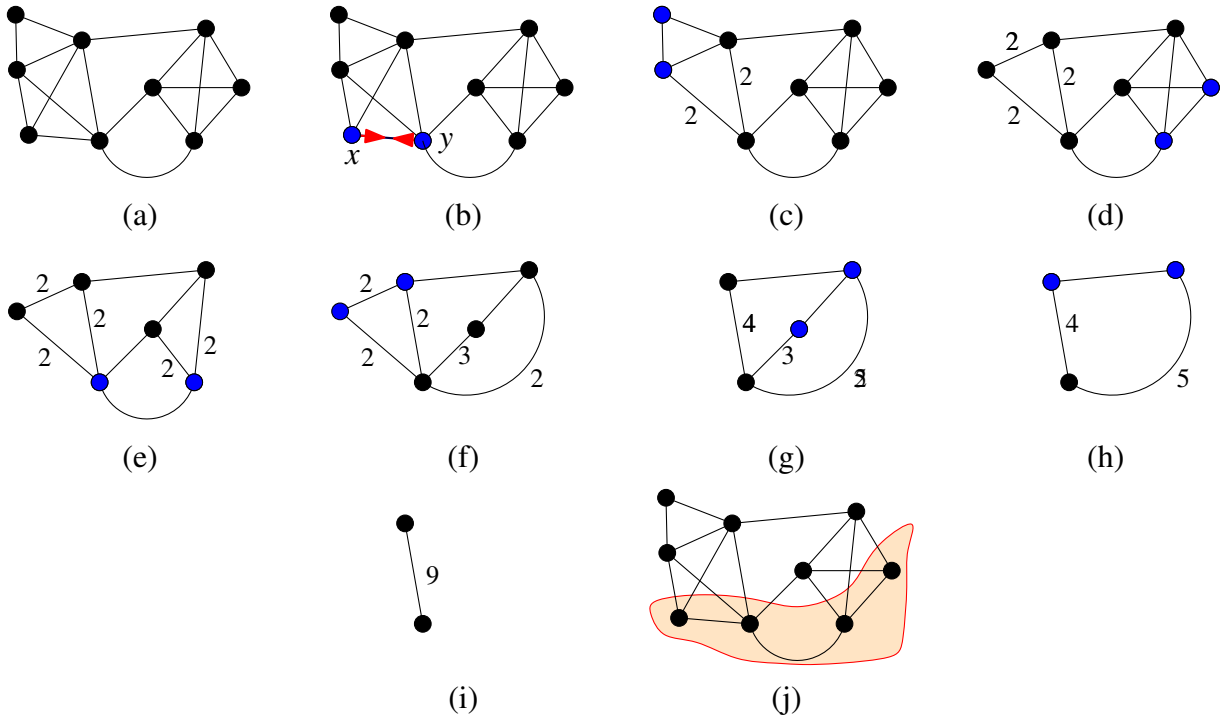


Figure 15.4: (a) Original graph. (b)–(j) a sequence of contractions in the graph, and (h) the cut in the original graph, corresponding to the single edge in (h). Note that the cut of (h) is not a mincut in the original graph.

Algorithm MinCut(G)
  $\mathsf{G}_0 \leftarrow G$
  $i = 0$
  while G_i has more than two vertices **do**
    Pick randomly an edge e_i from the edges of G_i
    $\mathsf{G}_{i+1} \leftarrow G_i / e_i$
    $i \leftarrow i + 1$
  Let $(S, \mathsf{V} \setminus S)$ be the cut in the original graph
    corresponding to the single edge in G_i
  return $(S, \mathsf{V} \setminus S)$.

Figure 15.5: The minimum cut algorithm.

unique. In particular, another minimum cut might survived the sequence of contractions that destroyed other minimum cuts.

Using Observation 15.3.2 in an algorithm is problematic, since the argumentation is circular, how can we find a sequence of edges that are not in the cut without knowing what the cut is? The way to slice the Gordian knot here, is to randomly select an edge at each stage, and contract this random edge.

See Figure 15.5 for the resulting algorithm **MinCut**.

15.3.1. Analysis

15.3.1.1. The probability of success

Naturally, if we are extremely lucky, the algorithm would never pick an edge in the mincut, and the algorithm would succeed. The ultimate question here is what is the probability of success. If it is relatively "large" then this algorithm is useful since we can run it several times, and return the best result computed. If on the other hand, this probability is tiny, then we are working in vain since this approach would not work.

Lemma 15.3.3. *If a graph G has a minimum cut of size k and G has n vertices, then $|E(G)| \geq \frac{kn}{2}$.*

Proof: Each vertex degree is at least k, otherwise the vertex itself would form a minimum cut of size smaller than k. As such, there are at least $\sum_{v \in \mathsf{V}} \text{degree}(v)/2 \geq nk/2$ edges in the graph. ■

Lemma 15.3.4. *Fix a specific minimum cut $C = (S, \overline{S})$ in the graph. If we pick in random an edge e from a graph G, uniformly at random, then with probability at most $2/n$ it belongs to the minimum cut C.*

Proof: There are at least $nk/2$ edges in the graph and exactly k edges in the minimum cut. Thus, the probability of picking an edge from the minimum cut is smaller then $k/(nk/2) = 2/n$. ■

The following lemma shows (surprisingly) that **MinCut** succeeds with reasonable probability.

Lemma 15.3.5. **MinCut** *outputs the mincut with probability* $\geq \dfrac{2}{n(n-1)}$.

7

Proof: Let $\mathcal{E}_i$ be the event that e_i is not in the minimum cut of G_i. By Observation 15.3.2, **MinCut** outputs the minimum cut if the events $\mathcal{E}_0, \ldots, \mathcal{E}_{n-3}$ all happen (namely, all edges picked are outside the minimum cut).

By Lemma 15.3.4, it holds $\mathbb{P}\Big[\mathcal{E}_i \;\Big|\; \mathcal{E}_0 \cap \mathcal{E}_1 \cap \ldots \cap \mathcal{E}_{i-1}\Big] \geq 1 - \dfrac{2}{|\mathsf{V}(G_i)|} = 1 - \dfrac{2}{n-i}$. Implying that

$$\Delta = \mathbb{P}[\mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{n-3}] = \mathbb{P}[\mathcal{E}_0] \cdot \mathbb{P}\Big[\mathcal{E}_1 \;\Big|\; \mathcal{E}_0\Big] \cdot \mathbb{P}\Big[\mathcal{E}_2 \;\Big|\; \mathcal{E}_0 \cap \mathcal{E}_1\Big] \cdot \ldots \cdot \mathbb{P}\Big[\mathcal{E}_{n-3} \;\Big|\; \mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{n-4}\Big].$$

As such, we have

$$\begin{aligned}
\Delta &\geq \prod_{i=0}^{n-3} \left(1 - \frac{2}{n-i}\right) = \prod_{i=0}^{n-3} \frac{n-i-2}{n-i} \\
&= \frac{\cancel{n-2}}{n} * \frac{\cancel{n-3}}{n-1} * \frac{\cancel{n-4}}{\cancel{n-2}} * \frac{\cancel{n-5}}{\cancel{n-3}} * \frac{\cancel{n-6}}{\cancel{n-4}} * \cdots * \frac{\cancel{3}}{5} * \frac{2}{\cancel{4}} * \frac{1}{\cancel{3}} \\
&= \frac{2}{n(n-1)}.
\end{aligned}$$

■

15.3.1.2. Running time analysis.

Observation 15.3.6. **MinCut** *runs in* $O(n^2)$ *time.*

Observation 15.3.7. *The algorithm always outputs a cut, and the cut is not smaller than the minimum cut.*

Definition 15.3.8. (informal) Amplification is the process of running an experiment again and again till the things we want to happen, with good probability, do happen.

Let **MinCutRep** be the algorithm that runs **MinCut** $n(n-1)$ times and return the minimum cut computed in all those independent executions of **MinCut**.

Lemma 15.3.9. *The probability that* **MinCutRep** *fails to return the minimum cut is* < 0.14.

Proof: The probability of failure of **MinCut** to output the mincut in each execution is at most $1 - \frac{2}{n(n-1)}$, by Lemma 15.3.5. Now, **MinCutRep** fails, only if all the $n(n-1)$ executions of **MinCut** fail. But these executions are independent, as such, the probability to this happen is at most

$$\left(1 - \frac{2}{n(n-1)}\right)^{n(n-1)} \leq \exp\left(-\frac{2}{n(n-1)} \cdot n(n-1)\right) = \exp(-2) < 0.14,$$

since $1 - x \leq e^{-x}$ for $0 \leq x \leq 1$. ■

Theorem 15.3.10. *One can compute the minimum cut in* $O(n^4)$ *time with constant probability to get a correct result. In* $O\left(n^4 \log n\right)$ *time the minimum cut is returned with high probability.*

15.3.2. An alternative implementation using MST

The algorithm. The above algorithm can be restated as follows. Randomly assign weights to the edges of G (say, by picking numbers in $[0, 1]$). Next, compute the MST T of the graph according to these weights. Remove the heaviest edge in the MST. The resulting partition of T into two trees, corresponds to a cut in the original graph. Return this cut as a candidate to be the minimum cut.

The analysis. To see that this algorithm is equivalent to **MinCut** (Figure 15.5), observe that the contraction algorithm simulates Kruskal's MST algorithm when run on randomly weighted edges. First, imagine implementing **MinCut** so that it keeps parallel edges. Then, the edges connecting two vertices that are not contracted are exactly the edges between the two connected components. Picking a random edge to contract, is equivalent to picking the edge with the minimum random weight. Thus, the MST algorithm here just simulates **MinCut** (or vice versa).

A small optimization. It is possible to compute the heaviest edge in the MST, and the partition it induces in (deterministic) linear time – it is a nice example of the search and prune technique.

Exercise 15.3.11. Given a graph G with weights on the edges, show how to compute the maximum weight edge in the MST of G in $O(n + m)$ time, where n are m are the number of vertices and edges of G, respectively.

Thus, this yields a $O(n + m)$ implementation of **MinCut**. We get the following result.

Lemma 15.3.12. **MinCut** *can implemented to run in* $O(n + m)$ *time, and it outputs the mincut with probability* $\geq \frac{2}{n(n-1)}$.

```
Contract ( G, t )
begin
   while |(G)| > t do
        Pick a random edge e in G.
        G ← G/e
   return G
end
```

```
FastCut(G = (V, E))
   G – multi-graph
begin
   n ← |V(G)|
   if n ≤ 6 then
        Compute (via brute force) minimum cut
        of G and return cut.
   t ← ⌈1 + n/√2⌉
   H_1 ← Contract(G, t)
   H_2 ← Contract(G, t)
   /* Contract is randomized!!! */
   X_1 ← FastCut(H_1),
   X_2 ← FastCut(H_2)
   return minimum cut out of X_1 and X_2.
end
```

Figure 15.6: **Contract**(G,t) shrinks G till it has only t vertices. **FastCut** computes the minimum cut using **Contract**.

15.4. A faster algorithm

The algorithm presented in the previous section is extremely simple. Which raises the question of whether we can get a faster algorithm[②]?

So, why **MinCutRep** needs so many executions? Well, the probability of success in the first ν iterations is

$$\begin{aligned}\mathbb{P}[\mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{\nu-1}] &\geq \prod_{i=0}^{\nu-1}\left(1-\frac{2}{n-i}\right) = \prod_{i=0}^{\nu-1}\frac{n-i-2}{n-i} \\ &= \frac{n-2}{n} * \frac{n-3}{n-1} * \frac{n-4}{n-2}\ldots = \frac{(n-\nu)(n-\nu-1)}{n\cdot(n-1)}. \end{aligned} \tag{15.2}$$

Namely, this probability deteriorates very quickly toward the end of the execution, when the graph becomes small enough. (To see this, observe that for $\nu = n/2$, the probability of success is roughly $1/4$, but for $\nu = n-\sqrt{n}$ the probability of success is roughly $1/n$.)

So, the key observation is that as the graph get smaller the probability to make a bad choice increases. So, instead of doing the amplification from the outside of the algorithm, we will run the new algorithm more times when the graph is smaller. Namely, we put the amplification directly into the algorithm.

The basic new operation we use is **Contract**, depicted in Figure 15.6, which also depict the new algorithm **FastCut**.

Lemma 15.4.1. *The running time of* **FastCut**(G) *is* $O\left(n^2 \log n\right)$, *where* $n = |V(G)|$.

Proof: Well, we perform two calls to **Contract**(G,t) which takes $O(n^2)$ time. And then we perform two recursive calls on the resulting graphs. We have

$$T(n) = O(n^2) + 2T(n/\sqrt{2}).$$

The solution to this recurrence is $O\left(n^2 \log n\right)$ as one can easily (and should) verify. ■

[②]This would require a more involved algorithm, that is life.

Exercise 15.4.2. Show that one can modify **FastCut** so that it uses only $O(n^2)$ space.

Lemma 15.4.3. *The probability that* **Contract**$(\mathsf{G}, n/\sqrt{2})$ *had not contracted the minimum cut is at least* $1/2$.

Namely, the probability that the minimum cut in the contracted graph is still a minimum cut in the original graph is at least $1/2$.

Proof: Just plug in $\nu = n - t = n - \left\lceil 1 + n/\sqrt{2} \right\rceil$ into Eq. (15.2). We have

$$\mathbb{P}\left[\mathcal{E}_0 \cap \ldots \cap \mathcal{E}_{n-t}\right] \geq \frac{t(t-1)}{n \cdot (n-1)} = \frac{\left\lceil 1 + n/\sqrt{2} \right\rceil \left(\left\lceil 1 + n/\sqrt{2} \right\rceil - 1\right)}{n(n-1)} \geq \frac{1}{2}.$$ ■

The following lemma bounds the probability of success.

Lemma 15.4.4. **FastCut** *finds the minimum cut with probability larger than* $\Omega(1/\log n)$.

Proof: Let T_h be the recursion tree of the algorithm of depth $h = \Theta(\log n)$. Color an edge of recursion tree by black if the contraction succeeded. Clearly, the algorithm succeeds if there is a path from the root to a leaf that is all black. This is exactly the settings of Lemma 15.1.1, and we conclude that the probability of success is at least $1/(h+1) = \Theta(1/\log n)$, as desired. ■

Exercise 15.4.5. Prove, that running **FastCut** repeatedly $c \cdot \log^2 n$ times, guarantee that the algorithm outputs the minimum cut with probability $\geq 1 - 1/n^2$, say, for c a constant large enough.

Theorem 15.4.6. *One can compute the minimum cut in a graph* G *with* n *vertices in* $O(n^2 \log^3 n)$ *time. The algorithm succeeds with probability* $\geq 1 - 1/n^2$.

Proof: We do amplification on **FastCut** by running it $O(\log^2 n)$ times. The running time bound follows from Lemma 15.4.1. The bound on the probability follows from Lemma 15.4.4, and using the amplification analysis as done in Lemma 15.3.9 for **MinCutRep**. ■

15.5. Bibliographical Notes

The **MinCut** algorithm was developed by David Karger during his PhD thesis in Stanford. The fast algorithm is a joint work with Clifford Stein. The basic algorithm of the mincut is described in [MR95, pages 7–9], the faster algorithm is described in [MR95, pages 289–295].

Galton-Watson process. The idea of using coloring of the edges of a tree to analyze **FastCut** might be new (i.e., Section 15.1.2).

References

[Gre69] W. Greg. *Why are Women Redundant?* Trübner, 1869.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

[Ste12] E. Steinlight. Why novels are redundant: sensation fiction and the overpopulation of literature. *ELH*, 79(2): 501–535, 2012.

[WG75] H. W. Watson and F. Galton. On the probability of the extinction of families. *J. Anthrop. Inst. Great Britain*, 4: 138–144, 1875.

Chapter 16

Discrepancy and Derandomization

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "Shortly after the celebration of the four thousandth anniversary of the opening of space, Angary J. Gustible discovered Gustible's planet. The discovery turned out to be a tragic mistake.
> Gustible's planet was inhabited by highly intelligent life forms. They had moderate telepathic powers. They immediately mind-read Angary J. Gustible's entire mind and life history, and embarrassed him very deeply by making up an opera concerning his recent divorce."
>
> Gustible's Planet, Cordwainer Smith

16.1. Discrepancy

Consider a set system $(\mathsf{X}, \mathcal{R})$, where $n = |\mathsf{X}|$, and $\mathcal{R} \subseteq 2^{\mathsf{X}}$. A natural task is to partition X into two sets S, T, such that for any range $\mathbf{r} \in \mathcal{R}$, we have that $\chi(\mathbf{r}) = \big||S \cap \mathbf{r}| - |T \cap \mathbf{r}|\big|$ is minimized. In a perfect partition, we would have that $\chi(\mathbf{r}) = 0$ – the two sets S, T partition every range perfectly in half. A natural way to do so, is to consider this as a coloring problem – an element of X is colored by $+1$ if it is in S, and -1 if it is in T.

Definition 16.1.1. Consider a set system $\mathsf{S} = (\mathsf{X}, \mathcal{R})$, and let $\chi : \mathsf{X} \to \{-1, +1\}$ be a function (i.e., a coloring). The ***discrepancy*** of $\mathbf{r} \in \mathcal{R}$ is $\chi(\mathbf{r}) = |\sum_{x \in \mathbf{r}} \chi(x)|$. The ***discrepancy of*** χ is the maximum discrepancy over all the ranges – that is

$$\operatorname{disc}(\chi) = \max_{\mathbf{r} \in \mathcal{R}} \chi(\mathbf{r}).$$

The ***discrepancy*** of S is

$$\operatorname{disc}(\mathsf{S}) = \min_{\chi : \mathsf{X} \to \{-1,+1\}} \operatorname{disc}(\chi).$$

Bounding the discrepancy of a set system is quite important, as it provides a way to shrink the size of the set system, while introducing small error. Computing the discrepancy of a set system is generally quite challenging. A rather decent bound follows by using random coloring.

Definition 16.1.2. For a vector $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{R}^n$, $\|\mathbf{v}\|_\infty = \max_i |v_i|$.

For technical reasons, it is easy to think about the set system as an incidence matrix.

Definition 16.1.3. For a $m \times n$ a binary matrix M (i.e., each entry is either 0 or 1), consider a vector $\mathbf{b} \in \{-1, +1\}^n$. The ***discrepancy*** of $\mathbf{b}$ is $\|\mathsf{M}\mathbf{b}\|_\infty$.

Theorem 16.1.4. *Let* M *be an* $n \times n$ *binary matrix (i.e., each entry is either* 0 *or* 1*), then there always exists a vector* $\mathbf{b} \in \{-1, +1\}^n$*, such that* $\|\mathsf{M}\mathbf{b}\|_\infty \leq 4\sqrt{n \log n}$*. Specifically, a random coloring provides such a coloring with high probability.*

Proof: Let $v = (v_1, \ldots, v_n)$ be a row of M. Chose a random $\mathbf{b} = (b_1, \ldots, b_n) \in \{-1, +1\}^n$. Let $i_1, \ldots, i_\tau$ be the indices such that $v_{i_j} = 1$, and let

$$Y = \langle v, \mathbf{b} \rangle = \sum_{i=1}^{n} v_i b_i = \sum_{j=1}^{\tau} v_{i_j} b_{i_j} = \sum_{j=1}^{\tau} b_{i_j}.$$

As such Y is the sum of m independent random variables that accept values in $\{-1, +1\}$. Clearly,

$$\mathbb{E}[Y] = \mathbb{E}[\langle v, \mathbf{b} \rangle] = \mathbb{E}\Big[\sum_i v_i b_i\Big] = \sum_i \mathbb{E}[v_i b_i] = \sum_i v_i \, \mathbb{E}[b_i] = 0.$$

By Chernoff inequality and the symmetry of Y, we have that, for $\Delta = 4\sqrt{n \ln m}$, it holds

$$\mathbb{P}[|Y| \geq \Delta] = 2\,\mathbb{P}[\langle v, \mathbf{b} \rangle \geq \Delta] = 2\,\mathbb{P}\Big[\sum_{j=1}^{\tau} b_{i_j} \geq \Delta\Big] \leq 2\exp\left(-\frac{\Delta^2}{2\tau}\right) = 2\exp\left(-8\frac{n \ln m}{\tau}\right) \leq \frac{2}{m^8},$$

since $\tau \leq n$. In words, the probability that any entry in $\mathsf{M}b$ exceeds (in absolute values) $4\sqrt{n \ln}$, is smaller than $2/m^7$. Thus, with probability at least $1 - 2/m^7$, all the entries of $\mathsf{M}b$ have absolute value smaller than $4\sqrt{n \ln m}$.

In particular, there exists a vector $\mathbf{b} \in \{-1, +1\}^n$ such that $\| \mathsf{M}\mathbf{b} \|_\infty \leq 4\sqrt{n \ln m}$. ■

We might spend more time on discrepancy later on – it is a fascinating topic, well worth its own course.

16.2. The Method of Conditional Probabilities

In previous lectures, we encountered the following problem.

Problem 16.2.1 (Set Balancing/Discrepancy). Given a binary matrix M of size $n \times n$, find a vector $\mathbf{v} \in \{-1, +1\}^n$, such that $\|\mathsf{M}\mathbf{v}\|_\infty$ is minimized.

Using random assignment and the Chernoff inequality, we showed that there exists $\mathbf{v}$, such that $\|\mathsf{M}\mathbf{v}\|_\infty \leq 4\sqrt{n \ln n}$. Can we derandomize this algorithm? Namely, can we come up with an efficient *deterministic* algorithm that has low discrepancy?

To derandomize our algorithm, construct a computation tree of depth n, where in the ith level we expose the ith coordinate of $\mathbf{v}$. This tree T has depth n. The root represents all possible random choices, while a node at depth i, represents all computations when the first i bits are fixed. For a node $v \in T$, let $P(v)$ be the probability that a random computation starting from v succeeds – here randomly assigning the remaining bits can be interpreted as a random walk down the tree to a leaf.

Formally, the algorithm is ***successful*** if ends up with a vector $\mathbf{v}$, such that $\|\mathsf{M}\mathbf{v}\|_\infty \leq 4\sqrt{n \ln n}$.

Let v_l and v_r be the two children of v. Clearly, $P(v) = (P(v_l) + P(v_r))/2$. In particular, $\max(P(v_l), P(v_r)) \geq P(v)$. Thus, if we could compute $P(\cdot)$ quickly (and deterministically), then we could derandomize the algorithm.

Let C_m^+ be the bad event that $\mathbf{r}_m \cdot \mathbf{v} > 4\sqrt{n \log n}$, where $\mathbf{r}_m$ is the mth row of M. Similarly, C_m^- is the bad event that $\mathbf{r}_m \cdot \mathbf{v} < -4\sqrt{n \log n}$, and let $C_m = C_m^+ \cup C_m^-$. Consider the probability, $\mathbb{P}\Big[C_m^+ \;\Big|\; \mathbf{v}_1, \ldots, \mathbf{v}_k\Big]$ (namely, the first k coordinates of $\mathbf{v}$ are specified). Let $\mathbf{r}_m = (r_1, \ldots, r_n)$. We have that

$$\mathbb{P}[C_m^+ \;|\; \mathbf{v}_1, \ldots, \mathbf{v}_k] = \mathbb{P}\Big[\sum_{i=k+1}^{n} \mathbf{v}_i r_i > 4\sqrt{n \log n} - \sum_{i=1}^{k} \mathbf{v}_i r_i\Big] = \mathbb{P}\Big[\sum_{i \geq k+1, r_i \neq 0} \mathbf{v}_i r_i > L\Big] = \mathbb{P}\Big[\sum_{i \geq k+1, r_i = 1} \mathbf{v}_i > L\Big],$$

where $L = 4\sqrt{n \log n} - \sum_{i=1}^{k} \mathbf{v}_i r_i$ is a known quantity (since $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are known). Let $V = \sum_{i \geq k+1, r_i = 1} 1$. We have,

$$\mathbb{P}\Big[C_m^+ \,\Big|\, \mathbf{v}_1, \ldots, \mathbf{v}_k\Big] = \mathbb{P}\Big[\sum_{\substack{i \geq k+1 \\ \alpha_i = 1}} (\mathbf{v}_i + 1) > L + V\Big] = \mathbb{P}\Big[\sum_{\substack{i \geq k+1 \\ \alpha_i = 1}} \frac{\mathbf{v}_i + 1}{2} > \frac{L+V}{2}\Big],$$

The last quantity is the probability that in V flips of a fair $0/1$ coin one gets more than $(L+V)/2$ heads. Thus,

$$P_m^+ = \mathbb{P}\Big[C_m^+ \,\Big|\, \mathbf{v}_1, \ldots, \mathbf{v}_k\Big] = \sum_{i=\lceil (L+V)/2 \rceil}^{\mathsf{V}} \binom{\mathsf{V}}{i} \frac{1}{2^n} = \frac{1}{2^n} \sum_{i=\lceil (L+V)/2 \rceil}^{\mathsf{V}} \binom{\mathsf{V}}{i}.$$

This implies, that we can compute P_m^+ in polynomial time! Indeed, we are adding $V \leq n$ numbers, each one of them is a binomial coefficient that has polynomial size representation in n, and can be computed in polynomial time (why?). One can define in similar fashion P_m^-, and let $P_m = P_m^+ + P_m^-$. Clearly, P_m can be computed in polynomial time, by applying a similar argument to the computation of $P_m^- = \mathbb{P}\Big[C_m^- \,\Big|\, \mathbf{v}_1, \ldots, \mathbf{v}_k\Big]$.

For a node $v \in T$, let $\mathbf{v}_v$ denote the portion of $\mathbf{v}$ that was fixed when traversing from the root of T to v. Let $P(v) = \sum_{m=1}^{n} \mathbb{P}\Big[C_m \,\Big|\, \mathbf{v}_v\Big]$. By the above discussion $P(v)$ can be computed in polynomial time. Furthermore, we know, by the previous result on discrepancy that $P(r) < 1$ (that was the bound used to show that there exist a good assignment).

As before, for any $v \in T$, we have $P(v) \geq \min(P(v_l), P(v_r))$. Thus, we have a polynomial *deterministic* algorithm for computing a set balancing with discrepancy smaller than $4\sqrt{n \log n}$. Indeed, set $v = root(T)$. And start traversing down the tree. At each stage, compute $P(v_l)$ and $P(v_r)$ (in polynomial time), and set v to the child with lower value of $P(\cdot)$. Clearly, after n steps, we reach a leaf, that corresponds to a vector $\mathbf{v}'$ such that $\|A\mathbf{v}'\|_\infty \leq 4\sqrt{n \log n}$.

Theorem 16.2.2. *Using the method of conditional probabilities, one can compute in polynomial time in n, a vector* $\mathbf{v} \in \{-1, 1\}^n$*, such that* $\|A\mathbf{v}\|_\infty \leq 4\sqrt{n \log n}$.

Note, that this method might fail to find the best assignment.

16.3. Bibliographical Notes

There is a lot of nice work on discrepancy in geometric settings. See the books [**c-dmr-01**, Mat99].

References

[Mat99] J. Matoušek. *Geometric Discrepancy*. Vol. 18. Algorithms and Combinatorics. Springer, 1999.

Chapter 17

Independent set – Turán's theorem

I don't know why it should be, I am sure; but the sight of another man asleep in bed when I am up, maddens me. It seems to me so shocking to see the precious hours of a man's life - the priceless moments that will never come back to him again - being wasted in mere brutish sleep.

Jerome K. Jerome, Three men in a boat

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

17.1. Turán's theorem

17.1.1. Some silly helper lemmas

We will need the following well-known inequality.

Lemma 17.1.1 (AM-GM inequality: Arithmetic and geometric means inequality.). *For any $x_1, \ldots, x_n \geq 0$ we have $\frac{x_1 + x_2 + \cdots + x_n}{n} \geq \sqrt[n]{x_1 x_2 \cdots x_n}$.*

This inequality readily implies the "inverse" inequality: $\frac{1}{\sqrt[n]{x_1 x_2 \cdots x_n}} \geq \frac{n}{x_1 + x_2 + \cdots + x_n}$

Lemma 17.1.2. *Let $x_1, \ldots, x_n \geq 0$ be n numbers. We have that $\sum_{i=1}^n \frac{1}{x_i} \geq \frac{n}{(\sum_i x_i)/n}$.*

Proof: By the SM-GM inequality and then its "inverse" form, we have

$$\frac{\sum_{i=1}^n \frac{1}{x_i}}{n} = \frac{1/x_1 + 1/x_2 + \cdots + 1/x_n}{n} \geq \sqrt[n]{(1/x_1)(1/x_2)\cdots(1/x_n)} = \frac{1}{\sqrt[n]{x_1 x_2 \cdots x_n}} \geq \frac{n}{x_1 + x_2 + \cdots + x_n}. \quad \blacksquare$$

Lemma 17.1.3. *Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be a graph with n vertices, and let d_{G} be the average degree in the graph. We have that $\sum_{v \in \mathsf{V}} \frac{1}{1 + d(v)} \geq \frac{n}{1 + d_{\mathsf{G}}}$.*

Proof: Let the ith vertex in G be v_i. Set $x_i = 1 + d(v_i)$, for all i. By Lemma 17.1.2, we have

$$\sum_{i=1}^n \frac{1}{1 + d(v_i)} = \sum_{i=1}^n \frac{1}{x_i} \geq \frac{n}{(\sum_i x_i)/n} = \frac{n}{[\sum_i (1 + d(v_i))]/n} = \frac{n}{1 + d_{\mathsf{G}}}. \quad \blacksquare$$

17.1.2. Statement and proof

Theorem 17.1.4 (Turán's theorem). *Let* $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ *be a graph with* n *vertices. The graph* G *has an independent set of size at least* $\frac{n}{1+d_G}$*, where* d_G *is the average vertex degree in* G*.*

Proof: Let $\pi = (\pi_1, \ldots, \pi_n)$ be a random permutation of the vertices of G. Pick the vertex π_i into the independent set if none of its neighbors appear before it in π. Clearly, v appears in the independent set if and only if it appears in the permutation before all its $d(v)$ neighbors. The probability for this is $1/(1 + d(v))$. Thus, the expected size of the independent set is (exactly)

$$\tau = \sum_{v \in V} \frac{1}{1 + d(v)}, \tag{17.1}$$

by linearity of expectations. Thus, by the probabilistic method, there exists an independent set in G of size at least τ. The claim now readily follows from Lemma 17.1.3. ■

17.1.3. An alternative proof of Turán's theorem

Following a post of this write-up on my blog, readers suggested two modifications. We present an alternative proof incorporating both suggestions.

Alternative proof of Theorem 18.1.3: We associate a charge of size $1/(d(v) + 1)$ with each vertex of G. Let $\gamma(G)$ denote the total charge of the vertices of G. We prove, using induction, that there is always an independent set in G of size at least $\gamma(G)$. If G is the empty graph, then the claim trivially holds. Otherwise, assume that it holds if the graph has at most $n - 1$ vertices, and consider the vertex v of lowest degree in G. The total charge of v and its neighbors is

$$\frac{1}{d(v)+1} + \sum_{uv \in E} \frac{1}{d(u)+1} \leq \frac{1}{d(v)+1} + \sum_{uv \in E} \frac{1}{d(v)+1} = \frac{d(v)+1}{d(v)+1} = 1,$$

since $d(u) \geq d(v)$, for all $uv \in E$. Now, consider the graph H resulting from removing v and its neighbors from G. Clearly, $\gamma(H)$ is larger (or equal) to the total charge of the vertices of $V(H)$ in G, as their degree had either decreased (or remained the same). As such, by induction, we have an independent set in H of size at least $\gamma(H)$. Together with v this forms an independent set in G of size at least $\gamma(H) + 1 \geq \gamma(G)$. Implying that there exists an independent set in G of size

$$\tau = \sum_{v \in V} \frac{1}{1 + d(v)}, \tag{17.2}$$

Now, set $x_v = 1 + d(v)$, and observe that

$$(n + 2|E|)\tau = \left(\sum_{v \in V} x_v\right)\left(\sum_{v \in V} \frac{1}{x_v}\right) \geq \left(\sum_{v \in V} \sqrt{x_v}\frac{1}{\sqrt{x_v}}\right)^2 = n^2,$$

using Cauchy-Schwartz inequality. Namely, $\tau \geq \dfrac{n^2}{n + 2|E|} = \dfrac{n}{1 + 2|E|/n} = \dfrac{n}{1 + d_G}$. ■

Lemma 17.1.5 (Cauchy-Schwartz inequality). *For positive numbers* $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$*, we have*

$$\sum_i \alpha_i \beta_i \leq \sqrt{\sum_i \alpha_i^2} \sqrt{\sum_i \beta_i^2}.$$

17.1.4. An algorithm for the weighted case

In the weighted case, we associate weight $w(v)$ with each vertex of G, and we are interested in the maximum weight independent set in G. Deploying the algorithm described in the first proof of Theorem 18.1.3, implies the following.

Lemma 17.1.6. *The graph $G = (V, E)$ has an independent set of size $\geq \sum_{v \in V} \frac{w(v)}{1 + d(v)}$.*

Proof: By linearity of expectations, we have that the expected weight of the independent set computed is equal to

$$\sum_{v \in V} w(v) \cdot \mathbb{P}[v \text{ in the independent set}] = \sum_{v \in V} \frac{w(v)}{1 + d(v)},$$

■

References

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

Chapter 18

Derandomization using Conditional Expectations

Yes, my guard stood hard when abstract threats
Too noble to neglect
Deceived me into thinking
I had something to protect
Good and bad, I define these terms
Quite clear, no doubt, somehow
Ah, but I was so much older then
Im younger than that now

My Back Pages, Bob Dylan

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

18.1. Method of conditional expectations

Imagine that we have a randomized algorithm that uses as randomized input n bits $X_1, \ldots, X_n$, and outputs a solution of quality $f(X_1, \ldots, X_n)$. Assume that given values $v_1, \ldots, v_k \in \{0, 1\}$, one can compute, efficiently and deterministicly, the quantity

$$\mathbb{E}f(v_1, \ldots, v_k) = \mathbb{E}[f(v_1, \ldots, v_k, X_{k+1}, \ldots, X_n)] = \mathbb{E}[f(X_1, \ldots, X_n) \mid X_1 = v_1, \ldots, X_k = v_k]$$

by a given procedure $\mathbf{eval}_{\mathbb{E}f}$. In such settings, one can compute efficiently and deterministicly an assignment $v_1, \ldots, v_n$, such that

$$f(v_1, \ldots, v_n) \geq \mathbb{E}f, \qquad \text{where} \qquad \mathbb{E}f = \mathbb{E}[f(X_1, \ldots, X_n)].$$

Or alternatively, one can find an assignment $u_1, \ldots, u_n$ such that $f(u_1, \ldots, u_n) \leq \mathbb{E}[f(X_1, \ldots, X_n)]$.

The algorithm. Assume the algorithm had computed a partial assignment for $v_1, \ldots, v_k$, such that $\alpha_k = \mathbb{E}f(v_1, \ldots, v_k) \geq \mathbb{E}f$. The algorithm then would compute the two values

$$\alpha_{k,0} = \mathbb{E}f(v_1, \ldots, v_k, 0) \qquad \text{and} \qquad \alpha_{k,1} = \mathbb{E}f(v_1, \ldots, v_k, 1).$$

Observe that

$$\alpha_k = \mathbb{E}f(v_1, \ldots, v_k) = \mathbb{P}[X_{k+1} = 0]\mathbb{E}f(v_1, \ldots, v_k, 0) + \mathbb{P}[X_{k+1} = 1]\mathbb{E}f(v_1, \ldots, v_k, 1) = \frac{\alpha_{k,0} + \alpha_{k,1}}{2}.$$

As such, there is an i, such that $\alpha_{k,i} \geq \alpha_k$. The algorithm sets $v_{k+1} = i$, and continues to the next iteration.

Correctness. This is hopefully clear. Initially, $\alpha_0 = \mathbb{E}f$. In each iteration, the algorithm makes a choice, such that $\alpha_k \geq \alpha_{k-1}$. Thus,

$$\alpha_n = \mathbb{E}f(v_1, \ldots, v_n) = f(v_1, \ldots, v_n) \geq \alpha_{n-1} \geq \cdots \geq \alpha_0 = \mathbb{E}f.$$

Running time. The algorithm performs $2n$ invocations of $\mathbf{eval}_{\mathbb{E}f}$.

Result.

Theorem 18.1.1. *Given a function $f(X_1, \ldots, X_n)$ over n random binary variables, such that one can compute determinedly $\mathbb{E}f(v_1, \ldots v_k) = \mathbb{E}[f(X_1, \ldots, X_n) \mid X_1 = v_1, \ldots, X_k = v_k]$ in $T(n)$ time. Then, one can compute an assignment $v_1, \ldots, v_n$, such that $f(v_1, \ldots, v_n) \geq \mathbb{E}f = \mathbb{E}[f(X_1, \ldots, X_n)]$. The running time of the algorithm is $O(n + nT(n))$.*

18.1.1. Applications

18.1.1.1. Max kSAT

Given a boolean formula F with n variables and m clauses, where each clause has exactly k literals, let $f(X_1, \ldots, X_n)$ be the number of clauses the assignment $X_1, \ldots, X_n$ satisfies. Clearly, one can compute f in $O(mk)$ time. More generally, given a partial assignment $v_1, \ldots, v_k$, one can compute $\alpha_k = \mathbb{E}f(v_1, \ldots, v_k)$. Indeed, scan F and assign all the literals that depends on the variables $X_1, \ldots, X_k$ their values. A literal evaluating to one satisfies its clause, and we count it as such. What remains are clauses with at most k literals. A literal with i literals, have probability *exactly* $1 - 1/2^i$ to be satisfied. Thus, summing these probabilities on these leftover clauses given use the desired value. This takes $O(mk)$ time. Using Theorem 18.1.1 we get the following.

Lemma 18.1.2. *Let F be a kSAT formula with n variables and m clauses. One can compute deterministicly an assignment that satisfies at least $(1 - 1/2^k)m$ clauses of F. This takes $O(mnk)$ time.*

18.1.1.2. Max cut

18.1.1.3. Turán theorem

Lemma 18.1.3 (Turán's theorem). *Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be a graph with n vertices and m edges. One can compute determinedly, in $O(nm)$ time, an independent set of size at least $\dfrac{n}{1 + 2m/n}$.*

Proof: Exercise. ■

References

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

Chapter 19

Martingales

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> 'After that he always chose out a "dog command" and sent them ahead. It had the task of informing the inhabitants in the village where we were going to stay overnight that no dog must be allowed to bark in the night otherwise it would be liquidated. I was also on one of those commands and when we came to a village in the region of Milevsko I got mixed up and told the mayor that every dog-owner whose dog barked in the night would be liquidated for strategic reasons. The mayor got frightened, immediately harnessed his horses and rode to headquarters to beg mercy for the whole village. They didn't let him in, the sentries nearly shot him and so he returned home, but before we got to the village everybody on his advice had tied rags round the dogs muzzles with the result that three of them went mad.'
>
> The good soldier Svejk, Jaroslav Hasek

19.1. Martingales

19.1.1. Preliminaries

Let X and Y be two random variables. Let $\rho(x,y) = \mathbb{P}[(X = x) \cap (Y = y)]$. Observe that

$$\mathbb{P}[X = x \mid Y = y] = \frac{\rho(x,y)}{\mathbb{P}[Y = y]} = \frac{\rho(x,y)}{\sum_z \rho(z,y)}$$

and $\mathbb{E}\Big[X \;\Big|\; Y = y\Big] = \sum_x x\, \mathbb{P}\Big[X = x \;\Big|\; Y = y\Big] = \dfrac{\sum_x x\rho(x,y)}{\sum_z \rho(z,y)} = \dfrac{\sum_x x\rho(x,y)}{\mathbb{P}[Y = y]}$.

The *conditional expectation* of X given Y, is the random variable $\mathbb{E}\Big[X \;\Big|\; Y\Big]$ is the random variable $f(y) = \mathbb{E}\Big[X \;\Big|\; Y = y\Big]$.

As a reminder, for any two random variables X and Y, we have

(I) Lemma 11.1.2: $\mathbb{E}[\mathbb{E}[X \mid Y]] = \mathbb{E}[X]$.

(II) Lemma 11.1.3: $\mathbb{E}[Y \cdot \mathbb{E}[X \mid Y]] = \mathbb{E}[XY]$.

19.1.2. Martingales

Intuitively, martingales are a sequence of random variables describing a process, where the only thing that matters at the beginning of the ith step is where the process was in the end of the $(i-1)$th step. That is, it does not matter how the process arrived to a certain state, only that it is currently at this state.

Definition 19.1.1. A sequence of random variables $X_0, X_1, \ldots$, is said to be a ***martingale sequence*** if for all $i > 0$, we have $\mathbb{E}[X_i \mid X_0, \ldots, X_{i-1}] = X_{i-1}$.

In particular, note that for a martingale, we have $\mathbb{E}[X_i \mid X_0, \ldots, X_{i-1}] = \mathbb{E}[X_i \mid X_{i-1}] = X_{i-1}$.

Lemma 19.1.2. *Let $X_0, X_1, \ldots,$ be a martingale sequence. Then, for all $i \geq 0$, we have $\mathbb{E}[X_i] = \mathbb{E}[X_0]$.*

Proof: By (I), and the martingale property, we have

$$\mathbb{E}[X_i] = \mathbb{E}[\mathbb{E}[X_i \mid X_{i-1}]] = \mathbb{E}[X_{i-1}] = \mathbb{E}[X_{i-2}] = \cdots = \mathbb{E}[X_0]\,. \qquad \blacksquare$$

19.1.2.1. Examples of martingales

Example 19.1.3. Consider the sum of money after participating in a sequence of fair bets. That is, let X_i be the amount of money a gambler has after playing i rounds. In each round it either gains one dollar, or loses one dollar (with equal probability). Clearly, we have

$$\mathbb{E}[X_i \mid X_0, \ldots, X_{i-1}] = \mathbb{E}[X_i \mid X_{i-1}] = X_{i-1} + \frac{1}{2} \cdot (+1) + \frac{1}{2} \cdot (-1) = X_{i-1}.$$

Example 19.1.4. Let $Y_i = X_i^2 - i$, where X_i is as defined in the above example. We claim that $Y_0, Y_1, \ldots$ is a martingale. Let us verify that this is true. Given Y_{i-1}, we have $Y_{i-1} = X_{i-1}^2 - (i-1)$. We have that

$$\begin{aligned}\mathbb{E}\Big[Y_i \;\Big|\; Y_{i-1}\Big] &= \mathbb{E}\Big[X_i^2 - i \;\Big|\; X_{i-1}^2 - (i-1)\Big] = \frac{1}{2}\Big((X_{i-1}+1)^2 - i)\Big) + \frac{1}{2}\Big((X_{i-1}-1)^2 - i\Big) \\ &= X_{i-1}^2 + 1 - i = X_{i-1}^2 - (i-1) = Y_{i-1},\end{aligned}$$

which implies that indeed it is a martingale.

Example 19.1.5. Let U be a urn with b black balls, and w white balls. We repeatedly select a ball and replace it by c balls having the same color. Let X_i be the fraction of black balls after the first i trials. We claim that the sequence $X_0, X_1, \ldots$ is a martingale.

Indeed, let $n_i = b + w + i(c-1)$ be the number of balls in the urn after the ith trial. Clearly,

$$\begin{aligned}\mathbb{E}\Big[X_i \;\Big|\; X_{i-1}, \ldots, X_0\Big] &= X_{i-1} \cdot \frac{(c-1) + X_{i-1} n_{i-1}}{n_i} + (1 - X_{i-1}) \cdot \frac{X_{i-1} n_{i-1}}{n_i} \\ &= \frac{X_{i-1}(c-1) + X_{i-1} n_{i-1}}{n_i} = X_{i-1} \frac{c - 1 + n_{i-1}}{n_i} = X_{i-1} \frac{n_i}{n_i} = X_{i-1}.\end{aligned}$$

Example 19.1.6. Let G be a random graph on the vertex set $V = \{1, \ldots, n\}$ obtained by independently choosing to include each possible edge with probability p. The underlying probability space over ***random graphs*** is denoted by $\mathbf{G}_{n,p}$. Arbitrarily label the $m = n(n-1)/2$ possible edges with the sequence $1, \ldots, m$. For $1 \leq j \leq m$, define the indicator random variable I_j, which takes values 1 if the edge j is present in G, and has value 0 otherwise. These indicator variables are independent and each takes value 1 with probability p.

Consider any real valued function f defined over the space of all graphs, e.g., the clique number, which is defined as being the size of the largest complete subgraph. The ***edge exposure martingale*** is the sequence of random variables $X_0, \ldots, X_m$ such that

$$X_i = \mathbb{E}[f(\mathsf{G}) \mid I_1, \ldots, I_i]\,,$$

while $X_0 = \mathbb{E}[f(G)]$ and $X_m = f(G)$. This sequence of random variable begin a martingale follows immediately from a theorem that would be described in the next lecture.

One can define similarly a ***vertex exposure martingale***, where the graph G_i is the graph induced on the first i vertices of the random graph G.

Example 19.1.7 (The sheep of Mabinogion). The following is taken from medieval Welsh manuscript based on Celtic mythology:

> "And he came towards a valley, through which ran a river; and the borders of the valley were wooded, and on each side of the river were level meadows. And on one side of the river he saw a flock of white sheep, and on the other a flock of black sheep. And whenever one of the white sheep bleated, one of the black sheep would cross over and become white; and when one of the black sheep bleated, one of the white sheep would cross over and become black." – *Peredur the son of Evrawk*, from the *Mabinogion*.

More concretely, we start at time 0 with w_0 white sheep, and b_0 black sheep. At every iteration, a random sheep is picked, it bleats, and a sheep of the other color turns to this color. the game stops as soon as all the sheep have the same color. No sheep dies or get born during the game. Let X_i be the expected number of black sheep in the end of the game, after the ith iteration. For reasons that we would see later on, this sequence is a martingale.

The original question is somewhat more interesting – if we are allowed to take a way sheep in the end of each iteration, what is the optimal strategy to maximize X_i?

19.1.2.2. Azuma's inequality

A sequence of random variables $X_0, X_1, \ldots$ has ***bounded differences*** if $|X_i - X_{i-1}| \le \Delta$, for some Δ.

Theorem 19.1.8 (Azuma's Inequality.). *Let $X_0, \ldots, X_m$ be a martingale with $X_0 = 0$, and*

$$|X_{i+1} - X_i| \le 1, \qquad \text{for} \quad i = 0, \ldots, m-1.$$

For any $\lambda > 0$, we have $\mathbb{P}\left[X_m > \lambda\sqrt{m}\right] < \exp\left(-\lambda^2/2\right)$.

Proof: Let $\alpha = \lambda/\sqrt{m}$. Let $Y_i = X_i - X_{i-1}$, so that $|Y_i| \le 1$ and $\mathbb{E}[Y_i \mid X_0, \ldots, X_{i-1}] = 0$.

We are interested in bounding $\mathbb{E}\left[e^{\alpha Y_i} \,\middle|\, X_0, \ldots, X_{i-1}\right]$. Note that, for $-1 \le x \le 1$, we have

$$f(x) = e^{\alpha x} \le h(x) = \frac{e^{\alpha} + e^{-\alpha}}{2} + \frac{e^{\alpha} - e^{-\alpha}}{2} x,$$

as $f(x) = e^{\alpha x}$ is a convex function, $h(-1) = e^{-\alpha} = f(-1)$, $h(1) = e^{\alpha} = f(+1)$, and $h(x)$ is a linear function. Thus,

$$\begin{aligned}
\mathbb{E}\left[e^{\alpha Y_i} \,\middle|\, X_0, \ldots, X_{i-1}\right] &\le \mathbb{E}\left[h(Y_i) \,\middle|\, X_0, \ldots, X_{i-1}\right] = h\left(\mathbb{E}\left[Y_i \,\middle|\, X_0, \ldots, X_{i-1}\right]\right) \\
&= h(0) = \frac{e^{\alpha} + e^{-\alpha}}{2} \\
&= \frac{(1 + \alpha + \frac{\alpha^2}{2!} + \frac{\alpha^3}{3!} + \cdots) + (1 - \alpha + \frac{\alpha^2}{2!} - \frac{\alpha^3}{3!} + \cdots)}{2} \\
&= 1 + \frac{\alpha^2}{2} + \frac{\alpha^4}{4!} + \frac{\alpha^6}{6!} + \cdots \\
&\le 1 + \frac{1}{1!}\left(\frac{\alpha^2}{2}\right) + \frac{1}{2!}\left(\frac{\alpha^2}{2}\right)^2 + \frac{1}{3!}\left(\frac{\alpha^2}{2}\right)^3 + \cdots = \exp\left(\alpha^2/2\right),
\end{aligned}$$

as $(2i)! \ge 2^i i!$.

We have that

$$\tau = \mathbb{E}\Big[e^{\alpha X_m}\Big] = \mathbb{E}\Big[\prod_{i=1}^{m} e^{\alpha Y_i}\Big] = \mathbb{E}\Big[g(X_0,\ldots,X_{m-1})e^{\alpha Y_m}\Big], \qquad \text{where} \qquad g(X_0,\ldots,X_{m-1}) = \prod_{i=1}^{m-1} e^{\alpha Y_i}.$$

By the martingale property, we have that

$$\mathbb{E}[Y_i \mid X_0,\ldots,X_{m-1}] = \mathbb{E}[Y_i \mid g(X_0,\ldots,X_{m-1})] = 0.$$

By the above, this implies that $\mathbb{E}[e^{\alpha Y_i} \mid g(X_0,\ldots,X_{i-1})] \le \exp\big(\alpha^2/2\big)$. Hence, by Lemma 11.1.3, we have that

$$\begin{aligned}
\tau = \mathbb{E}\Big[e^{\alpha X_m}\Big] &= \mathbb{E}\Big[\prod_{i=1}^{m} e^{\alpha Y_i}\Big] = \mathbb{E}\Big[g(X_0,\ldots,X_{m-1})e^{\alpha Y_m}\Big] \\
&= \mathbb{E}\Big[g(X_0,\ldots,X_{m-1})\,\mathbb{E}\Big[e^{\alpha Y_m} \;\Big|\; g(X_0,\ldots,X_{m-1})\Big]\Big] \le e^{\alpha^2/2}\,\mathbb{E}[g(X_0,\ldots,X_{m-1})] \\
&\le \exp\big(m\alpha^2/2\big).
\end{aligned}$$

Therefore, by Markov's inequality, we have

$$\begin{aligned}
\mathbb{P}\Big[X_m > \lambda\sqrt{m}\Big] = \mathbb{P}\Big[e^{\alpha X_m} > e^{\alpha\lambda\sqrt{m}}\Big] &= \frac{\mathbb{E}\Big[e^{\alpha X_m}\Big]}{e^{\alpha\lambda\sqrt{m}}} = e^{m\alpha^2/2-\alpha\lambda\sqrt{m}} \\
&= \exp\big(m(\lambda/\sqrt{m})^2/2 - (\lambda/\sqrt{m})\lambda\sqrt{m}\big) = e^{-\lambda^2/2},
\end{aligned}$$

implying the result. ■

Here is an alternative form.

Theorem 19.1.9 (Azuma's Inequality). *Let $X_0,\ldots,X_m$ be a martingale sequence such that and $|X_{i+1}-X_i| \le 1$ for all $0 \le i < m$. Let $\lambda > 0$ be arbitrary. Then $\mathbb{P}\Big[|X_m - X_0| > \lambda\sqrt{m}\Big] < 2\exp\big(-\lambda^2/2\big)$.*

Example 19.1.10. Let $\chi(H)$ be the chromatic number of a graph H. What is chromatic number of a random graph? How does this random variable behaves?

Consider the vertex exposure martingale, and let $X_i = \mathbb{E}\Big[\chi(G) \;\Big|\; G_i\Big]$. Again, without proving it, we claim that $X_0,\ldots,X_n = X$ is a martingale, and as such, we have: $\mathbb{P}\Big[|X_n - X_0| > \lambda\sqrt{n}\Big] \le e^{-\lambda^2/2}$. However, $X_0 = \mathbb{E}[\chi(G)]$, and $X_n = \mathbb{E}\Big[\chi(G) \;\Big|\; G_n\Big] = \chi(G)$. Thus,

$$\mathbb{P}\Big[\big|\chi(G) - \mathbb{E}[\chi(G)]\big| > \lambda\sqrt{n}\Big] \le e^{-\lambda^2/2}.$$

Namely, the chromatic number of a random graph is highly concentrated! And we do not even (need to) know what is the expectation of this variable!

19.2. Bibliographical notes

Our presentation follows [MR95].

References

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

Chapter 20

Martingales II

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "The Electric Monk was a labor-saving device, like a dishwasher or a video recorder. Dishwashers washed tedious dishes for you, thus saving you the bother of washing them yourself, video recorders watched tedious television for you, thus saving you the bother of looking at it yourself; Electric Monks believed things for you, thus saving you what was becoming an increasingly onerous task, that of believing all the things the world expected you to believe."
>
> Dirk Gently's Holistic Detective Agency, Douglas Adams

20.1. Filters and Martingales

Definition 20.1.1. A ***σ-field*** $(\Omega, \mathcal{F})$ consists of a sample space Ω (i.e., the atomic events) and a collection of subsets $\mathcal{F}$ satisfying the following conditions:
(A) $\emptyset \in \mathcal{F}$.
(B) $C \in \mathcal{F} \Rightarrow \overline{C} \in \mathcal{F}$.
(C) $C_1, C_2, \ldots \in \mathcal{F} \Rightarrow C_1 \cup C_2 \ldots \in \mathcal{F}$.

Definition 20.1.2. Given a σ-field $(\Omega, \mathcal{F})$, a ***probability measure*** $\mathbb{P} : \mathcal{F} \to \mathbb{R}^+$ is a function that satisfies the following conditions.
(A) $\forall A \in \mathcal{F}, 0 \leq \mathbb{P}[A] \leq 1$.
(B) $\mathbb{P}[\Omega] = 1$.
(C) For mutually disjoint events $C_1, C_2, \ldots$, we have $\mathbb{P}[\cup_i C_i] = \sum_i \mathbb{P}[C_i]$.

Definition 20.1.3. A ***probability space*** $(\Omega, \mathcal{F}, \mathbb{P})$ consists of a σ-field $(\Omega, \mathcal{F})$ with a probability measure $\mathbb{P}$ defined on it.

Definition 20.1.4. Given a σ-field $(\Omega, \mathcal{F})$ with $\mathcal{F} = 2^\Omega$, a ***filter*** (also ***filtration***) is a nested sequence $\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{F}_n$ of subsets of 2^Ω, such that:
(A) $\mathcal{F}_0 = \{\emptyset, \Omega\}$.
(B) $\mathcal{F}_n = 2^\Omega$.
(C) For $0 \leq i \leq n$, $(\Omega, \mathcal{F}_i)$ is a σ-field.

Definition 20.1.5. An ***elementary event*** or ***atomic event*** is a subset of a sample space that contains only one element of Ω.

Intuitively, when we consider a probability space, we usually consider a random variable X. The value of X is a function of the elementary event that happens in the probability space. Formally, a random variable is a mapping $X : \Omega \to \mathbb{R}$. Thus, each $\mathcal{F}_i$ defines a partition of Ω into *atomic events*. This partition is getting more and more refined as we progress down the filter.

Example 20.1.6. Consider an algorithm **Alg** that uses n random bits. As such, the underlying sample space is $\Omega = \{b_1 b_2 \ldots b_n \mid b_1, \ldots, b_n \in \{0,1\}\}$. That is, the set of all binary strings of length n. Next, let $\mathcal{F}_i$ be the σ-field generated by the partition of Ω into the atomic events B_w, where $w \in \{0,1\}^i$; here w is the string encoding the first i random bits used by the algorithm. Specifically,

$$B_w = \left\{ wx \in \Omega \;\middle|\; x \in \{0,1\}^{n-i} \right\},$$

and the set of atomic events in $\mathcal{F}_i$ is $\mathcal{A}_i = \left\{ B_w \;\middle|\; w \in \{0,1\}^i \right\}$. The set $\mathcal{F}_i$ is the *closure* of this set of atomic events under complement and union. In particular, we conclude that $\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_n$ form a filter.

As a concrete example, for $i = 3$, the set $\mathcal{A}_3$ contains $2^3 = 8$ sets, and the set $\mathcal{F}_3$ would contain all sets formed by finite unions of these sets (including the empty union). As such, the set $\mathcal{F}_3$ would have $2^{2^3} = 256$ sets.

Definition 20.1.7. A random variable X is said to be ***$\mathcal{F}_i$-measurable*** if for each $x \in \mathbb{R}$, the event $X \le x$ is in $\mathcal{F}_i$; that is, the set $\{\omega \in \Omega \mid X(\omega) \le x\}$ is in $\mathcal{F}_i$.

Example 20.1.8. Let $\mathcal{F}_0, \ldots, \mathcal{F}_n$ be the filter defined in Example 20.1.6. Let X be the parity of the n bits. Clearly, $X = 1$ is a valid event only in $\mathcal{F}_n$ (why?). Namely, it is only measurable in $\mathcal{F}_n$, but not in $\mathcal{F}_i$, for $i < n$.

As such, a random variable X is $\mathcal{F}_i$-measurable, only if it is a constant on the elementary events of $\mathcal{F}_i$. This gives us a new interpretation of what a filter is – its a sequence of refinements of the underlying probability space, that is achieved by splitting the atomic events of $\mathcal{F}_i$ into smaller atomic events in $\mathcal{F}_{i+1}$. Putting it explicitly, an atomic event $\mathcal{E}$ of $\mathcal{F}_i$, is a subset of 2^Σ. As we move to $\mathcal{F}_{i+1}$ the event $\mathcal{E}$ might now be split into several atomic (and disjoint events) $\mathcal{E}_1, \ldots, \mathcal{E}_k$. Now, naturally, the atomic event that really happens is an atomic event of $\mathcal{F}_n$. As we progress down the filter, we "zoom" into this event.

Definition 20.1.9 (Conditional expectation in a filter). Let $(\Omega, \mathcal{F})$ be any σ-field, and Y any random variable that takes on distinct values on the elementary events in $\mathcal{F}$. Then $\mathbb{E}[X \mid \mathcal{F}] = \mathbb{E}[X \mid Y]$.

20.2. Martingales

Definition 20.2.1. A sequence of random variables $Y_1, Y_2, \ldots$, is a ***martingale difference*** sequence if for all $i \ge 0$, we have $\mathbb{E}\left[Y_i \;\middle|\; Y_1, \ldots, Y_{i-1} \right] = 0$.

Clearly, $X_1, \ldots,$ is a martingale sequence if and only if $Y_1, Y_2, \ldots,$ is a martingale difference sequence where $Y_i = X_i - X_{i-1}$.

Definition 20.2.2. A sequence of random variables $Y_1, Y_2, \ldots,$ is

$$\text{a } \textbf{\textit{super martingale}} \text{ sequence if} \quad \forall i \quad \mathbb{E}\left[Y_i \;\middle|\; Y_1, \ldots, Y_{i-1} \right] \le Y_{i-1},$$
$$\text{and a } \textbf{\textit{sub martingale}} \text{ sequence if} \quad \forall i \quad \mathbb{E}\left[Y_i \;\middle|\; Y_1, \ldots, Y_{i-1} \right] \ge Y_{i-1}.$$

20.2.1. Martingales – an alternative definition

Definition 20.2.3. Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space with a filter $\mathcal{F}_0, \mathcal{F}_1, \ldots$. Suppose that $X_0, X_1, \ldots$, are random variables such that, for all $i \geq 0$, X_i is $\mathcal{F}_i$-measurable. The sequence $X_0, \ldots, X_n$ is a ***martingale*** provided that, for all $i \geq 0$, we have $\mathbb{E}[X_{i+1} \mid \mathcal{F}_i] = X_i$.

Lemma 20.2.4. *Let $(\Omega, \mathcal{F})$ and $(\Omega, \mathcal{G})$ be two σ-fields such that $\mathcal{F} \subseteq \mathcal{G}$. Then, for any random variable X, we have $\mathbb{E}[\mathbb{E}[X \mid \mathcal{G}] \,\big|\, \mathcal{F}] = \mathbb{E}[X \mid \mathcal{F}]$.*

Proof: $\mathbb{E}\Big[\mathbb{E}\Big[X \,\Big|\, \mathcal{G}\Big] \,\Big|\, \mathcal{F}\Big] = \mathbb{E}\Big[\mathbb{E}\Big[X \,\Big|\, G = g\Big] \,\Big|\, F = f\Big]$

$$
\begin{aligned}
&= \mathbb{E}\left[\frac{\sum_x x\,\mathbb{P}[X = x \cap G = g]}{\mathbb{P}[G = g]} \,\Big|\, F = f\right] = \sum_{g \in G} \frac{\frac{\sum_x x \mathbb{P}[X=x \cap G=g]}{\mathbb{P}[G=g]} \cdot \mathbb{P}[G = g \cap F = f]}{\mathbb{P}[F = f]} \\
&= \sum_{g \in \mathcal{G}, g \subseteq f} \frac{\frac{\sum_x x\,\mathbb{P}[X=x \cap G=g]}{\mathbb{P}[G=g]} \cdot \mathbb{P}[G = g \cap F = f]}{\mathbb{P}[F = f]} = \sum_{g \in \mathcal{G}, g \subseteq f} \frac{\frac{\sum_x x\,\mathbb{P}[X=x \cap G=g]}{\mathbb{P}[G=g]} \cdot \mathbb{P}[G = g]}{\mathbb{P}[F = f]} \\
&= \sum_{g \in \mathcal{G}, g \subseteq f} \frac{\sum_x x\,\mathbb{P}[X = x \cap G = g]}{\mathbb{P}[F = f]} = \frac{\sum_x x\Big(\sum_{g \in \mathcal{G}, g \subseteq f} \mathbb{P}[X = x \cap G = g]\Big)}{\mathbb{P}[F = f]} \\
&= \frac{\sum_x x\,\mathbb{P}[X = x \cap F = f]}{\mathbb{P}[F = f]} = \mathbb{E}\Big[X \,\Big|\, \mathcal{F}\Big].
\end{aligned}
$$

■

Theorem 20.2.5. *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space, and let $\mathcal{F}_0, \ldots, \mathcal{F}_n$ be a filter with respect to it. Let X be any random variable over this probability space and define $X_i = \mathbb{E}\Big[X \,\Big|\, \mathcal{F}_i\Big]$ then, the sequence $X_0, \ldots, X_n$ is a martingale.*

Proof: We need to show that $\mathbb{E}\Big[X_{i+1} \,\Big|\, \mathcal{F}_i\Big] = X_i$. Namely,

$$
\mathbb{E}[X_{i+1} \mid \mathcal{F}_i] = \mathbb{E}\Big[\mathbb{E}\Big[X \,\Big|\, \mathcal{F}_{i+1}\Big] \,\Big|\, \mathcal{F}_i\Big] = \mathbb{E}\Big[X \,\Big|\, \mathcal{F}_i\Big] = X_i,
$$

by Lemma 20.2.4 and by definition of X_i. ■

Definition 20.2.6. Let $f : \mathcal{D}_1 \times \cdots \times \mathcal{D}_n \to \mathbb{R}$ be a real-valued function with a arguments from possibly distinct domains. The function f is said to satisfy the ***Lipschitz condition*** if for any $x_1 \in \mathcal{D}_1, \ldots, x_n \in \mathcal{D}_n$, and $i \in \{1, \ldots, n\}$ and any $y_i \in \mathcal{D}_i$, we have

$$
\Big| f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n) \Big| \leq 1.
$$

Specifically, a function is ***c-Lipschitz***, if the inequality holds with a constant c (instead of 1).

Definition 20.2.7. Let $X_1, \ldots, X_n$ be a sequence of *independent* random variables, and a function $f = f(X_1, \ldots, X_n)$ defined over them, such that f satisfies the Lipschitz condition. The ***Doob martingale*** sequence $Y_0, \ldots, Y_m$ is defined by $Y_0 = \mathbb{E}[f(X_1, \ldots, X_n)]$ and

$$
Y_i = \mathbb{E}[f(X_1, \ldots, X_n) \,\big|\, X_1, \ldots, X_i], \qquad \text{for} \quad i = 1, \ldots, n.
$$

Clearly, a Doob martingale $Y_0, \ldots, Y_n$ is a martingale, by Theorem 20.2.5. Furthermore, if $|X_i - X_{i-1}| \leq 1$, for $i = 1, \ldots, n$, then $|Y_i - Y_{i-1}| \leq 1$. and we can use Azuma's inequality on such a sequence.

20.3. Occupancy Revisited

We have m balls thrown independently and uniformly into n bins. Let Z denote the number of bins that remains empty in the end of the process. Let X_i be the bin chosen in the ith trial, and let $Z = F(X_1, \ldots, X_m)$, where F returns the number of empty bins given that m balls had thrown into bins $X_1, \ldots, X_m$. , By Azuma's inequality we have that $\mathbb{P}\Big[\big|Z - \mathbb{E}[Z]\big| > \lambda\sqrt{m}\Big] \leq 2\exp\big(-\lambda^2/2\big)$.

The following is an extension of Azuma's inequality shown in class. We do not provide a proof but it is similar to what we saw.

Theorem 20.3.1 (Azuma's Inequality - Stronger Form). *Let $X_0, X_1, \ldots,$ be a martingale sequence such that for each k, $|X_k - X_{k-1}| \leq c_k$, where c_k may depend on k. Then, for all $t \geq 0$, and any $\lambda > 0$, we have*

$$\mathbb{P}[|X_t - X_0| \geq \lambda] \leq 2\exp\Big(-\frac{\lambda^2}{2\sum_{k=1}^{t} c_k^2}\Big).$$

Theorem 20.3.2. *Let $r = m/n$, and Z_{end} be the number of empty bins when m balls are thrown randomly into n bins. Then $\mu = \mathbb{E}[Z_{end}] = n\Big(1 - \frac{1}{n}\Big)^m \approx n\exp(-r)$, and for any $\lambda > 0$, we have*

$$\mathbb{P}\Big[\big|Z_{end} - \mu\big| \geq \lambda\Big] \leq 2\exp\left(-\frac{\lambda^2(n - 1/2)}{n^2 - \mu^2}\right).$$

Proof: Let $z(Y, t)$ be the expected number of empty bins in the end, if there are Y empty bins in time t. The probability of an empty bin to remain empty is $(1 - 1/n)^{m-t}$, and as such

$$z(Y, t) = Y\Big(1 - \frac{1}{n}\Big)^{m-t}.$$

In particular, $\mu = z(n, 0) = n(1 - 1/n)^m$.

Let $\mathcal{F}_t$ be the σ-field generated by the bins chosen in the first t steps. Let Z_{end} be the number of empty bins at time m, and let $Z_t = \mathbb{E}\Big[Z_{\text{end}} \,\Big|\, \mathcal{F}_t\Big]$. Namely, Z_t is the expected number of empty bins after we know where the first t balls had been placed. The random variables $Z_0, Z_1, \ldots, Z_m$ form a martingale. Let Y_t be the number of empty bins after t balls where thrown. We have $Z_{t-1} = z(Y_{t-1}, t - 1)$. Consider the ball thrown in the t-step. Clearly:

(A) With probability $1 - Y_{t-1}/n$ the ball falls into a non-empty bin. Then $Y_t = Y_{t-1}$, and $Z_t = z(Y_{t-1}, t)$. Thus,

$$\begin{aligned}\Delta_t = Z_t - Z_{t-1} &= z(Y_{t-1}, t) - z(Y_{t-1}, t - 1) = Y_{t-1}\left(\left(1 - \frac{1}{n}\right)^{m-t} - \left(1 - \frac{1}{n}\right)^{m-t+1}\right) \\ &= \frac{Y_{t-1}}{n}\left(1 - \frac{1}{n}\right)^{m-t} \leq \left(1 - \frac{1}{n}\right)^{m-t}.\end{aligned}$$

(B) Otherwise, with probability Y_{t-1}/n the ball falls into an empty bin, and $Y_t = Y_{t-1} - 1$. Namely, $Z_t = z(Y_t - 1, t)$. And we have that

$$\begin{aligned}\Delta_t = Z_t - Z_{t-1} &= z(Y_{t-1} - 1, t) - z(Y_{t-1}, t - 1) = (Y_{t-1} - 1)\left(1 - \frac{1}{n}\right)^{m-t} - Y_{t-1}\left(1 - \frac{1}{n}\right)^{m-t+1} \\ &= \left(1 - \frac{1}{n}\right)^{m-t}\left(Y_{t-1} - 1 - Y_{t-1}\left(1 - \frac{1}{n}\right)\right) = \left(1 - \frac{1}{n}\right)^{m-t}\left(-1 + \frac{Y_{t-1}}{n}\right) = -\left(1 - \frac{1}{n}\right)^{m-t}\left(1 - \frac{Y_{t-1}}{n}\right) \\ &\geq -\left(1 - \frac{1}{n}\right)^{m-t}.\end{aligned}$$

Thus, $Z_0, \ldots, Z_m$ is a martingale sequence, where $|Z_t - Z_{t-1}| \leq |\Delta_t| \leq c_t$, where $c_t = \left(1 - \frac{1}{n}\right)^{m-t}$. We have

$$\sum_{t=1}^{m} c_t^2 = \sum_{t=1}^{m} \left(1 - \frac{1}{n}\right)^{2(m-t)} = \sum_{t=0}^{m-1} \left(1 - \frac{1}{n}\right)^{2t} = \frac{1 - (1 - 1/n)^{2m}}{1 - (1 - 1/n)^2} = \frac{n^2\left(1 - (1 - 1/n)^{2m}\right)}{2n - 1} = \frac{n^2 - \mu^2}{2n - 1}.$$

Now, deploying Azuma's inequality, yield the result. ■

20.3.1. Lets verify this is indeed an improvement

Consider the case where $m = n \ln n$. Then, $\mu = n\left(1 - \frac{1}{n}\right)^m \leq 1$. And using the "weak" Azuma's inequality implies that

$$\mathbb{P}\left[\left|Z_{\text{end}} - \mu\right| \geq \lambda \sqrt{n}\right] = \mathbb{P}\left[\left|Z_{\text{end}} - \mu\right| \geq \lambda \sqrt{\frac{n}{m}} \sqrt{m}\right] \leq 2 \exp\left(-\frac{\lambda^2 n}{2m}\right) = 2 \exp\left(-\frac{\lambda^2}{2 \ln n}\right),$$

which is interesting only if $\lambda > \sqrt{2 \ln n}$. On the other hand, Theorem 20.3.2 implies that

$$\mathbb{P}\left[|Z_{\text{end}} - \mu| \geq \lambda \sqrt{n}\right] \leq 2 \exp\left(-\frac{\lambda^2 n(n - 1/2)}{n^2 - \mu^2}\right) \leq 2 \exp\left(-\lambda^2\right),$$

which is interesting for any $\lambda \geq 1$ (say).

Chapter 21

The power of two choices

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> The Peace of Olivia. How sweat and peaceful it sounds! There the great powers noticed for the first time that the land of the Poles lends itself admirably to partition.
>
> The tin drum, Gunter Grass

Consider the problem of throwing n balls into n bins. It is well known that the maximum load is $\Theta(\log n / \log\log n)$ with high probability. Here we show that if one is allowed to pick d bins for each ball, and throw it into the bin that contains less balls, then the maximum load of a bin decreases to $\Theta(\log\log n / \log d)$. A variant of this approach leads to maximum load $\Theta((\log\log n)/d)$.

As a concrete example, for $n = 10^9$, this leads to maximum load 13 in the regular case, compared to maximum load of 4, with only two-choices – see Figure 21.1.

21.1. Balls and bins with many rows

21.1.1. The game

Consider throwing n balls into n bins. Every bin can contain a single ball. As such, as we throw the balls, some balls would be rejected because their assigned bin already contains a ball. We collect all the rejected balls, and throw them again into a second row of n bins. We repeat this process till all the balls had found a good and loving home (i.e., an empty bin). How many rows one needs before this process is completed?

21.1.2. Analysis

Lemma 21.1.1. *Let $m = \alpha n$ balls be thrown into n bins. Let Y_{end} the number of bins that are not empty in the end of the process (here, we allow more than one ball into a bin).*

(A) For $\alpha \in (0,1]$, we have $\mu = \mathbb{E}[Y_{\text{end}}] \geq (m-\alpha)\exp(-\alpha) \geq \alpha n - \alpha^2 n - 1$.

(B) If $\alpha \geq 1$, then $\mu = \mathbb{E}[Y_{\text{end}}] \geq n(1-\exp(-\alpha))$.

(C) We have $\mathbb{P}[|Y_{\text{end}} - \mu| > \sqrt{3cm\log n}] \leq 1/n^c$.

Proof: (A) The probability of the ith ball to be the first ball in its bin, is $\left(1-\frac{1}{n}\right)^{i-1}$. To see this we use backward analysis – throw in the ith ball, and now throw in the earlier $i-1$ balls. The probability that none of the earlier

balls hit the same bin as the ith ball is as stated. Now, the expected number of non-empty bins is the number of balls that are first in their bins, which in turn is

$$\begin{aligned}\mu = \sum_{i=0}^{m-1}\left(1-\frac{1}{n}\right)^i &\geq m(1-1/n)^{m-1} \geq (m-\alpha)(1-1/n)^{m-\alpha} \\ &= (m-\alpha)(1-1/n)^{\alpha(n-1)} \geq (m-\alpha)\exp(-\alpha) \\ &\geq (m-\alpha)(1-\alpha) = \alpha n - \alpha^2 n - \alpha + \alpha^2 \geq \frac{m-\alpha}{e},\end{aligned}$$

using $m = \alpha n \leq n$, and $(1-1/n)^{n-1} \geq 1/e$, see Lemma 6.1.1.

(B) We repeat the above analysis from the point of view of the bin. The probability of a bin to be empty is $(1-1/n)^{\alpha n}$. As such, we have that

$$\mu = \mathbb{E}[Y_{\text{end}}] = n(1-(1-1/n)^{\alpha n}) \geq n(1-\exp(-\alpha)),$$

using $1 - 1/n \leq \exp(-1/n)$.

(C) Let X_i be the index of the bin the ith ball picked. Let $Y_i = \mathbb{E}[Y_{\text{end}} \mid X_1, \ldots, X_i]$. This is a Doob martingale, with $|Y_i - Y_{i-1}| \leq 1$. As such, Azuma's inequality implies, for $\lambda = \sqrt{3cm \ln n}$, that

$$\mathbb{P}[|Y_{\text{end}} - \mathbb{E}[Y_{\text{end}}]| \geq \lambda] \leq 2\exp\left(-\lambda^2/2m\right) \leq 1/n^c. \qquad \blacksquare$$

Remark. The reader might be confused by cases (A) and (B) of Lemma 21.1.1 for $\alpha = 1$, as the two lower bounds are different. Observe that (A) is loose if α is relatively large and close to 1.

Back to the problem. Let $\alpha_1 = 1$ and $n_1 = \alpha_1 n$. For $i > 1$, inductively, assume that numbers of balls being thrown in the ith round is

$$n_i = \alpha_i n + O(\sqrt{\alpha_{i-1} n \log n}).$$

By Lemma 21.1.1, with high probability, the number of balls stored in the ith row is

$$s_i = n_i \exp(-\alpha_i) \pm O(\sqrt{n_i \log n}).$$

As such, as long as the first term is significantly large than the second therm, we have that $s_i = n\alpha_i \exp(-\alpha_i)(1 \pm o(1))$. For the time being, let us ignore the $o(1)$ term. We have that

$$n_{i+1} = n_i - s_i = n(\alpha_i - \alpha_i \exp(-\alpha_i)) \leq n(\alpha_i - \alpha_i(1-\alpha_i)) = n\alpha_i^2,$$

since $\exp(-\alpha_i) \geq 1 - \alpha_i$.

Definition. For a number $x > 0$, we use $\lg x = \log_2 x$.

Observation 21.1.2. *Consider the sequence $\alpha_1 = 1$, $c = \alpha_2 = 1 - 1/e$, and $\alpha_{i+1} = \alpha_i^2$, for $i > 2$. We have that $\alpha_{i+1} = c^{2^{i-2}}$. In particular, for*

$$\Delta = 3 + \lg \log_{1/c} n = 3 + \lg \frac{\lg n}{\lg(1/c)} = 3 + \lg\lg n - \lg\lg \frac{1}{1-1/e} \leq 3 + \lg\lg n.$$

we have that $\alpha_\Delta = c^{2^{\Delta-2}} < 1/n$.

The above observation almost implies that we need Δ rows. The problem is that the above calculations (i.e., the high probability guarantee in Lemma 21.1.1) breaks down when $n_i = O(\log n)$ – that is, when $\alpha_i = O((\log n)/n)$. However, if one throws in $O(\log n)$ balls into n bins, the probability of a single collision is at most $O((\log n)^2/n)$. In particular, this implies that after roughly additional c rows, the probability of any ball left is $\leq 1/n^c$.

The above argumentation, done more carefully, implies the following – we omit the details because (essentially) the same analysis for a more involved case is done next (the lower bound stated follows also from the same argumentation).

Theorem 21.1.3. *Consider the process of throwing n balls into n bins in several rounds. Here, a ball that can not be placed in a round, because their chosen bin is already occupied, are promoted to the next round. The next round throws all the rejected balls from the previous round into a new row of n empty bins. This process, with high probability, ends after $M = \lg\lg n + \Theta(1)$ rounds (i.e., after M rounds, all balls are placed in bins).*

21.1.3. With only d rows

Lemma 21.1.4. *For $\alpha \in (0, 1/4]$, let $\gamma_1 = \alpha$, and $\gamma_i = 2\gamma_{i-1}^2$. We have that $\gamma_{d+1} \leq \alpha^{(2^d+1)/2}$.*

Proof: The proof, minimal as it may be, is by induction:

$$\gamma_{i+1} = 2\gamma_i^2 \leq 2\left(\alpha^{(2^{i-1}+1)/2}\right)^2 = 2\alpha^{(2^i+2)/2} \leq \alpha^{(2^i+1)/2},$$

since $2\sqrt{\alpha} \leq 1$. ■

Lemma 21.1.5. *Let $m = \alpha n$ balls be thrown into n bins, with d rows, where $\alpha > 0$. Here every bin can contain only a single ball, and if inserting the ball into ith row failed, then we throw it in the next row, and so on, till it finds an empty bin, or it is rejected because it failed on the dth row. Let $Y(d, n, m)$ be the number of balls that did not get stored in this matrix of bins. We have*

(A) For a constant $\alpha < 1/4$, we have $Y(d, n, \alpha n) \leq n\alpha^{(2^d+1)/2}$, with high probability.

(B) We $\mathbb{E}[Y(d, n, dn)] = O(n \log d)$.

(C) For a constant $c > 1$, we have $\mathbb{E}[Y(d, n, cn \log d)] = n/e^{-d/2}$, assuming d is sufficiently large.

Proof: (A) By Lemma 21.1.1, in expectation, at least $s_1 = n\alpha \exp(-\alpha)$ balls are placed in the first row. As such, in expectation $n_2 = n\alpha(1 - \exp(-\alpha)) \leq n\alpha^2$ balls get thrown into the second row. Using Chenroff inequality, we get that $n_2 \leq 2\alpha^2 n$, with high probability. Setting $\gamma_1 = \alpha$, and $\gamma_i = 2\gamma_{i-1}^2$, we get the claim via Lemma 21.1.4.

(B) As long as we throw $\Omega(n \log d)$ balls into a row, we expect by Lemma 21.1.1 that at least $n(1 - 1/d^{O(1)})$ balls to get stored in this row. As such, let $D = O(\log d)$, and observe that the first $d - D$ rows in expectation contains $n(d - D)(1 - 1/d^{O(1)})$ balls. This implies that only $O(Dn)$ are not stored in these first $d - D$ rows, which implies the claim.

(C) Break the d rows into two groups. The first group of size

$$D = \lceil (c \log d - 1)/(1 - 1/e) \rceil + 1 = O(\log d),$$

and the second group is the remaining rows. As long as the number of balls arriving to a row is larger than n, we expect at least $n(1 - 1/e)$ of them to be stored in this row. As such, after the first D rows, we expect the number of remaining balls to be $\leq n$. Indeed, if we have i such rows, then the expected number of balls moving on to the $(i + 1)$th row is at most

$$n_{i+1} = cn \log d - in(1 - 1/e).$$

Solving for $n_{i+1} \leq n$, we have $cn \log d - in(1-1/e) \leq n \implies i(1-1/e) \geq c \log d - 1 \implies i \geq (c \log d - 1)/(1-1/e) \geq D - 1$. As such, $n_D \leq n$, for $i \geq D$.

The same argumentation implies that the number of balls arriving to the $D+i$ row, in expectation, is at most n/e^i. In particular, we get that the number of balls failed to be placed is at most $n/e^{d-D} \leq n/e^{d/2}$. ■

21.2. The power of two choices

Making d choices. Let us throw n balls into n bins. For each ball, we first pick randomly $d \geq 2$ bins, and place the ball in the bin (among these d bins) that currently contains the smallest number of balls (here, a bin might contain an arbitrary number of balls). If there are several bins with the same minimum number of bins, we resolve it arbitrarily.

Here, we will show the surprising result that the maximum number of balls in any bin is bounded by $O(\frac{\log \log n}{\log d})$ with high probability in the end of this process. For $d = 1$, which is the regular balls into bins setting, we already seen that this quantity is $\Theta(\frac{\log n}{\log \log n})$, so this result is quite surprising.

21.2.1. Upper bound

Definition 21.2.1. The ***load*** of a bin is the number of balls in it. The ***height*** of a ball, is the load of the bin it was inserted into, just after it was inserted.

Some notations:

(A) β_i: An upper bound on the number of bins that have load at least i by the end of the process.
(B) $h(i)$: The height of the ith ball.
(C) $\sqcup_{\geq i}(t)$: Number of bins with load at least i at time t.
(D) $\text{⚽}_{\geq i}(t)$: Number of balls with height at least i at time t.

Observation 21.2.2. $\sqcup_{\geq i}(t) \leq \text{⚽}_{\geq i}(t)$.

Let $⊎_{\geq i} = \sqcup_{\geq i}(n)$ be the number of bins, in the end of the process, that have load $\geq i$.

Observation 21.2.3. *Since every bin counted in $⊎_{\geq i}$ contains at least i balls, and there are n balls, it follows that $⊎_{\geq i} \leq n/i$. In particular, we have $⊎_{\geq 4} \leq n/4$.*

Lemma 21.2.4. *Let $\beta_1 = n, \beta_2 = n/2, \beta_3 = n/3$, and $\beta_4 = n/4$, and let*

$$\beta_{i+1} = 2n(\beta_i/n)^d,$$

for $i \geq 4$. Let I be the last iteration, such that $\beta_I \geq 16c \ln n$, where $c > 1$ is an arbitrary constant. Then, with probability $\geq 1 - 1/n^c$, we have that

(A) $⊎_{\geq i} \leq \beta_i$, for $i = 1, \ldots, I$.
(B) $⊎_{\geq I+1} \leq c' \log n$, for some constant c'.
(C) For $j > 0$, and any constant $\varepsilon > 0$, we have $\mathbb{P}\left[⊎_{\geq I+1+j} > 0\right] \leq O(1/n^{(d-1-\varepsilon)j})$.
(D) With probability $\geq 1 - 1/n^c$, the maximum load of a bin is $I + O(c)$.

Proof: (A) The claim for $i = 1, 2, 3, 4$ follows readily from Observation 21.2.3.

Let $\mathcal{B}_i$ be the bad event that $⊎_{\geq i} > \beta_i$, for $i = 1, \ldots, n$. The following analysis is conditioned on none of these bad events happening. Let $\mathcal{G}_k = \cap_{i=1}^{k} \overline{\mathcal{B}_i}$ be the good event. Let Y_t be an indicator variable that is one

$\iff$ $h(t) \geq i+1$ conditioned on $\mathcal{G}_{i-1}$ (for clarity, we omit mentioning this conditioning explicitly). We have that

$$\tau_j = \mathbb{P}\Big[Y_j = 1\Big] \leq p_i \qquad \text{for} \qquad p_i = (\beta_i/n)^d,$$

as all d probes must hit bins of height at least i, and there are at most β_i such bins. This readily implies that $\mathbb{E}[\text{⚽}_{\geq i+1}(n)] \leq p_i n$. The variables $Y_1, \ldots, Y_n$ are not independent, but consider a variable Y'_j that is 1 if $Y_j = 1$, or if $Y_j = 0$, then Y'_j is 1 with probability $p_i - \tau_j$. Clearly, the variables $Y'_1, \ldots, Y'_n$ are independent, and $\sum_i Y'_j \geq \sum_i Y_i$. For $i < I$, setting

$$\beta_{i+1} = 2np_i = 2n(\beta_i/n)^d,$$

we have, by Chernoff's inequality, that

$$\begin{aligned}\alpha_{i+1} = \mathbb{P}[\mathcal{B}_{i+1}] = \mathbb{P}[\text{⚽}_{\geq i+1}(n) > \beta_{i+1}] = \mathbb{P}[\text{⚽}_{\geq i+1}(n) > 2np_i] &\leq \mathbb{P}\Big[\sum_i Y'_t > (1+1)np_i\Big] \\ &\leq \exp(-np_i/4) = \exp(-\beta_{i+1}/8) < 1/n^{2c}.\end{aligned}$$

(B) We have $\beta_{I+1} \leq 16c \log n$. Setting $\Delta = 2e \cdot 16c \log n$, and conditioning on the good event $\mathcal{G}_1$, consider the sequence $Y'_1, \ldots Y'_n$ as above, where the Y_i is the indicator that the ith ball has height $\geq I+1$. Arguing as above, for $Y' = \sum_i Y'_i$, we have $\mathbb{E}[Y] \leq \beta_{i+1}$. As such, we have

$$\mathbb{P}[\text{⊎}_{\geq I+1} > \Delta] \leq \mathbb{P}[\text{⚽}_{\geq I+1}(n) > \Delta] \leq \mathbb{P}\left[Y' > \frac{\Delta}{\mathbb{E}[Y']}\,\mathbb{E}[Y']\right] \leq 2^{-\Delta} \leq \frac{1}{n^c},$$

by Lemma 13.2.8, as $\mathbb{E}[Y'] \leq \beta_{I+1}$, and $\Delta/\beta_{I+1} > 2e$.

As for the conditioning used in the above, we have that

$$\mathbb{P}[\mathcal{G}_{I+1}] = \prod_{\ell=4}^{I+1} \mathbb{P}\Big[\overline{\mathcal{B}_{\ell+1}} \;\Big|\; \cap_{k=1}^{\ell} \overline{\mathcal{B}_1}\Big] = \prod_i (1-\alpha_i) \geq 1 - 1/n^{c-1},$$

since $I \leq n$.

(C) Observe that $\sqcup_{\geq i+1}(n) \leq \sqcup_{\geq i}(n)$. As such, for all $j > 0$, we have that $\sqcup_{\geq I+1+j}(n) \leq \text{⚽}_{\geq I+1}(n) \leq \Delta = 2e \cdot 16c \log n$, by (B). As such, we have

$$\mathbb{E}\Big[\text{⚽}_{\geq I+1+j}(n)\Big] \leq n(\Delta/n)^d = O(\log^d n/n^{d-1}) = O(1/n^{d-1-\varepsilon}) \ll 1,$$

for $\varepsilon > 0$ an arbitrary constant, and n sufficient large. Using Markov's inequality, we get that $q = \mathbb{P}\Big[\text{⚽}_{\geq I+1+j}(n) \geq 1\Big] = O(1/n^{d-1-\varepsilon})$. The probability that the first j such rounds fail (i.e., that $\text{⚽}_{\geq I+1+j}(n) > 0$) is at most q^j, as claimed.

(D) This follows immediately by picking $\varepsilon = 1/2$, and then using (C) with $j = O(c)$. ■

Lemma 21.2.5. *For $i = 4, \ldots, I$, we have that $\beta_i \leq n/2^{d^{i-4}+1}$.*

Proof: The proof is by induction. For $i = 4$, we have $\beta_4 \leq n/4$, as claimed. Otherwise, we have

$$\beta_{i+1} = 2n(\beta_i/n)^d \leq 2n\Big(1/2^{d^{i-4}+1}\Big)^d = n/2^{d^{i+1-4}+d-1} \leq n/2^{d^{i+1-4}+1}.$$ ■

Theorem 21.2.6. *When throwing n balls into n bins, with d choices, with probability $\geq 1 - 1/n^{O(1)}$, we have that the maximum load of a bin is $O(1) + \frac{\lg\lg n}{\lg d}$*

Proof: By Lemma 21.2.4, with the desired probability the β_is bound the load in the bins for $i \leq I$. By Lemma 21.2.5, it follows that for $I = O(1) + \frac{\lg\lg n}{\lg d}$, we have that $\beta_I \leq o(\log n)$. Thus giving us the desired bound. ■

It is not hard to verify that our upper bounds (i.e., β_i) are not too badly off, and as such the maximum load in the worst case is (up to additive constant) the same. We state the result without proof.

Theorem 21.2.7. *When throwing n balls into n bins, with d choices (where the ball is placed with the bin with the least load), with probability $\geq 1 - o(1/n)$, we have that the maximum load of a bin is at least $\frac{\lg\lg n}{\lg d} - O(1)$.*

21.2.2. Applications

As a direct application, we can use this approach for open hashing, where we use two hash functions, and place an element in the bucket of the hash table with fewer elements. By the above, this improves the worst case search time from $O(\log n/\log\log n)$ to $O(\log\log n)$. This comes at the cost of doubling the time it takes to do lookup on average.

21.2.3. The power of restricted d choices: Always go left

The always go left rule. Consider throwing a ball into n bins (which might already have some balls in them) as follows – you pick uniformly a number $X_i \in [\![n/d]\!]$, for $i = 1, \ldots, d$. Next, you try locations $Y_1, \ldots, Y_d$, where $Y_j = X_j + j(n/d)$, for $j = 1, \ldots, d$. Let L_j be the load of bin Y_j, for $j = 1, \ldots, d$, and let $L = \min_j L_j$ be the minimum load of any bin. Let τ be the minimum index such that $L_j = L$. We throw the ball into Y_τ.

What the above scheme does, is to partition the n bins into d groups each of size n/d, placed from left to right. We pick a bin uniformly from each group, and always throw the ball in the leftmost location that realizes the minimum load.

The following proof is informal for the sake of simplicity.

Theorem 21.2.8. *When throwing n balls into n bins, using the always-go-left rule, with d groups of size n/d, the maximum load of a bin is $O(1) + \frac{\log\log n}{d}$, with high probability.*

Proof: (Sketch.) We consider each of the d groups to be a row in the matrix being filled. So each row has n/d entries, and there are d rows. We can now think about the above algorithm as first trying to place the ball in the first row (if there is an empty bin), otherwise, trying the new row and so on. If all the d locations are full, in the row filling game we fail to place this ball. By Lemma 21.1.5 (B), we have that the number of unplaced balls is $\mathbb{E}[Y(d, n/d, (n/d)d)] = O((n/d)\log d)$. Thus, we have that the number of balls that get placed as the first ball in their bin is

$$\geq n\Big(1 - \frac{O(\log d)}{d}\Big),$$

and the height of these balls is one.

We now use the same argumentation for balls of height 2 – Lemma 21.1.5 (C) implies that at most $dn/e^{-d/2}$ balls have height strictly larger than 2.

Lemma 21.1.5 (A) implies that now we can repeat the same analysis as the power of two choices, the critical difference is that every one of the d groups, behaves like a separate height. Since there are $O(\log\log n)$ maximum height in the regular analysis, this implies that we get $O((\log\log n)/d)$ maximum load, with high probability. ■

# balls in bin	Regular	2-choices	2-choices+go left
0	369,899,815	240,525,897	228,976,604
1	365,902,266	528,332,061	546,613,797
2	182,901,437	221,765,420	219,842,639
3	61,604,865	9,369,389	4,566,915
4	15,760,559	7,233	45
5	3,262,678		
6	568,919		
7	86,265		
8	11,685		
9	1,347		
10	143		
11	17		
12	2		
13	2		

Figure 21.1: Simulation of the three schemes described here. This was done with $n = 1,000,000,000$ balls thrown into n bins. Since $\log\log n$ is so small (i.e., ≈ 3 in this case, there does not seem to be any reasonable cases where the is a significant differences between d-choices and the go-left variant. In the simulations, the *go-left* variant always has a somewhat better distribution, as shown above.

21.3. Avoiding terrible choices

Interestingly, one can prove that two choices are not really necessary. Indeed, consider the variant where the ith ball randomly chooses a random location r_i. The ball then is placed in the bin with least load among the bins r_i and r_{i-1} (the first ball inspects only a single bin – r_1). It is not difficult to show that the above analysis applies in this settings, and the maximum load is $O(\log\log n)$ – despite making only n choices for n balls. Intuitively, what is going on is that the power of two choices lies in the ability to avoid following a horrible, no good, terrible choice, by having an alternative. This alternative choice does not have to be quite of the same quality as the original choice - it can be stolen from the previous ball, etc.

21.4. Escalated choices

A variant that seems to work even better in practice, is the following ***escalated choices*** algorithm: The idea is to try more than one bin only if you need to. To this end, try a random bin. If it is empty, then the algorithm stores the ball in it. Otherwise, the algorithm tries harder. In the jth iteration, for $j > 1$, the algorithm picks a random location. If any of the j locations have load $< \lceil j/2 \rceil$, then the algorithm places the ball in the min-load bin among these. Otherwise, the algorithm continues to the next iteration.

Experiments shows that on average, this algorithm only probes 1.96 bins per ball (thus, making less probes than 2-choices). In this settings, the experiments show that 4-choices with move left do better, but if one use the threshold $< \lceil j/3 \rceil$, then the average number of probes is 2.30179, while having again a better performance. The intuition is that a sequence of really bad choices are rare, and one can afford to try harder in such cases to get out of them.

A theoretical analysis of this variant should be interesting.

(I "invented" this variant, but it might already be known.)

21.5. Bibliographical notes

The multi-row balls into bins (Section 21.1) is from the work by Broder and Karlin [BK90]. The power of two choices (Section 21.2) is from Azar *et al.* [ABKU99].

The restricted d choices structure, the always go-left rule, described in Section 21.2.3, is from [Vöc03].

References

[ABKU99] Y. Azar, A. Broder, A. Karlin, and E. Upfal. Balanced allocations. *SIAM Journal on Computing*, 29(1): 180–200, 1999.

[BK90] A. Z. Broder and A. R. Karlin. Multilevel adaptive hashing. *Proc. 1th ACM-SIAM Sympos. Discrete Algs.* (SODA), 43–53, 1990.

[Vöc03] B. Vöcking. How asymmetry helps load balancing. *J. ACM*, 50(4): 568–589, 2003.

Chapter 22

Evaluating And/Or Trees

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

That's all a prophet is good for - to admit somebody else is an ass or a whore.

The Violent Bear It Away, Flannery O'connor

22.1. Evaluating an And/Or Tree

Let T_{2k} denote a complete binary tree of height $2k$ – this tree has $n = 2^{2k}$ leaves. The inputs to the tree are boolean values stored in the leafs, where nodes are AND/OR nodes alternatingly. The task at hand is to evaluate the tree - where the value of a internal node, is the operation associated with the node, applied to the values returned from evaluating its two children.

Figure 22.1: The tree T_2, inputs in the leafs, and the output.

Defined recursively, T_2 is a tree with the root being an AND gate, and its children are OR gates. This tree has four inputs. More generally, T_{2k}, is T_2, with each leaf replaced by T_{2k-2}. Let $n = 2^{2k}$.

So the input here is T_{2k}, together with 2^{2k} values stored in each leaf of the tree. Consider here the *query model* – instead of read the values in the leafs, the algorithm has to explicitly perform a query to get the value stored in the leaf. The question thus is can we minimize the number of queries the algorithm needs to perform.

It is straightforward to evaluate such a tree using a recursive algorithm in $O(n)$ time. In particular, it following is not too difficult to show.

Exercise 22.1.1. Show that any deterministic algorithm, in the worst case, requires $\Omega(n)$ time to evaluate a tree T_{2k}.

The key observation is that AND (i.e., $\wedge$) gate evaluation can be shortcut – that is, if $x = 0$ then $x \wedge y = 0$ independently on what value y has. Similarly, an OR (i.e., $\vee$) gate evaluation can be shortcut – since if $x = 1$, then $x \vee y = 1$ independently of what y value is.

22.1.1. Randomized evaluation algorithm for T_{2k}

The algorithm is recursive. If the current node v is a leaf, the algorithm returns the value stored at the leaf. Otherwise, the algorithm randomly chooses (with equal probability) one of the children of v, and evaluate them recursively. If the returned value, is sufficient to evaluate the gate at v, then the algorithm shortcut. Otherwise, the algorithm evaluates recursively the other child, computes the value of the gate and return it.

22.1.2. Analysis

Lemma 22.1.2. *The above algorithm when applied to T_{2k}, in expectation, reads the value of at most 3^k leaves, and this also bounds its running time.*

Proof: The proof is by induction. Let start with T_2. There are two possibilities:

(i) The tree evaluates to 0, then one of the children of the AND gate evaluates zero. But then, with probability half the algorithm would guess the right child, and evaluate it first. Thus, in this case, the algorithm would evaluate (in expectation) $\leq (1/2)2 + (1/2)4 = 3$ leafs.

(ii) If the output of the tree is 1, then both children of the root must evaluate to 1. Each one of them is an OR gate. Arguing as above, an OR gate evaluating to one, requires in expectation to read $(1/2)1+(1/2)2 = 3/2$ leafs to be evaluated by the randomized algorithm. It follows, that in this case, the algorithm would read (in expectation) $2(3/2) = 3$. (Note, that this is an upper bound – if all the four inputs are 1, this algorithm would read only 2 leafs.)

For $k > 1$, consider the four grandchildren of the root c_1, c_2, c_3, c_4. By induction, in expectation, evaluating each of $c_1, \ldots, c_4$, takes 3^{k-1} leaf evaluations. Let $X_1, \ldots, X_4$ be indicator variables that are one if c_i is evaluated by the recursive algorithm. Let Y_i be the expected number of leafs read when evaluating c_i (i.e., $\mathbb{E}[Y_i] = 3^k$). By the above, we have that $\mathbb{E}[\sum_i X_i] = 3$. Observe that X_i and Y_i are independent. (Note, that the X_i are not independent of each other.) We thus have that the expected number of leafs to be evaluated by the randomized algorithm is

$$\mathbb{E}\left[\sum_i X_i Y_i\right] = \sum_i \mathbb{E}[X_i Y_i] = \sum_i \mathbb{E}[X_i]\,\mathbb{E}[Y_i] \leq 3\,\mathbb{E}[Y_i] = 3 \cdot 3^{k-1} = 3^k. \qquad \blacksquare$$

Corollary 22.1.3. *Given an AND/OR tree with n leafs, the above algorithm in expectation evaluates*

$$3^k = 2^{k \log_2 3} = \left(2^{2k(\log_2 3)/2}\right) = n^{(\log_2 3)/2} = n^{0.79248}$$

leafs.

22.2. Bibliographical notes

The AND/OR tree algorithm is from Marc Snir work [Sni85]. One can show a lower bound using Yao's min-max principle, which is implied by the minimax principle of zero sum games.

References

[Sni85] M. Snir. Lower bounds on probabilistic linear decision trees. *Theor. Comput. Sci.*, 38: 69–82, 1985.

Chapter 23

The Probabilistic Method II

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "Today I know that everything watches, that nothing goes unseen, and that even wallpaper has a better memory than ours. It isn't God in His heaven that sees all. A kitchen chair, a coat-hanger a half-filled ash tray, or the wood replica of a woman name Niobe, can perfectly well serve as an unforgetting witness to every one of our acts."
>
> Gunter Grass, The tin drum

23.1. Expanding Graphs

In this lecture, we are going to discuss *expanding graphs*.

Definition 23.1.1. An (n, d, α, c) ***OR-concentrator*** is a bipartite multigraph $\mathsf{G}(L, R, E)$, with the independent sets of vertices L and R each of cardinality n, such that

(i) Every vertex in L has degree at most d.
(ii) Any subset S of vertices of L, with $|S| \leq \alpha n$ has at least $c|S|$ neighbors in R.

A good (n, d, α, c) OR-concentrator should have d as small as possible[①], and c as large as possible.

Theorem 23.1.2. *There is an integer n_0, such that for all $n \geq n_0$, there is an $(n, 18, 1/3, 2)$ OR-concentrator.*

Proof: Let every vertex of L choose neighbors by sampling (with replacement) d vertices independently and uniformly from R. We discard multiple parallel edges in the resulting graph.

Let $\mathcal{E}_s$ be the event that a subset of s vertices of L has fewer than cs neighbors in R. Clearly,

$$\mathbb{P}[\mathcal{E}_s] \leq \binom{n}{s}\binom{n}{cs}\left(\frac{cs}{n}\right)^{ds} \leq \left(\frac{ne}{s}\right)^{s}\left(\frac{ne}{cs}\right)^{cs}\left(\frac{cs}{n}\right)^{ds} = \left[\left(\frac{s}{n}\right)^{d-c-1}\exp(1+c)\,c^{d-c}\right]^{s},$$

since $\binom{n}{k} \leq \left(\frac{ne}{k}\right)^{k}$. Setting $\alpha = 1/3$ using $s \leq \alpha n$, and $c = 2$, we have

$$\begin{aligned}\mathbb{P}[\mathcal{E}_s] &\leq \left[\left(\frac{1}{3}\right)^{d-c-1}e^{1+c}c^{d-c}\right]^{s} \leq \left[\left(\frac{1}{3}\right)^{d}3^{1+c}e^{1+c}c^{d-c}\right]^{s} \leq \left[\left(\frac{1}{3}\right)^{d}3^{1+c}e^{1+c}c^{d}\right]^{s} \\ &\leq \left[\left(\frac{c}{3}\right)^{d}(3e)^{1+c}\right]^{s} \leq \left[\left(\frac{2}{3}\right)^{18}(3e)^{1+2}\right]^{s} \leq (0.4)^{s},\end{aligned}$$

[①] Or smaller!

as $c = 2$ and $d = 18$. Thus,

$$\sum_{s \geq 1} \mathbb{P}[\mathcal{E}_s] \leq \sum_{s \geq 1} (0.4)^s < 1.$$

It thus follows that the random graph we generated has the required properties with positive probability. ■

23.1.1. An alternative construction

Theorem 23.1.3. *Consider a bipartite graph over left and right sets L and R, such that $n = |L| = |R|$. Consider a random graph G formed by the union of $d = 18$ random perfect matchings between L and R. Let G be the resulting graph. Then, for $d \geq 18$, the resulting graph is $(n, 18, 1/3, 2)$ OR-concentrator. Furthermore, G has maximum degree d.*

Proof: Let $\mathcal{E}_s$ be the event that a subset of s vertices of L has fewer than cs neighbors in R. For a choice of such a set $S \subseteq L$, and a set T of size cs in R, we have that number of ways to chose a matching such that all the vertices of S has neighbors in T is $cs \cdot (cs-1) \cdots (cs-s+1)$ – indeed, we fix an ordering of the items in S, and assign them their match in T one by one. As such, we have

$$\Xi = \mathbb{P}[\mathcal{E}_s] \leq \binom{n}{s}\binom{n}{cs}\left(\frac{cs(cs-1)\cdots(cs-s+1)}{n(n-1)\cdots(n-s+1)}\right)^d.$$

Using $\frac{cs}{n} \cdot \frac{cs-1}{n-1} \cdot \cdots \cdot \frac{cs-s+1}{n-s+1} \leq \left(\frac{cs}{n}\right)^s$, we have

$$\Xi \leq \left(\frac{ne}{s}\right)^s \left(\frac{ne}{cs}\right)^{cs} \left(\frac{cs}{n}\right)^{ds}.$$

The quantity in the right, in the above inequality, is the same quantity bounded in the proof of Theorem 23.1.2, and the result follows by the same argumentation. ■

23.1.2. An expander

Definition 23.1.4. An (n, d, c)-***expander*** is a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ over n vertices, n, such that

(i) Every vertex in G has degree at most d.

(ii) Any subset S of vertices of V, with $|S| \leq n/3$ has at least $c|S|$ neighbors.

Theorem 23.1.5. *One can construct a $(n, 36, 2)$-expander*

Proof: Let G be a graph with the set of vertices being $[\![n]\!]$. Construction the graph of Theorem 23.1.3, and let G' be this graph. For every edge $v_i u_j$ in G' create an edge ij in G. Clearly, G has the desired properties. ■

23.2. Probability Amplification

Let **Alg** be an algorithm in **RP**, such that given x, **Alg** picks a random number r from the range $\mathbb{Z}_n = \{0, \ldots, n-1\}$, for a suitable choice of a prime n, and computes a binary value $\mathbf{Alg}(x, r)$ with the following properties:

(A) If $x \in L$, then $\mathbf{Alg}(x, r) = 1$ for at least half the possible values of r.

(B) If $x \notin L$, then $\mathbf{Alg}(x, r) = 0$ for all possible choices of r.

Next, we show that using $\lg^2 n$ bits[②] one can achieve $1/n^{\lg n}$ confidence, compared with the naive $1/n$, and the $1/t$ confidence achieved by t (dependent) executions of the algorithm using two-point sampling.

Theorem 23.2.1. *For n large enough, there exists a bipartite graph $\mathsf{G}(V, R, E)$ with $|V| = n$, $|R| = 2^{\lg^2 n}$ such that:*

(i) Every subset of $n/2$ vertices of V has at least $2^{\lg^2 n} - n$ neighbors in R.

(ii) No vertex of R has more than $12 \lg^2 n$ neighbors.

Proof: Each vertex of V chooses $d = 2^{\lg^2 n}(4 \lg^2 n)/n$ neighbors independently in R. We show that the resulting graph violate the required properties with probability less than half.[③]

The probability for a set of $n/2$ vertices on the left to fail to have enough neighbors, is

$$\begin{aligned}\tau &\le \binom{n}{n/2}\binom{2^{\lg^2 n}}{n}\left(1 - \frac{n}{2^{\lg^2 n}}\right)^{dn/2} \le 2^n \left(\frac{2^{\lg^2 n} e}{n}\right)^n \exp\left(-\frac{dn}{2}\frac{n}{2^{\lg^2 n}}\right) \\ &\le 2^n \underbrace{\left(\frac{2^{\lg^2 n} e}{n}\right)^n}_{*} \exp\left(-\frac{2^{\lg^2 n}(4\lg^2 n)/n}{2}\frac{n^2}{2^{\lg^2 n}}\right) \le \exp\Big(n + \underbrace{n \ln \frac{2^{\lg^2 n} e}{n}}_{*} - 2n\lg^2 n\Big),\end{aligned}$$

since $\binom{n}{n/2} \le 2^n$ and $\binom{2^{\lg^2 n}}{2^{\lg^2 n} - n} = \binom{2^{\lg^2 n}}{n}$, and $\binom{x}{y} \le (\frac{xe}{y})^y$[④]. Now, we have

$$\rho = n \ln \frac{2^{\lg^2 n} e}{n} = n\Big(\ln 2^{\lg^2 n} + \ln e - \ln n\Big) \le (\ln 2) n \lg^2 n \le 0.7 n \lg^2 n,$$

for $n \ge 3$. As such, we have $\tau \le \exp\Big(n + (0.7 - 2) n \lg^2 n\Big) \ll 1/4$.

As for the second property, note that the expected number of neighbors of a vertex $v \in R$ is $4 \lg^2 n$. Indeed, the probability of a vertex on R to become adjacent to a random edge is $\rho = 1/|R|$, and this "experiment" is repeated independently dn times. As such, the expected degree of a vertex is $\mu \mathbb{E}[Y] = dn/|R| = 4 \lg^2 n$. The Chernoff bound (Theorem 13.2.1$_{\text{p95}}$) implies that

$$\alpha = \mathbb{P}\Big[Y > 12 \lg^2 n\Big] = \mathbb{P}\Big[Y > (1+2)\mu\Big] < \exp\Big(-\mu 2^2/4\Big) = \exp\Big(-4 \lg^2 n\Big).$$

Since there are $2^{\lg^2 n}$ vertices in R, we have that the probability that any vertex in R has a degree that exceeds $12 \lg^2 n$, is, by the union bound, at most $|R|\,\alpha \le 2^{\lg^2 n} \exp\Big(-4 \lg^2 n\Big) \le \exp\Big(-3 \lg^2 n\Big) \ll 1/4$, concluding our tedious calculations[⑤].

Thus, with constant positive probability, the random graph has the required property, as the union of the two bad events has probability $\ll 1/2$. ■

We assume that given a vertex (of the above graph) we can compute its neighbors, without computing the whole graph.

[②] Everybody knows that $\lg n = \log_2 n$. Everybody knows that the captain lied.

[③] Here, we keep parallel edges if they happen – which is unlikely. The reader can ignore this minor technicality, on her way to ignore this whole write-up.

[④] The reader might want to verify that one can use significantly weaker upper bounds and the result still follows – we are using the tighter bounds here for educational reasons, and because we can.

[⑤] Once again, our verbosity in applying the Chernoff inequality is for educational reasons – usually such calculations would be swept under the rag. No wonder than that everybody is afraid to look under the rag.

So, we are given an input x. Use $\lg^2 n$ bits to pick a vertex $v \in R$. We next identify the neighbors of v in V: $r_1, \ldots, r_k$. We then compute $\mathbf{Alg}(x, r_i)$, for $i = 1, \ldots k$. Note that $k = O\left(\lg^2 n\right)$. If all k calls return 0, then we return that **Alg** is not in the language. Otherwise, we return that x belongs to V.

If x is in the language, then consider the subset $U \subseteq V$, such that running **Alg** on any of the strings of U returns TRUE. We know that $|U| \geq n/2$. The set U is connected to all the vertices of R except for at most $|R| - \left(2^{\lg^2 n} - n\right) = n$ of them. As such, the probability of a failure in this case, is

$$\mathbb{P}\Big[x \in L \text{ but } r_1, r_2, \ldots, r_k \notin U\Big] = \mathbb{P}\Big[v \text{ not connected to } U\Big] \leq \frac{n}{|R|} \leq \frac{n}{2^{\lg^2 n}}.$$

We summarize the result.

Lemma 23.2.2. *Given an algorithm* **Alg** *in* **RP** *that uses* $\lg n$ *random bits, and an access explicit access to the graph of Theorem 23.2.1, one can decide if an input word is in the language of* **Alg** *using* $\lg^2 n$ *bits, and the probability of f failure is at most* $n/2^{\lg^2 n}$.

Let us compare the various results we now have about running an algorithm in **RP** using $\lg^2 n$ bits. We have three options:

(A) Randomly run the algorithm $\lg n$ times independently. The probability of failure is at most $1/2^{\lg n} = 1/n$.
(B) Lemma 23.2.2, which as probability of failure at most $1/2^{\lg n} = 1/n$.
(C) The third option is to use pairwise independent sampling (see Lemma 7.2.13$_{\text{p60}}$). While it is not directly comparable to the above two options, it is clearly inferior, and is thus less useful.

Unfortunately, there is no explicit construction of the expanders used here. However, there are alternative techniques that achieve a similar result.

23.3. Oblivious routing revisited

Theorem 23.3.1. *Consider any randomized oblivious algorithm for permutation routing on the hypercube with* $N = 2^n$ *nodes. If this algorithm uses* k *random bits, then its expected running time is* $\Omega\left(2^{-k}\sqrt{N/n}\right)$.

Corollary 23.3.2. *Any randomized oblivious algorithm for permutation routing on the hypercube with* $N = 2^n$ *nodes must use* $\Omega(n)$ *random bits in order to achieve expected running time* $O(n)$.

Theorem 23.3.3. *For every* n*, there exists a randomized oblivious scheme for permutation routing on a hypercube with* $n = 2^n$ *nodes that uses* $3n$ *random bits and runs in expected time at most* $15n$.

Chapter 24

Dimension Reduction

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

24.1. Introduction to dimension reduction

Given a set P of n points in $\mathbb{R}^d$, we need nd numbers to describe them. In many scenarios, d might be quite large, or even larger than n (in some applications, where the access to the points is given only through dot-product, it is useful to think about the dimension as being unbounded). If we care only about the distances between any pairs of points, then all we need to store are the pairwise distances between the points. This would require roughly n^2 numbers, if we just write down the distance matrix.

But can we do better? (I.e., use less space.) A natural idea is to reduce the dimension of the points. Namely, replace the ith point $\mathsf{p}_i \in P$, by a point $\mathsf{u}_i \in \mathbb{R}^k$, where $k \ll d$ and $k \ll n$. We would like k to be small. If we can do that, then we compress the data from size dn to size kn, which might be a large compression.

Of course, one can do such compression of information without losing some information. In particular, we are willing to let the distances to be a bit off. Formally, we would like to have the property that $(1-\varepsilon)\|\mathsf{p}_i - \mathsf{p}_j\| \leq \|\mathsf{u}_i - \mathsf{u}_j\| \leq (1+\varepsilon)\|\mathsf{p}_i - \mathsf{p}_j\|$, for all i, j, where u_i is the image of $\mathsf{p}_i \in P$ after the dimension reduction.

To this end, we generate a random matrix M of dimensions $d \times k$, where $k = \Theta(\varepsilon^{-2} \log n)$ (the exact details of how to generate this matrix are below, but informally every entry is going to be picked from a normal distribution and scaled appropriately). We then set $\mathsf{u}_i = \mathsf{M}\mathsf{p}_i$, for all $\mathsf{p}_i \in P$.

Before dwelling on the details, we need to better understand the normal distribution.

24.2. Normal distribution

The ***standard normal distribution*** has

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-x^2/2\right) \tag{24.1}$$

as its density function. We denote that X is distributed according to such distribution, using $X \sim \mathbf{N}(0,1)$. It is depicted in Figure 24.1.

Somewhat strangely, it would be convenient to consider two such independent variables X and Y together. Their probability space (X, Y) is the plane, and it defines a two dimensional density function

$$g(x,y) = f(x)f(y) = \frac{1}{2\pi} \exp\left(-(x^2+y^2)/2\right). \tag{24.2}$$

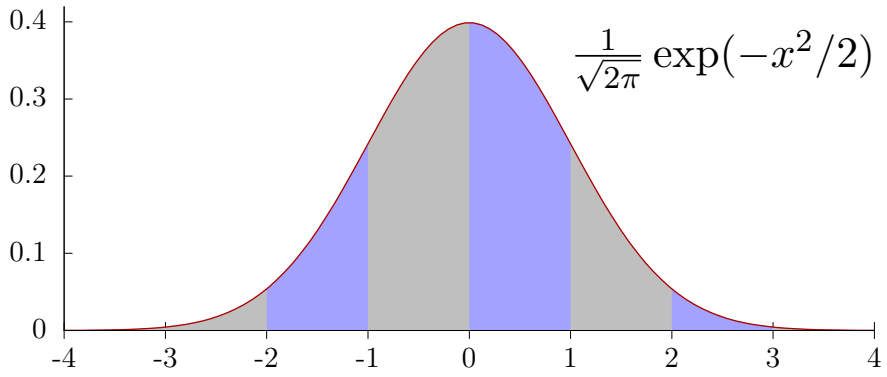


Figure 24.1

The key property of this function is that $g(x,y) = g(x',y') \iff \|(x,y)\|^2 = x^2+y^2 = \|(x',y')\|^2$. Namely, $g(x,y)$ is symmetric around the origin (i.e., all the points in the same distance from the origin have the same density). We next use this property in verifying that $f(\cdot)$ it is indeed a valid density function.

Lemma 24.2.1. *We have* $I = \int_{-\infty}^{\infty} f(x)\,\mathrm{d}x = 1$, *where* $f(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-x^2/2\right)$.

Proof: Observe that

$$I^2 = \left(\int_{x=-\infty}^{\infty} f(x)\ \mathrm{d}x\right)^2 = \left(\int_{x=-\infty}^{\infty} f(x)\ \mathrm{d}x\right)\left(\int_{y=-\infty}^{\infty} f(y)\ \mathrm{d}y\right) = \int_{x=-\infty}^{\infty}\int_{y=-\infty}^{\infty} f(x)f(y)\,dx\,dy$$
$$= \int_{x=-\infty}^{\infty}\int_{y=-\infty}^{\infty} g(x,y)\,\mathrm{d}x\,\mathrm{d}y.$$

Change the variables to $x = r\cos\alpha$, $y = r\sin\alpha$, and observe that the determinant of the Jacobian is

$$J = \det\begin{vmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \alpha} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \alpha} \end{vmatrix} = \det\begin{vmatrix} \cos\alpha & -r\sin\alpha \\ \sin\alpha & r\cos\alpha \end{vmatrix} = r\left(\cos^2\alpha + \sin^2\alpha\right) = r.$$

As such,

$$I^2 = \frac{1}{2\pi}\int_{r=0}^{\infty}\int_{\alpha=0}^{2\pi}\exp\left(-\frac{r^2}{2}\right)|J|\ d\alpha\,dr = \frac{1}{2\pi}\int_{r=0}^{\infty}\int_{\alpha=0}^{2\pi}\exp\left(-\frac{r^2}{2}\right)r\,d\alpha\,dr$$
$$= \int_{r=0}^{\infty}\exp\left(-\frac{r^2}{2}\right)r\,dr = \left[-\exp\left(-\frac{r^2}{2}\right)\right]_{r=0}^{r=\infty} = -\exp(-\infty) - (-\exp(0)) = 1. \qquad \blacksquare$$

Lemma 24.2.2. *For* $X \sim \mathbf{N}(0,1)$, *we have that* $\mathbb{E}[X] = 0$ *and* $\mathbb{V}[X] = 1$.

Proof: The density function of X, see Eq. (24.2) is symmetric around 0, which implies that $\mathbb{E}[X] = 0$. As for the variance, we have

$$\mathbb{V}[X] = \mathbb{E}\left[X^2\right] - (\mathbb{E}[X])^2 = \mathbb{E}\left[X^2\right] = \int_{x=-\infty}^{\infty} x^2\,\mathbb{P}[X=x]\,\mathrm{d}x = \frac{1}{\sqrt{2\pi}}\int_{x=-\infty}^{\infty} x^2\exp(-x^2/2)\,\mathrm{d}x.$$

Observing that

$$x^2\exp(-x^2/2) = \left(-x\exp(-x^2/2)\right)' + \exp(-x^2/2),$$

implies (using integration by guessing) that

$$\mathbb{V}[X] = \frac{1}{\sqrt{2\pi}}\left[-x\exp(-x^2/2)\right]_{x=-\infty}^{\infty} + \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\exp(-x^2/2)\,\mathrm{d}x = 0 + 1 = 1. \qquad \blacksquare$$

24.2.1. The standard multi-dimensional normal distribution

The ***multi-dimensional normal distribution***, denoted by $\mathbf{N}^d$, is the distribution in $\mathbb{R}^d$ that assigns a point $\mathsf{p} = (\mathsf{p}_1, \ldots, \mathsf{p}_d)$ the density $g(\mathsf{p}) = \frac{1}{(2\pi)^{d/2}} \exp\Big(-\frac{1}{2}\sum_{i=1}^{d} \mathsf{p}_i^2\Big)$.

It is easy to verify, using the above, that $\int_{\mathbb{R}^d} g(\mathsf{p})d\mathsf{p} = 1$. Furthermore, we have the following useful but easy properties.[①]

Lemma 24.2.3. *We have the following properties:*

(A) Consider d independent variables $X_1, \ldots, X_d \sim \mathbf{N}(0,1)$, the point $\mathsf{u} = (X_1, \ldots, X_d)$ has the multi-dimensional normal distribution $\mathbf{N}^d$.

(B) The multi-dimensional normal distribution is symmetric. For any two points $\mathsf{p}, \mathsf{u} \in \mathbb{R}^d$ such that $\|\mathsf{p}\| = \|\mathsf{u}\|$, we have that $g(\mathsf{p}) = g(\mathsf{u})$, where $g(\cdot)$ is the density function of the multi-dimensional normal distribution $\mathbf{N}^d$.

(C) The projection of the normal distribution on any direction (i.e., any vector of length 1) is a one-dimensional normal distribution.

Proof: (A) Let $f(\cdot)$ denote the density function of $\mathbf{N}(0,1)$, and observe that the density function of u is $f(X_1)f(X_2)\cdots f(X_d), = \frac{1}{\sqrt{2\pi}}\exp\Big(-X_1^2/2\Big)\cdots\frac{1}{\sqrt{2\pi}}\exp\Big(-X_d^2/2\Big)$, which readily implies the claim.

(B) Readily follows from observing that $g(\mathsf{p}) = \frac{1}{(2\pi)^{d/2}}\exp(-\|\mathsf{p}\|^2/2)$.

(C) Let $\mathsf{p} = (X_1, \ldots, X_d)$, where $X_1, \ldots, X_d \sim \mathbf{N}(0,1)$. Let v be any unit vector in $\mathbb{R}^d$, and observe that by the symmetry of the density function, we can (rigidly) rotate the space around the origin in any way we want, and the measure of sets does not change. In particular rotate space so that v becomes the unit vector $(1, 0, \ldots, 0)$. We have that

$$\mathbb{P}[\langle v, \mathsf{p}\rangle \le \alpha] = \mathbb{P}[\langle (1,0,\ldots,0), \mathsf{p}\rangle \le \alpha] = \mathbb{P}[X_1 \le \alpha],$$

which implies that $\langle v, \mathsf{p}\rangle \sim X_1 \sim \mathbf{N}(0,1)$. ■

The generalized multi-dimensional distribution is a ***Gaussian***. Fortunately, we only need the simpler notion.

24.3. Dimension reduction

24.3.1. The construction

The input is a set $P \subseteq \mathbb{R}^d$ of n points (where d is potentially very large), and let $\varepsilon > 0$ be an approximation parameter. For

$$k = \Big\lceil 24\varepsilon^{-2}\ln n \Big\rceil \tag{24.3}$$

we pick k vectors $u_1, \ldots, u_k$ independently from the d-dimensional normal distribution $\mathbf{N}^d$. Given a point $\mathsf{p} \in \mathbb{R}^d$, its image is

$$h(v) = \frac{1}{\sqrt{k}}\Big(\langle u_1, \mathsf{p}\rangle, \cdots, \langle u_k, \mathsf{p}\rangle\Big).$$

[①] The normal distribution has such useful properties that it seems that the only thing normal about it is its name.

In matrix notation, let

$$\mathsf{M} = \frac{1}{\sqrt{k}} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_k \end{pmatrix}.$$

For every point $\mathsf{p}_i \in P$, we set $\mathsf{u}_i = h(\mathsf{p}_i) = \mathsf{M}\mathsf{p}_i$.

24.3.2. Analysis

24.3.2.1. A single unit vector is preserved

Consider a vector v of length one in $\mathbb{R}^d$. The natural question is what is the value of k needed, so that the length of $h(v)$ is a good approximation to v. Since $\langle u_i, v\rangle \sim \mathbf{N}(0,1)$, by Lemma 24.2.3, this question can boil down to the following: Given k variables $X_1, \ldots, X_k \sim \mathbf{N}(0,1)$, sampled independently, how concentrated is the random variable

$$Y = \|(X_1, \ldots, X_k)\|^2 = \sum_{i=1}^{k} X_i^2.$$

We have that $\mathbb{E}[Y] = k\,\mathbb{E}\left[X_i^2\right] = k\,\mathbb{V}[X_i] = k$, since $X_i \sim \mathbf{N}(0,1)$, for any i. The distribution of Y is known as the ***chi-square distribution with k degrees of freedom***.

Lemma 24.3.1. *Let $\varphi \in (0,1)$, and $\varepsilon \in (0, 1/2)$ be parameters, and let $k \geq \lceil \frac{16}{\varepsilon^2} \ln \frac{2}{\varphi} \rceil$ be an integer. Then, for k independent random variables $X_1, \ldots, X_k \sim \mathbf{N}(0,1)$, we have that $Z = \sum_i X_i^2/k$ is strongly concentrated. Formally, we have that $\mathbb{P}[Z \leq 1+\varepsilon] \geq 1 - \varphi$.*

Proof: Arguing as in the proof of Chernoff's inequality, using $t = \varepsilon/4 < 1/2$, we have

$$\mathbb{P}[Z \geq 1+\varepsilon] \leq \mathbb{P}[\exp(tkZ) \geq \exp(tk(1+\varepsilon))] \leq \frac{\mathbb{E}[\exp(tkZ)]}{\exp(tk(1+\varepsilon))} = \prod_{i=1}^{k} \frac{\mathbb{E}\left[\exp(tX_i^2)\right]}{\exp(t(1+\varepsilon))}.$$

Using the substitution $x = \frac{y}{\sqrt{1-2t}}$ and $\mathrm{d}x = \frac{1}{\sqrt{1-2t}}\,\mathrm{d}y$, we have

$$\begin{aligned}
\mathbb{E}\left[\exp(tX_i^2)\right] &= \int_{x=-\infty}^{\infty} \frac{\exp(tx^2)}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \mathrm{d}x = \frac{1}{\sqrt{2\pi}} \int_{x=-\infty}^{\infty} \exp\left(-(1-2t)\frac{x^2}{2}\right) \mathrm{d}x \\
&= \frac{1}{\sqrt{2\pi}} \int_{y=-\infty}^{\infty} \exp\left(-\frac{1-2t}{2}\left(\frac{y}{\sqrt{1-2t}}\right)^2\right) \frac{1}{\sqrt{1-2t}}\,\mathrm{d}y = \frac{1}{\sqrt{1-2t}} \cdot \frac{1}{\sqrt{2\pi}} \int_{y=-\infty}^{\infty} \exp\left(-\frac{y^2}{2}\right) \mathrm{d}y \\
&= \frac{1}{\sqrt{1-2t}}.
\end{aligned}$$

We have that $\frac{1}{1-z} = \sum_{i=0}^{\infty} z^i$, for $0 \leq z < 1$, and thus

$$\frac{1}{1-\varepsilon/2} = \sum_i \left(\frac{\varepsilon}{2}\right)^i \leq \left[1 + \frac{1}{2}\sum_{i=1}^{\infty}\left(\frac{\varepsilon}{2}\right)^i\right]^2 \leq \exp\left[\frac{1}{2}\sum_{i=1}^{\infty}\left(\frac{\varepsilon}{2}\right)^i\right]^2.$$

Since $t = \varepsilon/4$, we have

$$\mathbb{E}\left[\exp(tX_i^2)\right] = \frac{1}{\sqrt{1-2t}} = \frac{1}{\sqrt{1-\varepsilon/2}} \leq \exp\left(\frac{1}{2}\sum_{i=1}^{\infty}\left(\frac{\varepsilon}{2}\right)^i\right).$$

As such, we have

$$\mathbb{P}[Z \geq 1+\varepsilon] \leq \exp\left(\frac{1}{2}\sum_{i=1}^{\infty}\left(\frac{\varepsilon}{2}\right)^i - \frac{\varepsilon}{4}(1+\varepsilon)\right)^k = \exp\left(-\frac{\varepsilon^2}{8} + \frac{1}{2}\sum_{i=3}^{\infty}\left(\frac{\varepsilon}{2}\right)^i\right)^k \leq \exp\left(-\frac{k\varepsilon^2}{16}\right) \leq \frac{\varphi}{2},$$

since, for $\varepsilon < 1/2$, we have $\frac{1}{2}\sum_{i=3}^{\infty}(\varepsilon/2)^i \leq (\varepsilon/2)^3 \leq \varepsilon^2/16$. The last step in the above inequality follows by substituting in the lower bound on the value of k. ■

The other direction we need follows in a similar fashion. We state the needed result without proof [LM00, Lemma 1] (which also yields better constants):

Lemma 24.3.2. *Let $Y_1, \ldots, Y_k$ be k independent random variables with $Y_i \sim \mathbf{N}(0,1)$. Let $Z = \sum_{i=1}^{k} Y_i^2/k$. For any $x > 0$, we have that*

$$\mathbb{P}\left[Z \leq 1 - 2\sqrt{x/k}\right] \leq \exp(-x) \qquad \text{and} \qquad \mathbb{P}\left[Z \geq 1 + 2\sqrt{x/k} + 2x/k\right] \leq \exp(-x).$$

For our purposes, we require that $\exp(-x) \leq \varphi/2$, which implies $x = \ln(2/\varphi)$. We further require that $2\sqrt{x/k} \leq \varepsilon$ and $2\sqrt{x/k} + 2x/k \leq \varepsilon$, which hold for $k = 8\varepsilon^{-2}\ln\frac{2}{\varphi}$, for $\varepsilon \leq 1$. We thus get the following result.

Corollary 24.3.3. *Let $\varphi \in (0,1)$, and $\varepsilon \in (0, 1/2)$ be parameters, and let $k \geq \lceil \frac{8}{\varepsilon^2}\ln\frac{2}{\varphi}\rceil$ be an integer. Then, for k independent random variables $X_1, \ldots, X_k \sim \mathbf{N}(0,1)$, we have for $Z = \sum_i X_i^2/k$ that that $\mathbb{P}[1-\varepsilon \leq Z \leq 1+\varepsilon] \geq 1 - \varphi$.*

Remark 24.3.4. The result of Corollary 24.3.3 is surprising. It says that if we pick a point according to the k-dimensional normal distribution, then its distance to the origin is strongly concentrated around $\sqrt{k}$. Namely, the normal distribution "converges" to a sphere, as the dimension increases. The mind boggles.

Lemma 24.3.5. *Let v be a unit vector in $\mathbb{R}^d$, then*

$$\mathbb{P}\left[1-\varepsilon \leq \|\mathsf{M}v\| \leq 1+\varepsilon\right] \geq 1 - \frac{1}{n^2}.$$

Proof: Observe that if for a number x, if $1-\varepsilon \leq x^2 \leq 1+\varepsilon$, then $1-\varepsilon \leq x \leq 1+\varepsilon$. As such, the claim holds if $1-\varepsilon \leq \|\mathsf{M}v\|^2 \leq 1+\varepsilon$. By Corollary 24.3.3, setting $\varphi = 1/n^2$, we need

$$k \geq 8\varepsilon^{-2}\ln(2/\varphi) = 8\varepsilon^{-2}\ln(2n^2) = 24\varepsilon^{-2}\ln n,$$

which holds for the value picked for k in Eq. (24.3). ■

24.3.3. All pairwise distances are preserved

Lemma 24.3.6. *With probability at least half, for all points $\mathsf{p}, \mathsf{p}' \in P$, we have that*

$$(1-\varepsilon)\|\mathsf{p}-\mathsf{p}'\| \leq \|\mathsf{M}\mathsf{p} - \mathsf{M}\mathsf{u}\| \leq (1+\varepsilon)\|\mathsf{p}-\mathsf{u}\|.$$

Proof: The key observation is that M is a linear operator. As such, let $v = (\mathsf{p}-\mathsf{p}')/\|\mathsf{p}-\mathsf{p}'\|$ be a unit vector, and observe that

$$\begin{aligned}
(1-\varepsilon)\|\mathsf{p}-\mathsf{p}'\| \leq \|\mathsf{M}\mathsf{p} - \mathsf{M}\mathsf{p}'\| = \|\mathsf{M}(\mathsf{p}-\mathsf{p}')\| &\leq (1+\varepsilon)\|\mathsf{p}-\mathsf{p}'\| \\
\iff \qquad 1-\varepsilon \leq \left\|\mathsf{M}\frac{\mathsf{p}-\mathsf{p}'}{\|\mathsf{p}-\mathsf{p}'\|}\right\| &\leq 1+\varepsilon \\
\iff \qquad (1-\varepsilon)\|v\| \leq \|\mathsf{M}v\| &\leq (1+\varepsilon)\|v\|.
\end{aligned}$$

The probability the later condition does not hold is at most $1/n^2$, by Lemma 24.3.5. As such, for all possible pairs of points, the probability of failure is $\binom{n}{2}\cdot\frac{1}{n^2} \leq 1/2$, as claimed. ■

We thus got the famous JL-Lemma.

Theorem 24.3.7 (The Johnson-Lindenstrauss Lemma). *Given a set P of n points in $\mathbb{R}^d$, and a parameter ε, one can reduce the dimension of P to $k = O(\varepsilon^{-2} \log n)$ dimensions, such that all pairwise distances are $1 \pm \varepsilon$ preserved.*

24.4. Even more on the normal distribution

The following is not used anywhere in the above, and is provided as additional information about the normal distribution.

Lemma 24.4.1. *Let $X \sim \mathbf{N}(0,1)$, and let $\sigma > 0$ and μ be two real numbers. The random variable $Y = \sigma X + \mu$ has the density function*

$$f_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{24.4}$$

The variable Y has the **normal distribution** *with variance σ^2, and expectation μ, denoted by $Y \sim \mathbf{N}\left(\mu, \sigma^2\right)$.*

Proof: We have $\mathbb{P}[Y \le \alpha] = \mathbb{P}[\sigma X + \mu \le \alpha] = \mathbb{P}\left[X \le \frac{\alpha-\mu}{\sigma}\right] = \int_{y=-\infty}^{(\alpha-\mu)/\sigma} f(y)\,\mathrm{d}y$, where $f(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-x^2/2\right)$. Substituting $y = (x-\mu)/\sigma$, and observing that $\mathrm{d}y/\,\mathrm{d}x = 1/\sigma$, we have

$$\mathbb{P}[Y \le \alpha] = \int_{x=-\infty}^{\alpha} f\left(\frac{x-\mu}{\sigma}\right)\frac{1}{\sigma}\,\mathrm{d}x = \frac{1}{\sqrt{2\pi}\sigma}\int_{x=-\infty}^{\alpha} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\mathrm{d}x,$$

as claimed.

As for the second part, observe that $\mathbb{E}[Y] = \mathbb{E}[\sigma X + \mu] = \sigma\,\mathbb{E}[X] + \mu = \mu$ and $\mathbb{V}[Y] = \mathbb{V}[\sigma X + \mu] = \mathbb{V}[\sigma X] = \sigma^2\,\mathbb{V}[X] = \sigma^2$. ■

Lemma 24.4.2. *Consider two independent variables $X \sim \mathbf{N}(0,1)$ and $Y \sim \mathbf{N}(0,1)$. For $\alpha, \beta > 0$, we have $Z = \alpha X + \beta Y \sim \mathbf{N}\left(0, \sigma^2\right)$, where $\sigma = \sqrt{\alpha^2 + \beta^2}$.*

Proof: Consider the region in the plane $H^- = \left\{(x,y) \in \mathbb{R}^2 \;\middle|\; \alpha x + \beta y \le z\right\}$ – this is a halfspace bounded by the line $\ell \equiv \alpha x + \beta y = z$. This line is orthogonal to the vector $(-\beta, \alpha)$. We have that $\ell \equiv \frac{\alpha}{\sigma}x + \frac{\beta}{\sigma}y = \frac{z}{\sigma}$. Observe that $\left\|\left(\frac{\alpha}{\sigma}, \frac{\beta}{\sigma}\right)\right\| = 1$, which implies that the distance of ℓ from the origin is $d = z/\sigma$.

Now, we have

$$\mathbb{P}[Z \le z] = \mathbb{P}[\alpha X + \beta Y \le z] = \mathbb{P}[H^-] = \int_{p=(x,y)\in H^-} g(x,y)\,\mathrm{d}p,$$

see Eq. (24.2). Since, the two dimensional density function g is symmetric around the origin. any halfspace containing the origin, which its boundary is in distance d from the origin, has the same probability. In particular, consider the halfspace $T = \left\{(x,y) \in \mathbb{R}^2 \;\middle|\; x \le d\right\}$. We have that

$$\begin{aligned}\mathbb{P}[Z \le z] = \mathbb{P}[H^-] = \mathbb{P}[T] = \mathbb{P}[X \le d] &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{d} \exp\left(-\frac{x^2}{2}\right)\mathrm{d}x = \frac{1}{\sqrt{2\pi}}\int_{y=-\infty}^{z} \exp\left(-\frac{y^2}{2\sigma^2}\right)\frac{\mathrm{d}x}{\mathrm{d}y}\,\mathrm{d}y,\\ &= \frac{1}{\sqrt{2\pi}\sigma}\int_{y=-\infty}^{z} \exp\left(-\frac{y^2}{2\sigma^2}\right)\mathrm{d}y,\end{aligned}$$

by change of variables $x = y/\sigma$, and observing that $\mathrm{d}x/\,\mathrm{d}y = 1/\sigma$. By Eq. (24.4), the above integral is the probability of a variable distributed $\mathbf{N}\left(0, \sigma^2\right)$ to be smaller than z, establishing the claim. ■

Lemma 24.4.3. *Consider two independent variables* $X \sim \mathbf{N}\left(\mu_1, \sigma_1^2\right)$ *and* $Y \sim \mathbf{N}\left(\mu_2, \sigma_2^2\right)$. *We have* $Z = X + Y \sim \mathbf{N}\left(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2\right)$,

Proof: Let $\widehat{X} \sim \mathbf{N}(0,1)$ and $\widehat{Y} \sim \mathbf{N}(0,1)$, and observe that we can write $X = \sigma_1 \widehat{X} + \mu_1$ and $Y = \sigma_2 \widehat{Y} + \mu_2$. As such, we have

$$Z = X + Y = \sigma_1 \widehat{X} + \sigma_2 \widehat{Y} + \mu_1 + \mu_2.$$

The variable $W = \sigma_1 \widehat{X} + \sigma_2 \widehat{Y} \sim \mathbf{N}\left(0, \sigma_1^2 + \sigma_2^2\right)$, by Lemma 24.4.2. Adding $\mu_1 + \mu_2$ to W, just shifts its expectation, implying the claim. ■

24.5. Bibliographical notes

The original result is due to Johnson and Lindenstrauss [JL84]. By now there are many proofs of this lemma. Our proof follows class notes of Anupam Gupta, which in turn follows Indyk and Motwani [IM98],

References

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. *Proc. 30th Annu. ACM Sympos. Theory Comput.* (STOC), 604–613, 1998.

[JL84] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mapping into hilbert space. *Contemporary Mathematics*, 26: 189–206, 1984.

[LM00] B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. *Ann. Statist.*, 28(5): 1302–1338, 2000.

Chapter 25

Streaming and the Multipass Model

I don't know why it should be, I am sure; but the sight of another man asleep in bed when I am up, maddens me. It seems to me so shocking to see the precious hours of a man's life - the priceless moments that will never come back to him again - being wasted in mere brutish sleep.

Jerome K. Jerome, Three men in a boat

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

25.1. The secretary problem

Assume that we are seeing n applications: $\alpha_1, \ldots, \alpha_n$, where the quality of each one of them is an independent random event. We would like to hire the best one, but we have to make an immediate decision – we see candidate α_i, we either hire them (and then we are stuck with them), or we let them go and see the next candidate. We win the game if we hire the best candidate out of the n candidate.

The question is what is the natural strategy to win the game? Let $P(r)$ be the probability that we win, when the strategy is to see the first $r-1$ candidates, and then hire the first candidate we see in $\alpha_r, \ldots, \alpha_n$ that is better than all the candidates seen in $\alpha_1, \ldots, \alpha_{r-1}$. We have that

$$\begin{aligned}
P(r) &= \sum_{i=1}^{n} \mathbb{P}[\text{applicant } i \text{ is selected} \cap \text{applicant } i \text{ is the best}] \\
&= \sum_{i=1}^{n} \mathbb{P}[\text{applicant } i \text{ is selected} \,\big|\, \text{applicant } i \text{ is the best}] \cdot \mathbb{P}[\text{applicant } i \text{ is the best}] \\
&= \left[\sum_{i=1}^{r-1} 0 + \sum_{i=r}^{n} P\left(\begin{array}{l} \text{the best of the first } i-1 \text{ applicants} \\ \text{is in the first } r-1 \text{ applicants} \end{array} \,\middle|\, \text{applicant } i \text{ is the best} \right) \right] \cdot \frac{1}{n} \\
&= \left[\sum_{i=r}^{n} \frac{r-1}{i-1} \right] \cdot \frac{1}{n} \\
&= \frac{r-1}{n} \sum_{i=r}^{n} \frac{1}{i-1}.
\end{aligned}$$

Observe that

$$\sum_{i=r}^{n} \frac{1}{i-1} \leq \int_{x=r-2}^{n-1} \frac{1}{x}\,\mathrm{d}x = \ln(n-2) - \ln(r-2) \approx \ln \frac{n}{r}.$$

For $r = n/e$, we have that $P(r) \approx \frac{r}{n} \ln \frac{n}{r} = \frac{1}{e} \ln e = 1/e$.

25.2. Reservoir sampling: Fishing a sample from a stream

Imagine that you are given a stream of elements $s_1, s_2, \ldots$, and you need to sample k numbers from this stream (say, without repetition) – assume that you do not know the length of the stream in advance, and furthermore, you have only $O(k)$ space available for you. How to do that efficiently?

There are two natural schemes:

(A) Whenever an element arrives, generate a random number for it in the range $[0, 1]$. Maintain a heap with the k elements with the lowest priority. Implemented naively this requires $O(\log k)$ comparisons after each insertion, but it is not difficult to improve this to $O(1)$ comparisons in the amortized sense per insertion. Clearly, the resulting set is the desired random sample

(B) Let S_t be the random sample maintained in the tth iteration. When the ith element arrives, the algorithm flip a coin that is heads with probability $\min(1, k/i)$. If the coin is heads then it inserts s_i to S_{i-1} to get S_i. If S_{i-1} already have k elements, then first randomly delete one of the elements.

Theorem 25.2.1. *Given a stream of elements, one can uniformly sample k elements (without repetition), from the stream using $O(k)$ space, where $O(1)$ time is spent for handling each incoming element.*

Proof: We implement the scheme (B) above. We only need to argue that this is a uniform random sample. The claim trivially hold for $i = k$. So assume the claim holds for $i < t$, and we need to prove that the set after getting tth element is still a uniform random sample.

So, consider a specific set $K \subseteq \{s_1, \ldots, s_t\}$ of k elements. The probability of K to be a random sample of size k from a set of t elements is $1/\binom{t}{k}$. We need to argue that this probability remains the same for this scheme.

So, if $s_t \notin K$, then we have

$$\mathbb{P}[K = S_t] = \mathbb{P}[K = S_{t-1} \text{ and } s_t \text{ was not inserted}] = \frac{1}{\binom{t-1}{k}}\left(1 - \frac{k}{t}\right) = \frac{k!(t-1-k)!(t-k)}{(t-1)!t} = \frac{1}{\binom{t}{k}}.$$

If $s_t \in K$, then

$$\mathbb{P}[K = S_t] = \mathbb{P}\left[\begin{array}{c} K \setminus \{s_t\} \subseteq S_{t-1}, \\ s_t \text{ was inserted} \\ \text{and } S_{t-1} \setminus K \text{ thrown out of } S_{t-1} \end{array}\right] = \frac{t-1-(k-1)}{\binom{t-1}{k}} \frac{k}{t} \frac{1}{k} = \frac{(t-k)k!(t-1-k)!}{(t-1)!t} = \frac{1}{\binom{t}{k}},$$

as desired. Indeed, there are $t - 1 - (k-1)$ subsets of size k of $\{s_1, \ldots, s_{t-1}\}$ that contains $K \setminus \{s_t\}$ – since we fix $k - 1$ of the $t - 1$ elements. ■

25.3. Sampling and median selection revisited

Let $B[1, \ldots, n]$ be a set of n numbers. We would like to estimate the median, without computing it outright. A natural idea, would be to pick k elements $e_1, \ldots, e_k$ randomly from B, and return their median as the guess for the median of B.

In the following, let $B_{\langle t \rangle}$ be the tth smallest number in the array B.

Observation 25.3.1. *For any $\varepsilon \in (0,1)$, we have that $\frac{1}{1-\varepsilon} \geq 1+\varepsilon$.*

Lemma 25.3.2. *Let $\varepsilon \in (0, 1/2)$ be a fixed parameter, and let B be a set of n numbers. Let Z be the median of the random sample (with replacement) of B of size k. We have that*

$$\mathbb{P}\Big[B_{\langle \frac{1-\varepsilon}{2} n\rangle} \leq Z \leq B_{\langle \frac{1+\varepsilon}{2} n\rangle}\Big] \geq 1-\delta, \qquad \text{where} \qquad k \geq \left\lceil \frac{12}{\varepsilon^2} \ln \frac{2}{\delta} \right\rceil.$$

Namely, with probability at least $1-\delta$, the returned value Z is $(\varepsilon/2)n$ positions away from the true median.

Proof: Let $L = B_{\langle (1-\varepsilon)n/2\rangle}$, and let e_i be the ith sample number, for $i = 1, \ldots, k$. Let $X_i = 1$ if and only if $e_i \leq L$. We have that

$$\mathbb{P}[X_i = 1] = \frac{(1-\varepsilon)n/2}{n} = \frac{1-\varepsilon}{2}.$$

As such, setting $Y = \sum_{i=1}^{k} X_i$, we have

$$\mu = \mathbb{E}[Y] = \frac{1-\varepsilon}{2} k \geq \frac{k}{4} \geq \frac{3}{\varepsilon^2} \ln \frac{2}{\delta}.$$

One case of failure of the algorithm is if $Y \geq k/2$. Since $\frac{1}{1-\varepsilon} \geq 1+\varepsilon$, we have that

$$\mathbb{P}[Y \geq k/2] = \mathbb{P}\left[Y \geq \frac{1/2}{(1-\varepsilon)/2} \cdot \frac{1-\varepsilon}{2} k\right] \leq \mathbb{P}[Y \geq (1+\varepsilon)\mu] \leq \exp\left(-\frac{\varepsilon^2 \mu}{3}\right) \leq \exp\left(-\frac{\varepsilon^2}{3} \cdot \frac{3}{\varepsilon^2} \ln \frac{2}{\delta}\right) \leq \frac{\delta}{2}.$$

by Chernoff's inequality (see Lemma 13.2.5).

This implies that $\mathbb{P}[B_{\langle (1-\varepsilon)n/2\rangle} > Z] \leq \delta/2$. The claim now follows by realizing that by symmetry (i.e., revering the order), we have that

$$\mathbb{P}[Z > B_{\langle (1+\varepsilon)n/2\rangle}] \leq \delta/2,$$

and putting these two inequalities together. ■

The above already implies that we can get a good estimate for the median. We need something somewhat stronger – we state it without proof since it follows by similarly mucking around with Chernoff's inequality.

Lemma 25.3.3. *Let $\varepsilon \in (0, 1/2)$, B an array of n elements, and let $S = \{e_1, \ldots, e_k\}$ be a set of k samples picked uniformly and randomly from B. Then, for some absolute constant c, and an integer k, such that $k \geq \left\lceil \frac{c}{\varepsilon^2} \ln \frac{1}{\delta} \right\rceil$, we have that*

$$\mathbb{P}[S_{\langle k_- \rangle} \leq B_{\langle n/2\rangle} \leq S_{\langle k^+ \rangle}] \geq 1-\delta.$$

for $k_- = \lfloor (1-\varepsilon)k/2 \rfloor$, and $k^+ = \lfloor (1+\varepsilon)k/2 \rfloor$.

One can prove even a stronger statement:

$$\mathbb{P}[B_{\langle (1-2\varepsilon)n/2\rangle} \leq S_{\langle (1-\varepsilon)k/2\rangle} \leq B_{\langle n/2\rangle} \leq S_{\langle (1+\varepsilon)k/2\rangle} \leq B_{\langle (1+2\varepsilon)n/2\rangle}] \geq 1-\delta$$

(the constant c would have to be slightly bigger).

25.3.1. A median selection with few comparisons

The above suggests a natural algorithm for computing the median (i.e., the element of rank $n/2$ in B). Pick a random sample S of $k = O(n^{2/3} \log n)$ elements. Next, sort S, and pick the elements L and R of ranks $(1-\varepsilon)k$ and $(1+\varepsilon)k$ in S, respectively. Next, scan the elements, and compare them to L and R, and keep only the elements that are between. In the end of this process, we have computed:

(A) α: The rank of the number L in the set B.

(B) $T = \{x \in B \mid L \le x \le H\}$.

Compute, by brute force (i.e., sorting) the element of rank $n/2 - \alpha$ in T. Return it as the desired median. If $n/2 - \alpha$ is negative, then the algorithm failed, and it tries again.

Lemma 25.3.4. *The above algorithm performs $2n + O(n^{2/3} \log n)$ comparisons, and reports the median. This holds with high probability.*

Proof: Set $\varepsilon = 1/n^{1/3}$, and $\delta = 1/n^{O(1)}$, and observe that Lemma 25.3.3 implies that with probability $\ge 1 - 1/\delta$, we have that the desired median is between L and H. In addition, Lemma 25.3.3 also implies that $|T| \le 4\varepsilon n \le 4n^{2/3}$, which readily implies the correctness of the algorithm.

As for the bound on the number of comparisons, we have, with high probability, that the number of comparisons is

$$O(|S| \log |S| + |T| \log |T|) + 2n = O\Big(\sqrt{n} \log^2 n + n^{2/3} \log n\Big) + 2n,$$

since deciding if an element is between L and H requires two comparisons. ■

Lemma 25.3.5. *The above algorithm can be modified to perform $(3/2)n + O(n^{2/3} \log n)$ comparisons, and reports the median correctly. This holds with high probability.*

Proof: The trick is to randomly compare each element either first to L or first to H with equal probability. For elements that are either smaller than L or bigger than H, this requires $(3/2)n$ comparisons in expectation. Thus improving the bound from $2n$ to $(3/2)n$. ■

Lemma 25.3.6. *Consider a stream B of n numbers, and assume we can make two passes over the data. Then, one can compute exactly the median of B using:*

(I) $O(n^{2/3})$ *space.*

(II) $1.5n + O(n^{2/3} \log n)$ *comparisons.*

The algorithm reports the median correctly, and it succeeds with high probability.

Proof: Implement the above algorithm, using the random sampling from Theorem 25.2.1. ■

Remark 25.3.7. Interestingly, one can do better if one is more careful. The basic idea is to do thinning – given two sorted sequence of sizes s, consider merging the sets, and then picking all the even rank elements into a new sequence. Clearly, the element of rank i in the output sequence, has rank $2i$ in the union of the two original sequences. A sequence that is the result of i such rounds of thinning is of ***level*** i. We maintain $O(\log n)$ such sequences as we read the stream. At any time, we have two buffers of size s, that we fill up from the stream. Whenever the two buffers fill up completely, we perform the thinning operation on them, creating a sequence of level 1.

If during this process we have two sequences of the same level, we merge them and perform thinning on them. As such, we maintain $O(\log n)$ buffers sequences each of size s. Assume that our stream has size n, and n is a power for 2. Then in the end of process, we would have only a single sequence of level $h = \log_2(n/s)$.

By induction, it is easy to prove that an element of rank r in this sequence, has rank between $2^h(r-1)$ and $2^h r$ in the original stream.

Thus, setting $s = \sqrt{n}$, we get that after a single pass, using $O(\sqrt{n}\log n)$ space, we have a sorted sequence, where the rank of the elements is roughly $\sqrt{n}$ approximation to the true rank. We pick the two consecutive elements (or more carefully, the predecessor, and successor), and filter the stream again, keeping only the elements in between these two elements. It is to show that $O(\sqrt{n})$ would be kept, and we can extract the median using $O(\sqrt{n}\log n)$ time.

We thus got that one can compute the median in two passes using $O(\sqrt{n}\log n)$ space. It is not hard to extend this algorithm to α-passes, where the space required becomes $O(n^{1/\alpha}\log n)$.

This elegant algorithm goes back to 1980, and it is by Munro and Paterson [MP80].

25.4. Big data and the streaming model

Here, we are interested in doing some computational tasks when the amount of data we have to handle is quite large (think terabytes or larger). The main challenge in many of these cases is that even reading the data once is expensive. Running times of $O(n\log n)$ might not be acceptable. Furthermore, in many cases, we can *not* load all the data into memory.

In the ***streaming*** model, one reads the data as it comes in, but one can not afford to keep all the data. A natural example would be a internet router, which has gazillion of packets going through it every minute. We might still be interested in natural questions about these packets, but we want to do this without storing all the packets.

25.5. Heavy hitters

The problem. Imagine a stream $s_1, \ldots$, where elements might repeat, and we would like to maintain a list of elements that appear at least εn times, where $\varepsilon \in (0,1)$ is some parameter. The purpose here is to do this using as little space as possible.

25.5.1. A randomized algorithm

Am easy randomized algorithm, would maintain a random sample of size $m = \lceil (1/\varepsilon)\ln(1/\varphi)\rceil$, using reservoir sampling. The probability the sample fails to contain a heavy hitter after t insertions is

$$(1-\varepsilon)^m \le \exp(-\varepsilon m) \le \exp\left(-\varepsilon\left\lceil \frac{1}{\varepsilon}\ln\frac{1}{\varphi}\right\rceil\right) \le \exp\left(-\ln\frac{1}{\varphi}\right) = \exp(\varphi) = \varphi.$$

25.5.2. A deterministic algorithm

Disclaimer: The following is a deterministic algorithm, but it is too elegant to hold this against it, and we will present it anyway.

The algorithm. To this end, let

$$k = \lceil 1/\varepsilon \rceil .$$

At each point in time, we maintain a set S of k elements, with a counter for each element. Let S_t be the version of S after t were inserted. When s_{t+1} arrives, we increase its counter if it is already in S_t. If $|S_t| < k$, then we

just insert s_{t+1} to the set, and set its counter to 1. Otherwise, $|S_t| = k$ and $s_{t+1} \notin S_t$. We then decrease all the k counters of elements in S_t by 1. If a counter of an element in S_{t+1} is zero, then we delete it from the set.

Correctness.

Lemma 25.5.1. *The above algorithm, after the insertion of t elements, the set S_{t+1} would contain all the elements in the stream that appears at least εt times.*

Proof: Conceptually, imagine that the algorithm keeps counters for all the distinct elements seen in the stream. Whenever a decrease of the counters happens – the algorithm decrease not k counters – but $k+1$ counters – the additional counter being a counter of the new element, which has value one, and goes down to zero. Clearly, the number of distinct remaining elements in any point in time is at most k – that is, the number of counters that have a non-zero value. Consider an element e that appears $u \geq \varepsilon t$ times in the stream. The counter for e is going to be increased u times, and decreased at most α time, where $\alpha(k+1) \leq t$. We have that the counter for u in the end of the stream must have value at least

$$u - \frac{t}{k+1} \geq \varepsilon t - \frac{t}{k+1} = \varepsilon t - \frac{t}{\lceil 1/\varepsilon \rceil + 1} \geq t \frac{\lceil 1/\varepsilon \rceil \varepsilon + \varepsilon - 1}{\lceil 1/\varepsilon \rceil + 1} \geq t \frac{\varepsilon}{\lceil 1/\varepsilon \rceil + 1} > 0.$$

This implies that the counter of u is strictly larger than 0, which implies that u appears in S_{t+1}. ■

References

[MP80] J. I. Munro and M. Paterson. Selection and sorting with limited storage. *Theo. Comp. Sci.*, 12: 315–323, 1980.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

Chapter 26

Frequency Estimation over a Stream

> "See? Genuine-sounding indignation. I programmed that myself. It's the first thing you need in a university environment: the ability to take offense at any slight, real or imagined."
>
> Robert Sawyer, Factoring Humanity

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

26.1. The art of estimation

26.1.1. The problem

Assume we would like to estimate well some quantity $\rho > 0$ - specifically, for a fixed parameter $\varepsilon \in (0, 1)$, we would like to compute a quantity ρ' such that $\rho' \in [(1 - \varepsilon)\rho, (1 + \varepsilon)\rho]$ with good probability. To this end, assume we have access to a distribution $\mathcal{D}$, such that if we sample X according to this distribution (i.e., $X \sim \mathcal{D}$), we have that $\mathbb{E}[X] = \rho$. We can use X to estimate our desired quantity, but this might not provide the desired estimation.

Example: Estimating p for a coin. Assume we have a coin that is head with probability p. A natural way to estimate p is to flip the coin once and return 1 if it is head, and zero otherwise. Let X be the result of the coin flip, and observe that $\mathbb{E}[X] = p$. But this is not very useful estimator.

26.1.2. Averaging estimator: Success with constant probability

26.1.2.1. The challenge

The basic problem is that $X \sim \mathcal{D}$ might be much bigger than ρ. Or more specifically, its variance might be huge, where $\mathcal{D}$ is a distribution we have access to. Let

$$\rho = \mathbb{E}[\mathcal{D}] \qquad \text{and} \qquad \nu = \mathbb{V}[\mathcal{D}]\,.$$

We would to generate a variable Z, such that

$$\mathbb{E}[Z] = \rho \qquad \text{and} \qquad \mathbb{V}[Z] \leq (\varepsilon^2/4)\rho^2. \tag{26.1}$$

This would imply by Chebychev's inequality that

$$\mathbb{P}[|Z-\rho| \geq \varepsilon\rho] = \mathbb{P}\Big[|Z-\mathbb{E}[Z]| \geq 2\sqrt{(\varepsilon^2/4)\rho^2}\Big] \leq \mathbb{P}\Big[|Z-\mathbb{E}[Z]| \geq 2\sqrt{\mathbb{V}[Z]}\Big] \leq \frac{1}{4}.$$

26.1.2.2. Taming of the variance

The basic idea is to take $\alpha = \left\lceil \frac{\nu}{\rho} \right\rceil$ independent variables $X_1, \ldots, X_\alpha \sim \mathcal{D}$, and let $Y = \sum_i X_i/\alpha$. We have by linearity of expectation that

$$\mathbb{E}[Y] = \sum_i \mathbb{E}[X_i]/\alpha = \mathbb{E}[X] = \rho.$$

Using the independence of $X_1, \ldots, X_\alpha$, we have

$$\mathbb{V}[Y] = \mathbb{V}\left[\sum_i X_i/\alpha\right] = \frac{1}{\alpha^2}\mathbb{V}\left[\sum_i X_i\right] = \frac{1}{\alpha^2}\sum_i \mathbb{V}[X_i] = \frac{1}{\alpha^2}\alpha\nu = \frac{\nu}{\alpha}.$$

Guided by Eq. (26.1), we want this quantity to be smaller than $\leq (\varepsilon^2/4)\rho^2$. Thus,

$$\frac{\nu}{\alpha} \leq (\varepsilon^2/4)\rho^2 \qquad \Longleftarrow \qquad \alpha \geq \left\lceil \frac{4}{\varepsilon^2}\cdot\frac{\nu}{\rho^2} \right\rceil = \left\lceil \frac{4\,\mathbb{V}[X]}{\varepsilon^2\,\mathbb{E}[X]^2} \right\rceil.$$

We thus summarize the result.

Lemma 26.1.1. *Let $\mathcal{D}$ be a non-negative distribution with $\rho = \mathbb{E}[\mathcal{D}]$ and $\nu = \mathbb{V}[\mathcal{D}]$, and let $\varepsilon \in (0,1)$ be a parameter. For $\alpha \geq \left\lceil \frac{4\,\mathbb{V}[\mathcal{D}]}{\varepsilon^2(\mathbb{E}[\mathcal{D}])^2} \right\rceil$, consider sampling variables $X_1, \ldots, X_\alpha \sim \mathcal{D}$, and let $Z = \sum_{i=1}^{\alpha} X_i/\alpha$. Then Z is a "good" estimator for ρ. Formally, we have*

$$\mathbb{P}\Big[(1-\varepsilon)\rho \leq Z \leq (1+\varepsilon)\rho\Big] \geq \frac{3}{4}.$$

26.1.3. Median estimator: Success with high probability

We would like to get a better estimator, where the probability of success is high probability. Formally, we would have parameter φ, and we would like the estimator to succeed with probability $\geq 1-\varphi$. A natural approach is to try and use Chernoff to bound the probability of failure for the averaging estimator. This would work in some cases, but is limited to the case when Z lies in a small bounded range. This would not work in general if sampling from $\mathcal{D}$ might return a huge value with tiny probability. Instead, we are going to boost the averaging estimator. Assume, we generating

$$\beta = O\Big(\log \frac{1}{\varphi}\Big)$$

instances of the averaging estimators: $Z_1, \ldots, Z_\beta$ of Lemma 26.1.1. The ***median estimator*** returns the median value of the Zs as the desired estimate.

Analysis. Let $\mathcal{E}_i$ be the event that $Z_i \in [(1-\varepsilon)\rho, (1+\varepsilon)\rho]$. Let G_i be an indicator variable for $\mathcal{E}_i$. By Lemma 26.1.1, $\mathbb{P}[\mathcal{E}_i] = \mathbb{P}[G_i = 1] \geq 3/4$. The median estimator fails if $\sum_{i=1}^{\beta} G_i < \beta/2$. Using Chernoff inequality, we get that this happens with probability $\leq \varphi$. We thus get the following.

Theorem 26.1.2. *Let $\mathcal{D}$ be a non-negative distribution with $\mu = \mathbb{E}[\mathcal{D}]$ and $\nu = \mathbb{V}[\mathcal{D}]$, and let $\varepsilon, \varphi \in (0,1)$ be parameters. For some absolute constant $c > 0$, let $M \geq 24\left\lceil \frac{4\nu}{\varepsilon^2\mu^2} \right\rceil \ln \frac{1}{\varphi}$, and consider sampling variables $X_1, \ldots, X_M \sim \mathcal{D}$. One can compute, in, $O(M)$ time, a quantity Z from the sampled variables, such that*

$$\mathbb{P}\Big[(1-\varepsilon)\mu \leq Z \leq (1+\varepsilon)\mu\Big] \geq 1 - \varphi.$$

Proof: Let $m = \lceil 4\nu/(\varepsilon^2\mu^2) \rceil$ and $M = \left\lceil 24 \ln \frac{1}{\varphi} \right\rceil$. Build M averaging estimators, each one using m samples. That is let Z_i be the average of m samples $s_{i,1}, \ldots, s_{i,m}$ from $\mathcal{D}$, for $i = 1, \ldots, M$. Formally,

$$Z_i = \frac{1}{m} \sum_{j=1}^{m} s_{i,j} \qquad \text{for} \qquad i = 1, \ldots, M.$$

The estimate returned is the value median$(Z_1, \ldots, Z_M)$.

By Lemma 26.1.1 each one of the averaging estimator is in the "good" range with probability $\geq 3/4$. As such, let X_i, for $i = 1, \ldots, M$, be an indicator variable, that is 1 if the ith averaging estimator is in the range $[(1-\varepsilon)\mu, (1+\varepsilon)\mu]$. Let $Y = \sum_{i=1}^{M} X_i$. We have that $\mathbb{E}[Y] \geq (3/4)M$. As such, by Lemma **??**, we have

$$\mathbb{P}[\text{bad output}] = \mathbb{P}[Y < (1/2)M] \leq \mathbb{P}[Y < (1 - 1/3)\,\mathbb{E}[Y]] \leq \exp\left(-\frac{(1/3)^2}{2}\,\mathbb{E}[Y]\right).$$

The later quantity is bounded by $\exp\left(-\frac{1}{18}\frac{3}{4}M\right) = \exp(-M/24) = \exp\left(-\left\lceil 24 \ln \varphi^{-1} \right\rceil /24\right) \leq \varphi$. ■

26.2. Frequency estimation over a stream for the kth moment

Let $\mathcal{S} = (s_1, \ldots, s_m)$ be a stream (i.e., sequence) of m elements from $N = \{1, \ldots, n\}$. Let f_i be the number of times the number i appears in $\mathcal{S}$. For $k \geq 0$, let

$$F_k = \sum_{i=1}^{n} f_i^k$$

be the ***kth frequency moment*** of $\mathcal{S}$. The quantity, $F_1 = m$ is the length of the stream $\mathcal{S}$. Similarly, F_0 is the number of distinct elements (where we use the convention that $0^0 = 0$ and any other quantity to the power 0 is 1). It is natural to define $F_\infty = \max_i f_i$.

Here, we are interested in approximating up to a factor of $1 \pm \varepsilon$ the quantity F_k, for $k \geq 1$ using small space, and reading the stream $\mathcal{S}$ only once.

26.2.1. An estimator for the kth moment

26.2.1.1. Basic estimator

One can pick a representative element from a stream uniformly at random by using reservoir sampling. That is, sample the ith element s_i to be the representative with probability $1/i$. Once sampled, the algorithm counts

how many times it see the representative value later on in the stream (the counter is initialized to 1, to account for the chosen representative itself). In particular, if s_p is the chosen representative in the end of the stream (i.e., the algorithm might change the representative several times), then the counter value is

$$r = \left|\left\{ j \;\middle|\; j \geq p \text{ and } s_j = s_p \right\}\right|.$$

The output of the algorithm is the quantity

$$X = m\left(r^k - (r-1)^k\right),$$

where m is the number of elements seen in the stream. Let V be the random variable that is the value of the representative in the end of the sequence.

26.2.1.2. Analysis

Lemma 26.2.1. *We have* $\mathbb{E}[X] = F_k$.

Proof: Observe that since we choose the representative uniformly at random, we have

$$\mathbb{E}[X \mid V = i] = \sum_{j=1}^{f_i} \frac{1}{f_i} m(j^k - (j-1)^k) = \frac{m}{f_i} \sum_{j=1}^{f_i} (j^k - (j-1)^k) = \frac{m}{f_i} f_i^k.$$

As such, we have $\mathbb{E}[X] = \mathbb{E}[\mathbb{E}[X \mid V]] = \sum_{i:f_i \neq 0} \frac{f_i}{m} \frac{m}{f_i} f_i^k = \sum_i f_i^k = F_k$. ■

Remark 26.2.2. In the above, we estimated the function $g(x) = x^k$, over the frequency numbers $f_1, \ldots, f_k$, but the above argumentation, on the expectation of X, would work for any function $g(x)$ such that $g(0) = 0$, and $g(x) \geq 0$, for all $x \geq 0$.

Lemma 26.2.3. *For $k > 1$, we have* $\sum_{i=1}^{n} \left(i^k - (i-1)^k\right)^2 \leq k n^{2k-1}$.

Proof: Observe that for $x \geq 1$, we have that $x^k - (x-1)^k \leq k x^{k-1}$. As such, we have

$$\sum_{i=1}^{n} \left(i^k - (i-1)^k\right)^2 \leq \sum_{i=1}^{n} k i^{k-1} \left(i^k - (i-1)^k\right) \leq k n^{k-1} \sum_{i=1}^{n} \left(i^k - (i-1)^k\right) = k n^{k-1} n^k = k n^{2k-1}.$$ ■

Lemma 26.2.4. *We have* $\mathbb{E}\left[X^2\right] \leq k m F_{2k-1}$.

Proof: By Lemma 26.2.3, we have

$$\mathbb{E}\left[X^2 \;\middle|\; V = i\right] = \sum_{j=1}^{f_i} \frac{1}{f_i} m^2 (j^k - (j-1)^k)^2 \leq \frac{m^2}{f_i} k f_i^{2k-1} = m^2 k f_i^{2k-2},$$

and thus $\mathbb{E}[X^2] = \mathbb{E}[\mathbb{E}[X^2 \mid V]] = \sum_{i:f_i \neq 0} \frac{f_i}{m} \cdot m^2 k f_i^{2k-2} = mkF_{2k-1}$. ■

Lemma 26.2.5. *For any non-negative numbers $f_1, \ldots, f_n$, and $k \geq 1$, we have*

$$\sum_{i=1}^{n} f_i \leq n^{(k-1)/k} \left(\sum_{i=1}^{n} f_i^k\right)^{1/k}.$$

Proof: This is immediate from Hölder inequality, but here is a self contained proof. The above is equivalent to proving that $\sum_i f_i/n \le \left(\sum_{i=1}^n f_i^k/n\right)^{1/k}$. Raising both sides to the power k, we need to show that $(\sum_i f_i/n)^k \le \sum_{i=1}^n f_i^k/n$. Setting $g(x) = x^k$, we have $g(\sum_i f_i/n) \le \sum_{i=1}^n g(f_i)/n$. The last inequality holds by the convexity of the function $g(x)$ (indeed, $g'(x) = kx^{k-1}$ and $g''(x) = k(k-1)x^{k-2} \ge 0$, for $x \ge 0$). ■

Lemma 26.2.6. *For any n numbers $f_1, \ldots, f_n \ge 0$, we have $\left(\sum_i f_i\right)\left(\sum_i f_i^{2k-1}\right) \le n^{1-1/k}\left(\sum_i f_i^k\right)^2$.*

Proof: Let $M = \max_i f_i$ and $m = \sum_i f_i$. We have

$$\sum_i f_i^{2k-1} \le M^{k-1} \sum_i f_i^k \le M^{k(k-1)/k} \sum_i f_i^k \le \left(\sum_i f_i^k\right)^{(k-1)/k} \sum_i f_i^k \le \left(\sum_i f_i^k\right)^{(2k-1)/k}.$$

By Lemma 26.2.5, we have $\sum_{i=1}^n f_i \le n^{(k-1)/k}\left(\sum_i f_i^k\right)^{1/k}$. Multiplying the above two inequality implies the claim. ■

Lemma 26.2.7. *We have $\mathbb{V}[X] \le kn^{1-1/k}F_k^2$.*

Proof: Since $m = \sum_i f_i$, Lemma 26.2.4 and Lemma 26.2.6 together implies that

$$\mathbb{V}[X] = \mathbb{E}\left[X^2\right] - (\mathbb{E}[X])^2 \le \mathbb{E}\left[X^2\right] \overbrace{\le}^{\text{L26.2.4}} kmF_{2k-2} = k\left(\sum_i f_i\right)\left(\sum_i f_i^{2k-1}\right) \overbrace{\le}^{\text{L26.2.6}} kn^{1-1/k}F_k^2.$$ ■

26.2.2. An improved estimator: Plugin

We have an estimator for F_k using constant space $O(1)$. Specifically, $\mu = \mathbb{E}[X] = F_k$ see Lemma 49.1.1, and $\nu = \mathbb{V}[X] \le kn^{1-1/k}F_k^2$. Let

$$M = 24\left\lceil \frac{4\nu}{\varepsilon^2\mu^2} \right\rceil \ln \frac{1}{\varphi}$$

We compute M estimators as the above (in parallel on the stream), and combine them as specified by Theorem 26.1.2, to get a new estimate Z. We have that

$$\mathbb{P}\left[(1-\varepsilon)\mu \le Z \le (1+\varepsilon)\mu\right] \ge 1 - \varphi.$$

Thus, the amount of space this streaming algorithm is using is proportional to M, and we have

$$M = O\left(\frac{\nu}{\varepsilon^2\mu^2} \ln \frac{1}{\varphi}\right) = O\left(\frac{kn^{1-1/k}F_k^2}{\varepsilon^2 F_k^2} \ln \frac{1}{\varphi}\right) = O\left(\frac{kn^{1-1/k}}{\varepsilon^2} \ln \frac{1}{\varphi}\right).$$

We thus proved the following.

In the following, we consider a computer ***word*** to be sufficiently large to contain $\lg n$ or $\lg m$ bits. This readily implies the following.

Theorem 26.2.8. *Let $\mathcal{S} = (s_1, \ldots, s_n)$ be a stream of numbers from the set $\{1, \ldots, n\}$. Let $k \ge 1$ be a parameter. Given $\varepsilon, \varphi \in (0,1)$, one can build a data-structure using $O(kn^{1-1/k}\varepsilon^{-2}\log\varphi^{-1})$ words, such that one can $(1 \pm \varepsilon)$-approximate the kth moment of the elements in the stream; that is, the algorithm outs a number Z, such that $(1-\varepsilon)F_k \le Z \le (1+\varepsilon)F_k$, where $F_k = \sum_{i=1}^n f_i^k$, and f_i is the number of times i appears in the stream $\mathcal{S}$. The algorithm succeeds with probability $\ge 1 - \varphi$.*

26.3. Better estimation for F_2

26.3.1. Pseudo-random k-wide independent sequence of signed bits

In the following, assume that we sample $O(\log n)$ bits, such that given an index i, one can compute (quickly!) a random signed bit $b(i) \in \{-1, +1\}$. We require that the resulting bits $b(1), b(2), \ldots, b(n)$ are 4-wise independent. To this end, pick a prime p, that is, say bigger than n^{10}. This can easily be done by sampling a number in the range $[n^{10}, n^{11}]$, and checking if it is prime (which can done in polynomial time).

Once we have such a prime, we generate a random polynomial $g(i) = \sum_{i=0}^{5} c_i x^i \bmod p$, by choosing $c_0, \ldots, c_5$ from $\mathbb{Z}_p = \{0, \ldots, p-1\}$. We had seen that $g(0), g(1), \ldots, g(n)$ are uniformly distributed in $\mathbb{Z}_p$, and they are, say, 6-wise independent (see Theorem 7.2.10).

We define

$$b(i) = \begin{cases} 0 & g(i) = p-1 \\ +1 & g(i) \text{ is odd} \\ -1 & g(i) \text{ is even.} \end{cases}$$

Clearly, the sequence $b(1), \ldots, b(n)$ are 6-wise independent. There is a chance that one of these bits might be zero, but the probability for that is at most n/p, which is so small, that we just assume it does not happen. There are known constructions that do not have this issue at all (one of the bits is zero), but they are more complicated.

Lemma 26.3.1. *Given a parameter $\varphi \in (0, 1)$, in polynomial time in $O(\log(n/\varphi))$, one can construct a function $b(\cdot)$, requiring $O(\log(n/\varphi))$ bits of storage (or $O(1)$ words), such that $b(1), \ldots, b(n) \in \{-1, +1\}$ with equal probability, an they are 6-wise independent. Furthermore, given i, one can compute $b(i)$ in $O(1)$ time.*

The probability of this sequence to fail having the desired properties is smaller than φ.

Proof: We repeat the above construction, but picking a prime p in the range, say, $n^{10}/\varphi \ldots n^{11}/\varphi$. ■

26.3.2. Estimator construction for F_2

26.3.2.1. The basic estimator

As before we have the stream $\mathcal{S} = s_1 \ldots, s_m$ of numbers from the set $1, \ldots, n$. We compute the 6-wise independent sequence of random bits of Lemma 26.3.1, and in the following we assume this sequence is good (i.e., has only -1 and $+1$ in it). We compute the quantity

$$T = \sum_{i=1}^{m} b(i) f_i = \sum_{j=1}^{m} b(s_j),$$

which can be computed on the fly using $O(1)$ words of memory, and $O(1)$ time per time in the stream.

The algorithm returns $X = T^2$ as the desired estimate.

Analysis.

Lemma 26.3.2. *We have $\mathbb{E}[X] = \sum_i f_i^2 = F_2$ and $\mathbb{V}[X] \le 2F_2^2$.*

Proof: We have that $\mathbb{E}[X] = \mathbb{E}\Big[\Big(\sum_{i=1}^{n} b(i) f_i\Big)^2\Big]$, and as such

$$\mathbb{E}[X] = \mathbb{E}\Big[\sum_{i=1}^{n} (b(i))^2 f_i^2 + 2\sum_{i<j} b(i)b(j) f_i f_j\Big] = \sum_{i=1}^{m} f_i^2 + 2\sum_{i<j} f_i f_j \,\mathbb{E}[b(i)b(j)] = \sum_{i=1}^{m} f_i^2 = F_2,$$

since $\mathbb{E}[b(i)] = 0$, $\mathbb{E}\Big[b(i)^2\Big] = 1$, and $\mathbb{E}[b(i)b(j)] = \mathbb{E}[b(i)]\,\mathbb{E}[b(j)] = 0$ (assuming the sequence $b(1), \ldots, b(n)$ has not failed), by the 6-wise Independence of the sequence of signed bits.

We next compute $\mathbb{E}\Big[X^2\Big]$. To this end, let $N = \{1, \ldots, n\}$, and $\Gamma = N \times N \times N \times N$. We split this set into several sets, as follows:

(i) $\Gamma_0 = \Big\{(i,i,i,i) \in N^4\Big\}$: All quadruples that are all the same value.

(ii) Γ_1: Set of all quadruples (i, j, k, l) where there is at least one value that appears exactly once.

(iii) Γ_2: Set of all (i, j, k, ℓ) with only two distinct values, each appearing exactly twice.

Clearly, we have $N^4 = \Gamma_0 \cup \Gamma_1 \cup \Gamma_2$.

For a tuple $(i,i,i,i) \in \Gamma_0$, we have $\mathbb{E}[b(i)b(i)b(i)b(i)] = \mathbb{E}\Big[b(i)^4\Big] = 1$.

For a tuple $(i, j, k, \ell) \in \Gamma_1$ with i being the unique value, we have that

$$\mathbb{E}[b(i)b(j)b(k)b(\ell)] = \mathbb{E}[b(i)]\,\mathbb{E}[b(j)b(k)b(\ell)] = 0\,\mathbb{E}[b(j)b(k)b(\ell)] = 0,$$

using that the signed bits are 4-wise independent.

For a tuple $(i,i,j,j) \in \Gamma_2$, we have $\mathbb{E}[b(i)b(i)b(j)b(j)] = \mathbb{E}\Big[b(i)^2 b(j)^2\Big] = \mathbb{E}\Big[b(i)^2\Big]\,\mathbb{E}\Big[b(j)^2\Big] = 1$, and the same argumentation applies to any tuple of Γ_2. Observe that for any $i < j$, there are $\binom{4}{2} = 6$ different tuples in Γ_2 that are made out of i and j. As such, we have

$$\begin{aligned}
\mathbb{E}\Big[X^2\Big] &= \mathbb{E}\Big[\Big(\sum_{i=1}^{n} b(i) f_i\Big)^4\Big] = \mathbb{E}\Big[\sum_{(i,j,k,\ell)\in\Gamma} b(i)b(j)b(k)b(\ell) f_i f_j f_k f_\ell\Big] \\
&= \sum_{(i,i,i,i)\in\Gamma_0} \mathbb{E}\Big[b(i)^4\Big] f_i^4 + \sum_{(i,j,k,\ell)\in\Gamma_1} f_i f_j f_k f_\ell \,\mathbb{E}[b(i)b(j)b(k)b(\ell)] + 6\sum_{i<j} \mathbb{E}\Big[b(i)^2 b(j)^2\Big] f_i^2 f_j^2 \\
&= \sum_{i=1}^{n} f_i^4 + 6\sum_{i<j} f_i^2 f_j^2.
\end{aligned}$$

As such, we have

$$\mathbb{V}[X] = \mathbb{E}\Big[X^2\Big] - (\mathbb{E}[X])^2 = \sum_{i=1}^{n} f_i^4 + 6\sum_{i<j} f_i^2 f_j^2 - \Big(\sum_{i=1}^{m} f_i^2\Big)^2 = 4\sum_{i<j} f_i^2 f_j^2 \le 2F_2^2. \qquad \blacksquare$$

26.3.3. Improving the estimator

We repeat the same scheme as above. Let $\varphi, \varepsilon \in (0,1)$ be parameters. In the following, let

$$\alpha = 16/\varepsilon^2 \qquad \text{and} \qquad \beta = 4\ln\frac{1}{\varphi}.$$

Let $X_{i,j}$ be a basic estimator for F_2, using the estimator of Section 26.3.2.1, for $i = 1, \ldots, \beta$ and $j = 1, \ldots, \alpha$. Let $Y_i = \sum_{j=1}^{\alpha} X_{i,j}/\alpha$, for $i = 1, \ldots, \beta$. Let Z be the median of $Y_1, \ldots, Y_\beta$, and the algorithm returns Z as the estimator.

Theorem 26.3.3. *Given a stream $\mathcal{S} = s_1, \ldots, s_m$ of numbers from $\{1, \ldots, n\}$, and parameters $\varepsilon, \varphi \in (0,1)$, one can compute an estimate Z for $F_2(\mathcal{S})$, such that $\mathbb{P}[|Z - F_2| > \varepsilon F_2] \leq \varphi$. This algorithm requires $O(\varepsilon^{-2} \log \varphi^{-1})$ space (in words), and this is also the time to handle a new element in the stream.*

Proof: The scheme is described above. As before, using Chebychev's inequality, we have that

$$\mathbb{P}[|Y_i - F_2| > \varepsilon F_2] = \mathbb{P}\left[|Y_i - F_2| > \frac{\varepsilon F_2}{\sqrt{\mathbb{V}[Y_i]}} \sqrt{\mathbb{V}[Y_i]}\right] \leq \frac{\mathbb{V}[Y_i]}{\varepsilon^2 F_2^2} = \frac{\mathbb{V}[X]/\alpha}{\varepsilon^2 F_2^2} \leq \frac{2F_2^2}{\alpha \varepsilon^2 F_2^2} = \frac{1}{8},$$

by Lemma 26.3.2. Let U be the number of estimators in $Y_1, \ldots, Y_\beta$ that are outside the acceptable range. Arguing as in Lemma **??**, we have

$$\mathbb{P}[Z \text{ is bad}] \leq \mathbb{P}[U \geq \beta/2] = \mathbb{P}[U \geq (1+3)\beta/8] \leq \exp(-(\beta/8)3^2/4) \leq \exp\left(-\ln \frac{1}{\varphi}\right) = \varphi,$$

by Chernoff inequality (Lemma 13.2.6), and ■

26.4. Bibliographical notes

The beautiful results of this chapter are from a paper from Alon *et al.* [AMS99].

References

[AMS99] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1): 137–147, 1999.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

Chapter 27

Approximating the Number of Distinct Elements in a Stream

"See? Genuine-sounding indignation. I programmed that myself. It's the first thing you need in a university environment: the ability to take offense at any slight, real or imagined."

Robert Sawyer, Factoring Humanity

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

27.1. Counting number of distinct elements

27.1.1. First order statistic

Let $X_1, \ldots, X_u$ be u random variables uniformly distributed in $[0, 1]$. Let $Y = \min(X_1, \ldots, X_u)$. The value Y is the ***first order statistic*** of $X_1, \ldots, X_u$.

For a continuous variable X, the ***probability density function*** (i.e., **pdf**) is the "probability" of X having this value. Since this is not well defined, one looks on the ***cumulative distribution function*** $F(x) = \mathbb{P}[X \leq]$. The pdf is then the derivative of the cdf. Somewhat abusing notations, the pdf of the X_is is $\mathbb{P}[X_i = x] = 1$.

The following proof is somewhat dense, check any standard text on probability for more details.

Lemma 27.1.1. *The* probability density function *of Y is* $f(x) = \binom{u}{1} 1(1-x)^{u-1}$.

Proof: Considering the pdf of X_1 being x, and all other X_is being bigger. We have that this pdf is

$$g(x) = \mathbb{P}\Big[(X_1 = x) \cap \bigcap_{i=2}^{u} (X_i > X_1)\Big] = \mathbb{P}\Big[\bigcap_{i=2}^{u} (X_i > X_1) \;\Big|\; X_1 = x\Big] \mathbb{P}[X_1 = x] = (1-x)^{u-1}.$$

Since every one of the X_i has equal probability to realize Y, we have $f(x) = ug(x)$. ■

Lemma 27.1.2. *We have* $\mathbb{E}[Y] = \frac{1}{u+1}$, $\mathbb{E}\Big[Y^2\Big] = \frac{2}{(u+1)(u+2)}$, *and* $\mathbb{V}[Y] = \frac{u}{(u+1)^2(u+2)}$.

Proof: Using integration by guessing, we have

$$\mathbb{E}[Y] = \int_{y=0}^{1} y\, \mathbb{P}[Y = y]\, \mathrm{d}y = \int_{y=0}^{1} y \cdot \binom{u}{1} 1(1-y)^{u-1}\, \mathrm{d}y = \int_{y=0}^{1} uy(1-y)^{u-1}\, \mathrm{d}y$$

$$= \Big[-y(1-y)^u - \frac{(1-y)^{u+1}}{u+1}\Big]_{y=0}^{1} = \frac{1}{u+1}.$$

Using integration by guessing again, we have

$$\begin{aligned}\mathbb{E}\Big[Y^2\Big] &= \int_{y=0}^{1} y^2\,\mathbb{P}[Y=y]\,\mathrm{d}y = \int_{y=0}^{1} y^2 \cdot \binom{u}{1} 1(1-y)^{u-1}\,\mathrm{d}y = \int_{y=0}^{1} uy^2(1-y)^{u-1}\,\mathrm{d}y \\ &= \Big[-y^2(1-y)^u - \frac{2y(1-y)^{u+1}}{u+1} - \frac{2(1-y)^{u+2}}{(u+1)(u+2)}\Big]_{y=0}^{1} = \frac{2}{(u+1)(u+2)}.\end{aligned}$$

We conclude that

$$\mathbb{V}[Y] = \mathbb{E}\Big[X^2\Big] - (\mathbb{E}[X])^2 = \frac{2}{(u+1)(u+2)} - \frac{1}{(u+1)^2} = \frac{1}{u+1}\left(\frac{2}{u+2} - \frac{1}{u+1}\right) = \frac{u}{(u+1)^2(u+2)}.$$ ■

27.1.2. The algorithm

A single estimator. Assume that we have a perfectly random hash function h that randomly maps $N = \{1,\ldots,n\}$ to $[0,1]$. Assume that the stream has u unique numbers in N. Then the set $\{h(s_1),\ldots,h(s_m)\}$ contains u random numbers uniformly distributed in [0.1]. The algorithm as such, would compute $X = \min_i h(s_i)$.

Explanation. Note, that X is *not* an estimator for u – instead, as $\mathbb{E}[X] = 1/(u+1)$, we are estimating $1/(u+1)$. The key observation is that an $1\pm\varepsilon$ estimator for $1/(u+1)$, is $1\pm O(\varepsilon)$ estimator for $u+1$, which is in turn an $1\pm O(\varepsilon)$ estimator for u.

Lemma 27.1.3. *Let $\varepsilon,\varphi \in (0,1)$ be parameters. Given a stream $\mathcal{S}$ of items from $\{1,\ldots,n\}$ one can return an estimate X, such that $\mathbb{P}\Big[(1-\varepsilon/4)\frac{1}{u+1} \le X \le (1+\varepsilon/4)\frac{1}{u+1}\Big] \ge 1-\varphi$, where u is the number of unique elements in $\mathcal{S}$. This requires $O\Big(\frac{1}{\varepsilon^2}\log\frac{1}{\varphi}\Big)$ space.*

Proof: The basic estimator Y has $\mu = \mathbb{E}[Y] = \frac{1}{u+1}$ and $\nu = \mathbb{V}[Y] = \frac{u}{(u+1)^2(u+2)}$. We now plug this estimator into the mean/median framework. By Lemma 27.1.2, for c some absolute constant, this requires maintaining M estimators, where M is larger than

$$c\frac{4\cdot 16\nu}{\varepsilon^2\mu^2}\log\frac{1}{\varphi} = O\left(\frac{u^2}{\varepsilon^2 u^2}\log\frac{1}{\varphi}\right) = O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varphi}\right).$$ ■

Observe that if $(1-\varepsilon/4)\frac{1}{u+1} \le X \le (1+\varepsilon/4)\frac{1}{u+1}$ then

$$\frac{u+1}{1-\varepsilon/4} - 1 \ge \frac{1}{X} - 1 \ge \frac{u+1}{1+\varepsilon/4} - 1,$$

which implies

$$(1+\varepsilon)u \ge \frac{(1+\varepsilon/4)u}{1-\varepsilon/4} \ge \frac{u+\varepsilon/4}{1-\varepsilon/4} \ge \frac{1}{X} - 1 \ge \frac{u+1}{1+\varepsilon/4} - 1 \ge (1-\varepsilon)u.$$

Namely, $1/X - 1$ is a good estimator for the number of distinct elements.

The algorithm revisited. Compute X as above, and output the quantity $1/X - 1$.

This immediately implies the following.

Lemma 27.1.4. *Under the unreasonable assumption that we can sample perfectly random functions from $\{1, \ldots, n\}$ to $[0, 1]$, and storing such a function requires $O(1)$ words, then one can estimate the number of unique elements in a stream, using $O(\varepsilon^{-2} \log \varphi^{-1})$ words.*

27.2. Sampling from a stream with "low quality" randomness

Assume that we have a stream of elements $\mathcal{S} = s_1, \ldots, s_m$, all taken from the set $\{1, \ldots, n\}$. In the following, let $\mathrm{set}(S)$ denote the set of values that appear in S. That is

$$F_0 = F_0(\mathcal{S}) = |\mathrm{set}(S)|$$

is the number of distinct values in the stream $\mathcal{S}$.

Assume that we have a random sequence of bits $\mathcal{B} \equiv B_1, \ldots, B_n$, such that $\mathbb{P}[B_i = 1] = p$, for some p. Furthermore, we can compute B_i efficiently. Assume that the bits of $\mathcal{B}$ are pairwise independent.

The sampling algorithm. When the ith arrives s_i, we compute B_{s_i}. If this bit is 1, then we insert s_i into the random sample R (if it is already in R, there is no need to store a second copy, naturally).

This defines a natural random sample

$$R = \{i \mid B_i = 1 \text{ and } i \in S\} \subseteq S.$$

Lemma 27.2.1. *For the above random sample R, let $X = |R|$. We have that $\mathbb{E}[X] = p\nu$ and $\mathbb{V}[X] = p\nu - p^2\nu$, where $\nu = F_0(\mathcal{S})$ is the number of district elements in S.*

Proof: Let $X = |R|$, and we have

$$\mathbb{E}[X] = \mathbb{E}\Big[\sum_{i \in S} B_i\Big] = \sum_{i \in S} \mathbb{E}[B_i] = p\nu.$$

As for the $\mathbb{E}\Big[X^2\Big]$, we have

$$\mathbb{E}\Big[X^2\Big] = \mathbb{E}\Big[(\sum_{i \in S} B_i)^2\Big] = \sum_{i \in S} \mathbb{E}\Big[B_i^2\Big] + 2 \sum_{i,j \in S,\, i<j} \mathbb{E}\Big[B_i B_j\Big] = p\nu + 2 \sum_{i,j \in S,\, i<j} \mathbb{E}[B_i]\,\mathbb{E}[B_j] = p\nu + 2p^2 \binom{\nu}{2}.$$

As such, we have

$$\begin{aligned}\mathbb{V}[X] = \mathbb{V}[|R|] &= \mathbb{E}\Big[X^2\Big] - (\mathbb{E}[X])^2 = p\nu + 2p^2\binom{\nu}{2} - p^2\nu^2 = p\nu + 2p^2\frac{\nu(\nu-1)}{2} - p^2\nu^2 \\ &= p\nu + p^2\nu(\nu-1) - p^2\nu^2 = p\nu - p^2\nu.\end{aligned}$$

■

Lemma 27.2.2. *Let $\varepsilon \in (0, 1/4)$. Given $O(1/\varepsilon^2)$ space, and a parameter N. Consider the task of estimating the size of $F_0 = |\mathrm{set}(\mathcal{S})|$, where $F_0 > N/4$. Then, the algorithm described below outputs one of the following:*

(A) $F_0 > 2N$.

(B) Output a number ρ such that $(1 - \varepsilon)F_0 \le \rho \le (1 + \varepsilon)F_0$.

(Note, that the two options are not disjoint.) The output of this algorithm is correct, with probability $\ge 7/8$.

Proof: We set $p = \frac{c}{N\varepsilon^2}$, where c is a constant to be determined shortly. Let $T = pN = O(1/\varepsilon^2)$. We sample a random sample R from S, by scanning the elements of S, and adding $i \in S$ to R if $B_i = 1$, If the random sample is larger than $8T$, at any point, then the algorithm outputs that $|S| > 2N$.

In all other cases, the algorithm outputs $|R|/p$ as the estimate for the size of S, together with R.

To bound the failure probability, consider first the case that $N/4 < |\mathrm{set}(\mathcal{S})|$. In this case, we have by the above, that

$$\mathbb{P}[|X - \mathbb{E}[X]| > \varepsilon\,\mathbb{E}[X]] \le \mathbb{P}\left[|X - \mathbb{E}[X]| > \varepsilon \frac{\mathbb{E}[X]}{\sqrt{\mathbb{V}[X]}} \sqrt{\mathbb{V}[X]}\right] \le \varepsilon^2 \frac{\mathbb{V}[X]}{(\mathbb{E}[X])^2} \le \frac{1}{8},$$

if $\frac{\mathbb{V}[X]}{\varepsilon^2(\mathbb{E}[X])^2} \le \frac{1}{8}$, For $\nu = F_0 \ge N/4$, this happens if $\frac{p\nu}{\varepsilon^2 p^2 \nu^2} \le \frac{1}{8}$. This in turn is equivalent to $8/\varepsilon^2 \le p\nu$. This is in turn happens if

$$\frac{c}{N\varepsilon^2} \cdot \frac{N}{4} \ge \frac{8}{\varepsilon^2},$$

which implies that this holds for $c = 32$. Namely, the algorithm in this case would output a $(1 \pm \varepsilon)$-estimate for $|S|$.

If the sample get bigger than $8T$, then the above readily implies that with probability at least $7/8$, the size of S is at least $(1 - \varepsilon)8T/p > 2N$, Namely, the output of the algorithm is correct in this case. ■

Lemma 27.2.3. *Let $\varepsilon \in (0, 1/4)$ and $\varphi \in (0, 1)$. Given $O(\varepsilon^{-2} \log \varphi^{-1})$ space, and a parameter N, and the task is to estimate F_0 of $\mathcal{S}$, given that $F_0 > N/4$. Then, there is an algorithm that would output one of the following:*

(A) $F_0 > 2N$.

(B) Output a number ρ such that $(1 - \varepsilon)F_0 \le \rho \le (1 + \varepsilon)F_0$.

(Note, that the two options are not disjoint.) The output of this algorithm is correct, with probability $\ge 1 - \varphi$.

Proof: We run $O(\log \varphi^{-1})$ copies of the of Lemma 27.2.2. If half of them returns that $F_0 > 2N$, then the algorithm returns that $F_0 > 2N$. Otherwise, the algorithm returns the median of the estimates returned, and return it as the desired estimated. The correctness readily follows by a repeated application of Chernoff's inequality. ■

Lemma 27.2.4. *Let $\varepsilon \in (0, 1/4)$. Given $O(\varepsilon^{-2} \log^2 n)$ space, one can read the stream $\mathcal{S}$ once, and output a number ρ, such that $(1-\varepsilon)F_0 \le \rho \le (1+\varepsilon)F_0$. The estimate is correct with high probability (i.e., $\ge 1 - 1/n^{O(1)}$).*

Proof: Let $N_i = 2^i$, for $i = 1, \ldots, M = \lceil \lg n \rceil$. Run M copies of Lemma 27.2.3, for each value of N_i, with $\varphi = 1/n^{O(1)}$. Let $Y_1, \ldots, Y_M$ be the outputs of these algorithms for the stream. A prefix of these outputs, are going to be "$F_0 > 2N_i$", Let j be the first Y_j that is a number. Return this number as the desired estimate. The correctness is easy – the first estimate that is a number, is a correct estimate with high probability. Since $N_M \ge n$, it also follows that Y_M must be a number. As such, there is a first number in the sequence, and the algorithm would output an estimate.

More precisely, there is an index i, such that $N_i/4 \le F_0 \le 2F_0$, and Y_i is a good estimate, with high probability. If any of the Y_j, for $j < i$, is an estimate, then it is correct (again) with high probability. ■

27.3. Bibliographical notes

References

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

Chapter 28

Approximate Nearest Neighbor (ANN) Search in High Dimensions

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

Possession of anything new or expensive only reflected a person's lack of theology and geometry; it could even cast doubts upon one's soul.

A confederacy of Dunces, John Kennedy Toole

28.1. ANN on the hypercube

28.1.1. ANN for the hypercube and the Hamming distance

Definition 28.1.1. The set of points $\mathcal{H}^d = \{0,1\}^d$ is the ***d-dimensional hypercube***. A point $\mathsf{p} = (\mathsf{p}_1, \ldots, \mathsf{p}_d) \in \mathcal{H}^d$ can be interpreted, naturally, as a binary string $\mathsf{p}_1\mathsf{p}_2\ldots\mathsf{p}_d$. The ***Hamming distance*** $\mathsf{d}_H(\mathsf{p},\mathsf{u})$ between $\mathsf{p},\mathsf{u} \in \mathcal{H}^d$ is the number of coordinates where p and u disagree.

It is easy to verify that the Hamming distance, being the L_1-norm, complies with the triangle inequality and is thus a metric.

As we saw previously, to solve the $(1+\varepsilon)$-ANN problem efficiently, it is sufficient to solve the ***approximate near neighbor*** problem. Namely, given a set P of n points in $\mathcal{H}^d$, a radius $r > 0$, and parameter $\varepsilon > 0$, we want to decide for a query point q whether $\mathsf{d}_H(q,\mathsf{P}) \le r$ or $\mathsf{d}_H(q,\mathsf{P}) \ge (1+\varepsilon)r$, where $\mathsf{d}_H(q,\mathsf{P}) = \min_{\mathsf{p}\in\mathsf{P}} \mathsf{d}_H(q,\mathsf{p})$.

Definition 28.1.2. For a set P of points, a data-structure $\mathsf{D} = \mathcal{D}_{\approx\text{Near}}(\mathsf{P}, r, (1+\varepsilon)r)$ solves the ***approximate near neighbor*** problem if, given a query point q, the data-structure works as follows.

- NEAR: If $\mathsf{d}_H(q,\mathsf{P}) \le r$, then $\mathcal{D}$ outputs a point $\mathsf{p} \in \mathsf{P}$ such that $\mathsf{d}_H(\mathsf{p},q) \le (1+\varepsilon)r$.
- FAR: If $\mathsf{d}_H(q,\mathsf{P}) \ge (1+\varepsilon)r$, then $\mathcal{D}$ outputs "$\mathsf{d}_H(q,\mathsf{P}) \ge r$".
- DON'T CARE: If $r \le \mathsf{d}(q,\mathsf{P}) \le (1+\varepsilon)r$, then $\mathcal{D}$ can return either of the above answers.

Given such a data-structure, one can construct a data-structure that answers the approximate nearest neighbor query using $O\Big(\log\big(\varepsilon^{-1}\log d\big)\Big)$ queries using an approximate near neighbor data-structure. Indeed, the desired distance $\mathsf{d}_H(q,\mathsf{P})$ is an integer number in the range $0,1,\ldots,d$. We can build a $\mathcal{D}_{\approx\text{Near}}$ data-structure for distances $(1+\varepsilon)^i$, for $i = 1,\ldots,M$, where $M = O\big(\varepsilon^{-1}\log d\big)$. Performing a binary search over these distances would resolve the approximate nearest neighbor query and requires $O(\log M)$ queries.

As such, in the following, we concentrate on constructing the approximate near neighbor data-structure (i.e., $\mathcal{D}_{\approx\text{Near}}$).

28.1.2. Preliminaries

Definition 28.1.3. Consider a sequence m of k, not necessarily distinct, integers $i_1, i_2, \ldots, i_k \in [\![d]\!]$, where $[\![d]\!] = \{1, \ldots, d\}$. For a point $\mathsf{p} = (p_1, \ldots, p_d) \in \mathbb{R}^d$, its ***projection*** by m, denoted by $m\mathsf{p}$ is the point $(p_{i_1}, \ldots, p_{i_k}) \in \mathbb{R}^k$. Similarly, the *projection* of a point set $\mathsf{P} \subseteq \mathbb{R}^d$ by m is the point set $m\mathsf{P} = \{m\mathsf{p} \mid \mathsf{p} \in \mathsf{P}\}$.

Given two sequences $m = i_1, \ldots, i_k$ and $u = j_1, \ldots, j_{k'}$, let $m|u$ denote the *concatenated* sequence $m|u = i_1, \ldots, i_k, j_1, \ldots, j_{k'}$. Given a probability φ, a natural way to create such a projection, is to include the ith coordinate, for $i = 1, \ldots, d$, with probability φ. Let $\mathcal{D}_\varphi$ denote the distribution of such sequences of indices.

Definition 28.1.4. Let $\mathcal{D}_\varphi^T$ denote the distribution resulting from concatenating T independent sequences sampled from $\mathcal{D}_\varphi$. The length of a sampled sequence is dT.

Observe that for a point $\mathsf{p} \in \{0,1\}^d$, and $M \in \mathcal{D}_\varphi^T$, the projection $M\mathsf{p}$ might be higher dimensional than the original point p, as it might contain repeated coordinates of the original point.

28.1.2.1. Algorithm

28.1.2.1.1. Input. The input is a set P of n points in the hypercube $\{0,1\}^d$, and parameters r and ε.

28.1.2.1.2. Preprocessing. We set parameters as follows:

$$\beta = \frac{1}{1+\varepsilon} \in (0,1), \quad \varphi = 1 - \exp\left(-\frac{1}{r}\right) \approx \frac{1}{r}, \quad T = \beta \ln n, \quad \text{and} \quad L = O(n^\beta \log n).$$

We randomly and independently pick L sequences $M_1, \ldots, M_L \in \mathcal{D}_\varphi^T$. Next, the algorithm computes the point sets $\mathsf{Q}_i = M_i\mathsf{P}_i$, for $i = 1, \ldots, L$, and stores them each in a hash table, denoted by D_i, for $i = 1, \ldots, L$.

28.1.2.1.3. Answering a query. Given a query point $\mathsf{q} \in \{0,1\}^d$, the algorithm computes $\mathsf{q}_i = M_i\mathsf{q}$, for $i = 1, \ldots, L$. From each D_i, the algorithm retrieves a list ℓ_i of all the points that collide with q_i. The algorithm scans the points in the lists $\ell_1, \ldots, \ell_L$. If any of these points is in Hamming distance smaller than $(1+\varepsilon)r$, the algorithm returns it as the desired near-neighbor (and stops). Otherwise, the algorithm returns that all the points in P are in distance at least r from q.

28.1.2.2. Analysis

Lemma 28.1.5. *Let K be a set of r marked/forbidden coordinates. The probability that a sequence $M = (m_1, \ldots, m_T)$ sampled from $\mathcal{D}_\varphi^T$ does not sample any of the coordinates of K is $1/n^\beta$. This probability increases if K contains fewer coordinates.*

Proof: For any i, the probability that m_i does not contain any of these coordinates is $(1-\varphi)^r = (e^{-1/r})^r = 1/e$. Since this experiment is repeated T times, the probability is $e^{-T} = e^{-\beta \ln n} = n^{-\beta}$. ■

Lemma 28.1.6. *Let p be the nearest-neighbor to q in P. If $\mathsf{d}_H(\mathsf{q}, \mathsf{p}) \le r$ then, with high probability, the data-structure returns a point that is in distance $\le (1+\varepsilon)r$ from q.*

Proof: The good event here is that p and q collide under one of the sequences of $M_1, \ldots, M_L$. However, the probability that $M_i\mathsf{p} = M_i\mathsf{q}$ is at least $1/n^\beta$, by Lemma 28.1.5, as this is the probability that M_i does not sample any of the (at most r) coordinates where p and q disagree. As such, the probability that all L data-structures fail (i.e., none of the lists $\ell_1, \ldots, \ell_L$ contains p), is at most $(1 - 1/n^\beta)^L < 1/n^{O(1)}$, as $L = O\left(n^\beta \log n\right)$. ■

Lemma 28.1.7. *In expectation, the total number of points in $\ell_1, \ldots, \ell_L$ that are in distance $\geq (1+\varepsilon)r$ from q is $\leq L$.*

Proof: Let $\mathsf{P}_{\geq}$ be the set of points in P that are in distance $\geq (1+\varepsilon)r$ from q. For a point $\mathsf{u} \in \mathsf{P}_{\geq}$, with $\Delta = \mathsf{d}_H(\mathsf{u}, \mathsf{q})$, we have that the probability for $M \in \mathcal{D}_\varphi^T$ misses all the Δ coordinates, where u and q differ, is

$$(1-\varphi)^{\Delta T} \leq (1-\varphi)^{(1+\varepsilon)rT} = \left(e^{-1/r}\right)^{(1+\varepsilon)rT} = \exp(-(1+\varepsilon)\beta \ln n) = \frac{1}{n},$$

as $\varphi = 1 - e^{-1/r}$, $T = \beta \ln n$, and $\beta = 1/(1+\varepsilon)$. But then, for any i, we have

$$\mathbb{E}[|\ell_i|] = \sum_{\mathsf{p} \in \mathsf{P}_{\geq}} \mathbb{P}_{M_i}[M_i \mathsf{p} = M_i \mathsf{q}] \leq \left|\mathsf{P}_{\geq}\right| \frac{1}{n} \leq 1.$$

As such, the total number of far points in the lists is at most $L \cdot 1 = L$, implying the claim. ■

28.1.2.3. Running time

For each i, the query computes $M_i \mathsf{q}$ and that takes $O(dT) = O(d \log n)$ time. Repeated L times, this takes $O(Ld \log n)$ time overall. Let X be the random variable that is the number of points in the extracted lists that are in distance $\geq (1+\varepsilon)r$ from the query point. The time to scan the lists is $O(d(X+1))$, since the algorithm stops as soon as it finds a near point. As such, by Lemma 28.1.7, the expected query time is $O(Ld \log n + Ld) = O\left(dn^{1/(1+\varepsilon)} \log^2 n\right)$.

28.1.2.3.1. Improving the performance (a bit). Observe that for $M \in \mathcal{D}_\varphi^T$, and any two points $\mathsf{p}, \mathsf{u} \in \{0,1\}^d$, all the algorithm cares about is whether $M\mathsf{p} = M\mathsf{u}$. As such, if a coordinate is probed many times by M, we might as well probe this coordinate only once. In particular, for a sequence $M \in \mathcal{D}_\varphi^T$, let $M' = \mathrm{uniq}(M)$ be the projection sequence resulting from removing replications in M. Significantly, M' is only of length $\leq d$, and as such, computing $M'\mathsf{p}$, for a point p, takes only $O(d)$ time. It is not hard to verify that one can also sample directly $\mathrm{uniq}(M)$, for $M \in \mathcal{D}_\varphi^T$, in $O(d)$ time. This improves the query and processing by a logarithmic factor.

We thus get the following result.

Theorem 28.1.8. *Given a set P of n points in $\{0,1\}^d$, and parameters r, ε, one can preprocess P in*

$$O(dn^{1+1/(1+\varepsilon)} \log n)$$

time and space, such that given a query point q, the algorithm returns, in expected $O(dn^{1/(1+\varepsilon)} \log n)$ time, one of the following:

(A) a point $\mathsf{p} \in P$ such that $\mathsf{d}_H(\mathsf{q}, \mathsf{p}) \leq (1+\varepsilon)r$, or

(B) the distance of q from P is larger than r.

The algorithm may return either result if the distance of q from P is in the range $[r, (1+\varepsilon)r]$. The algorithm succeeds with high probability (per query).

One can also get a high-probability guarantee on the query time. For a parameter $\delta > 0$, create $O(\log \delta^{-1})$ LSH data-structures as above. Perform the query as above, except that when the query time exceeds (say) twice the expected time, move on to redo the query in the next LSH data-structure. The probability that the query had failed on one of these LSH data-structures is $\leq 1/2$, by Markov's inequality. As such, overall, the query time becomes $O(dn^{1/(1+\varepsilon)} \log n \log \delta^{-1})$, with probability $\geq 1 - \delta$.

28.2. Testing for good items

Imagine that we have n items. One of the items is *good* the rest are *bad*. We have two tests to check if an item is good – we have a cheap test, a really expensive test. We would like to use the expensive test as few times as possible, and classify correctly all the items. Let $T(x) \in \{\text{good}, \text{bad}\}$ denote the result of the cheap test on item x. We have that

$$\mathbb{P}[T(x) = \text{good} \mid x \text{ is good}] \geq \alpha > \beta \geq \mathbb{P}[T(x) = \text{good} \mid x \text{ is bad}].$$

Repeating, the experiment k times, we create a ***k-test*** where we turn an item is good if all k tests return "good". We then have

$$\mathbb{P}\Big[T^k(x) = \text{good} \mid x \text{ is good}\Big] \geq \alpha^k > \beta^k \geq \mathbb{P}\Big[T^k(x) = \text{good} \mid x \text{ is bad}\Big].$$

We need to make sure we discover the good item, so let us repeat the k-test enough times, till we discover it with good probability. A natural value would be to repeat the k-test for each item $M = (1/\alpha^k) \ln \varphi^{-1}$ times, so that the probability we fail to discover the good item is

$$(1 - \alpha^k)^M \leq \exp\Big(-\alpha^k M\Big) < \varphi.$$

As for the bad items, how many "false positive" would we have? Every k-test is going to return in expectation at most

$$(n-1)\beta^k \leq n\beta^k$$

items. As such, the total number of false positives over the M repeated k-tests is going to be

$$n\beta^k M = O(n(\beta/\alpha)^k \log \varphi^{-1}).$$

If everything is for free, that we will set k to be quite large, so that the number of false positives is practically zero. For our purposes it would be enough if every k-test returns (in expectation) one false positive. That is, we will require that

$$\beta^k n \leq 1.$$

This would set up the values we need for k and M.

28.3. LSH for the hypercube: An elaborate construction

We next present a similar scheme in a more systematic fashion – this would provide some intuition how we came up with the above construction.

28.3.0.1. On sense and sensitivity

Let $\mathsf{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ be a subset of vertices of the hypercube in d dimensions. In the following we assume that $d = n^{O(1)}$. Let $r, \varepsilon > 0$ be two prespecified parameters. We are interested in building an approximate near neighbor data-structure (i.e., $\mathcal{D}_{\approx \text{Near}}$) for balls of radius r in the Hamming distance.

Definition 28.3.1. A family $\mathcal{F}$ of functions (defined over $\mathcal{H}^d$) is ***$(r, R, \widehat{\alpha}, \widehat{\beta})$-sensitive*** if for any $q, \mathsf{v} \in \mathcal{H}^d$, we have the following

(A) If $\mathsf{v} \in \mathbf{b}(q, r)$, then $\mathbb{P}[f(q) = f(\mathsf{v})] \geq \widehat{\alpha}$.

(B) If $\mathsf{v} \notin \mathbf{b}(q, R)$, then $\mathbb{P}[f(q) = f(\mathsf{v})] \leq \widehat{\beta}$.

Here, f is a randomly picked function from $\mathcal{F}$, $r < R$, and $\widehat{\alpha} > \widehat{\beta}$.

Intuitively, if we can construct an (r, R, α, β)-sensitive family, then we can distinguish between two points which are close together and two points which are far away from each other. Of course, the probabilities α and β might be very close to each other, and we need a way to do amplification.

28.3.1. A simple sensitive family

A priori it is not even clear such a sensitive family exists, but it turns out that the family randomly exposing one coordinate is sensitive.

Lemma 28.3.2. *Let $f_i(\mathsf{p})$ denote the function that returns the ith coordinate of p, for $i = 1, \ldots, d$. Consider the family of functions $\mathcal{F} = \{f_1, \ldots, f_d\}$. Then, for any $r > 0$ and ε, the family $\mathcal{F}$ is $(r, (1+\varepsilon)r, \alpha, \beta)$-sensitive, where $\alpha = 1 - r/d$ and $\beta = 1 - r(1+\varepsilon)/d$.*

Proof: If $\mathsf{u}, \mathsf{v} \in \{0,1\}^d$ are within distance smaller than r from each other (under the Hamming distance), then they differ in at most r coordinates. The probability that a random $h \in \mathcal{F}$ would project into a coordinate that u and v agree on is $\geq 1 - r/d$.

Similarly, if $\mathsf{d}_H(\mathsf{u}, \mathsf{v}) \geq (1+\varepsilon)r$, then the probability that a random $h \in \mathcal{F}$ would map into a coordinate that u and v agree on is $\leq 1 - (1+\varepsilon)r/d$. ■

28.3.2. A family with a large sensitivity gap

Let k be a parameter to be specified shortly, and consider the family of functions $\mathcal{G}$ that concatenates k of the given functions. Formally, let

$$\mathcal{G} = \text{combine}(\mathcal{F}, k) = \left\{ g \;\middle|\; g(\mathsf{p}) = (f^1(\mathsf{p}), \ldots, f^k(\mathsf{p})), \text{ for } f^1, \ldots, f^k \in \mathcal{F} \right\}$$

be the set of all such functions.

Lemma 28.3.3. *For a (r, R, α, β)-sensitive family $\mathcal{F}$, the family $\mathcal{G} = \text{combine}(\mathcal{F}, k)$ is $(r, R, \alpha^k, \beta^k)$-sensitive.*

Proof: For two fixed points $\mathsf{u}, \mathsf{v} \in \mathcal{H}^d$ such that $\mathsf{d}_H(\mathsf{u}, \mathsf{v}) \leq r$, we have that for a random $h \in \mathcal{F}$, we have $\mathbb{P}[h(\mathsf{u}) = h(\mathsf{v})] \geq \alpha$. As such, for a random $g \in \mathcal{G}$, we have that

$$\begin{aligned}
\mathbb{P}[g(\mathsf{u}) = g(\mathsf{v})] &= \mathbb{P}\Big[f^1(\mathsf{u}) = f^1(\mathsf{v}) \text{ and } f^2(\mathsf{u}) = f^2(\mathsf{v}) \text{ and } \ldots \text{ and } f^k(\mathsf{u}) = f^k(\mathsf{v})\Big] \\
&= \prod_{i=1}^{k} \mathbb{P}\Big[f^i(\mathsf{u}) = f^i(\mathsf{v})\Big] \geq \alpha^k.
\end{aligned}$$

Similarly, if $\mathsf{d}_H(\mathsf{u}, \mathsf{v}) > R$, then $\mathbb{P}[g(\mathsf{u}) = g(\mathsf{v})] = \prod_{i=1}^{k} \mathbb{P}\Big[f^i(\mathsf{u}) = f^i(\mathsf{v})\Big] \leq \beta^k$. ■

The above lemma implies that we can build a family that has a gap between the lower and upper sensitivities; namely, $\alpha^k/\beta^k = (\alpha/\beta)^k$ is arbitrarily large. The problem is that if α^k is too small, then we will have to use too many functions to detect whether or not there is a point close to the query point.

Nevertheless, consider the task of building a data-structure that finds all the points of $\mathsf{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ that are equal, under a given function $g \in \mathcal{G} = \text{combine}(\mathcal{F}, k)$, to a query point. To this end, we compute the strings $g(\mathsf{p}_1), \ldots, g(\mathsf{p}_n)$ and store them (together with their associated point) in a hash table (or a prefix tree). Now, given a query point q, we compute $g(q)$ and fetch from this data-structure all the strings equal to it that are stored in it. Clearly, this is a simple and efficient data-structure. All the points colliding with q would be the natural candidates to be the nearest neighbor to q.

By not storing the points explicitly, but using a pointer to the original input set, we get the following easy result.

Lemma 28.3.4. *Given a function $g \in \mathcal{G} = \text{combine}(\mathcal{F}, k)$ (see Lemma 28.3.3) and a set $\mathsf{P} \subseteq \mathcal{H}^d$ of n points, one can construct a data-structure, in $O(nk)$ time and using $O(nk)$ additional space, such that given a query point q, one can report all the points in $X = \left\{ \mathsf{p} \in \mathsf{P} \;\middle|\; g(\mathsf{p}) = g(q) \right\}$ in $O(k + |X|)$ time.*

28.3.3. Amplifying sensitivity

Our task is now to amplify the sensitive family we currently have. To this end, for two τ-dimensional points x and y, let $x \eqcirc y$ be the Boolean function that returns *true* if there exists an index i such that $x_i = y_i$ and *false* otherwise. Now, the regular "=" operator requires vectors to be equal in all coordinates (i.e., it is equal to $\bigcap_i (x_i = y_i)$) while $x \eqcirc y$ is $\bigcup_i (x_i = y_i)$. The previous construction of Lemma 28.3.3 using this alternative equal operator provides us with the required amplification.

Lemma 28.3.5. *Given an $\left(r, R, \alpha^k, \beta^k\right)$-sensitive family $\mathcal{G}$, the family $\mathcal{H}_{\eqcirc} = \text{combine}(\mathcal{G}, \tau)$ if one uses the $\eqcirc$ operator to check for equality is $(r, R, 1-(1-\alpha^k)^\tau, 1-(1-\beta^k)^\tau)$-sensitive.*

Proof: For two fixed points $\mathsf{u}, \mathsf{v} \in \mathcal{H}^d$ such that $\mathsf{d}_H(\mathsf{u}, \mathsf{v}) \le r$, we have, for a random $g \in \mathcal{G}$, that $\mathbb{P}[g(\mathsf{u}) = g(\mathsf{v})] \ge \alpha^k$. As such, for a random $h \in \mathcal{H}_{\eqcirc}$, we have that

$$\begin{aligned}\mathbb{P}[h(\mathsf{u}) \eqcirc h(\mathsf{v})] &= \mathbb{P}\Big[g^1(\mathsf{u}) = g^1(\mathsf{v}) \text{ or } g^2(\mathsf{u}) = g^2(\mathsf{v}) \text{ or } \ldots \text{ or } g^\tau(\mathsf{u}) = g^\tau(\mathsf{v})\Big] \\ &= 1 - \prod_{i=1}^{\tau} \mathbb{P}\Big[g^i(\mathsf{u}) \neq g^i(\mathsf{v})\Big] \ge 1 - (1-\alpha^k)^\tau.\end{aligned}$$

Similarly, if $\mathsf{d}_H(\mathsf{u}, \mathsf{v}) > R$, then

$$\mathbb{P}[h(\mathsf{u}) \eqcirc h(\mathsf{v})] = 1 - \prod_{i=1}^{\tau} \mathbb{P}\Big[g^i(\mathsf{u}) \neq g^i(\mathsf{v})\Big] \le 1 - (1-\beta^k)^\tau. \qquad \blacksquare$$

To see the effect of Lemma 28.3.5, it is useful to play with a concrete example. Consider an $(r, R, \alpha^k, \beta^k)$-sensitive family where $\beta^k = \alpha^k/2$ and yet α^k is very small. Setting $\tau = 1/\alpha^k$, the resulting family is (roughly) $(r, R, 1-1/e, 1-1/\sqrt{e})$-sensitive. Namely, the gap shrank, but the threshold sensitivity is considerably higher. In particular, it is now a constant, and the gap is also a constant.

Using Lemma 28.3.5 as a data-structure to store P is more involved than before. Indeed, for a random function $h = \left(g^1, \ldots, g^\tau\right) \in \mathcal{H}_{\eqcirc} = \text{combine}(\mathcal{G}, \tau)$ building the associated data-structure requires us to build τ data-structures for each one of the functions $g^1, \ldots, g^\tau$, using Lemma 28.3.4. Now, given a query point, we retrieve all the points of P that collide with each one of these functions, by querying each of these data-structures.

Lemma 28.3.6. *Given a function $h \in \mathcal{H}_{\eqcirc} = \text{combine}(\mathcal{G}, \tau)$ (see Lemma 28.3.5) and a set $\mathsf{P} \subseteq \mathcal{H}^d$ of n points, one can construct a data-structure, in $O(nk\tau)$ time and using $O(nk\tau)$ additional space, such that given a query point q, one can report all the points in $X = \left\{\mathsf{p} \in \mathsf{P} \;\middle|\; h(\mathsf{p}) \eqcirc h(q)\right\}$ in $O(k\tau + |X|)$ time.*

28.3.4. The near neighbor data-structure and handling a query

We construct the data-structure $\mathcal{D}$ of Lemma 28.3.6 with parameters k and τ to be determined shortly, for a random function $h \in \mathcal{H}_{\eqcirc}$. Given a query point q, we retrieve all the points that collide with h and compute their distance to the query point. Next, scan these points one by one and compute their distance to q. As soon as encountering a point $\mathsf{v} \in \mathsf{P}$ such that $\mathsf{d}_H(q, \mathsf{v}) \le R$, the data-structures returns ***true*** together with v.

Let's assume that we know that the expected number of points of $\mathsf{P} \setminus \mathbf{b}(q, R)$ (i.e., $R = (1+\varepsilon)r$) that will collide with q in $\mathcal{D}$ is in expectation L (we will figure out the value of L below). To ensure the worst case query time, the query would abort after checking $4L + 1$ points and would return *false*. Naturally, the data-structure would also return false if all points encountered have distance larger than R from q.

Clearly, the query time of this data-structure is $O(k\tau + dL)$.

We are left with the task of fine-tuning the parameters τ and k to get the fastest possible query time, while the data-structure has reasonable probability to succeed. Figuring the right values is technically tedious, and we do it next.

28.3.4.1. Setting the parameters

If there exists $\mathsf{p} \in \mathsf{P}$ such that $\mathsf{d}_H(q, \mathsf{p}) \leq r$, then the probability of this point to collide with q under the function h is $\phi \geq 1 - \left(1 - \alpha^k\right)^\tau$. Let us demand that this data-structure succeeds with probability $\geq 3/4$. To this end, we set

$$\tau = 4\left\lceil 1/\alpha^k \right\rceil \implies \phi \geq 1 - (1-\alpha^k)^\tau \geq 1 - \exp(-\alpha^k \tau) \geq 1 - \exp(-4) \geq 3/4, \tag{28.1}$$

since $1 - x \leq \exp(-x)$, for $x \geq 0$.

Lemma 28.3.7. *The expected number of points of $\mathsf{P} \setminus \mathbf{b}(q, R)$ colliding with the query point is $L = O(n(\beta/\alpha)^k)$, where $R = (1+\varepsilon)r$.*

Proof: Consider the points in $\mathsf{P} \setminus \mathbf{b}(q, R)$. We would like to bound the number of points of this set that collide with the query point. Observe that in this case, the probability of a point $\mathsf{p} \in \mathsf{P} \setminus \mathbf{b}(q, R)$ to collide with the query point is

$$\leq \psi = 1 - (1-\beta^k)^\tau = \beta^k\Big(1 + (1-\beta^k) + (1-\beta^k)^2 + \ldots + (1-\beta^k)^{\tau-1}\Big) \leq \beta^k \tau \leq 8\Big(\frac{\beta}{\alpha}\Big)^k,$$

as $\tau = 4\left\lceil 1/\alpha^k \right\rceil$ and $\alpha, \beta \in (0,1)$. Namely, the expected number of points of $\mathsf{P} \setminus \mathbf{b}(q, R)$ colliding with the query point is $\leq \psi n$. ■

By Lemma 28.3.6, extracting the $O(L)$ points takes $O(k\tau + L)$ time. Computing the distance of the query time for each one of these points takes $O(k\tau + Ld)$ time. As such, by Lemma 28.3.7, the query time is

$$O(k\tau + Ld) = O\Big(k\tau + nd(\beta/\alpha)^k\Big).$$

To minimize this query time, we "approximately" solve the equation requiring the two terms, in the above bound, to be equal (we ignore d since, intuitively, it should be small compared to n). We get that

$$k\tau = n(\beta/\alpha)^k \rightsquigarrow \frac{k}{\alpha^k} \approx n\frac{\beta^k}{\alpha^k} \implies k \approx n\beta^k \rightsquigarrow 1/\beta^k \approx n \implies k \approx \ln_{1/\beta} n.$$

Thus, setting $k = \ln_{1/\beta} n$, we have that $\beta^k = 1/n$ and, by Eq. (28.1), that

$$\tau = 4\left\lceil 1/\alpha^k \right\rceil = \exp\left(\frac{\ln n}{\ln 1/\beta} \ln 1/\alpha\right) = O(n^\rho), \quad \text{for } \rho = \frac{\ln 1/\alpha}{\ln 1/\beta}. \tag{28.2}$$

As such, to minimize the query time, we need to minimize ρ.

Lemma 28.3.8. *(A) For $x \in [0,1)$ and $t \geq 1$ such that $1 - tx > 0$ we have $\dfrac{\ln(1-x)}{\ln(1-tx)} \leq \dfrac{1}{t}$.*

(B) For $\alpha = 1 - r/d$ and $\beta = 1 - r(1+\varepsilon)/d$, we have that $\rho = \dfrac{\ln 1/\alpha}{\ln 1/\beta} \leq \dfrac{1}{1+\varepsilon}$.

Proof: (A) Since $\ln(1-tx) < 0$, it follows that the claim is equivalent to $t\ln(1-x) \geq \ln(1-tx)$. This in turn is equivalent to

$$g(x) \equiv (1-tx) - (1-x)^t \leq 0.$$

This is trivially true for $x = 0$. Furthermore, taking the derivative, we see $g'(x) = -t + t(1-x)^{t-1}$, which is non-positive for $x \in [0,1)$ and $t > 0$. Therefore, g is non-increasing in the interval of interest, and so $g(x) \leq 0$ for all values in this interval.

(B) Indeed $\rho = \dfrac{\ln 1/\alpha}{\ln 1/\beta} = \dfrac{\ln \alpha}{\ln \beta} = \dfrac{\ln \frac{d-r}{d}}{\ln \frac{d-(1+\varepsilon)r}{d}} = \dfrac{\ln\left(1-\frac{r}{d}\right)}{\ln\left(1-(1+\varepsilon)\frac{r}{d}\right)} \leq \dfrac{1}{1+\varepsilon}$, by part (A). ■

In the following, it would be convenient to consider d to be considerably larger than r. This can be ensured by (conceptually) padding the points with fake coordinates that are all zero. It is easy to verify that this "hack" would not affect the algorithm's performance in any way and it is just a trick to make our analysis simpler. In particular, we assume that $d > 2(1+\varepsilon)r$.

Lemma 28.3.9. *For $\alpha = 1 - r/d$, $\beta = 1 - r(1+\varepsilon)/d$, n and d as above, we have that I. $\tau = O\left(n^{1/(1+\varepsilon)}\right)$, II. $k = O(\ln n)$, and III. $L = O(n^{1/(1+\varepsilon)})$.*

Proof: By Eq. (28.1), $\tau = 4\left\lceil 1/\alpha^k \right\rceil = O(n^\rho) = O(n^{1/(1+\varepsilon)})$, by Lemma 28.3.8(B).

Now, $\beta = 1 - r(1+\varepsilon)/d \leq 1/2$, since we assumed that $d > 2(1+\varepsilon)r$. As such, we have $k = \ln_{1/\beta} n = \dfrac{\ln n}{\ln 1/\beta} = O(\ln n)$.

By Lemma 28.3.7, $L = O(n(\beta/\alpha)^k)$. Now $\beta^k = 1/n$ and as such $L = O(1/\alpha^k) = O(\tau) = O(n^{1/(1+\varepsilon)})$. ■

28.3.5. The result

Theorem 28.3.10. *Given a set P of n points on the hypercube $\mathcal{H}^d$ and parameters $\varepsilon > 0$ and $r > 0$, one can build a data-structure $\mathcal{D} = \mathcal{D}_{\approx Near}(\mathsf{P}, r, (1+\varepsilon)r)$ that solves the approximate near neighbor problem (see Definition 28.1.2). The data-structure answers a query successfully with high probability. In addition we have the following:*

(A) The query time is $O(dn^{1/(1+\varepsilon)} \log n)$.

(B) The preprocessing time to build this data-structure is $O(n^{1+1/(1+\varepsilon)} \log^2 n)$.

(C) The space required to store this data-structure is $O(nd + n^{1+1/(1+\varepsilon)} \log^2 n)$.

Proof: Our building block is the data-structure described above. By Markov's inequality, the probability that the algorithm has to abort because of too many collisions with points of $\mathsf{P} \setminus \mathbf{b}(q, (1+\varepsilon)r)$ is bounded by $1/4$ (since the algorithm tries $4L+1$ points). Also, if there is a point inside $\mathbf{b}(q, r)$, the algorithm would find it with probability $\geq 3/4$, by Eq. (28.1). As such, with probability at least $1/2$ this data-structure returns the correct answer in this case. By Lemma 28.3.6, the query time is $O(k\tau + Ld)$.

This data-structure succeeds only with constant probability. To achieve high probability, we construct $O(\log n)$ such data-structures and perform the near neighbor query in each one of them. As such, the query time is

$$O((k\tau + Ld)\log n) = O\left(n^{1/(1+\varepsilon)} \log^2 n + dn^{1/(1+\varepsilon)} \log n\right) = O\left(dn^{1/(1+\varepsilon)} \log n\right),$$

by Lemma 28.3.9 and since $d = \Omega(\lg n)$ if P contains n distinct points of $\mathcal{H}^d$.

As for the preprocessing time, by Lemma 28.3.6 and Lemma 28.3.9, we have

$$O(nk\tau \log n) = O(n^{1+1/(1+\varepsilon)} \log^2 n).$$

Finally, this data-structure requires $O(dn)$ space to store the input points. Specifically, by Lemma 28.3.6, we need an additional $O(nk\tau \log n) = O(n^{1+1/(1+\varepsilon)} \log^2 n)$ space. ■

In the hypercube case, when $d = n^{O(1)}$, we can build $M = O(\log_{1+\varepsilon} d) = O(\varepsilon^{-1} \log d)$ such data-structures such that $(1+\varepsilon)$-ANN can be answered using binary search on those data-structures which correspond to radii $r_1, \ldots, r_M$, where $r_i = (1+\varepsilon)^i$, for $i = 1, \ldots, M$.

Theorem 28.3.11. *Given a set P of n points on the hypercube $\mathcal{H}^d$ (where $d = n^{O(1)}$) and a parameter $\varepsilon > 0$, one can build a data-structure to answer approximate nearest neighbor queries (under the Hamming distance) using $O(dn + n^{1/(1+\varepsilon)}\varepsilon^{-1} \log^2 n \log d)$ space, such that given a query point q, one can return a $(1+\varepsilon)$-ANN in P (under the Hamming distance) in $O(dn^{1/(1+\varepsilon)} \log n \log(\varepsilon^{-1} \log d))$ time. The result returned is correct with high probability.*

Remark 28.3.12. The result of Theorem 28.3.11 needs to be oblivious to the queries used. Indeed, for any instantiation of the data-structure of Theorem 28.3.11 there exist query points for which it would fail.

In particular, formally, if we perform a sequence of ANN queries using such a data-structure, where the queries depend on earlier returned answers, then the guarantee of a high probability of success is no longer implied by the above analysis (it might hold because of some other reasons, naturally).

28.4. LSH and ANN in Euclidean space

28.4.1. Preliminaries

Lemma 28.4.1. *Let $X = (X_1, \ldots, X_d)$ be a vector of d independent variables which have normal distribution $\mathbf{N}$, and let $v = (v_1, \ldots, v_d) \in \mathbb{R}^d$. We have that $\langle v, X \rangle = \sum_i v_i X_i$ is distributed as $\|v\| Z$, where $Z \sim \mathbf{N}$.*

Proof: By Lemma 24.2.3$_{\text{p157}}$ the point X has multi-dimensional normal distribution $\mathbf{N}^d$. As such, if $\|v\| = 1$, then this holds by the symmetry of the normal distribution. Indeed, let $e_1 = (1, 0, \ldots, 0)$. By the symmetry of the d-dimensional normal distribution, we have that $\langle v, X \rangle \sim \langle e_1, X \rangle = X_1 \sim \mathbf{N}$.

Otherwise, $\langle v, X \rangle / \|v\| \sim \mathbf{N}$, and as such $\langle v, X \rangle \sim N(0, \|v\|^2)$, which is indeed the distribution of $\|v\| Z$. ■

Definition 28.4.2. A distribution $\mathcal{D}$ over $\mathbb{R}$ is called ***p-stable*** if there exists $p \geq 0$ such that for any n real numbers $v_1, \ldots, v_n$ and n independent variables $X_1, \ldots, X_n$ with distribution $\mathcal{D}$, the random variable $\sum_i v_i X_i$ has the same distribution as the variable $(\sum_i |v_i|^p)^{1/p} X$, where X is a random variable with distribution $\mathcal{D}$.

By Lemma 28.4.1, the normal distribution is a ***2-stable distribution***.

28.4.2. Locality sensitive hashing (LSH)

Let p and u be two points in $\mathbb{R}^d$. We want to perform an experiment to decide if $\|\mathsf{p} - \mathsf{u}\| \leq 1$ or $\|\mathsf{p} - \mathsf{u}\| \geq \eta$, where $\eta = 1 + \varepsilon$. We will randomly choose a vector $\mathbf{v}$ from the d-dimensional normal distribution $\mathbf{N}^d$ (which is 2-stable). Next, let r be a parameter, and let t be a random number chosen uniformly from the interval $[0, r]$. For $\mathsf{p} \in \mathbb{R}^d$, consider the random hash function

$$h(\mathsf{p}) = \left\lfloor \frac{\langle \mathsf{p}, \mathbf{v} \rangle + t}{r} \right\rfloor. \tag{28.3}$$

Assume that the distance between p and u is η and the distance between the projection of the two points to the direction $\mathbf{v}$ is β. Then, the probability that p and u get the same hash value is $\max(1-\beta/r, 0)$, since this is the probability that the random sliding will not separate them. Indeed, consider the line through $\mathbf{v}$ to be the x-axis, and assume u is projected to r and v is projected to $r-\beta$ (assuming $r \geq \beta$). Clearly, u and v get mapped to the same value by $h(\cdot)$ if and only if $t \in [0, r-\beta]$, as claimed.

As such, we have that the probability of collusion is

$$\alpha(\eta, r) = \mathbb{P}[h(\mathsf{p}) = h(\mathsf{q})] = \int_{\beta=0}^{r} \mathbb{P}[|\langle \mathsf{p}, \mathbf{v}\rangle - \langle \mathsf{u}, \mathbf{v}\rangle| = \beta]\left(1 - \frac{\beta}{r}\right) d\beta.$$

However, since $\mathbf{v}$ is chosen from a 2-stable distribution, we have that

$$Z = \langle \mathsf{p}, \mathbf{v}\rangle - \langle \mathsf{u}, \mathbf{v}\rangle = \langle \mathsf{p} - \mathsf{u}, \mathbf{v}\rangle \sim \mathbf{N}(0, \|\mathsf{p} - \mathsf{u}\|^2).$$

Since we are considering the absolute value of the variable, we need to multiply this by two. Thus, we have

$$\alpha(\eta, r) = \int_{\beta=0}^{r} \frac{2}{\sqrt{2\pi}\eta} \exp\left(-\frac{\beta^2}{2\eta^2}\right)\left(1 - \frac{\beta}{r}\right) d\beta,$$

by plugging in the density of the normal distribution for Z. Intuitively, we care about the difference $\alpha(1+\varepsilon, r) - \alpha(1, r)$, and we would like to maximize it as much as possible (by choosing the right value of r). Unfortunately, this integral is unfriendly, and we have to resort to numerical computation.

Now, we are going to use this hashing scheme for constructing locality sensitive hashing, as in the hypercube case, and as such we care about the ratio

$$\rho(1+\varepsilon) = \min_r \frac{\log(1/\alpha(1, r))}{\log(1/\alpha(1+\varepsilon, r))};$$

see Eq. (28.2). The following is verified using numerical calculations.

Lemma 28.4.3 ([DIIM04]). *One can choose r, such that $\rho(1+\varepsilon) \leq \frac{1}{1+\varepsilon}$.*

Lemma 28.4.3 implies that the hash functions defined by Eq. (28.3) are $(1, 1+\varepsilon, \alpha', \beta')$-sensitive and, furthermore, $\rho = \frac{\log(1/\alpha')}{\log(1/\beta')} \leq \frac{1}{1+\varepsilon}$, for some values of α' and β'. As such, we can use this hashing family to construct an approximate near neighbor data-structure $\mathcal{D}_{\approx \mathrm{Near}}(\mathsf{P}, r, (1+\varepsilon)r)$ for the set P of points in $\mathbb{R}^d$. Following the same argumentation of Theorem 28.3.10, we have the following.

Theorem 28.4.4. *Given a set P of n points in $\mathbb{R}^d$ and parameters $\varepsilon > 0$ and $r > 0$, one can build a $\mathcal{D}_{\approx Near} = \mathcal{D}_{\approx Near}(\mathsf{P}, r, (1+\varepsilon)r)$, such that given a query point q, one can decide:*

(A) If $\mathbf{b}(q, r) \cap \mathsf{P} \neq \emptyset$, then $\mathcal{D}_{\approx Near}$ returns a point $u \in \mathsf{P}$, such that $\mathsf{d}_H(u, q) \leq (1+\varepsilon)r$.

(B) If $\mathbf{b}(q, (1+\varepsilon)r) \cap \mathsf{P} = \emptyset$, then $\mathcal{D}_{\approx Near}$ returns the result that no point is within distance $\leq r$ from q.

In any other case, any of the answers is correct. The query time is $O(dn^{1/(1+\varepsilon)} \log n)$ and the space used is $O\left(dn + n^{1+1/(1+\varepsilon)} \log n\right)$. The result returned is correct with high probability.

28.4.3. ANN in high-dimensional euclidean space

Unlike the binary hypercube case, where we could just do direct binary search on the distances, here we need to use the reduction from ANN to near neighbor queries.

28.4.3.1. The result

Plugging the above into known reduction from approximate nearest-neighbor to near-neighbor queries, yields the following:

Corollary 28.4.5. *Given a set* P *of* n *points in* $\mathbb{R}^d$*, one can construct a data-structure* $\mathcal{D}$ *that answers* $(1+\varepsilon)$*-*ANN *queries, by performing* $O(\log(n/\varepsilon))$ $(1+\varepsilon)$*-approximate near neighbor queries. The total number of points stored at these approximate near neighbor data-structure of* $\mathcal{D}$ *is* $O(n\varepsilon^{-1}\log(n/\varepsilon))$*.*

This in turn leads to the following:

Theorem 28.4.6. *Given a set* P *of* n *points in* $\mathbb{R}^d$ *and parameters* $\varepsilon > 0$ *and* $r > 0$*, one can build an* ANN *data-structure using*

$$O\Big(dn + n^{1+1/(1+\varepsilon)}\varepsilon^{-2}\log^3(n/\varepsilon)\Big)$$

space, such that given a query point q*, one can returns a* $(1+\varepsilon)$*-*ANN *in* P *in*

$$O\Big(dn^{1/(1+\varepsilon)}(\log n)\log\frac{n}{\varepsilon}\Big)$$

time. The result returned is correct with high probability.

The construction time is $O\Big(dn^{1+1/(1+\varepsilon)}\varepsilon^{-2}\log^3(n/\varepsilon)\Big)$*.*

28.5. Bibliographical notes

Section 28.1 follows the exposition of Indyk and Motwani [IM98]. Kushilevitz *et al.* [KOR00] offered an alternative data-structure with somewhat inferior performance. It is quite surprising that one can perform approximate nearest neighbor queries in high dimensions in time and space polynomial in the dimension (which is sublinear in the number of points). One can reduce the approximate near neighbor in Euclidean space to the same question on the hypercube "directly" (we show the details below). However, doing the LSH directly on the Euclidean space is more efficient.

The value of the results shown in this chapter depends to a large extent on the reader's perspective. Indeed, for a small value of $\varepsilon > 0$, the query time $O(dn^{1/(1+\varepsilon)})$ is very close to linear dependency on n and is almost equivalent to just scanning the points. Thus, from the low-dimensional perspective, where ε is assumed to be small, this result is slightly sublinear. On the other hand, if one is willing to pick ε to be large (say 10), then the result is clearly better than the naive algorithm, suggesting running time for an ANN query which takes (roughly) $O\big(n^{1/11}\big)$ time.

The idea of doing locality sensitive hashing directly on the Euclidean space, as done in Section 28.4, is not shocking after seeing the Johnson-Lindenstrauss lemma. Our description follows the paper of Datar *et al.* [DIIM04]. In particular, the current analysis which relies on computerized estimates is far from being satisfactory. It would be nice to have a simpler and more elegant scheme for this case. This is an open problem for further research.

Currently, the best LSH construction in $\mathbb{R}^d$ is due to Andoni and Indyk [AI06]. Its space usage is bounded by $O\big(dn + n^{1+1/(1+\varepsilon)^2+o(1)}\big)$ and its query time is bounded by $O\big(dn^{1/(1+\varepsilon)^2+o(1)}\big)$. This (almost) matches the lower bound of Motwani *et al.* [MNP06]. For a nice survey on LSH see [AI08].

From approximate near neighbor in $\mathbb{R}^d$ to approximate near neighbor on the hypercube.

The reduction is quite involved, and we only sketch the details. Let P be a set of n points in $\mathbb{R}^d$. We first reduce the dimension to $k = O(\varepsilon^{-2}\log n)$ using the Johnson-Lindenstrauss lemma. Next, we embed this

space into $\ell_1^{k'}$ (this is the space $\mathbb{R}^k$, where distances are the L_1 metric instead of the regular L_2 metric), where $k' = O(k/\varepsilon^2)$. This can be done with distortion $(1+\varepsilon)$.

Let Q' be the resulting set of points in $\mathbb{R}^{k'}$. We want to solve approximate near neighbor queries on this set of points, for radius r. As a first step, we partition the space into cells by taking a grid with sidelength (say) $k'r$ and randomly translating it, clipping the points inside each grid cell. It is now sufficient to solve the approximate near neighbor problem inside this grid cell (which has bounded diameter as a function of r), since with small probability the result would be correct. We amplify the probability by repeating this a polylogarithmic number of times.

Thus, we can assume that P is contained inside a cube of sidelength $\leq k'nr$, it is in $\mathbb{R}^{k'}$, and the distance metric is the L_1 metric. We next snap the points of P to a grid of sidelength (say) $\varepsilon r/k'$. Thus, every point of P now has an integer coordinate, which is bounded by a polynomial in $\log n$ and $1/\varepsilon$. Next, we write the coordinates of the points of P using unary notation. (Thus, a point $(2, 5)$ would be written as $(00011, 11111)$ assuming the number of bits for each coordinates is 5.) It is now easy to verify that the Hamming distance on the resulting strings is equivalent to the L_1 distance between the points.

Thus, we can solve the near neighbor problem for points in $\mathbb{R}^d$ by solving it on the hypercube under the Hamming distance.

See Indyk and Motwani [IM98] for more details.

We have only scratched the surface of proximity problems in high dimensions. The interested reader is referred to the survey by Indyk [Aga04] for more information.

References

[Aga04] P. K. Agarwal. Range searching. *Handbook of Discrete and Computational Geometry*. Ed. by J. E. Goodman and J. O'Rourke. 2nd. Boca Raton, FL, USA: CRC Press LLC, 2004. Chap. 36, pp. 809–838.

[AI06] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Proc. 47th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), 459–468, 2006.

[AI08] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1): 117–122, 2008.

[DIIM04] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. *Proc. 20th Annu. Sympos. Comput. Geom.* (SoCG), 253–262, 2004.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. *Proc. 30th Annu. ACM Sympos. Theory Comput.* (STOC), 604–613, 1998.

[KOR00] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 2(30): 457–474, 2000.

[MNP06] R. Motwani, A. Naor, and R. Panigrahi. Lower bounds on locality sensitive hashing. *Proc. 22nd Annu. Sympos. Comput. Geom.* (SoCG), 154–157, 2006.

Chapter 29

Random Walks I

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

"A drunk man will find his way home; a drunk bird may wander forever."

Anonymous,

29.1. Definitions

Let $\mathsf{G} = \mathsf{G}(\mathsf{V}, \mathsf{E})$ be an undirected connected graph. For $v \in \mathsf{V}$, let $\Gamma(v)$ denote the set of neighbors of v in G; that is, $\Gamma(v) = \left\{u \;\middle|\; vu \in \mathsf{E}(\mathsf{G})\right\}$. A ***random walk*** on G is the following process: Starting from a vertex v_0, we randomly choose one of the neighbors of v_0, and set it to be v_1. We continue in this fashion, in the ith step choosing v_i, such that $v_i \in \Gamma(v_{i-1})$. It would be interesting to investigate the random walk process. Questions of interest include:

(A) How long does it take to arrive from a vertex v to a vertex u in G?
(B) How long does it take to visit all the vertices in the graph.
(C) If we start from an arbitrary vertex v_0, how long the random walk has to be such that the location of the random walk in the ith step is uniformly (or near uniformly) distributed on $\mathsf{V}(\mathsf{G})$?

Example 29.1.1. In the complete graph K_n, visiting all the vertices takes in expectation $O(n \log n)$ time, as this is the coupon collector problem with $n-1$ coupons. Indeed, the probability we did not visit a specific vertex v by the ith step of the random walk is $\leq (1-1/n)^{i-1} \leq e^{-(i-1)/n} \leq 1/n^{10}$, for $i = \Omega(n \log n)$. As such, with high probability, the random walk visited all the vertex of K_n. Similarly, arriving from u to v, takes in expectation $n-1$ steps of a random walk, as the probability of visiting v at every step of the walk is $p = 1/(n-1)$, and the length of the walk till we visit v is a geometric random variable with expectation $1/p$.

29.1.1. Walking on grids and lines

Lemma 29.1.2 (Stirling's formula). *For any integer $n \geq 1$, it holds $n! \approx \sqrt{2\pi n}\,(n/e)^n$.*

29.1.1.1. Walking on the line

Lemma 29.1.3. *Consider the infinite random walk on the integer line, starting from 0. Here, the vertices are the integer numbers, and from a vertex k, one walks with probability $1/2$ either to $k-1$ or $k+1$. The expected number of times that such a walk visits 0 is unbounded.*

Proof: The probability that in the $2i$th step we visit 0 is $\frac{1}{2^{2i}}\binom{2i}{i}$, As such, the expected number of times we visit the origin is

$$\sum_{i=1}^{\infty} \frac{1}{2^{2i}}\binom{2i}{i} \geq \sum_{i=1}^{\infty} \frac{1}{2\sqrt{i}} = \infty,$$

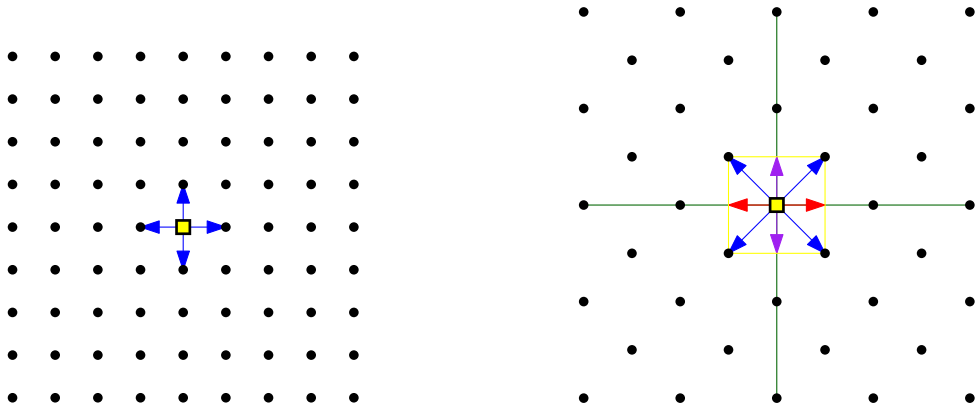


Figure 29.1: A walk in the integer grid, when rotated by 45 degrees, results, in two independent walks on one dimension.

since $\frac{2^{2i}}{2\sqrt{i}} \leq \binom{2i}{i} \leq \frac{2^{2i}}{\sqrt{2i}}$ [MN98, p. 84]. This can also be verified using the Stirling formula, and the resulting sequence diverges. ■

29.1.1.2. Walking on two dimensional grid

A random walk on the integer grid $\mathbb{Z}^d$, starts from a point of this integer grid, and at each step if it is at point $(i_1, i_2, \ldots, i_d)$, it chooses a coordinate and either increases it by one, or decreases it by one, with equal probability.

Lemma 29.1.4. *Consider the infinite random walk on the two dimensional integer grid $\mathbb{Z}^2$, starting from $(0,0)$. The expected number of times that such a walk visits the origin is unbounded.*

Proof: Rotate the grid by 45 degrees, and consider the two new axes X' and Y', see Figure 29.1.. Let x_i be the projection of the location of the ith step of the random walk on the X'-axis, and define y_i in a similar fashion. Clearly, x_i are of the form $j/\sqrt{2}$, where j is an integer. By scaling by a factor of $\sqrt{2}$, consider the resulting random walks $x'_i = \sqrt{2}x_i$ and $y'_i = \sqrt{2}y_i$. Clearly, x_i and y_i are random walks on the integer grid, and furthermore, they are *independent*. As such, the probability that we visit the origin at the $2i$th step is $\mathbb{P}\left[x'_{2i} = 0 \cap y'_{2i} = 0\right] = \mathbb{P}\left[x'_{2i} = 0\right]^2 = \left(\frac{1}{2^{2i}}\binom{2i}{i}\right)^2 \geq 1/4i$. We conclude, that the infinite random walk on the grid $\mathbb{Z}^2$ visits the origin in expectation

$$\sum_{i=0}^{\infty} \mathbb{P}[x'_i = 0 \cap y'_i = 0] \geq \sum_{i=0}^{\infty} \frac{1}{4i} = \infty,$$

as this sequence diverges. ■

29.1.1.3. Walking on three dimensional grid

In the following, let $\binom{i}{a\ b\ c} = \frac{i!}{a!\,b!\,c!}$ denote the multinomial coefficient. The multinomial theorem states that

$$(x_1 + x_2 + \cdots + x_m)^n = \sum_{k_1+k_2+\cdots+k_m=n\ k_1,k_2,\cdots,k_m \geq 0} \binom{n}{k_1, k_2, \ldots, k_m} \prod_{t=1}^{m} x_t^{k_t}.$$

In particular, we have

$$1^n = (1/3 + 1/3 + 1/3)^n = \sum_{a+b+c=n, a,b,c\geq 0} \binom{n}{a\ b\ c} \frac{1}{3^n}. \tag{29.1}$$

Lemma 29.1.5. *Consider the infinite random walk on the three dimensional integer grid $\mathbb{Z}^3$, starting from $(0,0,0)$. The expected number of times that such a walk visits the origin is bounded.*

Proof: The probability of a neighbor of a point (x,y,z) to be the next point in the walk is $1/6$. Assume that we performed a walk for $2i$ steps, and decided to perform $2a$ steps parallel to the x-axis, $2b$ steps parallel to the y-axis, and $2c$ steps parallel to the z-axis, where $a+b+c=i$. Furthermore, the walk on each dimension is balanced, that is we perform a steps to the left on the x-axis, and a steps to the right on the x-axis. Clearly, this corresponds to the only walks in $2i$ steps that arrives to the origin.

Next, the number of different ways we can perform such a walk is $\frac{(2i)!}{a!a!b!b!c!c!}$, and the probability to perform such a walk, summing over all possible values of a, b and c, is

$$\alpha_i = \sum_{\substack{a+b+c=i\\ a,b,c\geq 0}} \frac{(2i)!}{a!a!b!b!c!c!} \frac{1}{6^{2i}} = \binom{2i}{i} \frac{1}{2^{2i}} \sum_{\substack{a+b+c=i\\ a,b,c\geq 0}} \left(\frac{i!}{a!\,b!\,c!}\right)^2 \left(\frac{1}{3}\right)^{2i} = \binom{2i}{i} \frac{1}{2^{2i}} \sum_{\substack{a+b+c=i\\ a,b,c\geq 0}} \left(\binom{i}{a\ b\ c} \left(\frac{1}{3}\right)^i\right)^2$$

Consider the case where $i = 3m$. We have that $\binom{i}{a\ b\ c} \leq \binom{i}{m\ m\ m}$. As such, we have

$$\alpha_i \leq \binom{2i}{i} \frac{1}{2^{2i}} \left(\frac{1}{3}\right)^i \binom{i}{m\ m\ m} \underbrace{\sum_{\substack{a+b+c=i\\ a,b,c\geq 0}} \binom{i}{a\ b\ c} \left(\frac{1}{3}\right)^i}_{=1 \text{ by Eq. (29.1)}}.$$

By the Stirling formula, we have

$$\binom{i}{m\ m\ m} \approx \frac{\sqrt{2\pi i}(i/e)^i}{\left(\sqrt{2\pi i/3}\left(\frac{i}{3e}\right)^{i/3}\right)^3} = c\frac{3^i}{i},$$

for some constant c. As such, $\alpha_i = O\left(\frac{1}{\sqrt{i}}\left(\frac{1}{3}\right)^i \frac{3^i}{i}\right) = O\left(\frac{1}{i^{3/2}}\right)$. Thus,

$$\sum_{m=1}^{\infty} \alpha_{6m} = \sum_i O\left(\frac{1}{i^{3/2}}\right) = O(1).$$

Finally, observe that $\alpha_{6m} \geq (1/6)^2 \alpha_{6m-2}$ and $\alpha_{6m} \geq (1/6)^4 \alpha_{6m-4}$. Thus,

$$\sum_{m=1}^{\infty} \alpha_m = O(1).$$

■

29.2. Bibliographical notes

The presentation here follows [Nor98].

References

[MN98] J. Matoušek and J. Nešetřil. *Invitation to Discrete Mathematics*. Oxford Univ Press, 1998.

[Nor98] J. R. Norris. *Markov Chains*. Statistical and Probabilistic Mathematics. Cambridge Press, 1998.

Chapter 30

Random Walks II

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "Then you must begin a reading program immediately so that you man understand the crises of our age," Ignatius said solemnly. "Begin with the late Romans, including Boethius, of course. Then you should dip rather extensively into early Medieval. You may skip the Renaissance and the Enlightenment. That is mostly dangerous propaganda. Now, that I think about of it, you had better skip the Romantics and the Victorians, too. For the contemporary period, you should study some selected comic books."
> "You're fantastic."
> "I recommend Batman especially, for he tends to transcend the abysmal society in which he's found himself. His morality is rather rigid, also. I rather respect Batman."
>
> John Kennedy Toole, A confederacy of Dunces

30.1. Catalan numbers

For a sequence σ of symbols, let $\#(\sigma, X)$ be the number of times the symbol X appears in σ.

Definition 30.1.1. A sequence/word σ of length $2n$ elements/characters made out of two symbols X and Y, is ***balanced***, if

(I) X appears n times (i.e., $\#(\sigma, X) = n$),
(II) Y appears n times (i.e., $\#(\sigma, Y) = n$),
(III) In any prefix of the string, the number of Xs is at least as large as the number of Ys.

Such a string is known as a ***Dyck word***. If X and Y are the open and close parenthesis characters, respectively, then the word is a balanced/valid parenthesis pattern.

Definition 30.1.2. The ***Catalan number***, denoted by C_n, is the number of balanced strings of length $2n$.

There are many other equivalent definitions of Catalan number.

Definition 30.1.3. A sequence σ made out of two symbols X and Y is ***dominating***, if for any non-empty prefix of σ, the number of Xs is strictly larger than the number of Ys.

Lemma 30.1.4. *Let σ be a cyclic sequence made out symbols X and Y, where $n = \#(\sigma, X)$ and $m = \#(\sigma, y)$, with $n > m$. Then there are exactly $n - m$ locations where cutting the cyclic sequence at these locations, results in a dominating sequence.*

Proof: Consider a location in σ that contains X, and the next location contains Y. Clearly, such a location can not be a start for a dominating sequence. Of course, the next location can also not be a start position for a dominating sequence. As such, these two locations must be interior to a dominating sequence, and deleting both of these symbols from σ, results in a new cyclic sequence, where every dominating start location corresponds to a dominating start location in the original sequence. Observe, that as long as the number of Xs is larger than the number of Ys, there must be such a location with XY as the prefix. Repeatedly deleting XY substring, results in a string of length $n - m$, where every location is a good start of a dominating sequence. We conclude that there are exactly $n - m$ such locations. ■

Observation 30.1.5. *The number of distinct cyclic sequences of length $m + n$, with m appearances of X, and n appearances of Y is $\frac{(n+m-1)!}{m!n!} = \frac{1}{n+m}\binom{n+m}{n}$, since there are $(n + m - 1)!$ different cyclic ways to arrange $n + m$ distinct values.*

Theorem 30.1.6. *For $n \geq 1$, we have that the Catalan number $C_n = \frac{1}{n+1}\binom{2n}{n}$.*

Proof: Consider a dominating sequence σ of length $2n + 1$ with $\#(\sigma, X) = n + 1$, and $\#(\sigma, Y) = n$. Such a sequence must start with an X, and if we remove the leading X, then what remains is a balanced sequence. Such a sequence σ can be interpreted as a cyclic sequence. By the above lemma, there is a unique shift that is dominating. As such, the number of such cyclic sequence is the Catalan number C_n. By the above observation, the number of such cyclic sequences is

$$\frac{(n+m-1)!}{m!n!} = \frac{(n+n+1-1)!}{(n+1)!n!} = \frac{1}{n+1}\frac{2n!}{n!n!} = \frac{1}{n+1}\binom{2n}{n}.$$

■

30.2. Walking on the integer line revisited

30.2.1. Estimating the middle binomial coefficient

Lemma 30.2.1. *For $i \geq 11^2$, we have $\frac{1}{4} \cdot \frac{2^{2i}}{\sqrt{i}} \leq \binom{2i}{i} \leq 2 \cdot \frac{2^{2i}}{\sqrt{i}}$. and $\binom{2i}{i+\sqrt{i}} \geq \frac{1}{12} \cdot \frac{2^{2i}}{\sqrt{i}}$*

Proof: Observe that $\binom{2i}{i} \geq \binom{2i}{j}$, for any j. We assume that $i \geq 11^2$, and $\sqrt{i}$ is an integer. O bserve that $\binom{2i}{i+\tau} = \frac{2i!}{(i+\tau)!(i-\tau)!} = \frac{2i!}{i!i!}\frac{(i-\tau+1)\cdots(i-1)i}{(i+1)\cdots(i+\tau)} = \binom{2i}{i}\prod_{k=1}^{\tau}\frac{i-\tau+k}{i+k}$. Now, by Lemma 10.1.3, we have

$$\alpha = \prod_{k=1}^{\tau}\frac{i-\tau+k}{i+k} = \prod_{k=1}^{\tau}\left(1 - \frac{\tau}{i+k}\right) \geq \left(1 - \frac{\tau}{i}\right)^{\tau} \geq \left(1 - \frac{\tau^2}{i^2}\tau\right)\exp\left(-\frac{\tau^2}{i}\right) \geq \frac{1}{3},$$

for $\tau \leq \sqrt{i}$, and $i \geq 11^2$. Namely, for any k, such that $-\sqrt{i} \leq k \leq \sqrt{i}$, we have $\binom{2i}{i+k} \geq \binom{2i}{i}/3$. We thus have that

$$1 \geq \frac{1}{2^{2i}}\sum_{k=-\sqrt{i}+1}^{\sqrt{i}}\binom{2i}{i+k} \geq \frac{2\sqrt{i}}{3 \cdot 2^{2i}}\binom{2i}{i} \qquad \Longrightarrow \qquad \binom{2i}{i} \leq \frac{2}{3} \cdot \frac{2^{2i}}{\sqrt{i}}.$$

Let $\Delta = \sqrt{i} - 1$ and $X \sim \mathrm{bin}(2i, 1/2)$. We have that $\mathbb{E}[X] = i$, and $\mathbb{V}[X] = 2i(1/2)(1/2) = i/2$. Let $\beta = \frac{1}{2^{2i}}\sum_{k=-\Delta}^{\Delta}\binom{2i}{i+k}$. By Chebychev, we have that $1 - \beta = \mathbb{P}\left[|X - i| \geq \sqrt{2}\sqrt{i/2}\right] \leq 1/2$. which implies $\beta \geq 1/2$. We have

$$\frac{1}{2} \leq \beta \leq \frac{1}{2^{2i}}\sum_{k=-\Delta}^{\Delta}\binom{2i}{i+k} \leq \frac{2\Delta+1}{2^{2i}}\binom{2i}{i} \qquad \Longrightarrow \qquad \binom{2i}{i} \geq \frac{2^{2i}}{2(2\Delta+1)} \geq \frac{2^{2i}}{4\sqrt{i}}.$$

■

Lemma 30.2.2. *In a random walk on the line starting at zero, in expectation, after $48n^2$ steps, the walk had visited either $-n$ or $+n$.*

Proof: By Lemma 30.2.1, the probability that after $2i$ steps, for $i = 16n^2$, the walk is in the range $\{-\sqrt{i}+1, \ldots, \sqrt{i}-1\}$ is at most

$$2n\frac{1}{2^{2i}} \cdot \frac{2}{3} \cdot \frac{2^{2i}}{\sqrt{i}} = 2n\frac{2}{3} \cdot \frac{1}{4n} = \frac{1}{3}.$$

Namely, the walk arrived to either $-n$ or $+n$ during the first $32n^2$ steps (note that $n \leq i/2$) with probability $\geq 2/3$. If this did not happen, we continue the walk. As $i \geq 2n$, the same argumentation essentially implies that every $32n^2$ steps, the walk terminates with probability at least $2/3$. As such, in expectation, after $3/2$ such epochs, the walk would terminate. ■

30.3. Solving 2SAT using random walk

Let $\mathsf{G} = G(V, E)$ be a undirected connected graph. For $v \in V$, let $\Gamma(v)$ denote the neighbors of v in G. A random walk on G is the following process: Starting from a vertex v_0, we randomly choose one of the neighbors of v_0, and set it to be v_1. We continue in this fashion, such that $v_i \in \Gamma(v_{i-1})$. It would be interesting to investigate the process of the random walk. For example, questions like: (i) how long does it take to arrive from a vertex v to a vertex u in G? and (ii) how long does it take to visit all the vertices in the graph.

30.3.1. Solving 2SAT

Consider a 2SAT formula F with m clauses defined over n variables. Start from an arbitrary assignment to the variables, and consider a non-satisfied clause in F. Randomly pick one of the clause variables, and change its value. Repeat this till you arrive to a satisfying assignment.

Consider the random variable X_i, which is the number of variables assigned the correct value (according to the satisfying assignment) in the current assignment. Clearly, with probability (at least) half $X_i = X_{i-1} + 1$.

Thus, we can think about this algorithm as performing a random walk on the numbers $0, 1, \ldots, n$, where at each step, we go to the right probability at least half. The question is, how long does it take to arrive to n in such a settings.

Theorem 30.3.1. *The expected number of steps to arrive to a satisfying assignment is $O(n^2)$.*

Proof: For simplicity of exposition assume that n is divisible by 4. Consider the random walk on the integer line, starting from zero, where we go to the left with probability $1/2$, and to the right probability $1/2$. Let Y_i be the location of the walk at the i step. Clearly, $\mathbb{E}[Y_i] \geq \mathbb{E}[X_i]$. By defining the random walk on the integer line more carefully, one can ensure that $Y_i \leq X_i$. Thus, the expected number of steps till Y_i is equal to n is an upper bound on the required quantity.

For an i, Y_{2i} is an even number. Thus, consider the event that $Y_{2i} = 2\Delta \geq n$, let $Y_{2i} = R_{2i} - L_{2i}$, where R_{2i} is the number of steps to the right, and L_{2i} is the number of steps to the left. Observe that

$$\begin{cases} R_{2i} - L_{2i} = 2\Delta \\ R_{2i} + L_{2i} = 2i \end{cases} \implies \begin{cases} R_{2i} & = i + \Delta \\ L_{2i} = i - R_{2i} & = i - \Delta. \end{cases}$$

Thus, for $i \geq n/2$, we have that the probability that in the $2i$th step we have $Y_{2i} \geq n$ is

$$\rho = \sum_{\Delta=n/2}^{i} \frac{1}{2^{2i}} \binom{2i}{i+\Delta}.$$

Lemma 30.3.2 below, tells us that for $\rho > 1/3$, is implied if $\Delta \leq \sqrt{i}/6$. That is, $n/2 \leq \sqrt{i}/6$, which holds for $i = 9n^2$.

Next, if X_{2i} fails to arrive to n at the first μ steps, we will reset $Y_\mu = X_\mu$ and continue the random walk, repeating this process as many phases as necessary. The probability that the number of phases exceeds i is $\leq (2/3)^i$. As such, the expected number of steps in the walk is at most

$$\sum_i c' n^2 i (1-\rho)^i = O(n^2),$$

as claimed. ■

Lemma 30.3.2. *We have* $\sum_{k=i+\sqrt{i}/6}^{2i} \frac{1}{2^{2i}} \binom{2i}{k} \geq \frac{1}{3}$.

Proof: It is known[①] that $\binom{2i}{i} \leq 2^{2i}/\sqrt{i}$ (better constants are known). As such, since $\binom{2i}{i} \geq \binom{2i}{m}$, for all m, we have by symmetry that

$$\sum_{k=i+\sqrt{i}/6}^{2i} \frac{1}{2^{2i}} \binom{2i}{k} \geq \sum_{k=i+1}^{2i} \frac{1}{2^{2i}} \binom{2i}{k} - \sqrt{i}/6 \frac{1}{2^{2i}} \binom{2i}{i} \geq \frac{1}{2} - \sqrt{i}/6 \frac{1}{2^{2i}} \cdot \frac{2^{2i}}{\sqrt{i}} = \frac{1}{3}.$$

■

30.4. Markov chains

Let S denote a state space, which is either finite or countable. A ***Markov chain*** is at one state at any given time. There is a ***transition probability*** P_{ij}, which is the probability to move to the state j, if the Markov chain is currently at state i. As such, $\sum_j \mathsf{P}_{ij} = 1$ and $\forall i, j$ we have $0 \leq \mathsf{P}_{ij} \leq 1$. The matrix $\mathbf{P} = \left\{\mathsf{P}_{ij}\right\}_{ij}$ is the ***transition probabilities matrix***.

$$\mathbf{P} = \left(\begin{array}{ccc} & j\text{th column} & \\ i\text{th row} & \mathsf{P}_{ij} & \\ & & \end{array} \right)$$

The Markov chain start at an initial state X_0, and at each point in time moves according to the transition probabilities. This form a sequence of states $\{X_t\}$. We have a distribution over those sequences. Such a sequence would be referred to as a ***history***.

Similar to Martingales, the behavior of a Markov chain in the future, depends only on its location X_t at time t, and does not depends on the earlier stages that the Markov chain went through. This is the ***memorylessness property*** of the Markov chain, and it follows as P_{ij} is independent of time. Formally, the memorylessness property is

$$\mathbb{P}[X_{t+1} = j \mid X_0 = i_0, X_1 = i_1, \ldots, X_{t-1} = i_{t-1}, X_t = i] = \mathbb{P}[X_{t+1} = j \mid X_t = i] = \mathsf{P}_{ij}.$$

The initial state of the Markov chain might also be chosen randomly in some cases.

[①] Probably because you got it as a homework problem, if not wikipedia knows, and if you are bored you can try and prove it yourself.

For states $i, j \in \mathsf{S}$, the ***t-step transition probability*** is $\mathtt{P}_{ij}^{(t)} = \mathbb{P}\Big[X_t = j \;\Big|\; X_0 = i\Big]$. The probability that we visit j for the first time, starting from i after t steps, is denoted by

$$\mathtt{r}_{ij}^{(t)} = \mathbb{P}\Big[X_t = j \text{ and } X_1 \neq j, X_2 \neq j, \ldots, X_{t-1} \neq j \;\Big|\; X_0 = i\Big].$$

Let $\mathtt{f}_{ij} = \sum_{t>0} \mathtt{r}_{ij}^{(t)}$ denote the probability that the Markov chain visits state j, at any point in time, starting from state i. The expected number of steps to arrive to state j starting from i is

$$\mathtt{h}_{ij} = \sum_{t>0} t \cdot \mathtt{r}_{ij}^{(t)}.$$

Of course, if $\mathtt{f}_{ij} < 1$, then there is a positive probability that the Markov chain never arrives to j, and as such $\mathtt{h}_{ij} = \infty$ in this case.

Definition 30.4.1. A state $i \in \mathsf{S}$ for which $\mathtt{f}_{ii} < 1$ (i.e., the chain has positive probability of never visiting i again), is a ***transient*** state. If $\mathtt{f}_{ii} = 1$ then the state is ***persistent***.

A state i that is persistent but $\mathtt{h}_{ii} = \infty$ is ***null persistent***. A state i that is persistent and $\mathtt{h}_{ii} \neq \infty$ is ***non null persistent***.

Example 30.4.2. Consider the state 0 in the random walk on the integers. We already know that in expectation the random walk visits the origin infinite number of times, so this hints that this is a persistent state. Let figure out the probability $\mathtt{r}_{00}^{(2n)}$. To this end, consider a walk $X_0, X_1, \ldots, X_{2n}$ that starts at 0 and return to 0 only in the $2n$ step. Let $S_i = X_i - X_{i-1}$, for all i. Clearly, we have $S_i \in \{-1, +1\}$ (i.e., move left or move right). Assume the walk starts by $S_1 = +1$ (the case -1 is handled similarly). Clearly, the walk $S_2, \ldots, S_{2n-1}$ must be prefix balanced; that is, the number of 1s is always bigger (or equal) for any prefix of this sequence.

Strings with this property are known as ***Dyck words***, and the number of such words of length $2m$ is the ***Catalan number*** $C_m = \frac{1}{m+1}\binom{2m}{m}$. As such, the probability of the random walk to visit 0 for the first time (starting from 0) after $2n$ steps, is

$$\mathtt{r}_{00}^{(2n)} = 2\frac{1}{n}\binom{2n-2}{n-1}\frac{1}{2^{2n}} = \Theta\left(\frac{1}{n} \cdot \frac{1}{\sqrt{n}}\right) = \Theta\left(\frac{1}{n^{3/2}}\right).$$

(the 2 here is because the other option is that the sequence starts with -1), using that $\binom{2n}{n} = \Theta\left(2^{2n}/\sqrt{n}\right)$.

Observe that $\mathtt{f}_{00} = \sum_{n=0}^{\infty} \mathtt{r}_{00}^{(2n)} = O(1)$. However, one can be more precise – that is, $\mathtt{f}_{00} = 1$ (this requires a trick)! On the other hand, we have that

$$\mathtt{h}_{00} = \sum_{t>0} t \cdot \mathtt{r}_{00}^{(t)} \geq \sum_{n=1}^{\infty} 2n\mathtt{r}_{00}^{(2n)} = \sum_{n=1}^{\infty} \Theta\left(1/\sqrt{n}\right) = \infty.$$

Namely, 0 (and indeed all integers) are null persistent.

In finite Markov chains there are no null persistent states (this requires a proof, which is left as an exercise). There is a natural directed graph associated with a Markov chain. The states are the vertices, and the transition probability $\mathtt{P}_{ij}$ is the weight assigned to the edge $(i \to j)$. Note that we include only edges with $\mathtt{P}_{ij} > 0$.

Definition 30.4.3. A ***strong component*** (or a *strong connected component*) of a directed graph G is a maximal subgraph C of G such that for any pair of vertices i and j in the vertex set of C, there is a directed path from i to j, as well as a directed path from j to i.

Definition 30.4.4. A strong component C is a ***final strong component*** if there is no edge going from a vertex in C to a vertex that is not in C.

In a finite Markov chain, there is positive probability to arrive from any vertex on C to any other vertex of C in a finite number of steps. If C is a final strong component, then this probability is 1, since the Markov chain can never leave C once it enters it[2]. It follows that a state is persistent if and only if it lies in a final strong component.

Definition 30.4.5. A Markov chain is ***irreducible*** if its underlying graph consists of a single strong component.

Clearly, if a Markov chain is irreducible, then all states are persistent.

Definition 30.4.6. Let $\mathbf{q}^{(t)} = \left(q_1^{(t)}, q_2^{(t)}, \ldots, q_n^{(t)}\right)$ be the ***state probability vector*** (also known as the distribution of the chain at time t), to be the row vector whose ith component is the probability that the chain is in state i at time t.

The key observation is that

$$\mathbf{q}^{(t)} = \mathbf{q}^{(t-1)}\mathbf{P} = \mathbf{q}^{(0)}\mathbf{P}^t.$$

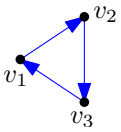
Namely, a Markov chain is fully defined by $\mathbf{q}^{(0)}$ and $\mathbf{P}$.

Definition 30.4.7. A ***stationary distribution*** for a Markov chain with the transition matrix $\mathbf{P}$ is a probability distribution π such that $\pi = \pi\mathbf{P}$.

In general, stationary distribution does not necessarily exist. We will mostly be interested in Markov chains that have stationary distribution. Intuitively it is clear that if a stationary distribution exists, then the Markov chain, given enough time, will converge to the stationary distribution.

Definition 30.4.8. The ***periodicity*** of a state i is the maximum integer T for which there exists an initial distribution $\mathbf{q}^{(0)}$ and positive integer a such that, for all t, if at time t we have $q_i^{(t)} > 0$ then t belongs to the arithmetic progression $\{a + Ti \mid i \geq 0\}$. A state is said to be ***periodic*** if it has periodicity greater than 1, and is ***aperiodic*** otherwise. A Markov chain in which every state is aperiodic is ***aperiodic***.

Example 30.4.9. The easiest example maybe of a periodic Markov chain is a directed cycle.



For example, the Markov chain on the right, has periodicity of three. In particular, the initial state probability vector $\mathbf{q}^{(0)} = (1, 0, 0)$ leads to the following sequence of state probability vectors

$$\mathbf{q}^{(0)} = (1,0,0) \implies \mathbf{q}^{(1)} = (0,1,0) \implies \mathbf{q}^{(2)} = (0,0,1) \implies \mathbf{q}^{(3)} = (1,0,0) \implies \ldots.$$

Note, that this chain still has a stationary distribution, that is $(1/3, 1/3, 1/3)$, but unless you start from this distribution, you are going to converge to it.

A neat trick that forces a Markov chain to be aperiodic, is to shrink all the probabilities by a factor of 2, and make every state to have a transition probability to itself equal to $1/2$. Clearly, the resulting Markov chain is aperiodic.

Definition 30.4.10. An ***ergodic*** state is aperiodic and (non-null) persistent.
An ***ergodic*** Markov chain is one in which all states are ergodic.

The following theorem is the fundamental property of Markov chains that we will need. The interested reader, should check the proof in [Nor98] (the proof is not hard).

[2] Think about it as hotel California.

Theorem 30.4.11 (Fundamental theorem of Markov chains). *Any irreducible, finite, and aperiodic Markov chain has the following properties.*

(i) All states are ergodic.

(ii) There is a unique stationary distribution π such that, for $1 \leq i \leq n$, we have $\pi_i > 0$.

(iii) For $1 \leq i \leq n$, we have $\mathtt{f}_{ii} = 1$ and $\mathtt{h}_{ii} = 1/\pi_i$.

(iv) Let $N(i, t)$ be the number of times the Markov chain visits state i in t steps. Then

$$\lim_{t \to \infty} \frac{N(i,t)}{t} = \pi_i.$$

Namely, independent of the starting distribution, the process converges to the stationary distribution.

References

[Nor98] J. R. Norris. *Markov Chains*. Statistical and Probabilistic Mathematics. Cambridge Press, 1998.

Chapter 31

Random Walks III

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "I gave the girl my protection, offering in my equivocal way to be her father. But I came too late, after she had ceased to believe in fathers. I wanted to do what was right, I wanted to make reparation: I will not deny this decent impulse, however mixed with more questionable motives: there must always be a place for penance and reparation. Nevertheless, I should never have allowed the gates of the town to be opened to people who assert that there are higher considerations that those of decency. They exposed her father to her naked and made him gibber with pain, they hurt her and he could not stop them (on a day I spent occupied with the ledgers in my office). Thereafter she was no longer fully human, sister to all of us. Certain sympathies died, certain movements of the heart became no longer possible to her. I too, if I live longer enough in this cell with its ghost not only of the father and the daughter but of the man who even by lamplight did not remove the black discs from his eyes and the subordinate whose work it was to keep the brazier fed, will be touched with the contagion and turned into a create that believes in nothing."
>
> J. M. Coetzee, Waiting for the Barbarians

31.1. Random walks on graphs

Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be a connected, non-bipartite, undirected graph, with n vertices. We define the natural Markov chain on G, where the transition probability is

$$\mathsf{P}_{uv} = \begin{cases} \frac{1}{d(u)} & \text{if } uv \in \mathsf{E} \\ 0 & \text{otherwise,} \end{cases}$$

where $d(w)$ is the degree of vertex w. Clearly, the resulting Markov chain M_G is irreducible. Note, that the graph must have an odd cycle, and it has a cycle of length 2. Thus, the gcd of the lengths of its cycles is 1. Namely, M_G is aperiodic. Now, by the Fundamental theorem of Markov chains, M_G has a unique stationary distribution π.

Lemma 31.1.1. *For all $v \in \mathsf{V}$, we have $\pi_v = d(v)/2m$.*

Proof: Since π is stationary, and the definition of P_{uv}, we get

$$\pi_v = [\pi \mathbf{P}]_v = \sum_{uv} \pi_u \mathsf{P}_{uv},$$

and this holds for all v. We only need to verify the claimed solution, since there is a unique stationary distribution. Indeed,

$$\frac{d(v)}{2m} = \pi_v = [\pi \mathbf{P}]_v = \sum_{uv} \frac{d(u)}{2m} \frac{1}{d(u)} = \frac{d(v)}{2m},$$

as claimed. ■

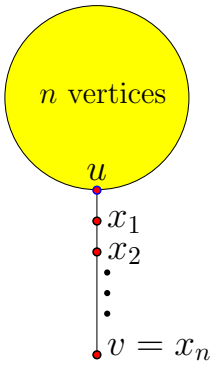
Definition 31.1.2. The ***hitting time*** h_{uv} is the expected number of steps in a random walk that starts at u and ends upon first reaching v.

The ***commute time*** between u and v is denoted by $\mathbf{CT}_{uv} = \mathsf{h}_{uv} + \mathsf{h}_{vu}$.

Let $\mathcal{C}_u(G)$ denote the expected length of a walk that starts at u and ends upon visiting every vertex in G at least once. The ***cover time*** of G denotes by $\mathcal{C}(G)$ is defined by $\mathcal{C}(G) = \max_u \mathcal{C}_u(G)$.

Lemma 31.1.3. *For all $v \in \mathsf{V}$, we have $\mathsf{h}_{vv} = 1/\pi_v = 2m/d(v)$.*

Example 31.1.4 (Lollipop). Let L_{2n} be the $2n$-vertex ***lollipop graph***, this graph consists of a clique on n vertices, and a path on the remaining n vertices. There is a vertex u in the clique which is where the path is attached to it. Let v denote the end of the path, see figure on the right.



Taking a random walk from u to v requires in expectation $O(n^2)$ steps, as we already saw in class. This ignores the probability of escape – that is, with probability $(n-1)/n$ when at u we enter the clique K_n (instead of the path). As such, it turns out that $\mathsf{h}_{uv} = \Theta(n^3)$, and $\mathsf{h}_{vu} = \Theta(n^2)$. (Thus, hitting times are not symmetric!)

Note, that the cover time is not monotone decreasing with the number of edges. Indeed, the path of length n, has cover time $O(n^2)$, but the larger graph L_n has cover time $\Omega(n^3)$.

Example 31.1.5 (More on walking on the Lollipop). To see why $\mathsf{h}_{uv} = \Theta(n^3)$, number the vertices on the stem $x_1, \ldots, x_n$. Let T_i be the expected time to arrive to the vertex x_i when starting a walk from u. Observe, that surprisingly, $T_1 = \Theta(n^2)$. Indeed, the walk has to visit the vertex u about n times in expectation, till the walk would decide to go to x_1 instead of falling back into the clique. The time between visits to u is in expectation $O(n)$ (assuming the walk is inside the clique).

Now, observe that $T_{2i} = T_i + \Theta(i^2) + \frac{1}{2}T_{2i}$. Indeed, starting with x_i, it takes in expectation $\Theta(i^2)$ steps of the walk to either arrive (with equal probability) at x_{2i} (good), or to get back to u (oopsi). In the later case, the game begins from scratch. As such, we have that

$$T_{2i} = 2T_i + \Theta\left(i^2\right) = 2\left(2T_{i/2} + \Theta\left((i/2)^2\right)\right) + \Theta\left(i^2\right) = \cdots = 2^{1+\log_2 i}T_1 + \Theta\left(i^2\right),$$

assuming i is a power of two (why not?). As such, $T_n = nT_1 + \Theta(n^2)$. Since $T_1 = \Theta(n^2)$, we have that $T_n = \Theta(n^3)$.

Definition 31.1.6. A $n \times n$ matrix M is ***stochastic*** if all its entries are non-negative and for each row i, it holds $\sum_k \mathsf{M}_{ik} = 1$. It is ***doubly stochastic*** if in addition, for any i, it holds $\sum_k \mathsf{M}_{ki} = 1$.

Lemma 31.1.7. *Let MC be a Markov chain, such that transition probability matrix $\mathbf{P}$ is* doubly stochastic. *Then, the distribution $u = (1/n, 1/n, \ldots, 1/n)$ is stationary for MC.*

Proof: $[u\mathbf{P}]_i = \sum_{k=1}^{n} \frac{\mathsf{P}_{ki}}{n} = \frac{1}{n}.$ ■

We can interpret every edge in G as corresponding to two directed edges. In particular, imagine performing a random walk in G, but remembering not only the current vertex in the walk, but also the (directed) edge used the walk to arrive to this vertex. One can interpret this as a random walk on the (directed) edges. Observe, that there are $2m$ directed edges. Furthermore, a vertex u of degree $d(u)$, has stationary distribution $\pi_u = d(u)/2m$. As such, the probability that the random walk would use any of the $d(u)$ outgoing edges from u is exactly

$\alpha = \pi_u/d(u) = 1/2m$. Namely, if we interpret the walk on the graph as walk on the directed edges, the stationary distribution is uniform. This readily implies that if $(u \to v)$ is in the graph, then $h_{(u\to v)(u\to v)}$ is $1/\alpha = 2m$. This readily implies that the expected time to go from u to v and back to u is at most $2m$. Next, we provide a more formal (and somewhat different) proof of this.

Lemma 31.1.8. *For any edge $(u \to v) \in \mathsf{E}$, we have $\mathtt{h}_{uv} + \mathtt{h}_{vu} \le 2m$.*

(Note, that $(u \to v)$ being an edge in the graph is crucial. Indeed, without it a significantly worst case bound holds, see Theorem 31.2.1.)

Proof: Consider a new Markov chain defined by the edges of the graph (where every edge is taken twice as two directed edges), where the current state is the last (directed) edge visited. There are $2m$ edges in the new Markov chain, and the new transition matrix, has $Q_{(u\to v),(v\to w)} = \mathtt{P}_{vw} = \frac{1}{d(v)}$. This matrix is *doubly stochastic*, meaning that not only do the rows sum to one, but the columns sum to one as well. Indeed, for an edge $(v \to w)$ we have

$$\sum_{x\in V, y\in\Gamma(x)} Q_{(x\to y),(v\to w)} = \sum_{u\in\Gamma(v)} Q_{(u\to v),(v\to w)} = \sum_{u\in\Gamma(v)} \mathtt{P}_{vw} = d(v) \times \frac{1}{d(v)} = 1.$$

Thus, the stationary distribution for this Markov chain is uniform, by Lemma 31.1.7. Namely, the stationary distribution of $e = (u \to v)$ is $\mathtt{h}_{ee} = \pi_e = 1/(2m)$. Thus, the expected time between successive traversals of e is $1/\pi_e = 2m$, by Theorem 30.4.11 (iii).

Consider $\mathtt{h}_{uv} + \mathtt{h}_{vu}$ and interpret this as the time to go from u to v and then return to u. Conditioned on the event that the initial entry into u was via the edge $(v \to u)$, we conclude that the expected time to go from there to v and then finally use $(v \to u)$ is $2m$. The memorylessness property of a Markov chains now allows us to remove the conditioning: since how we arrived to u is not relevant. Thus, the expected time to travel from u to v and back is at most $2m$. ■

31.2. Electrical networks and random walks

A ***resistive electrical network*** is an undirected graph. Each edge has ***branch resistance*** associated with it. The electrical flow is determined by two laws: ***Kirchhoff's law*** (preservation of flow - all the flow coming into a node, leaves it) and ***Ohm's law*** (the voltage across a resistor equals the product of the resistance times the current through it). Explicitly, Ohm's law states

$$\text{voltage} = \text{resistance} * \text{current}.$$

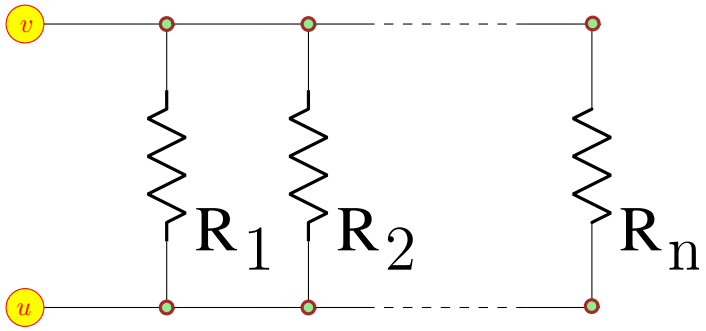
The ***effective resistance*** between nodes u and v is the voltage difference between u and v when one ampere is injected into u and removed from v (or injected into v and removed from u). The effective resistance is always bounded by the branch resistance, but it can be much lower.

Given an undirected graph G, let $\mathcal{N}(G)$ be the electrical network defined over G, associating one ohm resistance on the edges of $\mathcal{N}(G)$.

You might now see the connection between a random walk on a graph and electrical network. Intuitively (used in the most unscientific way possible), the electricity, is made out of electrons each one of them is doing a random walk on the electric network. The resistance of an edge, corresponds to the probability of taking the edge. The higher the resistance, the lower the probability that we will travel on this edge. Thus, if the effective resistance $\mathbf{R}_{uv}$ between u and v is low, then there is a good probability that travel from u to v in a random walk, and $\mathtt{h}_{uv}$ would be small.

31.2.1. A tangent on parallel and series resistors

Consider having n resistors in parallel with resistance $R_1, \ldots, R_n$, connecting two nodes u and v. As follows:

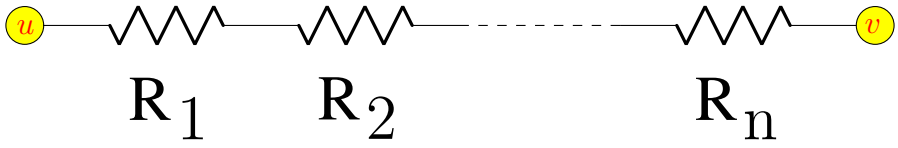


The *effective resistance* between u and v is

$$R_{uv} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} \cdots + \frac{1}{R_n}}.$$

In particular, if $R_1 = \cdots = R_n = R$, then we have that $R_{uv} = 1/(1/R + \cdots 1/R) = 1/(n/R) = R/n$.

Similarly, if we have n resistors in series, with resistance $R_1, R_2, \ldots, R_n$, between u and v:



Then, the effective resistance between u and v is

$$R_{uv} = R_1 + \cdots + R_n.$$

In particular, if $R_1 = \cdots = R_n$, then $R_{uv} = nR$.

31.2.2. Back to random walks

Theorem 31.2.1. *For any two vertices u and v in G, the commute time $\mathbf{CT}_{uv} = 2m\mathbf{R}_{uv}$, where $\mathbf{R}_{uv}$ is the effective resistance between u and v.*

Proof: Let ϕ_{uv} denote the voltage at u in $\mathcal{N}(G)$ with respect to v, where $d(x)$ amperes of current are injected into each node $x \in \mathsf{V}$, and $2m$ amperes are removed from v. We claim that

$$\mathtt{h}_{uv} = \phi_{uv}.$$

Note, that the voltage on an edge xy is $\phi_{xy} = \phi_{xv} - \phi_{yv}$. Thus, using Kirchhoff's Law and Ohm's Law, we obtain that

$$x \in V \setminus \{v\} \qquad d(x) = \sum_{w \in \Gamma(x)} \text{current}(xw) = \sum_{w \in \Gamma(x)} \frac{\phi_{xw}}{\text{resistance}(xw)} = \sum_{w \in \Gamma(x)} (\phi_{xv} - \phi_{wv}), \tag{31.1}$$

since the resistance of every edge is 1 ohm. (We also have the "trivial" equality that $\phi_{vv} = 0$.) Furthermore, we have only n variables in this system; that is, for every $x \in \mathsf{V}$, we have the variable ϕ_{xv}.

Now, for the random walk interpretation – by the definition of expectation, we have

$$x \in V \setminus \{v\} \qquad \mathbf{h}_{xv} = \frac{1}{d(x)} \sum_{w \in \Gamma(x)} (1 + \mathbf{h}_{wv}) \qquad \Longleftrightarrow \qquad d(x)\,\mathbf{h}_{xv} = \sum_{w \in \Gamma(x)} 1 + \sum_{w \in \Gamma(x)} \mathbf{h}_{wv}$$
$$\Longleftrightarrow \qquad \sum_{w \in \Gamma(x)} 1 = d(x)\,\mathbf{h}_{xv} - \sum_{w \in \Gamma(x)} \mathbf{h}_{wv} = \sum_{w \in \Gamma(x)} (\mathbf{h}_{xv} - \mathbf{h}_{wv}).$$

Since $d(x) = \sum_{w \in \Gamma(x)} 1$, this is equivalent to

$$x \in V \setminus \{v\} \qquad d(x) = \sum_{w \in \Gamma(x)} (\mathbf{h}_{xv} - \mathbf{h}_{wv}). \tag{31.2}$$

Again, we also have the trivial equality $\mathbf{h}_{vv} = 0$.[①] Note, that this system also has n equalities and n variables.

Eq. (31.1) and Eq. (31.2) show two systems of linear equalities. Furthermore, if we identify $\mathbf{h}_{uv}$ with ϕ_{xv} then they are exactly the same system of equalities. Furthermore, since Eq. (31.1) represents a physical system, we know that it has a unique solution. This implies that $\phi_{xv} = \mathbf{h}_{xv}$, for all $x \in \mathsf{V}$.

Imagine the network where u is injected with $2m$ amperes, and for all nodes w remove $d(w)$ units from w. In this new network, $\mathbf{h}_{vu} = -\phi'_{vu} = \phi'_{uv}$. Now, since flows behaves linearly, we can superimpose them (i.e., add them up). We have that in this new network $2m$ unites are being injected at u, and $2m$ units are being extracted at v, all other nodes the charge cancel itself out. The voltage difference between u and v in the new network is $\widehat{\phi} = \phi_{uv} + \phi'_{uv} = \mathbf{h}_{uv} + \mathbf{h}_{vu} = \mathbf{CT}_{uv}$. Now, in the new network there are $2m$ amperes going from u to v, and by Ohm's law, we have

$$\widehat{\phi} = \text{voltage} = \text{resistance} * \text{current} = 2m\mathbf{R}_{uv},$$

as claimed. ■

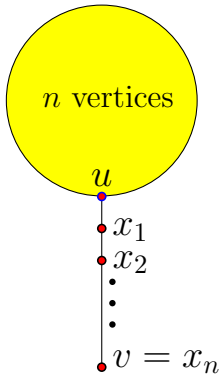


Figure 31.1: Lollipop again.

Example 31.2.2. Recall the lollipop L_n from Exercise 31.1.4, see Figure 31.1. Let u be the connecting vertex between the clique and the stem (i.e., the path). The effective resistance between u and v is n since there are n resistors in series along the stem. That is $R_{uv} = n$.

The number of edges in the lollipop is $\binom{n}{2} + n = n(n-1)/2 + n = n(n+1)/2$. As such, the commute time $\mathbf{h}_{vu} + \mathbf{h}_{uv} = \mathbf{CT}_{uv} = 2m\mathbf{R}_{uv} = 2(n(n+1)/2)n = n^2(n+1)$.

We already know that $\mathbf{h}_{vu} = \Theta(n^2)$. This implies that $\mathbf{h}_{uv} = \mathbf{CT}_{uv} - \mathbf{h}_{vu} = \Theta(n^3)$.

[①] In previous lectures, we interpreted $\mathbf{h}_{vv}$ as the expected length of a walk starting at v and coming back to v.

Lemma 31.2.3. *For any n vertex connected graph G, and for all $u, v \in V(G)$, we have $\mathbf{CT}_{uv} < n^3$.*

Proof: The effective resistance between any two nodes in the network is bounded by the length of the shortest path between the two nodes, which is at most $n - 1$. As such, plugging this into Theorem 31.2.1, yields the bound, since $m < n^2$. ■

31.3. Bibliographical Notes

A nice survey of the material covered here, is available online at http://arxiv.org/abs/math.PR/0001057 [DS00].

References

[DS00] P. G. Doyle and J. L. Snell. Random walks and electric networks. *ArXiv Mathematics e-prints*, 2000. eprint: math/0001057.

Chapter 32

Random Walks IV

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "Do not imagine, comrades, that leadership is a pleasure! On the contrary, it is a deep and heavy responsibility. No one believes more firmly than Comrade Napoleon that all animals are equal. He would be only too happy to let you make your decisions for yourselves. But sometimes you might make the wrong decisions, comrades, and then where should we be? Suppose you had decided to follow Snowball, with his moonshine of windmills-Snowball, who, as we now know, was no better than a criminal?"
>
> Animal Farm, George Orwell

32.1. Cover times

We remind the reader that the cover time of a graph is the expected time to visit all the vertices in the graph, starting from an arbitrary vertex (i.e., worst vertex). The cover time is denoted by $\mathcal{C}(\mathsf{G})$.

Theorem 32.1.1. *Let* G *be an undirected connected graph, then* $\mathcal{C}(\mathsf{G}) \leq 2m(n-1)$*, where* $n = |V(\mathsf{G})|$ *and* $m = |E(\mathsf{G})|$.

Proof: (Sketch.) Construct a spanning tree T of G, and consider the time to walk around T. The expected time to travel on this edge on both directions is $\mathbf{CT}_{uv} = \mathtt{h}_{uv} + \mathtt{h}_{vu}$, which is smaller than $2m$, by Lemma 31.1.8. Now, just connect up those bounds, to get the expected time to travel around the spanning tree. Note, that the bound is independent of the starting vertex. ■

Definition 32.1.2. The ***resistance*** of G is $\mathbf{R}(\mathsf{G}) = \max_{u,v \in V(\mathsf{G})} \mathbf{R}_{uv}$; namely, it is the maximum effective resistance in G.

Theorem 32.1.3. $m\mathbf{R}(\mathsf{G}) \leq \mathcal{C}(\mathsf{G}) \leq 2e^3 m\mathbf{R}(\mathsf{G}) \ln n + 2n$.

Proof: Consider the vertices u and v realizing $\mathbf{R}(\mathsf{G})$, and observe that $\max(\mathtt{h}_{uv}, \mathtt{h}_{vu}) \geq \mathbf{CT}_{uv}/2$, and $\mathbf{CT}_{uv} = 2m\mathbf{R}_{uv}$ by Theorem 31.2.1. Thus, $\mathcal{C}(\mathsf{G}) \geq \mathbf{CT}_{uv}/2 \geq m\mathbf{R}(\mathsf{G})$.

As for the upper bound. Consider a random walk, and divide it into ***epochs***, where a epoch is a random walk of length $2e^3 m\mathbf{R}(\mathsf{G})$. For any vertex v, the expected time to hit u is $\mathtt{h}_{vu} \leq 2m\mathbf{R}(\mathsf{G})$, by Theorem 31.2.1. Thus, the probability that u is not visited in a epoch is $1/e^3$ by the Markov inequality. Consider a random walk with $t = \ln n$ epochs. We have that the probability of not visiting u is $\leq (1/e^3)^{\ln n} \leq 1/n^3$. Thus, all vertices are visited after $\ln n$ epochs, with probability $\geq 1 - \binom{n}{2}/n^3 \geq 1 - 1/n$. Otherwise, after this walk, we perform a random walk till we visit all vertices. The length of this (fix-up) random walk is $\leq 2n^3$, by Theorem 32.1.1. Thus, expected length of the walk is $\leq 2e^3 m\mathbf{R}(\mathsf{G}) \ln n + 2n^3(1/n^2)$. ■

32.1.1. Rayleigh's Short-cut Principle.

Observe that effective resistance is never raised by lowering the resistance on an edge, and it is never lowered by raising the resistance on an edge. Similarly, resistance is never lowered by removing a vertex.

Interestingly, effective resistance comply with the triangle inequality.

Observation 32.1.4. *For a graph with minimum degree d, we have $\mathbf{R}(\mathsf{G}) \geq 1/d$ (collapse all vertices except the minimum-degree vertex into a single vertex).*

Lemma 32.1.5. *Suppose that G contains p edge-disjoint paths of length at most ℓ from s to t. Then $\mathbf{R}_{st} \leq \ell/p$.*

32.2. Graph Connectivity

Definition 32.2.1. A ***probabilistic log-space Turing machine*** for a language L is a Turing machine using space $O(\log n)$ and running in time $O(\mathrm{poly}(n))$, where n is the input size. A problem A is in RLP, if there exists a probabilistic log-space Turing machine M such that M accepts $x \in L(A)$ with probability larger than $1/2$, and if $x \notin L(A)$ then $M(x)$ always reject.

Theorem 32.2.2. *Let* USTCON *denote the problem of deciding if a vertex s is connected to a vertex t in an undirected graph. Then* USTCON $\in$ RLP.

Proof: Perform a random walk of length $2n^3$ in the input graph G, starting from s. Stop as soon as the random walk hit t. If u and v are in the same connected component, then $\mathsf{h}_{st} \leq n^3$. Thus, by the Markov inequality, the algorithm works. It is easy to verify that it can be implemented in $O(\log n)$ space. ■

Definition 32.2.3. A graph is ***d-regular***, if all its vertices are of degree d.

A d-regular graph is ***labeled*** if at each vertex of the graph, each of the d edges incident on that vertex has a unique label in $\{1, \ldots, d\}$.

Any sequence of symbols $\sigma = (\sigma_1, \sigma_2, \ldots)$ from $\{1, \ldots, d\}$ together with a starting vertex s in a labeled graph describes a ***walk*** in the graph. For our purposes, such a walk would almost always be finite.

A sequence σ is said to ***traverse*** a labeled graph if the walk visits every vertex of G regardless of the starting vertex. A sequence σ is said to be a ***universal traversal sequence*** of a labeled graph if it traverses all the graphs in this class.

Given such a universal traversal sequence, we can construct (a non-uniform) Turing machine that can solve USTCON for such d-regular graphs, by encoding the sequence in the machine.

Let $\mathcal{F}$ denote a family of graphs, and let $U(\mathcal{F})$ denote the length of the shortest universal traversal sequence for all the labeled graphs in $\mathcal{F}$. Let $\mathbf{R}(\mathcal{F})$ denote the maximum resistance of graphs in this family.

Theorem 32.2.4. *Let $\mathcal{F}$ be a family of d-regular graphs with n vertices, then $U(\mathcal{F}) \leq 5m\mathbf{R}(\mathcal{F}) \lg(n|\mathcal{F}|)$.*

Proof: Same old, same old. Break the string into *epochs*, each of length $L = 2m\mathbf{R}(\mathsf{G})$. Now, start random walks from all the possible vertices, for all graphs in $\mathcal{F}$. Continue the walks till all vertices are being visited. Initially, there are $n^2|\mathcal{F}|$ vertices that need to visited. In expectation, in each epoch half the vertices get visited. There are $n|\mathcal{F}|$ walks, each of them needs to visit n vertices. As such, the number of vertices waiting to be visited is bounded by $|\mathcal{F}|n^2$. As such, after $1 + \lg_2\left(n^2|\mathcal{F}|\right)$ epochs, the expected number of vertices still need visiting is $\leq 1/2$. Namely, with constant probability we are done. ■

Let $U(d, n)$ denote the length of the shortest universal traversal sequence of connected, labeled n-vertex, d-regular graphs.

Lemma 32.2.5. *The number of labeled n-vertex graphs that are d-regular is $(nd)^{O(nd)}$.*

Proof: Such a graph has $dn/2$ edges overall. Specifically, we encode this by listing for every vertex its d neighbors – there are $N = \binom{n-1}{d} \leq n^d$ possibilities. As such, there are at most $N^n \leq n^{nd}$ choices for edges in the graph[①]. Every vertex has $d!$ possible labeling of the edges adjacent to it, thus there are $(d!)^n \leq d^{nd}$ possible labelings. ■

Lemma 32.2.6. $U(d, n) = O\Big(n^3 d \log n\Big)$.

Proof: The diameter of every connected n-vertex, d-regular graph is $O(n/d)$. Indeed, consider the path realizing the diameter of the graph, and assume it has t vertices. Number the vertices along the path consecutively, and consider all the vertices that their number is a multiple of three. There are $\alpha \geq \lfloor t/3 \rfloor$ such vertices. No pair of these vertices can share a neighbor, and as such, the graph has at least $(d+1)\alpha$ vertices. We conclude that $n \geq (d+1)\alpha = (d+1)(t/3 - 1)$. We conclude that $t \leq \frac{3}{d+1}(n+1) \leq 3n/d$.

And so, this also bounds the resistance of such a graph. The number of edges is $m = nd/2$. Now, combine Lemma 32.2.5 and Theorem 32.2.4. ■

This is, as mentioned before, not a uniform algorithm. There is by now a known log-space deterministic algorithm for this problem, which is uniform.

32.2.1. Directed graphs

Theorem 32.2.7. *One can solve the $\overrightarrow{\text{STCON}}$ problem, for a given directed graph with n vertices, using a log-space randomized algorithm, that always output NO if there is no path from s to t, and output YES with probability at least $1/2$ if there is a path from s to t.*

Proof: (Sketch.) The basic idea is simple – start a random walk from s, if it fails to arrive to t after a certain number of steps, then restart. The only challenging thing is that the number of times we need to repeat this is exponentially large. Indeed, the probability of a random walk from s to arrive to t in n steps, is at least $p = 1/n^{n-1} \geq n^n$ if s is connected to t.

As such, we need to repeat this walk $N = O((1/p)\log \delta) = O(n^{n+1})$ times, for $\delta \geq 1/2^n$. If have all of these walks fail, then with probability $\geq 1 - \delta$, there is no path from s to t.

We can do the walk using logarithmic space. However, how do we count to N (reliably) using only logarithmic space? We leave this as an exercise to the reader, see Exercise 32.2.8, ■

Exercise 32.2.8. Let N be a large integer number. Show a randomized algorithm, that with, high probability, counts from 1 to M, where $M \geq N$, and always stops. The algorithm should use only $O(\log \log N)$ bits.

[①] This is a callous upper bound – better analysis is possible. But never analyze things better than you have to - it usually a waste of time.

Chapter 33

A Bit on Algebraic Graph Theory

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "The Party told you to reject the evidence of your eyes and ears. It was their final, most essential command."
>
> 1984, George Orwell

33.1. Graphs and Eigenvalues

Consider an undirected graph $\mathsf{G} = \mathsf{G}(V, E)$ with n vertices. The adjacency matrix $\mathsf{M}(\mathsf{G})$ of G is the $n \times n$ symmetric matrix where $\mathsf{M}_{ij} = \mathsf{M}_{ji}$ is the number of edges between the vertices v_i and v_j. If G is bipartite, we assume that V is made out of two independent sets X and Y. In this case the matrix $\mathsf{M}(\mathsf{G})$ can be written in block form.

33.1.1. Eigenvalues and eigenvectors

A non-zero vector v is an eigenvector of M, if there is a value λ, known as the ***eigenvalue*** of v, such that $\mathsf{M}v = \lambda v$. That is, the vector v is mapped to zero by the matrix $\mathsf{N} = \mathsf{M} - \lambda I$. This happens only if N is not full ranked, which in turn implies that $\det(\mathsf{N}) = 0$. We have that $f(\lambda) = \det(\mathsf{M} - \lambda I)$ is a polynomial of degree n. It has n roots (not necessarily real), which are the ***eigenvalues*** of M. A matrix $\mathsf{N} \in \mathbb{R}^{n \times n}$ is ***symmetric*** if $\mathsf{N}^T = \mathsf{N}$.

Lemma 33.1.1. *The eigenvalues of a symmetric real matrix $\mathsf{N} \in \mathbb{R}^{n \times n}$ are real numbers.*

Proof: Observe that for any real vector $v = (v_1, \ldots, v_n) \in \mathbb{R}^n$, we have that $\sum_{i=1}^n v_i^2 = \langle v, v \rangle \geq 0$. As such, for a vector v with eigenvalue λ, we have

$$0 \leq \langle \mathsf{N}v, \mathsf{N}v \rangle = (\mathsf{N}v)^T \mathsf{N}v = (\lambda v)^T \lambda v = \lambda^2 \langle v, v \rangle .$$

Namely, λ^2 is a non-negative number, which implies that the λ is a real number. ■

Lemma 33.1.2. *Let $\mathsf{N} \in \mathbb{R}^{n \times n}$ be a matrix. Consider two eigenvectors v_1, v_2 that corresponds to two eigenvalues λ_1, λ_2, where $\lambda_1 \neq \lambda_2$. Then v_1 and v_2 are orthogonal.*

Proof: Indeed, $v_1^T \mathsf{N} v_2 = \lambda_2 v_1^T v_2$. Similarly, we have $v_1^T \mathsf{N} v_2 = (\mathsf{N}^T v_1)^T v_2) = \lambda_1 v_1^T v_2$. We conclude that either $\lambda_1 = \lambda_2$, or v_1 and v_2 are orthogonal (i.e., $v_1^T v_2 = 0$). ■

33.1.2. Eigenvalues and eigenvectors of a graph

Since $\mathsf{N} = \mathsf{M}(\mathsf{G})$ the adjacency matrix of an undirected graph is symmetric, all its eigenvalues exists and are real numbers $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_n$, and their corresponding orthonormal basis vectors are $e_1, \ldots, e_n$.

We will need the following theorem.

Theorem 33.1.3 (Fundamental theorem of algebraic graph theory). *Let $\mathsf{G} = \mathsf{G}(V, E)$ be an undirected (multi)graph with maximum degree d and with n vertices. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ be the eigenvalues of $\mathsf{M}(\mathsf{G})$ and the corresponding orthonormal eigenvectors are $e_1, \ldots, e_n$. The following holds.*

(i) *If G is connected then $\lambda_2 < \lambda_1$.*

(ii) *For $i = 1, \ldots, n$, we have $|\lambda_i| \leq d$.*

(iii) *d is an eigenvalue if and only if G is regular.*

(iv) *If G is d-regular then the eigenvalue $\lambda_1 = d$ has the eigenvector $e_1 = \frac{1}{\sqrt{n}}(1, 1, 1, \ldots, 1)$.*

(v) *The graph G is bipartite if and only if for every eigenvalue λ there is an eigenvalue $-\lambda$ of the same multiplicity.*

(vi) *Suppose that G is connected. Then G is bipartite if and only if $-\lambda_1$ is an eigenvalue.*

(vii) *If G is d-regular and bipartite, then $\lambda_n = d$ and $e_n = \frac{1}{\sqrt{n}}(1, 1, \ldots, 1, -1, \ldots, -1)$, where there are equal numbers of 1s and -1s in e_n.*

33.2. Bibliographical Notes

A nice survey of algebraic graph theory appears in [Wes01] and in [Bol98].

References

[Bol98] B. Bollobas. *Modern Graph Theory*. Springer-Verlag, 1998.

[Wes01] D. B. West. *Intorudction to Graph Theory*. 2ed. Prentice Hall, 2001.

Chapter 34

Random Walks V

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "Is there anything in the Geneva Convention about the rules of war in peacetime?" Stanko wanted to know, crawling back toward the truck. "Absolutely nothing," Caulec assured him. "The rules of war apply only in wartime. In peacetime, anything goes."
>
> Romain Gary, Gasp

34.1. Explicit expander construction

We state here a few facts about expander graphs without proofs (see also Theorem 33.1.3).

Definition 34.1.1. And (n, d, c)-expander is a d-regular bipartite graph $\mathsf{G} = (X, Y, \mathsf{E})$, where $|X| = |Y| = n/2$. Here, we require that for any $S \subseteq X$, we have

$$|\Gamma(S)| \geq \left(1 + c\left(1 - \frac{2|S|}{n}\right)\right)|S|.$$

The Margulis-Gabber-Galil expander. For a positive m, let $n = 2m^2$. Each vertex in X and Y above, is interpreted as a pair (a, b), where $a, b \in \mathbb{Z}_m = \{0, \ldots, m-1\}$. A vertex $(a, b) \in X$ is connected to the vertices

$$(a, b), \quad (a, a+b), \quad (a, a+b+1), \quad (a+b, b), \quad \text{and} \quad (a+b+1, b),$$

in Y, where the addition is done module m.

Theorem 34.1.2 ([GG81]). *The above graph is* 5 *regular, and it is* $(n, 5, (2-\sqrt{3})/4)$*-expander.*

Specrtral gap and expansion. We remind the reader that a d-regular graph, then its adjacency matrix $\mathsf{M}(\mathsf{G})$ has (as its biggest eigenvalue) the eigenvalue $\lambda_1 = d$. In particular, let $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$ be the eigenvalues of $\mathsf{M}(\mathsf{G})$. We then have the following:

Theorem 34.1.3. *If* G *is an* (n, d, c)*-expander then* $\mathsf{M}(\mathsf{G})$ *has* $|\lambda_2| \leq d - \dfrac{c^2}{1024 + 2c^2}$.

Theorem 34.1.4. *If* $\mathsf{M}(\mathsf{G})$ *has* $|\lambda_2| \leq d - \varepsilon$*, then* G *is* (n, d, c)*-expander with* $c \geq \dfrac{2d\varepsilon - \varepsilon^2}{d^2}$.

34.2. Rapid mixing for expanders

Here is another equivalent definition of an expander.

Definition 34.2.1. Let $\mathsf{G} = (V, E)$ be an undirected d-regular graph. The graph G is a ***(n, d, c)-expander*** (or just ***c-expander***), for every set $S \subseteq V$ of size at most $|V|/2$, there are at least $cd|S|$ edges connecting S and $\overline{S} = V \setminus S$; that is $e(S, \overline{S}) \geq cd|S|$,

Guaranteeing aperiodicity. Let G be a (n, d, c)-expander. We would like to perform a random walk on G. The graph G is connected, but it might be periodic (i.e., bipartite). To overcome this, consider the random walk on G that either stay in the current state with probability $1/2$ or traverse one of the edges. Clearly, the resulting Markov Chain (MC) is aperiodic. The resulting ***transition matrix*** is

$$\mathsf{Q} = \mathsf{M}/2d + \mathcal{I}/2,$$

where M is the adjacency matrix of G and $\mathcal{I}$ is the identity $n \times n$ matrix. Clearly Q is doubly stochastic. Furthermore, if $\widehat{\lambda_i}$ is an eigenvalue of M, with eigenvector v_i, then

$$\mathsf{Q}v_i = \frac{1}{2}\left(\frac{\mathsf{M}}{d} + \mathcal{I}\right)v_i = \frac{1}{2}\left(\frac{\widehat{\lambda_i}}{d} + 1\right)v_i.$$

As such, $(\widehat{\lambda_i}/d + 1)/2$ is an eigenvalue of Q. Namely, if there is a spectral gap in the graph G, there would also be a similar spectral gap in the resulting MC. This MC can be generated by adding to each vertex d self loops, ending up with a $2d$-regular graph. Clearly, this graph is still an expander if the original graph is an expander, and the random walk on it is aperiodic.

From this point on, we would just assume our expander is aperiodic.

34.2.1. Bounding the mixing time

For a MC with n states, we denote by $\pi = (\pi_1, \ldots, \pi_n)$ its stationary distribution. We consider only nicely behave MC that fall under Theorem 30.4.11. As such, no state in the MC has zero stationary probability.

Definition 34.2.2. Let $\mathsf{q}^{(t)}$ denote the state probability vector of a Markov chain defined by a transition matrix Q at time $t \geq 0$, given an initial distribution $\mathsf{q}^{(0)}$. The ***relative pairwise distance*** of the Markov chain at time t is

$$\Delta(t) = \max_i \frac{\left|\mathsf{q}_i^{(t)} - \pi_i\right|}{\pi_i}.$$

Namely, if $\Delta(t)$ approaches zero then $\mathsf{q}^{(t)}$ approaches π.

We remind the reader that we saw a construction of a constant degree expander with constant expansion. In its transition matrix Q, we have that $\widehat{\lambda_1} = 1$, and $-1 \leq \widehat{\lambda_2} < 1$, and furthermore the ***spectral gap*** $\widehat{\lambda_1} - \widehat{\lambda_2}$ was a constant (the two properties are equivalent, but we proved only one direction of this).

We need a slightly stronger property (that does hold for our expander construction). We have that $\widehat{\lambda_2} \geq \max_{i=2}^{n} \left|\widehat{\lambda_i}\right|$.

Theorem 34.2.3. *Let Q be the transition matrix of an aperiodic (n, d, c)-expander. Then, for any initial distribution $\mathsf{q}^{(0)}$, we have that*

$$\Delta(t) \leq n^{3/2}(\widehat{\lambda_2})^t.$$

Since $\widehat{\lambda_2}$ is a constant smaller than 1, the distance $\Delta(t)$ drops exponentially with t.

Proof: We have that $\mathsf{q}^{(t)} = \mathsf{q}^{(0)}\mathsf{Q}^t$. Let $\mathcal{B}(\mathsf{Q}) = \langle v_1, \ldots, v_n \rangle$ denote the orthonormal eigenvector basis of Q (see Definition 45.2.3$_{\text{p289}}$), and write $\mathsf{q}^{(0)} = \sum_{i=1}^{n} \alpha_i v_i$. Since $\widehat{\lambda}_1 = 1$, we have that

$$\mathsf{q}^{(t)} = \mathsf{q}^{(0)}\mathsf{Q}^t = \sum_{i=1}^{n} \alpha_i (v_i \mathsf{Q}^t) = \sum_{i=1}^{n} \alpha_i (\widehat{\lambda}_i)^t v_i = \alpha_1 v_1 + \sum_{i=2}^{n} \alpha_i (\widehat{\lambda}_i)^t v_i.$$

Since $v_1 = (1/\sqrt{n}, \ldots, 1/\sqrt{n})$, and $\left|\widehat{\lambda}_i\right| \leq \widehat{\lambda}_2 < 1$, for $i > 1$, we have that $\lim_{t\to\infty} (\widehat{\lambda}_i)^t = 0$, and thus

$$\pi = \lim_{t\to\infty} \mathsf{q}^{(t)} = \alpha_1 v_1 + \sum_{i=2}^{n} \alpha_i \Big(\lim_{t\to\infty} (\widehat{\lambda}_i)^t \Big) v_i = \alpha_1 v_1.$$

Now, since $v_1, \ldots, v_n$ is an orthonormal basis, and $\mathsf{q}^{(0)} = \sum_{i=1}^{n} \alpha_i v_i$, we have that $\|\mathsf{q}^{(0)}\|_2 = \sqrt{\sum_{i=1}^{n} \alpha_i^2}$. Thus implies that

$$\left\|\mathsf{q}^{(t)} - \pi\right\|_1 = \left\|\mathsf{q}^{(t)} - \alpha_1 v_1\right\|_1 = \left\|\sum_{i=2}^{n} \alpha_i \left(\widehat{\lambda}_i\right)^t v_i\right\|_1 \leq \sqrt{n} \left\|\sum_{i=2}^{n} \alpha_i (\widehat{\lambda}_i)^t v_i\right\|_2 = \sqrt{n} \sqrt{\sum_{i=2}^{n} (\alpha_i (\widehat{\lambda}_i)^t)^2}$$

$$\leq \sqrt{n} (\widehat{\lambda}_2)^t \sqrt{\sum_{i=2}^{n} (\alpha_i)^2} \leq \sqrt{n} \left(\widehat{\lambda}_2\right)^t \left\|\mathsf{q}^{(0)}\right\|_2 \leq \sqrt{n} (\widehat{\lambda}_2)^t \left\|\mathsf{q}^{(0)}\right\|_1 = \sqrt{n} (\widehat{\lambda}_2)^t,$$

since $\mathsf{q}^{(0)}$ is a distribution. Now, since $\pi_i = 1/n$, we have

$$\Delta(t) = \max_i \frac{\left|\mathsf{q}_i^{(t)} - \pi_i\right|}{\pi_i} = \max_i n \left|\mathsf{q}_i^{(t)} - \pi_i\right| \leq n \max_i \|\mathsf{q}^{(t)} - \pi\|_1 \leq n \sqrt{n} (\widehat{\lambda}_2)^t. \qquad \blacksquare$$

34.3. Probability amplification by random walks on expanders

We are interested in performing probability amplification for an algorithm that is a **BPP** algorithm (see Definition 35.2.8). It would be convenient to work with an algorithm which is already somewhat amplified. That is, we assume that we are given a **BPP** algorithm **Alg** for a language L, such that

(A) If $x \in L$ then $\mathbb{P}[\textbf{Alg}(x) \text{ accepts}] \geq 199/200$.

(B) If $x \notin L$ then $\mathbb{P}[\textbf{Alg}(x) \text{ accepts}] \leq 1/200$.

We assume that **Alg** requires a random bit string of length n. So, we have a constant degree expander G (say of degree d) that has at least $200 \cdot 2^n$ vertices. In particular, let

$$U = |V(\mathsf{G})|,$$

and since our expander construction grow exponentially in size (but the base of the exponent is a constant), we have that $U = O(2^n)$. (Translation: We can not quite get an expander with a specific number of vertices. Rather, we can guarantee an expander that has more vertices than we need, but not many more.)

We label the vertices of G with all the binary strings of length n, in a round robin fashion (thus, each binary string of length n appears either $\lceil |V(\mathsf{G})| / 2^n \rceil$ or $\lfloor |V(\mathsf{G})| / 2^n \rfloor$ times). For a vertex $v \in V(\mathsf{G})$, let $\mathsf{s}(v)$ denote the binary string associated with v.

Consider a string x that we would like to decide if it is in L or not. We know that at least $99/100U$ vertices of G are labeled with "random" strings that would yield the right result if we feed them into **Alg** (the constant here deteriorated from $199/200$ to $99/100$ because the number of times a string appears is not identically the same for all strings).

The algorithm. We perform a random walk of length $\mu = \alpha\beta k$ on G, where α and β are constants to be determined shortly, and k is a parameter. To this end, we randomly choose a starting vertex X_0 (this would require $n + O(1)$ bits). Every step in the random walk, would require $O(1)$ random bits, as the expander is a constant degree expander, and as such overall, this would require $n + O(k)$ random bits.

Now, lets $X_0, X_1, \ldots, X_\mu$ be the resulting random walk. We compute the result of

$$Y_i = \mathbf{Alg}(x, r_i), \qquad \text{for} \qquad i = 0, \ldots, \nu, \qquad \text{and } \nu = \alpha k,$$

where $r_i = \mathsf{s}\left(X_{i\cdot\beta}\right)$. Specifically, we use the strings associated with nodes that are in distance β from each other along the path of the random walk. We return the majority of the bits $Y_0, \ldots, Y_{\alpha k}$ as the decision of whether $x \in L$ or not.

We assume here that we have a ***fully explicit*** construction of an expander. That is, given a vertex of an expander, we can compute all its neighbors in polynomial time (in the length of the index of the vertex). While the construction of expander shown is only explicit it can be made fully explicit with more effort.

34.3.1. The analysis

Intuition. Skipping every β nodes in the random walk corresponds to performing a random walk on the graph G^β; that is, we raise the graph to power β. This new graph is a much better expander (but the degree had deteriorated). Now, consider a specific input x, and mark the bad vertices for it in the graph G. Clearly, we mark at most $1/100$ fraction of the vertices. Conceptually, think about these vertices as being uniformly spread in the graph and far apart. From the execution of the algorithm to fail, the random walk needs to visit $\alpha k/2$ bad vertices in the random walk in G^k. However, the probability for that is extremely small - why would the random walk keep stumbling into bad vertices, when they are so infrequent?

The real thing. Let Q be the transition matrix of G. We assume, as usual, that the random walk on G is aperiodic (if not, we can easily fix it using standard tricks), and thus ergodic. Let $\mathsf{B} = \mathsf{Q}^\beta$ be the transition matrix of the random walk of the states we use in the algorithm. Note, that the eigenvalues (except the first one) of B "shrink". In particular, by picking β to be a sufficiently large constant, we have that

$$\widehat{\lambda_1}(\mathsf{B}) = 1 \quad \text{and} \quad |\widehat{\lambda_i}(\mathsf{B})| \leq \frac{1}{10}, \qquad \text{for } i = 2, \ldots, U.$$

For the input string x, let W be the matrix that has 1 in the diagonal entry W_{ii}, if and only $\mathbf{Alg}(x, \mathsf{s}(i))$ returns the right answer, for $i = 1, \ldots, U$. (We remind the reader that $\mathsf{s}(i)$ is the string associated with the ith vertex, and $U = |V(\mathsf{G})|$.) The matrix W is zero everywhere else. Similarly, let $\overline{\mathsf{W}} = \mathcal{I} - \mathsf{W}$ be the "complement" matrix having 1 at $\overline{\mathsf{W}}_{ii}$ iff $\mathbf{Alg}(x, \mathsf{s}(i))$ is incorrect. We know that W is a $U \times U$ matrix, that has at least $(99/100)U$ ones on its diagonal.

Lemma 34.3.1. *Let Q be a symmetric transition matrix, then all its eigenvalues of Q are in the range $[-1, 1]$.*

Proof: Let $p \in \mathbb{R}^n$ be an eigenvector with eigenvalue λ. Let p_i be the coordinate with the maximum absolute value in p. We have that

$$\left|\lambda p_i\right| = \left|(p\mathsf{Q})_i\right| = \left|\sum_{j=1}^{U} p_j \mathsf{Q}_{ji}\right| \leq \sum_{j=1}^{U} \left|p_j\right| \left|\mathsf{Q}_{ji}\right| \leq |p_i| \sum_{j=1}^{U} \left|\mathsf{Q}_{ji}\right| = \left|p_i\right|.$$

This implies that $|\lambda| \leq 1$.

(We used the symmetry of the matrix, in implying that Q eigenvalues are all real numbers.) ■

Lemma 34.3.2. *Let* Q *be a symmetric transition matrix, then for any* $p \in \mathbb{R}^n$*, we have that* $\|p\mathsf{Q}\|_2 \leq \|p\|_2$.

Proof: Let $\mathcal{B}(\mathsf{Q}) = \langle v_1, \ldots, v_n \rangle$ denote the orthonormal eigenvector basis of Q, with eigenvalues $1 = \lambda_1, \ldots, \lambda_n$. Write $p = \sum_i \alpha_i v_i$, and observe that

$$\left\|p\mathsf{Q}\right\|_2 = \left\|\sum_i \alpha_i v_i \mathsf{Q}\right\|_2 = \left\|\sum_i \alpha_i \lambda_i v_i\right\|_2 = \sqrt{\sum_i \alpha_i^2 \lambda_i^2} \leq \sqrt{\sum_i \alpha_i^2} = \left\|p\right\|_2,$$

since $|\lambda_i| \leq 1$, for $i = 1, \ldots, n$, by Lemma 34.3.1. ■

Lemma 34.3.3. *Let* $\mathsf{B} = \mathsf{Q}^\beta$ *be the transition matrix of the graph* G^β*. For all vectors* $p \in \mathbb{R}^n$*, we have:*

(i) $\|p\mathsf{B}\mathsf{W}\| \leq \|p\|$*, and*

(ii) $\left\|p\mathsf{B}\overline{\mathsf{W}}\right\| \leq \|p\|/5$.

Proof: (i) Since multiplying a vector by W has the effect of zeroing out some coordinates, its clear that it can not enlarge the norm of a matrix. As such, $\|p\mathsf{B}\mathsf{W}\|_2 \leq \|p\mathsf{B}\|_2 \leq \|p\|_2$ by Lemma 34.3.2.

(ii) Write $p = \sum_i \alpha_i v_i$, where $v_1, \ldots, v_n$ is the orthonormal basis of Q (and thus also of B), with eigenvalues $1 = \widehat{\lambda_1}, \ldots, \widehat{\lambda_n}$. We remind the reader that $v_1 = (1, 1, \ldots, 1)/\sqrt{n}$. Since $\overline{\mathsf{W}}$ zeroes out at least $99/100$ of the entries of a vectors it is multiplied by (and copy the rest as they are), we have that

$$\|v_1\overline{\mathsf{W}}\| \leq \sqrt{(n/100)(1/\sqrt{n})^2} \leq 1/10 = \|v_1\|/10.$$

Now, for any $x \in \mathbb{R}^U$, we have $\|x\overline{\mathsf{W}}\| \leq \|x\|$. As such, we have that

$$\begin{aligned}
\left\|p\mathsf{B}\overline{\mathsf{W}}\right\|_2 &= \left\|\sum_i \alpha_i v_i \mathsf{B}\overline{\mathsf{W}}\right\|_2 \leq \left\|\alpha_1 v_1 \mathsf{B}\overline{\mathsf{W}}\right\| + \left\|\sum_{i=2}^{U} \alpha_i v_i \mathsf{B}\overline{\mathsf{W}}\right\| \\
&\leq \left\|\alpha_1 v_1 \overline{\mathsf{W}}\right\| + \left\|\Big(\sum_{i=2}^{U} \alpha_i v_i \widehat{\lambda_i}^\beta\Big)\overline{\mathsf{W}}\right\| \leq \frac{|\alpha_1|}{10} + \left\|\sum_{i=2}^{U} \alpha_i v_i \widehat{\lambda_i}^\beta\right\| \\
&\leq \frac{|\alpha_1|}{10} + \sqrt{\sum_{i=2}^{U} \left(\alpha_i \widehat{\lambda_i}^\beta\right)^2} \leq \frac{|\alpha_1|}{10} + \frac{1}{10}\sqrt{\sum_{i=2}^{U} \alpha_i^2} \leq \frac{\|p\|}{10} + \frac{1}{10}\|p\| \leq \frac{\|p\|}{5},
\end{aligned}$$

since $\left|\lambda_i^\beta\right| \leq 1/10$, for $i = 2, \ldots, n$. ■

Consider the strings $r_0, \ldots, r_\nu$. For each one of these strings, we can write down whether its a "good" string (i.e., **Alg** return the correct result), or a bad string. This results in a binary pattern $b_0, \ldots, b_k$. Given a distribution $p \in \mathbb{R}^U$ on the states of the graph, its natural to ask what is the probability of being in a "good" state. Clearly, this is the quantity $\|p\mathsf{W}\|_1$. Thus, if we are interested in the probability of a specific pattern, then we should start with the initial distribution p^0, truncate away the coordinates that represent an invalid state, apply the transition matrix, again truncate away forbidden coordinates, and repeat in this fashion till we exhaust the pattern. Clearly, the ℓ_1-norm of the resulting vector is the probability of this pattern. To this end, given a pattern $b_0, \ldots, b_k$, let $\mathcal{S} = \langle S_0, \ldots, S_\nu \rangle$ denote the corresponding sequence of "truncating" matrices (i.e., S_i is either W or $\overline{\mathsf{W}}$). Formally, we set $S_i = \mathsf{W}$ if **Alg**(x, r_i) returns the correct answer, and set $S_i = \overline{\mathsf{W}}$ otherwise.

The above argument implies the following lemma.

Lemma 34.3.4. *For any fixed pattern* $b_0, \ldots, b_\nu$ *the probability of the random walk to generate this pattern of random strings is* $\|p^{(0)} S_0 \mathsf{B} S_1 \ldots \mathsf{B} S_\nu\|_1$*, where* $\mathcal{S} = \langle S_0, \ldots, S_\nu \rangle$ *is the sequence of* W *and* $\overline{\mathsf{W}}$ *encoded by this pattern.*

Theorem 34.3.5. *The probability that the majority of the outputs* $\mathbf{Alg}(x, r_0), \mathbf{Alg}(x, r_1), \ldots, \mathbf{Alg}(x, r_k)$ *is incorrect is at most* $1/2^k$.

Proof: The majority is wrong, only if (at least) half the elements of the sequence $\mathcal{S} = \langle S_0, \ldots, S_\nu \rangle$ belong to $\overline{\mathsf{W}}$. Fix such a "bad" sequence $\mathcal{S}$, and observe that the distributions we work with are vectors in $\mathbb{R}^U$. As such, if p^0 is the initial distribution, then we have that

$$\mathbb{P}[\mathcal{S}] = \left\| p^{(0)} S_0 \mathsf{B} S_1 \ldots \mathsf{B} S_\nu \right\|_{\mathbf{1}} \leq \sqrt{U} \left\| p^{(0)} S_0 \mathsf{B} S_1 \ldots \mathsf{B} S_\nu \right\|_{\mathbf{2}} \leq \sqrt{U} \frac{1}{5^{\nu/2}} \left\| p^{(0)} \right\|_2,$$

by Lemma 34.3.6 below (i.e., Cauchy-Schwarz inequality) and by repeatedly applying Lemma 34.3.3, since half of the sequence $\mathcal{S}$ are $\overline{\mathsf{W}}$, and the rest are W. The distribution $p^{(0)}$ was uniform, which implies that $\left\| p^{(0)} \right\|_2 = 1/\sqrt{U}$. As such, let $\mathcal{S}$ be the set of all bad patterns (there are $2^{\nu-1}$ such "bad" patterns). We have

$$\mathbb{P}\Big[\text{ majority is bad } \Big] \leq 2^\nu \sqrt{U} \frac{1}{5^{\nu/2}} \left\| p^{(0)} \right\|_2 = (4/5)^{\nu/2} = (4/5)^{\alpha k/2} \leq \frac{1}{2^k},$$

for $\alpha = 7$. ■

34.3.2. Some standard inequalities

Lemma 34.3.6. *For any vector* $\mathbf{v} = (v_1, \ldots, v_d) \in \mathbb{R}^d$, *we have that* $\|\mathbf{v}\|_1 \leq \sqrt{d}\, \|\mathbf{v}\|_2$.

Proof: We can safely assume all the coordinates of $\mathbf{v}$ are positive. Now,

$$\|\mathbf{v}\|_1 = \sum_{i=1}^{d} v_i = \sum_{i=1}^{d} v_i \cdot 1 = |\mathbf{v} \cdot (1, 1, \ldots, 1)| \leq \sqrt{\sum_{i=1}^{d} v_i^2} \sqrt{\sum_{i=1}^{d} 1^2} = \sqrt{d} \left\| \mathbf{v} \right\|,$$

by the Cauchy-Schwarz inequality. ■

References

[GG81] O. Gabber and Z. Galil. Explicit constructions of linear-sized superconcentrators. *J. Comput. Syst. Sci.*, 22(3): 407–420, 1981.

Chapter 35

Complexity classes

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

"I'm a simple man, a guileless man," Panin answered. "There is a norm. The norm is five gravities. My simple, uncomplicated organism cannot bear anything exceeding the norm. My organism tried six once, and got carried out at six minutes some seconds. With me along."

Almost the same, Arkady and Boris Strugatsky

35.1. Las Vegas and Monte Carlo algorithms

Definition 35.1.1. A ***Las Vegas algorithm*** is a randomized algorithms that *always* return the correct result. The only variant is that it's running time might change between executions.

An example for a Las Vegas algorithm is the **QuickSort** algorithm.

Definition 35.1.2. A ***Monte Carlo algorithm*** is a randomized algorithm that might output an incorrect result. However, the probability of error can be diminished by repeated executions of the algorithm.

The matrix multiplication algorithm is an example of a Monte Carlo algorithm.

35.2. Complexity classes

I assume people know what are Turing machines, **NP**, **NPC**, RAM machines, uniform model, logarithmic model. **PSPACE**, and **EXP**. If you do now know what are those things, you should read about them. Some of that is covered in the randomized algorithms book, and some other stuff is covered in any basic text on complexity theory[①].

Definition 35.2.1. The class **P** consists of all languages L that have a polynomial time algorithm **Alg**, such that for any input Σ^*, we have
(A) $x \in L \Rightarrow$ **Alg**(x) accepts,
(B) $x \notin L \Rightarrow$ **Alg**(x) rejects.

Definition 35.2.2. The class **NP** consists of all languages L that have a polynomial time algorithm **Alg**, such that for any input Σ^*, we have:
(i) If $x \in L \Rightarrow$ then $\exists y \in \Sigma^*$, **Alg**(x, y) accepts, where $|y|$ (i.e. the length of y) is bounded by a polynomial in $|x|$.

[①] There is also the internet.

(ii) If $x \notin L \Rightarrow$ then $\forall y \in \Sigma^* \mathbf{Alg}(x, y)$ rejects.

Definition 35.2.3. For a complexity class $\mathcal{C}$, we define the complementary class co-$\mathcal{C}$ as the set of languages whose complement is in the class $\mathcal{C}$. That is

$$\text{co}-\mathcal{C} = \{L \mid \overline{L} \in \mathcal{C}\},$$

4 where $\overline{L} = \Sigma^* \setminus L$.

It is obvious that $\mathbf{P} = \text{co}-\mathbf{P}$ and $\mathbf{P} \subseteq \mathbf{NP} \cap \text{co}-\mathbf{NP}$. (It is currently unknown if $\mathbf{P} = \mathbf{NP} \cap \text{co}-\mathbf{NP}$ or whether $\mathbf{NP} = \text{co}-\mathbf{NP}$, although both statements are believed to be false.)

Definition 35.2.4. The class **RP** (for Randomized Polynomial time) consists of all languages L that have a randomized algorithm **Alg** with worst case polynomial running time such that for any input $x \in \Sigma^*$, we have

(i) If $x \in L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] \geq 1/2$.
(ii) $x \notin L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] = 0$.

An **RP** algorithm is a Monte Carlo algorithm, but this algorithm can make a mistake only if $x \in L$. As such, co−**RP** is all the languages that have a Monte Carlo algorithm that make a mistake only if $x \notin L$. A problem which is in $\mathbf{RP} \cap \text{co}-\mathbf{RP}$ has an algorithm that does not make a mistake, namely a Las Vegas algorithm.

Definition 35.2.5. The class **ZPP** (for Zero-error Probabilistic Polynomial time) is the class of languages that have a Las Vegas algorithm that runs in expected polynomial time.

Definition 35.2.6. The class **PP** (for Probabilistic Polynomial time) is the class of languages that have a randomized algorithm **Alg**, with worst case polynomial running time, such that for any input $x \in \Sigma^*$, we have

(i) If $x \in L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] > 1/2$.
(ii) If $x \notin L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] < 1/2$.

The class **PP** is not very useful. Why?

Exercise 35.2.7. Provide a **PP** algorithm for 3SAT.

Consider the mind-boggling stupid randomized algorithm that returns either yes or no with probability half. This algorithm is almost in **PP**, as it return the correct answer with probability half. An algorithm is in **PP** needs to be *slightly* better, and be correct with probability better than half. However, how much better can be made to be arbitrarily close to $1/2$. In particular, there is no way to do effective amplification with such an algorithm.

Definition 35.2.8. The class **BPP** (for Bounded-error Probabilistic Polynomial time) is the class of languages that have a randomized algorithm **Alg** with worst case polynomial running time such that for any input $x \in \Sigma^*$, we have

(i) If $x \in L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] \geq 3/4$.
(ii) If $x \notin L$ then $\mathbb{P}[\mathbf{Alg}(x) \text{ accepts}] \leq 1/4$.

35.3. Bibliographical notes

Section 35.1 follows [MR95, Section 1.5].

References

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

Chapter 36

Backwards analysis

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

The idea of ***backwards analysis*** (or backward analysis) is a technique to analyze randomized algorithms by imagining as if it was running backwards in time, from output to input. Most of the more interesting applications of backward analysis are in Computational Geometry, but nevertheless, there are some other applications that are interesting and we survey some of them here.

36.1. How many times can the minimum change?

Let $\Pi = \pi_1 \dots \pi_n$ be a random permutation of $\{1, \dots, n\}$. Let $\mathcal{E}_i$ be the event that π_i is the minimum number seen so far as we read Π; that is, $\mathcal{E}_i$ is the event that $\pi_i = \min_{k=1}^{i} \pi_k$. Let X_i be the indicator variable that is one if $\mathcal{E}_i$ happens. We already seen, and it is easy to verify, that $\mathbb{E}[X_i] = 1/i$. We are interested in how many times the minimum might change[①]; that is $Z = \sum_i X_i$, and how concentrated is the distribution of Z. The following is maybe surprising.

Lemma 36.1.1. *The events $\mathcal{E}_1, \dots, \mathcal{E}_n$ are independent (as such, variables $X_1, \dots, X_n$ are independent).*

Proof: The trick is to think about the sampling process in a different way, and then the result readily follows. Indeed, we randomly pick a permutation of the given numbers, and set the first number to be π_n. We then, again, pick a random permutation of the remaining numbers and set the first number as the penultimate number (i.e., π_{n-1}) in the output permutation. We repeat this process till we generate the whole permutation.

Now, consider $1 \le i_1 < i_2 < \dots < i_k \le n$, and observe that $\mathbb{P}\Big[\mathcal{E}_{i_1} \Big| \mathcal{E}_{i_2} \cap \dots \cap \mathcal{E}_{i_k}\Big] = \mathbb{P}[\mathcal{E}_{i_1}]$, since by our thought experiment, $\mathcal{E}_{i_1}$ is determined after all the other variables $\mathcal{E}_{i_2}, \dots, \mathcal{E}_{i_k}$. In particular, the variable $\mathcal{E}_{i_1}$ is inherently not effected by these events happening or not. As such, we have

$$\begin{aligned}\mathbb{P}\Big[\mathcal{E}_{i_1} \cap \mathcal{E}_{i_2} \cap \dots \cap \mathcal{E}_{i_k}\Big] &= \mathbb{P}\Big[\mathcal{E}_{i_1} \Big| \mathcal{E}_{i_2} \cap \dots \cap \mathcal{E}_{i_k}\Big] \mathbb{P}[\mathcal{E}_{i_2} \cap \dots \cap \mathcal{E}_{i_k}] \\ &= \mathbb{P}\Big[\mathcal{E}_{i_1}\Big] \mathbb{P}\Big[\mathcal{E}_{i_2} \cap \mathcal{E}_{i_2} \cap \dots \cap \mathcal{E}_{i_k}\Big] = \prod_{j=1}^{k} \mathbb{P}\Big[\mathcal{E}_{i_j}\Big] = \prod_{j=1}^{k} \frac{1}{i_j},\end{aligned}$$

by induction. ■

Theorem 36.1.2. *Let $\Pi = \pi_1 \dots \pi_n$ be a random permutation of $1, \dots, n$, and let Z be the number of times, that π_i is the smallest number among $\pi_1, \dots, \pi_i$, for $i = 1, \dots, n$. Then, we have that for $t \ge 2e$ that $\mathbb{P}[Z > t \ln n] \le 1/n^{t \ln 2}$, and for $t \in [1, 2e]$, we have that $\mathbb{P}[Z > t \ln n] \le 1/n^{(t-1)^2/4}$.*

[①] The answer, my friend, is blowing in the permutation.

Proof: Follows readily from Chernoff's inequality, as $Z = \sum_i X_i$ is a sum of independent indicator variables, and, since by linearity of expectations, we have

$$\mu = \mathbb{E}\Big[Z\Big] = \sum_i \mathbb{E}\Big[X_i\Big] = \sum_{i=1}^{n} \frac{1}{i} \geq \int_{x=1}^{n+1} \frac{1}{x} \mathrm{d}x = \ln(n+1) \geq \ln n.$$

Next, we set $\delta = t - 1$, and use Chernoff inequality. ■

36.2. Computing a good ordering of the vertices of a graph

We are given a $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be an edge-weighted graph with n vertices and m edges. The task is to compute an ordering $\pi = \langle \pi_1, \ldots, \pi_n \rangle$ of the vertices, and for every vertex $v \in \mathsf{V}$, the list of vertices L_v, such that $\pi_i \in \text{Ł}_v$, if π_i is the closet vertex to v in the ith prefix $\langle \pi_1, \ldots, \pi_i \rangle$.

This situation can arise for example in a streaming scenario, where we install servers in a network. In the ith stage there i servers installed, and every client in the network wants to know its closest server. As we install more and more servers (ultimately, every node is going to be server), each client needs to maintain its current closest server.

The purpose is to minimize the total size of these lists $\mathcal{L} = \sum_{v \in \mathsf{V}} |L_v|$.

36.2.1. The algorithm

Take a random permutation $\pi_1, \ldots, \pi_n$ of the vertices V of G. Initially, we set $\delta(v) = +\infty$, for all $v \in \mathsf{V}$.

In the ith iteration, set $\delta(\pi_i)$ to 0, and start Dijkstra from the ith vertex π_i. The Dijkstra propagates only if it improves the current distance associated with a vertex. Specifically, in the ith iteration, we update $\delta(u)$ to $\mathsf{d}_{\mathsf{G}}(\pi_i, u)$ if and only if $\mathsf{d}_{\mathsf{G}}(\pi_i, u) < \delta(u)$ before this iteration started. If $\delta(u)$ is updated, then we add π_i to L_u. Note, that this Dijkstra propagation process might visit only small portions of the graph in some iterations – since it improves the current distance only for few vertices.

36.2.2. Analysis

Lemma 36.2.1. *The above algorithm computes a permutation π, such that $\mathbb{E}\Big[|\mathcal{L}|\Big] = O(n \log n)$, and the expected running time of the algorithm is $O\Big((n \log n + m) \log n\Big)$, where $n = |V(\mathsf{G})|$ and $m = |\mathsf{E}(\mathsf{G})|$. Note, that both bounds also hold with high probability.*

Proof: Fix a vertex $v \in \mathsf{V} = \{v_1, \ldots, v_n\}$. Consider the set of n numbers $\{\mathsf{d}_{\mathsf{G}}(v, v_1), \ldots, \mathsf{d}_{\mathsf{G}}(v, v_n)\}$. Clearly, $\mathsf{d}_{\mathsf{G}}(v, \pi_1), \ldots, \mathsf{d}_{\mathsf{G}}(v, \pi_n)$ is a random permutation of this set, and by Lemma 36.1.1 the random permutation π changes this minimum $O(\log n)$ time in expectations (and also with high probability). This readily implies that $|L_v| = O(\log n)$ both in expectations and high probability.

The more interesting claim is the running time. Consider an edge $uv \in \mathsf{E}(\mathsf{G})$, and observe that $\delta(u)$ or $\delta(v)$ changes $O(\log n)$ times. As such, an edge gets visited $O(\log n)$ times, which implies overall running time of $O(n \log^2 n + m \log n)$, as desired.

Indeed, overall there are $O(n \log n)$ changes in the value of $\delta(\cdot)$. Each such change might require one **delete-min** operation from the queue, which takes $O(\log n)$ time operation. Every edge, by the above, might trigger $O(\log n)$ **decrease-key** operations. Using Fibonacci heaps, each such operation takes $O(1)$ time. ■

36.3. Computing nets

36.3.1. Basic definitions

Definition 36.3.1. A ***metric space*** is a pair $(\mathcal{X}, \mathsf{d})$ where $\mathcal{X}$ is a set and $\mathsf{d} : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ is a ***metric*** satisfying the following axioms: (i) $\mathsf{d}(x, y) = 0$ if and only if $x = y$, (ii) $\mathsf{d}(x, y) = \mathsf{d}(y, x)$, and (iii) $\mathsf{d}(x, y) + \mathsf{d}(y, z) \geq \mathsf{d}(x, z)$ (triangle inequality).

For example, $\mathbb{R}^2$ with the regular Euclidean distance is a metric space. In the following, we assume that we are given ***black-box access*** to $\mathsf{d}_{\mathcal{M}}$. Namely, given two points $\mathsf{p}, \mathsf{u} \in \mathcal{X}$, we assume that $\mathsf{d}(\mathsf{p}, \mathsf{u})$ can be computed in constant time.

Another standard example for a finite metric space is a graph G with non-negative weights $\omega(\cdot)$ defined on its edges. Let $\mathsf{d}_{\mathsf{G}}(x, y)$ denote the shortest path (under the given weights) between any $x, y \in V(\mathsf{G})$. It is easy to verify that $\mathsf{d}_{\mathsf{G}}(\cdot, \cdot)$ is a metric. In fact, any ***finite metric*** (i.e., a metric defined over a finite set) can be represented by such a weighted graph.

36.3.1.1. Nets

Definition 36.3.2. For a point set P in a metric space with a metric d, and a parameter $r > 0$, an ***r-net*** of P is a subset $C \subseteq \mathsf{P}$, such that

(i) for every $\mathsf{p}, \mathsf{u} \in C$, $\mathsf{p} \neq \mathsf{u}$, we have that $\mathsf{d}(\mathsf{p}, \mathsf{u}) \geq r$, and
(ii) for all $\mathsf{p} \in \mathsf{P}$, we have that $\min_{\mathsf{u} \in C} \mathsf{d}(\mathsf{p}, \mathsf{u}) < r$.

Intuitively, an r-net represents P in resolution r.

36.3.2. Computing an r-net in a sparse graph

Given a $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be an edge-weighted graph with n vertices and m edges, and let $r > 0$ be a parameter. We are interested in the problem of computing an r-net for G. That is, a set of vertices of G that complies with Definition 36.3.2$_{\text{p227}}$.

36.3.2.1. The algorithm

We compute an r-net in a sparse graph using a variant of Dijkstra's algorithm with the sequence of starting vertices chosen in a random permutation.

Let π_i be the ith vertex in a random permutation π of V. For each vertex v we initialize $\delta(v)$ to $+\infty$. In the ith iteration, we test whether $\delta(\pi_i) \geq r$, and if so we do the following steps:

(A) Add π_i to the resulting net $\mathcal{N}$.
(B) Set $\delta(\pi_i)$ to zero.
(C) Perform Dijkstra's algorithm starting from π_i, modified to avoid adding a vertex u to the priority queue unless its tentative distance is smaller than the current value of $\delta(u)$. When such a vertex u is expanded, we set $\delta(u)$ to be its computed distance from π_i, and relax the edges adjacent to u in the graph.

36.3.2.2. Analysis

While the analysis here does not directly uses backward analysis, it is inspired to a large extent by such an analysis as in Section 36.2$_{\text{p226}}$.

Lemma 36.3.3. *The set $\mathcal{N}$ is an r-net in G.*

Proof: By the end of the algorithm, each $v \in \mathsf{V}$ has $\delta(v) < r$, for $\delta(v)$ is monotonically decreasing, and if it were larger than r when v was visited then v would have been added to the net.

An induction shows that if $\ell = \delta(v)$, for some vertex v, then the distance of v to the set $\mathcal{N}$ is at most ℓ. Indeed, for the sake of contradiction, let j be the (end of) the first iteration where this claim is false. It must be that $\pi_j \in \mathcal{N}$, and it is the nearest vertex in $\mathcal{N}$ to v. But then, consider the shortest path between π_j and v. The modified Dijkstra must have visited all the vertices on this path, thus computing $\delta(v)$ correctly at this iteration, which is a contradiction.

Finally, observe that every two points in $\mathcal{N}$ have distance $\geq r$. Indeed, when the algorithm handles vertex $v \in \mathcal{N}$, its distance from all the vertices currently in $\mathcal{N}$ is $\geq r$, implying the claim. ■

Lemma 36.3.4. *Consider an execution of the algorithm, and any vertex $v \in \mathsf{V}$. The expected number of times the algorithm updates the value of $\delta(v)$ during its execution is $O(\log n)$, and more strongly the number of updates is $O(\log n)$ with high probability.*

Proof: For simplicity of exposition, assume all distances in G are distinct. Let S_i be the set of all the vertices $x \in \mathsf{V}$, such that the following two properties both hold:

(A) $\mathsf{d}_{\mathsf{G}}(x, v) < \mathsf{d}_{\mathsf{G}}(v, \Pi_i)$, where $\Pi_i = \{\pi_1, \ldots, \pi_i\}$.

(B) If $\pi_{i+1} = x$ then $\delta(v)$ would change in the $(i+1)$th iteration.

Let $s_i = |S_i|$. Observe that $S_1 \supseteq S_2 \supseteq \cdots \supseteq S_n$, and $|S_n| = 0$.

In particular, let $\mathcal{E}_{i+1}$ be the event that $\delta(v)$ changed in iteration $(i+1)$ – we will refer to such an iteration as being ***active***. If iteration $(i+1)$ is active then one of the points of S_i is π_{i+1}. However, π_{i+1} has a uniform distribution over the vertices of S_i, and in particular, if $\mathcal{E}_{i+1}$ happens then $s_{i+1} \leq s_i/2$, with probability at least half, and we will refer to such an iteration as being ***lucky***. (It is possible that $s_{i+1} < s_i$ even if $\mathcal{E}_{i+1}$ does not happen, but this is only to our benefit.) After $O(\log n)$ lucky iterations the set S_i is empty, and we are done. Clearly, if both the ith and jth iteration are active, the events that they are each lucky are independent of each other. By the Chernoff inequality, after $c \log n$ active iterations, at least $\lceil \log_2 n \rceil$ iterations were lucky with high probability, implying the claim. Here c is a sufficiently large constant. ■

Interestingly, in the above proof, all we used was the monotonicity of the sets $S_1, \ldots, S_n$, and that if $\delta(v)$ changes in an iteration then the size of the set S_i shrinks by a constant factor with good probability in this iteration. This implies that there is some flexibility in deciding whether or not to initiate Dijkstra's algorithm from each vertex of the permutation, without damaging the number of times of the values of $\delta(v)$ are updated.

Theorem 36.3.5. *Given a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, with n vertices and m edges, the above algorithm computes an r-net of G in $O((n \log n + m) \log n)$ expected time.*

Proof: By Lemma 36.3.4, the two δ values associated with the endpoints of an edge get updated $O(\log n)$ times, in expectation, during the algorithm's execution. As such, a single edge creates $O(\log n)$ decrease-key operations in the heap maintained by the algorithm. Each such operation takes constant time if we use Fibonacci heaps to implement the algorithm. ■

36.4. Bibliographical notes

Backwards analysis was invented/discovered by Raimund Seidel, and the **QuickSort** example is taken from Seidel [Sei93]. The number of changes of the minimum result of Section 36.1 is by now folklore.

The good ordering of Section 36.2 is probably also folklore, although a similar idea was used by Mendel and Schwob [MS09] for a different problem.

Computing a net in a sparse graph, Section 36.3.2, is from [EHS14]. While backwards analysis fails to hold in this case, it provide a good intuition for the analysis, which is slightly more complicated and indirect.

References

[EHS14] D. Eppstein, S. Har-Peled, and A. Sidiropoulos. *On the Greedy Permutation and Counting Distances*. manuscript. 2014.

[MS09] M. Mendel and C. Schwob. Fast c-k-r partitions of sparse graphs. *Chicago J. Theor. Comput. Sci.*, 2009, 2009.

[Sei93] R. Seidel. Backwards analysis of randomized geometric algorithms. *New Trends in Discrete and Computational Geometry*. Ed. by J. Pach. Vol. 10. Algorithms and Combinatorics. Springer-Verlag, 1993, pp. 37–68.

Chapter 37

Multiplicative Weight Update: Expert Selection

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> Possession of anything new or expensive only reflected a person's lack of theology and geometry; it could even cast doubts upon one's soul.
>
> A confederacy of Dunces, John Kennedy Toole

37.1. The problem: Expert selection

We are given N experts $[\![N]\!] = \{1, 2, \ldots, N\}$. At each time t, an expert i makes a prediction what is going to happen at this time slot. To make things simple, assume the prediction is one of two values, say, 0 or 1. You are going to play this game for a while – at each iteration you are going to get the advice of the N experts, and you are going to select either decision as your own prediction. The purpose here is to come up with a strategy that minimizes the overall number of wrong predictions made.

If there is an expert that is never wrong. This situation is easy – initially start with all n experts as being viable – to this end, we assign $W(i) \leftarrow 1$, for all i. If an expert prediction turns out to be wrong, we set its weight to zero (i.e., it is no longer active). Clearly, if you follow the majority vote of the still viable experts, then at most $\log_2 n$ mistakes would be made, before one isolates the infallible experts.

37.2. Majority vote

The algorithm. Unfortunately, we are unlikely to be in the above scenario – experts makes mistakes. Throwing a way an expert because of a single mistake is a sure way to have no expert remaining. Instead, we are going to moderate our strategy. If expert i is wrong, in a round, we are going to decrease its weight – to be precise, we set $W(i) \leftarrow (1 - \varepsilon)W(i)$, where ε is some parameter. Note, that this weight update is done every round, independent on the decision output in the round. It is now natural, in each round, to compute the total weight of the experts predicting 0, and the total weight of the experts predicting 1, and return the prediction that has a heavier total weight supporting it.

Intuition. The algorithm keeps track of the quality of the experts. The useless experts would have weights very close to zero.

Analysis. We need the following easy calculation.

Lemma 37.2.1. *For $x \in [0, 1/2]$, we have $1 - x \geq \exp(-x - x^2)$.*

Proof: For $x \in (-1, 1)$, the Taylor expansion of $\ln(1 + x)$ is $\sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i}$. As such, for $x \in [0, 1/2]$ we have

$$\ln(1 - x) = -\sum_{i=1}^{\infty} \frac{x^i}{i} = -x - \frac{x^2}{2} - \frac{x^3}{3} \cdots \geq -x - x^2,$$

since $x^{2+i}/(2 + i) \leq x^2/2^i \iff x^i/(2 + i) \leq 1/2^i$, which is obviously true as $x \leq 1/2$. ■

Lemma 37.2.2. *Let assume we have N experts. Let $\boldsymbol{\beta}_t$ be the number of the mistakes the algorithm performs, and let $\beta_t(i)$ be the number of mistakes made by the ith expert, for $i \in [\![n]\!]$ (both till time t). Then, if we run this algorithm for T rounds, we have*

$$\forall i \in [\![n]\!] \qquad \boldsymbol{\beta}_T \leq 2(1 + \varepsilon)\beta_T(i) + \frac{2 \log N}{\varepsilon}.$$

Proof: Let Φ_t be the total weight of the experts at the beginning of round t. Observe that $\Phi_1 = N$, and if a mistake was made in the t round, then

$$\Phi_{t+1} \leq (1 - \varepsilon/2)\Phi_t \leq \exp(-\varepsilon \boldsymbol{\beta}_{t+1}/2)N.$$

On the other hand, an expert i made $\beta_i(t)$ mistakes in the first t rounds, and as such its weight, at this point in time, is $(1 - \varepsilon)^{\beta_t(i)}$. We thus have, at time T, and for any i, that

$$\exp\Big(-\Big(\varepsilon + \varepsilon^2\Big)\beta_T(i)\Big) \leq (1 - \varepsilon)^{\beta_T(i)} \leq \Phi_T \leq \exp\left(-\frac{\varepsilon \boldsymbol{\beta}_T}{2}\right)N.$$

Taking ln of both sides, we have $-\Big(\varepsilon + \varepsilon^2\Big)\beta_T(i) \leq -\frac{\varepsilon \boldsymbol{\beta}_T}{2} + \ln N. \iff \boldsymbol{\beta}_T \leq 2(1 + \varepsilon)\beta_T(i) + 2\frac{\ln N}{\varepsilon}$. ■

37.3. Randomized weighted majority

Let $W_t(i)$ be the weight assigned to the ith expert with in the beginning of the tth round. We modify the algorithm to choose expert i, at round t, with probability $W_t(i)/\Phi_t$. That is, the algorithm randomly choose an expert to follow according to their weights. Unlike before, all the experts that are wrong in a round get a weight decrease.

Proof: We have that $\Phi_t = \sum_{i=1}^{N} W_t(i)$. Let $m_t(i) = 1$ be a an indicator variable that is one if and only if expert i made a mistake at round t. Similarly, let $\mathbf{m}_t = 1 \iff$ the algorithm made a mistake at round t. By definition, we have that

$$\mathbb{E}[\mathbf{m}_t] = \sum_{i=1}^{N} \mathbb{P}[i \text{ expert chosen}]\, m_t(i) = \sum_{i=1}^{N} \frac{W_t(i)}{\Phi_t}\, m_t(i).$$

We then have that

$$W_{t+1}(i) = (1 - \varepsilon m_t(i))W_t(i).$$

As such, we have $\Phi_{t+1} = \sum_{i=1}^{N} W_{t+1}(i)$, and

$$\Phi_{t+1} = \sum_{i=1}^{N} (1 - \varepsilon m_t(i))W_t(i) = \Phi_t - \varepsilon \sum_{i=1}^{N} m_t(i)W_t(i) = \Phi_t - \varepsilon\Phi_t \sum_{i=1}^{N} m_t(i)\frac{W_t(i)}{\Phi_t} = \Big(1 - \varepsilon\, \mathbb{E}[\mathbf{m}_t]\Big)\Phi_t.$$

We now follow the same argument as before

$$(1 - \varepsilon)^{\beta_T(i)} \leq \Phi_T \leq N \prod_{t=1}^{T} \Big(1 - \varepsilon\, \mathbb{E}[\mathbf{m}_t]\Big) \leq N \exp(-\varepsilon\, \mathbb{E}[\boldsymbol{\beta}_T]) \implies (-\varepsilon - \varepsilon^2)\beta_T(i) \leq \ln N - \varepsilon\, \mathbb{E}[\boldsymbol{\beta}_T]$$

$$\implies \mathbb{E}[\boldsymbol{\beta}_T] \leq (1 + \varepsilon)\beta_T(i) + \frac{\ln N}{\varepsilon}.$$ ■

37.4. Bibliographical notes

Chapter 38

On Complexity, Sampling, and ε-Nets and ε-Samples

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "I've never touched the hard stuff, only smoked grass a few times with the boys to be polite, and that's all, though ten is the age when the big guys come around teaching you all sorts to things. But happiness doesn't mean much to me, I still think life is better. Happiness is a mean son of a bitch and needs to be put in his place. Him and me aren't on the same team, and I'm cutting him dead. I've never gone in for politics, because somebody always stand to gain by it, but happiness is an even crummier racket, and their ought to be laws to put it out of business."
>
> Momo, Emile Ajar

In this chapter we will try to quantify the notion of geometric complexity. It is intuitively clear that a ● (i.e., disk) is a simpler shape than an ⬬ (i.e., ellipse), which is in turn simpler than a ☻ (i.e., smiley). This becomes even more important when we consider several such shapes and how they interact with each other. As these examples might demonstrate, this notion of complexity is somewhat elusive.

To this end, we show that one can capture the structure of a distribution/point set by a small subset. The size here would depend on the complexity of the shapes/ranges we care about, but surprisingly it would be independent of the size of the point set.

38.1. VC dimension

Definition 38.1.1. A ***range space*** S is a pair $(\mathsf{X}, \mathcal{R})$, where X is a ***ground set*** (finite or infinite) and $\mathcal{R}$ is a (finite or infinite) family of subsets of X. The elements of X are ***points*** and the elements of $\mathcal{R}$ are ***ranges***.

Our interest is in the size/weight of the ranges in the range space. For technical reasons, it will be easier to consider a finite subset x as the underlining ground set.

Definition 38.1.2. Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ be a range space, and let x be a finite (fixed) subset of X. For a range $\mathbf{r} \in \mathcal{R}$, its ***measure*** is the quantity

$$\overline{m}(\mathbf{r}) = \frac{|\mathbf{r} \cap \mathsf{x}|}{|\mathsf{x}|}.$$

While x is finite, it might be very large. As such, we are interested in getting a good estimate to $\overline{m}(\mathbf{r})$ by using a more compact set to represent the range space.

Definition 38.1.3. Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ be a range space. For a subset N (which might be a multi-set) of x, its ***estimate*** of the measure of $\overline{m}(\mathbf{r})$, for $\mathbf{r} \in \mathcal{R}$, is the quantity

$$\overline{s}(\mathbf{r}) = \frac{|\mathbf{r} \cap N|}{|N|}.$$

The main purpose of this chapter is to come up with methods to generate a sample N, such that $\overline{m}(\mathbf{r}) \approx \overline{s}(\mathbf{r})$, for all the ranges $\mathbf{r} \in \mathcal{R}$.

It is easy to see that in the worst case, no sample can capture the measure of all ranges. Indeed, given a sample N, consider the range $\mathsf{x} \setminus N$ that is being completely missed by N. As such, we need to concentrate on range spaces that are "low dimensional", where not all subsets are allowable ranges. The notion of VC dimension (named after Vapnik and Chervonenkis [VC71]) is one way to limit the complexity of a range space.

Definition 38.1.4. Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ be a range space. For $Y \subseteq \mathsf{X}$, let

$$\mathcal{R}_{|Y} = \{\mathbf{r} \cap Y \mid \mathbf{r} \in \mathcal{R}\} \tag{38.1}$$

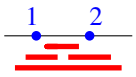
denote the ***projection*** of $\mathcal{R}$ on Y. The range space S projected to Y is $\mathsf{S}_{|Y} = (Y, \mathcal{R}_{|Y})$.

If $\mathcal{R}_{|Y}$ contains all subsets of Y (i.e., if Y is finite, we have $\left|\mathcal{R}_{|Y}\right| = 2^{|Y|}$), then Y is ***shattered*** by $\mathcal{R}$ (or equivalently Y is shattered by S).

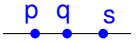
The ***Vapnik-Chervonenkis*** dimension (or VC ***dimension***) of S, denoted by $\dim_{\mathsf{VC}}(\mathsf{S})$, is the maximum cardinality of a shattered subset of X. If there are arbitrarily large shattered subsets, then $\dim_{\mathsf{VC}}(S) = \infty$.

38.1.1. Examples

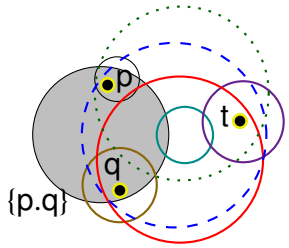
Intervals. Consider the set X to be the real line, and consider $\mathcal{R}$ to be the set of all intervals on the real line. Consider the set $Y = \{1, 2\}$. Clearly, one can find four intervals that contain all possible subsets of Y. Formally, the projection $\mathcal{R}_{|Y} = \{\{\}, \{1\}, \{2\}, \{1, 2\}\}$. The intervals realizing each of these subsets are depicted on the right.



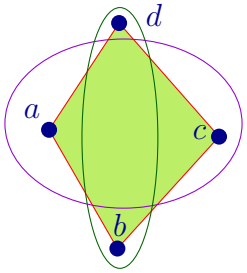
However, this is false for a set of three points $B = \{\mathsf{p}, \mathsf{u}, \mathsf{v}\}$, since there is no interval that can contain the two extreme points p and v without also containing u. Namely, the subset $\{\mathsf{p}, \mathsf{v}\}$ is not realizable for intervals, implying that the largest shattered set by the range space (real line, intervals) is of size two. We conclude that the VC dimension of this space is two.



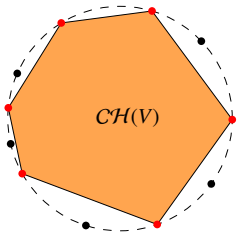
Disks. Let $\mathsf{X} = \mathbb{R}^2$, and let $\mathcal{R}$ be the set of disks in the plane. Clearly, for any three points in the plane (in general position), denoted by p, u, and v, one can find eight disks that realize all possible 2^3 different subsets. See the figure on the right.
But can disks shatter a set with four points? Consider such a set P of four points. If the convex hull of P has only three points on its boundary, then the subset X having only those three vertices (i.e., it does not include the middle point) is impossible, by convexity. Namely, there is no disk that contains only the points of X without the middle point.



Alternatively, if all four points are vertices of the convex hull and they are a, b, c, d along the boundary of the convex hull, either the set $\{a, c\}$ or the set $\{b, d\}$ is not realizable. Indeed, if both options are realizable, then consider the two disks D_1 and D_2 that realize those assignments. Clearly, ∂D_1 and ∂D_2 must intersect in four points, but this is not possible, since two circles have at most two intersection points. See the figure on the left. Hence the VC dimension of this range space is 3.



Convex sets. Consider the range space $\mathsf{S} = (\mathbb{R}^2, \mathcal{R})$, where $\mathcal{R}$ is the set of all (closed) convex sets in the plane. We claim that $\dim_{\mathsf{VC}}(\mathsf{S}) = \infty$. Indeed, consider a set U of n points $p_1, \ldots, p_n$ all lying on the boundary of the unit circle in the plane. Let V be any subset of U, and consider the convex hull $\mathcal{CH}(V)$. Clearly, $\mathcal{CH}(V) \in \mathcal{R}$, and furthermore, $\mathcal{CH}(V) \cap U = V$. Namely, any subset of U is realizable by S. Thus, S can shatter sets of arbitrary size, and its VC dimension is unbounded.



Complement. Consider the range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ with $\delta = \dim_{\mathsf{VC}}(\mathsf{S})$. Next, consider the complement space, $\overline{\mathsf{S}} = (\mathsf{X}, \overline{\mathcal{R}})$, where

$$\overline{\mathcal{R}} = \left\{ \mathsf{X} \setminus \mathbf{r} \;\middle|\; \mathbf{r} \in \mathcal{R} \right\}.$$

Namely, the ranges of $\overline{\mathsf{S}}$ are the complement of the ranges in S. What is the VC dimension of $\overline{\mathsf{S}}$? Well, a set $B \subseteq \mathsf{X}$ is shattered by $\overline{\mathsf{S}}$ if and only if it is shattered by S. Indeed, if S shatters B, then for any $Z \subseteq B$, we have that $(B \setminus Z) \in \mathcal{R}_{|B}$, which implies that $Z = B \setminus (B \setminus Z) \in \overline{\mathcal{R}}_{|B}$. Namely, $\overline{\mathcal{R}}_{|B}$ contains all the subsets of B, and $\overline{\mathsf{S}}$ shatters B. Thus, $\dim_{\mathsf{VC}}(\overline{\mathsf{S}}) = \dim_{\mathsf{VC}}(\mathsf{S})$.

Lemma 38.1.5. *For a range space $\mathsf{S} = (X, \mathcal{R})$ we have that $\dim_{\mathsf{VC}}(\mathsf{S}) = \dim_{\mathsf{VC}}\left(\overline{\mathsf{S}}\right)$, where $\overline{\mathsf{S}}$ is the complement range space.*

38.1.1.1. Halfspaces

Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$, where $\mathsf{X} = \mathbb{R}^d$ and $\mathcal{R}$ is the set of all (closed) halfspaces in $\mathbb{R}^d$. We need the following technical claim.

Claim 38.1.6. *Let $\mathsf{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_{d+2}\}$ be a set of $d+2$ points in $\mathbb{R}^d$. There are real numbers $\beta_1, \ldots, \beta_{d+2}$, not all of them zero, such that $\sum_i \beta_i \mathsf{p}_i = 0$ and $\sum_i \beta_i = 0$.*

Proof: Indeed, set $\mathsf{u}_i = (\mathsf{p}_i, 1)$, for $i = 1, \ldots, d+2$. Now, the points $\mathsf{u}_1, \ldots, \mathsf{u}_{d+2} \in \mathbb{R}^{d+1}$ are linearly dependent, and there are coefficients $\beta_1, \ldots, \beta_{d+2}$, not all of them zero, such that $\sum_{i=1}^{d+2} \beta_i \mathsf{u}_i = 0$. Considering only the first d coordinates of these points implies that $\sum_{i=1}^{d+2} \beta_i \mathsf{p}_i = 0$. Similarly, by considering only the $(d+1)$st coordinate of these points, we have that $\sum_{i=1}^{d+2} \beta_i = 0$. ■

To see what the VC dimension of halfspaces in $\mathbb{R}^d$ is, we need the following result of Radon. (For a reminder of the formal definition of convex hulls, see Definition 38.5.1.)

Theorem 38.1.7 (Radon's theorem). *Let $\mathsf{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_{d+2}\}$ be a set of $d+2$ points in $\mathbb{R}^d$. Then, there exist two disjoint subsets C and D of P, such that $\mathcal{CH}(C) \cap \mathcal{CH}(D) \neq \emptyset$ and $C \cup D = \mathsf{P}$.*

Proof: By Claim 38.1.6 there are real numbers $\beta_1, \ldots, \beta_{d+2}$, not all of them zero, such that $\sum_i \beta_i \mathsf{p}_i = 0$ and $\sum_i \beta_i = 0$.

Assume, for the sake of simplicity of exposition, that $\beta_1, \ldots, \beta_k \geq 0$ and $\beta_{k+1}, \ldots, \beta_{d+2} < 0$. Furthermore, let $\mu = \sum_{i=1}^{k} \beta_i = -\sum_{i=k+1}^{d+2} \beta_i$. We have that

$$\sum_{i=1}^{k} \beta_i \mathsf{p}_i = -\sum_{i=k+1}^{d+2} \beta_i \mathsf{p}_i.$$

In particular, $v = \sum_{i=1}^{k} (\beta_i/\mu) \mathsf{p}_i$ is a point in $\mathcal{CH}(\{\mathsf{p}_1, \ldots, \mathsf{p}_k\})$. Furthermore, for the same point v we have $v = \sum_{i=k+1}^{d+2} -(\beta_i/\mu) \mathsf{p}_i \in \mathcal{CH}(\{\mathsf{p}_{k+1}, \ldots, \mathsf{p}_{d+2}\})$. We conclude that v is in the intersection of the two convex hulls, as required. ■

The following is a trivial observation, and yet we provide a proof to demonstrate it is true.

Lemma 38.1.8. *Let* $\mathsf{P} \subseteq \mathbb{R}^d$ *be a finite set, let* v *be any point in* $\mathcal{CH}(\mathsf{P})$*, and let* h^+ *be a halfspace of* $\mathbb{R}^d$ *containing* v*. Then there exists a point of* P *contained inside* h^+*.*

Proof: The halfspace h^+ can be written as $h^+ = \left\{\mathsf{x} \in \mathbb{R}^d \;\middle|\; \langle \mathsf{x}, v\rangle \le c\right\}$. Now $\mathsf{v} \in \mathcal{CH}(\mathsf{P}) \cap h^+$, and as such there are numbers $\alpha_1, \ldots, \alpha_m \ge 0$ and points $\mathsf{p}_1, \ldots, \mathsf{p}_m \in \mathsf{P}$, such that $\sum_i \alpha_i = 1$ and $\sum_i \alpha_i \mathsf{p}_i = \mathsf{v}$. By the linearity of the dot product, we have that

$$\mathsf{v} \in h^+ \implies \langle \mathsf{v}, v\rangle \le c \implies \left\langle \sum_{i=1}^{m} \alpha_i \mathsf{p}_i, v \right\rangle \le c \implies \beta = \sum_{i=1}^{m} \alpha_i \langle \mathsf{p}_i, v\rangle \le c.$$

Setting $\beta_i = \langle \mathsf{p}_i, v\rangle$, for $i = 1, \ldots, m$, the above implies that β is a weighted average of $\beta_1, \ldots, \beta_m$. In particular, there must be a β_i that is no larger than the average. That is $\beta_i \le c$. This implies that $\langle \mathsf{p}_i, v\rangle \le c$. Namely, $\mathsf{p}_i \in h^+$ as claimed. ■

Let S be the range space having $\mathbb{R}^d$ as the ground set and all the close halfspaces as ranges. Radon's theorem implies that if a set Q of $d+2$ points is being shattered by S, then we can partition this set Q into two disjoint sets Y and Z such that $\mathcal{CH}(Y) \cap \mathcal{CH}(Z) \neq \emptyset$. In particular, let v be a point in $\mathcal{CH}(Y) \cap \mathcal{CH}(Z)$. If a halfspace h^+ contains all the points of Y, then $\mathcal{CH}(Y) \subseteq h^+$, since a halfspace is a convex set. Thus, any halfspace h^+ containing all the points of Y will contain the point $\mathsf{v} \in \mathcal{CH}(Y)$. But $\mathsf{v} \in \mathcal{CH}(Z) \cap h^+$, and this implies that a point of Z must lie in h^+, by Lemma 38.1.8. Namely, the subset $Y \subseteq \mathsf{Q}$ cannot be realized by a halfspace, which implies that Q cannot be shattered. Thus $\dim_{\mathsf{VC}}(S) < d+2$. It is also easy to verify that the regular simplex with $d+1$ vertices is shattered by S. Thus, $\dim_{\mathsf{VC}}(S) = d+1$.

38.2. Shattering dimension and the dual shattering dimension

The main property of a range space with bounded VC dimension is that the number of ranges for a set of n elements grows polynomially in n (with the power being the dimension) instead of exponentially. Formally, let the ***growth function*** be

$$\mathcal{G}_\delta(n) = \sum_{i=0}^{\delta} \binom{n}{i} \le \sum_{i=0}^{\delta} \frac{n^i}{i!} \le n^\delta, \tag{38.2}$$

for $\delta > 1$ (the cases where $\delta = 0$ or $\delta = 1$ are not interesting and we will just ignore them). Note that for all $n, \delta \ge 1$, we have $\mathcal{G}_\delta(n) = \mathcal{G}_\delta(n-1) + \mathcal{G}_{\delta-1}(n-1)$[①].

Lemma 38.2.1 (Sauer's lemma). *If* $(X, \mathcal{R})$ *is a range space of* VC *dimension* δ *with* $|X| = n$*, then* $|\mathcal{R}| \le \mathcal{G}_\delta(n)$*.*

Proof: The claim trivially holds for $\delta = 0$ or $n = 0$.

Let x be any element of X, and consider the sets

$$\mathcal{R}_x = \left\{\mathbf{r} \setminus \{x\} \;\middle|\; \mathbf{r} \cup \{x\} \in \mathcal{R} \text{ and } \mathbf{r} \setminus \{x\} \in \mathcal{R}\right\} \quad \text{and} \quad \mathcal{R} \setminus x = \left\{\mathbf{r} \setminus \{x\} \;\middle|\; \mathbf{r} \in \mathcal{R}\right\}.$$

Observe that $|\mathcal{R}| = |\mathcal{R}_x| + |\mathcal{R} \setminus x|$. Indeed, we charge the elements of $\mathcal{R}$ to their corresponding element in $\mathcal{R} \setminus x$. The only bad case is when there is a range $\mathbf{r}$ such that both $\mathbf{r} \cup \{x\} \in \mathcal{R}$ and $\mathbf{r} \setminus \{x\} \in \mathcal{R}$, because then these two

[①] Here is a cute (and standard) counting argument: $\mathcal{G}_\delta(n)$ is just the number of different subsets of size at most δ out of n elements. Now, we either decide to not include the first element in these subsets (i.e., $\mathcal{G}_\delta(n-1)$) or, alternatively, we include the first element in these subsets, but then there are only $\delta - 1$ elements left to pick (i.e., $\mathcal{G}_{\delta-1}(n-1)$).

distinct ranges get mapped to the same range in $\mathcal{R} \setminus x$. But such ranges contribute exactly one element to $\mathcal{R}_x$. Similarly, every element of $\mathcal{R}_x$ corresponds to two such "twin" ranges in $\mathcal{R}$.

Observe that $(X \setminus \{x\}, \mathcal{R}_x)$ has VC dimension $\delta - 1$, as the largest set that can be shattered is of size $\delta - 1$. Indeed, any set $B \subset \mathsf{X} \setminus \{x\}$ shattered by $\mathcal{R}_x$ implies that $B \cup \{x\}$ is shattered in $\mathcal{R}$.

Thus, we have

$$|\mathcal{R}| = |\mathcal{R}_x| + |\mathcal{R} \setminus x| \leq \mathcal{G}_{\delta-1}(n-1) + \mathcal{G}_\delta(n-1) = \mathcal{G}_\delta(n),$$

by induction. ■

Interestingly, Lemma 38.2.1 is tight.

Next, we show pretty tight bounds on $\mathcal{G}_\delta(n)$. The proof is technical and not very interesting, and it is delegated to Section 38.4.

Lemma 38.2.2. *For $n \geq 2\delta$ and $\delta \geq 1$, we have $\left(\frac{n}{\delta}\right)^\delta \leq \mathcal{G}_\delta(n) \leq 2\left(\frac{ne}{\delta}\right)^\delta$, where $\mathcal{G}_\delta(n) = \sum_{i=0}^{\delta} \binom{n}{i}$.*

Definition 38.2.3 (Shatter function). Given a range space $\mathsf{S} = (\mathsf{X}, \mathcal{R})$, its ***shatter function*** $\pi_{\mathsf{S}}(m)$ is the maximum number of sets that might be created by S when restricted to subsets of size m. Formally,

$$\pi_{\mathsf{S}}(m) = \max_{\substack{B \subset \mathsf{X} \\ |B| = m}} \left|\mathcal{R}_{|B}\right|;$$

see Eq. (38.1).

Our arch-nemesis in the following is the function $x/\ln x$. The following lemma states some properties of this function, and its proof is left as an exercise.

Lemma 38.2.4. *For the function $f(x) = x/\ln x$ the following hold.*

(A) $f(x)$ is monotonically increasing for $x \geq e$.
(B) $f(x) \geq e$, for $x > 1$.
(C) For $u \geq \sqrt{e}$, if $f(x) \leq u$, then $x \leq 2u \ln u$.
(D) For $u \geq \sqrt{e}$, if $x > 2u \ln u$, then $f(x) > u$.
(E) For $u \geq e$, if $f(x) \geq u$, then $x \geq u \ln u$.

38.2.1. Mixing range spaces

Lemma 38.2.5. *Let $\mathsf{S} = (X, \mathcal{R})$ and $\mathsf{T} = (X, \mathcal{R}')$ be two range spaces of VC dimension δ and δ', respectively, where $\delta, \delta' > 1$. Let $\widehat{\mathcal{R}} = \{\mathbf{r} \cup \mathbf{r}' \mid \mathbf{r} \in \mathcal{R}, \mathbf{r}' \in \mathcal{R}'\}$. Then, for the range space $\widehat{\mathsf{S}} = (X, \widehat{\mathcal{R}})$, we have that $\dim_{\mathsf{VC}}(\widehat{\mathsf{S}}) = O(\delta + \delta')$.*

Proof: As a warm-up exercise, we prove a somewhat weaker bound here of $O((\delta+\delta')\log(\delta+\delta'))$. The stronger bound follows from Theorem 38.2.6 below. Let B be a set of n points in X that are shattered by $\widehat{\mathsf{S}}$. There are at most $\mathcal{G}_\delta(n)$ and $\mathcal{G}_{\delta'}(n)$ different ranges of B in the range sets $\mathcal{R}_{|B}$ and $\mathcal{R}'_{|B}$, respectively, by Lemma 38.2.1. Every subset C of B realized by $\widehat{r} \in \widehat{\mathcal{R}}$ is a union of two subsets $B \cap \mathbf{r}$ and $B \cap \mathbf{r}'$, where $\mathbf{r} \in R$ and $\mathbf{r}' \in \mathcal{R}'$, respectively. Thus, the number of different subsets of B realized by $\widehat{\mathsf{S}}$ is bounded by $\mathcal{G}_\delta(n)\mathcal{G}_{\delta'}(n)$. Thus, $2^n \leq n^\delta n^{\delta'}$, for $\delta, \delta' > 1$. We conclude that $n \leq (\delta + \delta')\lg n$, which implies that $n = O((\delta + \delta')\log(\delta + \delta'))$, by Lemma 38.2.4(C). ■

Interestingly, one can prove a considerably more general result with tighter bounds. The required computations are somewhat more painful.

Theorem 38.2.6. *Let* $\mathsf{S}_1 = \left(X, \mathcal{R}^1\right), \ldots, \mathsf{S}_k = \left(X, \mathcal{R}^k\right)$ *be range spaces with* VC *dimension* $\delta_1, \ldots, \delta_k$*, respectively. Next, let* $f(\mathbf{r}_1, \ldots, \mathbf{r}_k)$ *be a function that maps any* k*-tuple of sets* $\mathbf{r}_1 \in \mathcal{R}^1, \ldots, \mathbf{r}_k \in \mathcal{R}^k$ *into a subset of* X*. Here, the function* f *is restricted to be defined by a sequence of set operations like complement, intersection and union. Consider the range set*

$$\mathcal{R}' = \{ f(\mathbf{r}_1, \ldots, \mathbf{r}_k) \;\big|\; \mathbf{r}_1 \in \mathcal{R}_1, \ldots, \mathbf{r}_k \in \mathcal{R}_k \}$$

and the associated range space $\mathsf{T} = (X, \mathcal{R}')$*. Then, the* VC *dimension of* T *is bounded by* $O(k\delta \lg k)$*, where* $\delta = \max_i \delta_i$*.*

Proof: Assume a set $Y \subseteq \mathsf{X}$ of size t is being shattered by $\mathcal{R}'$, and observe that

$$\begin{aligned}\left|\mathcal{R}'_{|Y}\right| &\leq \left| \{ (\mathbf{r}_1, \ldots, \mathbf{r}_k) \;\big|\; \mathbf{r}_1 \in \mathcal{R}^1_{|Y}, \ldots, \mathbf{r}_k \in \mathcal{R}^k_{|Y} \} \right| \leq \left|\mathcal{R}^1_{|Y}\right| \cdots \left|\mathcal{R}^k_{|Y}\right| \leq \mathcal{G}_{\delta_1}(t) \cdot \mathcal{G}_{\delta_2}(t) \cdots \mathcal{G}_{\delta_k}(t) \\ &\leq \left(\mathcal{G}_\delta(t)\right)^k \leq \left(2\left(te/\delta\right)^\delta\right)^k,\end{aligned}$$

by Lemma 38.2.1 and Lemma 38.2.2. On the other hand, since Y is being shattered by $\mathcal{R}'$, this implies that $\left|\mathcal{R}'_{|Y}\right| = 2^t$. Thus, we have the inequality $2^t \leq (2(te/\delta)^\delta)^k$, which implies $t \leq k(1 + \delta \lg(te/\delta))$. Assume that $t \geq e$ and $\delta \lg(te/\delta) \geq 1$ since otherwise the claim is trivial, and observe that $t \leq k(1 + \delta \lg(te/\delta)) \leq 3k\delta \lg(t/\delta)$. Setting $x = t/\delta$, we have

$$\frac{t}{\delta} \leq 3k \frac{\ln(t/\delta)}{\ln 2} \leq 6k \ln \frac{t}{\delta} \implies \frac{x}{\ln x} \leq 6k \implies x \leq 2 \cdot 6k \ln(6k) \implies x \leq 12k \ln(6k),$$

by Lemma 38.2.4(C). We conclude that $t \leq 12\delta k \ln(6k)$, as claimed. ■

Corollary 38.2.7. *Let* $\mathsf{S} = (X, \mathcal{R})$ *and* $\mathsf{T} = (X, \mathcal{R}')$ *be two range spaces of* VC *dimension* δ *and* δ'*, respectively, where* $\delta, \delta' > 1$*. Let* $\widehat{\mathcal{R}} = \{ \mathbf{r} \cap \mathbf{r}' \;\big|\; \mathbf{r} \in \mathcal{R}, \mathbf{r}' \in \mathcal{R}' \}$*. Then, for the range space* $\widehat{\mathsf{S}} = (X, \widehat{\mathcal{R}})$*, we have that* $\dim_{\mathsf{VC}}(\widehat{\mathsf{S}}) = O(\delta + \delta')$*.*

Corollary 38.2.8. *Any finite sequence of combining range spaces with finite* VC *dimension (by intersecting, complementing, or taking their union) results in a range space with a finite* VC *dimension.*

38.3. On ε-nets and ε-sampling

38.3.1. ε-nets and ε-samples

Definition 38.3.1 (ε-sample). Let $\mathsf{S} = (\mathsf{X}, \mathcal{R})$ be a range space, and let x be a finite subset of X. For $0 \leq \varepsilon \leq 1$, a subset $C \subseteq \mathsf{x}$ is an ***ε-sample*** for x if for any range $\mathbf{r} \in \mathcal{R}$, we have

$$\left| \overline{m}(\mathbf{r}) - \overline{s}(\mathbf{r}) \right| \leq \varepsilon,$$

where $\overline{m}(\mathbf{r}) = |\mathsf{x} \cap \mathbf{r}| \,/\, |\mathsf{x}|$ is the measure of $\mathbf{r}$ (see Definition 38.1.2) and $\overline{s}(\mathbf{r}) = |C \cap \mathbf{r}| \,/\, |C|$ is the estimate of $\mathbf{r}$ (see Definition 38.1.3). (Here C might be a multi-set, and as such $|C \cap \mathbf{r}|$ is counted with multiplicity.)

As such, an ε-sample is a subset of the ground set x that "captures" the range space up to an error of ε. Specifically, to estimate the fraction of the ground set covered by a range $\mathbf{r}$, it is sufficient to count the points of C that fall inside $\mathbf{r}$.

If X is a finite set, we will abuse notation slightly and refer to C as an ***ε-sample*** for S.

To see the usage of such a sample, consider $\mathsf{x} = \mathsf{X}$ to be, say, the population of a country (i.e., an element of X is a citizen). A range in $\mathcal{R}$ is the set of all people in the country that answer yes to a question (i.e., would you vote for party Y?, would you buy a bridge from me?, questions like that). An ε-sample of this range space enables us to estimate reliably (up to an error of ε) the answers for all these questions, by just asking the people in the sample.

The natural question of course is how to find such a subset of small (or minimal) size.

Theorem 38.3.2 (ε-sample theorem, [VC71]). *There is a positive constant c such that if $(X, \mathcal{R})$ is any range space with VC dimension at most δ, $\mathsf{x} \subseteq X$ is a finite subset and $\varepsilon, \varphi > 0$, then a random subset $C \subseteq \mathsf{x}$ of cardinality*

$$s = \frac{c}{\varepsilon^2}\left(\delta \log \frac{\delta}{\varepsilon} + \log \frac{1}{\varphi}\right)$$

is an ε-sample for x with probability at least $1 - \varphi$.

(In the above theorem, if $s > |\mathsf{x}|$, then we can just take all of x to be the ε-sample.)

Sometimes it is sufficient to have (hopefully smaller) samples with a weaker property – if a range is "heavy", then there is an element in our sample that is in this range.

Definition 38.3.3 (ε-net). A set $N \subseteq \mathsf{x}$ is an ***ε-net*** for x if for any range $\mathbf{r} \in \mathcal{R}$, if $\overline{m}(\mathbf{r}) \geq \varepsilon$ (i.e., $|\mathbf{r} \cap \mathsf{x}| \geq \varepsilon |\mathsf{x}|$), then $\mathbf{r}$ contains at least one point of N (i.e., $\mathbf{r} \cap N \neq \emptyset$).

Theorem 38.3.4 (ε-net theorem, [HW87]). *Let $(X, \mathcal{R})$ be a range space of VC dimension δ, let x be a finite subset of X, and suppose that $0 < \varepsilon \leq 1$ and $\varphi < 1$. Let N be a set obtained by m random independent draws from x, where*

$$m \geq \max\left(\frac{4}{\varepsilon} \lg \frac{4}{\varphi}, \frac{8\delta}{\varepsilon} \lg \frac{16}{\varepsilon}\right). \tag{38.3}$$

Then N is an ε-net for x with probability at least $1 - \varphi$.

(We remind the reader that $\lg = \log_2$.)

The proofs of the above theorems are somewhat involved and we first turn our attention to some applications before presenting the proofs.

Remark 38.3.5. The above two theorems also hold for spaces with shattering dimension at most δ, in which case the sample size is slightly larger. Specifically, for Theorem 38.3.4, the sample size needed is $O\left(\frac{1}{\varepsilon} \lg \frac{1}{\varphi} + \frac{\delta}{\varepsilon} \lg \frac{\delta}{\varepsilon}\right)$.

Remark 38.3.6. The ε-net theorem is a relatively easy consequence (up to constants) of the ε-sample theorem – see bibliographical notes for details.

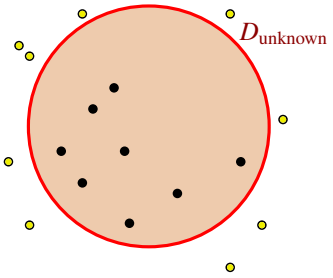
38.3.2. Some applications

We mention two (easy) applications of these theorems, which (hopefully) demonstrate their power.

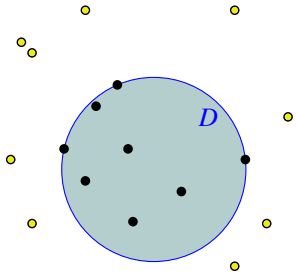
38.3.2.1. Range searching

So, consider a (very large) set of points P in the plane. We would like to be able to quickly decide how many points are included inside a query rectangle. Let us assume that we allow ourselves 1% error. What Theorem 38.3.2 tells us is that there is a subset of *constant size* (that depends only on ε) that can be used to perform this estimation, and it works for *all* query rectangles (we used here the fact that rectangles in the plane have finite VC dimension). In fact, a random sample of this size works with constant probability.

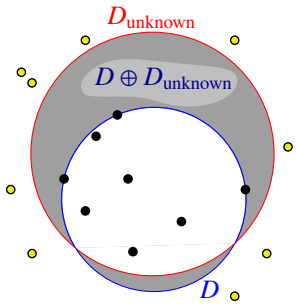
38.3.2.2. Learning a concept



Assume that we have a function f defined in the plane that returns '1' inside an (unknown) disk D_{unknown} and '0' outside it. There is some distribution $\mathcal{D}$ defined over the plane, and we pick points from this distribution. Furthermore, we can compute the function for these labels (i.e., we can compute f for certain values, but it is expensive). For a mystery value $\varepsilon > 0$, to be explained shortly, Theorem 38.3.4 tells us to pick (roughly) $O((1/\varepsilon)\log(1/\varepsilon))$ random points in a sample R from this distribution and to compute the labels for the samples. This is demonstrated in the figure on the right, where black dots are the sample points for which $f(\cdot)$ returned 1.



So, now we have positive examples and negative examples. We would like to find a hypothesis that agrees with all the samples we have and that hopefully is close to the true unknown disk underlying the function f. To this end, compute the smallest disk D that contains the sample labeled by '1' and does not contain any of the '0' points, and let $g : \mathbb{R}^2 \to \{0, 1\}$ be the function g that returns '1' inside the disk and '0' otherwise. We claim that g classifies correctly all but an ε-fraction of the points (i.e., the probability of misclassifying a point picked according to the given distribution is smaller than ε); that is, $\mathbf{Pr}_{\mathsf{p}\in\mathcal{D}}\left[f(\mathsf{p}) \neq g(\mathsf{p})\right] \leq \varepsilon$.



Geometrically, the region where g and f disagree is all the points in the symmetric difference between the two disks. That is, $\mathcal{E} = D \oplus D_{\text{unknown}}$; see the figure on the right.

Thus, consider the range space S having the plane as the ground set and the symmetric difference between any two disks as its ranges. By Corollary 38.2.8, this range space has finite VC dimension. Now, consider the (unknown) disk D' that induces f and the region $\mathbf{r} = D_{\text{unknown}} \oplus D$. Clearly, the learned classifier g returns incorrect answers only for points picked inside $\mathbf{r}$.

Thus, the probability of a mistake in the classification is the measure of $\mathbf{r}$ under the distribution $\mathcal{D}$. So, if $\mathbb{P}_{\mathcal{D}}[\mathbf{r}] > \varepsilon$ (i.e., the probability that a sample point falls inside $\mathbf{r}$), then by the ε-net theorem (i.e., Theorem 38.3.4) the set R is an ε-net for S (ignore for the time being the possibility that the random sample fails to be an ε-net) and as such, R contains a point u inside $\mathbf{r}$. But, it is not possible for g (which classifies correctly all the sampled points of R) to make a mistake on u, a contradiction, because by construction, the range $\mathbf{r}$ is where g misclassifies points. We conclude that $\mathbb{P}_{\mathcal{D}}[\mathbf{r}] \leq \varepsilon$, as desired.

Little lies. The careful reader might be tearing his or her hair out because of the above description. First, Theorem 38.3.4 might fail, and the above conclusion might not hold. This is of course true, and in real applications one might use a much larger sample to guarantee that the probability of failure is so small that it can be practically ignored. A more serious issue is that Theorem 38.3.4 is defined only for finite sets. Nowhere does it speak about a continuous distribution. Intuitively, one can approximate a continuous distribution to an arbitrary precision using a huge sample and apply the theorem to this sample as our ground set. A formal proof is more tedious and requires extending the proof of Theorem 38.3.4 to continuous distributions. This is straightforward and we will ignore this topic altogether.

38.4. A better bound on the growth function

In this section, we prove Lemma 38.2.2. Since the proof is straightforward but tedious, the reader can safely skip reading this section.

Lemma 38.4.1. *For any positive integer n, the following hold.*

(i) $(1+1/n)^n \le e$. *(ii)* $(1-1/n)^{n-1} \ge e^{-1}$.

(iii) $n! \ge (n/e)^n$. *(iv)* *For any $k \le n$, we have* $\left(\frac{n}{k}\right)^k \le \binom{n}{k} \le \left(\frac{ne}{k}\right)^k$.

Proof: (i) Indeed, $1 + 1/n \le \exp(1/n)$, since $1 + x \le e^x$, for $x \ge 0$. As such $(1+1/n)^n \le \exp(n(1/n)) = e$.

(ii) Rewriting the inequality, we have that we need to prove $\left(\frac{n-1}{n}\right)^{n-1} \ge \frac{1}{e}$. This is equivalent to proving $e \ge \left(\frac{n}{n-1}\right)^{n-1} = \left(1 + \frac{1}{n-1}\right)^{n-1}$, which is our friend from (i).

(iii) Indeed,

$$\frac{n^n}{n!} \le \sum_{i=0}^{\infty} \frac{n^i}{i!} = e^n,$$

by the Taylor expansion of $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$. This implies that $(n/e)^n \le n!$, as required.

(iv) Indeed, for any $k \le n$, we have $\frac{n}{k} \le \frac{n-1}{k-1}$, as can be easily verified. As such, $\frac{n}{k} \le \frac{n-i}{k-i}$, for $1 \le i \le k-1$. As such,

$$\left(\frac{n}{k}\right)^k \le \frac{n}{k} \cdot \frac{n-1}{k-1} \cdots \frac{n-k+1}{1} = \binom{n}{k}.$$

As for the other direction, by (iii), we have $\binom{n}{k} \le \frac{n^k}{k!} \le \frac{n^k}{\left(\frac{k}{e}\right)^k} = \left(\frac{ne}{k}\right)^k$. ■

Lemma 38.2.2 restated. *For $n \ge 2\delta$ and $\delta \ge 1$, we have* $\left(\frac{n}{\delta}\right)^{\delta} \le \mathcal{G}_\delta(n) \le 2\left(\frac{ne}{\delta}\right)^{\delta}$, *where* $\mathcal{G}_\delta(n) = \sum_{i=0}^{\delta} \binom{n}{i}$.

Proof: Note that by Lemma 38.4.1 (iv), we have $\mathcal{G}_\delta(n) = \sum_{i=0}^{\delta} \binom{n}{i} \le 1 + \sum_{i=1}^{\delta} \left(\frac{ne}{i}\right)^i$. This series behaves like a geometric series with constant larger than 2, since

$$\left(\frac{ne}{i}\right)^i \Big/ \left(\frac{ne}{i-1}\right)^{i-1} = \frac{ne}{i}\left(\frac{i-1}{i}\right)^{i-1} = \frac{ne}{i}\left(1 - \frac{1}{i}\right)^{i-1} \ge \frac{ne}{i}\frac{1}{e} = \frac{n}{i} \ge \frac{n}{\delta} \ge 2,$$

by Lemma 38.4.1. As such, this series is bounded by twice the largest element in the series, implying the claim. ■

38.5. Some required definitions

Definition 38.5.1 (Convex hull). The ***convex hull*** of a set $\mathsf{R} \subseteq \mathbb{R}^d$ is the set of all convex combinations of points of R; that is,

$$\mathcal{CH}(\mathsf{R}) = \left\{ \sum_{i=0}^{m} \alpha_i \mathsf{v}_i \;\middle|\; \forall i \;\; \mathsf{v}_i \in \mathsf{R}, \alpha_i \ge 0, \text{ and } \sum_{j=1}^{m} \alpha_i = 1 \right\}.$$

References

[HW87] D. Haussler and E. Welzl. ε-nets and simplex range queries. *Discrete Comput. Geom.*, 2: 127–151, 1987.

[VC71] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16: 264–280, 1971.

Chapter 39

Double sampling

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "What does not work when you apply force, would work when you apply even more force."
>
> , Anonymous

39.1. Double sampling

Double sampling is the idea that two random independent samples should look similar, and should *not* be completely different in the way they intersect a certain set. We use the following sampling model, which makes the computations somewhat easier.

Definition 39.1.1. Let $\mathsf{S} = \{f_1, \ldots, f_n\}$ be a set of objects, where the ith object has weight $\omega_i > 0$, for all i. Let $W = \sum_i \omega_i$. For a target size ρ, a ***ρ-sample*** is a random sample $\mathsf{R} \subseteq \mathsf{S}$, where object f_i is picked independently with probability $\rho\omega_i/W$. To simplify the discussion, we assume that $\rho\omega_i/W < 1$. Handling the more general case is easy if somewhat tedious.

Lemma 39.1.2. *Let R_1 and R_2 be two ρ-samples, and consider the merged sample $\mathsf{R} = \mathsf{R}_1 \cup \mathsf{R}_2$. Let $T \subseteq \mathsf{S}$ be a set of m elements. Then, we have that*

$$\mathbb{P}\Big[T \subseteq \mathsf{R}_1 \;\Big|\; T \subseteq \mathsf{R}\Big] \geq \frac{1}{2^m} \qquad \text{and} \qquad \mathbb{P}\Big[T \subseteq \mathsf{R}_1 \text{ and } T \cap \mathsf{R}_2 = \emptyset \;\Big|\; T \subseteq \mathsf{R}\Big] \leq \frac{1}{2^m}.$$

Proof: Consider an object $f \in T$, and observe that $\mathbb{P}[f \in \mathsf{R}_1 \text{ or } f \in \mathsf{R}_2 \mid f \in \mathsf{R}] = 1$. As such, by symmetry

$$\mathbb{P}[f \in \mathsf{R}_1 \mid f \in \mathsf{R}] = \mathbb{P}[f \in \mathsf{R}_2 \mid f \in \mathsf{R}] \geq 1/2,$$

Now, let $T = \{f_1, \ldots, f_m\}$. Since R_1 and R_2 are independent, and each element is being picked independently, we have that

$$\mathbb{P}[T \subseteq \mathsf{R}_1 \mid T \subseteq \mathsf{R}] = \mathbb{P}[f_1, \ldots, f_m \in \mathsf{R}_1 \mid f_1, \ldots, f_m \in \mathsf{R}] = \prod_{i=1}^{b} \mathbb{P}[f_i \in \mathsf{R}_1 \mid f_1, \ldots, f_m \in \mathsf{R}]$$

$$= \prod_{i=1}^{b} \mathbb{P}[f_i \in \mathsf{R}_1 \mid f_i \in \mathsf{R}] \geq \frac{1}{2^m}.$$

For the second claim, observe that again, by symmetry, we have that

$$\mathbb{P}[f \in \mathsf{R}_1 \text{ and } f \notin \mathsf{R}_2 \;\big|\; f \in \mathsf{R}] = \mathbb{P}[f \notin \mathsf{R}_1 \text{ and } f \in \mathsf{R}_2 \mid f \in \mathsf{R}] \leq 1/2,$$

as the two events are disjoint. Now, the claim follows by arguing as above. ■

39.1.1. Disagreement between samples on a specific set

We provide three proofs of the following lemma – the constants are somewhat different for each version.

Lemma 39.1.3. *Let* R_1 *and* R_2 *be two* ρ*-samples from a ground set* S*, and consider a fixed set* $T \subseteq \mathsf{S}$*. We have that*

$$\mathbb{P}\Big[\big||\mathsf{R}_1 \cap T| - |\mathsf{R}_2 \cap T|\big| > \varepsilon\rho\Big] \leq 3\exp\Big(-\varepsilon^2\rho/2\Big).$$

Proof: (Simplest proof.) By Chernoff's inequality, for $\delta \in (0,1)$, we have

$$\mathbb{P}\Big[\big||\mathsf{R}_1| - \rho\big| \geq (\varepsilon/2)\rho\Big] \leq 2\exp\Big(-(\varepsilon/2)^2\rho/4\Big) = 2\exp\Big(-\varepsilon^2\rho/16\Big).$$

The same holds for R_2, and as such we have

$$\begin{aligned}\mathbb{P}\Big[\big||\mathsf{R}_1| - |\mathsf{R}_2|\big| \geq \varepsilon\rho\Big] &\leq \mathbb{P}\Big[\big||\mathsf{R}_1| - \rho\big| + \big|\rho - |\mathsf{R}_2|\big| \geq \varepsilon\rho\Big] \\ &\leq \mathbb{P}\Big[\big||\mathsf{R}_1| - \rho\big| \geq (\varepsilon/2)\rho\Big] + \mathbb{P}\Big[\big|\rho - |\mathsf{R}_2|\big| \geq (\varepsilon/2)\rho\Big] \leq 4\exp\Big(-\varepsilon^2\rho/16\Big)\end{aligned}$$ ■

Proof: For an object $f_i \in \mathsf{S}$, let X_i be a random variable, where

$$X_i = \begin{cases} 1 & f_i \in \mathsf{R}_1 \text{ and } f_i \notin \mathsf{R}_2 \\ -1 & f_i \notin \mathsf{R}_1 \text{ and } f_i \in \mathsf{R}_2 \\ 0 & \text{otherwise.} \end{cases}$$

We have that $p_i = \mathbb{P}[X_i = 1] = \mathbb{P}[X_i = -1] = (\rho\omega_i/W)(1 - \rho\omega_i/W)$ and $\mathbb{E}[X_i] = 0$. Applying the regular concentration inequalities in this case is not immediate, since there are many X_is that are zero. To overcome this, let T be a random variable that is the number of variables in $X_1, \ldots, X_n$ that are non-zero. We have that T is a sum of n independent 0/1 random variables, where $\mathbb{E}[T] = \sum_i 2p_i = 2\rho$. In particular, by Chernoff's inequality, we have that

$$q_1 = \mathbb{P}[T > (1+\varepsilon)2\rho] \leq \exp\Big(-2\rho\varepsilon^2/4\Big) = \exp\Big(-\rho\varepsilon^2/2\Big).$$

and assume this happens. In particular, let $Z_1, \ldots, Z_T$ be the non-zero variables in $X_1, \ldots, X_n$, and observe that $\mathbb{P}[Z_i = 1] = \mathbb{P}[Z_i = -1] = 1/2$. Let $Y = \sum_i X_i = \sum_i Z_i$. Observe that $\mathbb{E}[Y] = 0$, and by Chernoff inequality, we have that

$$\begin{aligned}q_2 &= \mathbb{P}\Big[\big||\mathsf{R}_1 \cap \mathsf{S}| - |\mathsf{R}_1 \cap \mathsf{S}|\big| > \varepsilon\rho\Big] = \mathbb{P}\Big[|Y - \mathbb{E}[Y]| \geq \varepsilon\rho\Big] \leq \mathbb{P}\Big[\big|\textstyle\sum_i Z_i - 0\big| \geq \varepsilon\rho\Big] \\ &\leq 2\exp\left(-2\frac{(\varepsilon\rho)^2}{2T}\right) \leq 2\exp\left(-2\frac{(\varepsilon\rho)^2}{2(1+\varepsilon)\rho}\right) + q_1 = 2\exp\left(-\frac{\varepsilon^2\rho}{1+\varepsilon}\right) + q_1 \leq 3\exp\Big(-\varepsilon^2\rho/2\Big),\end{aligned}$$

using $T \leq (1+\xi)2\rho$. ■

39.1.2. Exponential decay for a single set

Lemma 39.1.4. *Consider a set* S *of* m *objects, where every object* $f_i \in \mathsf{S}$ *has weight* $\omega_i > 0$*, and* $W = \sum_{i=1}^m \omega_i$*. Next, consider a set* $\mathbf{r} \subseteq \mathsf{S}$ *such that* $\omega(\mathbf{r}) \geq tW/\rho$ *(such a set is* t**-heavy**$)$*. Let* R *be a* ρ*-sample from* S*. Then, the probability that* R *misses* $\mathbf{r}$ *is at most* e^{-t}*. Formally, we have* $\mathbb{P}[\mathbf{r} \cap \mathsf{R} = \emptyset] \leq \exp(-t)$*.*

Proof: Let $\mathbf{r} = \{f_1, \ldots, f_k\}$. Clearly, the probability that R fails to pick one of these conflicting objects, is bounded by $\mathbb{P}[\mathbf{r} \cap \mathsf{R} = \emptyset] = \mathbb{P}[\forall i \in \{1, \ldots, k\} \quad f_i \notin \mathsf{R}_2] = \prod_{i=1}^k \Big(1 - \rho\frac{\omega_i}{W}\Big) \leq \prod_{i=1}^k \exp\Big(-\rho\frac{\omega_i}{W}\Big) = \exp\Big(-\frac{\rho}{W}\sum_i \omega_i\Big) \leq \exp\Big(-\frac{\rho}{W} \cdot t\frac{W}{\rho}\Big) = \exp(-t)$. ■

39.1.3. Moments of the sample size

Lemma 39.1.5. *Let* R *an* m*-sample. And let* $f(t) \leq \alpha t^{\beta}$*, where* $\alpha \geq 1$ *and* $\beta \geq 1$ *are constants, such that* $m \geq 16\beta$*. Then* $U(m) = \mathbb{E}[f(|\mathsf{R}|)] \leq 2\alpha(2m)^{\beta}$.

Proof: The proof follows from Chernoff's inequality and some tedious but straightforward calculations. The reader is as such encouraged to skip reading it.

Let $X = |\mathsf{R}|$. This is a sum of $0/1$ random variables with expectation m. As such, we have

$$\nu = \mathbb{E}[f(|\mathsf{R}|)] \leq \sum_{i=0}^{\infty} \mathbb{P}[X = i] f(i) \leq \alpha \sum_{i=0}^{\infty} \mathbb{P}[X = i] i^{\beta}.$$

Considering the last sum, we have

$$\sum_{i=0}^{\infty} \mathbb{P}[X = i] i^{\beta} \leq \sum_{j=0}^{\infty} \mathbb{P}[X \geq jm]((j+1)m)^{\beta} \leq (2m)^{\beta} + m^{\beta} \sum_{j=2}^{\infty} \mathbb{P}[X \geq jm](j+1)^{\beta}.$$

We bound the last summation using Chernoff's inequality (see Theorem 13.2.1), we have

$$\begin{aligned}
\tau &= \sum_{j=2}^{5} \mathbb{P}[X \geq jm](j+1)^{\beta} + \sum_{j=6}^{\infty} \mathbb{P}[X \geq jm](j+1)^{\beta} \\
&\leq \sum_{j=2}^{5} \exp\left(-\frac{m(j-1)^2}{4}\right)(j+1)^{\beta} + \sum_{j=6}^{\infty} 2^{-jm}(j+1)^{\beta} \\
&\leq \exp\left(-\frac{m}{4}\right)3^{\beta} + \exp(-m)4^{\beta} + \exp(-2m)5^{\beta} + \exp(-4m)6^{\beta} + \sum_{j=6}^{\infty} 2^{-jm}(j+1)^{\beta} < 1,
\end{aligned}$$

since $m \geq 16\beta$. We conclude that $\nu \leq \alpha(2m)^{\beta} + \alpha m^{\beta}\tau \leq 2\alpha(2m)^{\beta}$. ■

Remark 39.1.6. The constant 16 in the above lemma is somewhat strange. A better constant can be derived by breaking the range of sizes into smaller intervals and using the right Chernoff inequality. Since this is somewhat tangential to the point of this write-up, we leave it as is (i.e., this constant is not critical to our discussion).

39.1.4. Growth function

The ***growth function*** $\mathcal{G}_{\delta}(n)$ is the maximum number of ranges in a range space with VC dimension δ, and with n elements. By Sauer's lemma, it is known that

$$\mathcal{G}_{\delta}(n) = \sum_{i=0}^{\delta} \binom{n}{i} \leq \sum_{i=0}^{\delta} \frac{n^i}{i!} \leq n^{\delta}, \tag{39.1}$$

The following is well known (the estimates are somewhat tedious to prove):

Lemma 39.1.7 ([Har11]). *For* $n \geq 2\delta$ *and* $\delta \geq 1$*, we have* $\left(\frac{n}{\delta}\right)^{\delta} \leq \mathcal{G}_{\delta}(n) \leq 2\left(\frac{ne}{\delta}\right)^{\delta}$*, where* $\mathcal{G}_{\delta}(n) = \sum_{i=0}^{\delta} \binom{n}{i}$.

Lemma 39.1.8. *Let* R *and* R' *be two independent* m*-samples from* x*. Assume that* $m \geq \delta$*. Then* $\mathbb{E}[\mathcal{G}_{\delta}(|\mathsf{R}| + |\mathsf{R}'|)] \leq G_{\delta}(2m)$*, where* $G_{\delta}(2m) = 4\,(4em/\delta)^{\delta}$.

Proof: We set $\alpha = 2\left(\frac{e}{\delta}\right)^{\delta}$, $\beta = \delta$, and $f(n) = \alpha n^{\beta}$. Duplicate every element in x, and let x' be the resulting set. Clearly, the size of a $2m$-sample R from x' is the same as $|\mathsf{R}| + |T|$. By Lemma 39.1.7, we have $\mathbb{E}\Big[\mathcal{G}_{\delta}(|\mathsf{R}|)\Big] \leq \mathbb{E}\Big[f(|\mathsf{R}|)\Big] \leq 2\alpha(4m)^{\beta} \leq 4\left(\frac{4em}{\delta}\right)^{\delta}$ The last inequality follows from Lemma 39.1.5. ■

39.2. Proof of the ε-net theorem

Here we are working in the unweighted settings (i.e., the weight of a single element is one).

Theorem 39.2.1 (ε-net theorem, [HW87]). *Let $(X, \mathcal{R})$ be a range space of VC dimension δ, let x be a finite subset of X, and suppose that $0 < \varepsilon \leq 1$ and $\varphi < 1$. Let N be an m-sample from x (see Definition 39.1.1), where*

$$m \geq \max\left(\frac{8}{\varepsilon} \lg \frac{4}{\varphi}, \frac{16\delta}{\varepsilon} \lg \frac{16}{\varepsilon} \right). \tag{39.2}$$

Then N is an ε-net for x with probability at least $1 - \varphi$.

39.2.1. The proof

39.2.1.1. Reduction to double sampling

Let $n = |\mathsf{x}|$. Let N be the m-sample from x. Let $\mathcal{E}_1$ be the probability that N fails to be an ε-net. Namely,

$$\mathcal{E}_1 = \left\{ \exists \mathbf{r} \in \mathcal{R} \;\middle|\; |\mathbf{r} \cap \mathsf{x}| \geq \varepsilon n \quad \text{and} \quad \mathbf{r} \cap N = \emptyset \right\}.$$

(Namely, there exists a "heavy" range $\mathbf{r}$ that does not contain any point of N.) To complete the proof, we must show that $\mathbb{P}[\mathcal{E}_1] \leq \varphi$. Let T be another m-sample generated in a similar fashion to N. Let $\mathcal{E}_2$ be the event that N fails but T "works". Formally

$$\mathcal{E}_2 = \left\{ \exists \mathbf{r} \in \mathcal{R} \;\middle|\; |\mathbf{r} \cap \mathsf{x}| \geq \varepsilon n,\ \mathbf{r} \cap N = \emptyset,\ \text{and}\ |\mathbf{r} \cap T| \geq \frac{\varepsilon m}{2} \right\}.$$

Intuitively, since $\mathbb{E}[|\mathbf{r} \cap \mathsf{x}|] \geq \varepsilon m$, we have that for the range $\mathbf{r}$ that N fails for, it follows with "good" probability that $|\mathbf{r} \cap T| \geq \varepsilon m/2$. Namely, $\mathcal{E}_1$ and $\mathcal{E}_2$ have more or less the same probability.

Claim 39.2.2. $\mathbb{P}[\mathcal{E}_2] \leq \mathbb{P}[\mathcal{E}_1] \leq 2\,\mathbb{P}[\mathcal{E}_2]$.

Proof: Clearly, $\mathcal{E}_2 \subseteq \mathcal{E}_1$, and thus $\mathbb{P}[\mathcal{E}_2] \leq \mathbb{P}[\mathcal{E}_1]$. As for the other part, note that by the definition of conditional probability, we have

$$\mathbb{P}[\mathcal{E}_2 \mid \mathcal{E}_1] = \mathbb{P}[\mathcal{E}_2 \cap \mathcal{E}_1] / \mathbb{P}[\mathcal{E}_1] = \mathbb{P}[\mathcal{E}_2] / \mathbb{P}[\mathcal{E}_1].$$

It is thus enough to show that $\mathbb{P}[\mathcal{E}_2 \mid \mathcal{E}_1] \geq 1/2$.

Assume that $\mathcal{E}_1$ occurs. There is $\mathbf{r} \in \mathcal{R}$, such that $|\mathbf{r} \cap \mathsf{x}| > \varepsilon n$ and $\mathbf{r} \cap N = \emptyset$. The required probability is at least the probability that for this specific $\mathbf{r}$, we have $X = |\mathbf{r} \cap T| \geq \frac{\varepsilon n}{2}$. The variable X is a sum of $t = |\mathbf{r} \cap \mathsf{x}| \geq \varepsilon n$ random independent 0/1 variables, each one has probability m/n to be one. Setting $\mu = \mathbb{E}[X] = tm/n \geq \varepsilon m$ and $\xi = 1/2$, we have by Chernoff's inequality that

$$\mathbb{P}[|\mathbf{r} \cap T| \leq \varepsilon m/2] \leq \mathbb{P}[X < (1 - \xi)\mu] \leq \exp\left(-\mu \xi^2/2\right) = \exp(-\varepsilon m/8) < 1/2,$$

if $\varepsilon m \geq 8$. Thus, for $\mathbf{r} \in \mathcal{E}_1$, we have $\mathbb{P}[\mathcal{E}_2] / \mathbb{P}[\mathcal{E}_1] \geq \mathbb{P}\left[|\mathbf{r} \cap T| \geq \frac{\varepsilon m}{2}\right] = 1 - \mathbb{P}\left[|\mathbf{r} \cap T| < \frac{\varepsilon m}{2}\right] \geq \frac{1}{2}$. ■

Claim 39.2.2 implies that to bound the probability of $\mathcal{E}_1$, it is enough to bound the probability of $\mathcal{E}_2$. Let

$$\mathcal{E}_2' = \left\{ \exists \mathbf{r} \in \mathcal{R} \;\middle|\; \mathbf{r} \cap N = \emptyset \text{ and } |\mathbf{r} \cap T| \geq \frac{\varepsilon m}{2} \right\}.$$

Clearly, $\mathcal{E}_2 \subseteq \mathcal{E}_2'$. Thus, bounding the probability of $\mathcal{E}_2'$ is enough to prove Theorem 39.2.1. Note, however, that a shocking thing happened! We no longer have x participating in our event. Namely, we turned bounding an event that depends on a global quantity (i.e., the ground set x) into bounding a quantity that depends only on a local quantity/experiment (involving only N and T). This is the crucial idea in this proof.

39.2.1.2. Using double sampling to finish the proof

Claim 39.2.3. $\mathbb{P}[\mathcal{E}_2] \leq \mathbb{P}\Big[\mathcal{E}_2'\Big] \leq 2^{-\varepsilon m/2} G_\delta(2m)$.

Proof: We fix the content of $\mathsf{R} = N \cup T$. The range space $(\mathsf{R}, \mathcal{R}_{|\mathsf{R}})$ has $\mathcal{G}_\delta(|\mathsf{R}|)$ ranges. Fix a range $\mathbf{r}$ in this range space. Let $T = \mathbf{r} \cap \mathsf{R}$. If $b = |T| < \varepsilon m/2$ then the $\mathcal{E}_2'$ can not happened. Otherwise, the probability that $\mathbf{r}$ is a bad range is $\mathbb{P}\Big[T \subseteq T \text{ and } T \cap N = \emptyset \;\Big|\; T \subseteq \mathsf{R}\Big] \leq \frac{1}{2^b}$, by Lemma 39.1.2. In particular, by the union bound over all ranges, we have $\mathbb{P}\Big[\mathcal{E}_2' \mid \mathsf{R}\Big] \leq 2^{-\varepsilon m/2} \mathcal{G}_\delta(|\mathsf{R}|)$. As such, we have

$$\mathbb{P}[\mathcal{E}_2'] = \sum_{\mathsf{R}} \mathbb{P}[\mathcal{E}_2' \mid \mathsf{R}]\, \mathbb{P}[\mathsf{R}] \leq \sum_{\mathsf{R}} 2^{-\varepsilon m/2} \mathcal{G}_\delta(|\mathsf{R}|)\, \mathbb{P}[\mathsf{R}] \leq 2^{-\varepsilon m/2}\, \mathbb{E}\Big[\mathcal{G}_\delta(|\mathsf{R}|)\Big] \leq 2^{-\varepsilon m/2} G_\delta(2m).$$

by Lemma 39.1.8. ■

Proof of Theorem 39.2.1. By Claim 39.2.2 and Claim 39.2.3, we have that $\mathbb{P}[\mathcal{E}_1] \leq 2 \cdot 2^{-\varepsilon m/2} G_\delta(2m)$. It thus remains to verify that if m satisfies Eq. (39.2), then the above is smaller than φ. Which is equivalent to

$$\begin{aligned} 2 \cdot 2^{-\varepsilon m/2} G_\delta(2m) \leq \varphi &\iff 16 \cdot 2^{-\varepsilon m/2} \left(\frac{4em}{\delta}\right)^\delta \leq \varphi \iff -4 + \frac{\varepsilon m}{2} - \delta \lg\left(\frac{4em}{\delta}\right) \geq \lg \frac{1}{\varphi} \\ &\iff \left(\frac{\varepsilon m}{8} - 4 - \delta \lg \frac{4e}{\delta}\right) + \left(\frac{\varepsilon m}{8} - \lg \frac{1}{\varphi}\right) + \left(\frac{\varepsilon m}{4} - \delta \lg\left(\frac{m}{\delta}\right)\right) \geq 0 \end{aligned}$$

We remind the reader that the value of m we pick is such that $m \geq \max\left(\frac{8}{\varepsilon} \lg \frac{4}{\varphi}, \frac{16\delta}{\varepsilon} \lg \frac{16}{\varepsilon}\right)$. In particular, $m \geq 64\delta/\varepsilon$ and $-4 - \delta \lg\left(\frac{4e}{\delta}\right) \geq -4 - 4\delta \leq -8\delta \geq -\varepsilon m/8$. Similarly, by the choice of m, we have $\varepsilon m/8 \geq \lg \frac{1}{\varphi}$. As such, we need to show that $\frac{\varepsilon m}{4} \geq \delta \lg\left(\frac{m}{\delta}\right) \iff m \geq \frac{4\delta}{\varepsilon} \lg \frac{m}{\delta}$, and one can verify using some easy but tedious calculations that this holds if $m \geq \frac{16\delta}{\varepsilon} \lg \frac{16}{\varepsilon}$. ■

References

[Har11] S. Har-Peled. *Geometric Approximation Algorithms*. Vol. 173. Math. Surveys & Monographs. Boston, MA, USA: Amer. Math. Soc., 2011.

[HW87] D. Haussler and E. Welzl. ε-nets and simplex range queries. *Discrete Comput. Geom.*, 2: 127–151, 1987.

Chapter 40

Finite Metric Spaces and Partitions

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

40.1. Finite Metric Spaces

Definition 40.1.1. A ***metric space*** is a pair $(\mathcal{X}, \mathsf{d})$ where $\mathcal{X}$ is a set and $\mathsf{d} : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ is a ***metric***, satisfying the following axioms:

(i) $\mathsf{d}(x, y) = 0 \iff x = y$,
(ii) $\mathsf{d}(x, y) = \mathsf{d}(y, x)$, and
(iii) $\mathsf{d}(x, y) + \mathsf{d}(y, z) \geq \mathsf{d}(x, z)$ (triangle inequality).

The plane, $\mathbb{R}^2$, with the regular Euclidean distance is a metric space.

Of special interest is the finite case, where $\mathcal{X}$ is an n-point set. Then, the function d can be specified by $\binom{n}{2}$ real numbers. Alternatively, one can think about $(\mathcal{X}, \mathsf{d})$ as a weighted complete graph, where positive weights are specified on the edges, and these weights comply with the triangle inequality.

Finite metric spaces rise naturally from (sparse) graphs. Indeed, let $\mathsf{G} = (\mathcal{X}, \mathsf{E})$ be an undirected weighted graph defined over $\mathcal{X}$, and let $\mathsf{d}_{\mathsf{G}}(x, y)$ be the length of the shortest path between x and y in G. It is easy to verify that $(\mathcal{X}, \mathsf{d}_G)$ is a finite metric space. As such if the graph G is sparse, it provides a compact representation to the finite space $(\mathcal{X}, \mathsf{d}_G)$.

Definition 40.1.2. Let $(\mathcal{X}, d)$ be an n-point metric space. We denote the ***open ball*** of radius r about $x \in \mathcal{X}$, by $\mathbf{b}(x, r) = \{y \in \mathcal{X} \mid \mathsf{d}(x, y) < r\}$.

Underling our discussion of metric spaces are algorithmic applications. The hardness of various computational problems depends heavily on the structure of the finite metric space. Thus, given a finite metric space, and a computational task, it is natural to try to map the given metric space into a new metric where the task at hand becomes easy.

Example 40.1.3. Computing the diameter of a point set is not trivial in two dimensions (if one wants near linear running time), but is easy in one dimension. Thus, if we could map points in two dimensions into points in one dimension, such that the diameter is preserved, then computing the diameter becomes easy. This approach yields an efficient approximation algorithm, see Exercise 40.7.3 below.

Of course, this mapping from one metric space to another, is going to introduce error. Naturally, one would like to minimize the error introduced by such a mapping.

Definition 40.1.4. Let $(\mathcal{X}, \mathsf{d}_{\mathcal{X}})$ and $(\mathcal{Y}, \mathsf{d}_{\mathcal{Y}})$ be two metric spaces. A mapping $f : \mathcal{X} \to \mathcal{Y}$ is an ***embedding***, and it is ***C-Lipschitz*** if $\mathsf{d}_Y(f(x), f(y)) \leq C \cdot \mathsf{d}_{\mathcal{X}}(x, y)$ for all $x, y \in \mathcal{X}$. The mapping f is ***K-bi-Lipschitz*** if there exists a $C > 0$ such that

$$CK^{-1} \cdot \mathsf{d}_{\mathcal{X}}(x, y) \leq \mathsf{d}_Y(f(x), f(y)) \leq C \cdot \mathsf{d}_{\mathcal{X}}(x, y),$$

for all $x, y \in \mathcal{X}$.

The least K for which f is K-bi-Lipschitz is the ***distortion*** of f, and is denoted $\operatorname{dist}(f)$. The least distortion with which $\mathcal{X}$ may be embedded in $\mathcal{Y}$ is denoted $c_{\mathcal{Y}}(\mathcal{X})$.

Informally, if $f : \mathcal{X} \to \mathcal{Y}$ has distortion K, then the distances in $\mathcal{X}$ and $f(\mathcal{X}) \subseteq \mathcal{Y}$ are the same up to a factor of K (one might need to scale up the distances by some constant C).

There are several powerful results about low distortion embeddings that would be presented:

(I) **Probabilistic trees**. Every finite metric can be randomly embedded into a tree such that the "expected" distortion for a specific pair of points is $O(\log n)$.

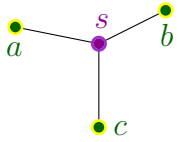
(II) **Bourgain embedding**. Any n-point metric space can be embedded into (finite dimensional) euclidean metric space with $O(\log n)$ distortion.

(III) **Johnson-Lindenstrauss lemma**. Any n-point set in Euclidean space with the regular Euclidean distance can be embedded into $\mathbb{R}^k$ with distortion $(1 + \varepsilon)$, where $k = O(\varepsilon^{-2} \log n)$.

40.2. Examples

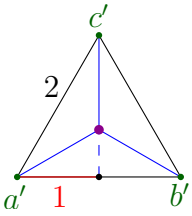
What is distortion? When considering a mapping $f : \mathcal{X} \to \mathbb{R}^d$ of a metric space $(\mathcal{X}, \mathsf{d})$ to $\mathbb{R}^d$, it would useful to observe that since $\mathbb{R}^d$ can be scaled, we can consider f to be an expansion (i.e., no distances shrink). Furthermore, we can assume that there is at least one pair of points $x, y \in \mathcal{X}$, such that $\mathsf{d}(x, y) = \|x - y\|$. As such, we have $\operatorname{dist}(f) = \max_{x,y} \frac{\|x-y\|}{\mathsf{d}(x,y)}$.

Why is distortion necessary? Consider the a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with one vertex s connected to three other vertices a, b, c, where the weights on the edges are all one (i.e., G is the star graph with three leafs). We claim that G can not be embedded into Euclidean space with distortion $\leq \sqrt{2}$. Indeed, consider the associated metric space $(\mathsf{V}, \mathsf{d}_G)$ and an (expansive) embedding $f : \mathsf{V} \to \mathbb{R}^d$.



Consider the triangle formed by $\triangle = a'b'c'$, where $a' = f(a), b' = f(b)$ and $c' = f(c)$. Next, consider the following quantity $\max(\|a' - s'\|, \|b' - s'\|, \|c' - s'\|)$ which lower bounds the distortion of f. This quantity is minimized when $r = \|a' - s'\| = \|b' - s'\| = \|c' - s'\|$. Namely, s' is the center of the smallest enclosing circle of $\triangle$. However, r is minimized when all the edges of $\triangle$ are of equal length, and are of length $\mathsf{d}_G(a, b) = 2$. It follows that $\operatorname{dist}(f) \geq r \geq 2/\sqrt{3}$.

This quantity is minimized when $r = \|a' - s'\| = \|b' - s'\| = \|c' - s'\|$. Namely, s' is the center of the smallest enclosing circle of $\triangle$. However, r is minimized when all the edges of $\triangle$ are of equal length and are of length $\mathsf{d}_{\mathsf{G}}(a, b) = 2$. Observe that the height of the equilateral triangle with sidelength 2 is $h = \sqrt{3}$, and the radius of its inscribing circle is $r = (2/3)h = 2/\sqrt{3}$; see the figure on the right. As such, it follows that $\operatorname{dist}(f) \geq r = 2/\sqrt{3}$.



Note that the above argument is independent of the target dimension d. A packing argument shows that embedding the star graph with n leaves into $\mathbb{R}^d$ requires distortion $\Omega\left(n^{1/d}\right)$; see Exercise **??**. It is known that $\Omega(\log n)$ distortion is necessary in the worst case when embedding a graph into Euclidean space (this is shown using expanders). A proof of distortion $\Omega(\log n / \log \log n)$ is sketched in the bibliographical notes.

40.2.1. Hierarchical Tree Metrics

The following metric is quite useful in practice, and nicely demonstrate why algorithmically finite metric spaces are useful.

Definition 40.2.1. ***Hierarchically well-separated tree*** (HST) is a metric space defined on the leaves of a rooted tree T. To each vertex $u \in T$ there is associated a label $\Delta_u \geq 0$ such that $\Delta_u = 0$ if and only if u is a leaf of T. The labels are such that if a vertex u is a child of a vertex v then $\Delta_u \leq \Delta_v$. The distance between two leaves $x, y \in T$ is defined as $\Delta_{\mathrm{lca}(x,y)}$, where $\mathrm{lca}(x, y)$ is the least common ancestor of x and y in T.

A HST T is a ***k-HST*** if for a vertex $v \in T$, we have that $\Delta_v \leq \Delta_{\overline{\mathrm{p}}(v)}/k$, where $\overline{\mathrm{p}}(v)$ is the parent of v in T.

Note that a HST is a very limited metric. For example, consider the cycle $G = C_n$ of n vertices, with weight one on the edges, and consider an expansive embedding f of G into a HST HST. It is easy to verify, that there must be two consecutive nodes of the cycle, which are mapped to two different subtrees of the root r of HST. Since HST is expansive, it follows that $\Delta_r \geq n/2$. As such, $\mathrm{dist}(f) \geq n/2$. Namely, HSTs fail to faithfully represent even very simple metrics.

40.2.2. Clustering

One natural problem we might want to solve on a graph (i.e., finite metric space) $(\mathcal{X}, \mathsf{d})$ is to partition it into clusters. One such natural clustering is the ***k-median clustering***, where we would like to choose a set $C \subseteq \mathcal{X}$ of k centers, such that $\nu_C(\mathcal{X}, \mathsf{d}) = \sum_{\mathsf{u} \in \mathcal{X}} \mathsf{d}(\mathsf{u}, C)$ is minimized, where $\mathsf{d}(\mathsf{u}, C) = \min_{c \in C} \mathsf{d}(\mathsf{u}, c)$ is the distance of u to its closest center in C.

It is known that finding the optimal k-median clustering in a (general weighted) graph is NP-complete. As such, the best we can hope for is an approximation algorithm. However, if the structure of the finite metric space $(\mathcal{X}, \mathsf{d})$ is simple, then the problem can be solved efficiently. For example, if the points of $\mathcal{X}$ are on the real line (and the distance between a and b is just $|a - b|$), then k-median can be solved using dynamic programming.

Another interesting case is when the metric space $(\mathcal{X}, \mathsf{d})$ is a HST. Is not too hard to prove the following lemma. See Exercise 40.7.1.

Lemma 40.2.2. *Let $(\mathcal{X}, \mathsf{d})$ be a HST defined over n points, and let $k > 0$ be an integer. One can compute the optimal k-median clustering of $\mathcal{X}$ in $O(k^2 n)$ time.*

Thus, if we can embed a general graph G into a HST HST, with low distortion, then we could approximate the k-median clustering on G by clustering the resulting HST, and "importing" the resulting partition to the original space. The quality of approximation, would be bounded by the distortion of the embedding of G into HST.

40.3. Random Partitions

Let $(\mathcal{X}, d)$ be a finite metric space. Given a partition $P = \{C_1, \ldots, C_m\}$ of $\mathcal{X}$, we refer to the sets C_i as ***clusters***. We write $\mathcal{P}_{\mathcal{X}}$ for the set of all partitions of $\mathcal{X}$. For $x \in \mathcal{X}$ and a partition $P \in \mathcal{P}_{\mathcal{X}}$ we denote by $P(x)$ the unique cluster of P containing x. Finally, the set of all probability distributions on $\mathcal{P}_{\mathcal{X}}$ is denoted $\mathcal{D}_{\mathcal{X}}$.

The following partition scheme is due to [CKR04].

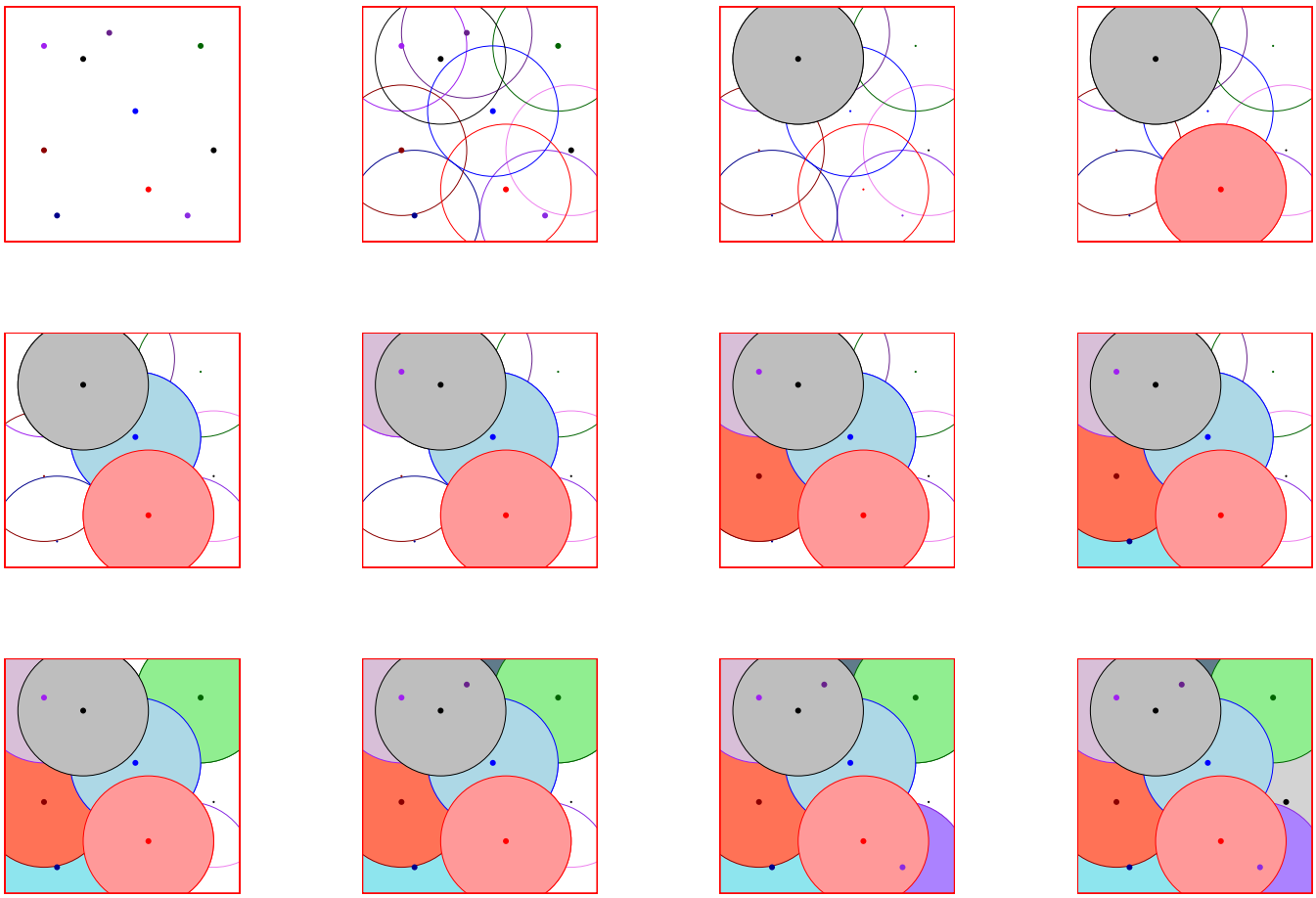


Figure 40.1: An example of the partition of a square (induced by a set of points) as described in Section 40.3.1.

40.3.1. Constructing the partition

Consider a given metric space $(\mathcal{X}, \mathsf{d})$, where $\mathcal{X}$ is a set of n points.

Let $\Delta = 2^u$ be a prescribed parameter, which is the required diameter of the resulting clusters. Choose, uniformly at random, a permutation π of $\mathcal{X}$ and a random value $\alpha \in [1/4, 1/2]$. Let $R = \alpha\Delta$, and observe that it is uniformly distributed in the interval $[\Delta/4, \Delta/2]$.

The partition is now defined as follows: A point $x \in \mathcal{X}$ is assigned to the cluster C_y of y, where y is the first point in the permutation in distance $\leq R$ from x. Formally,

$$C_y = \left\{ x \in \mathcal{X} \;\middle|\; x \in \mathbf{b}(y, R) \text{ and } \pi(y) \leq \pi(z) \text{ for all } z \in \mathcal{X} \text{ with } x \in \mathbf{b}(z, R) \right\}.$$

Let $P = \{C_y\}_{y \in \mathcal{X}}$ denote the resulting partition.

Here is a somewhat more intuitive explanation: Once we fix the radius of the clusters R, we start scooping out balls of radius R centered at the points of the random permutation π. At the ith stage, we scoop out only the remaining mass at the ball centered at x_i of radius r, where x_i is the ith point in the random permutation.

40.3.2. Properties

The following lemma quantifies the probability of a (crystal) ball of radius t centered at a point x is fully contained in one of the clusters of the partition? (Otherwise, the crystal ball is of course broken.)

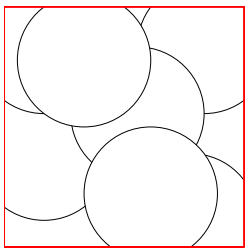


Figure 40.2: The resulting partition.

Lemma 40.3.1. *Let $(\mathcal{X}, d)$ be a finite metric space, $\Delta = 2^u$ a prescribed parameter, and let P be the partition of $\mathcal{X}$ generated by the above random partition. Then the following holds:*

(i) For any $C \in P$, we have $\operatorname{diam}(C) \leq \Delta$.

(ii) Let x be any point of $\mathcal{X}$, and t a parameter $\leq \Delta/8$. Then,

$$\mathbb{P}\Big[\mathbf{b}(x,t) \nsubseteq P(x)\Big] \leq \frac{8t}{\Delta} \ln \frac{b}{a},$$

where $a = |\mathbf{b}(x, \Delta/8)|$, and $b = |\mathbf{b}(x, \Delta)|$.

Proof: Since $C_y \subseteq \mathbf{b}(y, R)$, we have that $\operatorname{diam}(C_y) \leq \Delta$, and thus the first claim holds.

Let U be the set of points of $\mathbf{b}(x, \Delta)$, such that $w \in U$ iff $\mathbf{b}(w, R) \cap \mathbf{b}(x, t) \neq \emptyset$. Arrange the points of U in increasing distance from x, and let $w_1, \ldots, w_{b'}$ denote the resulting order, where $b' = |U|$. Let $I_k = [d(x, w_k) - t, d(x, w_k) + t]$ and write $\mathcal{E}_k$ for the event that w_k is the first point in π such that $\mathbf{b}(x, t) \cap C_{w_k} \neq \emptyset$, and yet $\mathbf{b}(x, t) \nsubseteq C_{w_k}$. Note that if $w_k \in \mathbf{b}(x, \Delta/8)$, then $\mathbb{P}[\mathcal{E}_k] = 0$ since $\mathbf{b}(x, t) \subseteq \mathbf{b}(x, \Delta/8) \subseteq \mathbf{b}(w_k, \Delta/4) \subseteq \mathbf{b}(w_k, R)$.

In particular, $w_1, \ldots, w_a \in \mathbf{b}(x, \Delta/8)$ and as such $\mathbb{P}[\mathcal{E}_1] = \cdots = \mathbb{P}[\mathcal{E}_a] = 0$. Also, note that if $\mathsf{d}(x, w_k) < R - t$ then $\mathbf{b}(w_k, R)$ contains $\mathbf{b}(x, t)$ and as such $\mathcal{E}_k$ can not happen. Similarly, if $\mathsf{d}(x, w_k) > R + t$ then $\mathbf{b}(w_k, R) \cap \mathbf{b}(x, t) = \emptyset$ and $\mathcal{E}_k$ can not happen. As such, if $\mathcal{E}_k$ happen then $R - t \leq \mathsf{d}(x, w_k) \leq R + t$. Namely, if $\mathcal{E}_k$ happen then $R \in I_k$. Namely, $\mathbb{P}[\mathcal{E}_k] = \mathbb{P}[\mathcal{E}_k \cap (R \in I_k)] = \mathbb{P}[R \in I_k] \cdot \mathbb{P}[\mathcal{E}_k \,|\, R \in I_k]$. Now, R is uniformly distributed in the interval $[\Delta/4, \Delta/2]$, and I_k is an interval of length $2t$. Thus, $\mathbb{P}[R \in I_k] \leq 2t/(\Delta/4) = 8t/\Delta$.

Next, to bound $\mathbb{P}[\mathcal{E}_k \,|\, R \in I_k]$, we observe that $w_1, \ldots, w_{k-1}$ are closer to x than w_k and their distance to $\mathbf{b}(x, t)$ is smaller than R. Thus, if any of them appear before w_k in π then $\mathcal{E}_k$ does not happen. Thus, $\mathbb{P}[\mathcal{E}_k \,|\, R \in I_k]$ is bounded by the probability that w_k is the first to appear in π out of $w_1, \ldots, w_k$. But this probability is $1/k$, and thus $\mathbb{P}[\mathcal{E}_k \,|\, R \in I_k] \leq 1/k$.

We are now ready for the kill. Indeed,

$$\begin{aligned}
\mathbb{P}\Big[\mathbf{b}(x,t) \nsubseteq P(x)\Big] &= \sum_{k=1}^{b'} \mathbb{P}[\mathcal{E}_k] = \sum_{k=a+1}^{b'} \mathbb{P}[\mathcal{E}_k] = \sum_{k=a+1}^{b'} \mathbb{P}[R \in I_k] \cdot \mathbb{P}[\mathcal{E}_k \,|\, R \in I_k] \\
&\leq \sum_{k=a+1}^{b'} \frac{8t}{\Delta} \cdot \frac{1}{k} \leq \frac{8t}{\Delta} \ln \frac{b'}{a} \leq \frac{8t}{\Delta} \ln \frac{b}{a},
\end{aligned}$$

since $\sum_{k=a+1}^{b} \frac{1}{k} \leq \int_a^b \frac{dx}{x} = \ln \frac{b}{a}$ and $b' \leq b$. ■

40.4. Probabilistic embedding into trees

In this section, given n-point finite metric $(\mathcal{X}, \mathsf{d})$. we would like to embed it into a HST. As mentioned above, one can verify that for any embedding into HST, the distortion in the worst case is $\Omega(n)$. Thus, we define

a randomized algorithm that embed $(\mathcal{X}, d)$ into a tree. Let T be the resulting tree, and consider two points $x, y \in \mathcal{X}$. Consider the ***random variable*** $\mathsf{d}_T(x, y)$. We constructed the tree T such that distances never shrink; i.e. $\mathsf{d}(x, y) \leq \mathsf{d}_T(x, y)$. The ***probabilistic distortion*** of this embedding is $\max_{x,y} \mathbb{E}\left[\frac{\mathsf{d}_T(x,y)}{\mathsf{d}(x,y)}\right]$. Somewhat surprisingly, one can find such an embedding with logarithmic probabilistic distortion.

Theorem 40.4.1. *Given n-point metric $(\mathcal{X}, d)$ one can randomly embed it into a 2-HST with probabilistic distortion $\leq 24 \ln n$.*

Proof: The construction is recursive. Let $\operatorname{diam}(P)$, and compute a random partition of $\mathcal{X}$ with cluster diameter $\operatorname{diam}(P)/2$, using the construction of Section 40.3.1. We recursively construct a 2-HST for each cluster, and hang the resulting clusters on the root node v, which is marked by $\Delta_v = \operatorname{diam}(P)$. Clearly, the resulting tree is a 2-HST.

For a node $v \in T$, let $\mathcal{X}(v)$ be the set of points of $\mathcal{X}$ contained in the subtree of v.

For the analysis, assume $\operatorname{diam}(P) = 1$, and consider two points $x, y \in \mathcal{X}$. We consider a node $v \in T$ to be in level i if $\operatorname{level}(v) = \lceil \lg \Delta_v \rceil = i$. The two points x and y correspond to two leaves in T, and let $\widehat{u}$ be the least common ancestor of x and y in t. We have $\mathsf{d}_T(x, y) \leq 2^{\operatorname{level}(v)}$. Furthermore, note that along a path the levels are strictly monotonically increasing.

Being more conservative, let w be the first ancestor of x, such that $\mathbf{b} = \mathbf{b}(x, \mathsf{d}(x, y))$ is not completely contained in $\mathcal{X}(u_1), \ldots, \mathcal{X}(u_m)$, where $u_1, \ldots, u_m$ are the children of w. Clearly, $\operatorname{level}(w) > \operatorname{level}(\widehat{u})$. Thus, $\mathsf{d}_T(x, y) \leq 2^{\operatorname{level}(w)}$.

Consider the path σ from the root of T to x, and let $\mathcal{E}_i$ be the event that $\mathbf{b}$ is not fully contained in $\mathcal{X}(v_i)$, where v_i is the node of σ of level i (if such a node exists). Furthermore, let Y_i be the indicator variable which is 1 if $\mathcal{E}_i$ is the first to happened out of the sequence of events $\mathcal{E}_0, \mathcal{E}_{-1}, \ldots$. Clearly, $\mathsf{d}_T(x, y) \leq \sum Y_i 2^i$.

Let $t = \mathsf{d}(x, y)$ and $j = \lfloor \lg \mathsf{d}(x, y) \rfloor$, and $n_i = \left|\mathbf{b}(x, 2^i)\right|$ for $i = 0, \ldots, -\infty$. We have

$$\mathbb{E}[\mathsf{d}_T(x, y)] \leq \sum_{i=j}^{0} \mathbb{E}[Y_i]\, 2^i \leq \sum_{i=j}^{0} 2^i\, \mathbb{P}\left[\mathcal{E}_i \cap \overline{\mathcal{E}_{i-1}} \cap \overline{\mathcal{E}_{i-1}} \cdots \overline{\mathcal{E}_0}\right] \leq \sum_{i=j}^{0} 2^i \cdot \frac{8t}{2^i} \ln \frac{n_i}{n_{i-3}},$$

by Lemma 40.3.1. Thus,

$$\mathbb{E}[\mathsf{d}_T(x, y)] \leq 8t \ln \left(\prod_{i=j}^{0} \frac{n_i}{n_{i-3}} \right) \leq 8t \ln(n_0 \cdot n_1 \cdot n_2) \leq 24t \ln n.$$

It thus follows, that the expected distortion for x and y is $\leq 24 \ln n$. ■

40.4.1. Application: approximation algorithm for k-median clustering

Let $(\mathcal{X}, \mathsf{d})$ be a n-point metric space, and let k be an integer number. We would like to compute the optimal k-median clustering. Number, find a subset $C_{\text{opt}} \subseteq \mathcal{X}$, such that $\nu_{C_{\text{opt}}}(\mathcal{X}, \mathsf{d})$ is minimized, see Section 40.2.2. To this end, we randomly embed $(\mathcal{X}, \mathsf{d})$ into a HST HST using Theorem 40.4.1. Next, using Lemma 40.2.2, we compute the optimal k-median clustering of HST. Let C be the set of centers computed. We return C together with the partition of $\mathcal{X}$ it induces as the required clustering.

Theorem 40.4.2. *Let $(\mathcal{X}, \mathsf{d})$ be a n-point metric space. One can compute in polynomial time a k-median clustering of $\mathcal{X}$ which has expected price $O(\alpha \log n)$, where α is the price of the optimal k-median clustering of $(\mathcal{X}, \mathsf{d})$.*

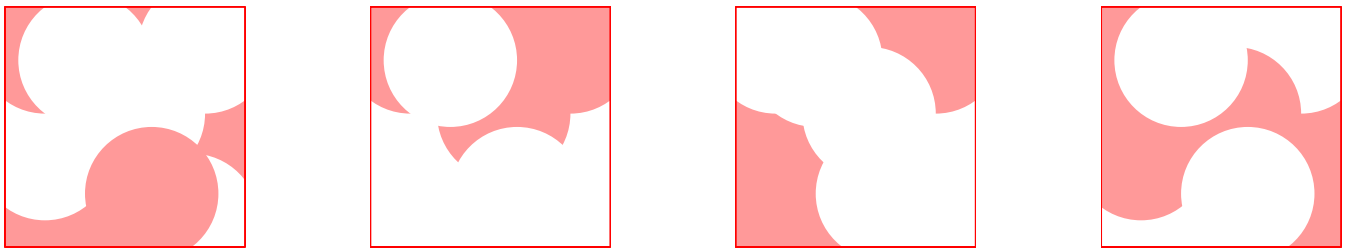


Figure 40.3: Examples of the sets resulting from the partition of Figure 40.1 and taking clusters into a set with probability $1/2$.

Proof: The algorithm is described above, and the fact that its running time is polynomial can be easily be verified. To prove the bound on the quality of the clustering, for any point $p \in \mathcal{X}$, let $\mathrm{cen}(p)$ denote the closest point in C_{opt} to p according to d, where C_{opt} is the set of k-medians in the optimal clustering. Let C be the set of k-medians returned by the algorithm, and let HST be the HST used by the algorithm. We have

$$\beta = \nu_C(\mathcal{X}, \mathsf{d}) \leq \nu_C(\mathcal{X}, \mathsf{d}_{\mathsf{HST}}) \leq \nu_{C_{\mathrm{opt}}}(\mathcal{X}, \mathsf{d}_{\mathsf{HST}}) \leq \sum_{p \in \mathcal{X}} \mathsf{d}_{\mathsf{HST}}(p, C_{\mathrm{opt}}) \leq \sum_{p \in \mathcal{X}} \mathsf{d}_{\mathsf{HST}}(p, \mathrm{cen}(p)).$$

Thus, in expectation we have

$$\begin{aligned}\mathbb{E}[\beta] &= \mathbb{E}\Big[\sum_{p \in \mathcal{X}} \mathsf{d}_{\mathsf{HST}}(p, \mathrm{cen}(p))\Big] = \sum_{p \in \mathcal{X}} \mathbb{E}[\mathsf{d}_{\mathsf{HST}}(p, \mathrm{cen}(p))] = \sum_{p \in \mathcal{X}} O(\mathsf{d}(p, \mathrm{cen}(p)) \log n) \\ &= O\Big((\log n) \sum_{p \in \mathcal{X}} \mathsf{d}(p, \mathrm{cen}(p))\Big) = O\Big(\nu_{C_{\mathrm{opt}}}(\mathcal{X}, \mathsf{d}) \log n\Big),\end{aligned}$$

by linearity of expectation and Theorem 40.4.1. ■

40.5. Embedding any metric space into Euclidean space

Lemma 40.5.1. *Let $(\mathcal{X}, \mathsf{d})$ be a metric, and let $Y \subset \mathcal{X}$. Consider the mapping $f : \mathcal{X} \to \mathbb{R}$, where $f(x) = \mathsf{d}(x, Y) = \min_{y \in Y} \mathsf{d}(x, y)$. Then for any $x, y \in \mathcal{X}$, we have $|f(x) - f(y)| \leq \mathsf{d}(x, y)$. Namely f is nonexpansive.*

Proof: Indeed, let x' and y' be the closet points of Y, to x and y, respectively. Observe that

$$f(x) = \mathsf{d}(x, x') \leq \mathsf{d}(x, y') \leq \mathsf{d}(x, y) + \mathsf{d}(y, y') = \mathsf{d}(x, y) + f(y)$$

by the triangle inequality. Thus, $f(x) - f(y) \leq \mathsf{d}(x, y)$. By symmetry, we have $f(y) - f(x) \leq \mathsf{d}(x, y)$. Thus, $|f(x) - f(y)| \leq \mathsf{d}(x, y)$. ■

40.5.1. The bounded spread case

Let $(\mathcal{X}, \mathsf{d})$ be a n-point metric. The ***spread*** of $\mathcal{X}$, denoted by

$$\Phi(\mathcal{X}) = \frac{\mathrm{diam}(\mathcal{X})}{\min_{x,y \in \mathcal{X}, x \neq y} \mathsf{d}(x, y)},$$

is the ratio between the diameter of $\mathcal{X}$ and the distance between the closest pair of points.

Theorem 40.5.2. *Given a n-point metric $\mathcal{Y} = (\mathcal{X}, d)$, with spread Φ, one can embed it into Euclidean space $\mathbb{R}^k$ with distortion $O(\sqrt{\ln \Phi} \ln n)$, where $k = O(\ln \Phi \ln n)$.*

Proof: Assume that $\operatorname{diam}(\mathcal{Y}) = \Phi$ (i.e., the smallest distance in $\mathcal{Y}$ is 1), and let $r_i = 2^{i-2}$, for $i = 1, \ldots, \alpha$, where $\alpha = \lceil \lg \Phi \rceil$. Let $P_{i,j}$ be a random partition of P with diameter r_i, using Theorem 40.4.1, for $i = 1, \ldots, \alpha$ and $j = 1, \ldots, \beta$, where $\beta = \lceil c \log n \rceil$ and c is a large enough constant to be determined shortly.

For each cluster of $P_{i,j}$ randomly toss a coin, and let $V_{i,j}$ be the all the points of $\mathcal{X}$ that belong to clusters in $P_{i,j}$ that got 'T' in their coin toss. For a point $u \in x$, let

$$f_{i,j}(x) = \mathsf{d}(x, \mathcal{X} \setminus V_{i,j}) = \min_{v \in \mathcal{X} \setminus V_{i,j}} \mathsf{d}(x, v),$$

for $i = 0, \ldots, m$ and $j = 1, \ldots, \beta$. Let $F : \mathcal{X} \to \mathbb{R}^{(m+1)\cdot\beta}$ be the embedding, such that

$$F(x) = (\underbrace{f_{0,1}(x), f_{0,2}(x), \ldots, f_{0,\beta}(x)}_{\text{first n resolution block}}, f_{1,1}(x), f_{1,2}(x), \ldots, f_{1,\beta}(x), \ldots, f_{m,1}(x), f_{m,2}(x), \ldots, f_{\alpha,\beta}(x)).$$

Next, consider two points $x, y \in \mathcal{X}$, with distance $\phi = \mathsf{d}(x, y)$. Let k be an integer such that $r_u \le \phi/2 \le r_{u+1}$. Clearly, in any partition of $P_{u,1}, \ldots, P_{u,\beta}$ the points x and y belong to different clusters. Furthermore, with probability half $x \in V_{u,j}$ and $y \notin V_{u,j}$ or $x \notin V_{u,j}$ and $y \in V_{u,j}$, for $1 \le j \le \beta$.

Let $\mathcal{E}_j$ denote the event that $\mathbf{b}(x, \rho) \subseteq V_{u,j}$ and $y \notin V_{u,j}$, for $j = 1, \ldots, \beta$, where $\rho = \phi/(64 \ln n)$. By Lemma 40.3.1, we have

$$\mathbb{P}\Big[\mathbf{b}(x, \rho) \nsubseteq P_{u,j}(x)\Big] \le \frac{8\rho}{r_u} \ln n \le \frac{\phi}{8 r_u} \le 1/2.$$

Thus,

$$\begin{aligned}\mathbb{P}\Big[\mathcal{E}_j\Big] &= \mathbb{P}\Big[\Big(\mathbf{b}(x, \rho) \subseteq P_{u,j}(x)\Big) \cap \Big(x \in V_{u,j}\Big) \cap \Big(y \notin V_{u,j}\Big)\Big] \\ &= \mathbb{P}\Big[\mathbf{b}(x, \rho) \subseteq P_{u,j}(x)\Big] \cdot \mathbb{P}\Big[x \in V_{u,j}\Big] \cdot \mathbb{P}\Big[y \notin V_{u,j}\Big] \ge 1/8,\end{aligned}$$

since those three events are independent. Notice, that if $\mathcal{E}_j$ happens, than $f_{u,j}(x) \ge \rho$ and $f_{u,j}(y) = 0$.

Let X_j be an indicator variable which is 1 if $\mathcal{E}_j$ happens, for $j = 1, \ldots, \beta$. Let $Z = \sum_j X_j$, and we have $\mu = \mathbb{E}[Z] = \mathbb{E}\Big[\sum_j X_j\Big] \ge \beta/8$. Thus, the probability that only $\beta/16$ of $\mathcal{E}_1, \ldots, \mathcal{E}_\beta$ happens, is $\mathbb{P}[Z < (1 - 1/2)\,\mathbb{E}[Z]]$. By the Chernoff inequality, we have $\mathbb{P}[Z < (1 - 1/2)\,\mathbb{E}[Z]] \le \exp\Big(-\mu 1/(2 \cdot 2^2)\Big) = \exp(-\beta/64) \le 1/n^{10}$, if we set $c = 640$.

Thus, with high probability

$$\|F(x) - F(y)\| \ge \sqrt{\sum_{j=1}^{\beta} \Big(f_{u,j}(x) - f_{u,j}(y)\Big)^2} \ge \sqrt{\rho^2 \frac{\beta}{16}} = \sqrt{\beta}\frac{\rho}{4} = \phi \cdot \frac{\sqrt{\beta}}{256 \ln n}.$$

On the other hand, $\Big|f_{i,j}(x) - f_{i,j}(y)\Big| \le \mathsf{d}(x, y) = \phi \le 64\rho \ln n$. Thus,

$$\|F(x) - F(y)\| \le \sqrt{\alpha\beta(64\rho \ln n)^2} \le 64\sqrt{\alpha\beta}\rho \ln n = \sqrt{\alpha\beta} \cdot \phi.$$

Thus, setting $G(x) = F(x)\frac{256 \ln n}{\sqrt{\beta}}$, we get a mapping that maps two points of distance ϕ from each other to two points with distance in the range $\Big[\phi, \phi \cdot \sqrt{\alpha\beta} \cdot \frac{256 \ln n}{\sqrt{\beta}}\Big]$. Namely, $G(\cdot)$ is an embedding with distortion $O(\sqrt{\alpha} \ln n) = O(\sqrt{\ln \Phi} \ln n)$.

The probability that G fails on one of the pairs, is smaller than $(1/n^{10}) \cdot \binom{n}{2} < 1/n^8$. In particular, we can check the distortion of G for all $\binom{n}{2}$ pairs, and if any of them fail (i.e., the distortion is too big), we restart the process. ■

40.5.2. The unbounded spread case

Our next task, is to extend Theorem 40.5.2 to the case of unbounded spread. Indeed, let $(\mathcal{X}, d)$ be a n-point metric, such that $\operatorname{diam}(\mathcal{X}) \leq 1/2$. Again, we look on the different resolutions $r_1, r_2, \ldots$, where $r_i = 1/2^{i-1}$. For each one of those resolutions r_i, we can embed this resolution into β coordinates, as done for the bounded case. Then we concatenate the coordinates together.

There are two problems with this approach: (i) the number of resulting coordinates is infinite, and (ii) a pair x, y, might be distorted a "lot" because it contributes to all resolutions, not only to its "relevant" resolutions.

Both problems can be overcome with careful tinkering. Indeed, for a resolution r_i, we are going to modify the metric, so that it ignores short distances (i.e., distances $\leq r_i/n^2$). Formally, for each resolution r_i, let $G_i = (\mathcal{X}, \widehat{E}_i)$ be the graph where two points x and y are connected if $\mathsf{d}(x, y) \leq r_i/n^2$. Consider a connected component $C \in G_i$. For any two points $x, y \in C$, we have $\mathsf{d}(x, y) \leq n(r_i/n^2) \leq r_i/n$. Let $\mathcal{X}_i$ be the set of connected components of G_i, and define the distances between two connected components $C, C' \in \mathcal{X}_i$, to be $\mathsf{d}_i(C, C') = \mathsf{d}(C, C') = \min_{c \in C, c' \in C'} \mathsf{d}(c, c')$.

It is easy to verify that $(\mathcal{X}_i, \mathsf{d}_i)$ is a metric space (see Exercise 40.7.2). Furthermore, we can naturally embed $(\mathcal{X}, \mathsf{d})$ into $(\mathcal{X}_i, \mathsf{d}_i)$ by mapping a point $x \in \mathcal{X}$ to its connected components in $\mathcal{X}_i$. Essentially $(\mathcal{X}_i, \mathsf{d}_i)$ is a snapped version of the metric $(\mathcal{X}, d)$, with the advantage that $\Phi((\mathcal{X}, \mathsf{d}_i)) = O(n^2)$. We now embed $\mathcal{X}_i$ into $\beta = O(\log n)$ coordinates. Next, for any point of $\mathcal{X}$ we embed it into those β coordinates, by using the embedding of its connected component in $\mathcal{X}_i$. Let E_i be the embedding for resolution r_i. Namely, $E_i(x) = (f_{i,1}(x), f_{i,2}(x), \ldots, f_{i,\beta}(x))$, where $f_{i,j}(x) = \min(\mathsf{d}_i(x, \mathcal{X} \setminus V_{i,j}), 2r_i)$. The resulting embedding is $F(x) = \oplus E_i(x) = (E_1(x), E_2(x), \ldots,)$.

Since we slightly modified the definition of $f_{i,j}(\cdot)$, we have to show that $f_{i,j}(\cdot)$ is nonexpansive. Indeed, consider two points $x, y \in \mathcal{X}_i$, and observe that

$$\left|f_{i,j}(x) - f_{i,j}(y)\right| \leq \left|\mathsf{d}_i(x, V_{i,j}) - \mathsf{d}_i(y, V_{i,j})\right| \leq \mathsf{d}_i(x, y) \leq \mathsf{d}(x, y),$$

as a simple case analysis[①] shows.

For a pair $x, y \in \mathcal{X}$, and let $\phi = \mathsf{d}(x, y)$. To see that $F(\cdot)$ is the required embedding (up to scaling), observe that, by the same argumentation of Theorem 40.5.2, we have that with high probability

$$\|F(x) - F(y)\| \geq \phi \cdot \frac{\sqrt{\beta}}{256 \ln n}.$$

To get an upper bound on this distance, observe that for i such that $r_i > \phi n^2$, we have $E_i(x) = E_i(y)$. Thus,

$$\begin{aligned}
\|F(x) - F(y)\|^2 &= \sum_i \|E_i(x) - E_i(y)\|^2 = \sum_{i, r_i < \phi n^2} \|E_i(x) - E_i(y)\|^2 \\
&= \sum_{i, \phi/n^2 < r_i < \phi n^2} \|E_i(x) - E_i(y)\|^2 + \sum_{i, r_i < \phi/n^2} \|E_i(x) - E_i(y)\|^2 \\
&= \beta\phi^2 \lg\left(n^4\right) + \sum_{i, r_i < \phi/n^2} (2r_i)^2 \beta \leq 4\beta\phi^2 \lg n + \frac{4\phi^2\beta}{n^4} \leq 5\beta\phi^2 \lg n.
\end{aligned}$$

Thus, $\|F(x) - F(y)\| \leq \phi\sqrt{5\beta \lg n}$. We conclude, that with high probability, $F(\cdot)$ is an embedding of $\mathcal{X}$ into Euclidean space with distortion $\left(\phi\sqrt{5\beta \lg n}\right) / \left(\phi \cdot \frac{\sqrt{\beta}}{256 \ln n}\right) = O(\log^{3/2} n)$.

We still have to handle the infinite number of coordinates problem. However, the above proof shows that we care about a resolution r_i (i.e., it contributes to the estimates in the above proof) only if there is a pair x

[①] Indeed, if $f_{i,j}(x) < \mathsf{d}_i(x, V_{i,j})$ and $f_{i,j}(y) < \mathsf{d}_i(x, V_{i,j})$ then $f_{i,j}(x) = 2r_i$ and $f_{i,j}(y) = 2r_i$, which implies the above inequality. If $f_{i,j}(x) = \mathsf{d}_i(x, V_{i,j})$ and $f_{i,j}(y) = \mathsf{d}_i(x, V_{i,j})$ then the inequality trivially holds. The other option is handled in a similar fashion.

and y such that $r_i/n^2 \le \mathsf{d}(x,y) \le r_i n^2$. Thus, for every pair of distances there are $O(\log n)$ relevant resolutions. Thus, there are at most $\eta = O(n^2\beta \log n) = O(n^2 \log^2 n)$ relevant coordinates, and we can ignore all the other coordinates. Next, consider the affine subspace h that spans $F(P)$. Clearly, it is $n-1$ dimensional, and consider the projection $G : \mathbb{R}^\eta \to \mathbb{R}^{n-1}$ that projects a point to its closest point in h. Clearly, $G(F(\cdot))$ is an embedding with the same distortion for P, and the target space is of dimension $n-1$.

Note, that all this process succeeds with high probability. If it fails, we try again. We conclude:

Theorem 40.5.3 (Low quality Bourgain theorem). *Given a n-point metric M, one can embed it into Euclidean space of dimension $n-1$, such that the distortion of the embedding is at most $O(\log^{3/2} n)$.*

Using the Johnson-Lindenstrauss lemma, the dimension can be further reduced to $O(\log n)$. Being more careful in the proof, it is possible to reduce the dimension to $O(\log n)$ directly.

40.6. Bibliographical notes

The partitions we use are due to Calinescu *et al.* [CKR04]. The idea of embedding into spanning trees is due to Alon *et al.* [AKPW95], which showed that one can get a probabilistic distortion of $2^{O\left(\sqrt{\log n \log\log n}\right)}$. Yair Bartal realized that by allowing trees with additional vertices, one can get a considerably better result. In particular, he showed [Bar96] that probabilistic embedding into trees can be done with polylogarithmic average distortion. He later improved the distortion to $O(\log n \log\log n)$ in [Bar98]. Improving this result was an open question, culminating in the work of Fakcharoenphol *et al.* [FRT04] which achieve the optimal $O(\log n)$ distortion.

Our proof of Lemma 40.3.1 (which is originally from [FRT04]) is taken from [KLMN05]. The proof of Theorem 40.5.3 is by Gupta [Gup00].

A good exposition of metric spaces is available in Matoušek [Mat02].

Embedding into spanning trees. The above embeds the graph into a Steiner tree. A more useful representation, would be a random embedding into a spanning tree. Surprisingly, this can be done, as shown by Emek *et al.* [EEST08]. This was improved to $O(\log n \cdot \log\log n \cdot (\log\log\log n)^3)$[②] by Abraham *et al.* [ABN08a, ABN08b].

Alternative proof of the tree embedding result. Interestingly, if one does not care about the optimal distortion, one can get similar result (for embedding into probabilistic trees), by first embedding the metric into Euclidean space, then reduce the dimension by the Johnson-Lindenstrauss lemma, and finally, construct an HST by constructing a quadtree over the points. The "trick" is to randomly translate the quadtree. It is easy to verify that this yields $O(\log^4 n)$ distortion. See the survey by Indyk [Ind01] for more details. This random shifting of quadtrees is a powerful technique that was used in getting several result, and it is a crucial ingredient in Arora [Aro98] approximation algorithm for Euclidean TSP.

40.7. Exercises

Exercise 40.7.1 (Clustering for HST). Let $(\mathcal{X}, \mathsf{d})$ be a HST defined over n points, and let $k > 0$ be an integer. Provide an algorithm that computes the optimal k-median clustering of $\mathcal{X}$ in $O(k^2 n)$ time.

[Transform the HST into a tree where every node has only two children. Next, run a dynamic programming algorithm on this tree.]

[②] Truely a polyglot of logs.

Exercise 40.7.2 (Partition induced metric).

(a) Give a counter example to the following claim: Let $(\mathcal{X}, \mathsf{d})$ be a metric space, and let P be a partition of $\mathcal{X}$. Then, the pair (P, d') is a metric, where $\mathsf{d}'(C, C') = \mathsf{d}(C, C') = \min_{x \in C, y \in C'} \mathsf{d}(x, y)$ and $C, C' \in P$.

(b) Let $(\mathcal{X}, \mathsf{d})$ be a n-point metric space, and consider the set $U = \left\{ i \;\middle|\; 2^i \le \mathsf{d}(x, y) \le 2^{i+1}, \text{ for } x, y \in \mathcal{X} \right\}$. Prove that $|U| = O(n)$. Namely, there are only n different resolutions that "matter" for a finite metric space.

Exercise 40.7.3 (Computing the diameter via embeddings).

(a) (h:1) Let ℓ be a line in the plane, and consider the embedding $f : \mathbb{R}^2 \to \ell$, which is the projection of the plane into ℓ. Prove that f is 1-Lipschitz, but it is not K-bi-Lipschitz for any constant K.

(b) (h:3) Prove that one can find a family of projections $\mathcal{F}$ of size $O(1/\sqrt{\varepsilon})$, such that for any two points $x, y \in \mathbb{R}^2$, for one of the projections $f \in \mathcal{F}$ we have $\mathsf{d}(f(x), f(y)) \ge (1 - \varepsilon)\mathsf{d}(x, y)$.

(c) (h:1) Given a set P of n in the plane, given a $O(n/\sqrt{\varepsilon})$ time algorithm that outputs two points $x, y \in P$, such that $\mathsf{d}(x, y) \ge (1 - \varepsilon)\mathrm{diam}(P)$, where $\mathrm{diam}(P) = \max_{z,w \in P} \mathsf{d}(z, w)$ is the diameter of P.

(d) (h:2) Given P, show how to extract, in $O(n)$ time, a set $Q \subseteq P$ of size $O(\varepsilon^{-2})$, such that $\mathrm{diam}(Q) \ge (1 - \varepsilon/2)\mathrm{diam}(P)$. (Hint: Construct a grid of appropriate resolution.)

In particular, give an $(1-\varepsilon)$-approximation algorithm to the diameter of P that works in $O(n + \varepsilon^{-2.5})$ time. (There are slightly faster approximation algorithms known for approximating the diameter.)

Acknowledgments

The presentation in this write-up follows closely the insightful suggestions of Manor Mendel.

References

[ABN08a] I. Abraham, Y. Bartal, and O. Neiman. Nearly tight low stretch spanning trees. *Proc. 49th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), 781–790, 2008.

[ABN08b] I. Abraham, Y. Bartal, and O. Neiman. Nearly tight low stretch spanning trees. *CoRR*, abs/0808.2017, 2008. arXiv: 0808.2017.

[AKPW95] N. Alon, R. M. Karp, D. Peleg, and D. West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1): 78–100, 1995.

[Aro98] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. *J. Assoc. Comput. Mach.*, 45(5): 753–782, 1998.

[Bar96] Y. Bartal. Probabilistic approximations of metric space and its algorithmic application. *Proc. 37th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), 183–193, 1996.

[Bar98] Y. Bartal. On approximating arbitrary metrics by tree metrics. *Proc. 30th Annu. ACM Sympos. Theory Comput.* (STOC), 161–168, 1998.

[CKR04] G. Călinescu, H. J. Karloff, and Y. Rabani. Approximation algorithms for the 0-extension problem. *SIAM J. Comput.*, 34(2): 358–372, 2004.

[EEST08] M. Elkin, Y. Emek, D. A. Spielman, and S. Teng. Lower-stretch spanning trees. *SIAM J. Comput.*, 38(2): 608–628, 2008.

[FRT04] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Sys. Sci.*, 69(3): 485–497, 2004.

[Gup00] A. Gupta. *Embeddings of Finite Metrics*. PhD thesis. University of California, Berkeley, 2000.

[Ind01] P. Indyk. Algorithmic applications of low-distortion geometric embeddings. *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), Tutorial. 10–31, 2001.

[KLMN05] R. Krauthgamer, J. R. Lee, M. Mendel, and A. Naor. Measured descent: a new embedding method for finite metric spaces. *Geom. funct. anal. (GAFA)*, 15(4): 839–858, 2005.

[Mat02] J. Matoušek. *Lectures on Discrete Geometry*. Vol. 212. Grad. Text in Math. Springer, 2002.

Chapter 41

Entropy, Randomness, and Information

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "If only once - only once - no matter where, no matter before what audience - I could better the record of the great Rastelli and juggle with thirteen balls, instead of my usual twelve, I would feel that I had truly accomplished something for my country. But I am not getting any younger, and although I am still at the peak of my powers there are moments - why deny it? - when I begin to doubt - and there is a time limit on all of us."
>
> Romain Gary, The talent scout

41.1. The entropy function

Definition 41.1.1. The ***entropy*** in bits of a discrete random variable X is given by

$$\mathbb{H}(X) = -\sum_x \mathbb{P}[X = x] \lg \mathbb{P}[X = x],$$

where $\lg x$ is the logarithm base 2 of x. Equivalently, $\mathbb{H}(X) = \mathbb{E}\left[\lg \frac{1}{\mathbb{P}[X]}\right]$.

The ***binary entropy*** function $\mathbb{H}(p)$ for a random binary variable that is 1 with probability p, is

$$\mathbb{H}(p) = -p \lg p - (1-p) \lg(1-p).$$

We define $\mathbb{H}(0) = \mathbb{H}(1) = 0$.

The function $\mathbb{H}(p)$ is a concave symmetric around $1/2$ on the interval $[0, 1]$ and achieves its maximum at $1/2$. For a concrete example, consider $\mathbb{H}(3/4) \approx 0.8113$ and $\mathbb{H}(7/8) \approx 0.5436$. Namely, a coin that has $3/4$ probably to be heads have higher amount of "randomness" in it than a coin that has probability $7/8$ for heads.

Writing $\lg n = (\ln n)/\ln 2$, we have that

$$\mathbb{H}(p) = \frac{1}{\ln 2}(-p \ln p - (1-p)\ln(1-p))$$

$$\text{and} \quad \mathbb{H}'(p) = \frac{1}{\ln 2}\left(-\ln p - \frac{p}{p} - (-1)\ln(1-p) - \frac{1-p}{1-p}(-1)\right) = \lg \frac{1-p}{p}.$$

Deploying our amazing ability to compute derivative of simple functions once more, we get that

$$\mathbb{H}''(p) = \frac{1}{\ln 2}\frac{p}{1-p}\left(\frac{p(-1) - (1-p)}{p^2}\right) = -\frac{1}{p(1-p)\ln 2}.$$

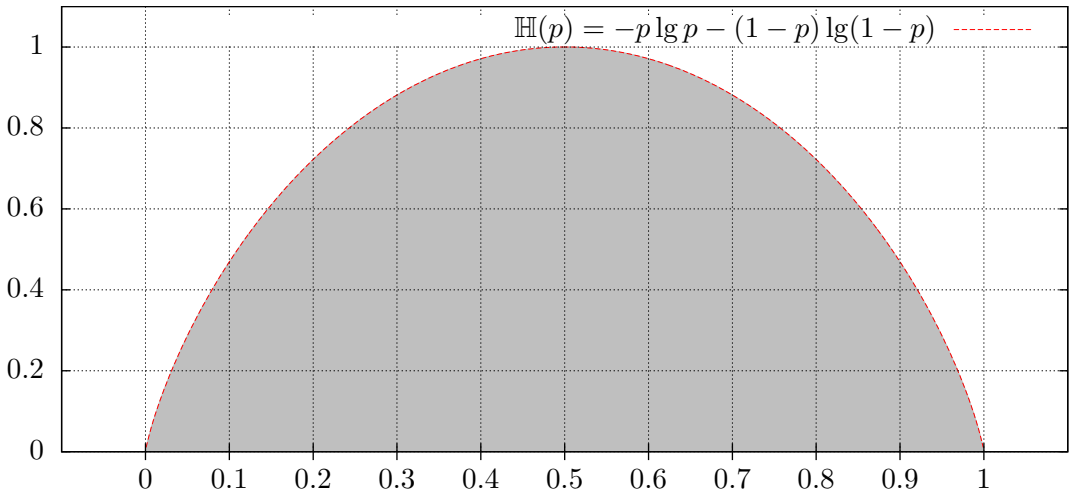


Figure 41.1: The binary entropy function.

Since $\ln 2 \approx 0.693$, we have that $\mathbb{H}''(p) \leq 0$, for all $p \in (0,1)$, and the $\mathbb{H}(\cdot)$ is concave in this range. Also, $\mathbb{H}'(1/2) = 0$, which implies that $\mathbb{H}(1/2) = 1$ is a maximum of the binary entropy. Namely, a balanced coin has the largest amount of randomness in it.

Example 41.1.2. A random variable X that has probability $1/n$ to be i, for $i = 1, \ldots, n$, has entropy $\mathbb{H}(X) = -\sum_{i=1}^{n} \frac{1}{n} \lg \frac{1}{n} = \lg n$.

Note, that the entropy is oblivious to the exact values that the random variable can have, and it is sensitive only to the probability distribution. Thus, a random variables that accepts $-1, +1$ with equal probability has the same entropy (i.e., 1) as a fair coin.

Lemma 41.1.3. *Let X and Y be two independent random variables, and let Z be the random variable (X, T). Then $\mathbb{H}(Z) = \mathbb{H}(X) + \mathbb{H}(Y)$.*

Proof: In the following, summation are over all possible values that the variables can have. By the independence of X and Y we have

$$\begin{aligned}
\mathbb{H}(Z) &= \sum_{x,y} \mathbb{P}[(X,Y) = (x,y)] \lg \frac{1}{\mathbb{P}[(X,Y) = (x,y)]} \\
&= \sum_{x,y} \mathbb{P}[X = x]\, \mathbb{P}[Y = y] \lg \frac{1}{\mathbb{P}[X = x]\, \mathbb{P}[Y = y]} \\
&= \sum_{x} \sum_{y} \mathbb{P}[X = x]\, \mathbb{P}[Y = y] \lg \frac{1}{\mathbb{P}[X = x]} \\
&\quad + \sum_{y} \sum_{x} \mathbb{P}[X = x]\, \mathbb{P}[Y = y] \lg \frac{1}{\mathbb{P}[Y = y]} \\
&= \sum_{x} \mathbb{P}[X = x] \lg \frac{1}{\mathbb{P}[X = x]} + \sum_{y} \mathbb{P}[Y = y] \lg \frac{1}{\mathbb{P}[Y = y]} = \mathbb{H}(X) + \mathbb{H}(Y).
\end{aligned}$$

■

Lemma 41.1.4. *Suppose that nq is integer in the range $[0, n]$. Then $\frac{2^{n\mathbb{H}(q)}}{n+1} \leq \binom{n}{nq} \leq 2^{n\mathbb{H}(q)}$.*

Proof: This trivially holds if $q = 0$ or $q = 1$, so assume $0 < q < 1$. We know that

$$\begin{aligned} & \binom{n}{nq} q^{nq}(1-q)^{n-nq} \le (q + (1-q))^n = 1 \\ \implies \quad & \binom{n}{nq} \le q^{-nq}(1-q)^{-n(1-q)} = 2^{n\left(-q\lg q - (1-q)\lg(1-q)\right)} = 2^{n\mathbb{H}(q)}. \end{aligned}$$

As for the other direction, let

$$\mu(k) = \binom{n}{k} q^k (1-q)^{n-k}.$$

The claim is that $\mu(nq)$ is the largest term in $\sum_{k=0}^{n} \mu(k) = 1$, where $\mu(k) = \binom{n}{k} q^k (1-q)^{n-k}$. Indeed,

$$\Delta_k = \mu(k) - \mu(k+1) = \binom{n}{k} q^k (1-q)^{n-k} \left(1 - \frac{n-k}{k+1}\frac{q}{1-q}\right),$$

and the sign of this quantity is the sign of $(k+1)(1-q) - (n-k)q = k + 1 - kq - q - nq + kq = 1 + k - q - nq$. Namely, $\Delta_k \ge 0$ when $k \ge nq + q - 1$, and $\Delta_k < 0$ otherwise. Namely, $\mu(k) < \mu(k+1)$, for $k < nq$, and $\mu(k) \ge \mu(k+1)$ for $k \ge nq$. Namely, $\mu(nq)$ is the largest term in $\sum_{k=0}^{n} \mu(k) = 1$, and as such it is larger than the average. We have $\mu(nq) = \binom{n}{nq} q^{nq}(1-q)^{n-nq} \ge \frac{1}{n+1}$, which implies

$$\binom{n}{nq} \ge \frac{1}{n+1} q^{-nq}(1-q)^{-(n-nq)} = \frac{1}{n+1} 2^{n\mathbb{H}(q)}.$$ ■

Lemma 41.1.4 can be extended to handle non-integer values of q. This is straightforward, and we omit the easy details.

Corollary 41.1.5. *We have:*

(i) $q \in [0, 1/2] \Rightarrow \binom{n}{\lfloor nq \rfloor} \le 2^{n\mathbb{H}(q)}$.

(ii) $q \in [1/2, 1] \Rightarrow \binom{n}{\lceil nq \rceil} \le 2^{n\mathbb{H}(q)}$.

(iii) $q \in [1/2, 1] \Rightarrow \frac{2^{n\mathbb{H}(q)}}{n+1} \le \binom{n}{\lfloor nq \rfloor}$.

(iv) $q \in [0, 1/2] \Rightarrow \frac{2^{n\mathbb{H}(q)}}{n+1} \le \binom{n}{\lceil nq \rceil}$.

The bounds of Lemma 41.1.4 and Corollary 41.1.5 are loose but sufficient for our purposes. As a sanity check, consider the case when we generate a sequence of n bits using a coin with probability q for head, then by the Chernoff inequality, we will get roughly nq heads in this sequence. As such, the generated sequence Y belongs to $\binom{n}{nq} \approx 2^{n\mathbb{H}(q)}$ possible sequences that have similar probability. As such, $\mathbb{H}(Y) \approx \lg \binom{n}{nq} = n\mathbb{H}(q)$, by Example 41.1.2, this also readily follows from Lemma 41.1.3.

41.2. Extracting randomness

The problem. We are given a random variable X that is chosen uniformly at random from $[\![0 : m-1]\!] = \{0, \ldots, m-1\}$. Our purpose is built an algorithm that given X output a binary string, such that the bits in the binary string can be interpreted as the coin flips of a fair balanced coin. That is, the probability of the ith bit of the output (if it exists) to be 0 (or 1) is exactly half, and the different bits of the output are independent.

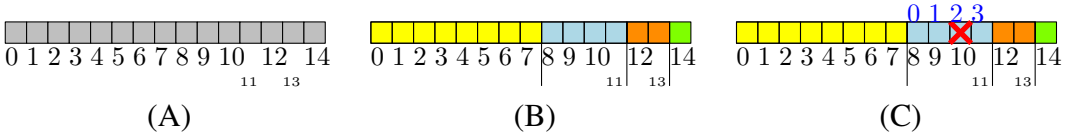


Figure 41.2: (A) $m = 15$. (B) The block decomposition. (C) If $X = 10$, then the extraction output is 2 in base 2, using 2 bits – that is `10`.

Idea. We break the $[\![0 : m-1]\!]$ into consecutive blocks that are powers of two. Given the value of X, we find which block contains it, and we output a binary representation of the location of X in the block containing it, where if a block is length 2^k, then we output k bits.

Entropy can be interpreted as the amount of unbiased random coin flips can be extracted from a random variable.

Definition 41.2.1. An ***extraction function*** $\texttt{Ext}$ takes as input the value of a random variable X and outputs a sequence of bits y, such that $\mathbb{P}[\texttt{Ext}(X) = y \mid |y| = k] = 1/2^k$. whenever $\mathbb{P}[|y| = k] \geq 0$, where $|y|$ denotes the length of y.

As a concrete (easy) example, consider X to be a uniform random integer variable out of $0, \ldots, 7$. All that $\texttt{Ext}(x)$ has to do in this case, is just to compute the binary representation of x.

The definition of the extraction function has two subtleties:

(A) It requires that all extracted sequences of the same length (say k), have the same probability to be output (i.e., $1/2^k$).

(B) If the extraction function can output a sequence of length k, then it needs to be able to output *all* 2^k such binary sequences.

Thus, for X a uniform random integer variable in the range $0, \ldots, 11$, the function $\texttt{Ext}(x)$ can output the binary representation for x if $0 \leq x \leq 7$. However, what do we do if x is between 8 and 11? The idea is to output the binary representation of $x - 8$ as a two bit number. Clearly, Definition 41.2.1 holds for this extraction function, since $\mathbb{P}\left[\texttt{Ext}(X) = 00 \mid |\texttt{Ext}(X)| = 2\right] = 1/4$. as required. This scheme can be of course extracted for any range.

Tedium 41.2.2. For $x \leq y$ positive integers, and any positive integer Δ, we have that

$$\frac{x}{y} \leq \frac{x+\Delta}{y+\Delta} \iff x(y+\Delta) \leq y(x+\Delta) \iff x\Delta \leq y\Delta \iff x \leq y.$$

Theorem 41.2.3. *Suppose that the value of a random variable X is chosen uniformly at random from the integers $\{0, \ldots, m-1\}$. Then there is an extraction function for X that outputs on average (i.e., in expectation) at least $\lfloor \lg m \rfloor - 1 = \lfloor \mathbb{H}(X) \rfloor - 1$ independent and unbiased bits.*

Proof: We represent m as a sum of unique powers of 2, namely $m = \sum_i a_i 2^i$, where $a_i \in \{0, 1\}$. Thus, we decomposed $\{0, \ldots, m-1\}$ into a disjoint union of blocks that have sizes which are distinct powers of 2. If a number falls inside such a block, we output its relative location in the block, using binary representation of the appropriate length (i.e., k if the block is of size 2^k). It is not difficult to verify that this function fulfills the conditions of Definition 41.2.1, and it is thus an extraction function.

Now, observe that the claim holds if m is a power of two, by Example 41.1.2 (i.e., if $m = 2^k$, then $\mathbb{H}(X) = k$). Thus, if m is not a power of 2, then in the decomposition if there is a block of size 2^k, and the X falls inside this block, then the entropy is k.

The remainder of the proof is by induction – assume the claim holds if the range used by the random variable is strictly smaller than m. In particular, let $K = 2^k$ be the largest power of 2 that is smaller than m, and let $U = 2^u$ be the largest power of two such that $U \le m - K \le 2U$.

If the random number $X \in [\![0 : K-1]\!]$, then the scheme outputs k bits. Otherwise, we can think about the extraction function as being recursive and extracting randomness from a random variable $X' = X - K$ that is uniformly distributed in $[\![0 : m-K]\!]$.

By Tedium 41.2.2, we have that

$$\frac{m-K}{m} \le \frac{m-K+(2U+K-m)}{m+(2U+K-m)} = \frac{2U}{2U+K}$$

Let Y be the random variable which is the number of random bits extracted. We have that

$$\begin{aligned}\mathbb{E}[Y] &\ge \frac{K}{m}k + \frac{m-K}{m}(\lfloor \lg(m-K)\rfloor - 1) = k - \frac{m-K}{m}k + \frac{m-K}{m}(u-1) = k + \frac{m-K}{m}(\overbrace{u-k-1}^{<0}) \\ &\ge k - \frac{2U}{2U+K}(u-k-1) = k - \frac{2U}{2U+K}(1+k-u).\end{aligned}$$

If $u = k-1$, then $\mathbb{H}(X) \ge k - \frac{1}{2}\cdot 2 = k-1$, as required. If $u = k-2$ then $\mathbb{H}(X) \ge k - \frac{1}{3}\cdot 3 = k-1$. Finally, if $u < k-2$ then

$$\mathbb{E}[Y] \ge k - \frac{2U}{2U+K}(1+k-u) \ge k - \frac{2U}{K}(1+k-u) = k - \frac{k-u+1}{2^{(k-u+1)-2}} \ge k-1,$$

since $k-u+1 \ge 4$ and $i/2^{i-2} \le 1$ for $i \ge 4$. ■

Theorem 41.2.4. *Consider a coin that comes up heads with probability $p > 1/2$. For any constant $\delta > 0$ and for n sufficiently large:*

(A) One can extract, from an input of a sequence of n flips, an output sequence of $(1-\delta)n\mathbb{H}(p)$ (unbiased) independent random bits.

(B) One can not extract more than $n\mathbb{H}(p)$ bits from such a sequence.

Proof: There are $\binom{n}{j}$ input sequences with exactly j heads, and each has probability $p^j(1-p)^{n-j}$. We map this sequence to the corresponding number in the set $\left\{0,\ldots,\binom{n}{j}-1\right\}$. Note, that this, conditional distribution on j, is uniform on this set, and we can apply the extraction algorithm of Theorem 41.2.3. Let Z be the random variables which is the number of heads in the input, and let B be the number of random bits extracted. We have

$$\mathbb{E}[B] = \sum_{k=0}^{n} \mathbb{P}[Z=k]\,\mathbb{E}\Big[B \,\Big|\, Z=k\Big],$$

and by Theorem 41.2.3, we have $\mathbb{E}\Big[B \,\Big|\, Z=k\Big] \ge \left\lfloor \lg\binom{n}{k}\right\rfloor - 1$. Let $\varepsilon < p - 1/2$ be a constant to be determined shortly. For $n(p-\varepsilon) \le k \le n(p+\varepsilon)$, we have

$$\binom{n}{k} \ge \binom{n}{\lfloor n(p+\varepsilon)\rfloor} \ge \frac{2^{n\mathbb{H}(p+\varepsilon)}}{n+1},$$

by Corollary 41.1.5 (iii). We have

$$\mathbb{E}[B] \;\ge \sum_{k=\lfloor n(p-\varepsilon)\rfloor}^{\lceil n(p-\varepsilon)\rceil} \mathbb{P}[Z=k]\,\mathbb{E}\Big[B \,\Big|\, Z=k\Big] \ge \sum_{k=\lfloor n(p-\varepsilon)\rfloor}^{\lceil n(p-\varepsilon)\rceil} \mathbb{P}[Z=k]\left(\left\lfloor \lg\binom{n}{k}\right\rfloor - 1\right)$$

$$\begin{aligned}
&\geq \sum_{k=\lfloor n(p-\varepsilon)\rfloor}^{\lceil n(p-\varepsilon)\rceil} \mathbb{P}[Z=k]\left(\lg \frac{2^{n\mathbb{H}(p+\varepsilon)}}{n+1} - 2\right) \\
&= (n\mathbb{H}(p+\varepsilon) - \lg(n+1))\,\mathbb{P}[|Z-np| \leq \varepsilon n] \\
&\geq (n\mathbb{H}(p+\varepsilon) - \lg(n+1))\left(1 - 2\exp\left(-\frac{n\varepsilon^2}{4p}\right)\right),
\end{aligned}$$

since $\mu = \mathbb{E}[Z] = np$ and $\mathbb{P}\left[|Z-np| \geq \frac{\varepsilon}{p}pn\right] \leq 2\exp\left(-\frac{np}{4}\left(\frac{\varepsilon}{p}\right)^2\right) = 2\exp\left(-\frac{n\varepsilon^2}{4p}\right)$, by the Chernoff inequality. In particular, fix $\varepsilon > 0$, such that $\mathbb{H}(p+\varepsilon) > (1-\delta/4)\mathbb{H}(p)$, and since p is fixed $n\mathbb{H}(p) = \Omega(n)$, in particular, for n sufficiently large, we have $-\lg(n+1) \geq -\frac{\delta}{10}n\mathbb{H}(p)$. Also, for n sufficiently large, we have $2\exp\left(-\frac{n\varepsilon^2}{4p}\right) \leq \frac{\delta}{10}$. Putting it together, we have that for n large enough, we have

$$\mathbb{E}[B] \geq \left(1 - \frac{\delta}{4} - \frac{\delta}{10}\right)n\mathbb{H}(p)\left(1 - \frac{\delta}{10}\right) \geq (1-\delta)n\mathbb{H}(p),$$

as claimed.

As for the upper bound, observe that if an input sequence x has probability q, then the output sequence $y = \texttt{Ext}(x)$ has probability to be generated which is at least q. Now, all sequences of length $|y|$ have equal probability to be generated. Thus, we have the following (trivial) inequality $2^{|\texttt{Ext}(x)|}q \leq 2^{|\texttt{Ext}(x)|}\,\mathbb{P}[y = \texttt{Ext}(X)] \leq 1$, implying that $|\texttt{Ext}(x)| \leq \lg(1/q)$. Thus,

$$\mathbb{E}[B] = \sum_x \mathbb{P}[X=x]\,|\texttt{Ext}(x)| \leq \sum_x \mathbb{P}[X=x]\lg\frac{1}{\mathbb{P}[X=x]} = \mathbb{H}(X). \qquad \blacksquare$$

41.3. Bibliographical Notes

The presentation here follows [MU05, Sec. 9.1-Sec 9.3].

References

[MU05] M. Mitzenmacher and U. Upfal. *Probability and Computing – randomized algorithms and probabilistic analysis*. Cambridge, 2005.

Chapter 42

Entropy II

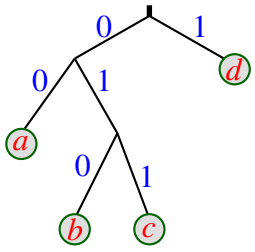
598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> The memory of my father is wrapped up in white paper, like sandwiches taken for a day at work. Just as a magician takes towers and rabbits out of his hat, he drew love from his small body, and the rivers of his hands overflowed with good deeds.
>
> Yehuda Amichai, My Father

42.1. Huffman coding

A ***binary code*** assigns a string of 0s and 1s to each character in the alphabet. A code assigns for each symbol in the input a codeword over some other alphabet. Such a coding is necessary, for example, for transmitting messages over a wire, were you can send only 0 or 1 on the wire (i.e., for example, consider the good old telegraph and Morse code). The receiver gets a binary stream of bits and needs to decode the message sent. A prefix code, is a code where one can decipher the message, a character by character, by reading a prefix of the input binary string, matching it to a code word (i.e., string), and continuing to decipher the rest of the stream. Such a code is a ***prefix code***.

A binary code (or a prefix code) is ***prefix-free*** if no code is a prefix of any other. ASCII and Unicode's UTF-8 are both prefix-free binary codes. Morse code is a binary code (and also a prefix code), but it is not prefix-free; for example, the code for `S` ($\cdots$) includes the code for `E` ($\cdot$) as a prefix. (Hopefully the receiver knows that when it gets $\cdots$ that it is extremely unlikely that this should be interpreted as `EEE`, but rather `S`.



Any prefix-free binary code can be visualized as a binary tree with the encoded characters stored at the leaves. The code word for any symbol is given by the path from the root to the corresponding leaf; 0 for left, 1 for right. The length of a codeword for a symbol is the depth of the corresponding leaf. Such trees are usually referred to as ***prefix trees*** or ***code trees***.

The beauty of prefix trees (and thus of prefix odes) is that decoding is easy. As a concrete example, consider the tree on the right. Given a string '010100', we can traverse down the tree from the root, going left if get a '0' and right if we get '1'. Whenever we get to a leaf, we output the character output in the leaf, and we jump back to the root for the next character we are about to read. For the example '010100', after reading '010' our traversal in the tree leads us to the leaf marked with 'b', we jump back to the root and read the next input digit, which is '1', and this leads us to the leaf marked with 'd', which we output, and jump back to the root. Finally, '00' leads us to the leaf marked by 'a', which the algorithm output. Thus, the binary string '010100' encodes the string **"bda"**.

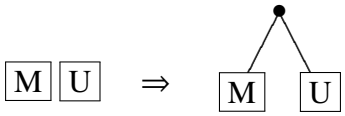
Suppose we want to encode messages in an n-character alphabet so that the encoded message is as short

as possible. Specifically, given an array frequency counts $f[1 \ldots n]$, we want to compute a prefix-free binary code that minimizes the total encoded length of the message. That is we would like to compute a tree T that minimizes

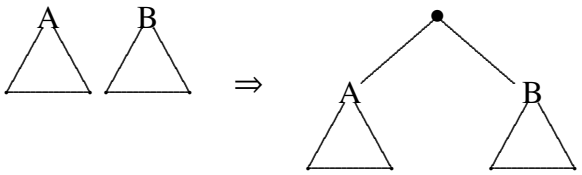
$$\text{cost}(T) = \sum_{i=1}^{n} f[i] * \text{len}(\text{code}(i)), \tag{42.1}$$

where code(i) is the binary string encoding the ith character and len(s) is the length (in bits) of the binary string s.

A nice property of this problem is that given two trees for some parts of the alphabet, we can easily put them together into a larger tree by just creating a new node and hanging the trees from this common node. For example, putting two characters together, we have the following.



Similarly, we can put together two subtrees.



42.1.1. The algorithm to build Hoffman's code

This suggests a simple algorithm that takes the two least frequent characters in the current frequency table, merge them into a tree, and put the merged tree back into the table (instead of the two old trees). The algorithm stops when there is a single tree. The intuition is that infrequent characters would participate in a large number of merges, and as such would be low in the tree – they would be assigned a long code word.

This algorithm is due to David Huffman, who developed it in 1952. Shockingly, this code is the best one can do. Namely, the resulting code is *asymptotically* gives the best possible compression of the data (of course, one can do better compression in practice using additional properties of the data and careful hacking). This ***Huffman coding*** is used widely and is the basic building block used by numerous other compression algorithms.

42.1.2. Analysis

Lemma 42.1.1. *Let T be an optimal code tree. Then T is a full binary tree (i.e., every node of T has either 0 or 2 children). In particular, if the height of T is d, then there are leafs nodes of height d that are sibling.*

Proof: If there is an internal node in T that has one child, we can remove this node from T, by connecting its only child directly with its parent. The resulting code tree is clearly a better compressor, in the sense of Eq. (42.1).

As for the second claim, consider a leaf u with maximum depth d in T, and consider it parent $v = \overline{\text{p}}(u)$. The node v has two children, and they are both leafs (otherwise u would not be the deepest node in the tree), as claimed. ■

Lemma 42.1.2. *Let x and y be the two least frequent characters (breaking ties between equally frequent characters arbitrarily). There is an optimal code tree in which x and y are siblings.*

Proof: More precisely, there is an optimal code in which x and y are siblings and have the largest depth of any leaf. Indeed, let T be an optimal code tree with depth d. The tree T has at least two leaves at depth d that are siblings, by Lemma 42.1.1.

Now, suppose those two leaves are not x and y, but some other characters α and β. Let $\mathcal{U}$ be the code tree obtained by swapping x and α. The depth of x increases by some amount Δ, and the depth of α decreases by the same amount. Thus,

$$\operatorname{cost}(\mathcal{U}) = \operatorname{cost}(T) - (f[\alpha] - f[x])\Delta.$$

By assumption, x is one of the two least frequent characters, but α is not, which implies that $f[\alpha] > f[x]$. Thus, swapping x and α does not increase the total cost of the code. Since T was an optimal code tree, swapping x and α does not decrease the cost, either. Thus, $\mathcal{U}$ is also an optimal code tree (and incidentally, $f[\alpha]$ actually equals $f[x]$). Similarly, swapping y and b must give yet another optimal code tree. In this final optimal code tree, x and y as maximum-depth siblings, as required. ■

Theorem 42.1.3. *Huffman codes are optimal prefix-free binary codes.*

Proof: If the message has only one or two different characters, the theorem is trivial. Otherwise, let $f[1 \ldots n]$ be the original input frequencies, where without loss of generality, $f[1]$ and $f[2]$ are the two smallest. To keep things simple, let $f[n+1] = f[1] + f[2]$. By the previous lemma, we know that some optimal code for $f[1..n]$ has characters 1 and 2 as siblings. Let $\mathcal{T}_{\text{opt}}$ be this optimal tree, and consider the tree formed by it by removing 1 and 2 as it leaves. We remain with a tree $\mathcal{T}'_{\text{opt}}$ that has as leafs the characters $3, \ldots, n$ and a "special" character $n+1$ (which is the parent of 1 and 2 in $\mathcal{T}_{\text{opt}}$) that has frequency $f[n+1]$. Now, since $f[n+1] = f[1] + f[2]$, we have

$$\begin{aligned}
\operatorname{cost}\left(\mathcal{T}_{\text{opt}}\right) &= \sum_{i=1}^{n} f[i]\operatorname{depth}_{\mathcal{T}_{\text{opt}}}(i) \\
&= \sum_{i=3}^{n+1} f[i]\operatorname{depth}_{\mathcal{T}_{\text{opt}}}(i) + f[1]\operatorname{depth}_{\mathcal{T}_{\text{opt}}}(1) + f[2]\operatorname{depth}_{\mathcal{T}_{\text{opt}}}(2) - f[n+1]\operatorname{depth}_{\mathcal{T}_{\text{opt}}}(n+1) \\
&= \operatorname{cost}\left(\mathcal{T}'_{\text{opt}}\right) + (f[1] + f[2])\operatorname{depth}\left(\mathcal{T}_{\text{opt}}\right) - (f[1] + f[2])\left(\operatorname{depth}\left(\mathcal{T}_{\text{opt}}\right) - 1\right) \\
&= \operatorname{cost}\left(\mathcal{T}'_{\text{opt}}\right) + f[1] + f[2]. \qquad (42.2)
\end{aligned}$$

This implies that minimizing the cost of $\mathcal{T}_{\text{opt}}$ is equivalent to minimizing the cost of $\mathcal{T}'_{\text{opt}}$. In particular, $\mathcal{T}'_{\text{opt}}$ must be an optimal coding tree for $f[3 \ldots n+1]$. Now, consider the Huffman tree $\mathcal{U}_H$ constructed for $f[3, \ldots, n+1]$ and the overall Huffman tree T_H constructed for $f[1, \ldots, n]$. By the way the construction algorithm works, we have that $\mathcal{U}_H$ is formed by removing the leafs of 1 and 2 from T. Now, by induction, we know that the Huffman tree generated for $f[3, \ldots, n+1]$ is optimal; namely, $\operatorname{cost}\left(\mathcal{T}'_{\text{opt}}\right) = \operatorname{cost}(\mathcal{U}_H)$. As such, arguing as above, we have

$$\operatorname{cost}(T_H) = \operatorname{cost}(\mathcal{U}_H) + f[1] + f[2] = \operatorname{cost}\left(\mathcal{T}'_{\text{opt}}\right) + f[1] + f[2] = \operatorname{cost}\left(\mathcal{T}_{\text{opt}}\right),$$

by Eq. (42.2). Namely, the Huffman tree has the same cost as the optimal tree. ■

42.1.3. A formula for the average size of a code word

Assume that our input is made out of n characters, where the ith character is p_i fraction of the input (one can think about p_i as the probability of seeing the ith character, if we were to pick a random character from the input).

Now, we can use these probabilities instead of frequencies to build a Huffman tree. The natural question is what is the length of the codewords assigned to characters as a function of their probabilities?

In general this question does not have a trivial answer, but there is a simple elegant answer, if all the probabilities are power of 2.

Lemma 42.1.4. *Let $1, \ldots, n$ be n symbols, such that the probability for the ith symbol is p_i, and furthermore, there is an integer $l_i \geq 0$, such that $p_i = 1/2^{l_i}$. Then, in the Huffman coding for this input, the code for i is of length l_i.*

Proof: The proof is by easy induction of the Huffman algorithm. Indeed, for $n = 2$ the claim trivially holds since there are only two characters with probability $1/2$. Otherwise, let i and j be the two characters with lowest probability. It must hold that $p_i = p_j$ (otherwise, $\sum_k p_k$ can not be equal to one). As such, Huffman's merges this two letters, into a single "character" that have probability $2p_i$, which would have encoding of length $l_i - 1$, by induction (on the remaining $n - 1$ symbols). Now, the resulting tree encodes i and j by code words of length $(l_i - 1) + 1 = l_i$, as claimed. ■

In particular, we have that $l_i = \lg 1/p_i$. This implies that the average length of a code word is

$$\sum_i p_i \lg \frac{1}{p_i}.$$

If we consider X to be a random variable that takes a value i with probability p_i, then this formula is

$$\mathbb{H}(X) = \sum_i \mathbb{P}[X = i] \lg \frac{1}{\mathbb{P}[X = i]},$$

which is the ***entropy*** of X.

Theorem 42.1.5. *Consider an input sequence S of m characters, where the characters are taken from an alphabet set Σ of size n. In particular, let f_i be the number of times the ith character of Σ appears in S, for $i = 1, \ldots, n$. Consider the compression of this string using Huffman's code. Then, the total length of the compressed string (ignoring the space needed to store the code itself) is $\leq m\,(\mathbb{H}(X) + 1)$, where X is a random variable that returns i with probability $p_i = f_i/m$.*

Proof: The trick is to replace p_i, which might not be a power of 2, by $q_i = 2^{\lfloor \lg p_i \rfloor}$. We have that $q_i \leq p_i \leq 2q_i$, and q_i is a power of 2, for all i. The leftover of this coding is $\Delta = 1 - \sum_i q_i$. We write Δ as a sum of powers of 2 (since the frequencies are fractions of the form i/m [since the input string is of length m] – this requires at most $\tau = O(\log m)$ numbers): $\Delta = \sum_{j=n+1}^{n+\tau} q_j$. We now create a Huffman code T for the frequencies $q_1, \ldots, q_n, q_{n+1}, \ldots, q_{n+\tau}$. The output length to encode the input string using this code, by Lemma 42.1.4, is

$$L = m \sum_{i=1}^{n} p_i \lg \frac{1}{q_i} \leq m \sum_{i=1}^{n} p_i \left(1 + \lg \frac{1}{p_i}\right) \leq m + m \sum_{i=1}^{n} p_i \lg \frac{1}{p_i} = m + m\mathbb{H}(X).$$

One can now restrict T to be a prefix tree only for the first n symbols. Indeed, delete the τ "fake" leafs/symbols, and repeatedly remove internal nodes that have only a single child. In the end of this process, we get a valid prefix tree for the first n symbols, and encoding the input string using this tree would require at most L bits, since process only shortened the code words. Finally, let $\mathcal{V}$ be the resulting tree.

Now, consider the Huffman tree code for the n input symbols using the original frequencies $p_1, \ldots p_n$. The resulting tree $\mathcal{U}$ is a better encoder for the input string than $\mathcal{V}$, by Theorem 42.1.3. As such, the compressed string, would have at most L bits – thus establishing the claim. ■

42.2. Compression

In this section, we consider the problem of how to compress a binary string. We map each binary string, into a new string (which is hopefully shorter). In general, by using a simple counting argument, one can show that no such mapping can achieve real compression (when the inputs are adversarial). However, the hope is that there is an underling distribution on the inputs, such that some strings are considerably more common than others.

Definition 42.2.1. A compression function `Compress` takes as input a sequence of n coin flips, given as an element of $\{H, T\}^n$, and outputs a sequence of bits such that each input sequence of n flips yields a distinct output sequence.

The following is easy to verify.

Lemma 42.2.2. *If a sequence S_1 is more likely than S_2 then the compression function that minimizes the expected number of bits in the output assigns a bit sequence to S_2 which is at least as long as S_1.*

Note, that this is weak. Usually, we would like the function to output a prefix code, like the Huffman code.

Theorem 42.2.3. *Consider a coin that comes up heads with probability $p > 1/2$. For any constant $\delta > 0$, when n is sufficiently large, the following holds.*

(i) There exists a compression function `Compress` *such that the expected number of bits output by* `Compress` *on an input sequence of n independent coin flips (each flip gets heads with probability p) is at most $(1+\delta)n\mathbb{H}(p)$; and*

(ii) The expected number of bits output by any compression function on an input sequence of n independent coin flips is at least $(1-\delta)n\mathbb{H}(p)$.

Proof: Let $\varepsilon > 0$ be a constant such that $p - \varepsilon > 1/2$. The first bit output by the compression procedure is '1' if the output string is just a copy of the input (using $n+1$ bits overall in the output), and '0' if it is compressed. We compress only if the number of ones in the input sequence, denoted by X is larger than $(p-\varepsilon)n$. By the Chernoff inequality, we know that $\mathbb{P}[X < (p-\varepsilon)n] \leq \exp\left(-n\varepsilon^2/2p\right)$.

If there are more than $(p-\varepsilon)n$ ones in the input, and since $p - \varepsilon > 1/2$, we have that

$$\sum_{j=\lceil n(p-\varepsilon)\rceil}^{n} \binom{n}{j} \leq \sum_{j=\lceil n(p-\varepsilon)\rceil}^{n} \binom{n}{\lceil n(p-\varepsilon)\rceil} \leq \frac{n}{2} 2^{n\mathbb{H}(p-\varepsilon)},$$

by Corollary 41.1.5. As such, we can assign each such input sequence a number in the range $0 \ldots \frac{n}{2}2^{n\mathbb{H}(p-\varepsilon)}$, and this requires (with the flag bit) $1 + \lfloor \lg n + n\mathbb{H}(p-\varepsilon) \rfloor$ random bits.

Thus, the expected number of bits output is bounded by

$$(n+1)\exp\left(-n\varepsilon^2/2p\right) + (1 + \lfloor \lg n + n\mathbb{H}(p-\varepsilon)\rfloor) \leq (1+\delta)n\mathbb{H}(p),$$

by carefully setting ε and n being sufficiently large. Establishing the upper bound.

As for the lower bound, observe that at least one of the sequences having exactly $\tau = \lfloor (p+\varepsilon)n \rfloor$ heads, must be compressed into a sequence having

$$\lg \binom{n}{\lfloor (p+\varepsilon)n \rfloor} - 1 \geq \lg \frac{2^{n\mathbb{H}(p+\varepsilon)}}{n+1} - 1 = n\mathbb{H}(p-\varepsilon) - \lg(n+1) - 1 = \mu,$$

by Corollary 41.1.5. Now, any input string with less than τ heads has lower probability to be generated. Indeed, for a specific strings with $\alpha < \tau$ ones the probability to generate them is $p^\alpha(1-p)^{n-\alpha}$ and $p^\tau(1-p)^{n-\tau}$, respectively. Now, observe that

$$p^\alpha(1-p)^{n-\alpha} = p^\tau(1-p)^{n-\tau} \cdot \frac{(1-p)^{\tau-\alpha}}{p^{\tau-\alpha}} = p^\tau(1-p)^{n-\tau}\left(\frac{1-p}{p}\right)^{\tau-\alpha} < p^\tau(1-p)^{n-\tau},$$

as $1-p < 1/2 < p$ implies that $(1-p)/p < 1$.

As such, Lemma 42.2.2 implies that all the input strings with less than τ ones, must be compressed into strings of length at least μ, by an optimal compresser. Now, the Chenroff inequality implies that $\mathbb{P}[X \le \tau] \ge 1 - \exp\left(-n\varepsilon^2/12p\right)$. Implying that an optimal compresser outputs on average at least $\left(1-\exp\left(-n\varepsilon^2/12p\right)\right)\mu$. Again, by carefully choosing ε and n sufficiently large, we have that the average output length of an optimal compressor is at least $(1-\delta)n\mathbb{H}(p)$. ■

42.3. Bibliographical Notes

The presentation here follows [MU05, Sec. 9.1-Sec 9.3].

References

[MU05] M. Mitzenmacher and U. Upfal. *Probability and Computing – randomized algorithms and probabilistic analysis*. Cambridge, 2005.

Chapter 43

Entropy III - Shannon's Theorem

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

The memory of my father is wrapped up in
white paper, like sandwiches taken for a day at work.

beginequation*-0.2cm] Just as a magician takes towers
and rabbits
out of his hat, he drew love from his small body,

beginequation*-0.2cm] and the rivers of his hands
overflowed with good deeds.

– Yehuda Amichai, My Father.,

43.1. Coding: Shannon's Theorem

We are interested in the problem sending messages over a noisy channel. We will assume that the channel noise is "nicely" behaved.

Definition 43.1.1. The input to a ***binary symmetric channel*** with parameter p is a sequence of bits $x_1, x_2, \ldots,$ and the output is a sequence of bits $y_1, y_2, \ldots,$ such that $\mathbb{P}[x_i = y_i] = 1 - p$ independently for each i.

Translation: Every bit transmitted have the same probability to be flipped by the channel. The question is how much information can we send on the channel with this level of noise. Naturally, a channel would have some capacity constraints (say, at most 4,000 bits per second can be sent on the channel), and the question is how to send the largest amount of information, so that the receiver can recover the original information sent.

Now, its important to realize that noise handling is unavoidable in the real world. Furthermore, there are tradeoffs between channel capacity and noise levels (i.e., we might be able to send considerably more bits on the channel but the probability of flipping (i.e., p) might be much larger). In designing a communication protocol over this channel, we need to figure out where is the optimal choice as far as the amount of information sent.

Definition 43.1.2. A (k, n) ***encoding function*** $\texttt{Enc} : \{0,1\}^k \to \{0,1\}^n$ takes as input a sequence of k bits and outputs a sequence of n bits. A (k, n) ***decoding function*** $\texttt{Dec} : \{0,1\}^n \to \{0,1\}^k$ takes as input a sequence of n bits and outputs a sequence of k bits.

Thus, the sender would use the encoding function to send its message, and the decoder would use the received string (with the noise in it), to recover the sent message. Thus, the sender starts with a message with k bits, it blow it up to n bits, using the encoding function, to get some robustness to noise, it send it over the (noisy) channel to the receiver. The receiver, takes the given (noisy) message with n bits, and use the decoding function to recover the original k bits of the message.

Naturally, we would like k to be as large as possible (for a fixed n), so that we can send as much information as possible on the channel. Naturally, there might be some failure probability; that is, the receiver might be unable to recover the original string, or recover an incorrect string.

The following celebrated result of Shannon[①] in 1948 states exactly how much information can be sent on such a channel.

Theorem 43.1.3 (Shannon's theorem). *For a binary symmetric channel with parameter $p < 1/2$ and for any constants $\delta, \gamma > 0$, where n is sufficiently large, the following holds:*

(i) For an $k \le n(1 - \mathbb{H}(p) - \delta)$ there exists (k, n) encoding and decoding functions such that the probability the receiver fails to obtain the correct message is at most γ for every possible k-bit input messages.

(ii) There are no (k, n) encoding and decoding functions with $k \ge n(1 - \mathbb{H}(p) + \delta)$ such that the probability of decoding correctly is at least γ for a k-bit input message chosen uniformly at random.

43.2. Proof of Shannon's theorem

The proof is not hard, but requires some care, and we will break it into parts.

43.2.1. How to encode and decode efficiently

43.2.1.1. The scheme

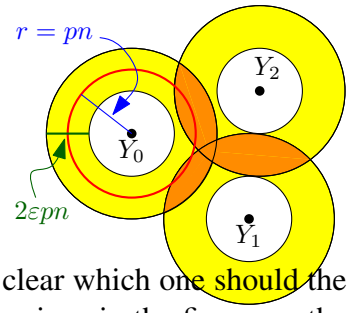
Our scheme would be simple. Pick $k \le n(1 - \mathbb{H}(p) - \delta)$. For any number $i = 0, \ldots, \widehat{K} = 2^{k+1} - 1$, randomly generate a binary string Y_i made out of n bits, each one chosen independently and uniformly. Let $Y_0, \ldots, Y_{\widehat{K}}$ denote these codewords.

For each of these codewords we will compute the probability that if we send this codeword, the receiver would fail. Let $X_0, \ldots, X_K$, where $K = 2^k - 1$, be the K codewords with the lowest probability of failure. We assign these words to the 2^k messages we need to encode in an arbitrary fashion. Specifically, for $i = 0, \ldots, 2^k - 1$, we encode i as the string X_i.

The decoding of a message w is done by going over all the codewords, and finding all the codewords that are in (Hamming) distance in the range $[p(1-\varepsilon)n, p(1+\varepsilon)n]$ from w. If there is only a single word X_i with this property, we return i as the decoded word. Otherwise, if there are no such word or there is more than one word then the decoder stops and report an error.

43.2.1.2. The proof

Intuition. Each code Y_i corresponds to a region that looks like a ring. The "ring" for Y_i is all the strings in Hamming distance between $(1 - \varepsilon)r$ and $(1 + \varepsilon)r$ from Y_i, where $r = pn$. Clearly, if we transmit a string Y_i, and the receiver gets a string inside the ring of Y_i, it is natural to try to recover the received string to the original code corresponding to Y_i. Naturally, there are two possible bad events here:



(A) The received string is outside the ring of Y_i.

(B) The received string is contained in several rings of different Ys, and it is not clear which one should the receiver decode the string to. These bad regions are depicted as the darker regions in the figure on the right.

[①] Claude Elwood Shannon (April 30, 1916 - February 24, 2001), an American electrical engineer and mathematician, has been called "the father of information theory".

Let $S_i = \mathcal{S}(Y_i)$ be all the binary strings (of length n) such that if the receiver gets this word, it would decipher it to be the original string assigned to Y_i (here are still using the extended set of codewords $Y_0, \ldots, Y_{\widehat{K}}$). Note, that if we remove some codewords from consideration, the set $\mathcal{S}(Y_i)$ just increases in size (i.e., the bad region in the ring of Y_i that is covered multiple times shrinks). Let W_i be the probability that Y_i was sent, but it was not deciphered correctly. Formally, let r denote the received word. We have that

$$W_i = \sum_{r \notin S_i} \mathbb{P}[r \text{ was received when } Y_i \text{ was sent}]. \tag{43.1}$$

To bound this quantity, let $\Delta(x, y)$ denote the Hamming distance between the binary strings x and y. Clearly, if x was sent the probability that y was received is

$$w(x, y) = p^{\Delta(x,y)}(1-p)^{n-\Delta(x,y)}.$$

As such, we have

$$\mathbb{P}[r \text{ received when } Y_i \text{ was sent}] = w(Y_i, r).$$

Definition 43.2.1. Let $\overline{S_{i,r}}$ be an indicator variable which is 1 if $r \notin S_i$. It is one if the receiver gets r, and does not decode it to Y_i (either because of failure, or because r is too close/far from Y_i).

We have that failure probability when sending r is

$$W_i = \sum_{r \notin S_i} \mathbb{P}[r \text{ received when } Y_i \text{ was sent}] = \sum_{r \notin S_i} w(Y_i, r) = \sum_r \overline{S_{i,r}} w(Y_i, r). \tag{43.2}$$

The value of W_i is a random variable over the choice of $Y_0, \ldots, Y_{\widehat{K}}$. As such, its natural to ask what is the expected value of W_i.

Consider the ring

$$\operatorname{ring}(r) = \left\{ x \in \{0,1\}^n \;\middle|\; (1-\varepsilon)np \le \Delta(x, r) \le (1+\varepsilon)np \right\},$$

where $\varepsilon > 0$ is a small enough constant. Observe that $x \in \operatorname{ring}(y)$ if and only if $y \in \operatorname{ring}(x)$. Suppose, that the code word Y_i was sent, and r was received. The decoder returns the original code associated with Y_i, if Y_i is the only codeword that falls inside $\operatorname{ring}(r)$.

Lemma 43.2.2. *Given that Y_i was sent, and r was received and furthermore $r \in \operatorname{ring}(Y_i)$, then the probability of the decoder failing, is*

$$\tau = \mathbb{P}[r \notin S_i \;\middle|\; r \in \operatorname{ring}(Y_i)] \le \frac{\gamma}{8},$$

where γ is the parameter of Theorem 43.1.3.

Proof: The decoder fails here, only if $\operatorname{ring}(r)$ contains some other codeword Y_j ($j \neq i$) in it. As such,

$$\tau = \mathbb{P}\Big[r \notin S_i \;\Big|\; r \in \operatorname{ring}(Y_i)\Big] \le \mathbb{P}\Big[Y_j \in \operatorname{ring}(r), \text{ for any } j \neq i\Big] \le \sum_{j \neq i} \mathbb{P}\Big[Y_j \in \operatorname{ring}(r)\Big].$$

Now, we remind the reader that the Y_js are generated by picking each bit randomly and independently, with probability $1/2$. As such, we have

$$\mathbb{P}\Big[Y_j \in \operatorname{ring}(r)\Big] = \frac{\big|\operatorname{ring}(r)\big|}{|\{0,1\}^n|} = \sum_{m=(1-\varepsilon)np}^{(1+\varepsilon)np} \frac{\binom{n}{m}}{2^n} \le \frac{n}{2^n}\binom{n}{\lfloor (1+\varepsilon)np \rfloor},$$

since $(1+\varepsilon)p < 1/2$ (for ε sufficiently small), and as such the last binomial coefficient in this summation is the largest. By Corollary 41.1.5 (i), we have

$$\mathbb{P}\Big[Y_j \in \mathrm{ring}(r)\Big] \leq \frac{n}{2^n}\binom{n}{\lfloor (1+\varepsilon)np \rfloor} \leq \frac{n}{2^n} 2^{n\mathbb{H}((1+\varepsilon)p)} = n2^{n\left(\mathbb{H}((1+\varepsilon)p)-1\right)}.$$

As such, we have

$$\begin{aligned}\tau = \mathbb{P}\Big[r \notin S_i \;\Big|\; r \in \mathrm{ring}(Y_i)\Big] &\leq \sum_{j\neq i} \mathbb{P}\Big[Y_j \in \mathrm{ring}(r)\Big] \leq \widehat{K}\,\mathbb{P}[Y_1 \in \mathrm{ring}(r)] \leq 2^{k+1} n 2^{n\left(\mathbb{H}((1+\varepsilon)p)-1\right)} \\ &\leq n2^{n\left(1-\mathbb{H}(p)-\delta\right)+1+n\left(\mathbb{H}((1+\varepsilon)p)-1\right)} \leq n2^{n\left(\mathbb{H}((1+\varepsilon)p)-\mathbb{H}(p)-\delta\right)+1}\end{aligned}$$

since $k \leq n(1-\mathbb{H}(p)-\delta)$. Now, we choose ε to be a small enough constant, so that the quantity $\mathbb{H}((1+\varepsilon)p) - \mathbb{H}(p) - \delta$ is equal to some (absolute) negative (constant), say $-\beta$, where $\beta > 0$. Then, $\tau \leq n2^{-\beta n+1}$, and choosing n large enough, we can make τ smaller than $\gamma/8$, as desired. As such, we just proved that

$$\tau = \mathbb{P}\Big[r \notin S_i \;\Big|\; r \in \mathrm{ring}(Y_i)\Big] \leq \frac{\gamma}{8}.$$ ■

Lemma 43.2.3. *Consider the situation where Y_i is sent, and the received string is r. We have that*

$$\mathbb{P}[r \notin \mathrm{ring}(Y_i)] = \sum_{r \notin \mathrm{ring}(Y_i)} w(Y_i, r) \leq \frac{\gamma}{8},$$

where γ is the parameter of Theorem 43.1.3.

Proof: This quantity, is the probability of sending Y_i when every bit is flipped with probability p, and receiving a string r such that more than $pn + \varepsilon pn$ bits where flipped (or less than $pn - \varepsilon pn$). But this quantity can be bounded using the Chernoff inequality. Indeed, let $Z = \Delta(Y_i, r)$, and observe that $\mathbb{E}[Z] = pn$, and it is the sum of n independent indicator variables. As such

$$\sum_{r \notin \mathrm{ring}(Y_i)} w(Y_i, r) = \mathbb{P}[|Z - \mathbb{E}[Z]| > \varepsilon pn] \leq 2\exp\left(-\frac{\varepsilon^2}{4} pn\right) < \frac{\gamma}{4},$$

since ε is a constant, and for n sufficiently large. ■

We remind the reader that $\overline{S_{i,r}}$ is an indicator variable that is one if receiving r (when sending Y_i) is "bad", see Definition 43.2.1. Importantly, this indicator variable also depends on all the other codewords – as they might cause some regions in the ring of Y_i to be covered multiple times.

Lemma 43.2.4. *We have that $f(Y_i) = \sum_{r \notin \mathrm{ring}(Y_i)} \mathbb{E}[\overline{S_{i,r}} w(Y_i, r)] \leq \gamma/8$ (the expectation is over all the choices of the Ys excluding Y_i).*

Proof: Observe that $\overline{S_{i,r}} w(Y_i, r) \leq w(Y_i, r)$ and for fixed Y_i and r we have that $\mathbb{E}[w(Y_i, r)] = w(Y_i, r)$. As such, we have that

$$f(Y_i) = \sum_{r \notin \mathrm{ring}(Y_i)} \mathbb{E}\Big[\overline{S_{i,r}} w(Y_i, r)\Big] \leq \sum_{r \notin \mathrm{ring}(Y_i)} \mathbb{E}[w(Y_i, r)] = \sum_{r \notin \mathrm{ring}(Y_i)} w(Y_i, r) \leq \frac{\gamma}{8},$$

by Lemma 43.2.3. ■

Lemma 43.2.5. *We have that* $g(Y_i) = \sum_{r \in \text{ring}(Y_i)} \mathbb{E}\left[\overline{S_{i,r}} w(Y_i, r)\right] \le \gamma/8$ *(the expectation is over all the choices of the Ys excluding* Y_i*).*

Proof: We have that $\overline{S_{i,r}} w(Y_i, r) \le \overline{S_{i,r}}$, as $0 \le w(Y_i, r) \le 1$. As such, we have that

$$\begin{aligned} g(Y_i) &= \sum_{r \in \text{ring}(Y_i)} \mathbb{E}\left[\overline{S_{i,r}} w(Y_i, r)\right] \le \sum_{r \in \text{ring}(Y_i)} \mathbb{E}\left[\overline{S_{i,r}}\right] = \sum_{r \in \text{ring}(Y_i)} \mathbb{P}[r \notin S_i] \\ &= \sum_r \mathbb{P}[r \notin S_i \cap (r \in \text{ring}(Y_i))] \\ &= \sum_r \mathbb{P}\left[r \notin S_i \,\middle|\, r \in \text{ring}(Y_i)\right] \mathbb{P}[r \in \text{ring}(Y_i)] \\ &\le \sum_r \frac{\gamma}{8} \mathbb{P}[r \in \text{ring}(Y_i)] \le \frac{\gamma}{8}, \end{aligned}$$

by Lemma 43.2.2. ■

Lemma 43.2.6. *For any i, we have* $\mu = \mathbb{E}[W_i] \le \gamma/4$*, where* γ *is the parameter of Theorem 43.1.3, where* W_i *is the probability of failure to recover* Y_i *if it was sent, see Eq. (43.1).*

Proof: We have by Eq. (43.2) that $W_i = \sum_r \overline{S_{i,r}} w(Y_i, r)$. For a fixed value of Y_i, we have by linearity of expectation, that

$$\begin{aligned} \mathbb{E}[W_i \mid Y_i] &= \mathbb{E}\left[\sum_r \overline{S_{i,r}} w(Y_i, r) \,\middle|\, Y_i\right] = \sum_r \mathbb{E}\left[\overline{S_{i,r}} w(Y_i, r) \,\middle|\, Y_i\right] \\ &= \sum_{r \in \text{ring}(Y_i)} \mathbb{E}\left[\overline{S_{i,r}} w(Y_i, r) \,\middle|\, Y_i\right] + \sum_{r \notin \text{ring}(Y_i)} \mathbb{E}\left[\overline{S_{i,r}} w(Y_i, r) \,\middle|\, Y_i\right] = g(Y_i) + f(Y_i) \le \frac{\gamma}{8} + \frac{\gamma}{8} = \frac{\gamma}{4}, \end{aligned}$$

by Lemma 43.2.4 and Lemma 43.2.5. Now $\mathbb{E}[W_i] = \mathbb{E}[\mathbb{E}[W_i \mid Y_i]] \le \mathbb{E}[\gamma/4] \le \gamma/4$. ■

In the following, we need the following trivial (but surprisingly deep) observation.

Observation 43.2.7. *For a random variable X, if* $\mathbb{E}[X] \le \psi$*, then there exists an event in the probability space, that assigns X a value* $\le \psi$*.*

Lemma 43.2.8. *For the codewords* $X_0, \ldots, X_K$*, the probability of failure in recovering them when sending them over the noisy channel is at most* γ*.*

Proof: We just proved that when using $Y_0, \ldots, Y_{\widehat{K}}$, the expected probability of failure when sending Y_i, is $\mathbb{E}[W_i] \le \gamma/4$, where $\widehat{K} = 2^{k+1} - 1$. As such, the expected total probability of failure is

$$\mathbb{E}\left[\sum_{i=0}^{\widehat{K}} W_i\right] = \sum_{i=0}^{\widehat{K}} \mathbb{E}[W_i] \le \frac{\gamma}{4} 2^{k+1} \le \gamma 2^k,$$

by Lemma 43.2.6. As such, by Observation 43.2.7, there exist a choice of Y_is, such that

$$\sum_{i=0}^{\widehat{K}} W_i \le 2^k \gamma.$$

Now, we use a similar argument used in proving Markov's inequality. Indeed, the W_i are always positive, and it can not be that 2^k of them have value larger than γ, because in the summation, we will get that

$$\sum_{i=0}^{\widehat{K}} W_i > 2^k \gamma.$$

Which is a contradiction. As such, there are 2^k codewords with failure probability smaller than γ. We set the 2^k codewords $X_0, \ldots, X_K$ to be these words, where $K = 2^k - 1$. Since we picked only a subset of the codewords for our code, the probability of failure for each codeword shrinks, and is at most γ. ■

Lemma 43.2.8 concludes the proof of the constructive part of Shannon's theorem.

43.2.2. Lower bound on the message size

We omit the proof of this part. It follows similar argumentation showing that for every ring associated with a codewords it must be that most of it is covered only by this ring (otherwise, there is no hope for recovery). Then an easy packing argument implies the claim.

43.3. Bibliographical Notes

The presentation here follows [MU05, Sec. 9.1-Sec 9.3].

References

[MU05] M. Mitzenmacher and U. Upfal. *Probability and Computing – randomized algorithms and probabilistic analysis*. Cambridge, 2005.

Chapter 44

Approximate Max Cut

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

We had encountered in the previous lecture examples of using rounding techniques for approximating discrete optimization problems. So far, we had seen such techniques when the relaxed optimization problem is a linear program. Interestingly, it is currently known how to solve optimization problems that are considerably more general than linear programs. Specifically, one can solve ***convex programming***. Here the feasible region is convex. How to solve such an optimization problems is outside the scope of this course. It is however natural to ask what can be done if one assumes that one can solve such general continuous optimization problems exactly.

In the following, we show that (optimization problem) max cut can be relaxed into a weird continuous optimization problem. Furthermore, this semi-definite program can be solved exactly efficiently. Maybe more surprisingly, we can round this continuous solution and get an improved approximation.

44.1. Problem Statement

Given an undirected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ and nonnegative weights ω_{ij}, for all $ij \in \mathsf{E}$, the ***maximum cut problem*** (MAX CUT) is that of finding the set of vertices S that maximizes the weight of the edges in the cut $(S, \overline{S})$; that is, the weight of the edges with one endpoint in S and the other in $\overline{S}$. For simplicity, we usually set $\omega_{ij} = O$ for $ij \notin \mathsf{E}$ and denote the weight of a cut $\left(S, \overline{S}\right)$ by $w\left(S, \overline{S}\right) = \sum_{i \in S, j \in \overline{S}} \omega_{ij}$.

This problem is NP-Complete, and hard to approximate within a certain constant.

Given a graph with vertex set $\mathsf{V} = \{1, \ldots, n\}$ and nonnegative weights ω_{ij}, the weight of the maximum cut $w(S, \overline{S})$ is given by the following integer quadratic program:

$$\begin{aligned} \text{(Q)} \qquad & \max \quad \frac{1}{2} \sum_{i<j} \omega_{ij}(1 - y_i y_j) \\ & \text{subject to:} \quad y_i \in \{-1, 1\} \qquad \forall i \in \mathsf{V}. \end{aligned}$$

Indeed, set $S = \left\{ i \;\middle|\; y_i = 1 \right\}$. Clearly, $\omega(S, \overline{S}) = \frac{1}{2} \sum_{i<j} \omega_{ij}(1 - y_i y_j)$.

Solving quadratic integer programming is of course NP-Hard. Thus, we will relax it, by thinking about the numbers y_i as unit vectors in higher dimensional space. If so, the multiplication of the two vectors, is now replaced by dot product. We have:

$$\text{(P)} \qquad \max \quad \gamma = \frac{1}{2} \sum_{i<j} \omega_{ij} \left(1 - \left\langle v_i, v_j \right\rangle\right)$$

$$\text{subject to:} \quad v_i \in \mathbb{S}^{(n)} \qquad \forall i \in V,$$

where $\mathbb{S}^{(n)}$ is the n dimensional unit sphere in $\mathbb{R}^{n+1}$. This is an instance of semi-definite programming, which is a special case of convex programming, which can be solved in polynomial time (solved here means approximated within a factor of $(1+\varepsilon)$ of optimal, for any arbitrarily small $\varepsilon > 0$, in polynomial time). Namely, the solver finds a feasible solution with a the target function being arbitrarily close to the optimal solution. Observe that (P) is a relaxation of (Q), and as such the optimal solution of (P) has value larger than the optimal value of (Q).

The intuition is that vectors that correspond to vertices that should be on one side of the cut, and vertices on the other sides, would have vectors which are faraway from each other in (P). Thus, we compute the optimal solution for (P), and we uniformly generate a random vector $\mathbf{r}$ on the unit sphere $\mathbb{S}^{(n)}$. This induces a hyperplane h which passes through the origin and is orthogonal to $\mathbf{r}$. We next assign all the vectors that are on one side of h to S, and the rest to $\overline{S}$.

Summarizing, the algorithm is as follows: First, we solve (P), next, we pick a random vector $\mathbf{r}$ uniformly on the unit sphere $\mathbb{S}^{(n)}$. Finally, we set

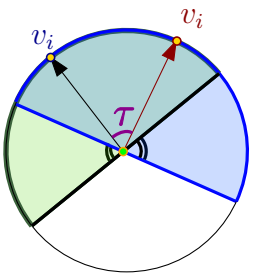
$$S = \{v_i \,|\, \langle v_i, \mathbf{r}\rangle \geq 0\}.$$

44.1.1. Analysis

The intuition of the above rounding procedure, is that with good probability, vectors in the solution of (P) that have large angle between them would be separated by this cut.

Lemma 44.1.1. *We have* $\mathbb{P}[\operatorname{sign}(\langle v_i, \mathbf{r}\rangle) \neq \operatorname{sign}(\langle v_j, \mathbf{r}\rangle)] = \dfrac{1}{\pi}\arccos(\langle v_i, v_j\rangle)$.

Proof: Let us think about the vectors v_i, v_j and $\mathbf{r}$ as being in the plane.



To see why this is a reasonable assumption, consider the plane g spanned by v_i and v_j, and observe that for the random events we consider, only the direction of $\mathbf{r}$ matter, which can be decided by projecting $\mathbf{r}$ on g, and normalizing it to have length 1. Now, the sphere is symmetric, and as such, sampling $\mathbf{r}$ randomly from $\mathbb{S}^{(n)}$, projecting it down to g, and then normalizing it, is equivalent to just choosing uniformly a vector from the unit circle.

Now, $\operatorname{sign}(\langle v_i, \mathbf{r}\rangle) \neq \operatorname{sign}(\langle v_j, \mathbf{r}\rangle)$ happens only if $\mathbf{r}$ falls in the double wedge formed by the lines perpendicular to v_i and v_j. The angle of this double wedge is exactly the angle between v_i and v_j. Now, since v_i and v_j are unit vectors, we have $\langle v_i, v_j\rangle = \cos(\tau)$, where $\tau = \angle v_i v_j$.

Thus,

$$\mathbb{P}\Big[\operatorname{sign}(\langle v_i, \mathbf{r}\rangle) \neq \operatorname{sign}(\langle v_j, \mathbf{r}\rangle)\Big] = \frac{2\tau}{2\pi} = \frac{1}{\pi}\arccos(\langle v_i, v_j\rangle),$$

as claimed. ■

Theorem 44.1.2. *Let W be the random variable which is the weight of the cut generated by the algorithm. We have*

$$\mathbb{E}[W] = \frac{1}{\pi}\sum_{i<j}\omega_{ij}\arccos(\langle v_i, v_j\rangle).$$

Proof: Let X_{ij} be an indicator variable which is 1 if and only if the edge ij is in the cut. We have

$$\mathbb{E}\Big[X_{ij}\Big] = \mathbb{P}\Big[\operatorname{sign}(\langle v_i, \mathbf{r}\,\rangle) \neq \operatorname{sign}(\langle v_j, \mathbf{r}\,\rangle)\Big] = \frac{1}{\pi}\arccos(\langle v_i, v_j\rangle),$$

by Lemma 44.1.1. Clearly, $W = \sum_{i<j} \omega_{ij} X_{ij}$, and by linearity of expectation, we have

$$\mathbb{E}[W] = \sum_{i<j} \omega_{ij} \mathbb{E}\Big[X_{ij}\Big] = \frac{1}{\pi} \sum_{i<j} \omega_{ij} \arccos(\langle v_i, v_j \rangle).$$

■

Lemma 44.1.3. *For $-1 \le y \le 1$, we have $\dfrac{\arccos(y)}{\pi} \ge \alpha \cdot \dfrac{1}{2}(1-y)$, where*

$$\alpha = \min_{0 \le \psi \le \pi} \frac{2}{\pi} \frac{\psi}{1-\cos(\psi)}. \tag{44.1}$$

Proof: Set $y = \cos(\psi)$. The inequality now becomes $\frac{\psi}{\pi} \ge \alpha \frac{1}{2}(1-\cos\psi)$. Reorganizing, the inequality becomes $\frac{2}{\pi} \frac{\psi}{1-\cos\psi} \ge \alpha$, which trivially holds by the definition of α. ■

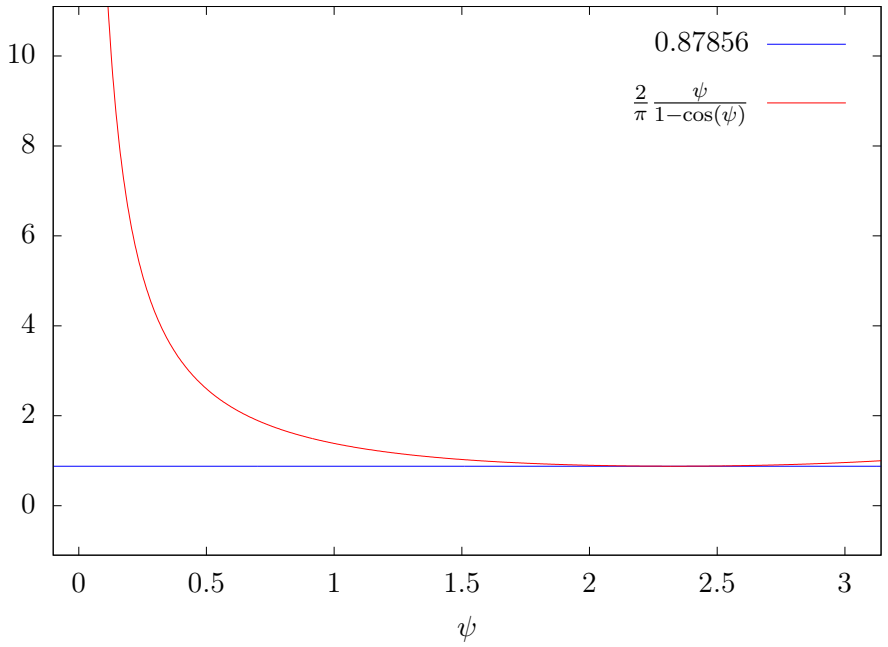


Figure 44.1: The function of Eq. (44.1).

Lemma 44.1.4. $\alpha > 0.87856$.

Proof: Using simple calculus, one can see that α achieves its value for $\psi = 2.331122...$, the nonzero root of $\cos\psi + \psi \sin\psi = 1$. ■

Theorem 44.1.5. *The above algorithm computes in expectation a cut with total weight $\alpha \cdot \text{Opt} \ge 0.87856\text{Opt}$, where* Opt *is the weight of the maximum weight cut.*

Proof: Consider the optimal solution to (P), and lets its value be $\gamma \ge \text{Opt}$. We have

$$\mathbb{E}[W] = \frac{1}{\pi} \sum_{i<j} \omega_{ij} \arccos(\langle v_i, v_j \rangle) \ge \sum_{i<j} \omega_{ij} \alpha \frac{1}{2}(1 - \langle v_i, v_j \rangle) = \alpha\gamma \ge \alpha \cdot \text{Opt},$$

by Lemma 44.1.3. ■

44.2. Semi-definite programming

Let us define a variable $x_{ij} = \langle v_i, v_j \rangle$, and consider the n by n matrix M formed by those variables, where $x_{ii} = 1$ for $i = 1, \ldots, n$. Let V be the matrix having $v_1, \ldots, v_n$ as its columns. Clearly, $M = V^T V$. In particular, this implies that for any non-zero vector $v \in \mathbb{R}^n$, we have $v^T M v = v^T A^T A v = (Av)^T (Av) \geq 0$. A matrix that has this property, is called ***positive semidefinite***. Interestingly, any positive semidefinite matrix P can be represented as a product of a matrix with its transpose; namely, $P = B^T B$. Furthermore, given such a matrix P of size $n \times n$, we can compute B such that $P = B^T B$ in $O(n^3)$ time. This is know as ***Cholesky decomposition***.

Observe, that if a semidefinite matrix $P = B^T B$ has a diagonal where all the entries are one, then B has columns which are unit vectors. Thus, if we solve (P) and get back a semi-definite matrix, then we can recover the vectors realizing the solution, and use them for the rounding.

In particular, (P) can now be restated as

$$\begin{aligned} (SD) \qquad \max \quad & \frac{1}{2} \sum_{i<j} \omega_{ij}(1 - x_{ij}) \\ \text{subject to:} \quad & x_{ii} = 1 && \text{for } i = 1, \ldots, n \\ & \left(x_{ij}\right)_{i=1,\ldots,n, j=1,\ldots,n} \text{ is a positive semi-definite matrix.} \end{aligned}$$

We are trying to find the optimal value of a linear function over a set which is the intersection of linear constraints and the set of positive semi-definite matrices.

Lemma 44.2.1. *Let $\mathcal{U}$ be the set of $n \times n$ positive semidefinite matrices. The set $\mathcal{U}$ is convex.*

Proof: Consider $A, B \in \mathcal{U}$, and observe that for any $t \in [0, 1]$, and vector $v \in \mathbb{R}^n$, we have:

$$v^T(tA + (1-t)B)v = v^T(tAv + (1-t)Bv) = tv^T Av + (1-t)v^T Bv \geq 0 + 0 \geq 0,$$

since A and B are positive semidefinite. ■

Positive semidefinite matrices corresponds to ellipsoids. Indeed, consider the set $x^T Ax = 1$: the set of vectors that solve this equation is an ellipsoid. Also, the eigenvalues of a positive semidefinite matrix are all non-negative real numbers. Thus, given a matrix, we can in polynomial time decide if it is positive semidefinite or not (by computing the eigenvalues of the matrix).

Thus, we are trying to optimize a linear function over a convex domain. There is by now machinery to approximately solve those problems to within any additive error in polynomial time. This is done by using the interior point method, or the ellipsoid method. See [BV04, GLS93] for more details. The key ingredient that is required to make these methods work, is the ability to decide in polynomial time, given a solution, whether its feasible or not. As demonstrated above, this can be done in polynomial time.

44.3. Bibliographical Notes

The approximation algorithm presented is from the work of Goemans and Williamson [GW95]. Håstad [Hås01b] showed that MAX CUT can not be approximated within a factor of $16/17 \approx 0.941176$. Recently, Khot *et al.* [KKMO04] showed a hardness result that matches the constant of Goemans and Williamson (i.e., one can not approximate it better than α, unless P = NP). However, this relies on two conjectures, the first one is the "Unique Games Conjecture", and the other one is "Majority is Stablest". The "Majority is Stablest"

conjecture was recently proved by Mossel *et al.* [MOO05]. However, it is not clear if the "Unique Games Conjecture" is true, see the discussion in [KKMO04].

The work of Goemans and Williamson was quite influential and spurred wide research on using SDP for approximation algorithms. For an extension of the MAX CUT problem where negative weights are allowed and relevant references, see the work by Alon and Naor [AN04].

References

[AN04] N. Alon and A. Naor. Approximating the cut-norm via grothendieck's inequality. *Proc. 36th Annu. ACM Sympos. Theory Comput.* (STOC), 72–80, 2004.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge, 2004.

[GLS93] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. 2nd. Vol. 2. Algorithms and Combinatorics. Berlin Heidelberg: Springer-Verlag, 1993.

[GW95] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42(6): 1115–1145, 1995.

[Hås01b] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4): 798–859, 2001.

[KKMO04] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for max cut and other 2-variable csps. *Proc. 45th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), To appear in SICOMP. 146–154, 2004.

[MOO05] E. Mossel, R. O'Donnell, and K. Oleszkiewicz. Noise stability of functions with low influences invariance and optimality. *Proc. 46th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), 21–30, 2005.

Chapter 45

Expanders I

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

"Mr. Matzerath has just seen fit to inform me that this partisan, unlike so many of them, was an authentic partisan. For - to quote the rest of my patient's lecture - there is no such thing as a part-time partisan. Real partisans are partisans always and as long as they live. They put fallen governments back in power and over throw governments that have just been put in power with the help of partisans. Mr. Matzerath contended - and this thesis struck me as perfectly plausible - that among all those who go in for politics your incorrigible partisan, who undermines what he has just set up, is closest to the artiest because he consistently rejects what he has just created."

Gunter Grass, The tin drum

45.1. Preliminaries on expanders

45.1.1. Definitions

Let $\mathsf{G} = (V, E)$ Be an undirected graph, where $V = \{1, \ldots, n\}$. A ***d*-regular graph** is a graph where all vertices have degree d. A d-regular graph $\mathsf{G} = (V, E)$ is a δ-edge expander (or just, ***δ*-expander**) if for every set $S \subseteq V$ of size at most $|V|/2$, there are at least $\delta d |S|$ edges connecting S and $\overline{S} = V \setminus S$; that is

$$e(S, \overline{S}) \geq \delta d |S|, \tag{45.1}$$

where

$$e(X, Y) = \left| \left\{ uv \;\middle|\; u \in X, v \in Y \right\} \right|.$$

A graph is **$[n, d, \delta]$-expander** if it is a n vertex, d-regular, δ-expander.

A **(n, d)-graph** G is a connected d-regular undirected (multi) graph. We will consider the set of vertices of such a graph to be the set $[\![n]\!] = \{1, \ldots, n\}$.

For a (multi) graph G with n nodes, its ***adjacency matrix*** is a $n \times n$ matrix M, where M_{ij} is the number of edges between i and j. It would be convenient to work the ***transition matrix*** Q associated with the random walk on G. If G is d-regular then $\mathsf{Q} = \mathsf{M}(\mathsf{G})/d$ and it is doubly stochastic.

A vector x is ***eigenvector*** of a matrix M with ***eigenvalue*** μ, if $x\mathsf{M} = \mu x$. In particular, by taking the dot product of both size by x, we get $\langle x\mathsf{M}, x\rangle = \langle \mu x, x\rangle$, which implies $\mu = \langle x\mathsf{M}, x\rangle / \langle x, x\rangle$. Since the adjacency matrix M of G is symmetric, all its eigenvalues are real numbers (this is a special case of the spectral theorem from linear algebra). Two eigenvectors with different eigenvectors are orthogonal to each other.

We denote the eigenvalues of M by $\widehat{\lambda_1} \geq \widehat{\lambda_2} \geq \cdots \widehat{\lambda_n}$, and the eigenvalues of Q by $\widehat{\lambda_1} \geq \widehat{\lambda_2} \geq \cdots \widehat{\lambda_n}$. Note, that for a d-regular graph, the eigenvalues of Q are the eigenvalues of M scaled down by a factor of $1/d$; that is $\widehat{\lambda_i} = \widehat{\lambda_i}/d$.

Lemma 45.1.1. *Let G be an undirected graph, and let Δ denote the maximum degree in G. Then, $\left|\widehat{\lambda_1}(\mathsf{G})\right| = \left|\widehat{\lambda_1}(\mathsf{M})\right| = \Delta$ if and only one connected component of G is Δ-regular. The multiplicity of Δ as an eigenvector is the number of Δ-regular connected components. Furthermore, we have $\left|\widehat{\lambda_i}(\mathsf{G})\right| \leq \Delta$, for all i.*

Proof: The ith entry of $\mathsf{M}\mathbf{1}_n$ is the degree of the ith vertex v_i of G (i.e., $\mathsf{M}\mathbf{1}_n = d(v_i)$, where $\mathbf{1}_n = (1, 1, \ldots, 1) \in \mathbb{R}^n$. So, let x be an eigenvector of M with eigenvalue λ, and let $x_j \neq 0$ be the coordinate with the largest (absolute value) among all coordinates of x corresponding to a connect component H of G. We have that

$$|\lambda| \left|x_j\right| = \left|(\mathsf{M}x)_j\right| = \left|\sum_{v_i \in N(v_j)} x_i\right| \leq \Delta \left|x_j\right|,$$

where $N(v_j)$ are the neighbors of v_i in G. Thus, all the eigenvalues of G have $\left|\widehat{\lambda_i}\right| \leq \Delta$, for $i = 1, \ldots, n$. If $\lambda = \Delta$, then this implies that $x_i = x_j$ if $v_i \in N(v_j)$, and $d(v_j) = \Delta$. Applying this argument to the vertices of $N(v_j)$, implies that H must be Δ-regular, and furthermore, $x_j = x_i$, if $x_i \in V(H)$. Clearly, the dimension of the subspace with eigenvalue (in absolute value) Δ is exactly the number of such connected components. ■

The following is also known. We do not provide a proof since we do not need it in our argumentation.

Lemma 45.1.2. *If G is bipartite, then if λ is eigenvalue of $\mathsf{M}(\mathsf{G})$ with multiplicity k, then $-\lambda$ is also its eigenvalue also with multiplicity k.*

45.2. Tension and expansion

Let $\mathsf{G} = (V, E)$, where $V = \{1, \ldots, n\}$ and G is a d regular graph.

Definition 45.2.1. For a graph G, let $\gamma(\mathsf{G})$ denote the ***tension*** of G; that is, the smallest constant, such that for any function $f : V(\mathsf{G}) \to \mathbb{R}$, we have that

$$\mathop{\mathbb{E}}_{x,y \in V}\left[|f(x) - f(y)|^2\right] \leq \gamma(\mathsf{G}) \mathop{\mathbb{E}}_{xy \in E}\left[|f(x) - f(y)|^2\right]. \tag{45.2}$$

Intuitively, the tension captures how close is estimating the variance of a function defined over the vertices of G, by just considering the edges of G. Note, that a disconnected graph would have infinite tension, and the clique has tension 1.

Surprisingly, tension is directly related to expansion as the following lemma testifies.

Lemma 45.2.2. *Let $\mathsf{G} = (V, E)$ be a given connected d-regular graph with n vertices. Then, G is a δ-expander, where $\delta \geq \dfrac{1}{2\gamma(\mathsf{G})}$ and $\gamma(\mathsf{G})$ is the tension of G.*

Proof: Consider a set $S \subseteq V$, where $|S| \leq n/2$. Let $f_S(v)$ be the function assigning 1 if $v \in S$, and zero otherwise. Observe that if $(u, v) \in \left(S \times \overline{S}\right) \cup \left(\overline{S} \times S\right)$ then $|f_S(u) - f_S(v)| = 1$, and $|f_S(u) - f_S(v)| = 0$ otherwise. As such, we have

$$\frac{2\,|S|\,(n - |S|)}{n^2} = \mathop{\mathbb{E}}_{x,y \in V}\left[|f_S(x) - f_S(y)|^2\right] \leq \gamma(\mathsf{G}) \mathop{\mathbb{E}}_{xy \in E}\left[|f_S(x) - f_S(y)|^2\right] = \gamma(\mathsf{G}) \frac{e\left(S, \overline{S}\right)}{|E|},$$

by Lemma 45.2.4. Now, since G is d-regular, we have that $|E| = nd/2$. Furthermore, $n - |S| \geq n/2$, which implies that

$$e\Big(S, \overline{S}\Big) \geq \frac{2\,|E| \cdot |S|\,(n - |S|)}{\gamma(\mathsf{G})n^2} = \frac{2(nd/2)(n/2)\,|S|}{\gamma(\mathsf{G})n^2} = \frac{1}{2\gamma(\mathsf{G})}\,d\,|S|\,.$$

which implies the claim (see Eq. (45.1)). ■

Now, a clique has tension 1, and it has the best expansion possible. As such, the smaller the tension of a graph, the better expander it is.

Definition 45.2.3. Given a random walk matrix Q associated with a d-regular graph, let $\mathcal{B}(\mathsf{Q}) = \langle v_1, \ldots, v_n \rangle$ denote the ***orthonormal eigenvector basis*** defined by Q. That is, $v_1, \ldots, v_n$ is an orthonormal basis for $\mathbb{R}^n$, where all these vectors are eigenvectors of Q and $v_1 = 1^n/\sqrt{n}$. Furthermore, let $\widehat{\lambda}_i$ denote the ith eigenvalue of Q, associated with the eigenvector v_i, such that $\widehat{\lambda}_1 \geq \widehat{\lambda}_2 \geq \cdots \geq \widehat{\lambda}_n$.

Lemma 45.2.4. *Let $\mathsf{G} = (V, E)$ be a given connected d-regular graph with n vertices. Then $\gamma(\mathsf{G}) = \frac{1}{1-\widehat{\lambda}_2}$, where $\widehat{\lambda}_2 = \lambda_2/d$ is the second largest eigenvalue of Q.*

Proof: Let $f : V \to \mathbb{R}$. Since in Eq. (45.2), we only look on the difference between two values of f, we can add a constant to f, and would not change the quantities involved in Eq. (45.2). As such, we assume that $\mathbb{E}[f(x)] = 0$. As such, we have that

$$\begin{aligned}
\mathop{\mathbb{E}}_{x,y\in V}\Big[|f(x) - f(y)|^2\Big] &= \mathop{\mathbb{E}}_{x,y\in V}\Big[(f(x) - f(y))^2\Big] = \mathop{\mathbb{E}}_{x,y\in V}\Big[(f(x))^2 - 2f(x)f(y) + (f(y))^2\Big] \\
&= \mathop{\mathbb{E}}_{x,y\in V}\Big[(f(x))^2\Big] - 2\mathop{\mathbb{E}}_{x,y\in V}[f(x)f(y)] + \mathop{\mathbb{E}}_{x,y\in V}\Big[(f(y))^2\Big] \\
&= \mathop{\mathbb{E}}_{x\in V}\Big[(f(x))^2\Big] - 2\mathop{\mathbb{E}}_{x\in V}[f(x)]\mathop{\mathbb{E}}_{y\in V}[f(y)] + \mathop{\mathbb{E}}_{y\in V}\Big[(f(y))^2\Big] = 2\mathop{\mathbb{E}}_{x\in V}\Big[(f(x))^2\Big].
\end{aligned} \tag{45.3}$$

Now, let $\mathcal{I}$ be the $n \times n$ identity matrix (i.e., one on its diagonal, and zero everywhere else). We have that

$$\begin{aligned}
\rho &= \frac{1}{d}\sum_{xy\in E}(f(x) - f(y))^2 = \frac{1}{d}\left(\sum_{x\in V} d(f(x))^2 - 2\sum_{xy\in E} f(x)f(y)\right) = \sum_{x\in V}(f(x))^2 - \frac{2}{d}\sum_{xy\in E} f(x)f(y) \\
&= \sum_{x,y\in V}(\mathcal{I} - \mathsf{Q})_{xy} f(x)f(y).
\end{aligned}$$

Note, that 1^n is an eigenvector of Q with eigenvalue 1, and this is the largest eigenvalue of Q. Let $\mathcal{B}(\mathsf{Q}) = \langle v_1, \ldots, v_n \rangle$ be the orthonormal eigenvector basis defined by Q, with eigenvalues $\widehat{\lambda}_1 \geq \widehat{\lambda}_2 \geq \cdots \geq \widehat{\lambda}_n$, respectively. Write $f = \sum_{i=1}^n \alpha_i v_i$, and observe that

$$0 = \mathbb{E}[f(x)] = \sum_{i=1}^n \frac{f(i)}{n} = \left\langle f, \frac{v_1}{\sqrt{n}} \right\rangle = \left\langle \sum_i \alpha_i v_i, \frac{v_1}{\sqrt{n}} \right\rangle = \frac{1}{\sqrt{n}}\langle \alpha_1 v_1, v_1 \rangle = \frac{\alpha_1}{\sqrt{n}},$$

since $v_i \perp v_1$ for $i \geq 2$. Hence $\alpha_1 = 0$, and we have

$$\begin{aligned}
\rho &= \sum_{x,y\in V}(\mathcal{I} - \mathsf{Q})_{xy} f(x)f(y) = \sum_{x,y\in V}(\mathcal{I} - \mathsf{Q})_{xy} \sum_{i=2}^n \alpha_{i=1}^n \alpha_i v_i(x) \sum_{j=1}^n \alpha_j v_j(y) \\
&= \sum_{i,j} \alpha_i \alpha_j \sum_{x\in V} v_i(x) \sum_{y\in V}(\mathcal{I} - \mathsf{Q})_{xy} v_j(y).
\end{aligned}$$

Now, we have that

$$\sum_{y\in V}(\mathcal{I}-\mathsf{Q})_{xy}v_j(y) = \left\langle \begin{bmatrix} x\text{th row of} \\ (\mathcal{I}-\mathsf{Q}) \end{bmatrix}, v_j \right\rangle = \Big((\mathcal{I}-\mathsf{Q})v_j\Big)(x) = \Big(\Big(1-\widehat{\lambda_j}\Big)v_j\Big)(x) = \Big(1-\widehat{\lambda_j}\Big)v_j(x),$$

since v_j is eigenvector of Q with eigenvalue $\widehat{\lambda_j}$. Since $v_1,\ldots,v_n$ is an orthonormal basis, and $f=\sum_{i=1}^n \alpha_i v_i$, we have that $\|f\|^2=\sum_j \alpha_j^2$. Going back to ρ, we have that

$$\begin{aligned}
\rho &= \sum_{i,j}\alpha_i\alpha_j \sum_{x\in V} v_i(x)\Big(1-\widehat{\lambda_j}\Big)v_j(x) = \sum_{i,j}\alpha_i\alpha_j\Big(1-\widehat{\lambda_j}\Big)\sum_{x\in V} v_i(x)v_j(x) \\
&= \sum_{i,j}\alpha_i\alpha_j\Big(1-\widehat{\lambda_j}\Big)\Big\langle v_i, v_j\Big\rangle = \sum_{j=1}^{n}\alpha_j^2\Big(1-\widehat{\lambda_j}\Big)\Big\langle v_j, v_j\Big\rangle \\
&\geq \Big(1-\widehat{\lambda_2}\Big)\sum_{j=2}^{n}\alpha_j^2\sum_{x\in V}\Big(v_j(x)\Big)^2 = \Big(1-\widehat{\lambda_2}\Big)\sum_{j=2}^{n}\alpha_j^2 = \Big(1-\widehat{\lambda_2}\Big)\|f\|^2 = \Big(1-\widehat{\lambda_2}\Big)\sum_{j=1}^{n}(f(x))^2 \qquad (45.4)\\
&= n\Big(1-\widehat{\lambda_2}\Big)\mathop{\mathbb{E}}_{x\in V}\Big[(f(x))^2\Big],
\end{aligned}$$

since $\alpha_1=0$ and $\widehat{\lambda_1}\geq\widehat{\lambda_2}\geq\cdots\geq\widehat{\lambda_n}$.

We are now ready for the kill. Indeed, by Eq. (45.3), and the above, we have that

$$\begin{aligned}
\mathop{\mathbb{E}}_{x,y\in V}\Big[|f(x)-f(y)|^2\Big] = 2\mathop{\mathbb{E}}_{x\in V}\Big[(f(x))^2\Big] &\leq \frac{2}{n\Big(1-\widehat{\lambda_2}\Big)}\rho = \frac{2}{dn\Big(1-\widehat{\lambda_2}\Big)}\sum_{xy\in E}(f(x)-f(y))^2 \\
&= \frac{1}{1-\widehat{\lambda_2}}\cdot\frac{1}{|E|}\sum_{xy\in E}(f(x)-f(y))^2 = \frac{1}{1-\widehat{\lambda_2}}\mathop{\mathbb{E}}_{xy\in E}\Big[|f(x)-f(y)|^2\Big].
\end{aligned}$$

This implies that $\gamma(\mathsf{G})\leq\dfrac{1}{1-\widehat{\lambda_2}}$. Observe, that the inequality in our analysis, had risen from Eq. (45.4), but if we take $f=v_2$, then the inequality there holds with equality, which implies that $\gamma(\mathsf{G})\geq\frac{1}{1-\widehat{\lambda_2}}$, which implies the claim. ■

Lemma 45.2.2 together with the above lemma, implies that the expansion δ of a d-regular graph G is at least $\delta = 1/2\gamma(\mathsf{G}) = (1-\lambda_2/d)/2$, where λ_2 is the second eigenvalue of the adjacency matrix of G. Since the tension of a graph is direct function of its second eigenvalue, we could either argue about the tension of a graph or its second eigenvalue when bounding the graph expansion.

Chapter 46

Expanders II

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

Be that as it may, it is to night school that I owe what education I possess; I am the first to own that it doesn't amount to much, though there is something rather grandiose about the gaps in it.

Gunter Grass, The tin drum

46.1. Bi-tension

Our construction of good expanders, would use the idea of composing graphs together. To this end, in our analysis, we will need the notion of bi-tension. Let $\widetilde{\mathsf{E}}(\mathsf{G})$ be the set of *directed* edges of G; that is, every edge $xy \in E(\mathsf{G})$ appears twice as $(x \to y)$ and $(y \to x)$ in $\widetilde{\mathsf{E}}$.

Definition 46.1.1. For a graph G, let $\gamma_2(\mathsf{G})$ denote the ***bi-tension*** of G; that is, the smallest constant, such that for any two function $f, g : V(\mathsf{G}) \to \mathbb{R}$, we have that

$$\mathop{\mathbb{E}}_{x,y\in V}\Big[|f(x) - g(y)|^2\Big] \leq \gamma_2(\mathsf{G}) \mathop{\mathbb{E}}_{(x\to y)\in\widetilde{\mathsf{E}}}\Big[|f(x) - g(y)|^2\Big]. \tag{46.1}$$

The proof of the following lemma is similar to the proof of Lemma 45.2.4. The proof is provided for the sake of completeness, but there is little new in it.

Lemma 46.1.2. *Let $\mathsf{G} = (V, E)$ be a connected d-regular graph with n vertices. Then $\gamma_2(\mathsf{G}) = \dfrac{1}{1-\widehat{\lambda}}$, where $\widehat{\lambda} = \widehat{\lambda}(\mathsf{G})$, where $\widehat{\lambda}(\mathsf{G}) = \max\Big(\widehat{\lambda}_2, -\widehat{\lambda}_n\Big)$, where $\widehat{\lambda}_i$ is the ith largest eigenvalue of the random walk matrix associated with G.*

Proof: We can assume that $\mathbb{E}[f(x)] = 0$. As such, we have that

$$\mathop{\mathbb{E}}_{x,y\in V}\Big[|f(x) - g(y)|^2\Big] = \mathop{\mathbb{E}}_{x,y\in V}\Big[(f(x))^2\Big] - 2\mathop{\mathbb{E}}_{x,y\in V}[f(x)g(y)] + \mathop{\mathbb{E}}_{y\in V}\Big[(g(y))^2\Big] = \mathop{\mathbb{E}}_{x,y\in V}\Big[(f(x))^2\Big] + \mathop{\mathbb{E}}_{y\in V}\Big[(g(y))^2\Big]. \tag{46.2}$$

Let Q be the matrix associated with the random walk on G (each entry is either zero or $1/d$), we have

$$\begin{aligned}\rho = \mathop{\mathbb{E}}_{(x\to y)\in\widetilde{\mathsf{E}}}\Big[|f(x) - g(y)|^2\Big] &= \frac{1}{nd}\sum_{(x\to y)\in\widetilde{\mathsf{E}}}(f(x) - g(y))^2 = \frac{1}{n}\sum_{x,y\in V}\mathsf{Q}_{xy}(f(x) - g(y))^2 \\ &= \frac{1}{n}\sum_{x\in V}\Big((f(x))^2 + (g(x))^2\Big) - \frac{2}{n}\sum_{x,y\in V}\mathsf{Q}_{xy}f(x)g(y).\end{aligned}$$

Let $\mathcal{B}(\mathsf{Q}) = \langle v_1, \ldots, v_n \rangle$ be the orthonormal eigenvector basis defined by Q (see Definition 45.2.3), with eigenvalues $\widehat{\lambda_1} \geq \widehat{\lambda_2} \geq \cdots \geq \widehat{\lambda_n}$, respectively. Write $f = \sum_{i=1}^{n} \alpha_i v_i$ and $g = \sum_{i=1}^{n} \beta_i v_i$. Since $\mathbb{E}[f(x)] = 0$, we have that $\alpha_1 = 0$. Now, $\mathsf{Q}_{xy} = \mathsf{Q}_{yx}$, and we have

$$\begin{aligned}
\sum_{x,y \in V} \mathsf{Q}_{xy} f(x) g(y) &= \sum_{x,y \in V} \mathsf{Q}_{yx} \left(\sum_i \alpha_i v_i(x) \right) \left(\sum_j \beta_j v_j(y) \right) = \sum_{i,j} \alpha_i \beta_j \sum_{y \in V} v_j(y) \sum_{x \in V} \mathsf{Q}_{yx} v_i(x) \\
&= \sum_{i,j} \alpha_i \beta_j \sum_{y \in V} v_j(y) \left(\widehat{\lambda_i} v_i(y) \right) = \sum_{i,j} \alpha_i \beta_j \widehat{\lambda_i} \left\langle v_j, v_i \right\rangle = \sum_{i=2}^{n} \alpha_i \beta_i \widehat{\lambda_i} \sum_{y \in V} (v_i(y))^2 \\
&\leq \widehat{\lambda} \sum_{i=2}^{n} \frac{\alpha_i^2 + \beta_i^2}{2} \sum_{y \in V} (v_i(y))^2 \leq \frac{\widehat{\lambda}}{2} \sum_{i=1}^{n} \sum_{y \in V} \left((\alpha_i v_i(y))^2 + (\beta_i v_i(y))^2 \right) \\
&= \frac{\widehat{\lambda}}{2} \sum_{y \in V} \left((f(y))^2 + (g(y))^2 \right)
\end{aligned}$$

As such,

$$\begin{aligned}
\mathop{\mathbb{E}}_{(x \to y) \in \widetilde{\mathsf{E}}} \left[|f(x) - g(y)|^2 \right] &= \frac{1}{nd} \sum_{(x \to y) \in \widetilde{\mathsf{E}}} |f(x) - g(y)|^2 = \frac{1}{n} \sum_{y \in V} \left((f(y))^2 + (g(y))^2 \right) - \frac{1}{n} \sum_{x,y \in V} \frac{2 f(x) g(y)}{d} \\
&= \frac{1}{n} \sum_{y \in V} \left((f(y))^2 + (g(y))^2 \right) - \frac{2}{n} \sum_{x,y \in V} \mathsf{Q}_{xy} f(x) g(y) \\
&\geq \left(\frac{1}{n} - \frac{2}{n} \cdot \frac{\widehat{\lambda}}{2} \right) \sum_{y \in V} \left((f(y))^2 + (g(y))^2 \right) = \left(1 - \widehat{\lambda} \right) \left(\mathop{\mathbb{E}}_{y \in V} \left[(f(y))^2 \right] + \mathop{\mathbb{E}}_{y \in V} \left[(g(y))^2 \right] \right) \\
&= \left(1 - \widehat{\lambda} \right) \mathop{\mathbb{E}}_{x,y \in V} \left[|f(x) - g(y)|^2 \right],
\end{aligned}$$

by Eq. (46.2). This implies that $\gamma_2(\mathsf{G}) \leq 1/\left(1 - \widehat{\lambda}\right)$. Again, by trying either $f = g = v_2$ or $f = v_n$ and $g = -v_n$, we get that the inequality above holds with equality, which implies $\gamma_2(\mathsf{G}) \geq 1/\left(1 - \widehat{\lambda}\right)$. Together, the claim now follows. ■

46.2. Explicit construction

For a set $U \subseteq V$ of vertices, its ***characteristic vector***, denoted by $x = \chi_U$, is the n dimensional vector, where $x_i = 1$ if and only if $i \in U$.

The following is an easy consequence of Lemma 45.1.1.

Lemma 46.2.1. *For a d-regular graph G the vector $\mathbf{1}^n = (1, 1, \ldots, 1)$ is the only eigenvector with eigenvalue d (of the adjacency matrix $\mathsf{M}(\mathsf{G})$), if and only if G is connected. Furthermore, we have $|\lambda_i| \leq d$, for all i.*

Our main interest would be in the second largest eigenvalue of M. Formally, let

$$\lambda_2(\mathsf{G}) = \max_{x \perp \mathbf{1}^n, x \neq 0} \left| \frac{\langle x\mathsf{M}, x \rangle}{\langle x, x \rangle} \right|.$$

We state the following result but do not prove it since we do not need it for our nafarious purposes (however, we did prove the left side of the inequality).

Theorem 46.2.2. *Let* G *be a* δ*-expander with adjacency matrix* M *and let* $\lambda_2 = \lambda_2(\mathsf{G})$ *be the second-largest eigenvalue of* M*. Then*

$$\frac{1}{2}\left(1 - \frac{\lambda_2}{d}\right) \leq \delta \leq \sqrt{2\left(1 - \frac{\lambda_2}{d}\right)}.$$

What the above theorem says, is that the expansion of a $[n, d, \delta]$-expander is a function of how far is its second eigenvalue (i.e., λ_2) from its first eigenvalue (i.e., d). This is usually referred to as the ***spectral gap***.

We will start by explicitly constructing an expander that has "many" edges, and then we will show to reduce its degree till it become a constant degree expander.

46.2.1. Explicit construction of a small expander

46.2.1.1. A quicky reminder of fields

A ***field*** is a set $\mathbb{F}$ together with two operations, called addition and multiplication, and denoted by $+$ and $\cdot$, respectively, such that the following axioms hold:

(i) Closure: $\forall x, y \in \mathbb{F}$, we have $x + y \in \mathbb{F}$ and $x \cdot y \in \mathbb{F}$.

(ii) Associativity: $\forall x, y, z \in \mathbb{F}$, we have $x + (y + z) = (x + y) + z$ and $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.

(iii) Commutativity: $\forall x, y \in \mathbb{F}$, we have $x + y = y + x$ and $x \cdot y = y \cdot x$.

(iv) Identity: There exists two distinct special elements $0, 1 \in \mathbb{F}$. We have that $\forall x \in \mathbb{F}$ it holds $x + 0 = a$ and $x \cdot 1 = x$.

(v) Inverse: There exists two distinct special elements $0, 1 \in \mathbb{F}$, and we have that $\forall x \in \mathbb{F}$ there exists an element $-x \in \mathbb{F}$, such that $x + (-x) = 0$.

Similarly, $\forall x \in \mathbb{F}$, $x \neq 0$, there exists an element $y = x^{-1} = 1/x \in \mathbb{F}$ such that $x \cdot y = 1$.

(vi) Distributivity: $\forall x, y, z \in \mathbb{F}$ we have $x \cdot (y + z) = x \cdot y + x \cdot z$.

Let $q = 2^t$, and $r > 0$ be an integer. Consider the finite field $\mathbb{F}_q$. It is the field of polynomials of degree at most $t - 1$, where the coefficients are over $\mathbb{Z}_2$ (i.e., all calculations are done modulus 2). Formally, consider the polynomial

$$p(x) = x^t + x + 1.$$

It it irreducible over $\mathbb{F}_2 = \{0, 1\}$ (i.e., $p(0) = p(1) \neq 0$). We can now do polynomial arithmetic over polynomials (with coefficients from $\mathbb{F}_2$), where we do the calculations modulus $p(x)$. Note, that any irreducible polynomial of degree n yields the same field up to isomorphism. Intuitively, we are introducing the n distinct roots of $p(x)$ into $\mathbb{F}$ by creating an extension field of $\mathbb{F}$ with those roots.

An element of $\mathbb{F}_q = \mathbb{F}_{2^t}$ can be interpreted as a binary string $b = b_0 b_1 \ldots, b_{t-1}$ of length t, where the corresponding polynomial is

$$\mathrm{poly}(b) = \sum_{i=0}^{t-1} b_i x^i.$$

The nice property of $\mathbb{F}_q$ is that addition can be interpreted as a `xor` operation. That is, for any $x, y \in \mathbb{F}_q$, we have that $x + y + y = x$ and $x - y - y = x$. The key properties of $\mathbb{F}_q$ we need is that multiplications and addition can be computed in it in polynomial time in t, and it is a field (i.e., each non-zero element has a unique inverse).

46.2.1.1.1. Computing multiplication in $\mathbb{F}_q$. Consider two elements $\alpha, \beta \in \mathbb{F}_q$. Multiply the two polynomials $\mathrm{poly}(\alpha)$ by $\mathrm{poly}(\beta)$, let $\mathrm{poly}(\gamma)$ be the resulting polynomial (of degree at most $2t - 2$), and compute the remainder $\mathrm{poly}(\beta)$ when dividing it by the irreducible polynomial $p(x)$. For this remainder polynomial, normalize the coefficients by computing their modules base 2. The resulting polynomial is the product of α and β.

For more details on this field, see any standard text on abstract algebra.

46.2.1.2. The construction

Let $q = 2^t$, and $r > 0$ be an integer. Consider the linear space $\mathbb{G} = \mathbb{F}_q^r$. Here, a member $\alpha = (\alpha_0, \ldots, \alpha_r) \in \mathbb{G}$ can be thought of as being a string (of length $r+1$) over $\mathbb{F}_q$, or alternatively, as a binary string of length $n = t(r+1)$.

For $\alpha = (\alpha_0, \ldots, \alpha_r) \in \mathbb{G}$, and $x, y \in \mathbb{F}_q$, define the operator

$$\rho(\alpha, x, y) = \alpha + y \cdot \left(1, x, x^2, \ldots, x^r\right) = \left(\alpha_0 + y, \alpha_1 + yx, \alpha_2 + yx^2, \ldots, \alpha_r + yx^r\right) \in \mathbb{G}.$$

Since addition over $\mathbb{F}_q$ is equivalent to a xor operation we have that

$$\begin{aligned}\rho(\rho(\alpha, x, y), x, y) &= \left(\alpha_0 + y + y, \alpha_1 + yx + yx, \alpha_2 + yx^2 + yx^2, \ldots, \alpha_r + yx^r + yx^r\right) \\ &= (\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_r) = \alpha.\end{aligned}$$

Furthermore, if $(x, y) \neq (x', y')$ then $\rho(\alpha, x, y) \neq \rho(\alpha, x', y')$.

We now define a graph $\mathsf{LD}(q, r) = (\mathbb{G}, E)$, where

$$E = \left\{ \alpha\beta \;\middle|\; \begin{array}{l} \alpha \in \mathbb{G}, x, y \in \mathbb{F}_q \\ \beta = \rho(\alpha, x, y) \end{array} \right\}$$

Note, that this graph is well defined, as $\rho(\beta, x, y) = \alpha$. The degree of a vertex of $\mathsf{LD}(q, r)$ is $\left|\mathbb{F}_q\right|^2 = q^2$, and $\mathsf{LD}(q, r)$ has $N = |\mathbb{G}| = q^{r+1} = 2^{t(r+1)} = 2^n$ vertices.

Theorem 46.2.3. *For any $t > 0, r > 0$ and $q = 2^t$, where $r < q$, we have that $\mathsf{LD}(q, r)$ is a graph with q^{r+1} vertices. Furthermore, $\lambda_1(\mathsf{LD}(q, r)) = q^2$, and $\lambda_i(\mathsf{LD}(q, r)) \leq rq$, for $i = 2, \ldots, n$.*

In particular, if $r \leq q/2$, then $\mathsf{LD}(q, r)$ is a $\left[q^{r+1}, q^2, \frac{1}{4}\right]$-expander.

Proof: Let M be the $N \times N$ adjacency matrix of $\mathsf{LD}(q, r)$. Let $L : \mathbb{F}_q \to \{0, 1\}$ be a linear map which is onto. It is easy to verify that $\left|L^{-1}(0)\right| = \left|L^{-1}(1)\right|$[①]

We are interested in the eigenvalues of the matrix M. To this end, we consider vectors in $\mathbb{R}^N$. The ith row an ith column of M is associated with a unique element $b_i \in \mathbb{G}$. As such, for a vector $v \in \mathbb{R}^N$, we denote by $v[b_i]$ the ith coordinate of v. In particular, for $\alpha = (\alpha_0, \ldots, \alpha_r) \in \mathbb{G}$, let $v_\alpha \in \mathbb{R}^N$ denote the vector, where its $\beta = (\beta_0, \ldots, \beta_r) \in \mathbb{G}$ coordinate is

$$v_\alpha[\beta] = (-1)^{L\left(\sum_{i=0}^r \alpha_i \beta_i\right)}.$$

Let $V = \left\{ v_\alpha \;\middle|\; \alpha \in \mathbb{G} \right\}$. For $\alpha \neq \alpha' \in V$, observe that

$$\langle v_\alpha, v_{\alpha'} \rangle = \sum_{\beta \in \mathbb{G}} (-1)^{L\left(\sum_{i=0}^r \alpha_i \beta_i\right)} \cdot (-1)^{L\left(\sum_{i=0}^r \alpha'_i \beta_i\right)} = \sum_{\beta \in \mathbb{G}} (-1)^{L\left(\sum_{i=0}^r (\alpha_i + \alpha'_i) \beta_i\right)} = \sum_{\beta \in \mathbb{G}} v_{\alpha+\alpha'}[\beta].$$

So, consider $\psi = \alpha + \alpha' \neq 0$. Assume, for the simplicity of exposition that all the coordinates of ψ are non-zero. We have, by the linearity of L that

$$\langle v_\alpha, v_{\alpha'} \rangle = \sum_{\beta \in \mathbb{G}} (-1)^{L\left(\sum_{i=0}^r \alpha_i \beta_i\right)} = \sum_{\beta_0 \in \mathbb{F}_q, \ldots, \beta_{r-1} \in \mathbb{F}_q} (-1)^{L(\psi_0 \beta_0 + \cdots + \psi_{r-1} \beta_{r-1})} \sum_{\beta_r \in \mathbb{F}_q} (-1)^{L(\psi_r \beta_r)}.$$

[①] Indeed, if $Z = L^{-1}(0)$, and $L(x) = 1$, then $L(y) = 1$, for all $y \in U = \left\{ x + z \;\middle|\; z \in Z \right\}$. Now, its clear that $|Z| = |U|$.

However, since $\psi_r \neq 0$, the quantity $\left\{\psi_r \beta_r \;\middle|\; \beta_r \in \mathbb{F}_q\right\} = \mathbb{F}_q$. Thus, the summation $\sum_{\beta_r \in \mathbb{F}_q} (-1)^{L(\psi_r \beta_r)}$ gets $\left|L^{-1}(0)\right|$ terms that are 1, and $\left|L^{-1}(0)\right|$ terms that are -1. As such, this summation is zero, implying that $\langle v_\alpha, v_{\alpha'} \rangle = 0$. Namely, the vectors of V are orthogonal.

Observe, that for $\alpha, \beta, \psi \in \mathbb{G}$, we have $v_\alpha[\beta + \psi] = v_\alpha[\beta]\, v_\alpha[\psi]$. For $\alpha \in \mathbb{G}$, consider the vector $\mathsf{M} v_\alpha$. We have, for $\beta \in \mathbb{G}$, that

$$(\mathsf{M} v_\alpha)[\beta] = \sum_{\psi \in \mathbb{G}} \mathsf{M}_{\beta\psi} \cdot v_\alpha[\psi] = \sum_{\substack{x,y \,\in\, \mathbb{F}_q \\ \psi = \rho(\beta,x,y)}} v_\alpha[\psi] = \sum_{x,y \,\in\, \mathbb{F}_q} v_\alpha[\beta + y(1, x, \ldots, x^r)]$$
$$= \left(\sum_{x,y \,\in\, \mathbb{F}_q} v_\alpha[y(1, x, \ldots, x^r)] \right) \cdot v_\alpha[\beta]\,.$$

Thus, setting $\lambda(\alpha) = \sum_{x,y \,\in\, \mathbb{F}_q} v_\alpha[y(1, x, \ldots, x^r)] \in \mathbb{R}$, we have that $\mathsf{M} v_\alpha = \lambda(\alpha) \cdot v_\alpha$. Namely, v_α is an eigenvector, with eigenvalue $\lambda(\alpha)$.

Let $p_\alpha(x) = \sum_{i=0}^{r} \alpha_i x^i$, and let

$$\lambda(\alpha) = \sum_{x,y \,\in\, \mathbb{F}_q} v_\alpha[y(1, x, \ldots, x^r)] \in \mathbb{R} = \sum_{x,y \in \mathbb{F}_q} (-1)^{L(y p_\alpha(x))}$$
$$= \sum_{\substack{x,y \in \mathbb{F}_q \\ p_\alpha(x) = 0}} (-1)^{L(y\; p_\alpha(x))} + \sum_{\substack{x,y \in \mathbb{F}_q \\ p_\alpha(x) \neq 0}} (-1)^{L(y\; p_\alpha(x))}.$$

If $p_\alpha(x) = 0$ then $(-1)^{L(y\; p_\alpha(x))} = 1$, for all y. As such, each such x contributes q to $\lambda(\alpha)$.

If $p_\alpha(x) \neq 0$ then $y\; p_\alpha(x)$ takes all the values of $\mathbb{F}_q$, and as such, $L(y\; p_\alpha(x))$ is 0 for half of these values, and 1 for the other half. Implying that these kind terms contribute 0 to $\lambda(\alpha)$. But $p_\alpha(x)$ is a polynomial of degree r, and as such there could be at most r values of x for which the first term is taken. As such, if $\alpha \neq 0$ then $\lambda(\alpha) \leq rq$. If $\alpha = 0$ then $\lambda(\alpha) = q^2$, which implies the theorem. ■

This construction provides an expander with constant degree only if the number of vertices is a constant. Indeed, if we want an expander with constant degree, we have to take q to be as small as possible. We get the relation $n = q^{r+1} \leq q^q$, since $r \leq r$, which implies that $q = \Omega(\log n / \log\log n)$. Now, the expander of Theorem 46.2.3 is q^2-regular, which means that it is not going to provide us with a constant degree expander.

However, we are going to use it as our building block in a construction that would start with this expander and would inflate it up to the desired size.

Chapter 47

Expanders III - The Zig Zag Product

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> Gradually, but not as gradually as it seemed to some parts of his brain, he began to infuse his tones with a sarcastic wounding bitterness. Nobody outside a madhouse, he tried to imply, could take seriously a single phrase of this conjectural, nugatory, deluded, tedious rubbish. Within quite a short time he was contriving to sound like an unusually fanatical Nazi trooper in charge of a book-burning reading out to the crowd excerpts from a pamphlet written by a pacifist, Jewish, literate Communist. A growing mutter, half-amused, half-indignant, arose about him, but he closed his ears to it and read on. Almost unconsciously he began to adopt an unnameable foreign accent and to read faster and faster, his head spinning. As if in a dream he heard Welch stirring, then whispering, then talking at his side. he began punctuating his discourse with smothered snorts of derision. He read on, spitting out the syllables like curses, leaving mispronunciations, omissions, spoonerisms uncorrected, turning over the pages of his script like a score-reader following a presto movement, raising his voice higher and higher. At last he found his final paragraph confronting him, stopped, and look at his audience.
>
> Kingsley Amis, Lucky Jim

47.1. Building a large expander with constant degree

47.1.1. Notations

For a vertex $v \in V(\mathsf{G})$, we will denote by $v_{\mathsf{G}}[i] = v[i]$ the ith neighbor of v in the graph G (we order the neighbors of a vertex in an arbitrary order).

The regular graphs we next discuss have ***consistent labeling***. That is, for a regular graph G (we assume here that G is regular). This means that if u is the ith neighbor v then v is the ith neighbor of u. Formally, this means that $v[i][i] = v$, for all v and i. This is a non-trivial property, but its easy to verify that the low quality expander of Theorem 46.2.3 has this property. It is also easy to verify that the complete graph can be easily be made into having consistent labeling (exercise). These two graphs would be sufficient for our construction.

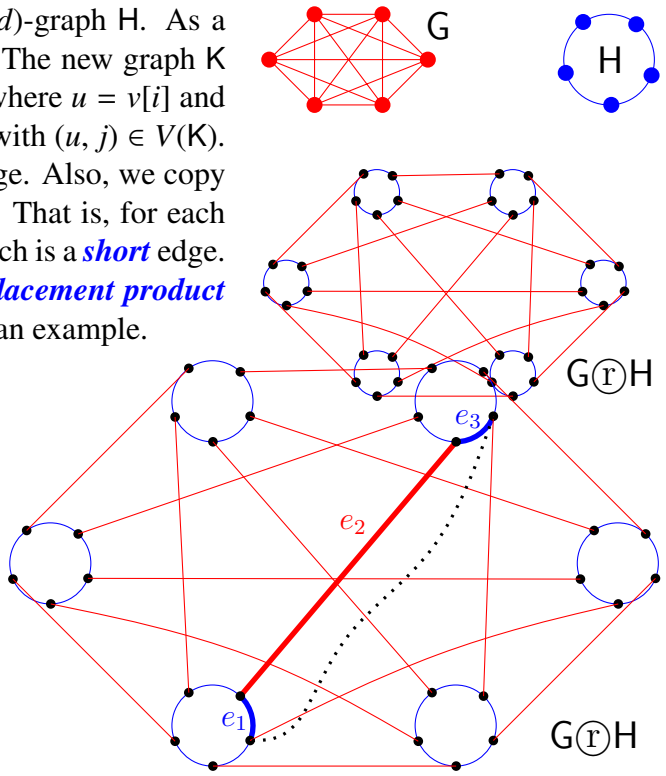
47.1.2. The Zig-Zag product

At this point, we know how to construct a good "small" expander. The question is how to build a large expander (i.e., large number of vertices) and with constant degree.

The intuition of the construction is the following: It is easy to improve the expansion qualities of a graph by squaring it. The problem is that the resulting graph G has a degree which is too large. To overcome this, we will replace every vertex in G by a copy of a small graph that is connected and has low degree. For example, we could replace every vertex of degree d in G by a path having d vertices. Every such vertex is now in charge of original edge of the graph. Naturally, such a replacement operation reduces the quality of the expansion of the

resulting graph. In this case, replacing a vertex with a path is a potential "disaster", since every such subpath increases the lengths of the paths of the original graph by a factor of d (and intuitively, a good expander have "short" paths between any pair of vertices).

Consider a "large" (n, D)-graph G and a "small" (D, d)-graph H. As a first stage, we replace every vertex of G by a copy of H. The new graph K has $[\![n]\!] \times [\![D]\!]$ as a vertex set. Here, the edge $vu \in V(\mathsf{G})$, where $u = v[i]$ and $v = u[j]$, is replaced by the edge connecting $(v, i) \in V(\mathsf{K})$ with $(u, j) \in V(\mathsf{K})$. We will refer to this resulting edge $(v, i)(u, j)$ as a ***long*** edge. Also, we copy all the edges of the small graph to each one of its copies. That is, for each $i \in [\![n]\!]$, and $uv \in E(\mathsf{H})$, we add the edge $(i, u)(i, v)$ to K, which is a ***short*** edge. We will refer to K, which is a $(nD, d + 1)$-graph, as a ***replacement product*** of G and H, denoted by $\mathsf{G} \textcircled{r} \mathsf{H}$. See figure on the right for an example.



Again, intuitively, we are losing because the expansion of the resulting graph had deteriorated too much. To overcome this problem, we will perform local shortcuts to shorten the paths in the resulting graph (and thus improve its expansion properties). A ***zig-zag-zig path*** in the replacement product graph K, is a three edge path $e_1e_2e_3$, where e_1 and e_3 are short edges, and the middle edge e_2 is a long edge. That is, if $e_1 = (i, u)(i, v)$, $e_2 = (i, v)(j, v')$, and $e_3 = (j, v')(j, u')$, then $e_1, e_2, e_3 \in E(\mathsf{K})$, $ij \in E(\mathsf{G})$, $uv \in E(\mathsf{H})$ and $v'u' \in E(\mathsf{H})$. Intuitively, you can think about e_1 as a small "zig" step in H, e_2 is a long "zag" step in G, and finally e_3 is a "zig" step in H.

Another way of representing a zig-zag-zig path $v_1v_2v_3v_4$ starting at the vertex $v_1 = (i, v) \in V(\mathsf{F})$, is to parameterize it by two integers $\ell, \ell' \in [\![d]\!]$, where

$$v_1 = (i, v), \quad v_2 = (i, v_{\mathsf{H}}[\ell]) \quad v_3 = (i_{\mathsf{G}}[v_{\mathsf{H}}[\ell]], v_{\mathsf{H}}[\ell]) \quad v_4 = (i_{\mathsf{G}}[v_{\mathsf{H}}[\ell]] \ , \ (v_{\mathsf{H}}[\ell])_{\mathsf{H}}[\ell']).$$

Let Z be the set of all (unordered) pairs of vertices of K connected by such a zig-zag-zig path. Note, that every vertex (i, v) of K has d^2 paths having (i, v) as an end point. Consider the graph $\mathsf{F} = (V(\mathsf{K}), Z)$. The graph F has nD vertices, and it is d^2 regular. Furthermore, since we shortcut all these zig-zag-zig paths in K, the graph F is a much better expander (intuitively) than K. We will refer to the graph F as the ***zig-zag*** product of G and H.

Definition 47.1.1. The ***zig-zag product*** of (n, D)-graph G and a (D, d)-graph H, is the (nD, d^2) graph $\mathsf{F} = \mathsf{G} \textcircled{z} \mathsf{H}$, where the set of vertices is $[\![n]\!] \times [\![D]\!]$ and for any $v \in [\![n]\!]$, $i \in [\![D]\!]$, and $\ell, \ell' \in [\![d]\!]$ we have in F the edge connecting the vertex (i, v) with the vertex $(i_{\mathsf{G}}[v_{\mathsf{H}}[\ell]] \ , \ (v_{\mathsf{H}}[\ell])_{\mathsf{H}}[\ell'])$.

Remark 47.1.2. We need the resulting zig-zag graph to have consistent labeling. For the sake of simplicity of exposition, we are just going to assume this property.

We next bound the tension of the zig-zag product graph.

Theorem 47.1.3. *We have* $\gamma(\mathsf{G} \textcircled{z} \mathsf{H}) \leq \gamma_2(\mathsf{G})(\gamma_2(\mathsf{H}))^2$. *and* $\gamma_2(\mathsf{G} \textcircled{z} \mathsf{H}) \leq \gamma_2(\mathsf{G})(\gamma_2(\mathsf{H}))^2$.

Proof: Let $\mathsf{G} = ([\![n]\!], E)$ be a (n, D)-graph and $\mathsf{H} = ([\![D]\!], E')$ be a (D, d)-graph. Fix any function $f : [\![n]\!] \times [\![D]\!] \to \mathbb{R}$, and observe that

$$\psi = \mathop{\mathbb{E}}_{\substack{u,v \in [\![n]\!] \\ k,\ell \in [\![D]\!]}} \left[|f(u, k) - f(v, \ell)|^2 \right] = \mathop{\mathbb{E}}_{k,\ell \in [\![D]\!]} \left[\mathop{\mathbb{E}}_{u,v \in [\![n]\!]} \left[|f(u, k) - f(v, \ell)|^2 \right] \right]$$

$$\leq \mathop{\mathbb{E}}_{k,\ell\in[\![D]\!]}\left[\gamma_2(\mathsf{G})\mathop{\mathbb{E}}_{uv\in E(\mathsf{G})}\left[\left|f(u,k)-f(v,\ell)\right|^2\right]\right] = \gamma_2(\mathsf{G})\underbrace{\mathop{\mathbb{E}}_{k,\ell\in[\![D]\!]}\left[\mathop{\mathbb{E}}_{\substack{u\in[\![n]\!]\\ p\in[\![D]\!]}}\left[\left|f(u,k)-f(u[p]\,,\ell)\right|^2\right]\right]}_{=\Delta_1}.$$

Now,

$$\Delta_1 = \mathop{\mathbb{E}}_{\substack{u\in[\![n]\!]\\ \ell\in[\![D]\!]}}\left[\mathop{\mathbb{E}}_{k,p\in[\![D]\!]}\left[\left|f(u,k)-f(u[p]\,,\ell)\right|^2\right]\right] \leq \mathop{\mathbb{E}}_{\substack{u\in[\![n]\!]\\ \ell\in[\![D]\!]}}\left[\gamma_2(\mathsf{H})\mathop{\mathbb{E}}_{kp\in E(\mathsf{H})}\left[\left|f(u,k)-f(u[p]\,,\ell)\right|^2\right]\right]$$

$$= \gamma_2(\mathsf{H})\underbrace{\mathop{\mathbb{E}}_{\substack{u\in[\![n]\!]\\ \ell\in[\![D]\!]}}\left[\mathop{\mathbb{E}}_{\substack{p\in[\![D]\!]\\ j\in[\![d]\!]}}\left[\left|f(u,p[j])-f(u[p]\,,\ell)\right|^2\right]\right]}_{=\Delta_2}.$$

Now,

$$\Delta_2 = \mathop{\mathbb{E}}_{\substack{j\in[\![d]\!]\\ \ell\in[\![D]\!]}}\left[\mathop{\mathbb{E}}_{\substack{u\in[\![n]\!]\\ p\in[\![D]\!]}}\left[\left|f(u,p[j])-f(u[p]\,,\ell)\right|^2\right]\right] = \mathop{\mathbb{E}}_{\substack{j\in[\![d]\!]\\ \ell\in[\![D]\!]}}\left[\mathop{\mathbb{E}}_{\substack{v\in[\![n]\!]\\ p\in[\![D]\!]}}\left[\left|f(v[p]\,,p[j])-f(v,\ell)\right|^2\right]\right]$$

$$= \mathop{\mathbb{E}}_{\substack{j\in[\![d]\!]\\ v\in[\![n]\!]}}\left[\mathop{\mathbb{E}}_{\substack{p\in[\![D]\!]\\ \ell\in[\![D]\!]}}\left[\left|f(v[p]\,,p[j])-f(v,\ell)\right|^2\right]\right]$$

$$= \gamma_2(\mathsf{H})\underbrace{\mathop{\mathbb{E}}_{\substack{j\in[\![d]\!]\\ v\in[\![n]\!]}}\left[\mathop{\mathbb{E}}_{p\ell\in E(\mathsf{H})}\left[\left|f(v[p]\,,p[j])-f(v,\ell)\right|^2\right]\right]}_{=\Delta_3}.$$

Now, we have

$$\Delta_3 = \mathop{\mathbb{E}}_{\substack{j\in[\![d]\!]\\ v\in[\![n]\!]}}\left[\mathop{\mathbb{E}}_{\substack{p\in[\![D]\!]\\ i\in[\![d]\!]}}\left[\left|f(v[p]\,,p[j])-f(v,p[i])\right|^2\right]\right] = \mathop{\mathbb{E}}_{(u,k)(\ell,v)\in E(\mathsf{G}\circledcirc\mathsf{H})}\left[\left|f(u,k)-f(\ell,v)\right|\right],$$

as $(v[p]\,,p[j])$ is adjacent to $(v[p]\,,p)$ (a short edge), which is in turn adjacent to (v,p) (a long edge), which is adjacent to $(v,p[i])$ (a short edge). Namely, $(v[p]\,,p[j])$ and $(v,p[i])$ form the endpoints of a zig-zag path in the replacement product of G and H. That is, these two endpoints are connected by an edge in the zig-zag product graph. Furthermore, it is easy to verify that each zig-zag edge get accounted for in this representation exactly once, implying the above inequality. Thus, we have $\psi \leq \gamma_2(\mathsf{G})(\gamma_2(\mathsf{H}))^2\Delta_3$, which implies the claim.

The second claim follows by similar argumentation. ■

47.1.3. Squaring

The last component in our construction, is ***squaring***squaring!graph a graph. Given a (n,d)-graph G, consider the multigraph G^2 formed by connecting any vertices connected in G by a path of length 2. Clearly, if M is the adjacency matrix of G, then the adjacency matrix of G^2 is the matrix M^2. Note, that $\left(\mathsf{M}^2\right)_{ij}$ is the number of distinct paths of length 2 in G from i to j. Note, that the new graph might have self loops, which does not effect our analysis, so we keep them in.

Lemma 47.1.4. *Let* G *be a* (n,d)*-graph. The graph* G^2 *is a* (n,d^2)*-graph. Furthermore* $\gamma_2\left(\mathsf{G}^2\right) = \frac{(\gamma_2(\mathsf{G}))^2}{2\gamma_2(\mathsf{G})-1}$.

Proof: The graph G^2 has eigenvalues $\left(\widehat{\lambda}_1(\mathsf{G})\right)^2, \ldots, \left(\widehat{\lambda}_1(\mathsf{G})\right)^2$ for its matrix Q^2. As such, we have that

$$\widehat{\lambda}\left(\mathsf{G}^2\right) = \max\left(\widehat{\lambda}_2\left(\mathsf{G}^2\right), -\widehat{\lambda}_n\left(\mathsf{G}^2\right)\right).$$

Now, $\widehat{\lambda}_1\left(\mathsf{G}^2\right) = 1$. Now, if $\widehat{\lambda}_2(\mathsf{G}) \geq \left|\widehat{\lambda}_n(\mathsf{G})\right| < 1$ then $\widehat{\lambda}\left(\mathsf{G}^2\right) = \widehat{\lambda}_2\left(\mathsf{G}^2\right) = \left(\widehat{\lambda}_2(\mathsf{G})\right)^2 = \left(\widehat{\lambda}(\mathsf{G})\right)^2$.

If $\widehat{\lambda}_2(\mathsf{G}) < \left|\widehat{\lambda}_n(\mathsf{G})\right|$ then $\widehat{\lambda}\left(\mathsf{G}^2\right) = \widehat{\lambda}_2\left(\mathsf{G}^2\right) = \left(\widehat{\lambda}_n(\mathsf{G})\right)^2 = \left(\widehat{\lambda}(\mathsf{G})\right)^2$..

Thus, in either case $\widehat{\lambda}\left(\mathsf{G}^2\right) = \left(\widehat{\lambda}(\mathsf{G})\right)^2$. Now, By Lemma 46.1.2 $\gamma_2(\mathsf{G}) = \frac{1}{1-\widehat{\lambda}(\mathsf{G})}$, which implies that $\widehat{\lambda}(\mathsf{G}) = 1 - 1/\gamma_2(\mathsf{G})$, and thus

$$\gamma_2\left(\mathsf{G}^2\right) = \frac{1}{1-\widehat{\lambda}(\mathsf{G}^2)} = \frac{1}{1-\left(\widehat{\lambda}(\mathsf{G})\right)^2} = \frac{1}{1-\left(1-\frac{1}{\gamma_2(\mathsf{G})}\right)^2} = \frac{\gamma_2(\mathsf{G})}{2-\frac{1}{\gamma_2(\mathsf{G})}} = \frac{(\gamma_2(\mathsf{G}))^2}{2\gamma_2(\mathsf{G})-1}.$$

■

47.1.4. The construction

So, let build an expander using Theorem 46.2.3, with parameters $r = 7$ $q = 2^4 = 32$. Let $d = q^2 = 256$. The resulting graph H has $N = q^{r+1} = d^4$ vertices, and it is $d = q^2$ regular. Furthermore, $\widehat{\lambda}_i \leq r/q = 7/32$, for all $i \geq 2$. As such, we have

$$\gamma(\mathsf{H}) = \gamma_2(\mathsf{H}) = \frac{1}{1-7/32} = \frac{32}{25}.$$

Let G_0 be any graph that its square is the complete graph over $n_0 = N+1$ vertices. Observe that G_0^2 is d^4-regular. Set $\mathsf{G}_i = \left(\mathsf{G}_{i-1}^2 \circledcirc \mathsf{H}\right)$, Clearly, the graph G_i has

$$n_i = n_{i-1}N$$

vertices. The graph $\mathsf{G}_{i-1}^2 \circledcirc \mathsf{H}$ is d^2 regular. As far as the bi-tension, let $\alpha_i = \gamma_2(\mathsf{G}_i)$. We have that

$$\alpha_i = \frac{\alpha_{i-1}^2}{2\alpha_{i-1}-1}(\gamma_2(\mathsf{H}))^2 = \frac{\alpha_{i-1}^2}{2\alpha_{i-1}-1}\left(\frac{32}{25}\right)^2 \leq 1.64\frac{\alpha_{i-1}^2}{2\alpha_{i-1}-1}.$$

It is now easy to verify, that α_i can not be bigger than 5.

Theorem 47.1.5. *For any* $i \geq 0$*, one can compute deterministically a graph* G_i *with* $n_i = (d^4+1)d^{4i}$ *vertices, which is* d^2 *regular, where* $d = 256$*. The graph* G_i *is a* $(1/10)$*-expander.*

Proof: The construction is described above. As for the expansion, since the bi-tension bounds the tension of a graph, we have that $\gamma(\mathsf{G}_i) \leq \gamma_2(\mathsf{G}_i) \leq 5$. Now, by Lemma 45.2.2, we have that G_i is a δ-expander, where $\delta \geq 1/(2\gamma(\mathsf{G}_i)) \geq 1/10$. ■

47.2. Bibliographical notes

A good survey on expanders is the monograph by Hoory *et al.* [HLW06]. The small expander construction is from the paper by Alon *et al.* [ASS08] (but its originally from the work by Along and Roichman [AR94]). The work by Alon *et al.* [ASS08] contains a construction of an expander that is constant degree, which is of similar complexity to the one we presented here. Instead, we used the zig-zag expander construction from the influential work of Reingold *et al.* [RVW02]. Our analysis however, is from an upcoming paper by Mendel and Naor [MN08]. This analysis is arguably reasonably simple (as simplicity is in the eye of the beholder, we will avoid claim that its the simplest), and (even better) provide a good intuition and a systematic approach to analyzing the expansion.

We took a creative freedom in naming notations, and the name tension and bi-tension are the author's own invention.

47.3. Exercises

Exercise 47.3.1 (Expanders made easy.). By considering a random bipartite three-regular graph on $2n$ vertices obtained by picking three random permutations between the two sides of the bipartite graph, prove that there is a $c > 0$ such that for every n there exits a $(2n, 3, c)$-expander. (What is the value of c in your construction?)

Exercise 47.3.2 (Is your consistency in vain?). In the construction, we assumed that the graphs we are dealing with when building expanders have consistent labeling. This can be enforced by working with bipartite graphs, which implies modifying the construction slightly.

(A) Prove that a d-regular bipartite graph always has a consistent labeling (hint: consider matchings in this graph).
(B) Prove that if G is bipartite so is the graph G^3 (the cubed graph).
(C) Let G be a (n, D)-graph and let H be a (D, d)-graph. Prove that if G is bipartite then $\mathsf{G}\mathsf{G}ⓩ\mathsf{H}$ is bipartite.
(D) Describe in detail a construction of an expander that is: (i) bipartite, and (ii) has consistent labeling at every stage of the construction (prove this property if necessary). For the ith graph in your series, what is its vertex degree, how many vertices it has, and what is the quality of expansion it provides?

Exercise 47.3.3 (Tension and bi-tension.). [**30 points**]

Disprove (i.e., give a counter example) that there exists a universal constant c, such that for any connected graph G, we have that $\gamma(\mathsf{G}) \leq \gamma_2(\mathsf{G}) \leq c\gamma(\mathsf{G})$.

Acknowledgments

Much of the presentation was followed suggestions by Manor Mendel. He also contributed some of the figures.

References

[AR94] N. Alon and Y. Roichman. Random cayley graphs and expanders. *Random Struct. Algorithms*, 5(2): 271–285, 1994.

[ASS08] N. Alon, O. Schwartz, and A. Shapira. An elementary construction of constant-degree expanders. *Combin. Probab. Comput.*, 17(3): 319–327, 2008.

[HLW06] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin Amer. Math. Soc.*, 43: 439–561, 2006.

[MN08] M. Mendel and A. Naor. *Towards a calculus for non-linear spectral gaps*. manuscript. 2008.

[RVW02] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. *Annals Math.*, 155(1): 157–187, 2002.

Chapter 48

The Probabilistic Method

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "Shortly after the celebration of the four thousandth anniversary of the opening of space, Angary J. Gustible discovered Gustible's planet. The discovery turned out to be a tragic mistake.
> Gustible's planet was inhabited by highly intelligent life forms. They had moderate telepathic powers. They immediately mind-read Angary J. Gustible's entire mind and life history, and embarrassed him very deeply by making up an opera concerning his recent divorce."
>
> Gustible's Planet, Cordwainer Smith

48.1. Introduction

The probabilistic method is a combinatorial technique to use probabilistic algorithms to create objects having desirable properties, and furthermore, prove that such objects exist. The basic technique is based on two basic observations:

1. If $\mathbb{E}[X] = \mu$, then there exists a value x of X, such that $x \geq \mathbb{E}[X]$.

2. If the probability of event $\mathcal{E}$ is larger than zero, then $\mathcal{E}$ exists and it is not empty.

The surprising thing is that despite the elementary nature of those two observations, they lead to a powerful technique that leads to numerous nice and strong results. Including some elementary proofs of theorems that previously had very complicated and involved proofs.

The main proponent of the probabilistic method, was Paul Erdős. An excellent text on the topic is the book by Noga Alon and Joel Spencer [AS00].

This topic is worthy of its own course. The interested student is refereed to the course "Math 475 — The Probabilistic Method".

48.1.1. Examples

48.1.1.1. Max cut

Computing the ***maximum cut*** (i.e., ***max cut***) in a graph is a NP-Complete problem, which is APX-Hard (i.e., no better than a constant approximation is possible if P $\neq$ NP). We present later on a better approximation algorithm, but the following simple algorithm already gives a pretty good approximation.

Theorem 48.1.1. *For any undirected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with n vertices and m edges, there is a partition of the vertex set V into two sets S and T, such that $|(S,T)| = |\{uv \in \mathsf{E} \mid u \in S \text{ and } v \in T\}| \geq \frac{m}{2}$. One can compute a partition, in $O(n)$ time, such that $\mathbb{E}[|(S,T)|] = m/2$.*

Proof: Consider the following experiment: randomly assign each vertex to S or T, independently and equal probability.

For an edge $e = uv$, the probability that one endpoint is in S, and the other in T is $1/2$, and let X_e be the indicator variable with value 1 if this happens. Clearly,

$$\mathbb{E}\Big[\Big|\{uv \in \mathsf{E} \;\Big|\; (u,v) \in S \times T \cup T \times S\}\Big|\Big] = \sum_{e \in E(G)} \mathbb{E}[X_e] = \sum_{e \in E(G)} \frac{1}{2} = \frac{m}{2}.$$

Thus, there must be an execution of the algorithm that computes a cut that is at least as large as the expectation – namely, a partition of V that satisfies the realizes a cut with $\geq m/2$ edges. ■

48.2. Maximum Satisfiability

In the MAX-SAT problem, we are given a binary formula F in ***CNF*** (Conjunctive normal form), and we would like to find an assignment that satisfies as many clauses as possible of F, for example $F = (x \vee y) \wedge (\overline{x} \vee z)$. Of course, an assignment satisfying all the clauses of the formula, and thus F itself, would be even better – but this problem is of course NPC. As such, we are looking for how well can be we do when we relax the problem to maximizing the number of clauses to be satisfied..

Theorem 48.2.1. *For any set of m clauses, there is a truth assignment of variables that satisfies at least $m/2$ clauses.*

Proof: Assign every variable a random value. Clearly, a clause with k variables, has probability $1 - 2^{-k}$ to be satisfied. Using linearity of expectation, and the fact that every clause has at least one variable, it follows, that $\mathbb{E}[X] = m/2$, where X is the random variable counting the number of clauses being satisfied. In particular, there exists an assignment for which $X \geq m/2$. ■

For an instant I, let $m_{\mathrm{opt}}(I)$, denote the maximum number of clauses that can be satisfied by the "best" assignment. For an algorithm **Alg**, let $m_{\mathbf{Alg}}(I)$ denote the number of clauses satisfied computed by the algorithm **Alg**. The ***approximation factor*** of **Alg**, is $m_{\mathbf{Alg}}(I)/m_{\mathrm{opt}}(I)$. Clearly, the algorithm of Theorem 48.2.1 provides us with $1/2$-approximation algorithm.

For every clause, C_j in the given instance, let $z_j \in \{0,1\}$ be a variable indicating whether C_j is satisfied or not. Similarly, let $x_i = 1$ if the ith variable is being assigned the value TRUE. Let C_j^+ be indices of the variables that appear in C_j in the positive, and C_j^- the indices of the variables that appear in the negative. Clearly, to solve MAX-SAT, we need to solve:

$$\begin{aligned}
\max \quad & \sum_{j=1}^{m} z_j & \\
\text{subject to} \quad & \sum_{i \in C_j^+} x_i + \sum_{i \in C_j^-} (1 - x_i) \geq z_j & \text{for all } j \\
& x_i, z_j \in \{0,1\} & \text{for all } i, j
\end{aligned}$$

We relax this into the following linear program:

$$\begin{aligned} \max \quad & \sum_{j=1}^{m} z_j \\ \textit{subject to} \quad & 0 \le y_i, z_j \le 1 \text{ for all } i, j \\ & \sum_{i \in C_j^+} y_i + \sum_{i \in C_j^-} (1 - y_i) \ge z_j \text{ for all } j. \end{aligned}$$

Which can be solved in polynomial time. Let $\widehat{t}$ denote the values assigned to the variable t by the linear-programming solution. Clearly, $\sum_{j=1}^{m} \widehat{z_j}$ is an upper bound on the number of clauses of I that can be satisfied.

We set the variable y_i to 1 with probability $\widehat{y_i}$. This is an instance ***randomized rounding***.

Lemma 48.2.2. *Let C_j be a clause with k literals. The probability that it is satisfied by randomized rounding is at least $\beta_k \widehat{z_j} \ge (1 - 1/e)\widehat{z_j}$, where*

$$\beta_k = 1 - \left(1 - \frac{1}{k}\right)^k \approx 1 - \frac{1}{e}.$$

Proof: Assume $C_j = y_1 \vee v_2 \ldots \vee v_k$. By the LP, we have $\widehat{y_1} + \cdots + \widehat{y_k} \ge \widehat{z_j}$. Furthermore, the probability that C_j is not satisfied is $\prod_{i=1}^{k}(1 - \widehat{y_i})$. Note that $1 - \prod_{i=1}^{k}(1 - \widehat{y_i})$ is minimized when all the $\widehat{y_i}$'s are equal (by symmetry). Namely, when $\widehat{y_i} = \widehat{z_j}/k$. Consider the function $f(x) = 1 - (1 - x/k)^k$. This function is larger than $g(x) = \beta_k x$, for all $0 \le x \le 1$, as can be easily verified (see Tedium 48.2.3).

Thus,

$$\mathbb{P}\left[C_j \text{ is satisfied}\right] = 1 - \prod_{i=1}^{k}(1 - \widehat{y_i}) \ge f\left(\widehat{z_j}\right) \ge \beta_k \widehat{z_j}.$$

The second part of the inequality, follows from the fact that $\beta_k \ge 1 - 1/e$, for all $k \ge 0$. Indeed, for $k = 1, 2$ the claim trivially holds. Furthermore,

$$1 - \left(1 - \frac{1}{k}\right)^k \ge 1 - \frac{1}{e} \quad \Leftrightarrow \quad \left(1 - \frac{1}{k}\right)^k \le \frac{1}{e},$$

but this holds since $1 - x \le e^{-x}$ implies that $1 - \frac{1}{k} \le e^{-1/k}$, and as such $\left(1 - \frac{1}{k}\right)^k \le e^{-k/k} = 1/e$. ■

Tedium 48.2.3. Consider the two functions

$$f(x) = 1 - (1 - x/k)^k \qquad \text{and} \qquad g(x) = \left(1 - \left(1 - \frac{1}{k}\right)^k\right)x.$$

We have $f'(x) = (1 - x/k)^{k-1}$ and $f''(x) = -\frac{k-1}{k}(1 - x/k)^{k-2}$. That is $f''(x) \le 0$, for $x \in [0, 1]$. As such f is a concave function.

Observe that $f(0) = 0 = g(0)$ and $f(1) = \left(1 - \left(1 - \frac{1}{k}\right)^k\right) = g(1)$. Since f is concave, and g is linear, it follows that $f(x) \ge g(x)$, for all $x \in [0, 1]$.

Theorem 48.2.4. *Given an instance I of MAX-SAT, the expected number of clauses satisfied by linear programming and randomized rounding is at least $(1 - 1/e) \approx 0.632 m_{\text{opt}}(I)$, where $m_{\text{opt}}(I)$ is the maximum number of clauses that can be satisfied on that instance.*

Theorem 48.2.5. *Given an instance I of MAX-SAT, let n_1 be the expected number of clauses satisfied by randomized assignment, and let n_2 be the expected number of clauses satisfied by linear programming followed by randomized rounding. Then,* $\max(n_1, n_2) \geq (3/4)\sum_j \widehat{z_j} \geq (3/4)m_{\text{opt}}(I)$.

Proof: It is enough to show that $(n_1 + n_2)/2 \geq \frac{3}{4}\sum_j \widehat{z_j}$. Let S_k denote the set of clauses that contain k literals. We know that

$$n_1 = \sum_k \sum_{C_j \in S_k} \left(1 - 2^{-k}\right) \geq \sum_k \sum_{C_j \in S_k} \left(1 - 2^{-k}\right)\widehat{z_j}.$$

By Lemma 48.2.2 we have $n_2 \geq \sum_k \sum_{C_j \in S_k} \beta_k \widehat{z_j}$. Thus,

$$\frac{n_1 + n_2}{2} \geq \sum_k \sum_{C_j \in S_k} \frac{1 - 2^{-k} + \beta_k}{2}\widehat{z_j}.$$

One can verify that $\left(1 - 2^{-k}\right) + \beta_k \geq 3/2$, for all k. [1] Thus, we have

$$\frac{n_1 + n_2}{2} \geq \frac{3}{4}\sum_k \sum_{C_j \in S_k} \widehat{z_j} = \frac{3}{4}\sum_j \widehat{z_j}.$$

■

References

[AS00] N. Alon and J. H. Spencer. *The Probabilistic Method*. 2nd. Wiley InterScience, 2000.

[1] Indeed, by the proof of Lemma 48.2.2, we have that $\beta_k \geq 1 - 1/e$. Thus, $\left(1 - 2^{-k}\right) + \beta_k \geq 2 - 1/e - 2^{-k} \geq 3/2$ for $k \geq 3$. Thus, we only need to check the inequality for $k = 1$ and $k = 2$, which can be done directly.

Chapter 49

The Probabilistic Method III

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> At other times you seemed to me either pitiable or contemptible, eunuchs, artificially confined to an eternal childhood, childlike and childish in your cool, tightly fenced, neatly tidied playground and kindergarten, where every nose is carefully wiped and every troublesome emotion is soothed, every dangerous thought repressed, where everyone plays nice, safe, bloodless games for a lifetime and every jagged stirring of life, every strong feeling, every genuine passion, every rapture is promptly checked, deflected and neutralized by meditation therapy.
>
> The Glass Bead Game, Hermann Hesse

49.1. The Lovász Local Lemma

Lemma 49.1.1. *(i)* $\mathbb{P}\Big[A \;\Big|\; B \cap C\Big] = \dfrac{\mathbb{P}\Big[A \cap B \;\Big|\; C\Big]}{\mathbb{P}\Big[B \;\Big|\; C\Big]}$

(ii) Let $\eta_1, \ldots, \eta_n$ be n events which are not necessarily independent. Then,

$$\mathbb{P}\Big[\cap_{i=1}^{n} \eta_i\Big] = \mathbb{P}\Big[\eta_1\Big] * \mathbb{P}\Big[\eta_2 \;\Big|\; \eta_1\Big] \mathbb{P}\Big[\eta_3 \;\Big|\; \eta_1 \cap \eta_2\Big] * \ldots * \mathbb{P}\Big[\eta_n \;\Big|\; \eta_1 \cap \ldots \cap \eta_{n-1}\Big].$$

Proof: (i) We have that

$$\frac{\mathbb{P}\Big[A \cap B \;\Big|\; C\Big]}{\mathbb{P}\Big[B \;\Big|\; C\Big]} = \frac{\mathbb{P}[A \cap B \cap C]}{\mathbb{P}[C]} \Big/ \frac{\mathbb{P}[B \cap C]}{\mathbb{P}[C]} = \frac{\mathbb{P}[A \cap B \cap C]}{\mathbb{P}[B \cap C]} = \mathbb{P}\Big[A \;\Big|\; B \cap C\Big].$$

As for (ii), we already saw it and used it in the minimum cut algorithm lecture. ■

Definition 49.1.2. An event $\mathcal{E}$ is mutually independent of a set of events $\mathcal{C}$, if for any subset $\mathcal{U} \subseteq \mathcal{C}$, we have that $\mathbb{P}[\mathcal{E} \cap (\bigcap_{\mathcal{E}' \in U} \mathcal{E}')] = \mathbb{P}[\mathcal{E}]\, \mathbb{P}[\bigcap_{\mathcal{E}' \in U} \mathcal{E}']$.

Let $\mathcal{E}_1, \ldots, \mathcal{E}_n$ be events. A ***dependency graph*** for these events is a directed graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, where $\{1, \ldots, n\}$, such that $\mathcal{E}_i$ is mutually independent of all the events in $\Big\{\mathcal{E}_j \;\Big|\; (i, j) \notin \mathsf{E}\Big\}$.

Intuitively, an edge (i, j) in a dependency graph indicates that $\mathcal{E}_i$ and $\mathcal{E}_j$ have (maybe) some dependency between them. We are interested in settings where this dependency is limited enough, that we can claim something about the probability of all these events happening simultaneously.

Lemma 49.1.3 (Lovász Local Lemma). *Let $\mathsf{G}(\mathsf{V}, \mathsf{E})$ be a dependency graph for events $\mathcal{E}_1, \ldots, \mathcal{E}_n$. Suppose that there exist $x_i \in [0, 1]$, for $1 \le i \le n$ such that $\mathbb{P}[\mathcal{E}_i] \le x_i \prod_{(i,j) \in E} \Big(1 - x_j\Big)$. Then $\mathbb{P}\Big[\cap_{i=1}^{n} \overline{\mathcal{E}_i}\Big] \ge \prod_{i=1}^{n} (1 - x_i)$.*

We need the following technical lemma.

Lemma 49.1.4. *Let* $\mathsf{G}(\mathsf{V},\mathsf{E})$ *be a dependency graph for events* $\mathcal{E}_1,\ldots,\mathcal{E}_n$. *Suppose that there exist* $x_i \in [0,1]$, *for* $1 \le i \le n$ *such that* $\mathbb{P}[\mathcal{E}_i] \le x_i \prod_{(i,j)\in E}\Big(1-x_j\Big)$. *Now, let* S *be a subset of the vertices from* $\{1,\ldots,n\}$, *and let* i *be an index not in* S. *We have that*

$$\mathbb{P}\Big[\mathcal{E}_i \;\Big|\; \cap_{j\in S}\overline{\mathcal{E}_j}\Big] \le x_i. \tag{49.1}$$

Proof: The proof is by induction on $k = |S|$.

For $k=0$, we have by assumption that $\mathbb{P}\Big[\mathcal{E}_i \;\Big|\; \cap_{j\in S}\overline{\mathcal{E}_j}\Big] = \mathbb{P}[\mathcal{E}_i] \le x_i \prod_{(i,j)\in E}\Big(1-x_j\Big) \le x_i$.

Thus, let $N = \Big\{ j \in S \;\Big|\; (i,j)\in E\Big\}$, and let $R = S \setminus N$. If $N=\emptyset$, then we have that $\mathcal{E}_i$ is mutually independent of the events of $\mathcal{C}(R) = \Big\{\mathcal{E}_j \;\Big|\; j\in R\Big\}$. Thus, $\mathbb{P}\Big[\mathcal{E}_i \;\Big|\; \cap_{j\in S}\overline{\mathcal{E}_j}\Big] = \mathbb{P}\Big[\mathcal{E}_i \;\Big|\; \cap_{j\in R}\overline{\mathcal{E}_j}\Big] = \mathbb{P}[\mathcal{E}_i] \le x_i$, by arguing as above.

By Lemma 49.1.1 (i), we have that

$$\mathbb{P}\left[\mathcal{E}_i \;\middle|\; \bigcap_{j\in S}\overline{\mathcal{E}_j}\right] = \frac{\mathbb{P}\Big[\mathcal{E}_i \cap \Big(\cap_{j\in N}\overline{\mathcal{E}_j}\Big) \;\Big|\; \cap_{m\in R}\overline{\mathcal{E}_m}\Big]}{\mathbb{P}\Big[\cap_{j\in N}\overline{\mathcal{E}_j} \;\Big|\; \cap_{m\in R}\overline{\mathcal{E}_m}\Big]}.$$

We bound the numerator by

$$\mathbb{P}\Big[\mathcal{E}_i \cap \Big(\cap_{j\in N}\overline{\mathcal{E}_j}\Big) \;\Big|\; \cap_{m\in R}\overline{\mathcal{E}_m}\Big] \le \mathbb{P}\Big[\mathcal{E}_i \;\Big|\; \cap_{m\in R}\overline{\mathcal{E}_m}\Big] = \mathbb{P}[\mathcal{E}_i] \le x_i \prod_{(i,j)\in E}\Big(1-x_j\Big),$$

since $\mathcal{E}_i$ is mutually independent of $\mathcal{C}(R)$. As for the denominator, let $N = \{j_1,\ldots,j_r\}$. We have, by Lemma 49.1.1 (ii), that

$$\begin{aligned}
\mathbb{P}\Big[\overline{\mathcal{E}_{j_1}} \cap \ldots \cap \overline{\mathcal{E}_{j_r}} \;\Big|\; \cap_{m\in R}\overline{\mathcal{E}_m}\Big] &= \mathbb{P}\Big[\overline{\mathcal{E}_{j_1}} \;\Big|\; \cap_{m\in R}\overline{\mathcal{E}_m}\Big]\, \mathbb{P}\Big[\overline{\mathcal{E}_{j_2}} \;\Big|\; \overline{\mathcal{E}_{j_1}} \cap \Big(\cap_{m\in R}\overline{\mathcal{E}_m}\Big)\Big] \\
&\quad \cdots \mathbb{P}\Big[\overline{\mathcal{E}_{j_r}} \;\Big|\; \overline{\mathcal{E}_{j_1}} \cap \ldots \cap \overline{\mathcal{E}_{j_{r-1}}} \cap \Big(\cap_{m\in R}\overline{\mathcal{E}_m}\Big)\Big] \\
&= \Big(1 - \mathbb{P}\Big[\mathcal{E}_{j_1} \;\Big|\; \cap_{m\in R}\overline{\mathcal{E}_m}\Big]\Big)\Big(1 - \mathbb{P}\Big[\mathcal{E}_{j_2} \;\Big|\; \overline{\mathcal{E}_{j_1}} \cap \Big(\cap_{m\in R}\overline{\mathcal{E}_m}\Big)\Big]\Big) \\
&\quad \cdots \Big(1 - \mathbb{P}\Big[\mathcal{E}_{j_r} \;\Big|\; \overline{\mathcal{E}_{j_1}} \cap \ldots \cap \overline{\mathcal{E}_{j_{r-1}}} \cap \Big(\cap_{m\in R}\overline{\mathcal{E}_m}\Big)\Big]\Big) \\
&\ge \Big(1-x_{j_1}\Big)\cdots\Big(1-x_{j_r}\Big) \ge \prod_{(i,j)\in E}\Big(1-x_j\Big),
\end{aligned}$$

by Eq. (49.1) and induction, as every probability term in the above expression has less than $|S|$ items involved. It thus follows, that $\mathbb{P}\Big[\mathcal{E}_i \;\Big|\; \cap_{j\in S}\overline{\mathcal{E}_j}\Big] \le x_i$. ■

Proof of Lovász local lemma (Lemma 49.1.3): Using Lemma 49.1.4, we have that

$$\mathbb{P}\Big[\cap_{i=1}^{n}\overline{\mathcal{E}_i}\Big] = (1-\mathbb{P}[\mathcal{E}_1])\Big(1-\mathbb{P}\Big[\mathcal{E}_2 \;\Big|\; \overline{\mathcal{E}_1}\Big]\Big)\cdots\Big(1-\mathbb{P}\Big[\mathcal{E}_n \;\Big|\; \cap_{i=1}^{n-1}\overline{\mathcal{E}_i}\Big]\Big) \ge \prod_{i=1}^{n}(1-x_i).$$

■

Corollary 49.1.5. *Let $\mathcal{E}_1, \ldots, \mathcal{E}_n$ be events, with $\mathbb{P}[\mathcal{E}_i] \leq p$ for all i. If each event is mutually independent of all other events except for at most d, and if $ep(d+1) \leq 1$, then $\mathbb{P}\left[\cap_{i=1}^n \overline{\mathcal{E}_i}\right] > 0$.*

Proof: If $d = 0$ the result is trivial, as the events are independent. Otherwise, there is a dependency graph, with every vertex having degree at most d. Apply Lemma 49.1.3 with $x_i = \frac{1}{d+1}$. Observe that

$$x_i(1-x_i)^d = \frac{1}{d+1}\left(1 - \frac{1}{d+1}\right)^d > \frac{1}{d+1} \cdot \frac{1}{e} \geq p,$$

by assumption and the since $\left(1 - \frac{1}{d+1}\right)^d > 1/e$, see Lemma 49.1.6 below. ■

The following is standard by now, and we include it only for the sake of completeness.

Lemma 49.1.6. *For any $n \geq 1$, we have $\left(1 - \frac{1}{n+1}\right)^n > \frac{1}{e}$.*

Proof: This is equivalent to $\left(\frac{n}{n+1}\right)^n > \frac{1}{e}$. Namely, we need to prove $e > \left(\frac{n+1}{n}\right)^n$. But this obvious, since $\left(\frac{n+1}{n}\right)^n = \left(1 + \frac{1}{n}\right)^n < \exp(n(1/n)) = e$. ■

49.2. Application to k-SAT

We are given a instance I of k-SAT, where every clause contains k literals, there are m clauses, and every one of the n variables, appears in at most $2^{k/50}$ clauses.

Consider a random assignment, and let $\mathcal{E}_i$ be the event that the ith clause was not satisfied. We know that $p = \mathbb{P}[\mathcal{E}_i] = 2^{-k}$, and furthermore, $\mathcal{E}_i$ depends on at most $d = k2^{k/50}$ other events. Since $ep(d+1) = e\left(k \cdot 2^{k/50} + 1\right)2^{-k} < 1$, for $k \geq 4$, we conclude that by Corollary 49.1.5, that

$$\mathbb{P}\left[I \text{ have a satisfying assignment}\right] = \mathbb{P}\left[\cup_i \mathcal{E}_i\right] > 0.$$

49.2.1. An efficient algorithm

The above just proves that a satisfying assignment exists. We next show a polynomial algorithm (in m) for the computation of such an assignment (the algorithm will not be polynomial in k).

Let G be the dependency graph for I, where the vertices are the clauses of I, and two clauses are connected if they share a variable. In the first stage of the algorithm, we assign values to the variables one by one, in an arbitrary order. In the beginning of this process all variables are unspecified, at each step, we randomly assign a variable either 0 or 1 with equal probability.

Definition 49.2.1. A clause $\mathcal{E}_i$ is ***dangerous*** if both the following conditions hold:

(i) $k/2$ literals of $\mathcal{E}_i$ have been fixed.

(ii) $\mathcal{E}_i$ is still unsatisfied.

After assigning each value, we discover all the dangerous clauses, and we defer ("freeze") all the unassigned variables participating in such a clause. We continue in this fashion till all the unspecified variables are frozen. This completes the first stage of the algorithm.

At the second stage of the algorithm, we will compute a satisfying assignment to the variables using brute force. This would be done by taking the surviving formula I' and breaking it into fragments, so that each fragment does not share any variable with any other fragment (naively, it might be that all of I' is one fragment). We can find a satisfying assignment to each fragment separately, and if each such fragment is "small" the resulting algorithm would be "fast".

We need to show that I' has a satisfying assignment and that the fragments are indeed small.

49.2.1.1. Analysis

A clause had ***survived*** if it is not satisfied by the variables fixed in the first stage. Note, that a clause that survived must have a dangerous clause as a neighbor in the dependency graph G. Not that I', the instance remaining from I after the first stage, has at least $k/2$ unspecified variables in each clause. Furthermore, every clause of I' has at most $d = k2^{k/50}$ neighbors in G', where G' is the dependency graph for I'. It follows, that again, we can apply Lovász local lemma to conclude that I' has a satisfying assignment.

Definition 49.2.2. Two connected graphs $\mathsf{G}_1 = (V_1, E_1)$ and $\mathsf{G}_2 = (V_2, E_2)$, where $V_1, V_2 \subseteq \{1, \ldots, n\}$ are ***unique*** if $V_1 \neq V_2$.

Lemma 49.2.3. *Let G be a graph with degree at most d and with n vertices. Then, the number of unique subgraphs of G having r vertices is at most nd^{2r}.*

Proof: Consider a unique subgraph $\widehat{G}$ of G, which by definition is connected. Let H be a connected subtree of G spanning $\widehat{G}$. Duplicate every edge of H, and let H' denote the resulting graph. Clearly, H' is Eulerian, and as such posses a Eulerian path π of length at most $2(r-1)$, which can be specified, by picking a starting vertex v, and writing down for the i-th vertex of π which of the d possible neighbors, is the next vertex in π. Thus, there are st most $nd^{2(r-1)}$ ways of specifying π, and thus, there are at most $nd^{2(r-1)}$ unique subgraphs in G of size r. ■

Lemma 49.2.4. *With probability $1 - o(1)$, all connected components of G' have size at most $O(\log m)$, where G' denote the dependency graph for I'.*

Proof: Let G_4 be a graph formed from G by connecting any pair of vertices of G of distance *exactly* 4 from each other. The degree of a vertex of G_4 is at most $O(d^4)$.

Let U be a set of r vertices of G, such that every pair is in distance at least 4 from each other in G. We are interested in bounding the probability that all the clauses of U survive the first stage.

The probability of a clause to be dangerous is at most $2^{-k/2}$, as we assign (random) values to half of the variables of this clause. Now, a clause survive only if it is dangerous or one of its neighbors is dangerous. Thus, the probability that a clause survive is bounded by $2^{-k/2}(d+1)$.

Furthermore, the survival of two clauses $\mathcal{E}_i$ and $\mathcal{E}_j$ in U is an independent event, as no *neighbor* of $\mathcal{E}_i$ shares a variable with a neighbor of $\mathcal{E}_j$ (because of the distance 4 requirement). We conclude, that the probability that all the vertices of U to appear in G' is bounded by

$$\left(2^{-k/2}(d+1)\right)^r.$$

In fact, we are interested in sets U that induce a connected subgraphs of G_4. The number of unique such sets of size r is bounded by the number of unique subgraphs of G_4 of size r, which is bounded by md^{8r}, by Lemma 49.2.3. Thus, the probability of any connected subgraph of G_4 of size $r = \log_2 m$ to survive in G' is smaller than

$$md^{8r}\left(2^{-k/2}(d+1)\right)^r = m\left(k2^{k/50}\right)^{8r}\left(2^{-k/2}(k2^{k/50}+1)\right)^r \leq m2^{kr/5}\cdot 2^{-kr/4} = m2^{-kr/20} = o(1),$$

since $k \geq 50$. (Here, a subgraph survive of G_4 survive, if all its vertices appear in G'.) Note, however, that if a connected component of G' has more than L vertices, than there must be a connected component having L/d^3 vertices in G_4 that had survived in G'. We conclude, that with probability $o(1)$, no connected component of G' has more than $O(d^3 \log m) = O(\log m)$ vertices (note, that we consider k to be a constant, and thus, also d). ■

Thus, after the first stage, we are left with fragments of $(k/2)$-SAT, where every fragment has size at most $O(\log m)$, and thus having at most $O(\log m)$ variables. Thus, we can by brute force find the satisfying assignment to each such fragment in time polynomial in m. We conclude:

Theorem 49.2.5. *The above algorithm finds a satisfying truth assignment for any instance of k-SAT containing m clauses, which each variable is contained in at most $2^{k/50}$ clauses, in expected time polynomial in m.*

Chapter 50

The Probabilistic Method IV

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> Once I sat on the steps by a gate of David's Tower, I placed my two heavy baskets at my side. A group of tourists was standing around their guide and I became their target marker. "You see that man with the baskets? Just right of his head there's an arch from the Roman period. Just right of his head." "But he's moving, he's moving!" I said to myself: redemption will come only if their guide tells them, "You see that arch from the Roman period? It's not important: but next to it, left and down a bit, there sits a man who's bought fruit and vegetables for his family."
>
> Yehuda Amichai, Tourists

50.1. The Method of Conditional Probabilities

In previous lectures, we encountered the following problem.

Problem 50.1.1 (Set Balancing/Discrepancy). Given a binary matrix M of size $n \times n$, find a vector $\mathbf{v} \in \{-1, +1\}^n$, such that $\|\mathsf{M}\mathbf{v}\|_\infty$ is minimized.

Using random assignment and the Chernoff inequality, we showed that there exists $\mathbf{v}$, such that $\|\mathsf{M}\mathbf{v}\|_\infty \leq 4\sqrt{n \ln n}$. Can we derandomize this algorithm? Namely, can we come up with an efficient *deterministic* algorithm that has low discrepancy?

To derandomize our algorithm, construct a computation tree of depth n, where in the ith level we expose the ith coordinate of $\mathbf{v}$. This tree T has depth n. The root represents all possible random choices, while a node at depth i, represents all computations when the first i bits are fixed. For a node $v \in T$, let $P(v)$ be the probability that a random computation starting from v succeeds – here randomly assigning the remaining bits can be interpreted as a random walk down the tree to a leaf.

Formally, the algorithm is ***successful*** if ends up with a vector $\mathbf{v}$, such that $\|\mathsf{M}\mathbf{v}\|_\infty \leq 4\sqrt{n \ln n}$.

Let v_l and v_r be the two children of v. Clearly, $P(v) = (P(v_l) + P(v_r))/2$. In particular, $\max(P(v_l), P(v_r)) \geq P(v)$. Thus, if we could compute $P(\cdot)$ quickly (and deterministically), then we could derandomize the algorithm.

Let C_m^+ be the bad event that $\mathbf{r}_m \cdot \mathbf{v} > 4\sqrt{n \log n}$, where $\mathbf{r}_m$ is the mth row of M. Similarly, C_m^- is the bad event that $\mathbf{r}_m \cdot \mathbf{v} < -4\sqrt{n \log n}$, and let $C_m = C_m^+ \cup C_m^-$. Consider the probability, $\mathbb{P}\Big[C_m^+ \;\Big|\; \mathbf{v}_1, \ldots, \mathbf{v}_k\Big]$ (namely, the first k coordinates of $\mathbf{v}$ are specified). Let $\mathbf{r}_m = (r_1, \ldots, r_n)$. We have that

$$\mathbb{P}\Big[C_m^+ \;\Big|\; \mathbf{v}_1, \ldots, \mathbf{v}_k\Big] = \mathbb{P}\left[\sum_{i=k+1}^{n} \mathbf{v}_i r_i > 4\sqrt{n \log n} - \sum_{i=1}^{k} \mathbf{v}_i r_i\right] = \mathbb{P}\left[\sum_{i \geq k+1, r_i \neq 0} \mathbf{v}_i r_i > L\right] = \mathbb{P}\left[\sum_{i \geq k+1, r_i = 1} \mathbf{v}_i > L\right],$$

where $L = 4\sqrt{n \log n} - \sum_{i=1}^{k} \mathbf{v}_i r_i$ is a known quantity (since $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are known). Let $V = \sum_{i \geq k+1, r_i = 1} 1$. We have,

$$\mathbb{P}\Big[C_m^+ \;\Big|\; \mathbf{v}_1, \ldots, \mathbf{v}_k\Big] = \mathbb{P}\left[\sum_{\substack{i \geq k+1 \\ \alpha_i = 1}} (\mathbf{v}_i + 1) > L + V\right] = \mathbb{P}\left[\sum_{\substack{i \geq k+1 \\ \alpha_i = 1}} \frac{\mathbf{v}_i + 1}{2} > \frac{L+V}{2}\right],$$

The last quantity is the probability that in V flips of a fair $0/1$ coin one gets more than $(L+V)/2$ heads. Thus,

$$P_m^+ = \mathbb{P}\Big[C_m^+ \;\Big|\; \mathbf{v}_1, \ldots, \mathbf{v}_k\Big] = \sum_{i=\lceil (L+V)/2 \rceil}^{\mathsf{V}} \binom{\mathsf{V}}{i} \frac{1}{2^n} = \frac{1}{2^n} \sum_{i=\lceil (L+V)/2 \rceil}^{\mathsf{V}} \binom{\mathsf{V}}{i}.$$

This implies, that we can compute P_m^+ in polynomial time! Indeed, we are adding $V \leq n$ numbers, each one of them is a binomial coefficient that has polynomial size representation in n, and can be computed in polynomial time (why?). One can define in similar fashion P_m^-, and let $P_m = P_m^+ + P_m^-$. Clearly, P_m can be computed in polynomial time, by applying a similar argument to the computation of $P_m^- = \mathbb{P}\Big[C_m^- \;\Big|\; \mathbf{v}_1, \ldots, \mathbf{v}_k\Big]$.

For a node $v \in T$, let $\mathbf{v}_v$ denote the portion of $\mathbf{v}$ that was fixed when traversing from the root of T to v. Let $P(v) = \sum_{m=1}^{n} \mathbb{P}\Big[C_m \;\Big|\; \mathbf{v}_v\Big]$. By the above discussion $P(v)$ can be computed in polynomial time. Furthermore, we know, by the previous result on discrepancy that $P(r) < 1$ (that was the bound used to show that there exist a good assignment).

As before, for any $v \in T$, we have $P(v) \geq \min(P(v_l), P(v_r))$. Thus, we have a polynomial *deterministic* algorithm for computing a set balancing with discrepancy smaller than $4\sqrt{n \log n}$. Indeed, set $v = root(T)$. And start traversing down the tree. At each stage, compute $P(v_l)$ and $P(v_r)$ (in polynomial time), and set v to the child with lower value of $P(\cdot)$. Clearly, after n steps, we reach a leaf, that corresponds to a vector $\mathbf{v}'$ such that $\|A\mathbf{v}'\|_\infty \leq 4\sqrt{n \log n}$.

Theorem 50.1.2. *Using the method of conditional probabilities, one can compute in polynomial time in n, a vector* $\mathbf{v} \in \{-1, 1\}^n$*, such that* $\|A\mathbf{v}\|_\infty \leq 4\sqrt{n \log n}$.

Note, that this method might fail to find the best assignment.

50.2. Independent set in a graph

Theorem 50.2.1. *Consider a graph* $\mathsf{G} = (\llbracket n \rrbracket, \mathsf{E})$*, with n vertices an m edges. Then* G *contains an independent set of size*

$$\geq f(n, m) = n/(2m/n + 1).$$

In particular, a randomized algorithm can compute an independent set of expected size $\Omega(f(n, m))$.

Proof: Consider a random permutation of the vertices, and in the ith iteration add the vertex π_i to the independent set if none of its neighbors are in the independent set. Let I be the resulting independent set. We have for a vertex $v \in \llbracket n \rrbracket$ that

$$\mathbb{P}[v \in I] \geq \frac{1}{d(i) + 1}.$$

As such, the expected size of the computed independent set is

$$\Gamma = \sum_{i=1}^{n} \mathbb{P}[i \in I] \geq \sum_{i=1}^{n} \frac{1}{d(i) + 1}.$$

Observe that for $x > 0$, and $\alpha \geq x$, we have that

$$1/(1+x) + 1/(1+\alpha-x) = \frac{\alpha}{(1+x)(1+\alpha-x)}.$$

achieves its minimum when $x = \alpha/2$.

As such, $\sum_{i=1}^{n} \frac{1}{d(i)+1}$ is minimized when all the $d(\cdot)$ are equal. Which means that

$$\Gamma \geq \sum_{i=1}^{n} \frac{1}{d(i)+1}. \geq \sum_{i=1}^{n} \frac{1}{(2m/n)+1}. = \frac{n}{(2m/n)+1},$$

as claimed. ■

50.3. A Short Excursion into Combinatorics via the Probabilistic Method

In this section, we provide some additional examples of the Probabilistic Method to prove some results in combinatorics and discrete geometry. While the results are not directly related to our main course, their beauty, hopefully, will speak for itself.

50.3.1. High Girth and High Chromatic Number

Definition 50.3.1. For a graph G, let $\alpha(\mathsf{G})$ be the cardinality of the largest independent set in G, $\chi(\mathsf{G})$ denote the chromatic number of G, and let $\text{girth}(\mathsf{G})$ denote the length of the shortest circle in G.

Theorem 50.3.2. *For all K, L there exists a graph G with $\text{girth}(\mathsf{G}) > L$ and $\chi(\mathsf{G}) > K$.*

Proof: Fix $\mu < 1/L$, and let $\mathsf{G} \approx \mathsf{G}(n,p)$ with $p = n^{\mu-1}$; namely, G is a random graph on n vertices chosen by picking each pair of vertices to be an edge in G, randomly and independently with probability p. Let X be the number of cycles of size at most L. Then

$$\mathbb{E}[X] = \sum_{i=3}^{L} \frac{n!}{(n-i)!} \cdot \frac{1}{2i} \cdot p^i \leq \sum_{i=3}^{L} \frac{n^i}{2i} \cdot \left(n^{\mu-1}\right)^i \leq \sum_{i=3}^{L} \frac{n^{\mu i}}{2i} = o(n),$$

as $\mu L < 1$, and since the number of different sequence of i vertices is $\frac{n!}{(n-i)!}$, and every cycle is being counted in this sequence $2i$ times.

In particular, $\mathbb{P}[X \geq n/2] = o(1)$.

Let $x = \left\lceil \frac{3}{p} \ln n \right\rceil + 1$. We remind the reader that $\alpha(\mathsf{G})$ denotes the size of the largest independent set in G. We have that

$$\mathbb{P}\Big[\alpha(\mathsf{G}) \geq x\Big] \leq \binom{n}{x}(1-p)^{\binom{x}{2}} < \left(n \exp\left(-\frac{p(x-1)}{2}\right)\right)^x < \left(n \exp\left(-\frac{3}{2}\ln n\right)\right)^x < \Big(o(1)\Big)^x = o(1).$$

Let n be sufficiently large so that both these events have probability less than $1/2$. Then there is a specific G with less than $n/2$ cycles of length at most L and with $\alpha(\mathsf{G}) < 3n^{1-\mu}\ln n + 1$.

Remove from G a vertex from each cycle of length at most L. This gives a graph G^* with at least $n/2$ vertices. G^* has girth greater than L and $\alpha(G^*) \leq \alpha(\mathsf{G})$ (any independent set in G^* is also an independent set in G). Thus

$$\chi(G^*) \geq \frac{|V(G^*)|}{\alpha(G^*)} \geq \frac{n/2}{3n^{1-\mu}\ln n} \geq \frac{n^\mu}{12\ln n}.$$

To complete the proof, let n be sufficiently large so that this is greater than K. ■

50.3.2. Crossing Numbers and Incidences

The following problem has a long and very painful history. It is truly amazing that it can be solved by such a short and elegant proof.

And ***embedding*** of a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ in the plane is a planar representation of it, where each vertex is represented by a point in the plane, and each edge uv is represented by a curve connecting the points corresponding to the vertices u and v. The ***crossing number*** of such an embedding is the number of pairs of intersecting curves that correspond to pairs of edges with no common endpoints. The ***crossing number*** $\mathrm{cr}(\mathsf{G})$ of G is the minimum possible crossing number in an embedding of it in the plane.

Theorem 50.3.3. *The crossing number of any simple graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with $|\mathsf{E}| \geq 4|\mathsf{V}|$ is $\geq \dfrac{|\mathsf{E}|^3}{64|\mathsf{V}|^2}$.*

Proof: By Euler's formula any simple planar graph with n vertices has at most $3n-6$ edges. (Indeed, $f-e+v = 2$ in the case with maximum number of edges, we have that every face, has 3 edges around it. Namely, $3f = 2e$. Thus, $(2/3)e - e + v = 2$ in this case. Namely, $e = 3v - 6$.) This implies that the crossing number of any simple graph with n vertices and m edges is at least $m - 3n + 6 > m - 3n$. Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be a graph with $|\mathsf{E}| \geq 4|\mathsf{V}|$ embedded in the plane with $t = \mathrm{cr}(\mathsf{G})$ crossings. Let H be the random induced subgraph of G obtained by picking each vertex of G randomly and independently, to be a vertex of H with probabilistic p (where P will be specified shortly). The expected number of vertices of H is $p|\mathsf{V}|$, the expected number of its edges is $p^2|\mathsf{E}|$, and the expected number of crossings in the given embedding is $p^4 t$, implying that the expected value of its crossing number is at most $p^4 t$. Therefore, we have $p^4 t \geq p^2|\mathsf{E}| - 3p|\mathsf{V}|$, implying that

$$\mathrm{cr}(\mathsf{G}) \geq \frac{|\mathsf{E}|}{p^2} - \frac{3|\mathsf{V}|}{p^3},$$

let $p = 4|\mathsf{V}|/|\mathsf{E}| < 1$, and we have $\mathrm{cr}(\mathsf{G}) \geq (1/16 - 3/64)|\mathsf{E}|^3/|\mathsf{V}|^2 = |\mathsf{E}|^3/(64|\mathsf{V}|^2)$. ■

Theorem 50.3.4. *Let P be a set of n distinct points in the plane, and let L be a set of m distinct lines. Then, the number of incidences between the points of P and the lines of L (that is, the number of pairs (p, ℓ) with $p \in P$, $\ell \in L$, and $p \in \ell$) is at most $c\left(m^{2/3}n^{2/3} + m + n\right)$, for some absolute constant c.*

Proof: Let I denote the number of such incidences. Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be the graph whose vertices are all the points of P, where two are adjacent if and only if they are consecutive points of P on some line in L. Clearly $|\mathsf{V}| = n$, and $|\mathsf{E}| = I - m$. Note that G is already given embedded in the plane, where the edges are presented by segments of the corresponding lines of L.

Either, we can not apply Theorem 50.3.3, implying that $I - m = |\mathsf{E}| < 4|\mathsf{V}| = 4n$. Namely, $I \leq m + 4n$. Or alliteratively,

$$\frac{|\mathsf{E}|^3}{64|\mathsf{V}|^2} = \frac{(I-m)^3}{64n^2} \leq \mathrm{cr}(\mathsf{G}) \leq \binom{m}{2} \leq \frac{m^2}{2}.$$

Implying that $I \leq (32)^{1/3}m^{2/3}n^{2/3} + m$. In both cases, $I \leq 4(m^{2/3}n^{2/3} + m + n)$. ■

This technique has interesting and surprising results, as the following theorem shows.

Theorem 50.3.5. *For any three sets A, B and C of s real numbers each, we have*

$$|A \cdot B + C| = \left|\left\{ab + c \;\middle|\; a \in A, b \in B, mc \in C\right\}\right| \geq \Omega\left(s^{3/2}\right).$$

Proof: Let $R = A \cdot B + C$, $|R| = r$ and define $P = \left\{(a,t) \;\middle|\; a \in A, t \in R\right\}$, and $L = \left\{y = bx + c \;\middle|\; b \in B, c \in C\right\}$.

Clearly $n = |P| = sr$, and $m = |L| = s^2$. Furthermore, a line $y = bx + c$ of L is incident with s points of R, namely with $\left\{(a,t) \;\middle|\; a \in A, t = ab + c\right\}$. Thus, the overall number of incidences is at least s^3. By Theorem 50.3.4, we have

$$s^3 \le 4\Big(m^{2/3}n^{2/3} + m + n\Big) = 4\Big(\big(s^2\big)^{2/3}(sr)^{2/3} + s^2 + sr\Big) = 4\Big(s^2r^{2/3} + s^2 + sr\Big).$$

For $r < s^3$, we have that $sr \le s^2r^{2/3}$. Thus, for $r < s^3$, we have $s^3 \le 12s^2r^{2/3}$, implying that $s^{3/2} \le 12r$. Namely, $|R| = \Omega(s^{3/2})$, as claimed. ■

Among other things, the crossing number technique implies a better bounds for k-sets in the plane than what was previously known. The k-set problem had attracted a lot of research, and remains till this day one of the major open problems in discrete geometry.

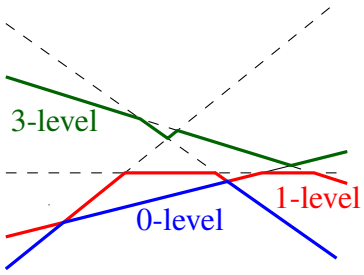
50.3.3. Bounding the at most k-level

Let L be a set of n lines in the plane. Assume, without loss of generality, that no three lines of L pass through a common point, and none of them is vertical. The complement of union of lines L break the plane into regions known as ***faces***. An intersection of two lines, is a ***vertex***, and the maximum interval on a line between two vertices is am ***edge***. The whole structure of vertices, edges and faces induced by L is known as ***arrangement*** of L, denoted by $\mathcal{A}(\mathsf{L})$.

Let L be a set of n lines in the plane. A point $\mathsf{p} \in \bigcup_{\ell \in \mathsf{L}} \ell$ is of ***level*** k if there are k lines of L strictly below it. The ***k-level*** is the closure of the set of points of level k. Namely, the k-level is an x-monotone curve along the lines of L.t

The 0-level is the boundary of the "bottom" face of the arrangement of L (i.e., the face containing the negative y-axis). It is easy to verify that the 0-level has at most $n - 1$ vertices, as each line might contribute at most one segment to the 0-level (which is an unbounded convex polygon).

It is natural to ask what the number of vertices at the k-level is (i.e., what the combinatorial complexity of the polygonal chain forming the k-level is). This is a surprisingly hard question, but the same question on the complexity of the at most k-level is considerably easier.



Theorem 50.3.6. *The number of vertices of level at most k in an arrangement of n lines in the plane is $O(nk)$.*

Proof: Pick a random sample R of L, by picking each line to be in the sample with probability $1/k$. Observe that

$$\mathbb{E}[|\mathsf{R}|] = \frac{n}{k}.$$

Let $\mathbb{L}_{\le k} = \mathbb{L}_{\le k}(\mathsf{L})$ be the set of all vertices of $\mathcal{A}(\mathsf{L})$ of level at most k, for $k > 1$. For a vertex $\mathsf{p} \in \mathbb{L}_{\le k}$, let X_{p} be an indicator variable which is 1 if p is a vertex of the 0-level of $\mathcal{A}(\mathsf{R})$. The probability that p is in the 0-level of $\mathcal{A}(\mathsf{R})$ is the probability that none of the j lines below it are picked to be in the sample, and the two lines that define it do get selected to be in the sample. Namely,

$$\mathbb{P}\Big[X_{\mathsf{p}} = 1\Big] = \left(1 - \frac{1}{k}\right)^j \left(\frac{1}{k}\right)^2 \ge \left(1 - \frac{1}{k}\right)^k \frac{1}{k^2} \ge \exp\left(-2\frac{k}{k}\right)\frac{1}{k^2} = \frac{1}{e^2k^2}$$

since $j \le k$ and $1 - x \ge e^{-2x}$, for $0 < x \le 1/2$.

On the other hand, the number of vertices on the 0-level of R is at most $|\mathsf{R}| - 1$. As such,

$$\sum_{\mathsf{p} \in \mathbb{L}_{\leq k}} X_{\mathsf{p}} \leq |\mathsf{R}| - 1.$$

Moreover this, of course, also holds in expectation, implying

$$\mathbb{E}\left[\sum_{\mathsf{p} \in \mathbb{L}_{\leq k}} X_{\mathsf{p}}\right] \leq \mathbb{E}\Big[|\mathsf{R}| - 1\Big] \leq \frac{n}{k}.$$

On the other hand, by linearity of expectation, we have

$$\mathbb{E}\left[\sum_{\mathsf{p} \in \mathbb{L}_{\leq k}} X_{\mathsf{p}}\right] = \sum_{\mathsf{p} \in \mathbb{L}_{\leq k}} \mathbb{E}\Big[X_{\mathsf{p}}\Big] \geq \frac{|\mathbb{L}_{\leq k}|}{e^2 k^2}.$$

Putting these two inequalities together, we get that $\frac{|\mathbb{L}_{\leq k}|}{e^2 k^2} \leq \frac{n}{k}$. Namely, $|\mathbb{L}_{\leq k}| \leq e^2 nk$. ■

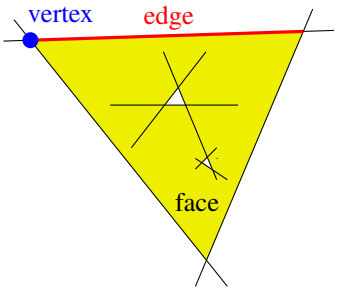
Chapter 51

Sampling and the Moments Technique

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> Sun and rain and bush had made the site look old, like the site of a dead civilization. The ruins, spreading over so many acres, seemed to speak of a final catastrophe. But the civilization wasn't dead. It was the civilization I existed in and in fact was still working towards. And that could make for an odd feeling: to be among the ruins was to have your time-sense unsettled. You felt like a ghost, not from the past, but from the future. You felt that your life and ambition had already been lived out for you and you were looking at the relics of that life. You were in a place where the future had come and gone.
>
> A bend in the river, V. S. Naipaul

51.1. Vertical decomposition



Given a set S of n segments in the plane, its ***arrangement***, denoted by $\mathcal{A}(\mathsf{S})$, is the decomposition of the plane into faces, edges, and vertices. The ***vertices*** of $\mathcal{A}(\mathsf{S})$ are the endpoints, or the intersection points of the segments of S, the ***edges*** are the maximal connected portions of the segments not containing any vertex, and the ***faces*** are the connected components of the complement of the union of the segments of S. These definitions are depicted on the right.

For numerical reasons (and also conceptually), a symbolic representation would be better than a numerical one. Thus, an intersection vertex would be represented by two pointers to the segments that their intersection is this vertex. Similarly, an edge would be represented as a pointer to the segment that contains it, and two pointers to the vertices forming its endpoints.

Naturally, we are assuming here that we have geometric primitives that can resolve any decision problem of interest that involve a few geometric entities. For example, for a given segment s and a point p, we would be interested in deciding if p lies vertically below s. From a theoretical point of view, all these primitives require a constant amount of computation, and are "easy". In the real world, numerical issues and degeneracies make implementing these primitives surprisingly challenging. We are going to ignore this major headache here, but the reader should be aware of it.

We will be interested in computing the arrangement $\mathcal{A}(\mathsf{S})$ and a representation of it that makes it easy to manipulate. In particular, we would like to be able to quickly resolve questions of the type

(i) are two points in the same face?

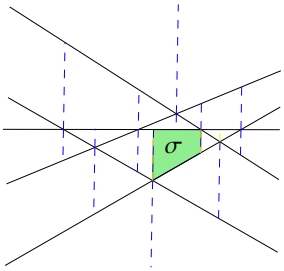
(ii) can one traverse from one point to the other without crossing any segment?

The naive representation of each face as polygons (potentially with holes) is not conducive to carrying out such tasks, since a polygon might be arbitrarily complicated. Instead, we will prefer to break the arrangement into smaller canonical tiles.

To this end, a ***vertical trapezoid*** is a quadrangle with two vertical sides. The breaking of the faces into such trapezoids is the ***vertical decomposition*** of the arrangement $\mathcal{A}(\mathsf{S})$.

Formally, for a subset $\mathsf{R} \subseteq \mathsf{S}$, let $\mathcal{A}^{|}(\mathsf{R})$ denote the ***vertical decomposition*** of the plane formed by the arrangement $\mathcal{A}(\mathsf{R})$ of the segments of R. This is the partition of the plane into interior disjoint vertical trapezoids formed by erecting vertical walls through each vertex of $\mathcal{A}^{|}(\mathsf{R})$.

A ***vertex*** of $\mathcal{A}^{|}(\mathsf{R})$ is either an endpoint of a segment of R or an intersection point of two of its segments. From each such vertex we shoot up (similarly, down) a vertical ray till it hits a segment of R or it continues all the way to infinity. See the figure on the right.



Note that a vertical trapezoid is defined by at most four segments: two segments defining its ceiling and floor and two segments defining the two intersection points that induce the two vertical walls on its boundary. Of course, a vertical trapezoid might be degenerate and thus be defined by fewer segments (i.e., an unbounded vertical trapezoid or a triangle with a vertical segment as one of its sides).

Vertical decomposition breaks the faces of the arrangement that might be arbitrarily complicated into entities (i.e., vertical trapezoids) of constant complexity. This makes handling arrangements (decomposed into vertical trapezoid) much easier computationally.

In the following, we assume that the n segments of S have k pairwise intersection points overall, and we want to compute the arrangement $\mathcal{A} = \mathcal{A}(\mathsf{S})$; namely, compute the edges, vertices, and faces of $\mathcal{A}(\mathsf{S})$. One possible way is the following: Compute a random permutation of the segments of S: $\mathsf{S} = \langle \mathsf{s}_1, \ldots, \mathsf{s}_n \rangle$. Let $\mathsf{S}_i = \langle \mathsf{s}_1, \ldots, \mathsf{s}_i \rangle$ be the prefix of length i of S. Compute $\mathcal{A}^{|}(\mathsf{S}_i)$ from $\mathcal{A}^{|}(\mathsf{S}_{i-1})$, for $i = 1, \ldots, n$. Clearly, $\mathcal{A}^{|}(\mathsf{S}) = \mathcal{A}^{|}(\mathsf{S}_n)$, and we can extract $\mathcal{A}(\mathsf{S})$ from it. Namely, in the ith iteration, we insert the segment s_i into the arrangement $\mathcal{A}^{|}(\mathsf{S}_{i-1})$.

This technique of building the arrangement by inserting the segments one by one is called ***randomized incremental construction***.

Who need these pesky arrangements anyway? The reader might wonder who needs arrangements? As a concrete examples, consider a situation where you are give several maps of a city containing different layers of information (i.e., streets map, sewer map, electric lines map, train lines map, etc). We would like to compute the overlay map formed by putting all these maps on top of each other. For example, we might be interested in figuring out if there are any buildings lying on a planned train line, etc.

More generally, think about a set of general constraints in $\mathbb{R}^d$. Each constraint is bounded by a surface, or a patch of a surface. The decomposition of $\mathbb{R}^d$ formed by the arrangement of these surfaces gives us a description of the parametric space in a way that is algorithmically useful. For example, finding if there is a point inside all the constraints, when all the constraints are induced by linear inequalities, is linear programming. Namely, arrangements are a useful way to think about any parametric space partitioned by various constraints.

51.1.1. Randomized incremental construction (RIC)

Imagine that we had computed the arrangement $\mathcal{B}_{i-1} = \mathcal{A}^{|}(\mathsf{S}_{i-1})$. In the ith iteration we compute $\mathcal{B}_i$ by inserting s_i into the arrangement $\mathcal{B}_{i-1}$. This involves splitting some trapezoids (and merging some others).

As a concrete example, consider Figure 51.1. Here we insert s in the arrangement. To this end we split the "vertical trapezoids" $\triangle\mathsf{pux}$ and $\triangle\mathsf{yux}$, each into three trapezoids. The two trapezoids σ' and σ'' now need to be merged together to form the new trapezoid which appears in the vertical decomposition of the new arrangement. (Note that the figure does not show all the trapezoids in the vertical decomposition.)

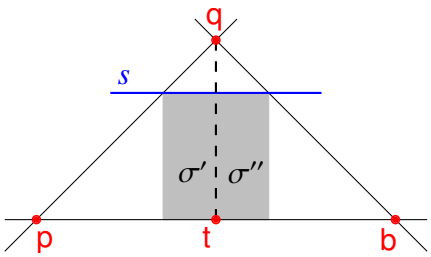
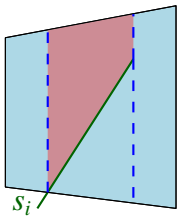


Figure 51.1

To facilitate this, we need to compute the trapezoids of $\mathcal{B}_{i-1}$ that intersect s_i. This is done by maintaining a ***conflict graph***. Each trapezoid $\sigma \in \mathcal{A}^{|}(\mathsf{S}_{i-1})$ maintains a ***conflict list*** $\mathrm{cl}(\sigma)$ of *all* the segments of S that intersect its interior. In particular, the conflict list of σ cannot contain any segment of S_{i-1}, and as such it contains only the segments of $\mathsf{S} \setminus \mathsf{S}_{i-1}$ that intersect its interior. We also maintain a similar structure for each segment, listing all the trapezoids of $\mathcal{A}^{|}(\mathsf{S}_{i-1})$ that it currently intersects (in its interior). We maintain those lists with cross pointers, so that given an entry (σ, s) in the conflict list of σ, we can find the entry (s, σ) in the conflict list of s in constant time.



Thus, given s_i, we know what trapezoids need to be split (i.e., all the trapezoids in $\mathrm{cl}(\mathsf{s}_i)$). Splitting a trapezoid σ by a segment s_i is the operation of computing a set of (at most) four trapezoids that cover σ and have s_i on their boundary. We compute those new trapezoids, and next we need to compute the conflict lists of the new trapezoids. This can be easily done by taking the conflict list of a trapezoid $\sigma \in \mathrm{cl}(\mathsf{s}_i)$ and distributing its segments among the $O(1)$ new trapezoids that cover σ. Using careful implementation, this requires a linear time in the size of the conflict list of σ.

Note that only trapezoids that intersect s_i in their interior get split. Also, we need to update the conflict lists for the segments (that were not inserted yet).

We next sketch the low-level details involved in maintaining these conflict lists. For a segment s that intersects the interior of a trapezoid σ, we maintain the pair (s, σ). For every trapezoid σ, in the current vertical decomposition, we maintain a doubly linked list of all such pairs that contain σ. Similarly, for each segment s we maintain the doubly linked list of all such pairs that contain s. Finally, each such pair contains two pointers to the location in the two respective lists where the pair is being stored.

It is now straightforward to verify that using this data-structure we can implement the required operations in linear time in the size of the relevant conflict lists.

In the above description, we ignored the need to merge adjacent trapezoids if they have identical floor and ceiling – this can be done by a somewhat straightforward and tedious implementation of the vertical decomposition data-structure, by providing pointers between adjacent vertical trapezoids and maintaining the conflict list sorted (or by using hashing) so that merge operations can be done quickly. In any case, this can be done in linear time in the input/output size involved, as can be verified.

51.1.1.1. Analysis

Claim 51.1.1. *The (amortized) running time of constructing $\mathcal{B}_i$ from $\mathcal{B}_{i-1}$ is proportional to the size of the conflict lists of the vertical trapezoids in $\mathcal{B}_i \setminus \mathcal{B}_{i-1}$ (and the number of such new trapezoids).*

Proof: Observe that we can charge all the work involved in the ith iteration to either the conflict lists of the newly created trapezoids or the deleted conflict lists. Clearly, the running time of the algorithm in the ith iteration is linear in the total size of these conflict lists. Observe that every conflict gets charged twice – when

it is being created and when it is being deleted. As such, the (amortized) running time in the ith iteration is proportional to the total length of the newly created conflict lists. ■

Thus, to bound the running time of the algorithm, it is enough to bound the expected size of the destroyed conflict lists in ith iteration (and sum this bound on the n iterations carried out by the algorithm). Or alternatively, bound the expected size of the conflict lists created in the ith iteration.

Lemma 51.1.2. *Let S be a set of n segments (in general position[1]) with k intersection points. Let S_i be the first i segments in a random permutation of S. The expected size of $\mathcal{B}_i = \mathcal{A}^{\mid}(\mathsf{S}_i)$, denoted by $\tau(i)$ (i.e., the number of trapezoids in $\mathcal{B}_i$), is $O\Big(i + k(i/n)^2\Big)$.*

Proof: Consider[2] an intersection point $\mathsf{p} = s \cap s'$, where $s, s' \in \mathsf{S}$. The probability that p is present in $\mathcal{A}^{\mid}(\mathsf{S}_i)$ is equivalent to the probability that both s and s' are in S_i. This probability is

$$\alpha = \frac{\binom{n-2}{i-2}}{\binom{n}{i}} = \frac{(n-2)!}{(i-2)!\,(n-i)!} \cdot \frac{i!\,(n-i)!}{n!} = \frac{i(i-1)}{n(n-1)}.$$

For each intersection point p in $\mathcal{A}(\mathsf{S})$ define an indicator variable X_{p}, which is 1 if the two segments defining p are in the random sample S_i and 0 otherwise. We have that $\mathbb{E}\Big[X_{\mathsf{p}}\Big] = \alpha$, and as such, by linearity of expectation, the expected number of intersection points in the arrangement $\mathcal{A}(\mathsf{S}_i)$ is

$$\mathbb{E}\left[\sum_{\mathsf{p}\in V} X_{\mathsf{p}}\right] = \sum_{\mathsf{p}\in V} \mathbb{E}\Big[X_{\mathsf{p}}\Big] = \sum_{\mathsf{p}\in V} \alpha = k\alpha,$$

where V is the set of k intersection points of $\mathcal{A}(\mathsf{S})$. Also, every endpoint of a segment of S_i contributed its two endpoints to the arrangement $\mathcal{A}(\mathsf{S}_i)$. Thus, we have that the expected number of vertices in $\mathcal{A}(\mathsf{S}_i)$ is

$$2i + \frac{i(i-1)}{n(n-1)}k.$$

Now, the number of trapezoids in $\mathcal{A}^{\mid}(\mathsf{S}_i)$ is proportional to the number of vertices of $\mathcal{A}(\mathsf{S}_i)$, which implies the claim. ■

51.1.2. Backward analysis

In the following, we would like to consider the total amount of work involved in the ith iteration of the algorithm. The way to analyze these iterations is (conceptually) to run the algorithm for the first i iterations and then run "backward" the last iteration.

So, imagine that the overall size of the conflict lists of the trapezoids of $\mathcal{B}_i$ is W_i and the total size of the conflict lists created only in the ith iteration is C_i.

[1] In this case, no two intersection points of input segments are the same, no two intersection points (or vertices) have the same x-coordinate, no two segments lie on the same line, etc. Making the geometric algorithm work correctly for all degenerate inputs is a huge task that can usually be handled by tedious and careful implementation. Thus, we will always assume general position of the input. In other words, in theory all geometric inputs are inherently good, while in practice they are all evil (as anybody who tried to implement geometric algorithms can testify). The reader is encouraged not to use this to draw any conclusions on the human condition.

[2] The proof is provided in excruciating detail to get the reader used to this kind of argumentation. I would apologize for this pain, but it is a minor trifle, not to be mentioned, when compared to the other offenses in this book.

We are interested in bounding the expected size of C_i, since this is (essentially) the amount of work done by the algorithm in this iteration. Observe that the structure of $\mathcal{B}_i$ is defined independently of the permutation S_i and depends only on the (unordered) set $\mathsf{S}_i = \{\mathsf{s}_1, \ldots, \mathsf{s}_i\}$. So, fix S_i. What is the probability that s_i is a specific segment s of S_i? Clearly, this is $1/i$ since this is the probability of s being the last element in a permutation of the i elements of S_i (i.e., we consider a random permutation of S_i).

Now, consider a trapezoid $\sigma \in \mathcal{B}_i$. If σ was created in the ith iteration, then s_i must be one of the (at most four) segments that define it. Indeed, if s_i is not one of the segments that define σ, then σ existed in the vertical decomposition before s_i was inserted. Since $\mathcal{B}_i$ is independent of the internal ordering of S_i, it follows that $\mathbb{P}[\sigma \in (\mathcal{B}_i \setminus \mathcal{B}_{i-1})] \leq 4/i$. In particular, the overall size of the conflict lists in the end of the ith iteration is

$$W_i = \sum_{\sigma \in \mathcal{B}_i} |\mathrm{cl}(\sigma)|.$$

As such, the expected overall size of the conflict lists created in the ith iteration is

$$\mathbb{E}\Big[C_i \;\Big|\; \mathcal{B}_i\Big] \leq \sum_{\sigma \in \mathcal{B}_i} \frac{4}{i} |\mathrm{cl}(\sigma)| \leq \frac{4}{i} W_i.$$

By Lemma 51.1.2, the expected size of $\mathcal{B}_i$ is $O\big(i + ki^2/n^2\big)$. Let us guess (for the time being) that on average the size of the conflict list of a trapezoid of $\mathcal{B}_i$ is about $O(n/i)$. In particular, assume that we know that

$$\mathbb{E}[W_i] = O\left(\left(i + \frac{i^2}{n^2}k\right)\frac{n}{i}\right) = O\left(n + k\frac{i}{n}\right),$$

by Lemma 51.1.2, implying

$$\mathbb{E}[C_i] = \mathbb{E}\Big[\mathbb{E}[C_i \;\big|\; \mathcal{B}_i]\Big] \leq \mathbb{E}\left[\frac{4}{i} W_i\right] = \frac{4}{i}\,\mathbb{E}[W_i] = O\left(\frac{4}{i}\left(n + \frac{ki}{n}\right)\right) = O\left(\frac{n}{i} + \frac{k}{n}\right), \tag{51.1}$$

using Lemma 11.1.2. In particular, the expected (amortized) amount of work in the ith iteration is proportional to $\mathbb{E}[C_i]$. Thus, the overall expected running time of the algorithm is

$$\mathbb{E}\left[\sum_{i=1}^{n} C_i\right] = \sum_{i=1}^{n} O\left(\frac{n}{i} + \frac{k}{n}\right) = O\big(n \log n + k\big).$$

Theorem 51.1.3. *Given a set S of n segments in the plane with k intersections, one can compute the vertical decomposition of $\mathcal{A}(\mathsf{S})$ in expected $O(n \log n + k)$ time.*

Intuition and discussion. What remains to be seen is how we came up with the guess that the average size of a conflict list of a trapezoid of $\mathcal{B}_i$ is about $O(n/i)$. Note that using ε-nets implies that the bound $O((n/i)\log i)$ holds with constant probability (see Theorem 38.3.4) for all trapezoids in this arrangement. As such, this result is only slightly surprising. To prove this, we present in the next section a "strengthening" of ε-nets to geometric settings.

To get some intuition on how we came up with this guess, consider a set P of n points on the line and a random sample R of i points from P. Let $\widehat{\mathcal{I}}$ be the partition of the real line into (maximal) open intervals by the endpoints of R, such that these intervals do not contain points of R in their interior.

Consider an interval (i.e., a one-dimensional trapezoid) of $\widehat{\mathcal{I}}$. It is intuitively clear that this interval (in expectation) would contain $O(n/i)$ points. Indeed, fix a point x on the real line, and imagine that we pick each point with probability i/n to be in the random sample. The random variable which is the number of points of

Notation	What it means	Example: Vertical decomposition of segments
S $\mathsf{R} \subseteq \mathsf{S}$	Set of n objects Subset of objects	segments
σ	Notation for a region induced by some objects of S	A vertical trapezoid
$D(\sigma) \subseteq \mathsf{S}$	*Defining set* of σ: Minimal set of objects inducing σ.	Subset of segments defining σ. See Figure 51.3.
$K(\sigma) \subseteq \mathsf{S}$	*Stopping set* of σ: All objects in S that prevents σ from being created.	All segments in S that intersects the interior of the vertical trapezoid σ. See Figure 51.3.
d	*Combinatorial dimension*: Max size of defining set.	$d = 4$: Every vertical trapezoid is defined by at most four segments.
$\omega(\sigma)$	$\omega(\sigma) = \lvert K(\sigma)\rvert$: *Weight* of σ.	
$\mathcal{F}(\mathsf{R})$	*Decomposition*: Set of regions defined by R	Set of vertical trapezoids defined by R
$\mathcal{T} = \mathcal{T}(\mathsf{S})$	Set of all possible regions defined by subsets of S	Set of all vertical trapezoids that can be induced by the segments of S.
$\rho_{r,n}(d,k)$	Probability of a region $\sigma \in \mathsf{S}$ to appear in the decomposition of a random sample $\mathsf{R} \subseteq \mathsf{S}$ of size r, where σ is defined by d objects, and its stopping set is of size k.	
$\sigma \in \mathcal{F}(\mathsf{R})$ $\mathcal{F}_{\geq t}(\mathsf{R})$	σ is t-*heavy* if $\omega(\sigma) \geq tn/r$, where $r = \lvert\mathsf{R}\rvert$. Set of all t-heavy regions of $\mathcal{F}(\mathsf{R})$	
$\mathbb{E}f(r)$	$\mathbb{E}f(r) = \mathbb{E}[\lvert\mathcal{F}(\mathsf{R})\rvert]$: Expected complexity of decomposition for sample of size r	
$\mathbb{E}f_{\geq t}(r)$	$\mathbb{E}f_{\geq t}(r) = \mathbb{E}[\lvert\mathcal{F}_{\geq t}(\mathsf{R})\rvert]$: Expected number of regions that are t heavy in the decomposition of a random sample of size r.	

Figure 51.2: Notation used in the analysis.

P we have to scan starting from x and going to the right of x till we "hit" a point that is in the random sample behaves like a geometric variable with probability i/n, and as such its expected value is n/i. The same argument works if we scan P to the left of x. We conclude that the number of points of P in the interval of $\widehat{\mathcal{I}}$ that contains x but does not contain any point of R is $O(n/i)$ in expectation.

Of course, the vertical decomposition case is more involved, as each vertical trapezoid is defined by four input segments. Furthermore, the number of possible vertical trapezoids is larger. Instead of proving the required result for this special case, we will prove a more general result which can be applied in a lot of other settings.

51.2. General settings

51.2.1. Notation

Let S be a set of objects. For a subset $\mathsf{R} \subseteq \mathsf{S}$, we define a collection of 'regions' denoted by $\mathcal{F}(\mathsf{R})$. For the case of vertical decomposition of segments (i.e., Theorem 51.1.3), the objects are segments, the regions are

trapezoids, and $\mathcal{F}(\mathsf{R})$ is the set of vertical trapezoids in $\mathcal{A}^{|}(\mathsf{R})$. Let

$$\mathcal{T} = \mathcal{T}(\mathsf{S}) = \bigcup_{\mathsf{R} \subseteq \mathsf{S}} \mathcal{F}(\mathsf{R})$$

denote the set of ***all possible regions*** defined by subsets of S.

In the vertical trapezoids case, the set $\mathcal{T}$ is the set of all vertical trapezoids that can be defined by any subset of the given input segments.

We associate two subsets $D(\sigma), K(\sigma) \subseteq \mathsf{S}$ with each region $\sigma \in \mathcal{T}$.

The ***defining set*** $D(\sigma)$ of σ is the subset of S defining the region σ (the precise requirements from this set are specified in the axioms below). We assume that for every $\sigma \in \mathcal{T}$, $|D(\sigma)| \leq d$ for a (small) constant d. The constant d is sometime referred to as the ***combinatorial dimension***. In the case of Theorem 51.1.3, each trapezoid σ is defined by at most four segments (or lines) of S that define the region covered by the trapezoid σ, and this set of segments is $D(\sigma)$. See Figure 51.3.

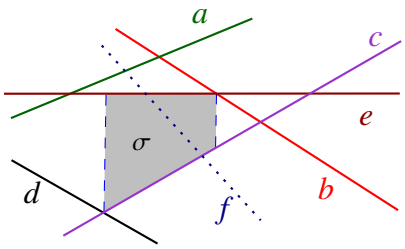


Figure 51.3: $D(\sigma) = \{b, c, d, e\}$ and $K(\sigma) = \{f\}$.

The ***stopping set*** $K(\sigma)$ of σ is the set of objects of S such that including any object of $K(\sigma)$ in R prevents σ from appearing in $\mathcal{F}(\mathsf{R})$. In many applications $K(\sigma)$ is just the set of objects intersecting the cell σ; this is also the case in Theorem 51.1.3, where $K(\sigma)$ is the set of segments of S intersecting the interior of the trapezoid σ (see Figure 51.3). Thus, the stopping set of a region σ, in many cases, is just the conflict list of this region, when it is being created by an RIC algorithm. The ***weight*** of σ is $\omega(\sigma) = |K(\sigma)|$.

Definition 51.2.1 (Framework axioms). Let $\mathsf{S}, \mathcal{F}(\mathsf{R}), D(\sigma)$, and $K(\sigma)$ be such that for any subset $\mathsf{R} \subseteq \mathsf{S}$, the set $\mathcal{F}(\mathsf{R})$ satisfies the following axioms:

(i) For any $\sigma \in \mathcal{F}(\mathsf{R})$, we have $D(\sigma) \subseteq \mathsf{R}$ and $\mathsf{R} \cap K(\sigma) = \emptyset$.

(ii) If $D(\sigma) \subseteq \mathsf{R}$ and $K(\sigma) \cap \mathsf{R} = \emptyset$, then $\sigma \in \mathcal{F}(\mathsf{R})$.

51.2.1.1. Examples of the general framework

(A) **Vertical decomposition.** Discussed above.

(B) **Points on a line.** Let S be a set of n points on the real line. For a set $\mathsf{R} \subseteq \mathsf{S}$, let $\mathcal{F}(\mathsf{R})$ be the set of atomic intervals of the real lines formed by R; that is, the partition of the real line into maximal connected sets (i.e., intervals and rays) that do not contain a point of R in their interior.

Clearly, in this case, an interval $\mathcal{I} \in \mathcal{F}(\mathsf{R})$ the defining set of $\mathcal{I}$ (i.e., $D(\mathcal{I})$) is the set containing the (one or two) endpoints of $\mathcal{I}$ in R. The stopping set of an $\mathcal{I}$ is the set $K(\mathcal{I})$, which is the set of all points of S contained in $\mathcal{I}$.

(C) **Vertices of the convex-hull in 2d.** Consider a set S of n points in the plane. A vertex on the convex hull is defined by the point defining the vertex, and the two edges before and after it on the convex hull. To this end, a ***certified vertex*** of the convex hull (say this vertex is u) is a triplet $(\mathsf{p}, \mathsf{u}, \mathsf{v})$, such that p, u and v are consecutive vertices of $\mathcal{CH}(\mathsf{S})$ (say, in clockwise order). Observe, that computing the convex-hull of S is equivalent to computing the set of certified vertices of S.

For a set $\mathsf{R} \subseteq \mathsf{S}$, let $\mathcal{F}(\mathsf{R})$ denote the set of certified vertices of R (i.e., this is equivalent to the set of vertices of the convex-hull of R. For a certified vertex $\sigma \in \mathcal{F}(\mathsf{R})$, its defining set is the set of three vertices $\mathsf{p}, \mathsf{u}, \mathsf{v}$ that (surprise, surprise) define it. Its stopping set, is the set of all points in S, that either on the "wrong" side of the line spanning pu, or on the "wrong" side of the line spanning uv. Equivalently, $K(\sigma)$ is the set of all points $\mathsf{x} \in \mathsf{S} \setminus \mathsf{R}$, such that the convex-hull of $\mathsf{p}, \mathsf{u}, \mathsf{v}$, and x does not form a convex quadrilateral.

(D) **Edges of the convex-hull in 3d.** Let S be a set of points in three dimensions. An edge e of the convex-hull of a set $\mathsf{R} \subseteq \mathsf{S}$ of points in $\mathbb{R}^3$ is defined by two vertices of S, and it can be certified as being on the convex hull $\mathcal{CH}(\mathsf{R})$, by the two faces f, f' adjacent to e. If all the points of R are on the "right" side of both these two faces then e is an edge of the convex hull of R. Computing all the certified edges of S is equivalent to computing the convex-hull of S.

In the following, assume that each face of any convex-hull of a subset of points of S is a triangle. As such, a face of the convex-hull would be defined by three points. Formally, the ***butterfly*** of an edge e of $\mathcal{CH}(\mathsf{R})$ is $(\mathsf{e}, \mathsf{p}, \mathsf{u})$, where $pnt, \mathsf{u} \in \mathsf{R}$, and such that all the points of R are on the same side as u of the plane spanned by e and p (we have symmetric condition requiring that all the points of S are on the same as p of the plane spanned by e and u).

For a set $\mathsf{R} \subseteq \mathsf{P}$, let $\mathcal{F}(\mathsf{R})$ be its set of butterflies. Clearly, computing all the butterflies of S (i.e., $\mathcal{F}(\mathsf{S})$) is equivalent to computing the convex-hull of S.

For a butterfly $\sigma = (\mathsf{e}, \mathsf{p}, \mathsf{u}) \in \mathcal{F}(\mathsf{R})$ its defining set (i.e., $D(\sigma)$) is a set of four points (i.e., the two points defining its edge e, and the to additional vertices defining the two faces $Face$ and f' adjacent to it). Its stopping set $K(\sigma)$, is the set of all the points of $\mathsf{S} \setminus \mathsf{R}$ that of different sides of the plane spanned by e and p (resp. e and u) than u (resp. p) [here, the stopping set is the union of these two sets].

(E) **Delaunay triangles in 2d.** For a set of S of n points in the plane. Consider a subset $\mathsf{R} \subseteq \mathsf{S}$. A ***Delaunay circle*** of R is a disc D that has three points $\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3$ of R on its boundary, and no points of R in its interior. Naturally, these three points define a ***Delaunay triangle*** $\triangle = \triangle \mathsf{p}_1 \mathsf{p}_2 \mathsf{p}_3$. The defining set is $D(\triangle) = \{\mathsf{p}_1, \mathsf{p}_2, \mathsf{p}_3\}$, and the stopping set $K(\triangle)$ is the set of all points in S that are contained in the interior of the disk D.

51.2.2. Analysis

In the following, S is a set of n objects complying with (i) and (ii) of Definition 51.2.1.

The challenge. What makes the analysis not easy is that there are dependencies between the defining set of a region and its stopping set (i.e., conflict list). In particular, we have the following difficulties

(A) The defining set might be of different sizes depending on the region σ being considered.

(B) Even if all the regions have a defining set of the same size d (say, 4 as in the case of vertical trapezoids), it is not true that every d objects define a valid region. For example, for the case of segments, the four segments might be vertically separated from each other (i.e., think about them as being four disjoint intervals on the real line), and they do not define a vertical trapezoid together. Thus, our analysis is going to be a bit loopy loop – we are going to assume we know how many regions exists (in expectation) for a random sample of certain size, and use this to derive the desired bounds.

51.2.2.1. On the probability of a region to be created

Inherently, to analyze a randomized algorithm using this framework, we will be interested in the probability that a certain region would be created. Thus, let

$$\rho_{r,n}(d, k)$$

denote the probability that a region $\sigma \in \mathcal{T}$ appears in $\mathcal{F}(\mathsf{R})$, where its defining set is of size d, its stopping set is of size k, R is a random sample of size r from a set S, and $n = |\mathsf{S}|$. Specifically, σ is a *feasible* region that might be created by an algorithm computing $\mathcal{F}(\mathsf{R})$.

The sampling model. For describing algorithms it is usually easier to work with samples created by picking a subset of a certain size (without repetition) from the original set of objects. Usually, in the algorithmic applications this would be done by randomly permuting the objects and interpreting a prefix of this permutation as a random sample. Insisting on analyzing this framework in the "right" sampling model creates some non-trivial technical pain.

Lemma 51.2.2. *We have that* $\rho_{r,n}(d,k) \approx \left(1-\frac{r}{n}\right)^k \left(\frac{r}{n}\right)^d$. *Formally,*

$$\frac{1}{2^{2d}}\left(1-4\cdot\frac{r}{n}\right)^k\left(\frac{r}{n}\right)^d \;\le\; \rho_{r,n}(d,k) \;\le\; 2^{2d}\left(1-\frac{1}{2}\cdot\frac{r}{n}\right)^k\left(\frac{r}{n}\right)^d. \tag{51.2}$$

Proof: Let σ be the region under consideration that is defined by d objects and having k stoppers (i.e., $k = K(\sigma)$). We are interested in the probability of σ being created when taking a sample of size r (without repetition) from a set S of n objects. Clearly, this probability is $\rho_{r,n}(d,k) = \binom{n-d-k}{r-d} / \binom{n}{r}$, as we have to pick the d defining objects into the random sample and avoid picking any of the k stoppers. A tedious but careful calculation, delegated to Section 51.4, implies Eq. (51.2).

Instead, here is an elegant argument for why this estimate is correct in a slightly different sampling model. We pick every element of S into the sample R with probability r/n, and this is done independently for each object. In expectation, the random sample is of size r, and clearly the probability that σ is created is the probability that we pick its d defining objects (that is, $(r/n)^d$) multiplied by the probability that we did not pick any of its k stoppers (that is, $(1-r/n)^k$). ■

Remark 51.2.3. The bounds of Eq. (51.2) hold only when r, d, and k are in certain (reasonable) ranges. For the sake of simplicity of exposition we ignore this minor issue. With care, all our arguments work when one pays careful attention to this minor technicality.

51.2.2.2. On exponential decay

For any natural number r and a number $t > 0$, consider R to be a random sample of size r from S without repetition. A region $\sigma \in \mathcal{F}(\mathsf{R})$ as being ***t-heavy*** if $\omega(\sigma) \ge t\cdot\frac{n}{r}$. Let $\mathcal{F}_{\ge t}(\mathsf{R})$ denote all the t-heavy regions of $\mathcal{F}(\mathsf{R})$.[3]

Intuitively, and somewhat incorrectly, we expect the average weight of a region of $\mathcal{F}(\mathsf{R})$ to be roughly n/r. We thus expect the size of this set to drop fast as t increases. Indeed, Lemma 51.2.2 tells us that a trapezoid of weight $t\,(n/r)$ has probability

$$\begin{aligned}\rho_{r,n}\left(d, t\cdot\frac{n}{r}\right) &\approx \left(1-\frac{r}{n}\right)^{t(n/r)}\left(\frac{r}{n}\right)^d \approx \exp(-t)\cdot\left(\frac{r}{n}\right)^d \approx \exp(-t+1)\cdot\left(1-\frac{r}{n}\right)^{n/r}\left(\frac{r}{n}\right)^d \\ &\approx \exp(-t+1)\cdot\rho_{r,n}(d, n/r)\end{aligned}$$

to be created, since $(1-r/n)^{n/r} \approx 1/e$. Namely, a t-heavy region has exponentially lower probability to be created than a region of weight n/r. We next formalize this argument.

Lemma 51.2.4. *Let $r \le n$ and let t be parameters, such that $1 \le t \le r/d$. Furthermore, let R be a sample of size r, and let R' be a sample of size $r' = \lfloor r/t \rfloor$, both from S. Let $\sigma \in \mathcal{T}$ be a region with weight $\omega(\sigma) \ge t\,(n/r)$. Then, $\mathbb{P}[\sigma \in \mathcal{F}(\mathsf{R})] = O\Big(\exp(-t/2)t^d\,\mathbb{P}[\sigma \in \mathcal{F}(\mathsf{R}')]\Big)$.*

[3] These are the regions that are at least t times overweight. Speak about an obesity problem.

Proof: For the sake of simplicity of exposition, assume that $k = \omega(\sigma) = t(n/r)$. By Lemma 51.2.2 (i.e., Eq. (51.2)) we have

$$\frac{\mathbb{P}[\sigma \in \mathcal{F}(\mathsf{R})]}{\mathbb{P}[\sigma \in \mathcal{F}(\mathsf{R}')]} = \frac{\rho_{r,n}(d,k)}{\rho_{r',n}(d,k)} \leq \frac{2^{2d}\Big(1-\frac{1}{2}\cdot\frac{r}{n}\Big)^k\Big(\frac{r}{n}\Big)^d}{\frac{1}{2^{2d}}\Big(1-4\frac{r'}{n}\Big)^k\Big(\frac{r'}{n}\Big)^d} \leq 2^{4d}\exp\!\left(-\frac{kr}{2n}\right)\!\left(1+8\frac{r'}{n}\right)^k\!\left(\frac{r}{r'}\right)^d$$
$$\leq 2^{4d}\exp\!\left(8\frac{kr'}{n}-\frac{kr}{2n}\right)\!\left(\frac{r}{r'}\right)^d = 2^{4d}\exp\!\left(8\frac{tn\lfloor r/t\rfloor}{nr}-\frac{tnr}{2nr}\right)\!\left(\frac{r}{\lfloor r/t\rfloor}\right)^d = O\Big(\exp(-t/2)t^d\Big),$$

since $1/(1-x) \leq 1+2x$ for $x \leq 1/2$ and $1+y \leq \exp(y)$, for all y. (The constant in the above $O(\cdot)$ depends exponentially on d.) ■

Let

$$\mathbb{E}f(r) = \mathbb{E}[|\mathcal{F}(\mathsf{R})|] \quad \text{and} \quad \mathbb{E}f_{\geq t}(r) = \mathbb{E}[|\mathcal{F}_{\geq t}(\mathsf{R})|]\,,$$

where the expectation is over random subsets $\mathsf{R} \subseteq \mathsf{S}$ of size r. Note that $\mathbb{E}f(r) = \mathbb{E}f_{\geq 0}(r)$ is the expected number of regions created by a random sample of size r. In words, $\mathbb{E}f_{\geq t}(r)$ is the expected number of regions in a structure created by a sample of r random objects, such that these regions have weight which is t times larger than the "expected" weight (i.e., n/r). In the following, we assume that $\mathbb{E}f(r)$ is a monotone increasing function.

Lemma 51.2.5 (The exponential decay lemma). *Given a set S of n objects and parameters $r \leq n$ and $1 \leq t \leq r/d$, where $d = \max_{\sigma\in\mathcal{T}(\mathsf{S})}|D(\sigma)|$, if axioms (i) and (ii) above hold for any subset of S, then*

$$\mathbb{E}f_{\geq t}(r) = O(t^d \exp(-t/2)\,\mathbb{E}f(r)). \tag{51.3}$$

Proof: Let R be a random sample of size r from S and let R' be a random sample of size $r' = \lfloor r/t\rfloor$ from S. Let $H = \bigcup_{X\subseteq\mathsf{S},|X|=r}\mathcal{F}_{\geq t}(X)$ denote the set of all t-heavy regions that might be created by a sample of size r. In the following, the expectation is taken over the content of the random samples R and R'.

For a region σ, let X_σ be the indicator variable that is 1 if and only if $\sigma \in \mathcal{F}(\mathsf{R})$. By linearity of expectation and since $\mathbb{E}[X_\sigma] = \mathbb{P}[\sigma \in \mathcal{F}(\mathsf{R})]$, we have

$$\begin{aligned}\mathbb{E}f_{\geq t}(r) &= \mathbb{E}[|\mathcal{F}_{\geq t}(\mathsf{R})|] = \mathbb{E}\Big[\sum_{\sigma\in H} X_\sigma\Big] = \sum_{\sigma\in H}\mathbb{E}[X_\sigma] = \sum_{\sigma\in H}\mathbb{P}[\sigma\in\mathcal{F}(\mathsf{R})]\\ &= O\Big(t^d\exp(-t/2)\sum_{\sigma\in H}\mathbb{P}[\sigma\in\mathcal{F}(\mathsf{R}')]\Big) = O\Big(t^d\exp(-t/2)\sum_{\sigma\in\mathcal{T}}\mathbb{P}[\sigma\in\mathcal{F}(\mathsf{R}')]\Big)\\ &= O\Big(t^d\exp(-t/2)\,\mathbb{E}f(r')\Big) = O\Big(t^d\exp(-t/2)\,\mathbb{E}f(r)\Big),\end{aligned}$$

by Lemma 51.2.4 and since $\mathbb{E}f(r)$ is a monotone increasing function. ■

51.2.2.3. Bounding the moments

Consider a different randomized algorithm that in a first round samples r objects, $\mathsf{R} \subseteq \mathsf{S}$ (say, segments), computes the arrangement induced by these r objects (i.e., $\mathcal{A}^{|}(\mathsf{R})$), and then inside each region σ it computes the arrangement of the $\omega(\sigma)$ objects intersecting the interior of this region, using an algorithm that takes $O(\omega(\sigma)^c)$ time, where $c > 0$ is some fixed constant. The overall expected running time of this algorithm is

$$\mathbb{E}\Big[\sum_{\sigma\in\mathcal{F}(\mathsf{R})}(\omega(\sigma))^c\Big].$$

We are now able to bound this quantity.

Theorem 51.2.6 (Bounded moments theorem). *Let* $\mathsf{R} \subseteq \mathsf{S}$ *be a random subset of size* r*. Let* $\mathbb{E}f(r) = \mathbb{E}[|\mathcal{F}(\mathsf{R})|]$ *and let* $c \geq 1$ *be an arbitrary constant. Then,*

$$\mathbb{E}\Big[\sum_{\sigma \in \mathcal{F}(\mathsf{R})} (\omega(\sigma))^c\Big] = O\Big(\mathbb{E}f(r)\left(\frac{n}{r}\right)^c\Big).$$

Proof: Let $\mathsf{R} \subseteq \mathsf{S}$ be a random sample of size r. Observe that all the regions with weight in the range $\left[(t-1)\frac{n}{r}\,,\, t\cdot\frac{n}{r}\right)$ are in the set $\mathcal{F}_{\geq t-1}(\mathsf{R}) \setminus \mathcal{F}_{\geq t}(\mathsf{R})$. As such, we have by Lemma 51.2.5 that

$$\begin{aligned}
W &= \mathbb{E}\Big[\sum_{\sigma\in\mathcal{F}(\mathsf{R})} \omega(\sigma)^c\Big] \leq \mathbb{E}\Big[\sum_{t\geq 1}\Big(t\frac{n}{r}\Big)^c (|\mathcal{F}_{\geq t-1}(\mathsf{R})| - |\mathcal{F}_{\geq t}(\mathsf{R})|\,)\Big] \leq \mathbb{E}\Big[\sum_{t\geq 1}\Big(t\frac{n}{r}\Big)^c |\mathcal{F}_{\geq t-1}(\mathsf{R})|\,\Big] \\
&\leq \left(\frac{n}{r}\right)^c \sum_{t\geq 0}(t+1)^c\,\mathbb{E}[|\mathcal{F}_{\geq t}(\mathsf{R})|] = \left(\frac{n}{r}\right)^c \sum_{t\geq 0}(t+1)^c\,\mathbb{E}f_{\geq t}(r) = \left(\frac{n}{r}\right)^c \sum_{t\geq 0} O\Big((t+1)^{c+d}\,\exp(-t/2)\,\mathbb{E}f(r)\Big) \\
&= O\Big(\mathbb{E}f(r)\left(\frac{n}{r}\right)^c \sum_{t\geq 0}(t+1)^{c+d}\,\exp(-t/2)\Big) = O\Big(\mathbb{E}f(r)\Big(\frac{n}{r}\Big)^c\Big),
\end{aligned}$$

since c and d are both constants. ■

51.3. Applications

51.3.1. Analyzing the RIC algorithm for vertical decomposition

We remind the reader that the input of the algorithm of Section 51.1.2 is a set S of n segments with k intersections, and it uses randomized incremental construction to compute the vertical decomposition of the arrangement $\mathcal{A}(\mathsf{S})$.

Lemma 51.1.2 shows that the number of vertical trapezoids in the randomized incremental construction is in expectation $\mathbb{E}f(i) = O\big(i + k\,(i/n)^2\big)$. Thus, by Theorem 51.2.6 (used with $c = 1$), we have that the total expected size of the conflict lists of the vertical decomposition computed in the ith step is

$$\mathbb{E}[W_i] = \mathbb{E}\Big[\sum_{\sigma\in\mathcal{B}_i}\omega(\sigma)\Big] = O\Big(\mathbb{E}f(i)\frac{n}{i}\Big) = O\Big(n + k\frac{i}{n}\Big).$$

This is the missing piece in the analysis of Section 51.1.2. Indeed, the amortized work in the ith step of the algorithm is $O(W_i/i)$ (see Eq. (51.1)), and as such, the expected running time of this algorithm is

$$\mathbb{E}\Big[O\Big(\sum_{i=1}^{n}\frac{W_i}{i}\Big)\Big] = O\Big(\sum_{i=1}^{n}\frac{1}{i}\Big(n + k\frac{i}{n}\Big)\Big) = O(n\log n + k).$$

This implies Theorem 51.1.3.

51.3.2. Cuttings

Let S be a set of n lines in the plane, and let r be an arbitrary parameter. A $(1/r)$-***cutting*** of S is a partition of the plane into constant complexity regions such that each region intersects at most n/r lines of S. It is natural to try to minimize the number of regions in the cutting, as cuttings are a natural tool for performing "divide and conquer".

A neat proof of the existence of suboptimal cuttings follows readily from the exponential decay lemma.

Lemma 51.3.1. *Let* S *be a set of* n *segments in the plane, and let* R *be a random sample from* S *of size* $\ell = cr \ln r$*, where* c *is a sufficiently large constant. Then, with probability* $\geq 1 - 1/r^{O(1)}$*, the vertical decomposition of* R *is a cutting of size* $O(r^2 \log^2 r)$*.*

Proof: In our case, the vertical decomposition complexity $\mathbb{E}f(\ell) = O(\ell^2)$ – as ℓ segments have at most $\binom{\ell}{2}$ intersections. For $t = c \ln r$, a vertical trapezoid σ in $\mathcal{A}^{\mid}(\mathsf{R})$ is bad if $\omega(\sigma) > r = t(n/\ell)$. But such a trapezoid is t-heavy. Let X be the random variable that is the number of bad trapezoids in $\mathcal{A}^{\mid}(\mathsf{R})$. The exponential decay lemma (Lemma 51.2.5) states that

$$\mathbb{E}[X] = \mathbb{E}f_{\geq t}(\ell) = O(t^2 \exp(-t/2)\, \mathbb{E}f(\ell)) = O((c \ln r)^2 \exp(-c \ln r/2)\, \ell^2) = O\Big((\ln^2 r) r^{-c/2} (r \log r)^2\Big) < \frac{1}{r^{c/4}},$$

if c is sufficiently large. As such, we have $\mathbb{P}[X \geq 1] \leq 1/r^{c/4}$ by Markov's inequality. ■

We provide an alternative proof to the above using the ε-net theorem.

Lemma 51.3.2. *There exists a* $(1/r)$*-cutting of a set of lines* S *in the plane of size* $O\Big((r \log r)^2\Big)$*.*

Proof: Consider the range space having S as its ground set and vertical trapezoids as its ranges (i.e., given a vertical trapezoid σ, its corresponding range is the set of all lines of S that intersect the interior of σ). This range space has a VC dimension which is a constant as can be easily verified. Let $X \subseteq \mathsf{S}$ be an ε-net for this range space, for $\varepsilon = 1/r$. By Theorem 38.3.4 (ε-net theorem), there exists such an ε-net X of this range space, of size $O((1/\varepsilon)\log(1/\varepsilon)) = O(r \log r)$. In fact, Theorem 38.3.4 states that an appropriate random sample is an ε-net with non-zero probability, which implies, by the probabilistic method, that such a net (of this size) exists.

Consider the vertical decomposition $\mathcal{A}^{\mid}(X)$, where X is as above. We claim that this collection of trapezoids is the desired cutting.

The bound on the size is immediate, as the complexity of $\mathcal{A}^{\mid}(X)$ is $O\Big(|X|^2\Big)$ and $|X| = O(r \log r)$.

As for correctness, consider a vertical trapezoid σ in the arrangement $\mathcal{A}^{\mid}(X)$. It does not intersect any of the lines of X in its interior, since it is a trapezoid in the vertical decomposition $\mathcal{A}^{\mid}(X)$. Now, if σ intersected more than n/r lines of S in its interior, where $n = |\mathsf{S}|$, then it must be that the interior of σ intersects one of the lines of X, since X is an ε-net for S, a contradiction.

It follows that σ intersects at most $\varepsilon n = n/r$ lines of S in its interior. ■

Claim 51.3.3. *Any* $(1/r)$*-cutting in the plane of* n *lines contains at least* $\Omega\Big(r^2\Big)$ *regions.*

Proof: An arrangement of n lines (in general position) has $M = \binom{n}{2}$ intersections. However, the number of intersections of the lines intersecting a single region in the cutting is at most $m = \binom{n/r}{2}$. This implies that any cutting must be of size at least $M/m = \Omega\Big(n^2/(n/r)^2\Big) = \Omega\Big(r^2\Big)$. ■

We can get cuttings of size matching the above lower bound using the moments technique.

Theorem 51.3.4. *Let* S *be a set of* n *lines in the plane, and let* r *be a parameter. One can compute a* $(1/r)$*-cutting of* S *of size* $O(r^2)$*.*

Proof: Let $\mathsf{R} \subseteq \mathsf{S}$ be a random sample of size r, and consider its vertical decomposition $\mathcal{A}^{\mid}(\mathsf{R})$. If a vertical trapezoid $\sigma \in \mathcal{A}^{\mid}(\mathsf{R})$ intersects at most n/r lines of S, then we can add it to the output cutting. The other possibility is that a σ intersects $t(n/r)$ lines of S, for some $t > 1$, and let $\mathrm{cl}(\sigma) \subset \mathsf{S}$ be the conflict list of σ (i.e., the list of lines of S that intersect the interior of σ). Clearly, a $(1/t)$-cutting for the set $\mathrm{cl}(\sigma)$ forms a vertical

decomposition (clipped inside σ) such that each trapezoid in this cutting intersects at most n/r lines of S. Thus, we compute such a cutting inside each such "heavy" trapezoid using the algorithm (implicit in the proof) of Lemma 51.3.2, and these subtrapezoids to the resulting cutting. Clearly, the size of the resulting cutting inside σ is $O\left(t^2 \log^2 t\right) = O\left(t^4\right)$. The resulting two-level partition is clearly the required cutting. By Theorem 51.2.6, the expected size of the cutting is

$$O\left(\mathbb{E}f(r) + \mathbb{E}\Big[\sum_{\sigma\in\mathcal{F}(\mathsf{R})} \left(2\frac{\omega(\sigma)}{n/r}\right)^4\Big]\right) = O\left(\mathbb{E}f(r) + \left(\frac{r}{n}\right)^4 \mathbb{E}\Big[\sum_{\sigma\in\mathcal{F}(\mathsf{R})} (\omega(\sigma))^4\Big]\right)$$
$$= O\left(\mathbb{E}f(r) + \left(\frac{r}{n}\right)^4 \mathbb{E}f(r)\left(\frac{n}{r}\right)^4\right) = O(\mathbb{E}f(r)) = O\left(r^2\right),$$

since $\mathbb{E}f(r)$ is proportional to the complexity of $\mathcal{A}(\mathsf{R})$ which is $O(r^2)$. ■

51.4. Bounds on the probability of a region to be created

Here we prove Lemma 51.2.2 in the "right" sampling model. The casual reader is encouraged to skip this section, as it contains mostly tedious (and not very insightful) calculations.

Let S be a given set of n objects. Let $\rho_{r,n}(d,k)$ be the probability that a region $\sigma \in \mathcal{T}$ whose defining set is of size d and whose stopping set is of size k appears in $\mathcal{F}(\mathsf{R})$, where R is a random sample from S of size r (without repetition).

Lemma 51.4.1. *We have* $\rho_{r,n}(d,k) = \dfrac{\binom{n-d-k}{r-d}}{\binom{n}{r}} = \dfrac{\binom{n-d-k}{r-d}}{\binom{n}{r-d}} \cdot \dfrac{\binom{r}{d}}{\binom{n-(r-d)}{d}} = \dfrac{\binom{n-d-k}{r-d}}{\binom{n-d}{r-d}} \cdot \dfrac{\binom{r}{d}}{\binom{n}{d}}$.

Proof: So, consider a region σ with d defining objects in $D(\sigma)$ and k detractors in $K(\sigma)$. We have to pick the d defining objects of $D(\sigma)$ to be in the random sample R of size r but avoid picking any of the k objects of $K(\sigma)$ to be in R.

The second part follows since $\binom{n}{r} = \binom{n}{r-d}\binom{n-(r-d)}{d} / \binom{r}{d}$. Indeed, for the right-hand side first pick a sample of size $r-d$ and then a sample of size d from the remaining objects. Merging the two random samples, we get a random sample of size r. However, since we do not care if an object is in the first sample or second sample, we observe that every such random sample is being counted $\binom{r}{d}$ times.

The third part is easier, as it follows from $\binom{n}{r-d}\binom{n-(r-d)}{d} = \binom{n}{d}\binom{n-d}{r-d}$. The two sides count the different ways to pick two subsets from a set of size n, the first one of size d and the second one of size $r-d$. ■

Lemma 51.4.2. *For $M \geq m \geq t \geq 0$, we have* $\left(\dfrac{m-t}{M-t}\right)^t \leq \dfrac{\binom{m}{t}}{\binom{M}{t}} \leq \left(\dfrac{m}{M}\right)^t$.

Proof: We have that $\alpha = \dfrac{\binom{m}{t}}{\binom{M}{t}} = \dfrac{m!}{(m-t)!t!}\dfrac{(M-t)!t!}{M!} = \dfrac{m}{M}\cdot\dfrac{m-1}{M-1}\cdots\dfrac{m-t+1}{M-t+1}$. Now, since $M \geq m$, we have that $\dfrac{m-i}{M-i} \leq \dfrac{m}{M}$, for all $i \geq 0$. As such, the maximum (resp. minimum) fraction on the right-hand size is m/M (resp. $\frac{m-t+1}{M-t+1}$). As such, we have $\left(\frac{m-t}{M-t}\right)^t \leq \left(\frac{m-t+1}{M-t+1}\right)^t \leq \alpha \leq (m/M)^t$. ■

Lemma 51.4.3. *Let $0 \leq X, Y \leq N$. We have that* $\left(1 - \frac{X}{N}\right)^Y \leq \left(1 - \frac{Y}{2N}\right)^X$.

Proof: Since $1 - \alpha \leq \exp(-\alpha) \leq (1 - \alpha/2)$, for $0 \leq \alpha \leq 1$, it follows that

$$\left(1 - \frac{X}{N}\right)^Y \leq \exp\left(-\frac{XY}{N}\right) = \left(\exp\left(-\frac{Y}{n}\right)\right)^X \leq \left(1 - \frac{Y}{2n}\right)^X. \qquad \blacksquare$$

Lemma 51.4.4. *For $2d \leq r \leq n/8$ and $k \leq n/2$, we have that*

$$\frac{1}{2^{2d}}\left(1 - 4 \cdot \frac{r}{n}\right)^k \left(\frac{r}{n}\right)^d \leq \rho_{r,n}(d,k) \leq 2^{2d}\left(1 - \frac{1}{2} \cdot \frac{r}{n}\right)^k \left(\frac{r}{n}\right)^d.$$

Proof: By Lemma 51.4.1, Lemma 51.4.2, and Lemma 51.4.3 we have

$$\begin{aligned}\rho_{r,n}(d,k) &= \frac{\binom{n-d-k}{r-d}}{\binom{n-d}{r-d}} \cdot \frac{\binom{r}{d}}{\binom{n}{d}} \leq \left(\frac{n-d-k}{n-d}\right)^{r-d}\left(\frac{r}{n}\right)^d \leq \left(1 - \frac{k}{n}\right)^{r-d}\left(\frac{r}{n}\right)^d \leq 2^d\left(1 - \frac{k}{n}\right)^r \left(\frac{r}{n}\right)^d \\ &\leq 2^d\left(1 - \frac{r}{2n}\right)^k \left(\frac{r}{n}\right)^d,\end{aligned}$$

since $k \leq n/2$. As for the other direction, by similar argumentation, we have

$$\begin{aligned}\rho_{r,n}(d,k) &= \frac{\binom{n-d-k}{r-d}}{\binom{n}{r-d}} \cdot \frac{\binom{r}{d}}{\binom{n-(r-d)}{d}} \geq \left(\frac{n-d-k-(r-d)}{n-(r-d)}\right)^{r-d}\left(\frac{r-d}{n-(r-d)-d}\right)^d \\ &= \left(1 - \frac{d+k}{n-(r-d)}\right)^{r-d}\left(\frac{r-d}{n-r}\right)^d \geq \left(1 - \frac{d+k}{n/2}\right)^r \left(\frac{r/2}{n}\right)^d \\ &\geq \frac{1}{2^d}\left(1 - \frac{4r}{n}\right)^{d+k}\left(\frac{r}{n}\right)^d \geq \frac{1}{2^{2d}}\left(1 - \frac{4r}{n}\right)^k \left(\frac{r}{n}\right)^d,\end{aligned}$$

by Lemma 51.4.3 (setting $N = n/4$, $X = r$, and $Y = d + k$) and since $r \geq 2d$ and $4r/n \leq 1/2$. $\blacksquare$

51.5. Bibliographical notes

The technique described in this chapter is generally attributed to the work by Clarkson and Shor [CS89], which is historically inaccurate as the technique was developed by Clarkson [Cla88]. Instead of mildly confusing the matter by referring to it as the Clarkson technique, we decided to make sure to really confuse the reader and refer to it as the ***moments technique***. The Clarkson technique [Cla88] is in fact more general and implies a connection between the number of "heavy" regions and "light" regions. The general framework can be traced back to the earlier paper [Cla87]. This implies several beautiful results, some of which we cover later in the book.

For the full details of the algorithm of Section 51.1, the interested reader is refereed to the books [BCKO08, BY98]. Interestingly, in some cases the merging stage can be skipped; see [Har00a].

Agarwal *et al.* [AMS98] presented a slightly stronger variant than the original version of Clarkson [Cla88] that allows a region to disappear even if none of the members of its stopping set are in the random sample. This stronger setting is used in computing the vertical decomposition of a single face in an arrangement (instead of the whole arrangement). Here an insertion of a faraway segment of the random sample might cut off a portion of the face of interest. In particular, in the settings of Agarwal *et al.* Axiom (ii) is replaced by the following:

(ii) If $\sigma \in \mathcal{F}(\mathsf{R})$ and R' is a subset of R with $D(\sigma) \subseteq \mathsf{R}'$, then $\sigma \in \mathcal{F}(\mathsf{R}')$.

Interestingly, Clarkson [Cla88] did not prove Theorem 51.2.6 using the exponential decay lemma but gave a direct proof. In fact, his proof implicitly contains the exponential decay lemma. We chose the current exposition since it is more modular and provides a better intuition of what is really going on and is hopefully slightly simpler. In particular, Lemma 51.2.2 is inspired by the work of Sharir [Sha03].

The exponential decay lemma (Lemma 51.2.5) was proved by Chazelle and Friedman [CF90]. The work of Agarwal *et al.* [AMS98] is a further extension of this result. Another analysis was provided by Clarkson *et al.* [CMS93].

Another way to reach similar results is using the technique of Mulmuley [Mul94], which relies on a direct analysis on 'stoppers' and 'triggers'. This technique is somewhat less convenient to use but is applicable to some settings where the moments technique does not apply directly. Also, his concept of the omega function might explain why randomized incremental algorithms perform better in practice than their worst case analysis [Mul89].

Backwards analysis in geometric settings was first used by Chew [Che86] and was formalized by Seidel [Sei93]. It is similar to the "leave one out" argument used in statistics for cross validation. The basic idea was probably known to the Greeks (or Russians or French) at some point in time.

(Naturally, our summary of the development is cursory at best and not necessarily accurate, and all possible disclaimers apply. A good summary is provided in the introduction of [Sei93].)

Sampling model. As a rule of thumb all the different sampling approaches are similar and yield similar results. For example, we used such an alternative sampling approach in the "proof" of Lemma 51.2.2. It is a good idea to use whichever sampling scheme is the easiest to analyze in figuring out what's going on. Of course, a formal proof requires analyzing the algorithm in the sampling model its uses.

Lazy randomized incremental construction. If one wants to compute a single face that contains a marking point in an arrangement of curves, then the problem in using randomized incremental construction is that as you add curves, the region of interest shrinks, and regions that were maintained should be ignored. One option is to perform flooding in the vertical decomposition to figure out what trapezoids are still reachable from the marking point and maintaining only these trapezoids in the conflict graph. Doing it in each iteration is way too expensive, but luckily one can use a lazy strategy that performs this cleanup only a logarithmic number of times (i.e., you perform a cleanup in an iteration if the iteration number is, say, a power of 2). This strategy complicates the analysis a bit; see [BDS95] for more details on this ***lazy randomized incremental construction*** technique. An alternative technique was suggested by the author for the (more restricted) case of planar arrangements; see [Har00b]. The idea is to compute only what the algorithm really needs to compute the output, by computing the vertical decomposition in an exploratory online fashion. The details are unfortunately overwhelming although the algorithm seems to perform quite well in practice.

Cuttings. The concept of cuttings was introduced by Clarkson. The first optimal size cuttings were constructed by Chazelle and Friedman [CF90], who proved the exponential decay lemma to this end. Our elegant proof follows the presentation by de Berg and Schwarzkopf [BS95]. The problem with this approach is that the constant involved in the cutting size is awful[4]. Matoušek [Mat98] showed that there $(1/r)$-cuttings with $8r^2 + 6r + 4$ trapezoids, by using level approximation. A different approach was taken by the author [Har00a], who showed how to get cuttings which seem to be quite small (i.e., constant-wise) in practice. The basic idea is to do randomized incremental construction but at each iteration greedily add all the trapezoids with conflict list small enough to the cutting being output. One can prove that this algorithm also generates $O(r^2)$ cuttings,

[4] This is why all computations related to cuttings should be done on a waiter's bill pad. As Douglas Adams put it: "On a waiter's bill pad, reality and unreality collide on such a fundamental level that each becomes the other and anything is possible, within certain parameters."

but the details are not trivial as the framework described in this chapter is not applicable for analyzing this algorithm.

Cuttings also can be computed in higher dimensions for hyperplanes. In the plane, cuttings can also be computed for well-behaved curves; see [SA95].

Another fascinating concept is ***shallow cuttings***. These are cuttings covering only portions of the arrangement that are in the "bottom" of the arrangement. Matoušek came up with the concept [Mat92]. See [AES99, CCH09] for extensions and applications of shallow cuttings.

Even more on randomized algorithms in geometry. We have only scratched the surface of this fascinating topic, which is one of the cornerstones of "modern" computational geometry. The interested reader should have a look at the books by Mulmuley [Mul94], Sharir and Agarwal [SA95], Matoušek [Mat02], and Boissonnat and Yvinec [BY98].

51.6. Exercises

Exercise 51.6.1 (Convex hulls incrementally). Let P be a set of n points in the plane.

(A) Describe a randomized incremental algorithm for computing the convex hull $\mathcal{CH}(\mathsf{P})$. Bound the expected running time of your algorithm.

(B) Assume that for any subset of P, its convex hull has complexity t (i.e., the convex hull of the subset has t edges). What is the expected running time of your algorithm in this case? If your algorithm is not faster for this case (for example, think about the case where $t = O(\log n)$), describe a variant of your algorithm which is faster for this case.

Exercise 51.6.2 (Compressed quadtree made incremental). Given a set P of n points in $\mathbb{R}^d$, describe a randomized incremental algorithm for building a compressed quadtree for P that works in expected $O(dn \log n)$ time. Prove the bound on the running time of your algorithm.

References

[AES99] P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM J. Comput.*, 29: 912–953, 1999.

[AMS98] P. K. Agarwal, J. Matoušek, and O. Schwarzkopf. Computing many faces in arrangements of lines and segments. *SIAM J. Comput.*, 27(2): 491–505, 1998.

[BCKO08] M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars. *Computational Geometry: Algorithms and Applications*. 3rd. Santa Clara, CA, USA: Springer, 2008.

[BDS95] M. de Berg, K. Dobrindt, and O. Schwarzkopf. On lazy randomized incremental construction. *Discrete Comput. Geom.*, 14: 261–286, 1995.

[BS95] M. de Berg and O. Schwarzkopf. Cuttings and applications. *Int. J. Comput. Geom. Appl.*, 5: 343–355, 1995.

[BY98] J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, 1998.

[CCH09] C. Chekuri, K. L. Clarkson., and S. Har-Peled. On the set multi-cover problem in geometric settings. *Proc. 25th Annu. Sympos. Comput. Geom.* (SoCG), 341–350, 2009.

[CF90] B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3): 229–249, 1990.

[Che86] L. P. Chew. *Building Voronoi diagrams for convex polygons in linear expected time*. Technical Report PCS-TR90-147. Hanover, NH: Dept. Math. Comput. Sci., Dartmouth College, 1986.

[Cla87] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2: 195–222, 1987.

[Cla88] K. L. Clarkson. Applications of random sampling in computational geometry, II. *Proc. 4th Annu. Sympos. Comput. Geom.* (SoCG), 1–11, 1988.

[CMS93] K. L. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. *Comput. Geom. Theory Appl.*, 3(4): 185–212, 1993.

[CS89] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4(5): 387–421, 1989.

[Har00a] S. Har-Peled. Constructing planar cuttings in theory and practice. *SIAM J. Comput.*, 29(6): 2016–2039, 2000.

[Har00b] S. Har-Peled. Taking a walk in a planar arrangement. *SIAM J. Comput.*, 30(4): 1341–1367, 2000.

[Mat02] J. Matoušek. *Lectures on Discrete Geometry*. Vol. 212. Grad. Text in Math. Springer, 2002.

[Mat92] J. Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2(3): 169–186, 1992.

[Mat98] J. Matoušek. On constants for cuttings in the plane. *Discrete Comput. Geom.*, 20: 427–448, 1998.

[Mul89] K. Mulmuley. An efficient algorithm for hidden surface removal. *Comput. Graph.*, 23(3): 379–388, 1989.

[Mul94] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Englewood Cliffs, NJ: Prentice Hall, 1994.

[SA95] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. New York: Cambridge University Press, 1995.

[Sei93] R. Seidel. Backwards analysis of randomized geometric algorithms. *New Trends in Discrete and Computational Geometry*. Ed. by J. Pach. Vol. 10. Algorithms and Combinatorics. Springer-Verlag, 1993, pp. 37–68.

[Sha03] M. Sharir. The Clarkson-Shor technique revisited and extended. *Comb., Prob. & Comput.*, 12(2): 191–201, 2003.

Chapter 52

Primality testing

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

"The world is what it is; men who are nothing, who allow themselves to become nothing, have no place in it."
— Bend in the river, V.S. Naipaul

Introduction – how to read this write-up

In this note, we present a simple randomized algorithms for primality testing. The challenge is that it requires a non-trivial amount of number theory, which is not the purpose of this course. Nevertheless, this note is more or less self contained, and all necessary background is provided (assuming some basic mathematical familiarity with groups, fields and modulo arithmetic). It is however not really necessary to understand all the number theory material needed, and the reader can take it as given. In particular, I recommend to read the number theory background part without reading all of the proofs (at least on first reading). Naturally, a complete and total understanding of this material one needs to read everything carefully.

The description of the primality testing algorithm in this write-up is not minimal – there are shorter descriptions out there. However, it is modular – assuming the number theory machinery used is correct, the algorithm description is relatively straightforward.

52.1. Number theory background

52.1.1. Modulo arithmetic

52.1.1.1. Prime and coprime

For integer numbers x and y, let $x \mid y$ denotes that x divides y. The *greatest common divisor* (***gcd***) of two numbers x and y, denoted by $\gcd(x, y)$, is the largest integer that divides both x and y. The *least common multiple* (***lcm***) of x and y, denoted by $\operatorname{lcm}(x, y) = xy/\gcd(x, y)$, is the smallest integer α, such that $x \mid \alpha$ and $y \mid \alpha$. An integer number $p > 0$ is ***prime*** if it is divisible only by 1 and itself (we will consider 1 not to be prime).

Some standard definitions:

$$\begin{aligned} x, y \text{ are } \textbf{\textit{coprime}} &\iff \gcd(x, y) = 1, \\ \textbf{\textit{quotient}} \text{ of } x/y &\iff x \operatorname{div} y = \lfloor x/y \rfloor, \end{aligned}$$

$$\textbf{\textit{remainder}} \text{ of } x/y \qquad \Longleftrightarrow \qquad x \bmod y = x - y\lfloor x/y \rfloor .$$

The remainder $x \bmod y$ is sometimes referred to as ***residue***.

52.1.1.2. Computing gcd

Computing the gcd of two numbers is a classical algorithm, see code on the right – proving that it indeed returns the right result follows by an easy induction. It is easy to verify that if the input is made out of $\log n$ bits, then this algorithm takes $O(\text{poly}(\log n))$ time (i.e., it is polynomial in the input size). Indeed, doing basic operations on numbers (i.e., multiplication, division, addition, subtraction, etc) with total of ℓ bits takes $O(\ell^2)$ time (naively – faster algorithms are known).

```
EuclidGCD(a, b):
    if (b = 0)
        return a
    else
        return EuclidGCD(b, a mod b)
```

Exercise 52.1.1. Show that $\gcd(F_n, F_{n-1}) = 1$, where F_i is the ith Fibonacci number. Argue that for two consecutive Fibonacci numbers **EuclidGCD**(F_n, F_{n-1}) takes $O(n)$ time, if every operation takes $O(1)$ time.

Lemma 52.1.2. *For all $\alpha, \beta > 0$ integers, there are integer numbers x and y, such that $\gcd(\alpha,\beta) = \alpha x + \beta y$, which can be computed in polynomial time; that is, $O(\text{poly}(\log \alpha + \log \beta))$.*

Proof: If $\alpha = \beta$ then the claim trivially holds. Otherwise, assume that $\alpha > \beta$ (otherwise, swap them), and observe that $\gcd(\alpha,\beta) = \gcd(\alpha \bmod \beta, \beta)$. In particular, by induction, there are integers x', y', such that $\gcd(\alpha \bmod \beta, \beta) = x'(\alpha \bmod \beta) + y'\beta$. However, $\tau = \alpha \bmod \beta = \alpha - \beta\lfloor \alpha/\beta \rfloor$. As such, we have

$$\gcd(\alpha,\beta) = \gcd(\alpha \bmod \beta, \beta) = x'\left(\alpha - \beta\lfloor \alpha/\beta \rfloor\right) + y'\beta = x'\alpha + (y' - \beta\lfloor \alpha/\beta \rfloor)\beta,$$

as claimed. The running time follows immediately by modifying **EuclidGCD** to compute these numbers. ■

We use $\alpha \equiv \beta \pmod n$ or $\alpha \equiv_n \beta$ to denote that α and β are ***congruent modulo*** n; that is $\alpha \bmod n = \beta \bmod n$. Or put differently, we have $n \mid (\alpha - \beta)$. The set $\mathbb{Z}_n = \{0, \ldots, n-1\}$ form a *group* under addition modulo n (see Definition 52.1.9$_{\text{p338}}$ for a formal definition of a group). The more interesting creature is $\mathbb{Z}_n^* = \left\{ x \;\middle|\; x \in \{1, \ldots, n\}, x > 0, \text{ and } \gcd(x, n) = 1 \right\}$, which is a ***group*** modulo n under multiplication.

Remark 52.1.3. Observe that $\mathbb{Z}_1^* = \{1\}$, while for $n > 1$, $\mathbb{Z}_n^*$ does not contain n.

Lemma 52.1.4. *For any element $\alpha \in \mathbb{Z}_n^*$, there exists a unique inverse element $\beta = \alpha^{-1} \in \mathbb{Z}_n^*$ such that $\alpha * \beta \equiv_n 1$. Furthermore, the inverse can be computed in polynomial time*[①].

Proof: Since $\alpha \in \mathbb{Z}_n^*$, we have that $\gcd(\alpha, n) = 1$. As such, by Lemma 52.1.2, there exists x and y integers, such that $x\alpha + yn = 1$. That is $x\alpha \equiv 1 \pmod n$, and clearly $\beta := x \bmod n$ is the desired inverse, and it can be computed in polynomial time by Lemma 52.1.2.

As for uniqueness, assume that there are two inverses β, β' to $\alpha < n$, such that $\beta < \beta' < n$. But then $\beta\alpha \equiv_n \beta'\alpha \equiv_n 1$, which implies that $n \mid (\beta' - \beta)\alpha$, which implies that $n \mid \beta' - \beta$, which is impossible as $0 < \beta' - \beta < n$. ■

It is now straightforward, but somewhat tedious, to verify the following (the interested reader that had not encountered this stuff before can spend some time proving this).

Lemma 52.1.5. *The set $\mathbb{Z}_n$ under the $+$ operation modulo n is a group, as is $\mathbb{Z}_n^*$ under multiplication modulo n. More importantly, for a prime number p, $\mathbb{Z}_p$ forms a field with the $+, *$ operations modulo p (see Definition 52.1.17$_{\text{p340}}$).*

[①] Again, as is everywhere in this chapter, the polynomial time is in the number of bits needed to specify the input.

52.1.1.3. The Chinese remainder theorem

Theorem 52.1.6 (Chinese remainder theorem). *Let $n_1, \ldots, n_k$ be coprime numbers, and let $n = n_1 n_2 \cdots n_k$. For any residues $r_1 \in \mathbb{Z}_{n_1}, \ldots, r_k \in \mathbb{Z}_{n_k}$, there is a unique $r \in \mathbb{Z}_n$, which can be computed in polynomial time, such that $r \equiv r_i \pmod{n_i}$, for $i = 1, \ldots, k$.*

Proof: By the coprime property of the n_is it follows that $\gcd(n_i, n/n_i) = 1$. As such, $n/n_i \in \mathbb{Z}^*_{n_i}$, and it has a unique inverse m_i modulo n_i; that is $(n/n_i)m_i \equiv 1 \pmod{n_i}$. So set $r = \sum_i r_i m_i n/n_i$. Observe that for $i \neq j$, we have that $n_j \mid (n/n_i)$, and as such $r_i m_i n/n_i \pmod{n_j} \equiv 0 \pmod{n_j}$. As such, we have

$$r \bmod n_j = \left(\sum_i \left(r_i m_i \frac{n}{n_i} \bmod n_j \right) \right) \bmod n_j = \left(r_j m_j \frac{n}{n_j} \bmod n_j \right) \bmod n_j = r_j * 1 \bmod n_j = r_j.$$

As for uniqueness, if there is another such number r', such that $r < r' < n$, then $r' - r \pmod{n_i} = 0$ implying that $n_i \mid r' - r$, for all i. Since all the n_is are coprime, this implies that $n \mid r' - r$, which is of course impossible. ■

Lemma 52.1.7 (Fast exponentiation). *Given numbers b, c, n, one can compute $b^c \bmod n$ in polynomial time.*

Proof: The key property we need is that

$$xy \bmod n = \Big((x \bmod n)\,(y \bmod n) \Big) \bmod n.$$

Now, if c is even, then we can compute

$$b^c \bmod n = \left(b^{c/2}\right)^2 \bmod n = \left(b^{c/2} \bmod n\right)^2 \bmod n.$$

Similarly, if c is odd, we have

$$b^c \bmod n = (b \bmod n)\left(b^{(c-1)/2}\right)^2 \bmod n = (b \bmod n)\left(b^{(c-1)/2} \bmod n\right)^2 \bmod n.$$

Namely, computing $b^c \bmod n$ can be reduced to recursively computing $b^{\lfloor c/2 \rfloor} \bmod n$, and a constant number of operations (on numbers that are smaller than n). Clearly, the depth of the recursion is $O(\log c)$. ■

52.1.1.4. Euler totient function

The ***Euler totient function*** $\phi(n) = \left|\mathbb{Z}^*_n\right|$ is the number of positive integer numbers that at most n and are coprime with n. If n is prime then $\phi(n) = n - 1$.

Lemma 52.1.8. *Let $n = p_1^{k_1} \cdots p_t^{k_t}$, where the p_is are prime numbers and the k_is are positive integers (this is the* **prime factorization** *of n). Then $\phi(n) = \prod_{i=1}^{t} p_i^{k_i - 1}(p_i - 1)$. and this quantity can be computed in polynomial time if the factorization is given.*

Proof: Observe that $\phi(1) = 1$ (see Remark 52.1.3), and for a prime number p, we have that $\phi(p) = p - 1$. Now, for $k > 1$, and p prime we have that $\phi(p^k) = p^{k-1}(p-1)$, as a number $x \leq p^k$ is coprime with p^k, if and only if $x \bmod p \neq 0$, and $(p-1)/p$ fraction of the numbers in this range have this property.

Now, if n and m are relative primes, then $\gcd(x, nm) = 1 \iff \gcd(x, n) = 1$ and $\gcd(x, m) = 1$. In particular, there are $\phi(n)\phi(m)$ pairs $(\alpha, \beta) \in \mathbb{Z}^*_n \times \mathbb{Z}^*_m$, such that $\gcd(\alpha, n) = 1$ and $\gcd(\beta, m) = 1$. By the Chinese remainder theorem (Theorem 52.1.6), each such pair represents a unique number in the range $1, \ldots, nm$, as desired.

Now, the claim follows by easy induction on the prime factorization of the given number. ■

52.1.2. Structure of the modulo group $\mathbb{Z}_n$

52.1.2.1. Some basic group theory

Definition 52.1.9. A ***group*** is a set, $\mathcal{G}$, together with an operation $\times$ that combines any two elements a and b to form another element, denoted $a \times b$ or ab. To qualify as a group, the set and operation, $(\mathsf{G}, \times)$, must satisfy the following:

(A) (Closure) For all $a, b \in \mathcal{G}$, the result of the operation, $a \times b \in \mathcal{G}$.
(B) (Associativity) For all $a, b, c \in \mathcal{G}$, we have $(a \times b) \times c = a \times (b \times c)$.
(C) (Identity element) There exists an element $\mathsf{i} \in \mathcal{G}$, called the ***identity element***, such that for every element $a \in \mathcal{G}$, the equation $\mathsf{i} \times a = a \times \mathsf{i} = a$ holds.
(D) (Inverse element) For each $a \in \mathcal{G}$, there exists an element $b \in \mathcal{G}$ such that $a \times b = b \times a = \mathsf{i}$.

A group is ***abelian*** (aka, ***commutative group***) if for all $a, b \in \mathcal{G}$, we have that $a \times b = b \times a$.

In the following we restrict our attention to abelian groups since it makes the discussion somewhat simpler. In particular, some of the claims below holds even without the restriction to abelian groups.

The identity element is unique. Indeed, if both $f, g \in \mathcal{G}$ are identity elements, then $f = f \times g = g$. Similarly, for every element $x \in \mathcal{G}$ there exists a unique inverse $y = x^{-1}$. Indeed, if there was another inverse z, then $y = y \times \mathsf{i} = y \times (x \times z) = (y \times x) \times z = \mathsf{i} \times z = z$.

52.1.2.2. Subgroups

For a group $\mathcal{G}$, a subset $\mathcal{H} \subseteq \mathcal{G}$ that is also a group (under the same operation) is a ***subgroup***.

For $x, y \in \mathcal{G}$, let us define $x \sim y$ if $x/y \in \mathcal{H}$. Here $x/y = xy^{-1}$ and y^{-1} is the inverse of y in $\mathcal{G}$. Observe that $(y/x)(x/y) = \left(yx^{-1}\right)\left(xy^{-1}\right) = \mathsf{i}$. That is y/x is the inverse of x/y, and it is in $\mathcal{H}$. But that implies that $x \sim y \implies y \sim x$. Now, if $x \sim y$ and $y \sim z$, then $x/y, y/z \in \mathcal{H}$. But then $x/y \times y/z \in \mathcal{H}$, and furthermore $x/y \times y/z = xy^{-1}yz^{-1} = xz^{-1} = x/z$. that is $x \sim z$. Together, this implies that $\sim$ is an equivalence relationship.

Furthermore, observe that if $x/y = x/z$ then $y^{-1} = x^{-1}(x/y) = x^{-1}(x/z) = z^{-1}$, that is $y = z$. In particular, the equivalence class of $x \in \mathcal{G}$, is $[x] = \left\{z \in \mathcal{G} \;\middle|\; x \sim z\right\}$. Observe that if $x \in \mathcal{H}$ then $\mathsf{i}/x = \mathsf{i}x^{-1} = x^{-1} \in \mathcal{H}$, and thus $\mathsf{i} \sim x$. That is $\mathcal{H} = [x]$. The following is now easy.

Lemma 52.1.10. *Let $\mathcal{G}$ be an abelian group, and let $\mathcal{H} \subseteq \mathcal{G}$ be a subgroup. Consider the set $\mathcal{G}/\mathcal{H} = \left\{[x] \;\middle|\; x \in \mathcal{G}\right\}$. We claim that $\left|[x]\right| = \left|[y]\right|$ for any $x, y \in \mathcal{G}$. Furthermore $\mathcal{G}/\mathcal{H}$ is a group (that is, the* **quotient group***), with $[x] \times [y] = [x \times y]$.*

Proof: Pick an element $\alpha \in [x]$, and $\beta \in [y]$, and consider the mapping $f(x) = x\alpha^{-1}\beta$. We claim that f is one to one and onto from $[x]$ to $[y]$. For any $\gamma \in [x]$, we have that $\gamma\alpha^{-1} = \gamma/\alpha \in \mathcal{H}$ As such, $f(\gamma) = \gamma\alpha^{-1}\beta \in [\beta] = [y]$. Now, for any $\gamma, \gamma' \in [x]$ such that $\gamma \neq \gamma'$, we have that if $f(\gamma) = \gamma\alpha^{-1}\beta = \gamma'\alpha^{-1}\beta = f(\gamma')$, then by multiplying by $\beta^{-1}\alpha$, we have that $\gamma = \gamma'$. That is, f is one to one, implying that $\left|[x]\right| = \left|[y]\right|$.

The second claim follows by careful but tediously checking that the conditions in the definition of a group holds. ■

Lemma 52.1.11. *For a finite abelian group $\mathcal{G}$ and a subgroup $\mathcal{H} \subseteq \mathcal{G}$, we have that $|\mathcal{H}|$ divides $|\mathcal{G}|$.*

Proof: By Lemma 52.1.10, we have that $|\mathcal{G}| = |\mathcal{H}| \cdot |\mathcal{G}/\mathcal{H}|$, as $\mathcal{H} = [\mathsf{i}]$. ■

52.1.2.3. Cyclic groups

Lemma 52.1.12. *For a finite group $\mathcal{G}$, and any element $g \in \mathcal{G}$, the set $\langle g \rangle = \left\{ g^i \;\middle|\; i \geq 0 \right\}$ is a group.*

Proof: Since $\mathcal{G}$ is finite, there are integers $i > j \geq 1$, such that $i \neq j$ and $g^i = g^j$, but then $g^j \times g^{i-j} = g^i = g^j$. That is $g^{i-j} = \mathsf{i}$ and, by definition, we have $g^{i-j} \in \langle g \rangle$. It is now straightforward to verify that the other properties of a group hold for $\langle g \rangle$. ■

In particular, for an element $g \in \mathcal{G}$, we define its ***order*** as $\operatorname{ord}(g) = \left|\langle g \rangle\right|$, which clearly is the minimum positive integer m, such that $g^m = \mathsf{i}$. Indeed, for $j > m$, observe that $g^j = g^{j \bmod m} \in X = \left\{ \mathsf{i}, g, g^2, \ldots, g^{m-1} \right\}$, which implies that $\langle g \rangle = X$.

A group $\mathcal{G}$ is ***cyclic***, if there is an element $g \in \mathcal{G}$, such that $\langle g \rangle = \mathcal{G}$. In such a case g is a ***generator*** of $\mathcal{G}$.

Lemma 52.1.13. *For any finite abelian group $\mathcal{G}$, and any $g \in \mathcal{G}$, we have that* $\operatorname{ord}(g)$ *divides* $|\mathcal{G}|$, *and* $g^{|\mathcal{G}|} = \mathsf{i}$.

Proof: By Lemma 52.1.12, the set $\langle g \rangle$ is a subgroup of $\mathcal{G}$. By Lemma 52.1.11, we have that $\operatorname{ord}(g) = \left|\langle g \rangle\right| \mid |\mathcal{G}|$. As such, $g^{|\mathcal{G}|} = \left(g^{\operatorname{ord}(g)}\right)^{|\mathcal{G}|/\operatorname{ord}(g)} = \left(\mathsf{i}\right)^{|\mathcal{G}|/\operatorname{ord}(g)} = \mathsf{i}$. ■

52.1.2.4. Modulo group

Lemma 52.1.14. *For any integer n, consider the additive group $\mathbb{Z}_n$. Then, for any $x \in \mathbb{Z}_n$, we have that $x \cdot \operatorname{ord}(x) = \operatorname{lcm}(x, n)$. In particular,* $\operatorname{ord}(x) = \dfrac{\operatorname{lcm}(n, x)}{x} = \dfrac{n}{\gcd(n, x)}$. *If n is prime, and $x \neq 0$ then* $\operatorname{ord}(x) = |\mathbb{Z}_n| = n$, *and $\mathbb{Z}_n$ is a cyclic group.*

Proof: We are working modulo n here under additions, and the identity element is 0. As such, $x \cdot \operatorname{ord}(x) \equiv_n 0$, which implies that $n \mid x \operatorname{ord}(x)$. By definition, $\operatorname{ord}(x)$ is the minimal number that has this property, implying that $\operatorname{ord}(x) = \dfrac{\operatorname{lcm}(n, x)}{x}$. Now, $\operatorname{lcm}(n, x) = nx/\gcd(n, x)$. The second claim is now easy. ■

Theorem 52.1.15. ***(Euler's theorem)*** *For all n and $x \in \mathbb{Z}_n^*$, we have $x^{\phi(n)} \equiv 1 \pmod{n}$.*

(Fermat's theorem) *If p is a prime then $\forall x \in \mathbb{Z}_p^* \quad x^{p-1} \equiv 1 \pmod{p}$.*

Proof: The group $\mathbb{Z}_n^*$ is abelian and has $\phi(n)$ elements, with 1 being the identity element (duh!). As such, by Lemma 52.1.13, we have that $x^{\phi(n)} = x^{|\mathbb{Z}_n^*|} \equiv 1 \pmod{n}$, as claimed.

The second claim follows by setting $n = p$, and recalling that $\phi(p) = p - 1$, if p is a prime. ■

One might be tempted to think that Lemma 52.1.14 implies that if p is a prime then $\mathbb{Z}_p^*$ is a cyclic group, but this does not follow, as the cardinality of $\mathbb{Z}_p^*$ is $\phi(p) = p - 1$, which is not a prime number (for $p > 2$). To prove that $\mathbb{Z}_p^*$ is cyclic, let us go back shortly to the totient function.

Lemma 52.1.16. *For any $n > 0$, we have $\sum_{d \mid n} \phi(d) = n$.*

Proof: For any $g > 0$, let $V_g = \left\{ x \;\middle|\; x \in \{1, \ldots, n\} \text{ and } \gcd(x, n) = g \right\}$. Now, $x \in V_g \iff \gcd(x, n) = g \iff \gcd(x/g, n/g) = 1 \iff x/g \in \mathbb{Z}_{n/g}^*$. Since $V_1, V_2, \ldots, V_n$ form a partition of $\{1, \ldots, n\}$, it follows that

$$n = \sum_g \left|V_g\right| = \sum_{g \mid n} \left|\mathbb{Z}_{n/g}^*\right| = \sum_{g \mid n} \phi(n/g) = \sum_{d \mid n} \phi(d).$$ ■

52.1.2.5. Fields

Definition 52.1.17. A ***field*** is an algebraic structure $\langle \mathbb{F}, +, *, 0, 1 \rangle$ consisting of two abelian groups:
(A) $\mathbb{F}$ under $+$, with 0 being the identity element.
(B) $\mathbb{F} \setminus \{0\}$ under $*$, with 1 as the identity element (here $0 \neq 1$).
Also, the following property (***distributivity of multiplication over addition***) holds:

$$\forall a, b, c \in \mathbb{F} \qquad a * (b + c) = (a * b) + (a * c).$$

We need the following: A polynomial p of degree k over a field $\mathbb{F}$ has at most k roots. indeed, if p has the root α then it can be written as $p(x) = (x - \alpha)q(x)$, where $q(x)$ is a polynomial of one degree lower. To see this, we divide $p(x)$ by the polynomial $(x - \alpha)$, and observe that $p(x) = (x - \alpha)q(x) + \beta$, but clearly $\beta = 0$ since $p(\alpha) = 0$. As such, if p had t roots $\alpha_1, \ldots, \alpha_t$, then $p(x) = q(x) \prod_{i=1}^{t} (x - \alpha_i)$, which implies that p would have degree at least t.

52.1.2.6. $\mathbb{Z}_p^*$ is cyclic for prime numbers

For a prime number p, the group $\mathbb{Z}_p^*$ has size $\phi(p) = p - 1$, which is not a prime number for $p > 2$. As such, Lemma 52.1.13 does not imply that there must be an element in $\mathbb{Z}_p^*$ that has order $p - 1$ (and thus $\mathbb{Z}_p^*$ is cyclic). Instead, our argument is going to be more involved and less direct.

Lemma 52.1.18. *For $k < n$, let $R_k = \left\{ x \in \mathbb{Z}_p^* \;\middle|\; \operatorname{ord}(x) = k \right\}$ be the set of all numbers in $\mathbb{Z}_p^*$ that are of order k. We have that $|R_k| \leq \phi(k)$.*

Proof: Clearly, all the elements of R_k are roots of the polynomial $x^k - 1 = 0 \pmod n$. By the above, this polynomial has at most k roots. Now, if R_k is not empty, then it contains an element $x \in R_k$ of order k, which implies that for all $i < j \leq k$, we have that $x^i \not\equiv x^j \pmod n$, as the order of x is the size of $\langle x \rangle$, and the minimum k such that $x^k \equiv 1 \pmod n$. In particular, we have that $R_k \subseteq \langle x \rangle$, as for $y = x^j$, we have that $y^k \equiv_n x^{jk} \equiv_n 1^j \equiv_n 1$.

Observe that for $y = x^i$, if $g = \gcd(k, i) > 1$, then $y^{k/g} \equiv_n x^{i(k/g)} \equiv_n x^{\operatorname{lcm}(i,k)} \equiv_n 1$; that is, $\operatorname{ord}(y) \leq k/g < k$, and $y \notin R_k$. As such, R_k contains only elements of x^i such that $\gcd(i, k) = 1$. That is $R_k \subseteq \mathbb{Z}_k^*$. The claim now readily follows as $\left|\mathbb{Z}_k^*\right| = \phi(k)$. ■

Lemma 52.1.19. *For any prime p, the group $\mathbb{Z}_p^*$ is cyclic.*

Proof: For $p = 2$ the claim trivially holds, so assume $p > 2$. If the set R_{p-1}, from Lemma 52.1.18, is not empty, then there is $g \in R_{p-1}$, it has order $p - 1$, and it is a generator of $\mathbb{Z}_p^*$, as $\left|\mathbb{Z}_p^*\right| = p - 1$, implying that $\mathbb{Z}_p^* = \langle g \rangle$ and this group is cyclic.

Now, by Lemma 52.1.13, we have that for any $y \in \mathbb{Z}_p^*$, we have that $\operatorname{ord}(y) \mid p - 1 = \left|\mathbb{Z}_p^*\right|$. This implies that R_k is empty if k does not divides $p - 1$. On the other hand, $R_1, \ldots, R_{p-1}$ form a partition of $\mathbb{Z}_p^*$. As such, we have that

$$p - 1 = \left|\mathbb{Z}_p^*\right| = \sum_{k \mid p-1} |R_k| \leq \sum_{k \mid p-1} \phi(k) = p - 1,$$

by Lemma 52.1.18 and Lemma 52.1.16$_{\text{p339}}$, implying that the inequality in the above display is equality, and for all $k \mid p - 1$, we have that $|R_k| = \phi(k)$. In particular, $\left|R_{p-1}\right| = \phi(p - 1) > 0$, and by the above the claim follows. ■

52.1.2.7. $\mathbb{Z}_n^*$ is cyclic for powers of a prime

Lemma 52.1.20. *Consider any odd prime p, and any integer $c \geq 1$, then the group $\mathbb{Z}_n^*$ is cyclic, where $n = p^c$.*

Proof: Let g be a generator of $\mathbb{Z}_p^*$. Observe that $g^{p-1} \equiv 1 \pmod{p}$. The number $g < p$, and as such p does not divide g, and also p does not divide g^{p-2}, and also p does not divide $p-1$. As such, p^2 does not divide $\Delta = (p-1)g^{p-2}p$; that is, $\Delta \not\equiv 0 \pmod{p^2}$. As such, we have that

$$(g+p)^{p-1} \equiv g^{p-1} + \binom{p-1}{1} g^{p-2}p \equiv g^{p-1} + \Delta \not\equiv g^{p-1} \pmod{p^2}$$

$$\implies \qquad (g+p)^{p-1} \not\equiv 1 \pmod{p^2} \qquad \text{or} \qquad g^{p-1} \not\equiv 1 \pmod{p^2}.$$

Renaming $g+p$ to be g, if necessary, we have that $g^{p-1} \not\equiv 1 \pmod{p^2}$, but by Theorem 52.1.15$_{\text{p339}}$, $g^{p-1} \equiv 1 \pmod{p}$. As such, $g^{p-1} = 1 + \beta p$, where p does not divide β. Now, we have

$$g^{p(p-1)} = (1+\beta p)^p = 1 + \binom{p}{1}\beta p + \beta p^3 \text{<whatever>} = 1 + \gamma_1 p^2,$$

where γ_1 is an integer (the p^3 is not a typo – the binomial coefficient contributes at least one factor of p – here we are using that $p > 2$). In particular, as p does not divides β, it follows that p does not divides γ_1 either. Let us apply this argumentation again to

$$g^{p^2(p-1)} = \left(1 + \gamma_1 p^2\right)^p = 1 + \gamma_1 p^3 + p^4 \text{<whatever>} = 1 + \gamma_2 p^3,$$

where again p does not divides γ_2. Repeating this argument, for $i = 1, \ldots, c-2$, we have

$$\alpha_i = g^{p^i(p-1)} = \left(g^{p^{i-1}(p-1)}\right)^p = \left(1 + \gamma_{i-1}p^i\right)^p = 1 + \gamma_{i-1}p^{i+1} + p^{i+2} \text{<whatever>} = 1 + \gamma_i p^{i+1},$$

where p does not divides γ_i. In particular, this implies that $\alpha_{c-2} = 1 + \gamma_{c-2}p^{c-1}$ and p does not divides γ_{c-2}. This in turn implies that $\alpha_{c-2} \not\equiv 1 \pmod{p^c}$.

Now, the order of g in $\mathbb{Z}_n$, denoted by k, must divide $\left|\mathbb{Z}_n^*\right|$ by Lemma 52.1.13$_{\text{p339}}$. Now $\left|\mathbb{Z}_n^*\right| = \phi(n) = p^{c-1}(p-1)$, see Lemma 52.1.8$_{\text{p337}}$. So, $k \mid p^{c-1}(p-1)$. Also, $\alpha_{c-2} \not\equiv 1 \pmod{p^c}$. implies that k does not divides $p^{c-2}(p-1)$. It follows that $p^{c-1} \mid k$. So, let us write $k = p^{c-1}k'$, where $k' \leq (p-1)$. This, by definition, implies that $g^k \equiv 1 \pmod{p^c}$. Now, $g^p \equiv g \pmod{p}$, because g is a generator of $\mathbb{Z}_p^*$. As such, we have that

$$g^k \equiv_p g^{p^\delta k'} \equiv_p (g^p)^{p^{\delta-1}k'} \equiv_p (g)^{p^{\delta-1}k'} \equiv_p \ldots \equiv_p (g)^{k'} \equiv_p \left(g^k \bmod p^c\right) \bmod p \equiv_p 1.$$

Namely, $g^{k'} \equiv 1 \pmod{p}$, which implies, as g as a generator of $\mathbb{Z}_p^*$, that either $k' = 1$ or $k' = p-1$. The case $k' = 1$ is impossible, as this implies that $g = 1$, and it can not be the generator of $\mathbb{Z}_p^*$. We conclude that $k = p^{c-1}(p-1)$; that is, $\mathbb{Z}_n^*$ is cyclic. ■

52.1.3. Quadratic residues

52.1.3.1. Quadratic residue

Definition 52.1.21. An integer α is a ***quadratic residue*** modulo a positive integer n, if $\gcd(\alpha, n) = 1$ and for some integer β, we have $\alpha \equiv \beta^2 \pmod{n}$.

Theorem 52.1.22 (Euler's criterion). *Let p be an odd prime, and $\alpha \in \mathbb{Z}_p^*$. We have that*

(A) $\alpha^{(p-1)/2} \equiv_p \pm 1$.
(B) If α *is a quadratic residue, then* $\alpha^{(p-1)/2} \equiv_p 1$.
(C) If α *is not a quadratic residue, then* $\alpha^{(p-1)/2} \equiv_p -1$.

Proof: (A) Let $\gamma = \alpha^{(p-1)/2}$, and observe that $\gamma^2 \equiv_p \alpha^{p-1} \equiv 1$, by Fermat's theorem (Theorem 52.1.15$_{\text{p339}}$), which implies that γ is either $+1$ or -1, as the polynomial $x^2 - 1$ has at most two roots over a field.

(B) Let $\alpha \equiv_p \beta^2$, and again by Fermat's theorem, we have $\alpha^{(p-1)/2} \equiv_p \beta^{p-1} \equiv_p 1$.

(C) Let X be the set of elements in $\mathbb{Z}_p^*$ that are not quadratic residues, and consider $\alpha \in X$. Since $\mathbb{Z}_p^*$ is a group, for any $x \in \mathbb{Z}_p^*$ there is a unique $y \in \mathbb{Z}_p^*$ such that $xy \equiv_p \alpha$. As such, we partition $\mathbb{Z}_p^*$ into pairs $C = \left\{ \{x, y\} \;\middle|\; x, y \in \mathbb{Z}_p^* \text{ and } xy \equiv_p \alpha \right\}$. We have that

$$\tau \equiv_p \prod_{\beta \in \mathbb{Z}_p^*} \beta \equiv_p \prod_{\{x,y\} \in C} xy \equiv_p \prod_{\{x,y\} \in C} \alpha \equiv_p \alpha^{(p-1)/2}.$$

Let consider a similar set of pair, but this time for 1: $D = \left\{ \{x, y\} \;\middle|\; x, y \in \mathbb{Z}_p^*, x \neq y \text{ and } xy \equiv_p 1 \right\}$. Clearly, D does not contain -1 and 1, but all other elements in $\mathbb{Z}_p^*$ are in D. As such,

$$\tau \equiv_p \prod_{\beta \in \mathbb{Z}_p^*} \beta \equiv_p (-1)1 \prod_{\{x,y\} \in D} xy \equiv_p \prod_{\{x,y\} \in D} 1 \equiv_p -1.$$ ■

52.1.3.2. Legendre symbol

For an odd prime p, and an integer a with $\gcd(a, n) = 1$, the ***Legendre symbol*** $(a \mid p)$ is one if a is a quadratic residue modulo p, and -1 otherwise (if $p \mid a$, we define $(a \mid p) = 0$). Euler's criterion (Theorem 52.1.22) implies the following equivalent definition.

Definition 52.1.23. The ***Legendre symbol***, for a prime number p, and $a \in \mathbb{Z}_p^*$, is

$$(a \mid p) = a^{(p-1)/2} \pmod{p}.$$

The following is easy to verify.

Lemma 52.1.24. *Let* p *be an odd prime, and let* a, b *be integer numbers. We have:*
(i) $(-1 \mid p) = (-1)^{(p-1)/2}$.
(ii) $(a \mid p)(b \mid p) = (ab \mid p)$.
(iii) If $a \equiv_p b$ *then* $(a \mid p) = (b \mid p)$.

Lemma 52.1.25 (Gauss' lemma). *Let* p *be an odd prime and let* a *be an integer that is not divisible by* p. *Let* $X = \left\{ \alpha_j = ja \pmod{p} \;\middle|\; j = 1, \ldots, (p-1)/2 \right\}$, *and* $L = \left\{ x \in X \;\middle|\; x > p/2 \right\} \subseteq X$. *Then* $(a \mid p) = (-1)^n$, *where* $n = |L|$.

Proof: Observe that for any distinct i, j, such that $1 \leq i \leq j \leq (p-1)/2$, we have that $ja \equiv ia \pmod{p}$ implies that $(j - i)a \equiv 0 \pmod{p}$, which is impossible as $j - i < p$ and $\gcd(a, p) = 1$. As such, all the elements of X are distinct, and $|X| = (p-1)/2$. We have a somewhat stronger property: If $ja \equiv p - ia \pmod{p}$ implies $(j + i)a \equiv 0 \pmod{p}$, which is impossible. That is, $S = X \setminus L$, and $\overline{L} = \left\{ p - \ell \;\middle|\; \ell \in L \right\}$ are disjoint, and $S \cup \overline{L} = \left\{ 1, \ldots, (p-1)/2 \right\}$. As such,

$$\left(\frac{p-1}{2}\right)! \equiv \prod_{x \in S} x \cdot \prod_{y \in L} (p - y) \equiv (-1)^n \prod_{x \in S} x \cdot \prod_{y \in L} y \equiv (-1)^n \prod_{j=1}^{(p-1)/2} ja \equiv (-1)^n a^{(p-1)/2} \left(\frac{p-1}{2}\right)! \pmod{p}.$$

Dividing both sides by $(-1)^n((p-1)/2)!$, we have that $(a \mid p) \equiv a^{(p-1)/2} \equiv (-1)^n \pmod{p}$, as claimed. ■

Lemma 52.1.26. *If p is an odd prime, and $a > 2$ and $\gcd(a,p) = 1$ then $(a \mid p) = (-1)^{\Delta}$, where $\Delta = \sum_{j=1}^{(p-1)/2} \lfloor ja/p \rfloor$. Furthermore, we have $(2 \mid p) = (-1)^{(p^2-1)/8}$.*

Proof: Using the notation of Lemma 52.1.25, we have

$$\sum_{j=1}^{(p-1)/2} ja = \sum_{j=1}^{(p-1)/2} \Big(\lfloor ja/p \rfloor p + (ja \bmod p) \Big) = \Delta p + \sum_{x \in S} x + \sum_{y \in L} y = (\Delta + n)p + \sum_{x \in S} x - \sum_{y \in \overline{L}} y$$
$$= (\Delta + n)p + \sum_{j=1}^{(p-1)/2} j - 2 \sum_{y \in \overline{L}} y.$$

Rearranging, and observing that $\sum_{j=1}^{(p-1)/2} j = \frac{p-1}{2} \cdot \frac{1}{2}\Big(\frac{p-1}{2} + 1\Big) = \frac{p^2-1}{8}$. We have that

$$(a-1)\frac{p^2-1}{8} = (\Delta + n)p - 2\sum_{y \in \overline{L}} y. \qquad \Longrightarrow \qquad (a-1)\frac{p^2-1}{8} \equiv (\Delta + n)p \pmod 2. \tag{52.1}$$

Observe that $p \equiv 1 \pmod 2$, and for any x we have that $x \equiv -x \pmod 2$. As such, and if a is odd, then the above implies that $n \equiv \Delta \pmod 2$. Now the claim readily follows from Lemma 52.1.25.

As for $(2 \mid p)$, setting $a = 2$, observe that $\lfloor ja/p \rfloor = 0$, for $j = 0, \ldots (p-1)/2$, and as such $\Delta = 0$. Now, Eq. (52.1) implies that $\frac{p^2-1}{8} \equiv n \pmod 2$, and the claim follows from Lemma 52.1.25. ■

Theorem 52.1.27 (*Law of quadratic reciprocity*). *If p and q are distinct odd primes, then*

$$(p \mid q) = (-1)^{\frac{p-1}{2}\frac{q-1}{2}} (q \mid p).$$

Proof: Let $S = \Big\{(x,y) \;\Big|\; 1 \le x \le (p-1)/2 \text{ and } 1 \le y \le (q-1)/2\Big\}$. As $\operatorname{lcm}(p,q) = pq$, it follows that there are no $(x,y) \in S$, such that $qx = py$, as all such numbers are strict smaller than pq. Now, let

$$S_1 = \Big\{(x,y) \in S \;\Big|\; qx > py\Big\} \qquad \text{and} \qquad S_2 = \Big\{(x,y) \in S \;\Big|\; qx < py\Big\}.$$

Now, $(x,y) \in S_1 \iff 1 \le x \le (p-1)$, and $1 \le y \le \lfloor qx/p \rfloor$. As such, we have $|S_1| = \sum_{x=1}^{(p-1)/2} \lfloor qx/p \rfloor$, and similarly $|S_2| = \sum_{y=1}^{(q-1)/2} \lfloor py/q \rfloor$. We have

$$\tau = \frac{p-1}{2} \cdot \frac{q-1}{2} = |S| = |S_1| + |S_2| = \underbrace{\sum_{x=1}^{(p-1)/2} \lfloor qx/p \rfloor}_{\tau_1} + \underbrace{\sum_{y=1}^{(q-1)/2} \lfloor py/q \rfloor}_{\tau_2}.$$

The claim now readily follows by Lemma 52.1.26, as $(-1)^{\tau} = (-1)^{\tau_1}(-1)^{\tau_2} = (p \mid q)(q \mid p)$. ■

52.1.3.3. Jacobi symbol

Definition 52.1.28. For any integer a, and an odd number n with prime factorization $n = p_1^{k_1} \cdots p_t^{k_t}$, its ***Jacobi symbol*** is

$$[\![a \mid n]\!] = \prod_{i=1}^{t} (a \mid p_i)^{k_i}.$$

Claim 52.1.29. *For odd integers $n_1, \ldots, n_k$, we have that $\sum_{i=1}^{k}(n_i-1)/2 \equiv \left(\prod_{i=1}^{k} n_i - 1\right)/2 \pmod 2$.*

Proof: We prove for two odd integers x and y, and apply this repeatedly to get the claim. Indeed, we have $\frac{x-1}{2} + \frac{y-1}{2} \equiv \frac{xy-1}{2} \pmod 2 \iff 0 \equiv \frac{xy-x+1-y+1-1}{2} \pmod 2 \iff 0 \equiv \frac{xy-x-y+1}{2} \pmod 2 \iff 0 \equiv \frac{(x-1)(y-1)}{2} \pmod 2$, which is obviously true. ■

Lemma 52.1.30 (*Law of quadratic reciprocity*). *For n and m positive odd integers, we have that $[\![n \mid m]\!] = (-1)^{\frac{n-1}{2}\frac{m-1}{2}} [\![m \mid n]\!]$.*

Proof: Let $n = \prod_{i=1}^{\nu} p_i$ and Let $m = \prod_{j=1}^{\mu} q_j$ be the prime factorization of the two numbers (allowing repeated factors). If they share a common factor p, then both $[\![n \mid m]\!]$ and $[\![m \mid n]\!]$ contain a zero term when expanded, as $(n \mid p) = (m \mid p) = 0$. Otherwise, we have

$$\begin{aligned}
[\![n \mid m]\!] &= \prod_{i=1}^{\nu}\prod_{j=1}^{\mu} [\![p_i \mid q_j]\!] = \prod_{i=1}^{\nu}\prod_{j=1}^{\mu} \left(p_i \mid q_j\right) = \prod_{i=1}^{\nu}\prod_{j=1}^{\mu} (-1)^{(q_j-1)/2\cdot(p_i-1)/2} \left(q_j \mid p_i\right) \\
&= \underbrace{\prod_{i=1}^{\nu}\prod_{j=1}^{\mu} (-1)^{(q_j-1)/2\cdot(p_i-1)/2}}_{s} \cdot \left(\prod_{i=1}^{\nu}\prod_{j=1}^{\mu} \left(q_j \mid p_i\right)\right) = s\,[\![m \mid n]\!].
\end{aligned}$$

by Theorem 52.1.27. As for the value of s, observe that

$$s = \prod_{i=1}^{\nu}\left(\prod_{j=1}^{\mu} (-1)^{(q_j-1)/2}\right)^{(p_i-1)/2} = \prod_{i=1}^{\nu}\left((-1)^{(m-1)/2}\right)^{(p_i-1)/2} = \left(\prod_{i=1}^{\nu}(-1)^{(p_i-1)/2}\right)^{(m-1)/2} = (-1)^{(n-1)/2\cdot(m-1)/2},$$

by repeated usage of Claim 52.1.29. ■

Lemma 52.1.31. *For odd integers n and m, we have that $\frac{n^2-1}{8} + \frac{m^2-1}{8} \equiv \frac{n^2m^2-1}{8} \pmod 2$.*

Proof: For an odd integer n, we have that either (i) $2 \mid n-1$ and $4 \mid n+1$, or (ii) $4 \mid n-1$ and $2 \mid n+1$. As such, $8 \mid n^2-1 = (n-1)(n+1)$. In particular, $64 \mid \left(n^2-1\right)\left(m^2-1\right)$. We thus have that

$$\begin{aligned}
\frac{\left(n^2-1\right)\left(m^2-1\right)}{8} \equiv 0 \pmod 2 &\iff \frac{n^2m^2-n^2-m^2+1}{8} \equiv 0 \pmod 2 \\
&\iff \frac{n^2m^2-1}{8} \equiv \frac{n^2-m^2-2}{8} \pmod 2 \\
&\iff \frac{n^2-1}{8} + \frac{m^2-1}{8} \equiv \frac{n^2m^2-1}{8} \pmod 2.
\end{aligned}$$

■

Lemma 52.1.32. *Let m, n be odd integers, and a, b be any integers. We have the following:*

(A) $[\![ab \mid n]\!] = [\![a \mid n]\!]\,[\![b \mid n]\!]$.
(B) $[\![a \mid nm]\!] = [\![a \mid n]\!]\,[\![a \mid m]\!]$.
(C) If $a \equiv b \pmod n$ *then* $[\![a \mid n]\!] = [\![b \mid n]\!]$.
(D) If $\gcd(a, n) > 1$ *then* $[\![a \mid n]\!] = 0$.
(E) $[\![1 \mid n]\!] = 1$.

(F) $[\![2 \mid n]\!] = (-1)^{(n^2-1)/8}$.
(G) $[\![n \mid m]\!] = (-1)^{\frac{n-1}{2}\frac{m-1}{2}} [\![m \mid n]\!]$.

Proof: (A) Follows immediately, as $(ab \mid p_i) = (a \mid p_i)(b \mid p_i)$, see Lemma 52.1.24$_{\text{p342}}$.
(B) Immediate from definition.
(C) Follows readily from Lemma 52.1.24$_{\text{p342}}$ (iii).
(D) Indeed, if $p \mid \gcd(a, n)$ and $p > 1$, then $(a \mid p)^k = (0 \mid p)^k = 0$ appears as a term in $[\![a \mid n]\!]$.
(E) Obvious by definition.
(F) By Lemma 52.1.26$_{\text{p343}}$, for a prime p, we have $(2 \mid p) = (-1)^{(p^2-1)/8}$. As such, writing $n = \prod_{i=1}^{t} p_i$ as a product of primes (allowing repeated primes), we have

$$[\![2 \mid n]\!] = \prod_{i=1}^{t} (2 \mid p_i) = \prod_{i=1}^{t} (-1)^{(p_i^2-1)/8} = (-1)^{\Delta},$$

where $\Delta = \sum_{i=1}^{t} (p_i^2 - 1)/8$. As such, we need to compute the $\Delta \pmod 2$, which by Lemma 52.1.31, is

$$\Delta \equiv \sum_{i=1}^{t} \frac{p_i^2 - 1}{8} \equiv \frac{\prod_{i=1}^{t} p_i^2 - 1}{8} \equiv \frac{n^2 - 1}{8} \pmod 2,$$

and as such $[\![2 \mid n]\!] = (-1)^{\Delta} = (-1)^{(n^2-1)/8}$.
(G) This is Lemma 52.1.30. ■

52.1.3.4. Jacobi(a, n): Computing the Jacobi symbol

Given a and n (n is an odd number), we are interested in computing (in polynomial time) the Jacobi symbol $[\![a \mid n]\!]$. The algorithm **Jacobi**(a, n) works as follows:

(A) If $a = 0$ then **return** 0 `// Since` $[\![0 \mid n]\!] = 0$.
(B) If $a > n$ then **return** **Jacobi**$(a \pmod n, n)$ `// Lemma 52.1.32 (C)`
(C) If $\gcd(a, n) > 1$ then **return** 0 `// Lemma 52.1.32 (D)`
(D) If $a = 2$ then
  (I) Compute $\Delta = n^2 - 1 \pmod{16}$,
  (II) Return $(-1)^{\Delta/8 \pmod 2}$ `// As` $(n^2 - 1)/8 \equiv \Delta/8 \pmod 2$, `and by Lemma 52.1.32 (F)`
(E) If $2 \mid a$ then **return** **Jacobi**$(2, n)$ * **Jacobi**$(a/2, n)$ `// Lemma 52.1.32 (A)`

`// Must be that` a `and` b `are both odd,` $a < n$, `and they are coprime`
(F) $a' := a \pmod 4$, $n' := n \pmod 4$, $\beta = (a' - 1)(n' - 1)/4$.
return $(-1)^{\beta}$ **Jacobi**(n, a) `// By Lemma 52.1.32 (G)`

Ignoring the recursive calls, all the operations takes polynomial time. Clearly, computing **Jacobi**$(2, n)$ takes polynomial time. Otherwise, observe that **Jacobi** reduces its input size by say, one bit, at least every two recursive calls, and except the $a = 2$ case, it always perform only a single call. Thus, it follows that its running time is polynomial. We thus get the following.

Lemma 52.1.33. *Given integers a and n, where n is odd, then $[\![a \mid n]\!]$ can be computed in polynomial time.*

52.1.3.5. Subgroups induced by the Jacobi symbol

For an n, consider the set

$$J_n = \left\{ a \in \mathbb{Z}_n^* \;\middle|\; \left[\!\left[a \mid n \right]\!\right] \equiv a^{(n-1)/2} \bmod n \right\}. \tag{52.2}$$

Claim 52.1.34. *The set J_n is a subgroup of $\mathbb{Z}_n^*$.*

Proof: For $a, b \in J_n$, we have that $\left[\!\left[ab \mid n \right]\!\right] \equiv \left[\!\left[a \mid n \right]\!\right] \left[\!\left[b \mid n \right]\!\right] \equiv a^{(n-1)/2} b^{(n-1)/2} \equiv (ab)^{(n-1)/2} \bmod n$, implying that $ab \in J_n$. Now, $\left[\!\left[1 \mid n \right]\!\right] = 1$, so $1 \in J_n$. Now, for $a \in J_n$, let a^{-1} the inverse of a (which is a number in $\mathbb{Z}_n^*$). Observe that $a(a^{-1}) = kn + 1$, for some k, and as such, we have

$$1 = \left[\!\left[1 \mid n \right]\!\right] = \left[\!\left[kn + 1 \mid n \right]\!\right] = \left[\!\left[aa^{-1} \mid n \right]\!\right] = \left[\!\left[kn + 1 \mid n \right]\!\right] = \left[\!\left[a \mid n \right]\!\right] \left[\!\left[a^{-1} \mid n \right]\!\right].$$

And modulo n, we have

$$1 \equiv \left[\!\left[a \mid n \right]\!\right] \left[\!\left[a^{-1} \mid n \right]\!\right] \equiv a^{(n-1)/2} \left[\!\left[a^{-1} \mid n \right]\!\right] \bmod n.$$

Which implies that $\left(a^{-1}\right)^{(n-1)/2} \equiv \left[\!\left[a^{-1} \mid n \right]\!\right] \bmod n$. That is $a^{-1} \in J_n$.

Namely, J_n contains the identity, it is closed under inverse and multiplication, and it is now easy to verify that fulfill the other requirements to be a group. ■

Lemma 52.1.35. *Let n be an odd integer that is composite, then $|J_n| \leq \left|\mathbb{Z}_n^*\right|/2$.*

Proof: Let has the prime factorization $n = \prod_{i=1}^{t} p_i^{k_i}$. Let $q = p_1^{k_1}$, and $m = n/q$. By Lemma 52.1.20$_{\text{p341}}$, the group $\mathbb{Z}_q^*$ is cyclic, and let g be its generator. Consider the element $a \in \mathbb{Z}_n^*$ such that

$$a \equiv g \bmod q \qquad \text{and} \qquad a \equiv 1 \bmod m.$$

Such a number a exists and its unique, by the Chinese remainder theorem (Theorem 52.1.6$_{\text{p337}}$). In particular, let $m = \prod_{i=2}^{t} p_i^{k_i}$, and observe that, for all i, we have $a \equiv 1 \pmod{p_i}$, as $p_i \mid m$. As such, writing the Jacobi symbol explicitly, we have

$$\left[\!\left[a \mid n \right]\!\right] = \left[\!\left[a \mid q \right]\!\right] \prod_{i=2}^{t} (a \mid p_i)^{k_i} = \left[\!\left[a \mid q \right]\!\right] \prod_{i=2}^{t} (1 \mid p_i)^{k_i} = \left[\!\left[a \mid q \right]\!\right] \prod_{i=2}^{t} 1 = \left[\!\left[a \mid q \right]\!\right] = \left[\!\left[g \mid q \right]\!\right].$$

since $a \equiv g \pmod{q}$, and Lemma 52.1.32$_{\text{p344}}$ (C). At this point there are two possibilities:

(A) If $k_1 = 1$, then $q = p_1$, and $\left[\!\left[g \mid q \right]\!\right] = (g \mid q) = g^{(q-1)/2} \pmod{q}$. But g is a generator of $\mathbb{Z}_q^*$, and its order is $q - 1$. As such $g^{(q-1)/2} \equiv -1 \pmod{q}$, see Definition 52.1.23$_{\text{p342}}$. We conclude that $\left[\!\left[a \mid n \right]\!\right] = -1$. If we assume that $J_n = \mathbb{Z}_n^*$, then $\left[\!\left[a \mid n \right]\!\right] \equiv a^{(n-1)/2} \equiv -1 \pmod{n}$. Now, as $m \mid n$, we have

$$a^{(n-1)/2} \equiv_m \left(a^{(n-1)/2} \bmod n \right) \bmod m \equiv_m -1.$$

But this contradicts the choice of a as $a \equiv 1 \pmod{m}$.

(B) If $k_1 > 1$ then $q = p_1^{k_1}$. Arguing as above, we have that $\left[\!\left[a \mid n \right]\!\right] = (-1)^{k_1}$. Thus, if we assume that $J_n = \mathbb{Z}_n^*$, then $a^{(n-1)/2} \equiv -1 \pmod{n}$ or $a^{(n-1)/2} \equiv 1 \pmod{n}$. This implies that $a^{n-1} \equiv 1 \pmod{n}$. Thus, $a^{n-1} \equiv 1 \pmod{q}$.

Now $a \equiv g \bmod q$, and thus $g^{n-1} \equiv 1 \pmod{q}$. This implies that the order of g in $\mathbb{Z}_q^*$ must divide $n - 1$. That is $\operatorname{ord}(g) = \phi(q) \mid n - 1$. Now, since $k_1 \geq 2$, we have that $p_1 \mid \phi(q) = \left(p_1^{k_1}\right)(p_1 - 1)$, see Lemma 52.1.8$_{\text{p337}}$. We conclude that $p_1 \mid n - 1$ and $p_1 \mid n$, which is of course impossible, as $p_1 > 1$.

We conclude that J_n must be a proper subgroup of $\mathbb{Z}_n^*$, but, by Lemma 52.1.11$_{\text{p338}}$, it must be that $|J_n| \mid \left|\mathbb{Z}_n^*\right|$. But this implies that $|J_n| \leq \left|\mathbb{Z}_n^*\right|/2$. ■

52.2. Primality testing

The primality test is now easy[②]. Indeed, given a number n, first check if it is even (duh!). Otherwise, randomly pick a number $r \in \{2, \ldots, n-1\}$. If $\gcd(r, n) > 1$ then the number is composite. Otherwise, check if $r \in J_n$ (see Eq. (52.2)$_{\text{p346}}$), by computing $x = [\![r \mid n]\!]$ in polynomial time, see Section 52.1.3.4$_{\text{p345}}$, and $x' = a^{(n-1)/2} \bmod n$. (see Lemma 52.1.7$_{\text{p337}}$). If $x = x'$ then the algorithm returns is prime, otherwise it returns it is composite.

Theorem 52.2.1. *Given a number n, and a parameter $\delta > 0$, there is a randomized algorithm that, decides if the given number is prime or composite. The running time of the algorithm is $O((\log n)^c \log(1/\delta))$, where c is some constant. If the algorithm returns that n is composite then it is. If the algorithm returns that n is prime, then is wrong with probability at most δ.*

Proof: Run the above algorithm $m = O(\log(1/\delta))$ times. If any of the runs returns that it is composite then the algorithm return that n is composite, otherwise the algorithms returns that it is a prime.

If the algorithm fails, then n is a composite, and let $r_1, \ldots, r_m$ be the random numbers the algorithm picked. The algorithm fails only if $r_1, \ldots, r_m \in J_n$, but since $|J_n| \leq \left|\mathbb{Z}_n^2\right|/2$, by Lemma 52.1.35$_{\text{p346}}$, it follows that this happens with probability at most $\left(|J_n| / \left|\mathbb{Z}_n^2\right|\right)^m \leq 1/2^m \leq \delta$, as claimed. ■

52.2.1. Distribution of primes

In the following, let $\pi(n)$ denote the number of primes between 1 and n. Here, we prove that $\pi(n) = \Theta(n/\log n)$.

Lemma 52.2.2. *Let Δ be the product of all the prime numbers p, where $m < p \leq 2m$. We have that $\Delta \leq \binom{2m}{m}$.*

Proof: Let X be the product of the all composite numbers between m and $2m$, we have

$$\binom{2m}{m} = \frac{2m \cdot (2m-1) \cdots (m+2) \cdot (m+1)}{m \cdot (m-1) \cdots 2 \cdot 1} = \frac{X \cdot \Delta}{m \cdot (m-1) \cdots 2 \cdot 1}.$$

Since none of the numbers between 2 and m divides any of the factors of Δ, it must be that the number $\frac{X}{m \cdot (m-1) \cdots 2 \cdot 1}$ is an integer number, as $\binom{2m}{m}$ is an integer. Therefore, $\binom{2m}{m} = c \cdot \Delta$, for some integer $c > 0$, implying the claim. ■

Lemma 52.2.3. *The number of prime numbers between m and $2m$ is $O(m/\ln m)$.*

Proof: Let us denote all primes between m and $2m$ as $p_1 < p_2 < \cdots < p_k$. Since $p_1 \geq m$, it follows from Lemma 52.2.2 that $m^k \leq \prod_{i=1}^{k} p_i \leq \binom{2m}{m} \leq 2^{2m}$. Now, taking log of both sides, we have $k \lg m \leq 2m$. Namely, $k \leq 2m/\lg m$. ■

Lemma 52.2.4. $\pi(n) = O(n/\ln n)$.

Proof: Let the number of primes less than n be $\Pi(n)$, then by Lemma 52.2.3, there exist some positive constant C, such that for all $\forall n \geq N$, we have $\Pi(2n) - \Pi(n) \leq C \cdot n/\ln n$. Namely, $\Pi(2n) \leq C \cdot n/\ln n + \Pi(n)$. Thus, $\Pi(2n) \leq \sum_{i=0}^{\lceil \lg n \rceil} \left(\Pi\left(2n/2^i\right) - \Pi\left(2n/2^{i+1}\right)\right) \leq \sum_{i=0}^{\lceil \lg n \rceil} C \cdot \frac{n/2^i}{\ln(n/2^i)} = O\left(\frac{n}{\ln n}\right)$, by observing that the summation behaves like a decreasing geometric series. ■

[②] One could even say "trivial" with heavy Russian accent.

Lemma 52.2.5. *For integers m, k and a prime p, if $p^k \mid \binom{2m}{m}$, then $p^k \le 2m$.*

Proof: Let $T(p,m)$ be the number of times p appear in the prime factorization of $m!$. Formally, $T(p,m)$ is the highest number k such that p^k divides $m!$. We claim that $T(p,m) = \sum_{i=1}^{\infty} \left\lfloor m/p^i \right\rfloor$. Indeed, consider an integer $\beta \le m$, such that $\beta = p^t\gamma$, where γ is an integer that is not divisible by p. Observe that β contributes exactly to the first t terms of the summation of $T(p,m)$ – namely, its contribution to $m!$ as far as powers of p is counted correctly.

Let α be the maximum number such that p^α divides $\binom{2m}{m} = \frac{2m!}{m!m!}$. Clearly,

$$\alpha = T(p,2m) - 2T(p,m) = \sum_{i=1}^{\infty} \left(\left\lfloor \frac{2m}{p^i} \right\rfloor - 2\left\lfloor \frac{m}{p^i} \right\rfloor \right).$$

It is easy to verify that for any integers x, y, we have that $0 \le \left\lfloor \frac{2x}{y} \right\rfloor - 2\left\lfloor \frac{x}{y} \right\rfloor \le 1$. In particular, let k be the largest number such that $\left(\left\lfloor \frac{2m}{p^k} \right\rfloor - 2\left\lfloor \frac{m}{p^k} \right\rfloor \right) = 1$, and observe that $T(p,2m) \le k$ as only the proceedings $k-1$ terms might be non-zero in the summation of $T(p,2m)$. But this implies that $\left\lfloor 2m/p^k \right\rfloor \ge 1$, which implies in turn that $p^k \le 2m$, as desired. ■

Lemma 52.2.6. $\pi(n) = \Omega(n/\ln n)$.

Proof: Assume $\binom{2m}{m}$ have k prime factors, and thus can be written as $\binom{2m}{m} = \prod_{i=1}^{k} p_i^{n_i}$, By Lemma 52.2.5, we have $p_i^{n_i} \le 2m$. Of course, the above product might not include some prime numbers between 1 and $2m$, and as such k is a lower bound on the number of primes in this range; that is, $k \le \pi(2m)$. This implies $\frac{2^{2m}}{2m} \le \binom{2m}{m} \le \prod_{i=1}^{k} 2m = (2m)^k$. By taking lg of both sides, we have $\frac{2m - \lg(2m)}{\lg(2m)} \le k \le \pi(2m)$. ■

We summarize the result.

Theorem 52.2.7. *Let $\pi(n)$ be the number of distinct prime numbers between 1 and n. We have that $\pi(n) = \Theta(n/\ln n)$.*

52.3. Bibliographical notes

Miller [Mil76] presented the primality testing algorithm which runs in deterministic polynomial time but relies on Riemann's Hypothesis (which is still open). Later on, Rabin [Rab80] showed how to convert this algorithm to a randomized algorithm, without relying on the Riemann's hypothesis.

This write-up is based on various sources – starting with the description in [MR95], and then filling in some details from various sources on the web.

What is currently missing from the write-up is a description of the RSA encryption system. This would hopefully be added in the future. There are of course typos in these notes – let me know if you find any.

References

[Mil76] G. L. Miller. Riemann's hypothesis and tests for primality. *J. Comput. Sys. Sci.*, 13(3): 300–317, 1976.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

[Rab80] M. O. Rabin. Probabilistic algorithm for testing primality. *J. Number Theory*, 12(1): 128–138, 1980.

Chapter 53

Talagrand's Inequality

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> At an archaeological site I saw fragments of precious vessels, well cleaned and groomed and oiled and spoiled. And beside it I saw a heap of discarded dust which wasn't even good for thorns and thistles to grow on.
>
> I asked: What is this gray dust which has been pushed around and sifted and tortured and then thrown away?
>
> I answered in my heart: This dust is people like us, who during their lifetime lived separated from copper and gold and marble stones and all other precious things - and they remained so in death. We are this heap of dust, our bodies, our souls, all the words in our mouths, all hopes.

At an archaeological site, Yehuda Amichai

53.1. Introduction

Here, we want to prove a strong concentration inequality that is stronger than Azuma's inequality because it is independent of the underlying dimension of the process. This inequality is quite subtle, so we will need a quite elaborate way to get to it – be patient.

53.1.1. Talangrand's inequality, and the T-distance

For two numbers x, y, let $[x \neq y]$ be 1 if $x \neq y$, and 0 otherwise. For two points $\mathsf{p} = (p_1, \ldots, p_d), \mathsf{u} = (q_1, \ldots, q_d) \in \mathbb{R}^d$, let $H(\mathsf{p}, \mathsf{u})$ be the binary vector in $\{0,1\}^n$ that encodes the coordinates where they are different. Formally, we have

$$H(\mathsf{p}, \mathsf{u}) = \Big([p_1 \neq q_1], [p_2 \neq q_2], \ldots, [p_d \neq q_d]\Big). \tag{53.1}$$

For example $H((1,2,3),(0.1,2,-1)) = (1,0,1)$. Given a set $S \subseteq \mathbb{R}^d$, and a point $\mathsf{p} \in \mathbb{R}^d$, let

$$H(\mathsf{p}, S) = \{H(\mathsf{p}, \mathsf{u}) \mid \mathsf{u} \in S\}. \tag{53.2}$$

To understand this mysterious set, consider a point $\mathsf{p} \in \mathbb{R}^d$. If $\mathsf{p} \in S$, then $(0, \ldots, 0) = H(\mathsf{p}, \mathsf{p}) \in H(\mathsf{p}, S)$ (which would be an uninteresting case). Otherwise, every binary point $\mathsf{x} \in H(\mathsf{p}, S)$ specifies which coordinates in p one has to change, so that one can move to a point that belongs to S.

A natural measure of the distance of p from S, is then to ask for the vector that minimizes the *Hamming distance* from the origin to $H(\mathsf{p}, S)$ – that is, the minimum number of coordinates one has to change in p to get to a point of S.

This distance measure is not informative. Think about $H(\mathsf{p}, S) = \{(0,1)\}$ and $H(\mathsf{p}, S') = \{(0,1), (1,0)\}$. In this case, the Hamming measure would rank both sets as being of equal quality (i.e., 1). But clearly, S' is closer – there are two different ways to get from p to some point of S' by changing a single coordinate.

To capture this intuition, we consider the ***convex-hull*** of these sets:

$$C(\mathsf{p}, S) = \mathcal{CH}(\{H(\mathsf{p}, \mathsf{u}) \mid \mathsf{u} \in S\}).$$

And the corresponding ***T-distance***

$$\rho(\mathsf{p}, S) = \min_{\mathsf{u} \in C(\mathsf{p}, S)} \|\mathsf{u}\|. \tag{53.3}$$

Observation 53.1.1. *An easy upper bound on the T-distance of p to a set S (i.e., $\rho(\mathsf{p}, S)$) is the minimum number of coordinates one has to change in p to get to a point in S. As the next example shows, however, things are more subtle – if there are many different ways to get from p to a point in S, then the T-distance is going to be significantly smaller.*

Example 53.1.2. It would be useful to understand this somewhat mysterious T-distance. To this end, consider the ball in $\mathbb{R}^d$ of radius $100d$ centered in the origin, denote by $\mathbf{b}$, and let $S = \partial\mathbf{b}$ be its boundary sphere. For a point $\mathsf{p} \in \text{int}\mathbf{b}$, we have that

$$H = H(\mathsf{p}, S) = \{0, 1\}^d \setminus \{(0, 0, \ldots, 0)\}.$$

As such $C = H(\mathsf{p}, S)$ is the convex-hull of all the hypercube vertices, excluding the origin. it is easy to check that the closest point in C to the origin is the point $\mathsf{u} = (1/d, 1/d, \ldots, 1/d)$, As such, we have that $\rho(\mathsf{p}, S) = \|\mathsf{u}\| = \sqrt{1/d} = 1/\sqrt{d}$.

In particular, by monotonicity this implies that for any set T in $\mathbb{R}^d$ we have that $\rho(\mathsf{p}, T)$ is either 0 (i.e., $\mathsf{p} \in T$), or alternatively, $\rho(\mathsf{p}, T) \geq 1/\sqrt{d}$. Similarly, $\rho(\mathsf{p}, T) \leq \sqrt{d}$ as this is the maximum distance from the origin to any vertex of the hypercube $\{0, 1\}^d$.

As a concrete example, for the set $S = \partial\mathbf{b}$, and the point $\mathsf{p} = (200d, \ldots, 200d)$, we have $\rho(\mathsf{p}, S) = \sqrt{d}$.

In the following, think about the dimension d as being quite large. As such, the distance $1/\sqrt{d}$ is quite small. In particular, for a set $S \subseteq \mathbb{R}^d$, let

$$S_t = \left\{\mathsf{p} \in \mathbb{R}^d \;\middle|\; \rho(\mathsf{p}, S) \leq t\right\},$$

be the expansion of S by including points that are in distance at most t from S in the T-distance.

Since we interested in probability here, consider $\mathbb{R}^d$ to be the product of d probability spaces. Formally, let Ω_i be a probability space, and consider the product probability space $\Omega = \prod_{i=1}^{d} \Omega_i$. As we are given a probability measure on each Ω_i, this defines a natural probability measure on Ω. That is, a point from Ω is generated by packing each of its coordinates independently from Ω_i. for $i = 1, \ldots, d$.

The ***volume*** of a set $S \subseteq \Omega$ is thus $\mathbb{P}[S]$. We are now ready to state Talagrand inequality (not that it is going to help us much).

Theorem 53.1.3 (Talagrand's inequality). *For any set $S \subseteq \Omega$, we have*

$$\mathbb{P}[S]\,\mathbb{P}[\overline{S_t}] = \mathbb{P}[S](1 - \mathbb{P}[S_t]) \leq \exp(-t^2/4).$$

Example 53.1.4. To see why this inequality interesting, consider $\Omega = [0, 100]^d$ with uniform distribution on each coordinate. The probability measure of a set $S \subseteq \Omega$ is $\mathbb{P}[S] = \text{vol}(S)/100^d$. Let

$$S = \left\{\mathsf{p} = (p_1, \ldots, p_d) \in [0, 100]^d \;\middle|\; \sum_i p_i \leq \frac{100d}{2}\right\}.$$

It is easy to verify that $\text{vol}(S) = \text{vol}\Big([0, 100]^d\Big)/2$. Let $t = 4\sqrt{\ln d}$, and consider the set $\overline{S_t}$. Intuitively, and not quite correctly, it the set of all points in $[0, 100]^d$, such that one needs to change more than $4 \ln d$ coordinates before one can get to a point of S. These points are t-far from being in S.

By Talagrand inequality, we have that $\mathbb{P}\left[\overline{S_t}\right]/2 = \mathbb{P}[S](1-\mathbb{P}[S_t]) \leq \exp(-t^2/4) = 1/d^4$. Namely, only a tiny fraction of the cube is more than T-distance $4\sqrt{\ln d}$ from S!

Let us try to restate this – for any set S that is half the volume of the hypercube $[0,100]^d$, the set of points in T-distance $\leq 4\sqrt{\ln d}$ in this hypercube is small.

53.1.2. On the way to proving Talagrand's inequality

The following helper result is the core of the proof of Talagrand's inequality. The reader might want to skip reading the proof of this claim, at least at first reading.

Theorem 53.1.5. *For any set $S \subseteq \Omega = \prod_{i=1}^{d} \Omega_i$, we have*

$$\mathbb{E}\left[\exp\left(\frac{\rho^2(\mathsf{p},S)}{4}\right)\right] = \int_{\mathsf{p}\in\Omega} \exp\left(\frac{\rho^2(\mathsf{p},S)}{4}\right) d\mathsf{p} \leq \frac{1}{\mathbb{P}[S]}.$$

Proof: The proof is by induction on the dimension d. For $d=1$, then $\rho(\mathsf{p},S)=0$ if $\mathsf{p}\in S$, and $\rho(\mathsf{p},S)=1$ if $\mathsf{p}\notin S$. As such, we have

$$\gamma = \mathbb{E}\left[\exp\left(\frac{\rho^2(\mathsf{p},S)}{4}\right)\right] = e^{0^2/4}\,\mathbb{P}[S] + e^{1^2/4}(1-\mathbb{P}[S]) = \mathbb{P}[S] + e^{1/4}(1-\mathbb{P}[S]) = f(\mathbb{P}[S]),$$

where $f(x) = x + e^{1/4}(1-x)$. An easy argument (see Tedium 53.1.6) shows that $f(x) \leq 1/x$, which implies that $\gamma = f(\mathbb{P}[S]) \leq 1/\mathbb{P}[S]$, as claimed.

For $d = n+1$, let $O = \prod_{i=1}^{d} \Omega_i$, and $\mathcal{N} = \Omega_{d+1}$. Clearly, $\Omega = \prod_{i=1}^{d+1} \Omega_i = O \times \mathcal{N}$.

$$S_O = \{\mathsf{p} \in O \mid (\mathsf{p}, y) \in S, \text{ for some } y \in \mathcal{N}\}.$$

For $\nu \in \mathcal{N}$, let

$$S(\nu) = \{\mathsf{p} \in O \mid (\mathsf{p}, \nu) \in S\} \subseteq S_O.$$

Given a point $\mathsf{z} = (\mathsf{p}, y) \in \Omega$, we can get to a point in S, either by changing the new coordinate and then moving inside the old space O, or alternatively, keeping the new coordinate ν fixed and moving only in the old coordinates. In particular, we have that if

$$\begin{aligned} s \in H(\mathsf{p}, S_O) \subseteq \{0,1\}^d \quad &\Longrightarrow \quad (s,1) \in H(\mathsf{z},S) \qquad \text{(see Eq. (53.2))} \\ s' \in H(\mathsf{p}, S(\nu)) \quad &\Longrightarrow \quad (s',0) \in H(\mathsf{z},S). \end{aligned}$$

And similarly, for the corresponding convex-hulls, we have

$$s \in C(\mathsf{p}, S_O) \implies (s,1) \in C(\mathsf{z},S) \qquad \text{and} \qquad s' \in C(\mathsf{p}, S(\nu)) \implies (s',0) \in C(\mathsf{z},S).$$

In particular, for s, s' as above, we have (by convexity) that for any $\lambda \in [0,1]$, the point

$$h(\lambda) = (1-\lambda)(s,1) + \lambda(s',0) = ((1-\lambda)s + \lambda s', 1-\lambda) \in C(\mathsf{z},S) \subseteq [0,1]^{d+1}.$$

The function $\widehat{h}(\lambda) = \|(1-\lambda)s + \lambda s'\|^2$ is convex, see Tedium 53.1.7. We thus have

$$\rho^2(\mathsf{z},S) = \left(\min_{\mathsf{p}\in C(\mathsf{z},S)} \|\mathsf{p}\|^2\right) \leq \|h(\lambda)\|^2 = \|(1-\lambda)s + \lambda s'\|^2 + (1-\lambda)^2 \leq (1-\lambda)\|s\|^2 + \lambda\|s'\|^2 + (1-\lambda)^2.$$

We are still at the liberty of choosing s and s'. Let s be the point realizing $\rho(\mathsf{p}, S_O)$ – this is the closest point in $C(\mathsf{p}, S)$ to the origin (i.e., $\|s\| = \rho(\mathsf{p}, S_O)$). Similarly, let s' be the point realizing $\rho(\mathsf{p}, S(\nu))$. Plugging these two points into the above inequality, we have

$$\rho^2(\mathsf{z}, S) \leq (1-\lambda)\rho(\mathsf{p}, S_O)^2 + \lambda\rho(\mathsf{p}, S(\nu))^2 + (1-\lambda)^2.$$

Now, fix ν, and ride the following little integral:

$$\begin{aligned}
F(\nu) &= \int_{\mathsf{p}} \exp\left(\frac{\rho^2((\mathsf{p},\nu), S)}{4}\right) \leq \int_{\mathsf{p}} \exp\left(\frac{(1-\lambda)\rho(\mathsf{p}, S_O)^2 + \lambda\rho(\mathsf{p}, S(\nu))^2 + (1-\lambda)^2}{4}\right) \\
&\leq e^{(1-\lambda)^2/4} \int_{\mathsf{p}} \exp\left(\frac{1}{4}\rho(\mathsf{p}, S_O)^2\right)^{1-\lambda} \exp\left(\frac{1}{4}\rho(\mathsf{p}, S(\nu))^2\right)^{\lambda} \\
&\leq e^{(1-\lambda)^2/4} \left[\int_{\mathsf{p}} \exp\left(\frac{1}{4}\rho(\mathsf{p}, S_O)^2\right)\right]^{(1-\lambda)} \left[\int_{\mathsf{p}} \exp\left(\frac{1}{4}\rho(\mathsf{p}, S(\nu))^2\right)\right]^{\lambda} && \text{(by Hölder's ineq (53.4))} \\
&\leq e^{(1-\lambda)^2/4} \left(\frac{1}{\mathbb{P}[S_O]}\right)^{(1-\lambda)} \left(\frac{1}{\mathbb{P}[S(\nu)]}\right)^{\lambda} = e^{(1-\lambda)^2/4} \frac{1}{\mathbb{P}[S_O]} \left(\frac{\mathbb{P}[S(\nu)]}{\mathbb{P}[S_O]}\right)^{-\lambda} && \text{(induction)} \\
&= \frac{1}{\mathbb{P}[S_O]} \cdot e^{(1-\lambda)^2/4} r^{-\lambda}, && \text{for } r = \frac{\mathbb{P}[S(\nu)]}{\mathbb{P}[S_O]}
\end{aligned}$$

Observe that $\mathbb{P}[S_O] \geq \mathbb{P}[S(\nu)]$, and thus $r \leq 1$. To minimize the above, consider the function $f_3(\lambda, r) = \exp((1-\lambda)^2/4) r^{-\lambda}$. Easy calculation shows that $f_3(\lambda, r)$ is minimized, for a fixed r, by choosing

$$\lambda(r) = \begin{cases} 1 + 2\ln r & r \in [e^{-1/2}, 1] \\ 0 & r \in [0, e^{-1/2}] \end{cases},$$

see Tedium 53.1.9 (A). Furthermore, for this choice of λ, easy calculations shows that $f_4(r) = f_r(\lambda(r), r) \leq 2 - r$, see Tedium 53.1.9 (B). As such, we have

$$F(\nu) \leq \frac{1}{\mathbb{P}[S_O]} f_4(r) \leq \frac{1}{\mathbb{P}[S_O]}\left(2 - \frac{\mathbb{P}[S(\nu)]}{\mathbb{P}[S_O]}\right)$$

We remind the reader that our purpose is to bound

$$\begin{aligned}
\int_{\mathsf{z}} \exp\left(\frac{\rho^2(\mathsf{z}, S)}{4}\right) &= \int_{\nu \in \mathcal{N}} \int_{\mathsf{p} \in O} \exp\left(\frac{\rho^2((\mathsf{p},\nu), S)}{4}\right) \leq \int_{\nu \in \mathcal{N}} F(\nu) \leq \int_{\nu \in \mathcal{N}} \frac{1}{\mathbb{P}[S_O]}\left(2 - \frac{\mathbb{P}[S(\nu)]}{\mathbb{P}[S_O]}\right) \\
&= \frac{1}{\mathbb{P}[S_O]}\left(2 - \frac{\int_{\nu \in \mathcal{N}} \mathbb{P}[S(\nu)]}{\mathbb{P}[S_O]}\right) = \frac{1}{\mathbb{P}[S_O]}\left(2 - \frac{\mathbb{P}[S]}{\mathbb{P}[S_O]}\right) = \frac{1}{\mathbb{P}[S]} \cdot \frac{\mathbb{P}[S]}{\mathbb{P}[S_O]}\left(2 - \frac{\mathbb{P}[S]}{\mathbb{P}[S_O]}\right) \leq \frac{1}{\mathbb{P}[S]},
\end{aligned}$$

since for $x = \mathbb{P}[S]/\mathbb{P}[S_O]$, we have $x(2-x) \leq 1$, for any value of x (see Tedium 53.1.10). ■

53.1.2.1. The low level details used in the above proof

Tedium 53.1.6. Let $f(x) = x + e^{1/4}(1-x)$. We claim that, for $x \in (0, 1]$, $f(x) \leq 1/x$. Indeed, set $g(x) = 1/x$, and observe that $f(1) = 1 = 1/1 = g(1)$. We have that $f'(x) = e^{1/4} - 1 \approx -0.284$ and $g'(x) = -1/x^2$. In particular, $g'(x) \leq f'(x)$, for all $x \in (0, 1)$. Since and $f(1) = g(1)$. it follows that $f(x) \leq g(x)$, for $x \in (0, 1]$.

Tedium 53.1.7. For any $\mathsf{p}, \mathsf{u} \in \mathbb{R}^d$, the function $f(\lambda) = \|(1-\lambda)\mathsf{p} + \lambda\mathsf{u}\|^2$ is convex. Indeed, let $f_i(\lambda) = ((1-\lambda)p_i + \lambda q_i)^2$, for $i = 1, \ldots, d$. Observe that $f(\lambda) = \sum_i f_i(\lambda)$, and as such it is sufficient to prove that f_i is convex. We have $f_i'(\lambda) = 2(q_i - p_i)((1-\lambda)p_i + \lambda q_i)$, and $f_i''(\lambda) = 2(q_i - p_i)^2 > 0$, which implies convexity.

Fact 53.1.8 (Hölder's inequality.). *Let $p, q \geq 1$ be two numbers such that $1/p + 1/q = 1$. Then, for any two functions f, g, we have $\|fg\|_1 \leq \|f\|_p \|g\|_q$. Explicitly, stated as integrals, Hölder's inequality is $\int |f(x)g(x)|\mathrm{d}x \leq \left(\int |f(x)|^p \mathrm{d}x\right)^{1/p}\left(\int |f(x)|^q \mathrm{d}x\right)^{1/q}$. In particular, for $\lambda \in (0,1)$, $p = 1/(1-\lambda)$ and $q = 1/\lambda$, we have that*

$$\int \left|f^{\lambda}(x) g^{1-\lambda}(x)\right| \mathrm{d}x \leq \left(\int |f(x)|\mathrm{d}x\right)^{1-\lambda}\left(\int |f(x)|\mathrm{d}x\right)^{\lambda}. \tag{53.4}$$

Tedium 53.1.9. (A) We need to find the minimum of the following function $f(\lambda) = \exp((1-\lambda)^2/4) r^{-\lambda} = \exp((1-\lambda)^2/4 - \lambda \ln r)$. We have $f'(\lambda) = f_3(\lambda)((1-\lambda)/2 - \ln r)$. Solving for $f'(\lambda) = 0$, we have $(1-\lambda)/2 - \ln r = 0 \implies 1 - \lambda = 2\ln r 0 \implies \lambda = 1 - 2\ln r$, which works as long as $r \geq e^{-1/2}$. Otherwise, we set $\lambda = 0$.

(B) For $r \leq e^{-1/2}$, we have, by the above, that $f(0) = e^{1/4} \approx 1.28 \leq 1.39 \approx 2 - e^{-1/2} \leq 2 - r$. For $r > e^{-1/2}$, by the above, $\lambda = 1 - 2\ln r$, and thus

$$g(r) = f(\lambda) = \exp((1-\lambda)^2/4 - \lambda \ln r) = \exp((2\ln r)^2/4 + (1 - 2\ln r)\ln r) = \exp(\ln r - \ln^2 r) \leq 1 \leq 2 - r,$$

since $\ln r - \ln^2 r \leq \ln r \leq 0$, for $r \in (0, 1]$.

Tedium 53.1.10. The function $f(x) = x(2-x) = 2x - x^2$ is a parabola with a maximum at $2x = 2 \implies x = 1 \implies \forall y \quad f(y) \leq f(1) = 1$.

53.1.3. Proving Talagrand's inequality

Proving Talagrand's inequality is now easy peasy.

Talagrand's inequality restatement (Theorem 53.1.3). For any set $S \subseteq \Omega$, we have

$$\mathbb{P}[S]\,\mathbb{P}[\overline{S_t}] = \mathbb{P}[S](1 - \mathbb{P}[S_t]) \leq \exp(-t^2/4).$$

Proof: Consider a random point $\mathsf{p} \in \Omega$. We are interested in the probability $\mathsf{p} \notin S_t$. To this end, consider the random variable $X = \rho(\mathsf{p}, S)$. By definition, $\mathsf{p} \in \overline{S_t} \iff X \geq t$. As such, by Markov's inequality, we have

$$\mathbb{P}\left[\overline{S_t}\right] = \mathbb{P}[X \geq t] = \mathbb{P}\left[\exp\left(X^2/4\right) \geq \exp\left(t^2/4\right)\right] \leq \frac{\mathbb{E}[\exp\left(X^2/4\right)]}{\exp(t^2/4)} \leq \frac{\exp\left(-t^2/4\right)}{\mathbb{P}[S]},$$

by Theorem 53.1.5. ■

53.2. Concentration via certification

Example 53.2.1. Consider the process of throwing m balls into n bins. The ith ball X_i is uniformly distributed in $\Omega_i = [\![n]\!]$. For $\mathbf{x} = (X_1, \ldots, X_m) \in \Omega = \prod_{i=1}^m \Omega_i$, let $h(\mathbf{x})$ be the number of bins that are not empty. If $h(\mathbf{x}) \geq k$, then there is a set $I = \{i_1, \ldots, i_k\}$ of k indices, such that for any two distinct $i, j \in I$, we have that $X_i \neq X_j$.

Namely, I is a "compact" proof/certificate that $h(\mathbf{x}) \geq k$. Furthermore, if for $\mathbf{y} = (Y_1, \ldots, Y_m) \in \Omega$ we have that $X_\alpha = Y_\alpha$, for all $\alpha \in I$, then $h(\mathbf{y}) \geq k$. Here, the certificate for a value k, was a set of size k.

Definition 53.2.2. Let $\Omega = \prod_{i=1}^m \Omega_i$. For a function $h : \Omega \to \mathbb{N}$, it is ***$f$-certifiable***, for a function $f : \mathbb{N} \to \mathbb{N}$, if whenever $h(\mathbf{x}) \geq k$, there exists a set $I \subseteq [\![m]\!]$, with $|I| \leq f(k)$, such that, for any $\mathbf{y} \in \Omega$, if $\mathbf{y}$ agree with $\vec{x}$ on the coordinates of I, then $h(\vec{y}) \geq k$.

Example 53.2.3. In Example 53.2.1, the function h (i.e., number of bins that are not empty) is f-certifiable, where $f(k) = k$.

Example 53.2.4. Consider the random graph $\mathsf{G}(n, p)$ over n vertices, created by picking every edge with probability p. One can interpret such a graph as a random binary vector with $\binom{n}{2}$ coordinates, where the ith coordinate is 1 $\iff$ the ith edge is in the graph (for some canonical ordering of all possible $\binom{n}{2}$ edges).

A ***triangle*** in a graph G is a triple of vertices i, j, k, such that $ij, jk, ki \in \mathsf{E}(\mathsf{G})$. For a graph G, let $h(\mathsf{G})$ be the number of distinct triangles in G. In the above interpretation as a graph as a vector $\mathbf{x} \in \{0,1\}^{\binom{n}{2}}$, it is easy to verify that if $h(\mathsf{G}) \geq k$ then it can be certified by $3k$ coordinates. As such, the number of triangles in a graph is f-certifiable, for $f(k) = 3k$.

Note, that the certificate is only for the lower bound on the value of the function.

We need the following reinterpretation of the T-distance.

Lemma 53.2.5. *Consider a set $S \subseteq \mathbb{R}^d$ and a point $\mathsf{p} \in \mathbb{R}^d$. We have that $\rho(\mathsf{p}, S) \leq t \iff$ for all $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, with $\|\mathbf{x}\| = 1$, there exists $\mathbf{h} \in H(\mathsf{p}, S)$, such that $\langle \mathbf{x}, \mathbf{h} \rangle \leq t$.*

Proof: The quantity $\ell = \rho(\mathsf{p}, S)$ is the distance from the origin to the convex polytope $C(\mathsf{p}, S)$. In particular, let $\mathbf{y}$ be the closest point to the origin in this polytope, and observe that $\ell = \|\mathbf{y}\| = \langle \mathbf{y}, \mathbf{y}/\|\mathbf{y}\| \rangle$. In particular, for any other vector $\mathbf{x}$, with $\|\mathbf{x}\| = 1$, we have $\langle \mathbf{y}, \mathbf{x} \rangle \leq \langle \mathbf{y}, \mathbf{y}/\|\mathbf{y}\| \rangle \leq \ell$. Since $\mathbf{y}$ is in the convex-hull of $H(\mathsf{p}, S)$, it follows that there is $\mathbf{h} \in H(\mathsf{p}, S)$ such that $\langle \mathbf{y}, \mathbf{x} \rangle \leq \langle \mathbf{h}, \mathbf{x} \rangle \leq \ell$.

As for the other direction, assume that $\ell = \rho(\mathsf{p}, S) > t$, and let $\mathbf{y} \in C(\mathsf{p}, S)$ be the point realizing this distance. Arguing as above, we have that for the direction $\mathbf{y}/\|\mathbf{y}\|$, and any vertex $\mathbf{h} \in H(\mathsf{p}, S)$ we have that $\langle \mathbf{h}, \mathbf{x} \rangle \geq \langle \mathbf{h}, \mathbf{x} \rangle \geq \ell > t$. ■

Theorem 53.2.6. *Consider a probability space $\Omega = \prod_{i=1}^{m} \Omega_i$, and let $h : \Omega \to$ be 1-Lipschitz and f-certifiable, for some function f. Consider the random variable $X = h(\mathbf{x})$, for $\mathbf{x}$ picked randomly in Ω. Then, for any positive real numbers b and t, we have*

$$\mathbb{P}\Big[X \leq b - t\sqrt{f(b)}\Big]\mathbb{P}[X \geq b] \leq \exp(-t^2/4).$$

If h is k-Lipschitz then $\mathbb{P}\Big[X \leq b - tk\sqrt{f(b)}\Big]\mathbb{P}[X \geq b] \leq \exp(-t^2/4)$.

Proof: Set $S = \{\mathsf{p} \in \Omega \;\big|\; h(\mathsf{p}) < b - t\sqrt{f(b)}\}$. Consider a point u, such that $h(\mathsf{u}) \geq b$. Assume for the sake of contradiction that $\mathsf{u} \in S_t$. Let $I \subseteq [\![m]\!]$ be the certificate of size $\leq f(b)$ that $h(\mathsf{u}) \geq b$. And consider the vector $\mathbf{x} = (x_1, \ldots, x_d)$, such that $x_i = 1/\sqrt{|I|}$ if $i \in I$, and $x_i = 0$ otherwise. Observe that $\|\mathbf{x}\|^2 = |I|\,(1/|I|) = 1$, and thus $\|\mathbf{x}\| = 1$. By Lemma 53.2.5, there exists $\mathbf{h} \in H(\mathsf{u}, S)$, such that $\langle \mathbf{x}, \mathbf{h} \rangle \leq t$, since by assumption $\rho(\mathsf{u}, S) \leq t$. Let $\mathsf{v} \in S$ be the point realizing $\mathbf{h}$ – that is, $H(\mathsf{p}, \mathsf{v}) = \mathbf{h}$.

Let $J \subseteq I$ be the set of indices of coordinates that are in I, such that p and v differ on this coordinate. We have by the definition of $\mathbf{x}$, that $|J| / \sqrt{|I|} \leq \langle \mathbf{x}, \mathbf{h} \rangle \leq t$, which implies that $|J| \leq t\sqrt{|I|} \leq t\sqrt{f(b)}$.

Let u' be the point that agrees with u on the coordinates of I, and agrees with v on the other coordinates. The points u' and v disagree only on coordinates in I, but such coordinates of disagreement are exactly the coordinates (in I) where u disagrees with v – which is the set J of coordinates. As such, by the 1-Lipschitz condition, we have that

$$h(\mathsf{v}) \geq h(\mathsf{u}') - |J| \geq h(\mathsf{u}) - t\sqrt{f(b)},$$

but then, by the definition of S, we have $\mathsf{v} \notin S$, which is a contradiction as $\mathsf{v} \in S$.

We conclude that $\mathsf{u} \notin S_t \implies \mathsf{u} \in \overline{S_t}$. As such, we have $\mathbb{P}[X \geq b] \leq \mathbb{P}[\overline{S_t}]$. By Talagrand inequality, we have

$$\mathbb{P}[X < b - t\sqrt{f(b)}]\,\mathbb{P}[X \geq b] \leq \mathbb{P}[S]\,\mathbb{P}[\overline{S_t}] \leq \exp(-t^2/4).$$

The "<" on the left side can be replaced by "≤", as in the statement of the theorem, by using the value $t + \varepsilon$ instead of t, and taking the limit as $\varepsilon \to 0$.

The k-Lipschitz version follows by applying the above inequality to the function $h(\cdot)/k$. ■

53.3. Some examples

Definition 53.3.1. For a random variable $X \in \mathbb{R}$, let $\operatorname{med}(X)$ denote the maximum number m, such that $\mathbb{P}[X < m] \leq 1/2$ and $\mathbb{P}[X > m] \leq 1/2$. The number $\operatorname{med}(X)$ is the ***median*** of X.

53.3.1. Longest increasing subsequence

Let $\mathbf{x} = (X_1, \ldots, X_n)$ be a vector of n numbers picked randomly and uniformly from $[0, 1]$. Let $h(\mathbf{x})$ be the longest increasing subsequence in the associated sequence.

Lemma 53.3.2. *We have $h(\mathbf{x}) = \Theta(\sqrt{n})$ with high probability. Furthermore, for some constant c and any $t > 0$, we have that $\mathbb{P}\Big[|h(\mathbf{x}) - \operatorname{med}(h(\mathbf{x}))| - tcn^{1/4}\Big] \leq 4\exp(-t^2/4)$, Namely, the random variable $h(\mathbf{x})$ is strongly concentrated.*

Proof: Let Y_i be an indicator variable that is $1 \iff \mathbf{x}[i] \equiv x_{(i-1)\sqrt{n}+1}, \ldots, x_{i\sqrt{n}}$ contains a number in the interval $J(i) = [(i-1)/\sqrt{n}, i\sqrt{n}]$. We have $\mathbb{P}[Y_i = 1] = 1 - (1 - 1/\sqrt{n})^{\sqrt{n}} \geq 1 - 1/e \geq 1/2$, since $(1 - 1/m)^m \leq 1/e$. If Y_i happens, then we can take the number in $\mathbf{x}[i]$ that falls in $J(i)$, and add it to the generated sequence. As such, the length of the generated sequence, which is increasing, is $Y = \sum_{i=1}^{n} Y_i$. And in particular, $\mathbb{E}[Y] \geq \sqrt{n}/2$, and Chernoff's inequality implies that $\mathbb{P}\Big[Y \geq (1-\delta)\sqrt{n}/2\Big] \leq \exp(-\delta^2\sqrt{n}/8)$.

The upper bound is more interesting. The probability that a specific subsequence of t indices $i_1 < i_2 < \ldots < i_t$ form an increasing subsequence $X_{i_1} < X_{i_2} < \cdots < X_{i_t}$ is $1/t!$. As such, the expected number of such increasing sequences of length $\geq \ell$ is bounded by

$$\alpha = \sum_{t=\ell}^{n} \binom{n}{t}\frac{1}{t!} \leq \sum_{t=\ell}^{n} \left(\frac{ne}{t}\right)^t \frac{1}{t!} \leq \sum_{t=\ell}^{n} \left(\frac{ne}{t}\right)^t \frac{1}{(t/e)^t} = \sum_{t=\ell}^{n} \frac{n^t e^{2t}}{t^{2t}},$$

using Lemma 6.1.1. In particular, for $\ell = 4e\sqrt{n}$, we have

$$\alpha \leq \sum_{t=\ell}^{n} \frac{n^t e^{2t}}{\left(4e\sqrt{n}\right)^{2t}}, = \sum_{t=\ell}^{n} \frac{1}{4^{2t}} \leq 2/4^{8e\sqrt{n}} \ll 1.$$

By Markov's inequality, this implies that $\mathbb{P}\Big[h(\mathbf{x}) \geq 4e\sqrt{n}\Big] \leq 2/4^{8e\sqrt{n}}$.

The above readily implies that $\nu = \operatorname{med}(h(\mathbf{x})) = \Theta(\sqrt{n})$. Furthermore, $h(\mathbf{x})$ is $f(x) = x$ certifiable, and it is 1-Lipschitz. Theorem 53.2.6 now implies that

$$\mathbb{P}\Big[h(\mathbf{x}) \leq \nu - t\sqrt{\nu}\Big]/2 \leq \mathbb{P}\Big[h(\mathbf{x}) \leq \nu - t\sqrt{\nu}\Big]\mathbb{P}[X \geq \nu] \leq \exp(-t^2/4).$$

As $\nu = O(n^{1/4})$, we get the following (this requires some further tedious calculations which we omit).

$$\mathbb{P}\Big[\big|h(\mathbf{x}) - \nu\big| - tcn^{1/4}\Big] \leq 4\exp(-t^2/4),$$

where c is some constant. ■

53.3.2. Largest convex subset

A set of points P is in ***convex position*** if they are all vertices of the convex-hull of P.

Lemma 53.3.3. *Let P be a set of n points picked randomly and uniformly in the unit square $[0,1]^2$. Let Y be the size of the largest subset of point of P that are in convex position. Then, we have that $\mathbb{E}[Y] = \Omega(n^{1/3})$.*

Proof: Let $\mathsf{p} = (1/2, 1/2)$, and consider the regular N-gon Q, for $N = n^{1/3}$, that its vertices lie on the circle centered at p, and is of radius $r = 1/2$. Consider the triangle $\triangle_i$ formed by connecting three consecutive vertices $\mathsf{p}_{2i-1}, \mathsf{p}_{2i}, \mathsf{p}_{2i+1}$ of Q. We have that $\alpha = 2\pi/N$, and we pick n large enough, so that $\alpha \leq 1/4$. We remind the reader that $1 - x^2/4 \geq \cos x \geq 1 - x^2/2$, for $x \in (0, 1/4)$. As such, we have that $\alpha^2/4 \leq 1 - \cos\alpha \leq \alpha^2/2$. In particular, this implies that the height of $\triangle$ is $h = r(1 - \cos(\alpha))$, and we have $\alpha^2/8 = r\alpha^2/4 \leq h \leq r\alpha^2/2$.

Let $\ell = \|\mathsf{p}_{2i-1} - \mathsf{p}_{2i+1}\| = 2r\sin\alpha$, since $x/2 \leq \sin(x) \leq x$, we have that $\alpha/2 \leq \ell \leq \alpha$. As such, we have that

$$\operatorname{area}(\triangle_i) = h\ell/2 \geq (\alpha^2/8)(\alpha/2) = \alpha^3/16 = (2\pi/N)^3/16 = 8/n.$$

In particular, the probability that $\triangle_i$ does not contain a point of P is at most $(1 - \operatorname{area}(\triangle_i))^n \leq (1 - 8/n)^n \leq \exp(-8)$. We conclude that, in expectation, at least $(1 - \exp(-8))N/2$ triangles contains points of P. Selecting a point of P from each such triangle results in a convex subset, which implies the claim. ■

It is not hard to show that $Y = \Omega(n^{1/3})$, with high probability, see Exercise 53.5.1. This readily implies that $\operatorname{med}(Y) = \Omega(n^{1/3})$. It is significantly harder, but known, that $\mathbb{E}[Y] = O(n^{1/3})$, see [Val95]. We provide a weaker but easier upper bound next.

Lemma 53.3.4. *Let P be a set of n points picked randomly and uniformly in the unit square $[0,1]^2$. Let Y be the size of the largest subset of point of P that are in convex position. Then, $\mathbb{E}[Y] = O(n^{1/3}\log n/\log\log n)$, with high probability.*

Proof: Let V be a set of directions of size $O(n^c)$, where c is some constant, such that for any unit vector u, there is a vector in $\mathbf{v} \in V$, such that the angle between u and v is at most $1/n^c$. For a vector $\mathbf{v} \in V$, consider the grid $G(\mathbf{v})$ with directions $\mathbf{v}$, and orthogonal direction $\mathbf{v}^\perp$. Every cell of this grid is a rectangle with sidelength $1/n^{1/3}$ in the direction of $\mathbf{v}$, and $1/n^{2/3}$ in the orthogonal direction. In addition the origin is a vertex of $G(\mathbf{v})$. This grid is uniquely defined, and every cell in this grid has sidelength 1. The of number of grid cells of this grid intersecting the unit square is $O(n)$, as can be easily verified.

Let $\mathcal{F}$ be the set of all rectangles in all these grids that intersect the unit square. Clearly, the number of such cells is $O(|V|\, n) = O(n^{c+1})$. Each rectangle in $\mathcal{F}$ has area $1/n$, and as such by expectation it contains ≤ 1 point of P (the inequality is there because the rectangle might be partially outside the unit square). A standard application of Chernoff's inequality implies that the probability that a rectangle of $\mathcal{F}$ contains more than $10c\log n/\log\log n$ points of P is $\leq 1/n^{2c}$. As such, with high probability no rectangle in $\mathcal{F}$ contains more than $O(\log n/\log\log n)$ points of P.

Consider any convex body $C \subseteq [0,1]^2$. The key observation is that ∂C can be covered by $O(n^{1/3})$ rectangles of $\mathcal{F}$. Indeed, the perimeter of C is at most 4. As such, place $O(n^{1/3})$ points along ∂C that are at distance at most $1/(10n^{1/3})$ from each other. Similarly, place additional $O(n^{1/3})$ points on ∂C, such that the angle of the tangents between two consecutive points is at most $1/n^{1/3}$ (in radians) [a vertex of C might be picked repeatedly]. Let Q be the resulting set of points. Consider two consecutive points $\mathsf{p}, \mathsf{u} \in \mathsf{Q}$ along ∂C, and observe that the distance between them is at most $1/(10n^{1/3})$, and the angle between their two tangents is at most $\alpha = 1/n^{1/3}$. consider the triangle $\triangle$ formed by the two tangents to ∂C at p, u, and the segment p, u. This triangle has height bounded by $\|\mathsf{p} - \mathsf{u}\|\sin\alpha \leq 1/(10n^{2/3})$. It is now straightforward, if somewhat tedious to argue that one of the rectangles of $\mathcal{F}$ must contain $\triangle$.

Now we are almost done – if the maximum cardinality convex subset $\mathsf{Q} \subseteq \mathsf{P}$ was larger than $c' n^{1/3} \log n / \log\log n$, for some constant c', then let C be the convex-hull of this large subset. The above would imply that one of the rectangles of $\mathcal{F}$ must contain at least $\Omega(c' \log n / \log\log n)$ points of P, but this does not happen with high probability, for c' sufficiently large. Thus implying the claim. ■

In particular, the above implies that $\mathrm{med}(Y) = O(n^{1/3} \log n)$.

Theorem 53.3.5. *Let P be a set of n points picked randomly and uniformly in the unit square $[0,1]^2$. Let Y be the size of the largest subset of point of P that are in convex position. Then, for any $t > 0$, we have*

$$\mathbb{P}\Big[|Y - \mathrm{med}(Y)| \geq t c n^{1/6} \log^{1/2} n\Big] \leq 2\exp(-t^2/4),$$

for some constant c.

Proof: Observe that Y is 1-Lipschitz (i.e., changing the location of one point in P can decrease or increase the value of Y by at most 1. In addition Y is 1-certifiable, since we only need to list the points that form the convex subset. As such, Theorem 53.2.6 applies. Setting $b = \mathrm{med}(Y)$, we have by the above that $\mathrm{med}(Y) = \Omega(n^{1/3})$ and $\mathrm{med}(Y) = O(n^{1/3} \log n)$. As such, we have

$$\mathbb{P}\Big[Y \leq \mathrm{med}(Y) - t\sqrt{c n^{1/3} \log n}\Big] \mathbb{P}[X \geq \mathrm{med}(Y)] \leq \exp(-t^2/4).$$

Similarly, setting $b = \mathrm{med}(Y) + t\sqrt{c n^{1/3} \log n} \leq 2\mathrm{med}(Y)$, we have

$$\mathbb{P}[Y \leq \mathrm{med}(Y)] \mathbb{P}\Big[X \geq \mathrm{med}(Y) + t\sqrt{c n^{1/3} \log n}\Big] \leq \exp(-t^2/4).$$

Putting the two inequalities together, we get

$$\mathbb{P}\Big[|Y - \mathrm{med}(Y)| \geq t\sqrt{c n^{1/3} \log n}\Big] \leq 2\exp(-t^2/4).$$ ■

53.3.3. Balls into bins revisited

Given n balls, one throw them into b bins, where $\mathsf{b} \geq \mathsf{n}$. A ball that falls into a bin with i or more balls is ***i-heavy***. Let $h_{\geq i}$ be the number of i-heavy balls. It turns out that a strong concentration on $h_{\geq i}$ follows readily from Talagrand's inequality.

Lemma 53.3.6. *Consider throwing n balls into b bins, where $\mathsf{b} \geq 3\mathsf{n}$. Then, $e^{-2} F_i \leq \mathbb{E}[h_{\geq i}] \leq 6e^{i-1} F_i$, where $h_{\geq i}$ is the number of i-heavy balls, and $F_i = \mathsf{n}(\mathsf{n}/i\mathsf{b})^{i-1}$. Let β_i denote the expected number of pairs of i-heavy balls that are colliding. We have that $\beta_i = O(\mathsf{n} i (e\mathsf{n}/i\mathsf{b})^{i-1})$.*

Proof: Let $p = 1/\mathsf{b}$. A specific ball falls into a bin with exactly i balls, if there are $i-1$ balls, of the remaining $n-1$ balls that falls into the same bin. As such, the probability for that is $\gamma_i = p^{i-1}(1-p)^{\mathsf{n}-i}\binom{\mathsf{n}-1}{i-1}$. As such, a specific ball is i-heavy with probability

$$\alpha = \sum_{j=i}^{\mathsf{n}} \gamma_j = \sum_{j=i-1}^{\mathsf{n}-1} \binom{\mathsf{n}-1}{j} p^j (1-p)^{\mathsf{n}-j-1} \leq \sum_{j=i-1}^{\mathsf{n}-1} \left(\frac{e(\mathsf{n}-1)}{j\mathsf{b}}\right)^j \leq 2\left(\frac{e\mathsf{n}}{\mathsf{b}(i-1)}\right)^{i-1} \leq 6\left(\frac{e\mathsf{n}}{i\mathsf{b}}\right)^{i-1},$$

as $(n/i)^i \leq \binom{n}{i} \leq (en/i)^i$. Since $(1-p)^{\mathsf{n}-j-1} \geq (1-1/\mathsf{b})^{\mathsf{b}-1} \geq 1/e$, we have

$$\alpha \geq \frac{1}{e} \sum_{j=i-1}^{\mathsf{n}-1} \left(\frac{\mathsf{n}-1}{j\mathsf{b}}\right)^j \geq \frac{1}{e}\left(\frac{\mathsf{n}-1}{\mathsf{n}} \cdot \frac{\mathsf{n}}{(i-1)\mathsf{b}}\right)^{i-1} \geq \frac{1}{e^2}\left(\frac{\mathsf{n}}{i\mathsf{b}}\right)^{i-1}.$$

As such, we have $\mathbb{E}[h_{\geq i}] = \mathsf{n}\alpha = \Theta(\mathsf{n}(\mathsf{n}/\mathsf{b})^{i-1})$.

If a ball is in a bin with exactly j balls, for $j \geq i$, then it collides directly with $j-1$ other i-heavy balls. Thus, the expected number of collisions that a specific ball has with i-heavy balls is in expectation $\sum_{j=i}^{\mathsf{n}}(j-1)\gamma_j = \sum_{j=i-1}^{\mathsf{n}-1} j\gamma_{j+1}$. Summing over all balls, and dividing by two, as every i-heavy collision is counted twice, we have that the expected overall number of such collisions is

$$\beta_i = \frac{n}{2}\sum_{j=i-1}^{\mathsf{n}-1} j\gamma_{j+1} = \frac{\mathsf{n}}{2}\sum_{j=i-1}^{\mathsf{n}} j\binom{\mathsf{n}-1}{j}p^j(1-p)^{\mathsf{n}-j-1} = O\left(\mathsf{n}i\left(\frac{e\mathsf{n}}{i\mathsf{b}}\right)^{i-1}\right). \qquad \blacksquare$$

Lemma 53.3.7. *Consider throwing* n *balls into* b *bins, where* $\mathsf{b} \geq 3\mathsf{n}$. *Let* i *be a small constant integer,* $h_{\geq i}$ *be the number of* i*-heavy balls, and let* $\nu_i = \operatorname{med}(h_{\geq i})$. *Assume that* $\nu_i \geq 16i^2 c \log \mathsf{n}$, *where* c *is some arbitrary constant. Then, for some constant* c', *we have that* $|\nu_i - \mathbb{E}[h_{\geq i}]| \leq c' i\sqrt{\nu_i}$, *and*

$$\mathbb{P}\left[|h_{\geq i} - \nu_i| \geq 6i\sqrt{c\nu_i \ln \mathsf{n}}\right] \leq \frac{1}{\mathsf{n}^c} \qquad \text{and} \qquad \mathbb{P}\left[\left|h_{\geq i} - \mathbb{E}[h_{\geq i}]\right| \geq c' i\sqrt{\nu_i} + 6i\sqrt{c\nu_i \ln \mathsf{n}}\right] \leq \frac{1}{\mathsf{n}^c}.$$

Proof: Observe that $h_{\geq i}$ is 1-certifiable – indeed, the certificate is the list of indices of all the balls that are contained in bins with i or more balls. The variable $h_{\geq i}$ is also i-Lipschitz. Changing the location of a single ball, can make one bin that contains i balls, into a bin that contains only $i-1$ balls, thus decreasing $h_{\geq i}$ by i.

We require that $ti\sqrt{\nu_i} \leq \nu_i/2 \implies t \leq \sqrt{\nu_i}/(2i)$. Theorem 53.2.6 implies that

$$\mathbb{P}\left[h_{\geq i} \leq \nu_i - ti\sqrt{\nu_i}\right] \leq 2\exp(-t^2/4). \tag{53.5}$$

Setting $b = \nu_i + 2ti\sqrt{\nu_i}$, we have that

$$b - ti\sqrt{b} \geq b - ti\sqrt{\nu_i + 2ti\sqrt{\nu_i}} \geq b - ti\sqrt{2\nu_i} = \nu_i + 2ti\sqrt{\nu_i} - ti\sqrt{2\nu_i} \geq \nu_i.$$

This implies that $\mathbb{P}[h_{\geq i} \geq b]/2 \leq \mathbb{P}[h_{\geq i} \leq \nu_i]\,\mathbb{P}[X \geq b] \leq \mathbb{P}\left[h_{\geq i} \leq b - tk\sqrt{b}\right]\mathbb{P}[h_{\geq i} \geq b] \leq \exp(-t^2/4)$. We conclude that

$$\mathbb{P}\left[h_{\geq i} \geq \nu_i + 2ti\sqrt{\nu_i}\right] \leq 2\exp(-t^2/4). \tag{53.6}$$

Combining the above, we get that

$$\mathbb{P}\left[|h_{\geq i} - \nu_i| \geq 2ti\sqrt{\nu_i}\right] \leq 4\exp(-t^2/4)$$

We require that $4\exp(-t^2/4) \leq 1/\mathsf{n}^c$, which holds for $t = 3\sqrt{c\ln n}$. We get the inequality $\mathbb{P}[|h_{\geq i} - \nu_i| \geq 6i\sqrt{c\nu_i \ln n}] \leq 1/\mathsf{n}^c$, as claimed.

This in turn translates into the requirement that $3\sqrt{c\ln n} \leq \sqrt{\nu_i}/(2i) \implies 6i\sqrt{c\ln n} \leq \sqrt{\nu_i} \implies 36i^2 c\ln n \leq \nu_i$.

Next, we estimate the expectation. We have that

$$\mathbb{E}[h_{\geq i}] \geq \nu_i - \sum_{t=1}^{\infty} ti\sqrt{\nu_i}\,\mathbb{P}\left[h_{\geq i} \leq \nu_i - (t-1)i\sqrt{\nu_i}\right] \geq \nu_i - i\sqrt{\nu_i}\sum_{t=1}^{\infty} t2\exp(-(t-1)^2/4) \geq \nu_i - 10i\sqrt{\nu_i},$$

by Eq. (53.5). Similarly, by Eq. (53.6), we have

$$\mathbb{E}[h_{\geq i}] \leq \nu_i + \sum_{t=1}^{\infty} 2ti\sqrt{\nu_i}\,\mathbb{P}\left[h_{\geq i} \geq \nu_i + (t-1)i\sqrt{\nu_i}\right] \leq \nu_i + 4i\sqrt{\nu_i}\sum_{t=1}^{\infty} t\exp(-(t-1)^2/4) \leq \nu_i + 20i\sqrt{\nu_i},$$

As such, we have that $|\mathbb{E}[h_{\geq i}] - \nu_i| \leq 30i\sqrt{\nu_i}$, namely $c' \leq 30$. Combining the above inequalities implies the statement of the lemma. $\blacksquare$

Example 53.3.8. Consider throwing n into $\mathsf{b} = \mathsf{n}^{4/3}$ bins. Lemma 53.3.6 implies that $e^{-2}F_i \leq \mathbb{E}[h_{\geq i}] \leq 6e^{i-1}F_i$, where $F_i = \mathsf{n}/(i\mathsf{b}^{1/3})^{i-1}$. As such $\mathbb{E}[h_{\geq 2}] = \Theta(\mathsf{n}^{2/3})$, $\mathbb{E}[h_{\geq 3}] = \Theta(\mathsf{n}^{1/3})$, and $\mathbb{E}[h_{\geq 4}] = \Theta(1)$.

Applying Lemma 53.3.7, we get that the number of balls that collides (i.e., $h_{\geq 2}$), is strongly concentrated around some value $\nu_2 = \Theta(\mathsf{n}^{2/3})$, with the interval where it lies being of length $O(\mathsf{n}^{1/3}\sqrt{\log n})$.

53.4. Bibliographical notes

Our presentation follows closely Alon and Spencer [AS00]. Section 53.3.3 is from Har-Peled and Jones [HJ18].

53.5. Problems

Exercise 53.5.1. Elaborating on the argument of Lemma 53.3.3, prove that, with high probability, a random set of points picked uniformly in the unit square contains a convex subset of size $\Omega(n^{2/3})$.

References

[AS00] N. Alon and J. H. Spencer. *The Probabilistic Method*. 2nd. Wiley InterScience, 2000.

[HJ18] S. Har-Peled and M. Jones. On separating points by lines. *Proc. 29th ACM-SIAM Sympos. Discrete Algs.* (SODA), 918–932, 2018.

[Val95] P. Valtr. Probability that n random points are in convex position. *Discrete Comput. Geom.*, 13(3): 637–643, 1995.

Chapter 54

Low Dimensional Linear Programming

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

> "Napoleon has not been conquered by man. He was greater than all of us. But god punished him because he relied on his own intelligence alone, until that prodigious instrument was strained to breaking point. Everything breaks in the end."
>
> – Carl XIV Johan, King of Sweden.

54.1. Linear programming in constant dimension ($d > 2$)

Let assume that we have a set H of n linear inequalities defined over d (d is a small constant) variables. Every inequality in H defines a closed half space in $\mathbb{R}^d$. Given a vector $\overrightarrow{c} = (c_1, \ldots, c_d)$ we want to find $p = (p_1, \ldots, p_d) \in \mathbb{R}^d$ which is in all the half spaces $h \in H$ and $f(p) = \sum_i c_i p_i$ is maximized. Formally:

LP in d dimensions:$(H, \overrightarrow{c})$ H - set of n closed half spaces in $\mathbb{R}^d$ $\overrightarrow{c}$ - vector in d dimensions Find $p \in \mathbb{R}^d$ s.t. $\forall h \in H$ we have $p \in h$ and $f(p)$ is maximized. Where $f(p) = \left\langle p, \overrightarrow{c} \right\rangle$.

A closed half space in d dimensions is defined by an inequality of the form

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \leq b_n.$$

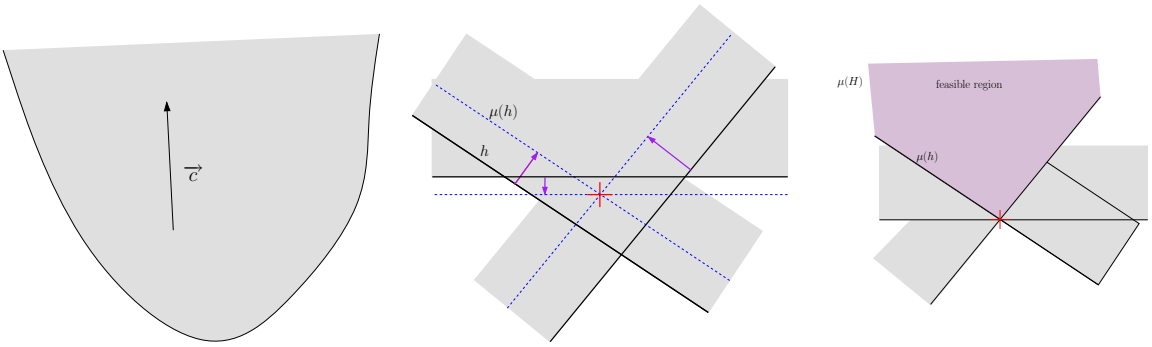
One difficulty that we ignored earlier, is that the optimal solution for the LP might be unbounded, see Figure 54.1.

Namely, we can find a solution with value ∞ to the target function.

For a half space h let $\eta(h)$ denote the normal of h directed into the feasible region. Let $\mu(h)$ denote the closed half space, resulting from h by translating it so that it passes through the origin. Let $\mu(H)$ be the resulting set of half spaces from H. See Figure 54.1 (b).

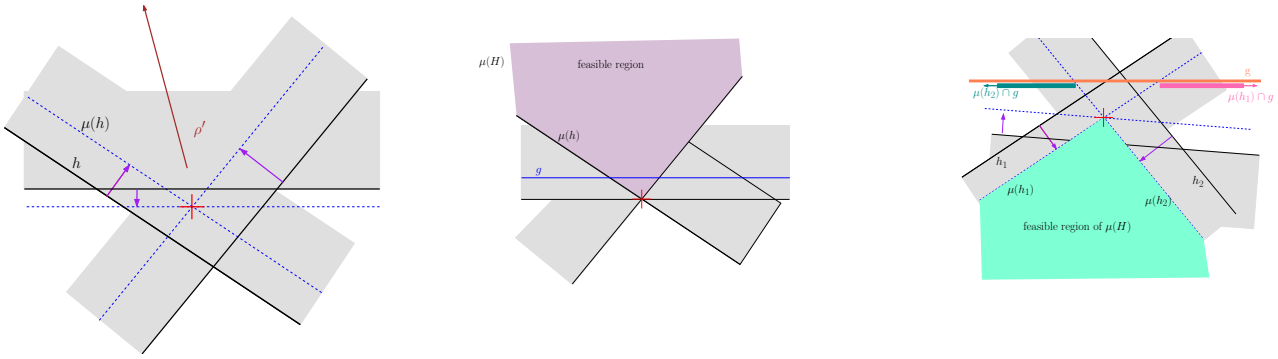
The new set of constraints $\mu(H)$ is depicted in Figure 54.1 (c).

Lemma 54.1.1. *$(H, \overrightarrow{c})$ is unbounded if and only if $(\mu(H), \overrightarrow{c})$ is unbounded.*



(a) (b) (c)

Figure 54.1: (a) Unbounded LP. (b). (c).



(a) (b) (c)

Figure 54.2: (a). (b). (c).

Proof: Consider the ρ' the unbounded ray in the feasible region of $(H, \overrightarrow{c})$ such that the line that contain it passes through the origin. Clearly, ρ' is unbounded also in $(H, \overrightarrow{c})$, and this is if and only if. See Figure 54.2 (a). ■

Lemma 54.1.2. *Deciding if $(\mu(H), \overrightarrow{c})$ is bounded can be done by solving a $d-1$ dimensional LP. Furthermore, if it is bounded, then we have a set of d constraints, such that their intersection prove this.*

Furthermore, the corresponding set of d constraints in H testify that $(H, \overrightarrow{c})$ is bounded.

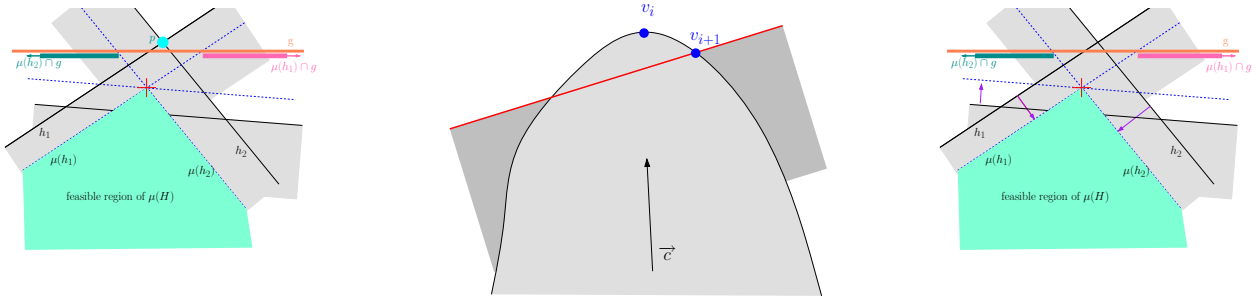
Proof: Rotate space, such that $\overrightarrow{c}$ is the vector $(0, 0, \ldots, 0, 1)$. And consider the hyperplane $g \equiv x_d = 1$. Clearly, $(\mu(H), \overrightarrow{c})$ is unbounded if and only if the region $g \cap \bigcap_{h \in \mu(H)} h$ is non-empty. By deciding if this region is unbounded, is equivalent to solving the following LP: $L' = (H', (1, 0, \ldots, 0))$ where

$$H' = \left\{ g \cap h \;\middle|\; h \in \mu(H) \right\}.$$

Let $h \equiv a_1 x_1 + \ldots + a_d x_d \leq 0$, the region corresponding to $g \cap h$ is $a_1 x_1 + \cdots + a_{d-1} x_{d-1} \leq -a_d$ which is a $d-1$ dimensional hyperplane. See Figure 54.2 (b).

But this is a $d-1$ dimensional LP, because everything happens on the hyperplane $x_d = 1$.

Notice that if $(\mu(H), \overrightarrow{c})$ is bounded (which happens if and only if $(H, \overrightarrow{c})$ is bounded), then L' is infeasible, and the LP L' would return us a set d constraints that their intersection is empty. Interpreting those constraints in the original LP, results in a set of constraints that their intersection is bounded in the direction of $\overrightarrow{c}$. See Figure 54.2 (c).



(a) (b) (c)

Figure 54.3: (a). (b). (c).

(In the above example, $\mu(H) \cap g$ is infeasible because the intersection of $\mu(h_2) \cap g$ and $\mu(h_1) \cap g$ is empty, which implies that $h_1 \cap h_2$ is bounded in the direction $\overrightarrow{c}$ which we care about. The positive y direction in this figure.) ■

We are now ready to show the algorithm for the LP for $L = (H, \overrightarrow{c})$. By solving a $d-1$ dimensional LP we decide whether L is unbounded. If it is unbounded, we are done (we also found the unbounded solution, if you go carefully through the details).

See Figure 54.3 (a).

(in the above figure, we computed p.)

In fact, we just computed a set $h_1, \ldots, h_d$ s.t. their intersection is bounded in the direction of $\overrightarrow{c}$ (thats what the boundness check returned).

Let us randomly permute the remaining half spaces of H, and let $h_1, h_2, \ldots, h_d, h_{d+1}, \ldots, h_n$ be the resulting permutation.

Let v_i be the vertex realizing the optimal solution for the LP:

$$L_i = \left(\{h_1, \ldots, h_i\}, \overrightarrow{c}\right)$$

There are two possibilities:

1. $v_i = v_{i+1}$. This means that $v_i \in h_{i+1}$ and it can be checked in constant time.
2. $v_i \neq v_{i+1}$. It must be that $v_i \notin h_{i+1}$ but then, we must have... What is depicted in Figure 54.3 (b).

B - the set of d constraints that define v_{i+1}. If $h_{i+1} \notin B$ then $v_i = v_{i+1}$. As such, the probability of $v_i \neq v_{i+1}$ is roughly d/i because this is the probability that one of the elements of B is h_{i+1}. Indeed, fix the first $i+1$ elements, and observe that there are d elements that are marked (those are the elements of B). Thus, we are asking what is the probability of one of d marked elements to be the last one in a random permutation of $h_{d+1}, \ldots, h_{i+1}$, which is exactly $d/(i+1-d)$.

Note that if some of the elements of B is $h_1, \ldots, h_d$ than the above expression just decreases (as there are less marked elements).

Well, let us restrict our attention to ∂h_{i+1}. Clearly, the optimal solution to L_{i+1} on h_{i+1} is the required v_{i+1}. Namely, we solve the LP $L_{i+1} \cap h_{i+1}$ using recursion.

This takes $T(i+1, d-1)$ time. What is the probability that $v_{i+1} \neq v_i$?

Well, one of the d constraints defining v_{i+1} has to be h_{i+1}.The probability for that is ≤ 1 for $i \leq 2d-1$, and it is

$$\leq \frac{d}{i+1-d},$$

otherwise.

Summarizing everything, we have:

$$\begin{aligned} T(n,d) &= O(n) + T(n,d-1) + \sum_{i=d+1}^{2d} T(i,d-1) \\ &+ \sum_{i=2d+1}^{n} \frac{d}{i+1-d} T(i,d-1) \end{aligned}$$

What is the solution of this monster? Well, one essentially to guess the solution and verify it. To guess solution, let us "simplify" (incorrectly) the recursion to :

$$T(n,d) = O(n) + T(n,d-1) + d \sum_{i=2d+1}^{n} \frac{T(i,d-1)}{i+1-d}$$

So think about the recursion tree. Now, every element in the sum is going to contribute a near constant factor, because we divide it by (roughly) $i+1-d$ and also, we are guessing the the optimal solution is linear/near linear.

In every level of the recursion we are going to penalized by a multiplicative factor of d. Thus, it is natural, to conjecture that $T(n,d) \leq (3d)^{3d} n$.

Which can be verified by tedious substitution into the recurrence, and is left as exercise.

Theorem 54.1.3. *Given an d dimensional LP $(H, \overrightarrow{c})$,it can be solved in expected $O\Big((3d)^{3d} n\Big)$ time (the constant in the O is dim independent).*

BTW, we are being a bit conservative about the constant. In fact, one can prove that the running time is $d!n$. Which is still exponential in d.

```
SolveLP((H, c⃗))
    /* initialization */
    Rotate (H, c⃗) s.t. c⃗ = (0, ..., 1)
    Solve recursively the d − 1 dim LP:
            L′ ≡ μ(H) ∩ (x_d = 1)
    if L′ has a solution then
            return "Unbounded"

    Let g_1, ..., g_d be the set of constraints of L′ that testifies that L′ is infeasible
    Let h_1, ..., h_d be the hyperplanes of H corresponding to g_1, ..., g_d
    Permute H s.t. h_1, ..., h_d are first.
    v_d = ∂h_1 ∩ ∂h_2 ∩ ··· ∩ ∂h_d
    /*v_d is a vertex that testifies that (H, c⃗) is bounded */

    /* the algorithm itself */
    for i ← d + 1 to n do
        if v_{i−1} ∈ h_i then
                v_i ← v_{i−1}
        else
                v_i ← SolveLP((H_{i−1} ∩ ∂h_i, c⃗))      (*)
                  where H_{i−1} = {h_1, ..., h_{i−1}}

    return v_n
```

54.2. Handling Infeasible Linear Programs

In the above discussion, we glossed over the question of how to handle LPs which are infeasible. This requires slightly modifying our algorithm to handle this case, and I am only describing the required modifications.

First, the simplest case, where we are given an LP L which is one dimensional (i.e., defined over one variable). Clearly, we can solve this LP in linear time (verify!), and furthermore, if there is no solution, we can return two input inequality $ax \leq b$ and $cx \geq d$ for which there is no solution together (i.e., those two inequalities [i.e., constraints] testifies that the LP is not satisfiable).

Next, assume that the algorithm `SolveLP` when called on a $d-1$ dimensional LP L', if L' is not feasible it return the d constraints of L' that together have non-empty intersection. Namely, those constraints are the witnesses that L' is infeasible.

So the only place, where we can get such answer, is when computing v_i (in the (*) line in the algorithm). Let $h'_1, \ldots, h'_d$ be the corresponding set of d constraints of H_{i-1} that testifies that $(H_{i-1} \cap \partial h_i, \overrightarrow{c})$ is an infeasible LP. Clearly, $h'_1, \ldots, h'_d, h_i$ must be a set of $d+1$ constraints that are together are infeasible, and that is what `SolveLP` returns.

54.3. References

The description in this class notes is loosely based on the description of low dimensional LP in the book of de Berg *et al.* [BCKO08].

References

[BCKO08] M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars. *Computational Geometry: Algorithms and Applications*. 3rd. Santa Clara, CA, USA: Springer, 2008.

Chapter 55

Algorithmic Version of Lovász Local Lemma

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

As for me, they took away my toy merchant, wishing with him to banish all toys from the world.

The tin drum, Gunter Grass

55.1. Introduction

Let $\mathcal{P}$ be a collection of independent random variables in some probability space Ω. We are interested in (bad) events that are determined by these variables. Specifically, for an event B, there is a set of variables $S \subseteq \mathcal{P}$ that determine if B happens. There minimal such set of variables is denoted by $\mathrm{vbl}(B)$ (note, that it is unique). We assume that the set $\mathrm{vbl}(B)$ is either easily computable or available, for any event of interest.

Consider a (finite) family $\mathcal{B}$ of such (bad) events. The ***dependency graph*** $\mathsf{G} = \mathsf{G}(\mathcal{B})$, with the vertices being the events of $\mathcal{B}$, and two events $B_1, B_2 \in \mathcal{B}$ are connected by an edge $\iff \mathrm{vbl}(B_1) \cap \mathrm{vbl}(B_2) \neq \emptyset$. Let $\Gamma(B)$ be all the neighbors B in G, and let $\Gamma_+(B) = \{B\} \cup \Gamma(B)$. Observe that B is independent of the events in $\mathcal{B} \setminus (\{B\} \cup \Gamma(B))$.

A specific assignment to the variables of S ***violates*** B, if it causes it to happen (remember, it is a bad event).

Task. Our purpose is to compute an assignment to the variables of $\mathcal{P}$, such that none of the bad events of $\mathcal{B}$ happens.

Algorithm. Initially, the algorithm assigns the variables of $\mathcal{P}$ random values. As long as there is a violated event $B \in \mathcal{B}$, resample all the variables of $\mathrm{vbl}(B)$ (independently according to their own distributions) – this is a ***resampling*** of B. The algorithm repeats this till no event is violated.

Finer details. We fix some arbitrary strategy (randomized or deterministic) of how to pick the next event. This now fully specify the algorithm.

Remark. Of course, it is not clear that the algorithm would always succeeds. Let us just assume, for the time being, that we are in a case where the algorithm always finds a good assignment (which always exists).

55.1.1. Analysis

Let $L(i) \in \mathcal{B}$ be the event that was resampled in the ith iteration of the algorithm, for $i > 0$. The sequence formed by L is the ***log*** of the execution.

A ***witness tree*** $T = (T, \sigma_T)$ is a rooted tree T together with a labeling σ_T. Here, every node $v \in T$ is labeled by an event $\sigma_T(v) \in \mathcal{B}$. If a node u is a child of v in T, then $\sigma_T(u) \in \Gamma_+(v)$. Two nodes in a tree are ***siblings*** if they have a common parent. If all siblings have distinct labels then the witness tree is ***proper***.

For a vertex v of T, let $[v] = \sigma_T(v)$.

Theorem 55.1.1. *Let $\mathcal{P}$ be a finite set of independent random variables in a probability space, and let $\mathcal{B}$ be a set of (bad) events determined by these variables. If there is an assignment $x : \mathcal{B} \to (0,1)$, such that*

$$\forall B \in \mathcal{B} \qquad \mathbb{P}[B] \leq x(B) \prod_{C \in \Gamma(B)} (1 - x(C)),$$

then there exists an assignment for the variables of $\mathcal{P}$ such that no event in $\mathcal{B}$ happens. Furthermore, in expectation, the algorithm described above resamples, any event $B \in \mathcal{B}$, at most $x(B)/(1-x(B))$ times. Overall, the expected number of resampling steps is at most $\sum_{B \in \mathcal{B}} x(B)/(1 - x(B))$.

Chapter 56

Some math stuff

598 - Class notes for Randomized Algorithms
Sariel Har-Peled
April 2, 2024

56.1. Some useful estimates

Lemma 56.1.1. *For any $n \geq 2$, and $m \geq 1$, we have that $(1 - 1/n)^m \geq 1 - m/n$.*

Proof: Follows by induction. Indeed, for $m = 1$ the claim is immediate. For $m \geq 2$, we have

$$\left(1 - \frac{1}{n}\right)^m = \left(1 - \frac{1}{n}\right)\left(1 - \frac{1}{n}\right)^{m-1} \geq \left(1 - \frac{1}{n}\right)\left(1 - \frac{m-1}{n}\right) \geq 1 - \frac{m}{n}.$$ ■

This implies the following.

Lemma 56.1.2. *For any $m \leq n$, we have that $1 - m/n \leq (1 - 1/n)^m \leq \exp(-m/n)$.*

Bibliography

[ABKU99] Y. Azar, A. Broder, A. Karlin, and E. Upfal. Balanced allocations. *SIAM Journal on Computing*, 29(1): 180–200, 1999.

[ABN08a] I. Abraham, Y. Bartal, and O. Neiman. Nearly tight low stretch spanning trees. *Proc. 49th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), 781–790, 2008.

[ABN08b] I. Abraham, Y. Bartal, and O. Neiman. Nearly tight low stretch spanning trees. *CoRR*, abs/0808.2017, 2008. arXiv: 0808.2017.

[AES99] P. K. Agarwal, A. Efrat, and M. Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM J. Comput.*, 29: 912–953, 1999.

[Aga04] P. K. Agarwal. Range searching. *Handbook of Discrete and Computational Geometry*. Ed. by J. E. Goodman and J. O'Rourke. 2nd. Boca Raton, FL, USA: CRC Press LLC, 2004. Chap. 36, pp. 809–838.

[AI06] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Proc. 47th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), 459–468, 2006.

[AI08] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1): 117–122, 2008.

[AKPW95] N. Alon, R. M. Karp, D. Peleg, and D. West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1): 78–100, 1995.

[AMS98] P. K. Agarwal, J. Matoušek, and O. Schwarzkopf. Computing many faces in arrangements of lines and segments. *SIAM J. Comput.*, 27(2): 491–505, 1998.

[AMS99] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1): 137–147, 1999.

[AN04] N. Alon and A. Naor. Approximating the cut-norm via grothendieck's inequality. *Proc. 36th Annu. ACM Sympos. Theory Comput.* (STOC), 72–80, 2004.

[AR94] N. Alon and Y. Roichman. Random cayley graphs and expanders. *Random Struct. Algorithms*, 5(2): 271–285, 1994.

[Aro98] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. *J. Assoc. Comput. Mach.*, 45(5): 753–782, 1998.

[AS00] N. Alon and J. H. Spencer. *The Probabilistic Method*. 2nd. Wiley InterScience, 2000.

[ASS08] N. Alon, O. Schwartz, and A. Shapira. An elementary construction of constant-degree expanders. *Combin. Probab. Comput.*, 17(3): 319–327, 2008.

[Bar96] Y. Bartal. Probabilistic approximations of metric space and its algorithmic application. *Proc. 37th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), 183–193, 1996.

[Bar98] Y. Bartal. On approximating arbitrary metrics by tree metrics. *Proc. 30th Annu. ACM Sympos. Theory Comput.* (STOC), 161–168, 1998.

[BCKO08] M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars. *Computational Geometry: Algorithms and Applications*. 3rd. Santa Clara, CA, USA: Springer, 2008.

[BDS95] M. de Berg, K. Dobrindt, and O. Schwarzkopf. On lazy randomized incremental construction. *Discrete Comput. Geom.*, 14: 261–286, 1995.

[BK90] A. Z. Broder and A. R. Karlin. Multilevel adaptive hashing. *Proc. 1th ACM-SIAM Sympos. Discrete Algs.* (SODA), 43–53, 1990.

[Bol98] B. Bollobas. *Modern Graph Theory*. Springer-Verlag, 1998.

[BS95] M. de Berg and O. Schwarzkopf. Cuttings and applications. *Int. J. Comput. Geom. Appl.*, 5: 343–355, 1995.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge, 2004.

[BY98] J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, 1998.

[CCH09] C. Chekuri, K. L. Clarkson., and S. Har-Peled. On the set multi-cover problem in geometric settings. *Proc. 25th Annu. Sympos. Comput. Geom.* (SoCG), 341–350, 2009.

[CF90] B. Chazelle and J. Friedman. A deterministic view of random sampling and its use in geometry. *Combinatorica*, 10(3): 229–249, 1990.

[Che86] L. P. Chew. *Building Voronoi diagrams for convex polygons in linear expected time*. Technical Report PCS-TR90-147. Hanover, NH: Dept. Math. Comput. Sci., Dartmouth College, 1986.

[CKR04] G. Călinescu, H. J. Karloff, and Y. Rabani. Approximation algorithms for the 0-extension problem. *SIAM J. Comput.*, 34(2): 358–372, 2004.

[Cla87] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2: 195–222, 1987.

[Cla88] K. L. Clarkson. Applications of random sampling in computational geometry, II. *Proc. 4th Annu. Sympos. Comput. Geom.* (SoCG), 1–11, 1988.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press / McGraw-Hill, 2001.

[CMS93] K. L. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. *Comput. Geom. Theory Appl.*, 3(4): 185–212, 1993.

[CS00] S. Cho and S. Sahni. A new weight balanced binary search tree. *Int. J. Found. Comput. Sci.*, 11(3): 485–513, 2000.

[CS89] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4(5): 387–421, 1989.

[DIIM04] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. *Proc. 20th Annu. Sympos. Comput. Geom.* (SoCG), 253–262, 2004.

[DP09] D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

[DS00] P. G. Doyle and J. L. Snell. Random walks and electric networks. *ArXiv Mathematics e-prints*, 2000. eprint: `math/0001057`.

[EEST08] M. Elkin, Y. Emek, D. A. Spielman, and S. Teng. Lower-stretch spanning trees. *SIAM J. Comput.*, 38(2): 608–628, 2008.

[EHS14] D. Eppstein, S. Har-Peled, and A. Sidiropoulos. *On the Greedy Permutation and Counting Distances*. manuscript. 2014.

[FRT04] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Sys. Sci.*, 69(3): 485–497, 2004.

[GG81] O. Gabber and Z. Galil. Explicit constructions of linear-sized superconcentrators. *J. Comput. Syst. Sci.*, 22(3): 407–420, 1981.

[GLS93] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. 2nd. Vol. 2. Algorithms and Combinatorics. Berlin Heidelberg: Springer-Verlag, 1993.

[Gre69] W. Greg. *Why are Women Redundant?* Trübner, 1869.

[GRSS95] M. Golin, R. Raman, C. Schwarz, and M. Smid. Simple randomized algorithms for closest pair problems. *Nordic J. Comput.*, 2: 3–27, 1995.

[Gup00] A. Gupta. *Embeddings of Finite Metrics*. PhD thesis. University of California, Berkeley, 2000.

[GW95] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42(6): 1115–1145, 1995.

[Har00a] S. Har-Peled. Constructing planar cuttings in theory and practice. *SIAM J. Comput.*, 29(6): 2016–2039, 2000.

[Har00b] S. Har-Peled. Taking a walk in a planar arrangement. *SIAM J. Comput.*, 30(4): 1341–1367, 2000.

[Har11] S. Har-Peled. *Geometric Approximation Algorithms*. Vol. 173. Math. Surveys & Monographs. Boston, MA, USA: Amer. Math. Soc., 2011.

[Hås01a] J. Håstad. Some optimal inapproximability results. *J. Assoc. Comput. Mach.*, 48(4): 798–859, 2001.

[Hås01b] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4): 798–859, 2001.

[HJ18] S. Har-Peled and M. Jones. On separating points by lines. *Proc. 29th ACM-SIAM Sympos. Discrete Algs.* (SODA), 918–932, 2018.

[HLW06] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin Amer. Math. Soc.*, 43: 439–561, 2006.

[HR15] S. Har-Peled and B. Raichel. Net and prune: A linear time algorithm for Euclidean distance problems. *J. Assoc. Comput. Mach.*, 62(6): 44:1–44:35, 2015.

[HW87] D. Haussler and E. Welzl. ε-nets and simplex range queries. *Discrete Comput. Geom.*, 2: 127–151, 1987.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. *Proc. 30th Annu. ACM Sympos. Theory Comput.* (STOC), 604–613, 1998.

[Ind01] P. Indyk. Algorithmic applications of low-distortion geometric embeddings. *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), Tutorial. 10–31, 2001.

[JL84] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mapping into hilbert space. *Contemporary Mathematics*, 26: 189–206, 1984.

[Kel56] J. L. Kelly. A new interpretation of information rate. *Bell Sys. Tech. J.*, 35(4): 917–926, 1956.

[KKMO04] S. Khot, G. Kindler, E. Mossel, and R. O'Donnell. Optimal inapproximability results for max cut and other 2-variable csps. *Proc. 45th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), To appear in SICOMP. 146–154, 2004.

[KKT91] C. Kaklamanis, D. Krizanc, and T. Tsantilas. Tight bounds for oblivious routing in the hypercube. *Math. sys. theory*, 24(1): 223–232, 1991.

[KLMN05] R. Krauthgamer, J. R. Lee, M. Mendel, and A. Naor. Measured descent: a new embedding method for finite metric spaces. *Geom. funct. anal. (GAFA)*, 15(4): 839–858, 2005.

[KOR00] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 2(30): 457–474, 2000.

[LM00] B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. *Ann. Statist.*, 28(5): 1302–1338, 2000.

[Mat02] J. Matoušek. *Lectures on Discrete Geometry*. Vol. 212. Grad. Text in Math. Springer, 2002.

[Mat92] J. Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2(3): 169–186, 1992.

[Mat98] J. Matoušek. On constants for cuttings in the plane. *Discrete Comput. Geom.*, 20: 427–448, 1998.

[Mat99] J. Matoušek. *Geometric Discrepancy*. Vol. 18. Algorithms and Combinatorics. Springer, 1999.

[McD89] C. McDiarmid. *Surveys in Combinatorics*. Ed. by J. Siemons. Cambridge University Press, 1989. Chap. On the method of bounded differences.

[Mil76] G. L. Miller. Riemann's hypothesis and tests for primality. *J. Comput. Sys. Sci.*, 13(3): 300–317, 1976.

[MN08] M. Mendel and A. Naor. *Towards a calculus for non-linear spectral gaps*. manuscript. 2008.

[MN98] J. Matoušek and J. Nešetřil. *Invitation to Discrete Mathematics*. Oxford Univ Press, 1998.

[MNP06] R. Motwani, A. Naor, and R. Panigrahi. Lower bounds on locality sensitive hashing. *Proc. 22nd Annu. Sympos. Comput. Geom.* (SoCG), 154–157, 2006.

[MOO05] E. Mossel, R. O'Donnell, and K. Oleszkiewicz. Noise stability of functions with low influences invariance and optimality. *Proc. 46th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), 21–30, 2005.

[MP80] J. I. Munro and M. Paterson. Selection and sorting with limited storage. *Theo. Comp. Sci.*, 12: 315–323, 1980.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.

[MS09] M. Mendel and C. Schwob. Fast c-k-r partitions of sparse graphs. *Chicago J. Theor. Comput. Sci.*, 2009, 2009.

[MU05] M. Mitzenmacher and U. Upfal. *Probability and Computing – randomized algorithms and probabilistic analysis*. Cambridge, 2005.

[Mul89] K. Mulmuley. An efficient algorithm for hidden surface removal. *Comput. Graph.*, 23(3): 379–388, 1989.

[Mul94] K. Mulmuley. *Computational Geometry: An Introduction Through Randomized Algorithms*. Englewood Cliffs, NJ: Prentice Hall, 1994.

[Nor98] J. R. Norris. *Markov Chains*. Statistical and Probabilistic Mathematics. Cambridge Press, 1998.

[Rab76] M. O. Rabin. Probabilistic algorithms. *Algorithms and Complexity: New Directions and Recent Results*. Ed. by J. F. Traub. Orlando, FL, USA: Academic Press, 1976, pp. 21–39.

[Rab80] M. O. Rabin. Probabilistic algorithm for testing primality. *J. Number Theory*, 12(1): 128–138, 1980.

[RVW02] O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. *Annals Math.*, 155(1): 157–187, 2002.

[SA95] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. New York: Cambridge University Press, 1995.

[SA96] R. Seidel and C. R. Aragon. Randomized search trees. *Algorithmica*, 16: 464–497, 1996.

[Sch79] A. Schönhage. On the power of random access machines. *Proc. 6th Int. Colloq. Automata Lang. Prog.* (ICALP), vol. 71. 520–529, 1979.

[Sei93] R. Seidel. Backwards analysis of randomized geometric algorithms. *New Trends in Discrete and Computational Geometry*. Ed. by J. Pach. Vol. 10. Algorithms and Combinatorics. Springer-Verlag, 1993, pp. 37–68.

[Sha03] M. Sharir. The Clarkson-Shor technique revisited and extended. *Comb., Prob. & Comput.*, 12(2): 191–201, 2003.

[Smi00] M. Smid. Closest-point problems in computational geometry. *Handbook of Computational Geometry*. Ed. by J.-R. Sack and J. Urrutia. Amsterdam, The Netherlands: Elsevier, 2000, pp. 877–935.

[Sni85] M. Snir. Lower bounds on probabilistic linear decision trees. *Theor. Comput. Sci.*, 38: 69–82, 1985.

[Ste12] E. Steinlight. Why novels are redundant: sensation fiction and the overpopulation of literature. *ELH*, 79(2): 501–535, 2012.

[Val95] P. Valtr. Probability that n random points are in convex position. *Discrete Comput. Geom.*, 13(3): 637–643, 1995.

[VC71] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.*, 16: 264–280, 1971.

[Vöc03] B. Vöcking. How asymmetry helps load balancing. *J. ACM*, 50(4): 568–589, 2003.

[Wes01] D. B. West. *Intorudction to Graph Theory*. 2ed. Prentice Hall, 2001.

[WG75] H. W. Watson and F. Galton. On the probability of the extinction of families. *J. Anthrop. Inst. Great Britain*, 4: 138–144, 1875.

Index

(k, n) decoding function, 275
(k, n) encoding function, 275
(n, d)-graph, 287
(n, d, c)-expander, 152
2-universal, 65
C-Lipschitz, 252
K-bi-Lipschitz, 252
T-distance, 352
δ-expander, 287
$\mathcal{F}_i$-measurable, 136
ρ-sample, 245
σ-algebra, 23
σ-field, 135
c-Lipschitz, 137
regular, d, 212
f-certifiable, 355
i-heavy, 359
k-HST, 253
k-median clustering, 253
k-test, 184
k-wise independent, 57, 61
kth frequency moment, 171
t-heavy, 246
t-step transition probability, 201
ε-nets and simplex range queries, 241, 248
pdf, 177
Davenport-Schinzel Sequences and Their Geometric Applications, 332

A deterministic view of random sampling and its use in geometry, 331
A graph-theoretic game and its application to the k-server problem, 260
A New Interpretation of Information Rate, 93
A New Weight Balanced Binary Search Tree, 87
A tight bound on approximating arbitrary metrics by tree metrics, 260
abelian, 338
Abraham, Ittai, 260
active, 228
Adaptive estimation of a quadratic functional by model selection, 159
adjacency matrix, 287
Agarwal, P. K., 192, 330–332
algorithm
Alg, 27, 47, 48, 60, 136, 152, 154, 219–224, 304
Alg, 27
Contract, 118, 119
decrease-key, 226
delete-min, 226
EuclidGCD, 336
EuclidGCD, 336
FastCut, 118, 119
Jacobi, 345
Las Vegas, 47, 223
Lookup, 63, 64
lookup, 63
MinCut, 115–117, 119
MinCutRep, 117–119
Monte Carlo, 47, 223
QuickSort, 33–35, 47, 83, 84, 87, 105, 106, 223, 229
QuickSelect, 34–36
randomized, 27
RandomRoute, 107
Algorithmic applications of low-distortion geometric embeddings, 260
Algorithmic Geometry, 330, 332
Alon, N., 260, 285, 301, 303, 361
Alon, Noga, 176
amortize, 65
An efficient algorithm for hidden surface removal, 331
An elementary construction of constant-degree expanders, 301
Andoni, A., 191

Andoni, Alexandr, 191
Applications of random sampling in computational geometry, II, 330, 331
approximate near neighbor, 181
approximate nearest neighbor, 181, 189–191
Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality, 161, 191, 192
Approximating the cut-norm via Grothendieck's inequality, 285
approximation
  maximization problem, 27
Approximation Algorithms for the 0*-Extension Problem*, 253, 260
approximation factor, 304
APX
  Hard, 303
Aragon, C. R., 87
Arora, S., 260
arrangement, 315, 317
atomic event, 23, 135
average-case analysis, 18
Azar, Y., 148

backwards analysis, 225
Backwards Analysis of Randomized Geometric Algorithms, 229, 331
bad, 67
balanced, 197
Balanced Allocations, 148
ball, 251
Bartal, Y., 260
Bartal, Yair, 260
Berg, M. de, 330, 331, 367
Bernoulli distribution, 25
bi-tension, 291
binary code, 269
binary symmetric channel, 275
binomial
  estimates, 243
binomial distribution, 26
bit fixing, 107
black-box access, 227
Boissonnat, J.-D., 330, 332
Bollobas, B., 216
bounded differences, 133
Boyd, S., 284
Broder, Andrei Z., 148
Building Voronoi diagrams for convex polygons in linear expected time, 331
butterfly, 324

Călinescu, G., 253, 260
Catalan number, 197, 201
central limit theorem, 89
certified vertex, 323
chaining, 64
characteristic vector, 292
Chazelle, B., 331
Chekuri, C., 332
Chernoff inequality, 95
  simplified form, 95
Chervonenkis, A. Y., 236, 241
Chew, L. P., 331
chi-square distribution with k degrees of freedom, 158
Cho, S., 87
Cholesky decomposition, 284
Clarkson, K. L., 330, 331
Clarkson, Kenneth L., 330
Clarkson., K. L., 332
clause
  dangerous, 309
  survived, 310
Closest-Point Problems in computational geometry, 76
clusters, 253
CNF, 27
collide, 64
coloring, 30
combinatorial dimension, 323
commutative group, 338
commute time, 206
Complexity
  co−, 47, 224
  BPP, 48, 224
  NP, 47, 223
  PP, 48, 224
  P, 47, 223
  RP, 48, 224
  ZPP, 48, 224
Computational Geometry: Algorithms and Applications, 330, 367
Computational Geometry: An Introduction Through Randomized Algorithms, 331, 332
Compute, 66

Computing Many Faces in Arrangements of Lines and Segments, 330, 331
Concentration of Measure for the Analysis of Randomized Algorithms, 103
conditional expectation, 81, 131
conditional probability, 24, 113
confidence, 44
conflict graph, 319
conflict list, 319
congruent modulo n, 336
congruent modulo p, 57, 66
consistent labeling, 297
Constructing planar cuttings in theory and practice, 330, 331
contraction
  edge, 114
convex hull, 243
Convex Optimization, 284
convex position, 358
convex programming, 281
convex-hull, 352
coprime, 335
Cormen, T. H., 73
cover time, 206
critical, 75
crossing number, 314
cuckoo hashing, 64
cumulative distribution function, 177
cut, 113
  minimum, 113
cuts, 113
cutting, 327
Cuttings and applications, 331
cyclic, 339
CNF, 304

Datar, M., 190, 191
defining set, 323
degree, 45
Delaunay
  circle, 324
  triangle, 324
dependency graph, 307, 369
dimension
  combinatorial, 323
discrepancy, 121
discrepancy of χ, 121
distance
  Hamming, 181
distortion, 252
distribution
  normal, 155, 189, 190
    multi-dimensional, 157
  stable
    2, 189
    p, 189
distributivity of multiplication over addition, 340
divides, 57, 66
Dobrindt, K., 331
dominating, 197
Doob martingale, 137
doubly stochastic, 206
Doyle, P. G., 210
Dubhashi, Devdatt P., 103
Dyck word, 197
Dyck words, 201
Dynamic, 63

edge, 315, 317
effective resistance, 207
Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces, 191
Efrat, A., 332
eigenvalue, 215, 287
eigenvalues, 215
eigenvector, 287
electrical network, 207
elementary event, 23, 135
Elkin, Michael, 260
embedding, 252, 314
Embeddings of Finite Metrics, 260
entropy, 263, 272
  binary, 263
Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors, 301
epochs, 211
Eppstein, D., 229
escalated choices, 147
estimate, 236
Euler totient function, 337
event, 23
expander
  $[n, d, \delta]$-expander, 287
  $[n, d, c]$-expander, 218

c, 218
Expander Graphs and Their Applications, 301
expectation, 24
Explicit Constructions of Linear-Sized Superconcentrators, 217
Extensions of Lipschitz mapping into Hilbert space, 161
extraction function, 266

face, 317
faces, 315
Fakcharoenphol, J., 260
family, 64
Fast C-K-R Partitions of Sparse Graphs, 229
field, 293, 340
filter, 135
filtration, 135
final strong component, 201
finite metric, 227
first order statistic, 177
Fold, 66
Four results on randomized incremental constructions, 331
Friedman, J., 331
fully explicit, 220
function
    sensitive, 184

Gabber, Ofer, 217
Galil, Zvi, 217
Galton, F., 111
Gaussian, 157
generator, 339
Geometric Algorithms and Combinatorial Optimization, 284
Geometric Approximation Algorithms, 247
Geometric Discrepancy, 103, 123
geometric distribution, 26
Goemans, M. X., 284
Golin, M., 76
Grötschel, M., 284
graph
    d-regular, 287
    labeled, 212
    lollipop, 206
Greg, W.R., 111
grid, 73
grid cell, 73
grid cluster, 73
ground set, 235
group, 336, 338
growth function, 238, 247
Gupta, A., 260
gcd, 335

Håstad, J., 284
Hamming distance, 181
harmonic number, 34
Har-Peled, S., 229, 247, 330–332
Har-Peled, Sariel, 76, 361
Haussler, D., 241, 248
heavy
    t-heavy, 325
height, 144
Hierarchically well-separated tree, 253
history, 200
hitting time, 206
Hoeffding's inequality, 103
Hoory, S., 301
How Asymmetry Helps Load Balancing, 148
HST, 253
HST, 253, 256, 257
Huffman coding, 270
hypercube, 106
    d-dimensional hypercube, 181
Håstad, J., 28

identity element, 338
Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming, 284
independent, 24, 55
indicator variable, 27
Indyk, P., 161, 191, 192, 260
Indyk, Piotr, 191
inequality
    Hoeffding, 103
Intorudction to Graph Theory, 216
Introduction to Algorithms, 73
inverse, 57, 66
Invitation to Discrete Mathematics, 194
irreducible, 202

Jacobi symbol, 343
Johnson, W. B., 161

Jones, Mitchell, 361

Kaklamanis, C., 107
Karlin, Anna R., 148
Karloff, H. J., 253, 260
Kelly criterion, 93
Kelly, J. L., 93
Khot, S., 284, 285
Kirchhoff's law, 207
Krauthgamer, R., 260
Krizanc, D., 107
Kushilevitz, E., 191

Laurent, B., 159
Law of quadratic reciprocity, 343, 344
lazy randomized incremental construction, 331
Lectures on Discrete Geometry, 260, 332
Legendre symbol, 342
level, 166, 315
    k-level, 315
Lindenstrauss, J., 161
Linearity of expectation, 25
Linial, N., 301
Lipschitz, 252
    bi-Lipschitz, 252
Lipschitz condition, 137
load, 144
load factor, 64
Locality-sensitive hashing scheme based on p-stable distributions, 190, 191
log, 369
lollipop graph, 206
long, 298
Lovász, L., 284
Lower bounds on locality sensitive hashing, 191
Lower Bounds on Probabilistic Linear Decision Trees, 150
Lower-Stretch Spanning Trees, 260
lucky, 228
lcm, 335
LSH, 183, 184, 189, 191

Markov chain, 200
    aperiodic, 202
    ergodic, 202
Markov Chains, 195, 202
martingale, 137
    edge exposure, 132
    vertex exposure, 132
martingale difference, 136
martingale sequence, 132
Massart, P., 159
Matias, Yossi, 176
Matoušek, J., 103, 123, 194, 260, 330–332
max cut, 303
maximization problem, 27
maximum cut, 303
maximum cut problem, 281
McDiarmid, C., 103
measure, 235
Measured descent: A new embedding method for finite metric spaces, 260
median, 357
median estimator, 170
Mehlhorn, K., 331
memorylessness property, 200
Mendel, M., 229, 301
metric, 227, 251
metric space, 227, 251–261
Miller, G. L., 348
mincut, 113
Mitzenmacher, M., 109, 268, 274, 280
Modern Graph Theory, 216
moments technique, 330
    all regions, 323
monomial, 45
Mossel, E., 285
Motwani, R., 48, 54, 79, 103, 109, 119, 134, 161, 191, 192, 224, 348
Mulmuley, K., 331, 332
multi-dimensional normal distribution, 157
Multilevel Adaptive Hashing, 148
Munro, J. I., 167

Naor, A., 191, 285, 301
Nešetřil, J., 194
near neighbor
    data-structure
        approximate, 181, 184, 188, 190
Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions, 191
Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, 191
Nearly Tight Low Stretch Spanning Trees, 260
Neiman, Ofer, 260

net, 227
ε-net, **241**
ε-net theorem, **241**, **248**
Net and Prune: A Linear Time Algorithm for Euclidean Distance Problems, 76
New applications of random sampling in computational geometry, 330
Noise stability of functions with low influences invariance and optimality, 285
normal distribution, 155, 160, 189, 190
multi-dimensional, 157
Norris, J. R., 195, 202
NP, 28, 284, 303
complete, 27, 76, 281, 303, 304
hard, 27, 281

O'Donnell, R., 285
oblivious, 107
Ohm's law, 207
Oleszkiewicz, K., 285
On approximating arbitrary metrics by tree metrics, 260
On constants for cuttings in the plane, 331
On lazy randomized incremental construction, 331
On Separating Points by Lines, 361
On the Greedy Permutation and Counting Distances, 229
On the Power of Random Access Machines, 76
On the Probability of the Extinction of Families, 111
On the set multi-cover problem in geometric settings, 332
On the uniform convergence of relative frequencies of events to their probabilities, 236, 241
open ball, 251
Optimal inapproximability results for Max Cut and other 2-variable CSPs, 284, 285
OR-concentrator, 151
order, 339
orthonormal eigenvector basis, 289
Ostrovsky, R., 191

pairwise independent, 55
Panconesi, Alessandro, 103
Panigrahi, R., 191
Paterson, M., 167
perfect matching, 46
periodicity, 202
Polynomial time approximation schemes for Euclidean TSP and other geometric problems, 260
positive semidefinite, 284
prefix code, 269
prefix-free, 269
prime, 66, 335
prime factorization, 337
Probabilistic algorithm for testing primality, 348
Probabilistic algorithms, 76
Probabilistic approximations of metric space and its algorithmic application, 260
probabilistic distortion, 256
probabilities, 23
Probability
Amplification, 117
probability, 24
Probability and Computing – randomized algorithms and probabilistic analysis, 109, 268, 274, 280
probability density function, 177
probability measure, 23, 135
probability space, 24, 135
Probability that n random points are in convex position, 358
Problem
3SAT
3SAT Max, 27
problem
3SAT, 27, 28
Max 3SAT, 27
MAX CUT, 281
MAX-SAT, 304–306
Sorting Nuts and Bolts, 86
projection, 182
proper, 369

quadratic residue, 341
quotation
Carl XIV Johan, King of Sweden, 363
quotient, 57, 66, 335
quotient group, 338

Rabani, Y., 191, 253, 260
Rabin, M. O., 76, 348
Radon's theorem, 237
Raghavan, P., 48, 54, 79, 103, 109, 119, 134, 224, 348
Raichel, Benjamin, 76
Random Cayley Graphs and Expanders, 301

random graphs, 132
random incremental construction, 318, 323, 327
    lazy, 331
random sample, 241, 242, 248, 315, 320, 321, 324, 325, 327–330
    ε-sample, 241
random variable, 24, 256
random walk, 193
Random Walks and Electric Networks, 210
Randomized Algorithms, 48, 54, 79, 103, 109, 119, 134, 224, 348
randomized rounding, 305
Randomized search trees, 87
range, 235
Range searching, 192
range space, 235
    projection, 236
rank, 35, 39, 87
Rao, S., 260
Reingold, O., 301
relative pairwise distance, 218
remainder, 57, 66, 336
replacement product, 298
Reporting points in halfspaces, 332
resampling, 369
residue, 336
resistance, 207, 211
Riemann's Hypothesis and Tests for Primality, 348
Roichman, Y., 301
running-time
    expected, 87

Sahni, S., 87
sample
    ε-sample, 241
      ε-sample theorem, **241**
    ε-sample, 240
sample space, 23
Schönhage, Arnold, 76
Schrijver, A., 284
Schwartz, O., 301
Schwarzkopf, O., 330, 331
Schwob, C., 229
Seidel, R., 87, 229, 331
Selection and Sorting with Limited Storage, 167
sensitive function, 184
set
    defining, 323
    stopping, 323
shallow cuttings, 332
Shapira, A., 301
Sharir, M., 331, 332
shatter function, 239
shattered, 236
Shor, Peter W., 330
short, 298
siblings, 369
Sidiropoulos, A., 229
sign, 46
Simple randomized algorithms for closest pair problems, 76
size, 63
Smid, M., 76
Snell, J. L., 210
Snir, Marc, 150
Some optimal inapproximability results, 28, 284
spectral gap, 218, 293
Spencer, J. H., 303, 361
spread, 257
squaring, 299
standard deviation, 25
standard normal distribution, 155
state
    aperiodic, 202
    ergodic, 202
    non null, 201
    null persistent, 201
    periodic, 202
    persistent, 201
    transient, 201
state probability vector, 202
Static, 63
stationary distribution, 202
Steinlight, E., 111
stochastic, 206
stopping set, 323
streaming, 167
strong component, 201
sub martingale, 136
subgraph
    unique, 310
subgroup, 338
successful, 122, 311
super martingale, 136

Surveys in Combinatorics, 103
symmetric, 215
Szegedy, Mario, 176

Taking a Walk in a Planar Arrangement, 331
Talwar, K., 260
tension, 288
The Probabilistic Method, 303, 361
The Space Complexity of Approximating the Frequency Moments, 176
The Clarkson-Shor Technique Revisited and Extended, 331
theorem
    ε-net, **241**, **248**
    Radon's, 237
    ε-sample, **241**
Tight bounds for oblivious routing in the hypercube, 107
Towards a calculus for non-linear spectral gaps, 301
transition matrix, 218, 287
transition probabilities matrix, 200
transition probability, 200
traverse, 212
treap, 84
tree
    code trees, 269
    prefix tree, 269
triangle, 356
true, 186
Tsantilas, T., 107
Turing machine
    log space, 212

union bound, 24
uniqueness, 75
universal traversal sequence, 212
Upfal, U., 109, 268, 274, 280

Vöcking, Berthold, 148
Vadhan, S., 301
Valtr, P., 358
Vandenberghe, L., 284
Vandermonde matrix, 58
Vapnik, V. N., 236, 241
variance, 25
VC
    dimension, 236
vertex, 315
vertical decomposition, 318
    vertex, 318
Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications, 332
vertical trapezoid, 318
vertices, 317
violates, 369
volume, 352

walk, 212
Watson, H. W., 111
weight
    region, 323
Welzl, E., 241, 248
West, D. B., 216
Why are Women Redundant?, 111
Why Novels are Redundant: Sensation Fiction and the Overpopulation of Literature, 111
width, 73
Wigderson, A., 301
Williamson, D. P., 284
witness tree, 369
word, 173

Yvinec, M., 330, 332

zero, 45
zero set, 45
zig-zag, 298
zig-zag product, 298
zig-zag-zig path, 298