Chapter 9

Randomized Algorithms

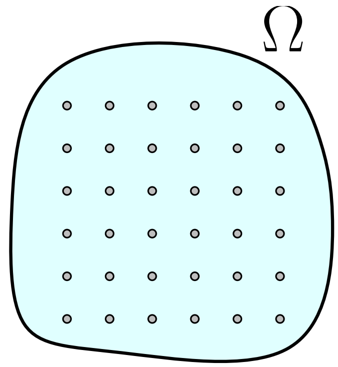
CS 573: Algorithms, Fall 2014
September 23, 2014

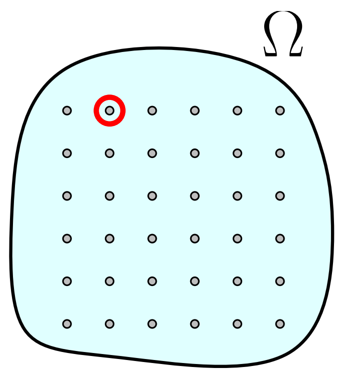
9.1 Randomized Algorithms

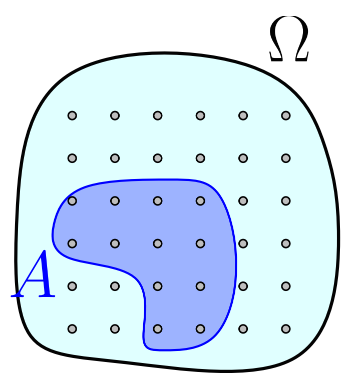
9.2 Some Probability

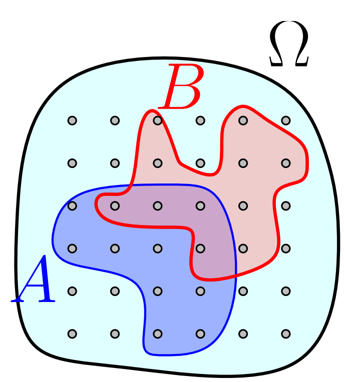
9.2.1 Probability - quick review

9.2.1.1 With pictures









(A) Ω: Sample space
(B) Ω: Is a set of ***elementary event/atomic event/simple event***.
(C) Every atomic event $x \in \Omega$ has ***Probability*** $\mathbf{Pr}[x]$.
(D) $X \equiv f(x)$: Random variable associate a value with each atomic event $x \in \Omega$.
(E) $\mathbf{E}[X]$: ***Expectation***:
The average value of the random variable $X \equiv f(x)$.
$\mathbf{E}[X] = \sum_{x \in X} f(x) * \mathbf{Pr}[X = x]$.
(F) An event $A \subseteq \Omega$ is a collection of atomic events.
$\mathbf{Pr}[A] = \sum_{a \in A} \mathbf{Pr}[a]$.

9.2.2 Probability - quick review

9.2.2.1 Definitions

Definition 9.2.1 (Informal). ***Random variable***: a function from probability space to $\mathbb{R}$. Associates value $\forall$ atomic events in probability space.

Definition The ***conditional probability*** of X given Y is

$$\mathbf{Pr}\Big[X = x \Big| Y = y\Big] = \frac{\mathbf{Pr}\Big[(X = x) \cap (Y = y)\Big]}{\mathbf{Pr}\Big[Y = y\Big]}.$$

Equivalent to

$$\mathbf{Pr}\Big[(X = x) \cap (Y = y)\Big] = \mathbf{Pr}\Big[X = x \;\Big|\; Y = y\Big] * \mathbf{Pr}\Big[Y = y\Big].$$

9.2.3 Probability - quick review

9.2.3.1 Even more definitions

Definition 9.2.2. The events $X = x$ and $Y = y$ are ***independent***, if

$$\mathbf{Pr}[X = x \cap Y = y] = \mathbf{Pr}[X = x] \cdot \mathbf{Pr}[Y = y].$$
$$\equiv \mathbf{Pr}\Big[X = x \;\Big|\; Y = y\Big] = \mathbf{Pr}[X = x].$$

Definition 9.2.3. The ***expectation*** of a random variable X its average value:

$$\mathbf{E}\Big[X\Big] = \sum_x x \cdot \mathbf{Pr}[X = x],$$

9.2.3.2 Linearity of expectations

Lemma 9.2.4 (Linearity of expectation.). *$\forall$ random variables X and Y:* $\mathbf{E}\Big[X + Y\Big] = \mathbf{E}\Big[X\Big] + \mathbf{E}\Big[Y\Big]$.

Proof: Use definitions, do the math. See notes for details. ∎

9.2.4 Probability - quick review

9.2.4.1 Conditional Expectation

Definition 9.2.5. X, Y: random variables. The ***conditional expectation*** of X given Y (i.e., you know $Y = y$):

$$\mathbf{E}\Big[X \;\Big|\; Y\Big] = \mathbf{E}\Big[X \;\Big|\; Y = y\Big] = \sum_x x * \mathbf{Pr}\Big[X = x \;\Big|\; Y = y\Big].$$

$\mathbf{E}[X]$ is a number.

$f(y) = \mathbf{E}\Big[X \;\Big|\; Y = y\Big]$ is a function.

9.2.4.2 Conditional Expectation

Lemma 9.2.6. $\forall\ X, Y$ *(not necessarily independent):* $\mathbf{E}[X] = \mathbf{E}\Big[\mathbf{E}\Big[X \;\Big|\, Y\Big]\Big]$.

$$\mathbf{E}\Big[\mathbf{E}\Big[X \;\Big|\, Y\Big]\Big] = \mathbf{E}_y\Big[\mathbf{E}\Big[X \;\Big|\, Y = y\Big]\Big]$$

Proof: Use definitions, and do the math. See class notes. ■

9.3 Sorting Nuts and Bolts

9.3.0.3 Sorting Nuts & Bolts

Problem 9.3.1 (**Sorting Nuts and Bolts**). (A) Input: Set n nuts + n bolts.

(B) Every nut have a matching bolt.

(C) All diff sizes.

(D) **Task:** Match nuts to bolts. (In sorted order).

(E) Restriction: You can only compare a nut to a bolt.

(F) Q: How to match the n nuts to the n bolts quickly?

9.3.1 Sorting nuts & bolts...

9.3.1.1 Algorithm

(A) Naive algorithm...
(B) ...better algorithm?

9.3.1.2 Sorting nuts & bolts...

```
MatchNutsAndBolts(N: nuts, B: bolts)
    Pick a random nut n_pivot from N
    Find its matching bolt b_pivot in B
    B_L ← All bolts in B smaller than n_pivot
    N_L ← All nuts in N smaller than b_pivot
    B_R ← All bolts in B larger than n_pivot
    N_R ← All nuts in N larger than b_pivot
    MatchNutsAndBolts(N_R,B_R)
    MatchNutsAndBolts(N_L,B_L)
```

QuickSort style...

9.3.2 Running time analysis

9.3.3 What is running time for randomized algorithms?

9.3.3.1 Definitions

Definition 9.3.2. $\mathcal{RT}(U)$: random variable – ***running time*** of the algorithm on input U.

Definition 9.3.3. Expected running time $\mathbf{E}[\mathcal{RT}(U)]$ for input U.

Definition 9.3.4. ***expected running-time*** of algorithm for input size n:

$$T(n) = \max_{U \text{ is an input of size } n} \mathbf{E}\Big[\mathcal{RT}(U)\Big].$$

9.3.4 What is running time for randomized algorithms?

9.3.4.1 More definitions

Definition 9.3.5. $\mathrm{rank}(x)$: ***rank*** of element $x \in S$ = number of elements in S smaller or equal to x.

9.3.4.2 Nuts and bolts running time

Theorem 9.3.6. *Expected running time* **MatchNutsAndBolts** *(***QuickSort***) is $T(n) = O(n \log n)$. Worst case is $O(n^2)$.*

Proof: $\mathbf{Pr}[\text{rank}(n_{pivot}) = k] = \frac{1}{n}$. Thus,

$$\begin{aligned} T(n) &= \mathop{\mathbf{E}}_{k=\text{rank}(n_{pivot})}\Big[O(n) + T(k-1) + T(n-k)\Big] \\ &= O(n) + \mathop{\mathbf{E}}_{k}[T(k-1) + T(n-k)] \\ &= O(n) + \sum_{k=1}^{n} \mathbf{Pr}[Rank(Pivot) = k] \\ &\qquad\qquad * (T(k-1) + T(n-k)) \\ &= O(n) + \sum_{k=1}^{n} \frac{1}{n} \cdot (T(k-1) + T(n-k))\,, \end{aligned}$$

Solution is $T(n) = O(n \log n)$. ■

9.3.4.3 Alternative incorrect solution

9.3.5 Alternative intuitive analysis...

9.3.5.1 Which is not formally correct

(A) **MatchNutsAndBolts** is ***lucky*** if $\frac{n}{4} \leq \text{rank}(n_{pivot}) \leq \frac{3}{4}n$.
(B) $\mathbf{Pr}[\text{“lucky”}] = 1/2$.
(C) $T(n) \leq O(n) + \mathbf{Pr}[\text{“lucky”}] * (T(n/4) + T(3n/4)) + \mathbf{Pr}[\text{“unlucky”}] * T(n)$.
(D) $T(n) = O(n) + \frac{1}{2} * \Big(T(\frac{n}{4}) + T(\frac{3}{4}n)\Big) + \frac{1}{2}T(n)$.
(E) Rewriting: $T(n) = O(n) + T(n/4) + T((3/4)n)$.
(F) ... solution is $O(n \log n)$.

9.3.6 What are randomized algorithms?

9.3.6.1 Worst case vs. average case

Expected running time of a randomized algorithm is

$$T(n) = \max_{U \text{ is an input of size } n} \mathbf{E}\Big[\mathcal{RT}(U)\Big]\,,$$

Worst case running time of deterministic algorithm:

$$T(n) = \max_{U \text{ is an input of size } n} \mathcal{RT}(U),$$

9.3.6.2 High Probability running time...

Definition 9.3.7. Running time **Alg** is $O(f(n))$ with ***high probability*** if

$$\mathbf{Pr}\Big[\mathcal{RT}(\mathbf{Alg}(n)) \geq c \cdot f(n)\Big] = o(1).$$

$\implies \mathbf{Pr}\Big[\mathcal{RT}(\mathbf{Alg}) > c * f(n)\Big] \to 0$ as $n \to \infty$.

Usually use weaker def:

$$\mathbf{Pr}\Big[\mathcal{RT}(\mathbf{Alg}(n)) \geq c \cdot f(n)\Big] \leq \frac{1}{n^d},$$

Technical reasons... also assume that $\mathbf{E}[\mathcal{RT}(\mathbf{Alg}(n))] = O(f(n))$.

9.4 Slick analysis of QuickSort

9.4.0.3 A Slick Analysis of QuickSort

Let $Q(A)$ be number of comparisons done on input array A:

(A) For $1 \leq i < j < n$ let R_{ij} be the event that rank i element is compared with rank j element.

(B) X_{ij}: ***indicator random*** variable for R_{ij}.
$X_{ij} = 1 \iff$ rank i element compared with rank j element, otherwise 0.

$$Q(A) = \sum_{1 \leq i < j \leq n} X_{ij}$$

and hence by linearity of expectation,

$$\mathbf{E}\Big[Q(A)\Big] = \sum_{1 \leq i < j \leq n} \mathbf{E}\Big[X_{ij}\Big] = \sum_{1 \leq i < j \leq n} \mathbf{Pr}\Big[R_{ij}\Big].$$

9.4.0.4 A Slick Analysis of QuickSort

R_{ij} = rank i element is compared with rank j element.

Question: What is $\mathbf{Pr}[R_{ij}]$?

7 5 9 1 3 4 8 6

With ranks:

7	5	9	1	3	4	8	6
6	4	8	1	2	3	7	5

As such, probability of comparing 5 to 8 is $\mathbf{Pr}[R_{4,7}]$.

(A) If pivot too small (say 3 [rank 2]). Partition and call recursively:

[7 5 9 1 (3) 4 8 6] $\Longrightarrow$ [1] [3] [7 5 9 4 8 6]

Decision if to compare 5 to 8 is moved to subproblem.

(B) If pivot too large (say 9 [rank 8]):

[7 5 (9) 1 3 4 8 6] $\Longrightarrow$ [7 5 1 3 4 8 6] [9]

Decision if to compare 5 to 8 moved to subproblem.

9.4.1 A Slick Analysis of QuickSort

7	5	9	1	3	4	8	6
6	4	8	1	2	3	7	5

As such, probability of comparing 5 to 8 is $\mathbf{Pr}[R_{4,7}]$.

9.4.1.1 Question: What is $\mathbf{Pr}[R_{i,j}]$?

(A) If pivot is 5 (rank 4). Bingo!

[7 (5) 9 1 3 4 8 6] $\Longrightarrow$ [1 3 4] [5] [7 9 8 6]

(B) If pivot is 8 (rank 7). Bingo!

[7 5 9 1 3 4 (8) 6] $\Longrightarrow$ [7 5 1 3 4 6] [8] [9]

(C) If pivot in between the two numbers (say 6 [rank 5]):

[7 5 9 1 3 4 8 (6)] $\Longrightarrow$ [5 1 3 4] [6] [7 8 9]

5 and 8 will never be compared to each other.

9.4.2 A Slick Analysis of QuickSort

9.4.2.1 Question: What is $\mathbf{Pr}[R_{i,j}]$?

Conclusion:

$R_{i,j}$ happens if and only if:

ith or jth ranked element is the first pivot out of ith to jth ranked elements.

How to analyze this?

Thinking acrobatics!
(A) Assign every element in the array a random priority (say in $[0, 1]$).
(B) Choose pivot to be the element with lowest priority in subproblem.
(C) Equivalent to picking pivot uniformly at random (as **QuickSort** do).

9.4.3 A Slick Analysis of QuickSort

9.4.3.1 Question: What is $\mathbf{Pr}[R_{i,j}]$?

How to analyze this?

Thinking acrobatics!
(A) Assign every element in the array a random priority (say in $[0, 1]$).
(B) Choose pivot to be the element with lowest priority in subproblem.
$\implies$ $R_{i,j}$ happens if either i or j have lowest priority out of elements rank i to j,
There are $k = j - i + 1$ relevant elements.

$$\mathbf{Pr}\Big[R_{i,j}\Big] = \frac{2}{k} = \frac{2}{j - i + 1}.$$

9.4.3.2 A Slick Analysis of QuickSort

Question: What is $\mathbf{Pr}[R_{ij}]$?

Lemma 9.4.1. $\mathbf{Pr}\Big[R_{ij}\Big] = \frac{2}{j-i+1}$.

Proof: Let $a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n$ be elements of A in sorted order. Let $S = \{a_i, a_{i+1}, \ldots, a_j\}$

Observation: If pivot is chosen outside S then all of S either in left array or right array.

Observation: a_i and a_j separated when a pivot is chosen from S for the first time. Once separated no comparison.

Observation: a_i is compared with a_j if and only if either a_i or a_j is chosen as a pivot from S at separation... ∎

9.4.4 A Slick Analysis of QuickSort

9.4.4.1 Continued...

Lemma 9.4.2. $\mathbf{Pr}\left[R_{ij}\right] = \frac{2}{j-i+1}$.

Proof: Let $a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n$ be sort of A. Let $S = \{a_i, a_{i+1}, \ldots, a_j\}$

Observation: a_i is compared with a_j if and only if either a_i or a_j is chosen as a pivot from S at separation.

Observation: Given that pivot is chosen from S the probability that it is a_i or a_j is exactly $2/|S| = 2/(j-i+1)$ since the pivot is chosen uniformly at random from the array. ∎

9.4.5 A Slick Analysis of QuickSort

9.4.5.1 Continued...

$$\mathbf{E}\left[Q(A)\right] = \sum_{1 \le i < j \le n} \mathbf{E}[X_{ij}] = \sum_{1 \le i < j \le n} \mathbf{Pr}[R_{ij}] \,.$$

Lemma 9.4.3. $\mathbf{Pr}[R_{ij}] = \frac{2}{j-i+1}$.

$$\begin{aligned}
\mathbf{E}\left[Q(A)\right] &= \sum_{1 \le i < j \le n} \mathbf{Pr}\left[R_{ij}\right] = \sum_{1 \le i < j \le n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \qquad\qquad = 2 \sum_{i=1}^{n-1} \sum_{i<j}^{n} \frac{1}{j-i+1} \le 2 \sum^{n}_{i=} \\
&\le 2 \sum_{i=1}^{n-1} (H_{n-i+1} - 1) \;\le\; 2 \sum_{1 \le i < n} H_n \\
&\le 2nH_n = O(n \log n)
\end{aligned}$$

9.5 Quick Select

9.6 Randomized Selection

9.6.0.2 Randomized Quick Selection

Input Unsorted array A of n integers

Goal Find the jth smallest number in A (*rank* j number)

Randomized Quick Selection

(A) Pick a pivot element *uniformly at random* from the array
(B) Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
(C) Return pivot if rank of pivot is j.
(D) Otherwise recurse on one of the arrays depending on j and their sizes.

9.6.0.3 Algorithm for Randomized Selection

Assume for simplicity that A has distinct elements.

```
QuickSelect(A, j):
    Pick pivot x uniformly at random fr
    Partition A into A_less, x, and A_grea
    if (|A_less| = j − 1) then
        return x
    if (|A_less| ≥ j) then
        return QuickSelect(A_less, j)
    else
        return QuickSelect(A_greater, j −
```

9.6.0.4 QuickSelect analysis

(A) $S_1, S_2, \ldots, S_k$ be the subproblems considered by the algorithm.
Here $|S_1| = n$.
(B) S_i would be ***successful*** if $|S_i| \leq (3/4)\,|S_{i-1}|$
(C) Y_1 = number of recursive calls till first successful iteration.
Clearly, total work till this happens is $O(Y_1 n)$.
(D) n_i = size of the subproblem immediately after the $(i-1)$th successful iteration.
(E) Y_i = number of recursive calls after the $(i-1)$th successful call, till the ith successful iteration.
(F) Running time is $O(\sum_i n_i Y_i)$.

9.6.0.5 QuickSelect analysis

Example

S_i = subarray used in ith recursive call
$|S_i|$ = size of this subarray
Red indicates successful iteration.

Inst'	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
$\|S_i\|$	100	70	60	50	40	30	25	5	2
Succ'	$Y_1 = 2$		$Y_2 = 4$				$Y_3 = 2$		$Y_4 = 1$
$n_i =$	$n_1 = 100$		$n_2 = 60$				$n_3 = 25$		$n_4 = 2$

(A) All the subproblems after $(i-1)$th successful iteration till ith successful iteration have size $\leq n_i$.
(B) Total work: $O(\sum_i n_i Y_i)$.

9.6.0.6 QuickSelect analysis

Total work: $O(\sum_i n_i Y_i)$.
We have:
(A) $n_i \leq (3/4) n_{i-1} \leq (3/4)^{i-1} n$.
(B) Y_i is a random variable with geometric distribution
Probability of $Y_i = k$ is $1/2^i$.
(C) $\mathbf{E}[Y_i] = 2$.
As such, expected work is proportional to

$$\mathbf{E}\Big[\sum_i n_i Y_i\Big] = \sum_i \mathbf{E}\Big[n_i Y_i\Big] \leq \sum_i \mathbf{E}\Big[(3/4)^{i-1} n Y_i\Big]$$
$$= n \sum_i (3/4)^{i-1}\,\mathbf{E}\Big[Y_i\Big] = n \sum_{i=1} (3/4)^{i-1} 2 \leq 8n.$$

9.6.0.7 QuickSelect analysis

Theorem 9.6.1. *The expected running time of* **QuickSelect** *is* $O(n)$.