Chapter 14

Introduction to Randomized Algorithms: QuickSort and QuickSelect

OLD CS 473: Fundamental Algorithms, Spring 2015
March 10, 2015

14.1 Introduction to Randomized Algorithms

14.2 Introduction

14.2.0.1 Randomized Algorithms

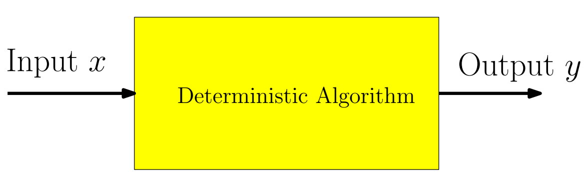
14.2.0.2 Example: Randomized QuickSort

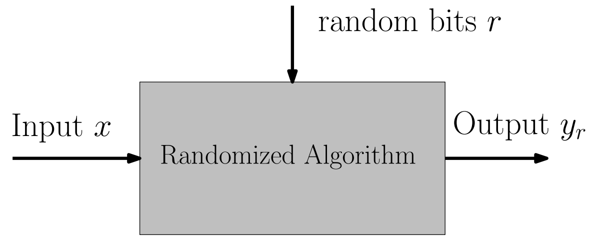
QuickSort Hoare [1962]

(A) Pick a pivot element from array
(B) Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
(C) Recursively sort the subarrays, and concatenate them.

Randomized **QuickSort**

(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) Recursively sort the subarrays, and concatenate them.





14.2.0.3 Example: Randomized Quicksort

Recall: **QuickSort** can take $\Omega(n^2)$ time to sort array of size n.

Theorem 14.2.1. *Randomized* **QuickSort** *sorts a given array of length n in $O(n \log n)$* expected *time.*

Note: On *every* input randomized **QuickSort** takes $O(n \log n)$ time in expectation. On *every* input it may take $\Omega(n^2)$ time with some small probability.

14.2.0.4 Example: Verifying Matrix Multiplication

Problem Given three $n \times n$ matrices A, B, C is $AB = C$?

Deterministic algorithm:

(A) Multiply A and B and check if equal to C.
(B) Running time? $O(n^3)$ by straight forward approach. $O(n^{2.37})$ with fast matrix multiplication (complicated and impractical).

14.2.0.5 Example: Verifying Matrix Multiplication

Problem Given three $n \times n$ matrices A, B, C is $AB = C$?

Randomized algorithm:

(A) Pick a random $n \times 1$ vector r.
(B) Return the answer of the equality $ABr = Cr$.
(C) Running time? $O(n^2)$!

Theorem 14.2.2. *If $AB = C$ then the algorithm will always say YES. If $AB \neq C$ then the algorithm will say YES with probability at most $1/2$. Can repeat the algorithm 100 times independently to reduce the probability of a false positive to $1/2^{100}$.*

14.2.0.6 Why randomized algorithms?

(A) Many applications: algorithms, data structures and CS.
(B) In some cases only known algorithms are randomized or randomness is provably necessary.
(C) Often randomized algorithms are (much) simpler and/or more efficient.
(D) Several deep connections to mathematics, physics etc.
(E) ...
(F) Lots of fun!

14.2.0.7 Where do I get random bits?

Question: Are true random bits available in practice?
(A) Buy them!
(B) CPUs use physical phenomena to generate random bits.
(C) Can use pseudo-random bits or semi-random bits from nature. Several fundamental unresolved questions in complexity theory on this topic. Beyond the scope of this course.
(D) In practice pseudo-random generators work quite well in many applications.
(E) The model is interesting to think in the abstract and is very useful even as a theoretical construct. One can *derandomize* randomized algorithms to obtain deterministic algorithms.

14.2.0.8 Average case analysis vs Randomized algorithms

Average case analysis:
(A) Fix a deterministic algorithm.
(B) Assume inputs comes from a probability distribution.
(C) Analyze the algorithm's *average* performance over the distribution over inputs.
Randomized algorithms:
(A) Algorithm uses random bits in addition to input.
(B) Analyze algorithms *average* performance over the given input where the average is over the random bits that the algorithm uses.
(C) On each input behaviour of algorithm is random. Analyze worst-case over all inputs of the (average) performance.

14.3 Basics of Discrete Probability

14.3.0.9 Discrete Probability

We restrict attention to finite probability spaces.

Definition 14.3.1. *A discrete probability space is a pair* $(\Omega, \mathbf{Pr})$ *consists of finite set* Ω *of* **elementary events** *and function* $p : \Omega \to [0, 1]$ *which assigns a probability* $\mathbf{Pr}[\omega]$ *for each* $\omega \in \Omega$ *such that* $\sum_{\omega\in\Omega} \mathbf{Pr}[\omega] = 1$.

Example 14.3.2. *An unbiased coin.* $\Omega = \{H, T\}$ *and* $\mathbf{Pr}[H] = \mathbf{Pr}[T] = 1/2$.

Example 14.3.3. *A* 6*-sided unbiased die.* $\Omega = \{1, 2, 3, 4, 5, 6\}$ *and* $\mathbf{Pr}[i] = 1/6$ *for* $1 \leq i \leq 6$.

14.3.1 Discrete Probability

14.3.1.1 And more examples

Example 14.3.4. *A biased coin.* $\Omega = \{H, T\}$ *and* $\mathbf{Pr}[H] = 2/3, \mathbf{Pr}[T] = 1/3$.

Example 14.3.5. *Two independent unbiased coins.* $\Omega = \{HH, TT, HT, TH\}$ *and* $\mathbf{Pr}[HH] = \mathbf{Pr}[TT] = \mathbf{Pr}[HT] = \mathbf{Pr}[TH] = 1/4$.

Example 14.3.6. *A pair of (highly) correlated dice.*
$\Omega = \{(i,j) \mid 1 \leq i \leq 6, 1 \leq j \leq 6\}$.
$\mathbf{Pr}[i,i] = 1/6$ *for* $1 \leq i \leq 6$ *and* $\mathbf{Pr}[i,j] = 0$ *if* $i \neq j$.

14.3.1.2 Events

Definition 14.3.7. *Given a probability space* $(\Omega, \mathbf{Pr})$ *an* **event** *is a subset of* Ω. *In other words an event is a collection of elementary events. The probability of an event* A*, denoted by* $\mathbf{Pr}[A]$*, is* $\sum_{\omega \in A} \mathbf{Pr}[\omega]$.

Definition 14.3.8. *The* **complement event** *of an event* $A \subseteq \Omega$ *is the event* $\Omega \setminus A$ *frequently denoted by* $\bar{A}$.

14.3.2 Events

14.3.2.1 Examples

Example 14.3.9. *A pair of independent dice.* $\Omega = \{(i,j) \mid 1 \leq i \leq 6, 1 \leq j \leq 6\}$.

(A) Let A *be the event that the sum of the two numbers on the dice is even.*
Then $A = \left\{ (i,j) \in \Omega \;\middle|\; (i+j) \text{ is even} \right\}$.
$\mathbf{Pr}[A] = |A|/36 = 1/2$.

(B) Let B *be the event that the first die has* 1. *Then* $B = \left\{ (1,1), (1,2), (1,3), (1,4), (1,5), (1,6) \right\}$.
$\mathbf{Pr}[B] = 6/36 = 1/6$.

14.3.2.2 Independent Events

Definition 14.3.10. *Given a probability space* $(\Omega, \mathbf{Pr})$ *and two events* A, B *are* **independent** *if and only if* $\mathbf{Pr}[A \cap B] = \mathbf{Pr}[A]\,\mathbf{Pr}[B]$. *Otherwise they are* dependent. *In other words* A, B *independent implies one does not affect the other.*

Example 14.3.11. *Two coins.* $\Omega = \{HH, TT, HT, TH\}$ *and* $\mathbf{Pr}[HH] = \mathbf{Pr}[TT] = \mathbf{Pr}[HT] = \mathbf{Pr}[TH] = 1/4$.

(A) A *is the event that the first coin is heads and* B *is the event that second coin is tails.* A, B *are independent.*

(B) A *is the event that the two coins are different.* B *is the event that the second coin is heads.* A, B *independent.*

14.3.3 Independent Events

14.3.3.1 Examples

Example 14.3.12. A *is the event that both are not tails and* B *is event that second coin is heads.* A, B *are dependent.*

14.3.4 Union bound

14.3.4.1 The probability of the union of two events, is $\leq$ the probability of the sum of their probabilities.

Lemma 14.3.13. *For any two events $\mathcal{E}$ and $\mathcal{F}$, we have that $\mathbf{Pr}\left[\mathcal{E} \cup \mathcal{F}\right] \leq \mathbf{Pr}\left[\mathcal{E}\right] + \mathbf{Pr}\left[\mathcal{F}\right]$.*

Proof: Consider $\mathcal{E}$ and $\mathcal{F}$ to be a collection of elmentery events (which they are). We have

$$\begin{aligned}\mathbf{Pr}\left[\mathcal{E} \cup \mathcal{F}\right] &= \sum_{x \in \mathcal{E} \cup \mathcal{F}} \mathbf{Pr}[x] \\ &\leq \sum_{x \in \mathcal{E}} \mathbf{Pr}[x] + \sum_{x \in \mathcal{F}} \mathbf{Pr}[x] = \mathbf{Pr}\left[\mathcal{E}\right] + \mathbf{Pr}\left[\mathcal{F}\right].\end{aligned}$$

■

14.3.4.2 Random Variables

Definition 14.3.14. *Given a probability space $(\Omega, \mathbf{Pr})$ a (real-valued) random variable X over Ω is a function that maps each elementary event to a real number. In other words $X : \Omega \to \mathbb{R}$.*

Example 14.3.15. *A 6-sided unbiased die. $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $\mathbf{Pr}[i] = 1/6$ for $1 \leq i \leq 6$.*
(A) $X : \Omega \to \mathbb{R}$ where $X(i) = i \mod 2$.
(B) $Y : \Omega \to \mathbb{R}$ where $Y(i) = i^2$.

Definition 14.3.16. *A* **binary random variable** *is one that takes on values in $\{0, 1\}$.*

14.3.4.3 Indicator Random Variables

Special type of random variables that are quite useful.

Definition 14.3.17. *Given a probability space $(\Omega, \mathbf{Pr})$ and an event $A \subseteq \Omega$ the indicator random variable X_A is a binary random variable where $X_A(\omega) = 1$ if $\omega \in A$ and $X_A(\omega) = 0$ if $\omega \notin A$.*

Example 14.3.18. *A 6-sided unbiased die. $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $\mathbf{Pr}[i] = 1/6$ for $1 \leq i \leq 6$. Let A be the even that i is divisible by 3. Then $X_A(i) = 1$ if $i = 3, 6$ and 0 otherwise.*

14.3.4.4 Expectation

Definition 14.3.19. *For a random variable X over a probability space $(\Omega, \mathbf{Pr})$ the* **expectation** *of X is defined as $\sum_{\omega \in \Omega} \mathbf{Pr}[\omega] X(\omega)$. In other words, the expectation is the average value of X according to the probabilities given by $\mathbf{Pr}[\cdot]$.*

Example 14.3.20. *A 6-sided unbiased die. $\Omega = \{1, 2, 3, 4, 5, 6\}$ and $\mathbf{Pr}[i] = 1/6$ for $1 \leq i \leq 6$.*
(A) $X : \Omega \to \mathbb{R}$ where $X(i) = i \mod 2$. Then $\mathbf{E}[X] = 1/2$.
(B) $Y : \Omega \to \mathbb{R}$ where $Y(i) = i^2$. Then $\mathbf{E}[Y] = \sum_{i=1}^{6} \frac{1}{6} \cdot i^2 = 91/6$.

14.3.4.5 Expectation

Proposition 14.3.21. *For an indicator variable X_A, $\mathbf{E}[X_A] = \mathbf{Pr}[A]$.*

Proof:

$$\begin{aligned}\mathbf{E}[X_A] &= \sum_{y\in\Omega} X_A(y)\,\mathbf{Pr}[y] \\ &= \sum_{y\in A} 1\cdot\mathbf{Pr}[y] + \sum_{y\in\Omega\setminus A} 0\cdot\mathbf{Pr}[y] \\ &= \sum_{y\in A}\mathbf{Pr}[y] \\ &= \mathbf{Pr}[A]\,.\end{aligned}$$

■

14.3.4.6 Linearity of Expectation

Lemma 14.3.22. *Let X, Y be two random variables (not necessarily independent) over a probability space $(\Omega, \mathbf{Pr})$. Then $\mathbf{E}[X+Y] = \mathbf{E}[X] + \mathbf{E}[Y]$.*

Proof:

$$\begin{aligned}\mathbf{E}[X+Y] &= \sum_{\omega\in\Omega}\mathbf{Pr}[\omega]\,(X(\omega)+Y(\omega)) \\ &= \sum_{\omega\in\Omega}\mathbf{Pr}[\omega]\,X(\omega) + \sum_{\omega\in\Omega}\mathbf{Pr}[\omega]\,Y(\omega) = \mathbf{E}[X] + \mathbf{E}[Y]\,.\end{aligned}$$

■

Corollary 14.3.23. $\mathbf{E}[a_1X_1 + a_2X_2 + \ldots + a_nX_n] = \sum_{i=1}^{n} a_i\,\mathbf{E}[X_i]$.

14.4 Analyzing Randomized Algorithms

14.4.0.7 Types of Randomized Algorithms

Typically one encounters the following types:

(A) ***Las Vegas randomized algorithms:*** for a given input x output of algorithm is always correct but the running time is a random variable. In this case we are interested in analyzing the *expected* running time.

(B) ***Monte Carlo randomized algorithms:*** for a given input x the running time is deterministic but the output is random; correct with some probability. In this case we are interested in analyzing the *probability* of the correct output (and also the running time).

(C) Algorithms whose running time and output may both be random variables.

14.4.0.8 Analyzing Las Vegas Algorithms

Deterministic algorithm Q for a problem Π:

(A) Let $Q(x)$ be the time for Q to run on input x of length $|x|$.
(B) Worst-case analysis: run time on worst input for a given size n.

$$T_{wc}(n) = \max_{x:|x|=n} Q(x).$$

Randomized algorithm R for a problem Π:

(A) Let $R(x)$ be the time for Q to run on input x of length $|x|$.
(B) $R(x)$ is a random variable: depends on random bits used by R.
(C) $\mathbf{E}[R(x)]$ is the expected running time for R on x
(D) Worst-case analysis: expected time on worst input of size n

$$T_{rand-wc}(n) = \max_{x:|x|=n} \mathbf{E}[Q(x)].$$

14.4.0.9 Analyzing Monte Carlo Algorithms

Randomized algorithm M for a problem Π:

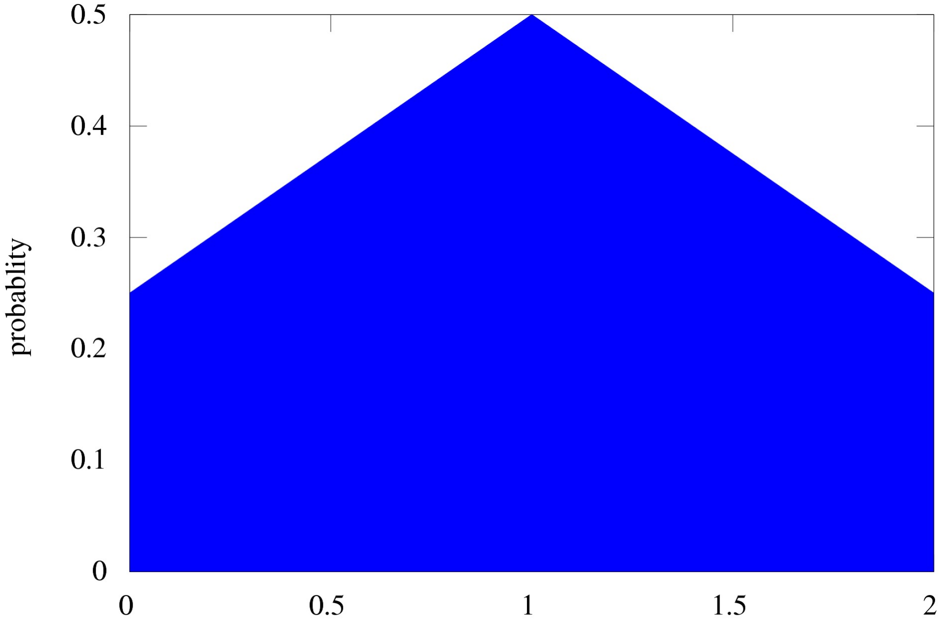
(A) Let $M(x)$ be the time for M to run on input x of length $|x|$. For Monte Carlo, assumption is that run time is deterministic.
(B) Let $\mathbf{Pr}[x]$ be the probability that M is correct on x.
(C) $\mathbf{Pr}[x]$ is a random variable: depends on random bits used by M.
(D) Worst-case analysis: success probability on worst input

$$P_{rand-wc}(n) = \min_{x:|x|=n} \mathbf{Pr}[x].$$

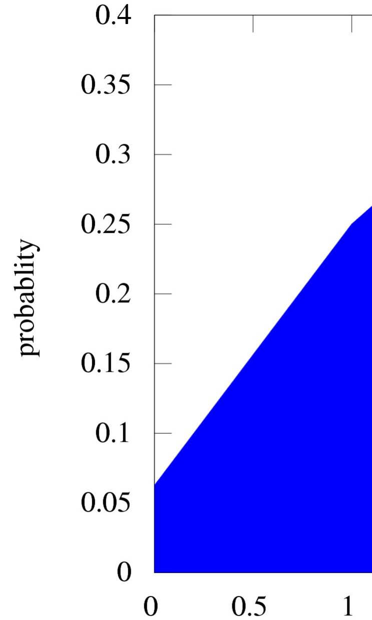
14.5 Why does randomization help?

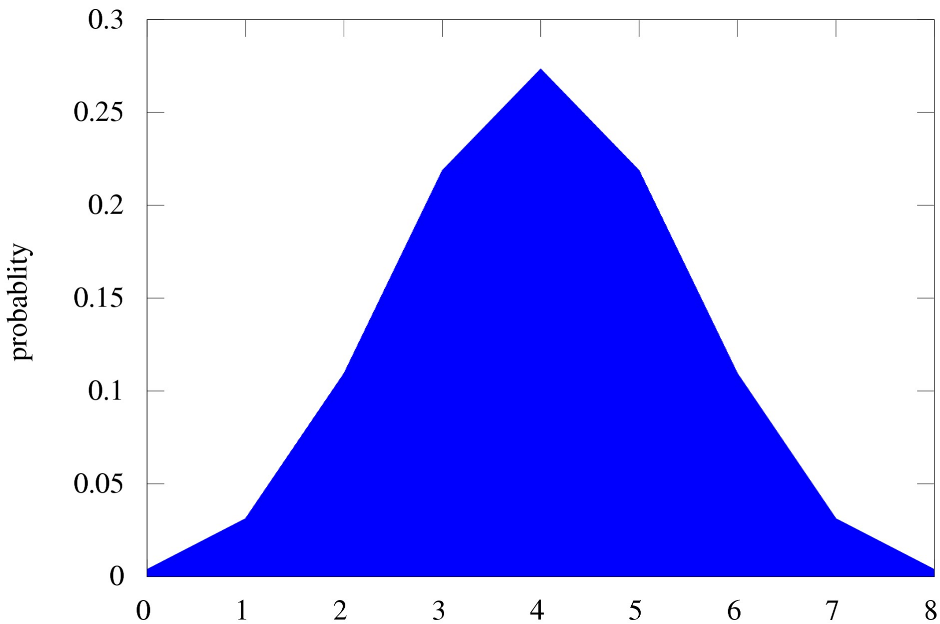
14.5.0.10 Massive randomness.. Is not that random.

Consider flipping a fair coin n times independently, head given 1, tail gives zero. How many heads? ...we get a binomial distribution.

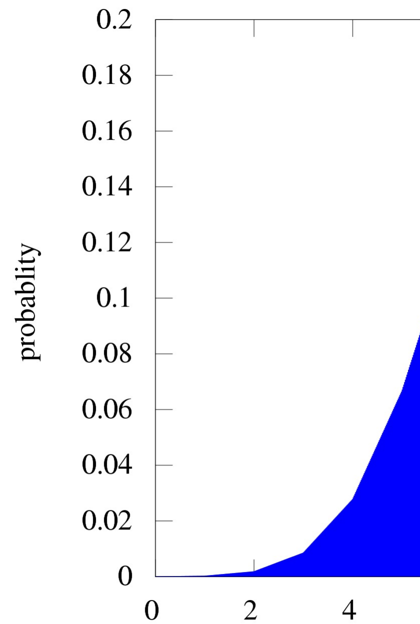


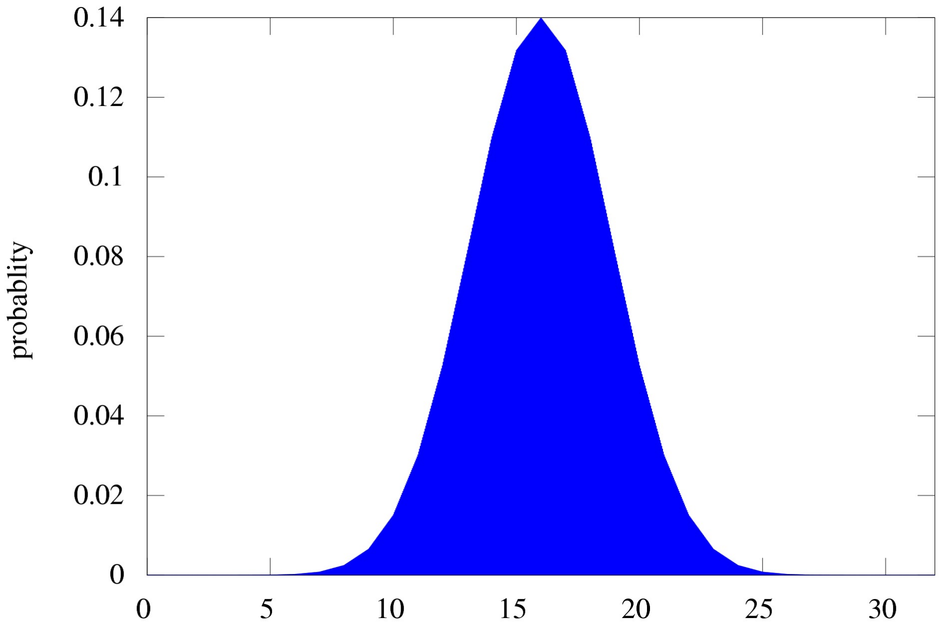
n = 2



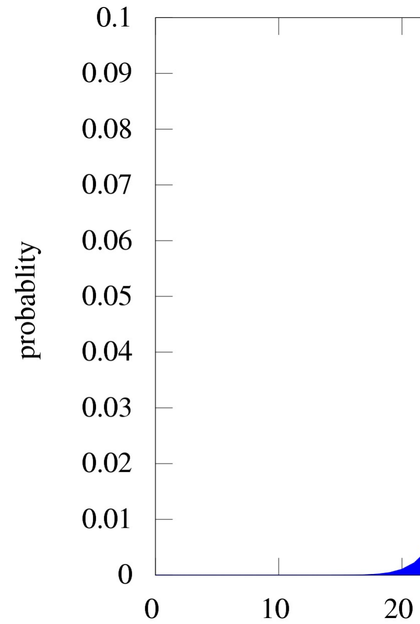


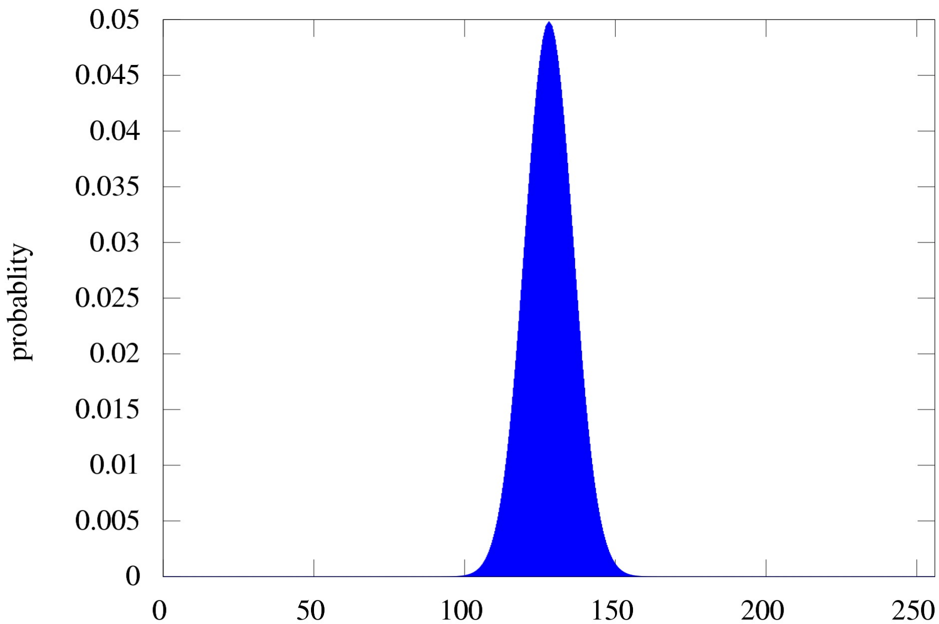
n = 8



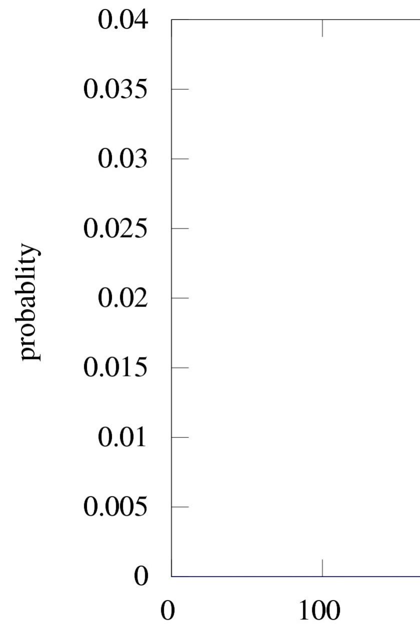

probablity
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
0
5
10
15
20
25
30

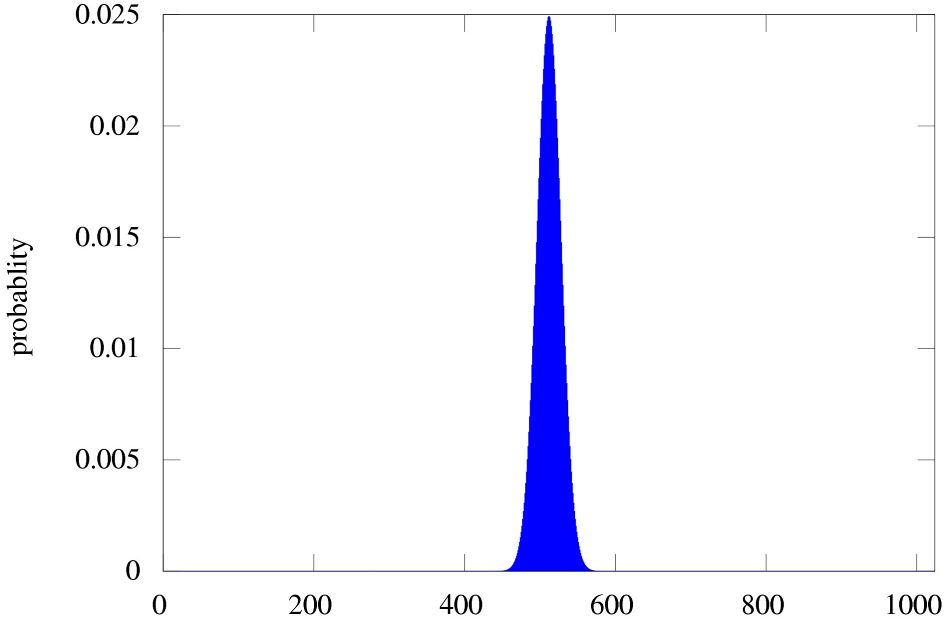
n = 32


probablity
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
10
20

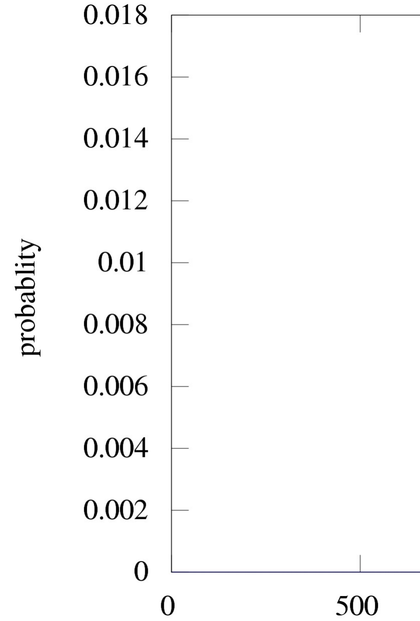

probablity
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
50
100
150
200
250

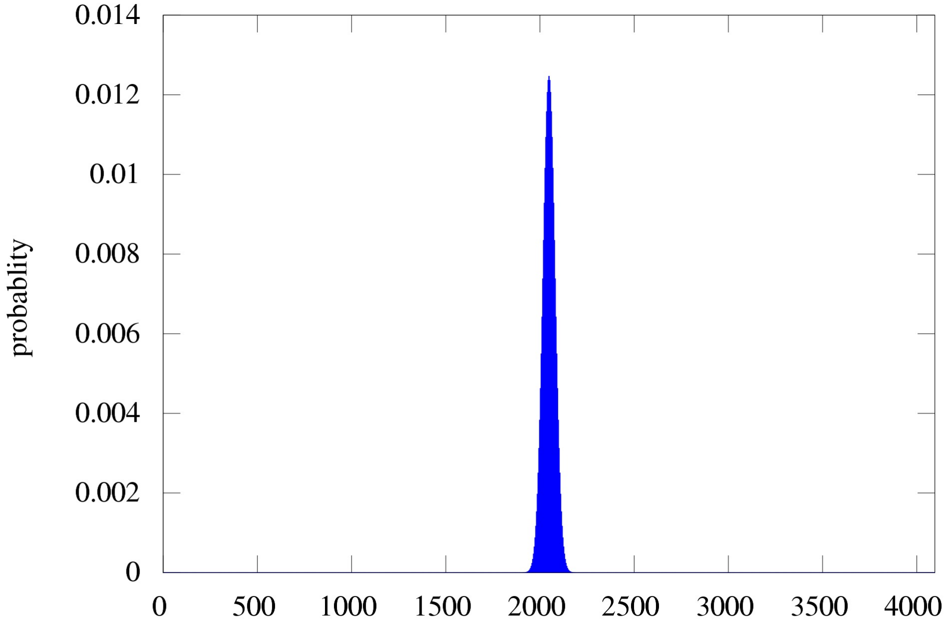
n = 256


probablity
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
100

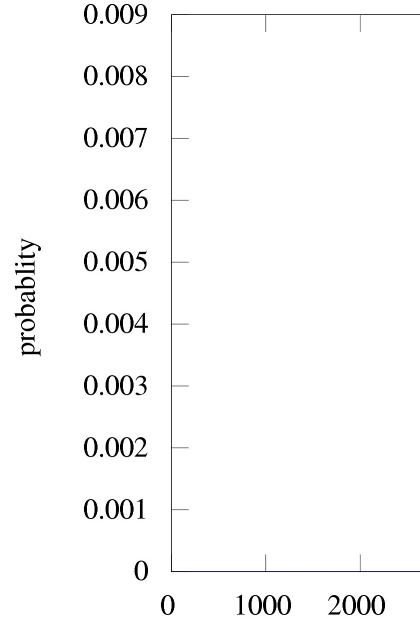

probablity
0.025
0.02
0.015
0.01
0.005
0
0
200
400
600
800
1000

n = 1024

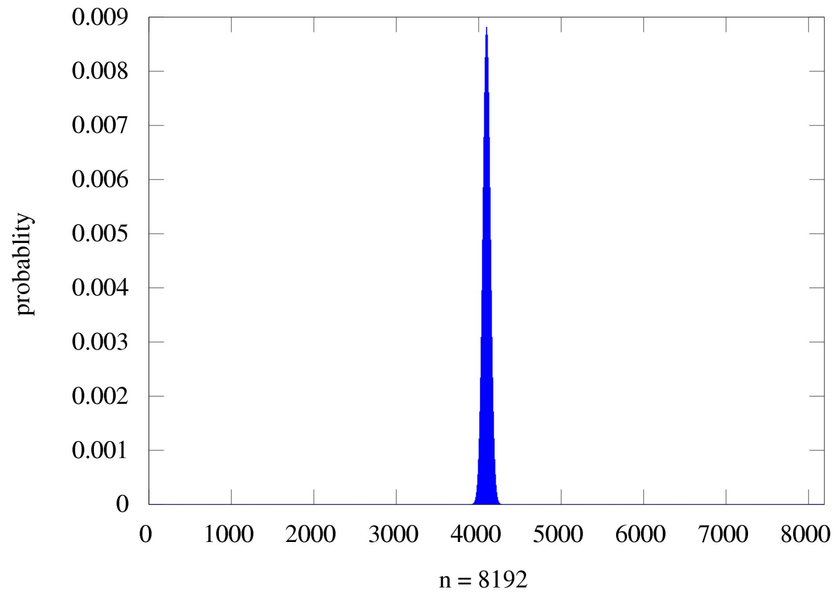

probablity
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
500


probablity
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
0
500
1000
1500
2000
2500
3000
3500
4000

n = 4096


probablity
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0
0
1000
2000

14.5.0.11 Massive randomness.. Is not that random.



This is known as ***concentration of mass***.

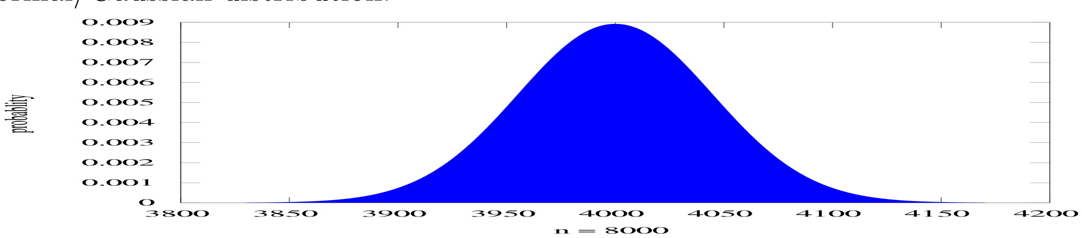
This is a very special case of the ***law of large numbers***.

14.5.1 Side note...

14.5.1.1 Law of large numbers (weakest form)...

Informal statement of law of large numbers

For n large enough, the middle portion of the binomial distribution looks like (converges to) the normal/Gaussian distribution.



14.5.1.2 Massive randomness.. Is not that random.

Intuitive conclusion

Randomized algorithm are unpredictable in the tactical level, but very predictable in the strategic level.

14.5.1.3 Binomial distribution

X_n = numbers of heads when flipping a coin n times.

Claim

$\mathbf{Pr}\Big[X_n = i\Big] = \frac{\binom{n}{i}}{2^n}$.

Where: $\binom{n}{k} = \frac{n!}{(n-k)!k!}$.

Indeed, $\binom{n}{i}$ is the number of ways to choose i elements out of n elements (i.e., pick which i coin flip come up heads).

Each specific such possibility (say 0100010...) had probability $1/2^n$.

We are interested in the bad event $\mathbf{Pr}[X_n \leq n/4]$ (way too few heads). We are going to prove this probability is tiny.

14.5.2 Binomial distribution

14.5.2.1 Playing around with binomial coefficients

Lemma 14.5.1. *$n! \geq (n/e)^n$.*

Proof:

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

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

14.5.3 Binomial distribution

14.5.3.1 Playing around with binomial coefficients

Lemma 14.5.2. *For any $k \leq n$, we have $\binom{n}{k} \leq \left(\frac{ne}{k}\right)^k$.*

Proof:

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \frac{n(n-1)(n-2)\ldots(n-k+1)}{k!} \leq \frac{n^k}{k!} \leq \frac{n^k}{\left(\frac{k}{e}\right)^k} = \left(\frac{ne}{k}\right)^k.$$

since $k! \geq (k/e)^k$ (by previous lemma). ■

14.5.4 Binomial distribution

14.5.4.1 Playing around with binomial coefficients

$$\mathbf{Pr}\Big[X_n \leq \frac{n}{4}\Big] = \sum_{k=0}^{n/4} \frac{1}{2^n}\binom{n}{k} = \frac{1}{2^n}\sum_{k=0}^{n/4}\binom{n}{k} \leq \frac{1}{2^n} 2 \cdot \binom{n}{n/4}$$

For $k \leq n/4$ the above sequence behave like a geometric variable.

$$\binom{n}{k+1} / \binom{n}{k} = \frac{n!}{(k+1)!(n-k-1)!} \Big/ \frac{n!}{(k)!(n-k)!}$$
$$= \frac{n-k}{k+1} \geq \frac{(3/4)n}{n/4+1} \geq 2.$$

14.5.5 Binomial distribution

14.5.5.1 Playing around with binomial coefficients

$$\mathbf{Pr}\left[X_n \leq \frac{n}{4}\right] \leq \frac{1}{2^n} 2 \cdot \binom{n}{n/4} \leq \frac{1}{2^n} 2 \cdot \left(\frac{ne}{n/4}\right)^{n/4} \leq 2 \cdot \left(\frac{4e}{2^4}\right)^{n/4}$$
$$\leq 2 \cdot 0.68^{n/4}.$$

We just proved the following theorem.

Theorem 14.5.3. *Let X_n be the random variable which is the number of heads when flipping an unbiased coin independently n times. Then*

$$\mathbf{Pr}\left[X_n \leq \frac{n}{4}\right] \leq 2 \cdot 0.68^{n/4} \text{ and } \mathbf{Pr}\left[X_n \geq \frac{3n}{4}\right] \leq 2 \cdot 0.68^{n/4}.$$

14.6 Randomized Quick Sort and Selection

14.7 Randomized Quick Sort

14.7.0.2 Randomized QuickSort

Randomized **QuickSort**

(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) Recursively sort the subarrays, and concatenate them.

14.7.0.3 Example

(A) array: 16, 12, 14, 20, 5, 3, 18, 19, 1

14.7.0.4 Analysis via Recurrence

(A) Given array A of size n, let $Q(A)$ be number of comparisons of randomized **QuickSort** on A.
(B) Note that $Q(A)$ is a random variable.
(C) Let A^i_{left} and A^i_{right} be the left and right arrays obtained if:
pivot is of rank i in A.

$$Q(A) = n + \sum_{i=1}^{n} \mathbf{Pr}\Big[\text{pivot has rank } i\Big] \left(Q(A^i_{\text{left}}) + Q(A^i_{\text{right}})\right).$$

Since each element of A has probability exactly of $1/n$ of being chosen:

$$Q(A) = n + \sum_{i=1}^{n} \frac{1}{n} \left(Q(A^i_{\text{left}}) + Q(A^i_{\text{right}})\right).$$

14.7.0.5 Analysis via Recurrence

Let $T(n) = \max_{A:|A|=n} \mathbf{E}[Q(A)]$ be the worst-case expected running time of randomized **QuickSort** on arrays of size n.

We have, for any A:

$$Q(A) = n + \sum_{i=1}^{n} \mathbf{Pr}\Big[\text{pivot has rank } i\Big] \left(Q(A^i_{\text{left}}) + Q(A^i_{\text{right}})\right)$$

Therefore, by linearity of expectation:

$$\mathbf{E}\Big[Q(A)\Big] = n + \sum_{i=1}^{n} \mathbf{Pr}\left[\begin{array}{c}\text{pivot is}\\ \text{of rank } i\end{array}\right] \left(\mathbf{E}\Big[Q(A^i_{\text{left}})\Big] + \mathbf{E}\Big[Q(A^i_{\text{right}})\Big]\right).$$

$$\Rightarrow \quad \mathbf{E}\Big[Q(A)\Big] \le n + \sum_{i=1}^{n} \frac{1}{n} \left(T(i-1) + T(n-i)\right).$$

14.7.0.6 Analysis via Recurrence

Let $T(n) = \max_{A:|A|=n} \mathbf{E}[Q(A)]$ be the worst-case expected running time of randomized **QuickSort** on arrays of size n.

We derived:

$$\mathbf{E}\Big[Q(A)\Big] \le n + \sum_{i=1}^{n} \frac{1}{n} \left(T(i-1) + T(n-i)\right).$$

Note that above holds for any A of size n. Therefore

$$\max_{A:|A|=n} \mathbf{E}[Q(A)] = T(n) \le n + \sum_{i=1}^{n} \frac{1}{n} \left(T(i-1) + T(n-i)\right).$$

14.7.0.7 Solving the Recurrence

$$T(n) \leq n + \sum_{i=1}^{n} \frac{1}{n} \left(T(i-1) + T(n-i)\right)$$

with base case $T(1) = 0$.

Lemma 14.7.1. $T(n) = O(n \log n)$.

Proof: (Guess and) Verify by induction. ∎

Bibliography

Hoare, C. A. R. (1962). Quicksort. *Comput. J.*, 5(1):10–15.