

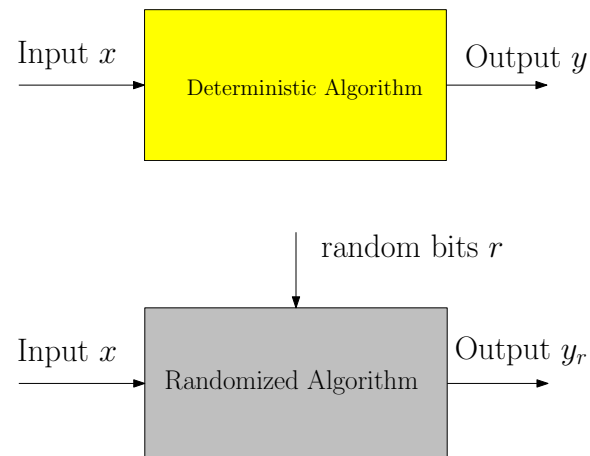
Introduction to Randomized Algorithms: QuickSort and QuickSelect

Lecture 13
March 6, 2013

Part I

Introduction to Randomized Algorithms

Randomized Algorithms



Example: Randomized QuickSort

QuickSort Hoare [1962]

- 1 Pick a pivot element from array
- 2 Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
- 3 Recursively sort the subarrays, and concatenate them.

Randomized QuickSort

- 1 Pick a pivot element *uniformly at random* from the array
- 2 Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
- 3 Recursively sort the subarrays, and concatenate them.

Example: Randomized Quicksort

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

Theorem

*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.

Example: Verifying Matrix Multiplication

Problem

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

Deterministic algorithm:

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

Example: Verifying Matrix Multiplication

Problem

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

Randomized algorithm:

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

Theorem

*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 $1/2^{100}$.*

Why randomized algorithms?

- 1 Many many applications in algorithms, data structures and computer science!
- 2 In some cases only known algorithms are randomized or randomness is provably necessary.
- 3 Often randomized algorithms are (much) simpler and/or more efficient.
- 4 Several deep connections to mathematics, physics etc.
- 5 ...
- 6 Lots of fun!

Where do I get random bits?

Question: Are true random bits available in practice?

- 1 Buy them!
- 2 CPUs use physical phenomena to generate random bits.
- 3 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.
- 4 In practice pseudo-random generators work quite well in many applications.
- 5 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.

Average case analysis vs Randomized algorithms

Average case analysis:

- 1 Fix a deterministic algorithm.
- 2 Assume inputs comes from a probability distribution.
- 3 Analyze the algorithm's *average* performance over the distribution over inputs.

Randomized algorithms:

- 1 Algorithm uses random bits in addition to input.
- 2 Analyze algorithms *average* performance over the given input where the average is over the random bits that the algorithm uses.
- 3 On each input behaviour of algorithm is random. Analyze worst-case over all inputs of the (average) performance.

Discrete Probability

We restrict attention to finite probability spaces.

Definition

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

Example

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

Example

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

Discrete Probability

And more examples

Example

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

Example

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

Example

A pair of (highly) correlated dice.

$\Omega = \{(i, j) \mid 1 \leq i \leq 6, 1 \leq j \leq 6\}$.

$\Pr[i, i] = 1/6$ for $1 \leq i \leq 6$ and $\Pr[i, j] = 0$ if $i \neq j$.

Events

Definition

Given a probability space (Ω, \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 $\Pr[A]$, is $\sum_{\omega \in A} \Pr[\omega]$.

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

Events

Examples

Example

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

- 1 Let A be the event that the sum of the two numbers on the dice is even.

$$\text{Then } A = \{(i, j) \in \Omega \mid (i + j) \text{ is even}\}.$$

$$\Pr[A] = |A|/36 = 1/2.$$

- 2 Let B be the event that the first die has 1. Then $B = \{(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6)\}$.

$$\Pr[B] = 6/36 = 1/6.$$

Independent Events

Definition

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

Example

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

- 1 A is the event that the first coin is heads and B is the event that second coin is tails. A, B are independent.
- 2 A is the event that the two coins are different. B is the event that the second coin is heads. A, B independent.

Independent Events

Examples

Example

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

Union bound

The probability of the union of two events, is no bigger than the probability of the sum of their probabilities.

Lemma

For any two events \mathcal{E} and \mathcal{F} , we have that

$$\Pr[\mathcal{E} \cup \mathcal{F}] \leq \Pr[\mathcal{E}] + \Pr[\mathcal{F}].$$

Proof.

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

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

Random Variables

Definition

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

Example

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

① $\mathbf{X} : \Omega \rightarrow \mathbb{R}$ where $\mathbf{X}(i) = i \bmod 2$.

② $\mathbf{Y} : \Omega \rightarrow \mathbb{R}$ where $\mathbf{Y}(i) = i^2$.

Definition

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

Indicator Random Variables

Special type of random variables that are quite useful.

Definition

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

Example

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

Expectation

Definition

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

Example

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

① $\mathbf{X} : \Omega \rightarrow \mathbb{R}$ where $\mathbf{X}(i) = i \bmod 2$. Then $\mathbf{E}[\mathbf{X}] = 1/2$.

② $\mathbf{Y} : \Omega \rightarrow \mathbb{R}$ where $\mathbf{Y}(i) = i^2$. Then

$$\mathbf{E}[\mathbf{Y}] = \sum_{i=1}^6 \frac{1}{6} \cdot i^2 = 91/6.$$

Expectation

Proposition

For an indicator variable X_A , $E[X_A] = \Pr[A]$.

Proof.

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

□

Linearity of Expectation

Lemma

Let X, Y be two random variables (not necessarily independent) over a probability space (Ω, \Pr) . Then $E[X + Y] = E[X] + E[Y]$.

Proof.

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

□

Corollary

$$E[a_1 X_1 + a_2 X_2 + \dots + a_n X_n] = \sum_{i=1}^n a_i E[X_i].$$

Types of Randomized Algorithms

Typically one encounters the following types:

- 1 **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.
- 2 **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).
- 3 Algorithms whose running time and output may both be random.

Analyzing Las Vegas Algorithms

Deterministic algorithm Q for a problem Π :

- 1 Let $Q(x)$ be the time for Q to run on input x of length $|x|$.
- 2 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 :

- 1 Let $R(x)$ be the time for Q to run on input x of length $|x|$.
- 2 $R(x)$ is a random variable: depends on random bits used by R .
- 3 $E[R(x)]$ is the expected running time for R on x
- 4 Worst-case analysis: expected time on worst input of size n

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

Analyzing Monte Carlo Algorithms

Randomized algorithm \mathbf{M} for a problem Π :

- 1 Let $\mathbf{M}(\mathbf{x})$ be the time for \mathbf{M} to run on input \mathbf{x} of length $|\mathbf{x}|$. For Monte Carlo, assumption is that run time is deterministic.
- 2 Let $\Pr[\mathbf{x}]$ be the probability that \mathbf{M} is correct on \mathbf{x} .
- 3 $\Pr[\mathbf{x}]$ is a random variable: depends on random bits used by \mathbf{M} .
- 4 Worst-case analysis: success probability on worst input

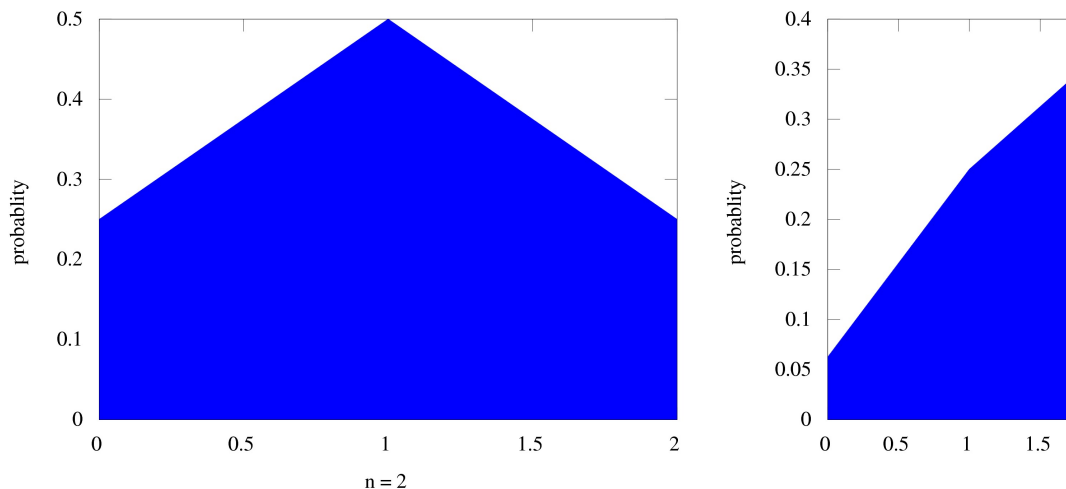
$$P_{\text{rand-wc}}(n) = \min_{\mathbf{x}: |\mathbf{x}|=n} \Pr[\mathbf{x}].$$

Part II

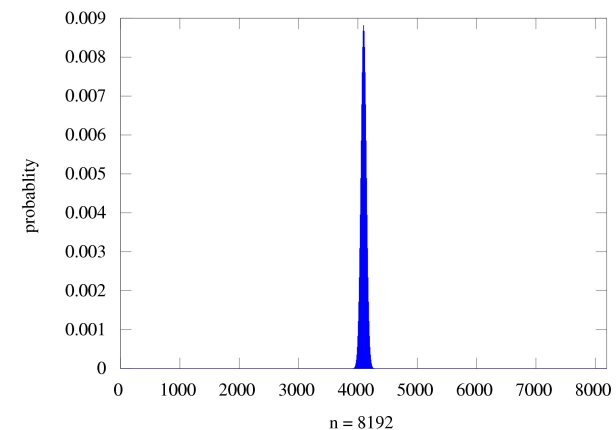
Why does randomization help?

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.



Massive randomness.. Is not that random.



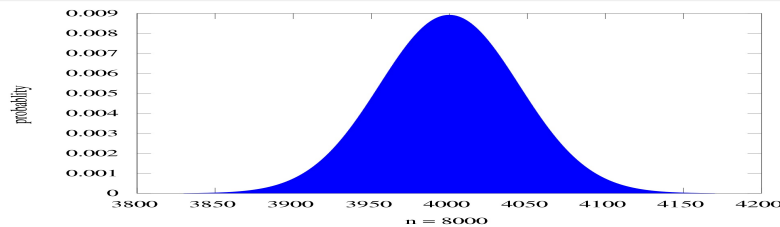
This is known as **concentration of mass**.
This is a very special case of the **law of large numbers**.

Side note...

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.



Massive randomness.. Is not that random.

Intuitive conclusion

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

Binomial distribution

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

Claim

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

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 $\Pr[X_n \leq n/4]$ (way too few heads). We are going to prove this probability is tiny.

Binomial distribution

Playing around with binomial coefficients

Lemma

$$n! \geq (n/e)^n.$$

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. \square

Binomial distribution

Playing around with binomial coefficients

Lemma

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

Proof.

$$\begin{aligned}\binom{n}{k} &= \frac{n!}{(n-k)!k!} = \frac{n(n-1)(n-2)\dots(n-k+1)}{k!} \\ &\leq \frac{n^k}{k!} \leq \frac{n^k}{\left(\frac{k}{e}\right)^k} = \left(\frac{ne}{k}\right)^k.\end{aligned}$$

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

Binomial distribution

Playing around with binomial coefficients

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

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

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

Binomial distribution

Playing around with binomial coefficients

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

We just proved the following theorem.

Theorem

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

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

Part III

Randomized Quick Sort and Selection

Randomized QuickSort

Randomized QuickSort

- 1 Pick a pivot element *uniformly at random* from the array.
- 2 Split array into 3 subarrays: those smaller than pivot, those larger than pivot, and the pivot itself.
- 3 Recursively sort the subarrays, and concatenate them.

Example

- 1 array: 16, 12, 14, 20, 5, 3, 18, 19, 1

Analysis via Recurrence

- 1 Given array \mathbf{A} of size n , let $Q(\mathbf{A})$ be number of comparisons of randomized **QuickSort** on \mathbf{A} .
- 2 Note that $Q(\mathbf{A})$ is a random variable.
- 3 Let $\mathbf{A}_{\text{left}}^i$ and $\mathbf{A}_{\text{right}}^i$ be the left and right arrays obtained if:

pivot is of rank i in \mathbf{A} .

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

Since each element of \mathbf{A} has probability exactly of $1/n$ of being chosen:

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

Analysis via Recurrence

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

We have, for any \mathbf{A} :

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

Therefore, by linearity of expectation:

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

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

Analysis via Recurrence

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

We derived:

$$E[Q(A)] \leq n + \sum_{i=1}^n \frac{1}{n} (T(i-1) + T(n-i)).$$

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

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

Solving the Recurrence

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

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

Lemma

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

Proof.

(Guess and) Verify by induction. □

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