Introduction to Randomized Algorithms: QuickSort and QuickSelect

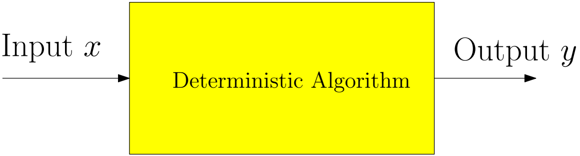
Lecture 13

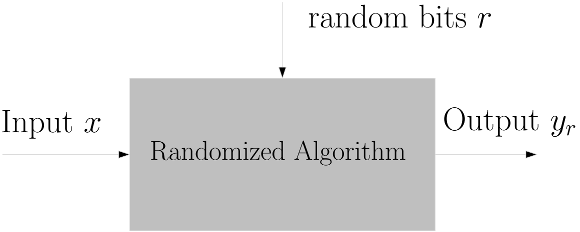
March 6, 2014

Part I

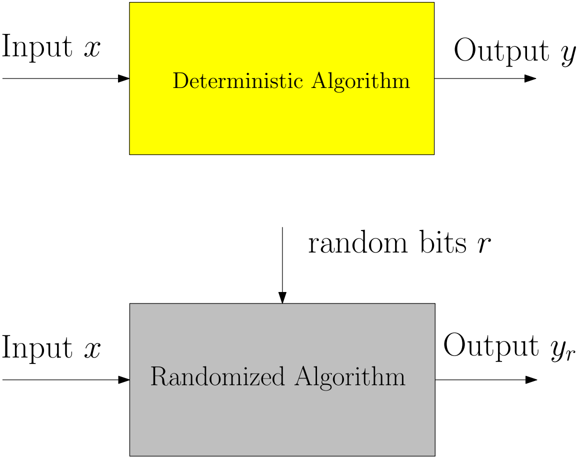
Introduction to Randomized Algorithms

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: 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: 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 $\mathbf{n} \times \mathbf{n}$ matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ is $\mathbf{AB} = \mathbf{C}$?

Deterministic algorithm:

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

Example: Verifying Matrix Multiplication

Problem

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

Deterministic algorithm:

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

Example: Verifying Matrix Multiplication

Problem

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

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 $\mathbf{n} \times \mathbf{n}$ matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ is $\mathbf{AB} = \mathbf{C}$?

Randomized algorithm:

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

Theorem

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

Example: Verifying Matrix Multiplication

Problem

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

Randomized algorithm:

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

Theorem

If $\mathbf{AB} = \mathbf{C}$ then the algorithm will always say YES. If $\mathbf{AB} \neq \mathbf{C}$ 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 $1/2^{100}$.

Example: Verifying Matrix Multiplication

Problem

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

Randomized algorithm:

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

Theorem

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

Example: Verifying Matrix Multiplication

Problem

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

Randomized algorithm:

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

Theorem

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

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 $p : \Omega \to [0, 1]$ 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

We restrict attention to finite probability spaces.

Definition

A discrete probability space is a pair $(\Omega, \Pr)$ consists of finite set Ω of **elementary events** and function $p : \Omega \rightarrow [0, 1]$ 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 $1 \leq i \leq 6$.

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

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

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.
   Then $A = \left\{(i, j) \in \Omega \;\middle|\; (i + j) \text{ is even}\right\}$.
   $\Pr[A] = |A|/36 = 1/2$.

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

Independent Events

Definition

Given a probability space $(\Omega, \Pr)$ 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

Definition

Given a probability space $(\Omega, \Pr)$ 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.

Example

$\mathbf{A}$ is the event that both are not tails and $\mathbf{B}$ is event that second coin is heads. $\mathbf{A}, \mathbf{B}$ are dependent.

Dependent or independent?

Consider two independent rolls of the dice.

1. **A** $=$ the event that the first roll is odd.
2. **B** $=$ the event that the sum of the two rolls is odd.

The events **A** and **B** are

(A) dependent.

(B) independent.

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

Proof.

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

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

Random Variables

Definition

Given a probability space $(\Omega, \Pr)$ 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

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

1. $X : \Omega \to \mathbb{R}$ where $X(i) = i \mod 2$.
2. $Y : \Omega \to \mathbb{R}$ where $Y(i) = i^2$.

Definition

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

Random Variables

Definition

Given a probability space $(\Omega, \Pr)$ 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

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

1. $X : \Omega \to \mathbb{R}$ where $X(i) = i \mod 2$.
2. $Y : \Omega \to \mathbb{R}$ where $Y(i) = i^2$.

Definition

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

Random Variables

Definition

Given a probability space $(\Omega, \Pr)$ 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

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

1. $X : \Omega \to \mathbb{R}$ where $X(i) = i \mod 2$.
2. $Y : \Omega \to \mathbb{R}$ where $Y(i) = i^2$.

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 $(\Omega, \Pr)$ 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

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

Indicator Random Variables

Special type of random variables that are quite useful.

Definition

Given a probability space $(\Omega, \Pr)$ 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

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

Expectation

Definition

For a random variable X over a probability space $(\Omega, \Pr)$ the **expectation** of X is defined as $\sum_{\omega \in \Omega} \Pr[\omega]\, X(\omega)$. In other words, the expectation is the average value of 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. $X : \Omega \to \mathbb{R}$ where $X(i) = i \mod 2$. Then $E[X] = 1/2$.
2. $Y : \Omega \to \mathbb{R}$ where $Y(i) = i^2$. Then $E[Y] = \sum_{i=1}^{6} \frac{1}{6} \cdot i^2 = 91/6$.

Expectation

Definition

For a random variable X over a probability space $(\Omega, \Pr)$ the **expectation** of X is defined as $\sum_{\omega \in \Omega} \Pr[\omega] \, X(\omega)$. In other words, the expectation is the average value of 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. $X : \Omega \to \mathbb{R}$ where $X(i) = i \mod 2$. Then $E[X] = 1/2$.
2. $Y : \Omega \to \mathbb{R}$ where $Y(i) = i^2$. Then $E[Y] = \sum_{i=1}^{6} \frac{1}{6} \cdot i^2 = 91/6$.

Expected number of vertices?

Let $G = (V, E)$ be a graph with n vertices and m edges. Let H be the graph resulting from independently deleting every vertex of G with probability $1/2$. The expected number of vertices in H is

(A) $n/2$.
(B) $n/4$.
(C) $m/2$.
(D) $m/4$.
(E) none of the above.

Expected number of edges?

Let $G = (V, E)$ be a graph with n vertices and m edges. Let H be the graph resulting from independently deleting every vertex of G with probability $1/2$. The expected number of edges in H is

(A) $n/2$.

(B) $n/4$.

(C) $m/2$.

(D) $m/4$.

(E) none of the above.

Expectation

Proposition

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

Proof.

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

$\square$

Linearity of Expectation

Lemma

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

Proof.

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

□

Corollary

$\mathbf{E}[a_1 \mathbf{X}_1 + a_2 \mathbf{X}_2 + \ldots + a_n \mathbf{X}_n] = \sum_{i=1}^{n} a_i \mathbf{E}[\mathbf{X}_i]$.

Linearity of Expectation

Lemma

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

Proof.

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

$\square$

Corollary

$\mathbf{E}[a_1 \mathbf{X}_1 + a_2 \mathbf{X}_2 + \ldots + a_n \mathbf{X}_n] = \sum_{i=1}^{n} a_i\, \mathbf{E}[\mathbf{X}_i]$.

Expected number of edges?

Let $G = (V, E)$ be a graph with n vertices and m edges. Let H be the graph resulting from independently deleting every vertex of G with probability $1/2$. The expected number of edges in H is

- **(A)** $n/2$.
- **(B)** $n/4$.
- **(C)** $m/2$.
- **(D)** $m/4$.
- **(E)** none of the above.

Expected number of triangles?

Let $G = (V, E)$ be a graph with n vertices and m edges. Assume G has t triangles (i.e., a triangle is a simple cycle with three vertices). Let H be the graph resulting from deleting independently each vertex of G with probability $1/2$. The expected number of triangles in H is

- **(A)** $t/2$.
- **(B)** $t/4$.
- **(C)** $t/8$.
- **(D)** $t/16$.
- **(E)** none of the above.

Types of Randomized Algorithms

Typically one encounters the following types:

1. **Las Vegas randomized algorithms:** for a given input $\mathbf{x}$ 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 $\mathbf{x}$ 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 $\mathbf{Q}$ for a problem $\mathbf{\Pi}$:

1. Let $\mathbf{Q(x)}$ be the time for $\mathbf{Q}$ to run on input $\mathbf{x}$ of length $|\mathbf{x}|$.
2. Worst-case analysis: run time on worst input for a given size $\mathbf{n}$.

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

Randomized algorithm $\mathbf{R}$ for a problem $\mathbf{\Pi}$:

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

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

Analyzing Las Vegas Algorithms

Deterministic algorithm $\mathbf{Q}$ for a problem $\mathbf{\Pi}$:

1. Let $\mathbf{Q(x)}$ be the time for $\mathbf{Q}$ to run on input $\mathbf{x}$ of length $|\mathbf{x}|$.
2. Worst-case analysis: run time on worst input for a given size $\mathbf{n}$.

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

Randomized algorithm $\mathbf{R}$ for a problem $\mathbf{\Pi}$:

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

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

Analyzing Monte Carlo Algorithms

Randomized algorithm $\mathbf{M}$ for a problem $\mathbf{\Pi}$:

1. Let $\mathbf{M(x)}$ 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 $\mathbf{Pr[x]}$ be the probability that $\mathbf{M}$ is correct on $\mathbf{x}$.
3. $\mathbf{Pr[x]}$ is a random variable: depends on random bits used by $\mathbf{M}$.
4. Worst-case analysis: success probability on worst input

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

Part II

Why does randomization help?

Ping and find.

Consider a deterministic algorithm $\mathbf{A}$ that is trying to find an element in an array $\mathbf{X}$ of size $\mathbf{n}$. At every step it is allowed to ask the value of one cell in the array, and the adversary is allowed after each such ping, to shuffle elements around in the array in any way it seems fit. For the best possible deterministic algorithm the number of rounds it has to play this game till it finds the required element is

(A) $\mathbf{O(1)}$

(B) $\mathbf{O(n)}$

(C) $\mathbf{O(n \log n)}$

(D) $\mathbf{O(n^2)}$

(E) ∞.

Ping and find randomized.

Consider an algorithm **randFind** that is trying to find an element in an array $\mathbf{X}$ of size $\mathbf{n}$. At every step it asks the value of one random cell in the array, and the adversary is allowed after each such ping, to shuffle elements around in the array in any way it seems fit. This algorithm would stop in expectation after

- **(A)** $\mathbf{O(1)}$
- **(B)** $\mathbf{O(\log n)}$
- **(C)** $\mathbf{O(n)}$
- **(D)** $\mathbf{O(n^2)}$
- **(E)** ∞.

steps.

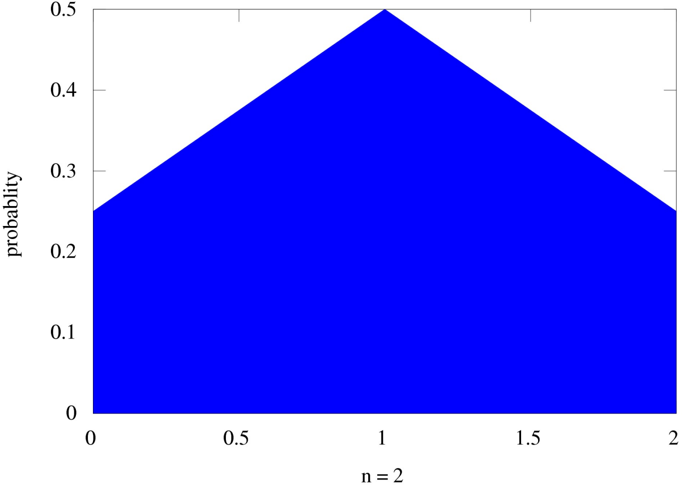
Abundance of witnesses

Consider the problem of finding an "approximate median" of an unsorted array $\mathbf{A[1..n]}$: an element of $\mathbf{A}$ with rank between $\mathbf{n/4}$ and $\mathbf{3n/4}$.

- We saw a complicated deterministic algorithm. Finding an approximate median is not any easier than a proper median.
- $\mathbf{n/2}$ elements of $\mathbf{A}$ qualify as approximate medians and hence a random element is good with probability $\mathbf{1/2}$!

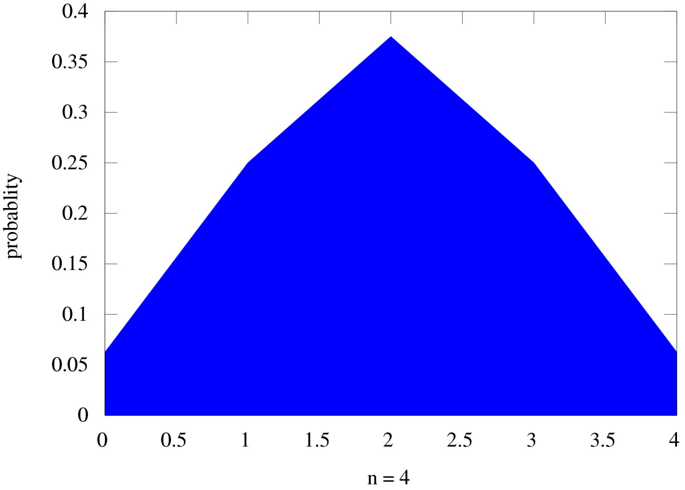
Massive randomness.. Is not that random.

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



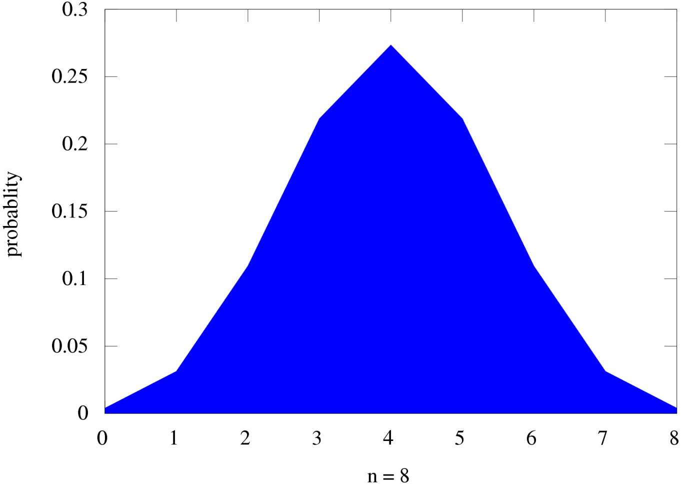
Massive randomness.. Is not that random.

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



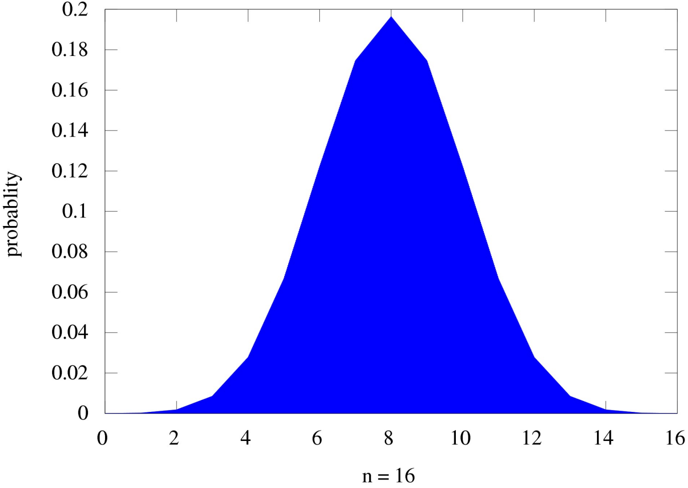
Massive randomness.. Is not that random.

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



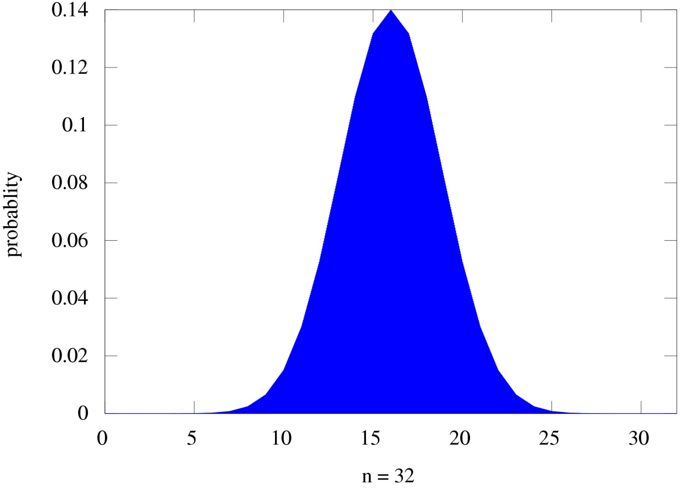
Massive randomness.. Is not that random.

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



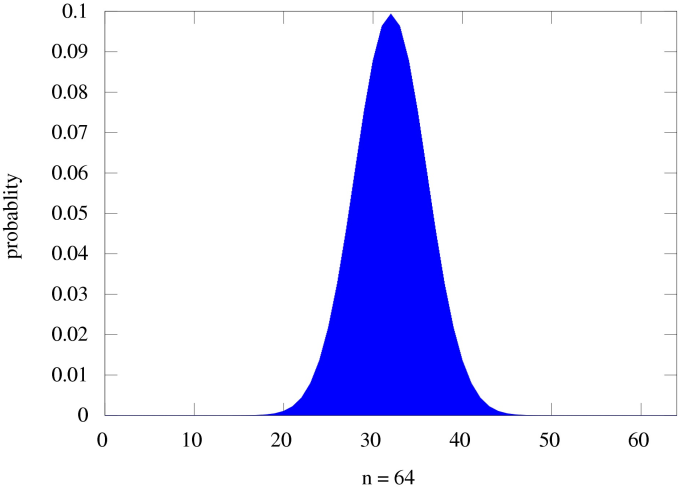
Massive randomness.. Is not that random.

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



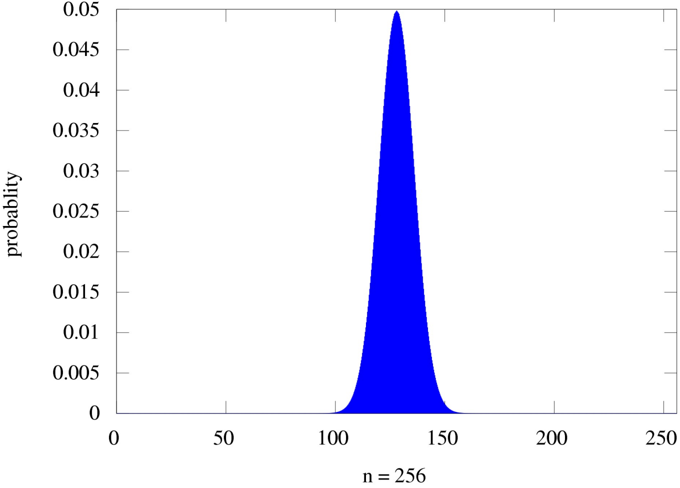
Massive randomness.. Is not that random.

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



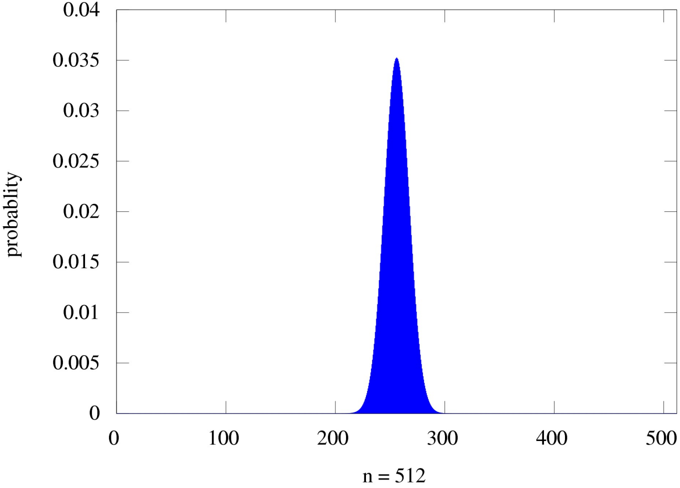
Massive randomness.. Is not that random.

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



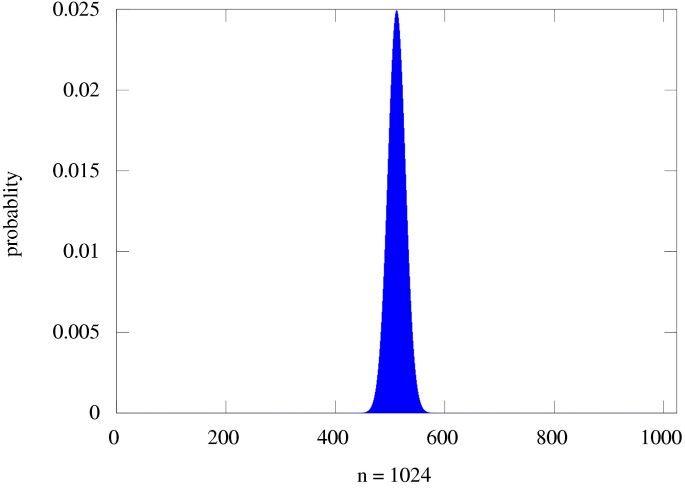
Massive randomness.. Is not that random.

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



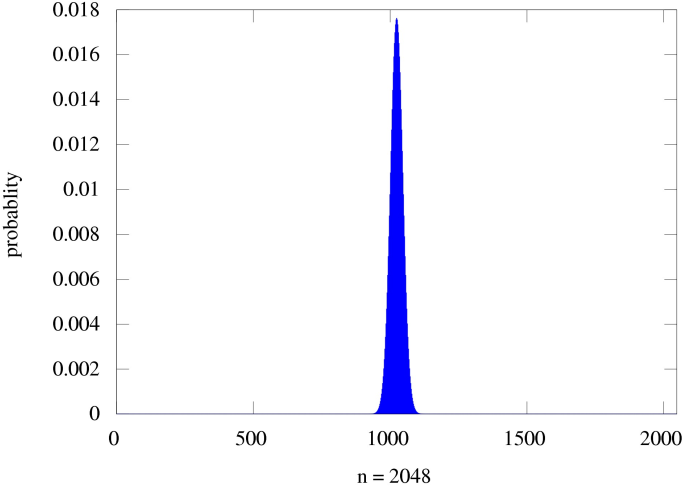
Massive randomness.. Is not that random.

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



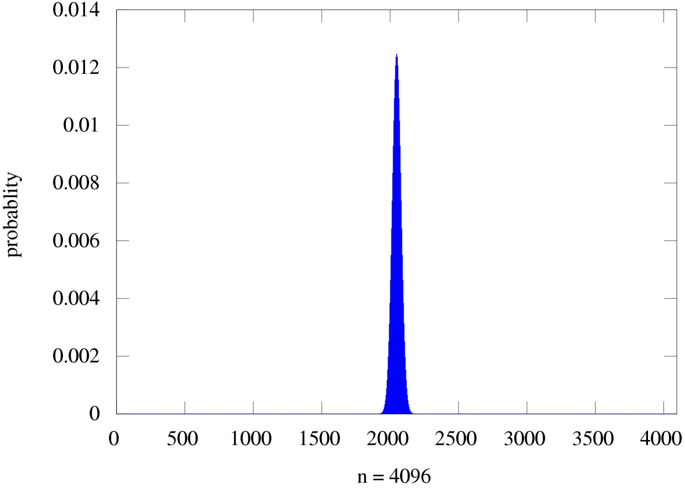
Massive randomness.. Is not that random.

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



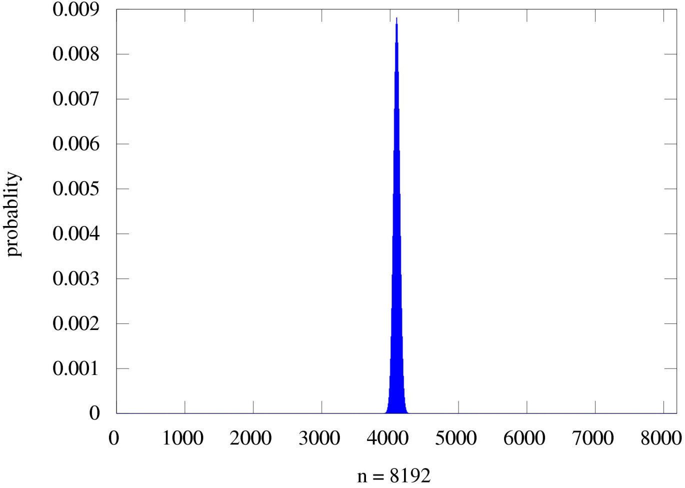
Massive randomness.. Is not that random.

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

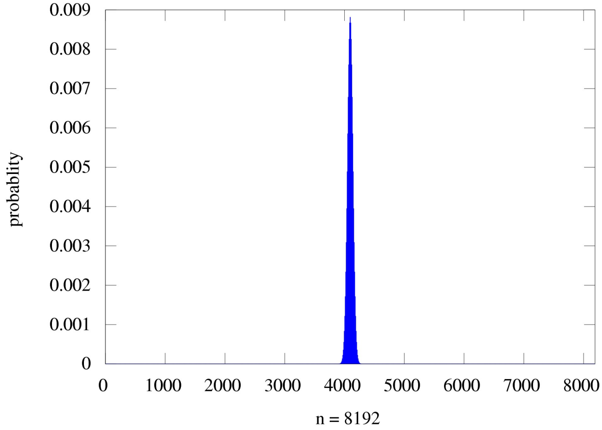


Massive randomness.. Is not that random.

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



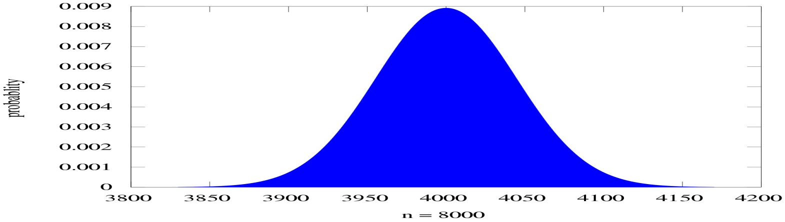
Massive randomness.. Is not that random.



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

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

$\mathbf{X_n} =$ numbers of heads when flipping a coin $\mathbf{n}$ times.

Claim

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

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

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

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

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.

Lemma

$\mathbf{n! \geq (n/e)^n}$.

Proof.

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

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

Lemma

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

Proof.

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

since $\mathbf{k!} \geq \mathbf{(k/e)^k}$ (by previous lemma). □

Binomial distribution

Playing around with binomial coefficients

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

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

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

Binomial distribution

Playing around with binomial coefficients

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

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

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

Flipping a coin.

If you flip an unbiased coin **1000** times, the probability you get at most **250** heads is at most

(A) $\leq 2 \cdot 0.68^{2000} \leq 10^{-336}$.

(B) $\leq 2 \cdot 0.68^{1000} \leq 10^{-168}$.

(C) $\leq 2 \cdot 0.68^{500} \leq 3.593 \cdot 10^{-84}$.

(D) $\leq 2 \cdot 0.68^{250} \leq 3 \cdot 10^{-42}$.

(E) 0.1.

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

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

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

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

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

Analysis via Recurrence

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

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

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

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

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

Analysis via Recurrence

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

We have, for any $\mathbf{A}$:

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

Therefore, by linearity of expectation:

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

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

Analysis via Recurrence

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

We have, for any $\mathbf{A}$:

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

Therefore, by linearity of expectation:

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

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

Analysis via Recurrence

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

We have, for any $\mathbf{A}$:

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

Therefore, by linearity of expectation:

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

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

Analysis via Recurrence

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

We have, for any $\mathbf{A}$:

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

Therefore, by linearity of expectation:

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

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

Analysis via Recurrence

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

We derived:

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

Note that above holds for any $\mathbf{A}$ of size $\mathbf{n}$. Therefore

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

Analysis via Recurrence

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

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

Note that above holds for any $\mathbf{A}$ of size $\mathbf{n}$. Therefore

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

Solving the Recurrence

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

with base case $\mathbf{T(1) = 0}$.

Lemma

$\mathbf{T(n) = O(n \log n)}$.

Proof.

(Guess and) Verify by induction. □

Solving the Recurrence

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

with base case $\mathbf{T(1) = 0}$.

Lemma

$\mathbf{T(n) = O(n \log n)}$.

Proof.

(Guess and) Verify by induction. □

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. □

Notes

Notes

Notes

Notes

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