Introduction to Randomized Algorithms: QuickSort

Lecture 7
Feb 6, 2018

Most slides are courtesy Prof. Chekuri
Randomization is very powerful

How do you play R-P-S?
Randomization is very powerful

How do you play R-P-S?
Calculating insurance.
Outline

Randomization is very powerful

How do you play R-P-S?
Calculating insurance.

Our goal

- Basics of randomization – probability space, expectation, events, random variables, etc.
- Randomized Algorithms – Two types
  - Las Vegas
  - Monte Carlo
- Randomized Quick Sort
Part I

Introduction to Randomized Algorithms
Randomized Algorithms

Input $x$ \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \q
Randomized Algorithms

Deterministic Algorithm

Randomized Algorithm

random bits $r$
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 straightforward approach. $O(n^2 \cdot 37)$ 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?
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 $1/2^{100}$. 
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?

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 $1/2^{100}$. 

Ruta (UIUC)  
CS473  7  
Spring 2018  7 / 53
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? $O(n^2)$!
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 $1/2^{100}$. 
Many many applications in algorithms, data structures and computer science!
Why randomized algorithms?

1. Many many applications in algorithms, data structures and computer science!

2. In some cases only known algorithms are randomized, i.e., *polynomial identity testing*. 
Why randomized algorithms?

1. Many many applications in algorithms, data structures and computer science!
2. In some cases only known algorithms are randomized, i.e., polynomial identity testing.
3. Often randomized algorithms are (much) simpler and/or more efficient.
Why randomized algorithms?

1. Many many applications in algorithms, data structures and computer science!
2. In some cases only known algorithms are randomized, i.e., *polynomial identity testing*.
3. Often randomized algorithms are (much) simpler and/or more efficient.
4. Several deep connections to mathematics, physics etc.
5. . . .
6. Lots of fun!
Average case analysis vs Randomized algorithms

**Average case analysis:**

1. Fix a deterministic algorithm.
2. Assume inputs come from a probability distribution.
3. Analyze the algorithm’s *average* performance over the distribution over inputs.
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. Input is arbitrary (worst case).
2. Algorithm uses random bits, and therefore on each input the behavior of the algorithm is random.
3. Analyze algorithms average performance over any given (worst case) input where the average is over the random bits that the algorithm uses.
Part II

Basics of Discrete Probability
We restrict attention to finite probability spaces.

**Definition**

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

We restrict attention to finite probability spaces.

**Definition**

A discrete probability space is a pair $(\Omega, \Pr)$ consists of finite set $\Omega$ 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 $1 \leq i \leq 6$. 

Events

**Definition**

Given a probability space \((\Omega, \Pr)\) an **event** is a subset of \(\Omega\). In other words an event is a collection of elementary events. The probability of an event \(A \subseteq \Omega\), 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 \(\bar{A}\).
**Definition**

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

The **complement event** of an event \(A \subseteq \Omega\) is the event \(\Omega \setminus A\) frequently denoted by \(\overline{A}\).

**Example**

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

Definition

Given a probability space $(\Omega, \Pr)$ an event is a subset of $\Omega$. In other words an event is a collection of elementary events. The probability of an event $A \subseteq \Omega$, 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 $\bar{A}$.

Example

A pair of independent dice. $\Omega = \{(i, j) \mid 1 \leq i \leq 6, 1 \leq j \leq 6\}$. Event $A$: the sum of the two numbers on the dice is even. Then $A = \{(i, j) \in \Omega \mid (i + j) \text{ is even}\}$. 
Events

Definition

Given a probability space \((\Omega, \Pr)\) an event is a subset of \(\Omega\). In other words an event is a collection of elementary events. The probability of an event \(A \subseteq \Omega\), 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 \(\bar{A}\).

Example

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

Event \(A\): the sum of the two numbers on the dice is even.

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

\(\Pr[A] = |A|/36 = 1/2\).
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 \(\Pr[HH] = \Pr[TT] = \Pr[HT] = \Pr[TH] = \frac{1}{4}\).

1. \(A\): the first coin is heads.
   \(B\): second coin is tails.
   \(\Pr[A] = \frac{1}{2}, \Pr[B] = \frac{1}{2}, \Pr[A \cap B] = \frac{1}{4}\). Independent.

2. \(A\): both are not tails.
   \(B\): second coin is heads.
   \(\Pr[A] = \frac{3}{4}, \Pr[B] = \frac{1}{2}, \Pr[A \cap B] = \frac{1}{2}\). Dependent.
Independent Events

Definition

Given a probability space \((\Omega, \Pr)\) and two events \(A, B\) are **independent** if and only if \(\Pr[A \cap B] = \Pr[A] \cdot \Pr[B]\). 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 

\[
\Pr[HH] = \Pr[TT] = \Pr[HT] = \Pr[TH] = \frac{1}{4}.
\]

1. \(A\) : the first coin is heads.  
   \(B\) : second coin is tails.
Independent Events

Definition

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

Example

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

\(A\) : the first coin is heads. \(B\) : second coin is tails.
\[
\Pr[A] =
\]
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
\[
\Pr[HH] = \Pr[TT] = \Pr[HT] = \Pr[TH] = \frac{1}{4}.
\]

1. \(A\) : the first coin is heads. \(B\) : second coin is tails.
\[
\Pr[A] = \frac{1}{2}, \Pr[B] = \frac{1}{2}, \Pr[A \cap B] = \]
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. \(A\) : the first coin is heads. \(B\) : second coin is tails.
   \(\Pr[A] = 1/2, \Pr[B] = 1/2, \Pr[A \cap B] = 1/4\). 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. \(A\) : the first coin is heads. \(B\) : second coin is tails. 
   \(\Pr[A] = 1/2, \Pr[B] = 1/2, \Pr[A \cap B] = 1/4\). independent.

2. \(A\) : both are not tails. \(B\) : second coin is heads. 
   \(\Pr[A] = \)
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. \(A\) : the first coin is heads. \(B\) : second coin is tails.
   \(\Pr[A] = 1/2, \Pr[B] = 1/2, \Pr[A \cap B] = 1/4\). independent.

2. \(A\) : both are not tails. \(B\) : second coin is heads.
   \(\Pr[A] = 3/4, \Pr[B] = 1/2, \Pr[A \cap B] = \)
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**: the first coin is heads. **B**: second coin is tails.
   \(\Pr[A] = 1/2, \ Pr[B] = 1/2, \ Pr[A \cap B] = 1/4\). independent.

2. **A**: both are not tails. **B**: second coin is heads.
   \(\Pr[A] = 3/4, \ Pr[B] = 1/2, \ Pr[A \cap B] = 1/2\). dependent.
Union bound

The probability of the union of two events, is no bigger than 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

$$\Pr[\mathcal{E} \cup \mathcal{F}] = \sum_{x \in \mathcal{E} \cup \mathcal{F}} \Pr[x]$$

$$\leq \sum_{x \in \mathcal{E}} \Pr[x] + \sum_{x \in \mathcal{F}} \Pr[x] = \Pr[\mathcal{E}] + \Pr[\mathcal{F}] .$$
Definition (Random Variable)

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

**Definition (Random Variable)**

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

**Definition (Expectation)**

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 \(\Pr[\cdot]\).
A 6-sided unbiased die. \( \Omega = \{1, 2, 3, 4, 5, 6\} \) and \( \text{Pr}[i] = 1/6 \) for each \( i \in \Omega \).

1. \( X : \Omega \to \mathbb{R} \) where \( X(i) = i \mod 2 \in \{0, 1\} \).
A 6-sided unbiased die. \( \Omega = \{1, 2, 3, 4, 5, 6\} \) and \( \Pr[i] = 1/6 \) for each \( i \in \Omega \).

\[ X : \Omega \to \mathbb{R} \text{ where } X(i) = i \mod 2 \in \{0, 1\}. \text{ Then } \]
\[ E[X] = \sum_{i=1}^{6} \Pr[i] \cdot X(i) = \frac{1}{6} \sum_{i=1}^{6} X(i) = 1/2. \]
A 6-sided unbiased die. \( \Omega = \{1, 2, 3, 4, 5, 6\} \) and \( \Pr[i] = 1/6 \) for each \( i \in \Omega \).

1. \( X : \Omega \rightarrow \mathbb{R} \) where \( X(i) = i \mod 2 \in \{0, 1\} \). Then
   \[
   E[X] = \sum_{i=1}^{6} Pr[i] \cdot X(i) = \frac{1}{6} \sum_{i=1}^{6} X(i) = 1/2.
   \]

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

Example

A 6-sided unbiased die. \( \Omega = \{1, 2, 3, 4, 5, 6\} \) and \( \Pr[i] = \frac{1}{6} \) for each \( i \in \Omega \).

1. \( X : \Omega \to \mathbb{R} \) where \( X(i) = i \mod 2 \in \{0, 1\} \). Then
   \[
   E[X] = \sum_{i=1}^{6} \Pr[i] \cdot X(i) = \frac{1}{6} \sum_{i=1}^{6} X(i) = \frac{1}{2}.
   \]

2. \( Y : \Omega \to \mathbb{R} \) where \( Y(i) = i^2 \). Then
   \[
   E[Y] = \sum_{i=1}^{6} \frac{1}{6} \cdot i^2 = \frac{91}{6}.
   \]
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 $1/2$. Compute the expected number of vertices in $H$.

(A) $n/2$.
(B) $n/4$.
(C) $m/2$.
(D) $m/4$.
(E) none of the above.
Expected number of vertices is:

**Probability Space**

- $\Omega = \{0, 1\}^n$. For $\omega \in \{0, 1\}^n$, $\omega_v = 1$ if vertex $v$ is present in $H$, else is zero.
- For each $\omega \in \Omega$, $\Pr[\omega] = \frac{1}{2^n}$. 

$X(\omega) = \# \text{ vertices in } H \text{ as per } \omega = \# 1s \text{ in } \omega$.

$E[X] = \sum_{\omega \in \Omega} \Pr[\omega] X(\omega) = \sum_{\omega \in \Omega} \frac{1}{2^n} X(\omega) = \frac{1}{2^n} \sum_{k=0}^{n} \binom{n}{k} k = \frac{1}{2^n} (2^n - n - 1) = \frac{n}{2^n}$.
Expected number of vertices is:

**Probability Space**

- $\Omega = \{0, 1\}^n$. For $\omega \in \{0, 1\}^n$, $\omega_v = 1$ if vertex $v$ is present in $H$, else is zero.
- For each $\omega \in \Omega$, $\Pr[\omega] = \frac{1}{2^n}$.
- $X(\omega) = \#$ vertices in $H$ as per $\omega = \# 1$s in $\omega$. 

$$E[X] = \sum_{\omega \in \Omega} \Pr[\omega] X(\omega) = \sum_{\omega \in \Omega} \frac{1}{2^n} X(\omega) = \frac{1}{2^n} \sum_{k=0}^{n} \binom{n}{k} k = \frac{1}{2^n} (2^n n 2^{n-1}) = \frac{n}{2}$$
Expected number of vertices is:

**Probability Space**
- \( \Omega = \{0, 1\}^n \). For \( \omega \in \{0, 1\}^n \), \( \omega_v = 1 \) if vertex \( v \) is present in \( H \), else is zero.
- For each \( \omega \in \Omega \), \( \Pr[\omega] = \frac{1}{2^n} \).
- \( X(\omega) = \# \) vertices in \( H \) as per \( \omega = \# 1s \) in \( \omega \).

\[
E[X] = \sum_{\omega \in \Omega} \Pr[\omega] X(\omega) \\
= \sum_{\omega \in \Omega} \frac{1}{2^n} X(\omega) \\
= \frac{1}{2^n} \sum_{k=0}^{n} \binom{n}{k} k \\
= \frac{1}{2^n} \left( 2^n \frac{n}{2} \right) \\
= \frac{n}{2}
\]
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 edges is:

**Probability Space**

- $\Omega = \{0, 1\}^n$. For $\omega \in \{0, 1\}^n$, $\omega_v = 1$ if vertex $v$ is present in $H$, else is zero.
- For each $\omega \in \Omega$, $\Pr[\omega] = \frac{1}{2^n}$. 

How to compute $E[X]$?
Expected number of edges is:

**Probability Space**

- $\Omega = \{0, 1\}^n$. For $\omega \in \{0, 1\}^n$, $\omega_v = 1$ if vertex $v$ is present in $H$, else is zero.

- For each $\omega \in \Omega$, $\Pr[\omega] = \frac{1}{2^n}$.

- $X(\omega) = \#$ edges present in $H$ as per $\omega = ??$
**Expected number of edges is:**

<table>
<thead>
<tr>
<th>Probability Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ω = {0, 1}^n. For ω ∈ {0, 1}^n, ω_v = 1 if vertex v is present in H, else is zero.</td>
</tr>
<tr>
<td>For each ω ∈ Ω, Pr[ω] = (\frac{1}{2^n}).</td>
</tr>
<tr>
<td>X(ω) = # edges present in H as per ω = ??</td>
</tr>
</tbody>
</table>

How to compute \(E[X]\)?
A binary random variable is one that takes on values in \( \{0, 1\} \).
Indicator Random Variables

**Definition**

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

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 \(\omega \not\in A\).
Definition

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

Special type of random variables that are quite useful.

Definition

Given a probability space \((\Omega, \text{Pr})\) 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 \(\omega \not\in A\).

Example

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

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

**Proof.**

\[
E[X_A] = \sum_{\omega \in \Omega} X_A(\omega) \Pr[\omega]
\]

\[
= \sum_{\omega \in A} 1 \cdot \Pr[\omega] + \sum_{\omega \in \Omega \setminus A} 0 \cdot \Pr[\omega]
\]

\[
= \sum_{\omega \in A} \Pr[\omega]
\]

\[
= \Pr[A].
\]
**Lemma**

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

**Proof.**

\[
E[X + Y] = \sum_{\omega \in \Omega} \Pr[\omega] (X(\omega) + Y(\omega))
\]

\[
= \sum_{\omega \in \Omega} \Pr[\omega] X(\omega) + \sum_{\omega \in \Omega} \Pr[\omega] Y(\omega) = E[X] + E[Y].
\]
Lemma

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

Proof.

$$E[X + Y] = \sum_{\omega \in \Omega} \text{Pr}[\omega] (X(\omega) + Y(\omega))$$

$$= \sum_{\omega \in \Omega} \text{Pr}[\omega] X(\omega) + \sum_{\omega \in \Omega} \text{Pr}[\omega] Y(\omega) = E[X] + E[Y].$$

Corollary

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

- Event $A_e = \text{edge } e \in E \text{ is present in } H$.
- $\Pr[A_{e=(u,v)}] = \Pr[u \text{ and } v \text{ both are present}] = \Pr[u \text{ is present}] \cdot \Pr[v \text{ is present}] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$. 
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

- Event $A_e = \text{edge } e \in E$ is present in $H$.
- $\Pr[A_{e=(u,v)}] = \Pr[u \text{ and } v \text{ both are present}] = \Pr[u \text{ is present}] \cdot \Pr[v \text{ is present}] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$.
- $X_{A_e}$ indicator random variables, then $\mathbb{E}[X_{A_e}] = \Pr[A_e]$. 

It is important to setup random variables carefully.
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

- Event $A_e = \text{edge } e \in E$ is present in $H$.
- $\Pr[A_e=(u,v)] = \Pr[u \text{ and } v \text{ both are present}] = \Pr[u \text{ is present}] \cdot \Pr[v \text{ is present}] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$.
- $X_{A_e}$ indicator random variables, then $E[X_{A_e}] = \Pr[A_e]$.
- Let $X = \sum_{e \in E} X_{A_e}$ (Number of edges in $H$)
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

- Event $A_e = \text{edge } e \in E \text{ is present in } H$.
- $\Pr[A_{e=(u,v)}] = \Pr[u \text{ and } v \text{ both are present}] = \Pr[u \text{ is present}] \cdot \Pr[v \text{ is present}] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$.
- $X_{A_e}$ indicator random variables, then $E[X_{A_e}] = \Pr[A_e]$.
- Let $X = \sum_{e \in E} X_{A_e}$ (Number of edges in $H$)

$$E[X] = E\left[ \sum_{e \in E} X_{A_e} \right] = \sum_{e \in E} E[X_{A_e}] = \sum_{e \in E} \Pr[A_e] = \frac{m}{4}$$
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

- Event $A_e = \text{edge } e \in E$ is present in $H$.
- $\Pr[A_{e=(u,v)}] = \Pr[u \text{ and } v \text{ both are present}] = \Pr[u \text{ is present}] \cdot \Pr[v \text{ is present}] = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$.
- $X_{A_e}$ indicator random variables, then $E[X_{A_e}] = \Pr[A_e]$.
- Let $X = \sum_{e \in E} X_{A_e}$ (Number of edges in $H$)

$$E[X] = E \left[ \sum_{e \in E} X_{A_e} \right] = \sum_{e \in E} E[X_{A_e}] = \sum_{e \in E} \Pr[A_e] = \frac{m}{4}$$

It is important to setup random variables carefully.
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.
Independent Random Variables

**Definition**
Random variables $X, Y$ are said to be independent if

$$\forall x, y \in \mathbb{R}, \quad \Pr[X = x \land Y = y] = \Pr[X = x] \Pr[Y = y]$$
Independent Random Variables

**Definition**

Random variables $X, Y$ are said to be independent if

$$\forall x, y \in \mathbb{R}, \quad \Pr[X = x \land Y = y] = \Pr[X = x] \Pr[Y = y].$$

**Examples**

Two independent un-biased coin flips: $\Omega = \{HH, HT, TH, TT\}$.

- $X = 1$ if first coin is $H$ else $0$. $Y = 1$ if second coin is $H$ else $0$. 
Independent Random Variables

Definition

Random variables \( X, Y \) are said to be independent if

\[
\forall x, y \in \mathbb{R}, \quad \Pr[X = x \land Y = y] = \Pr[X = x] \Pr[Y = y]
\]

Examples

Two independent un-biased coin flips: \( \Omega = \{HH, HT, TH, TT\} \).

- \( X = 1 \) if first coin is \( H \) else \( 0 \).
- \( Y = 1 \) if second coin is \( H \) else \( 0 \). Independent.
Independent Random Variables

Definition
Random variables $X, Y$ are said to be independent if

$$\forall x, y \in \mathbb{R}, \quad \Pr[X = x \land Y = y] = \Pr[X = x] \Pr[Y = y]$$

Examples
Two independent un-biased coin flips: $\Omega = \{HH, HT, TH, TT\}$.

- $X = 1$ if first coin is $H$ else $0$. $Y = 1$ if second coin is $H$ else $0$. Independent.
- $X = \#H, Y = \#T$. 
Independent Random Variables

**Definition**
Random variables $X$, $Y$ are said to be independent if

$$\forall x, y \in \mathbb{R}, \quad \Pr[X = x \land Y = y] = \Pr[X = x] \Pr[Y = y]$$

**Examples**
Two independent un-biased coin flips: $\Omega = \{HH, HT, TH, TT\}$.
- $X = 1$ if first coin is $H$ else $0$. $Y = 1$ if second coin is $H$ else $0$. Independent.
- $X = \#H, Y = \#T$. Dependent. Why?
Lemma

If $X$ and $Y$ are independent then $E[X \cdot Y] = E[X] \cdot E[Y]$

Proof.

$$E[X \cdot Y] = \sum_{\omega \in \Omega} \Pr[\omega] (X(\omega) \cdot Y(\omega))$$

$$= \sum_{x, y \in \mathbb{R}} \Pr[X = x \land Y = y] (x \cdot y)$$

$$= \sum_{x, y \in \mathbb{R}} \Pr[X = x] \cdot \Pr[Y = y] \cdot x \cdot y$$

$$= \left( \sum_{x \in \mathbb{R}} \Pr[X = x] x \right) \left( \sum_{y \in \mathbb{R}} \Pr[Y = y] y \right) = E[X] E[Y]$$
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.
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.
**Deterministic** algorithm $Q$ for a problem $\Pi$:

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 $n$.

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

**Deterministic** algorithm \( Q \) for a problem \( \Pi \):

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 \( \Pi \):

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 \( x \).
Analyzing Las Vegas Algorithms

Deterministic algorithm $Q$ for a problem $\Pi$:
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 $\Pi$:
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 $n$.

$$T_{rand-wc}(n) = \max_{x:|x|=n} E[R(x)].$$
Analyzing Monte Carlo Algorithms

Randomized algorithm $M$ for a problem $\Pi$:

1. 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.

2. Let $Pr[x]$ be the probability that $M$ is correct on $x$.

3. $Pr[x]$ is a random variable: depends on random bits used by $M$. 
Randomized algorithm $M$ for a problem $\Pi$:

1. 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.

2. Let $\Pr[x]$ be the probability that $M$ is correct on $x$.

3. $\Pr[x]$ is a random variable: depends on random bits used by $M$.

4. Worst-case analysis: success probability on worst input

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

Why does randomization help?
Consider a deterministic algorithm $A$ that is trying to find an element in an array $X$ of size $n$. 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) $O(1)$
(B) $O(n)$
(C) $O(n \log n)$
(D) $O(n^2)$
(E) $\infty$. 

Consider an algorithm **randFind** that is trying to find an element in an array $X$ of size $n$. 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) $O(1)$
- (B) $O(\log n)$
- (C) $O(n)$
- (D) $O(n^2)$
- (E) $\infty$.

steps.
Consider the problem of finding an “approximate median” of an unsorted array $A[1..n]$: an element of $A$ with rank between $n/4$ and $3n/4$.

- Finding an approximate median is not any easier than a proper median.
Abundance of witnesses

Consider the problem of finding an “approximate median” of an unsorted array $A[1..n]$: an element of $A$ with rank between $n/4$ and $3n/4$.

- Finding an approximate median is not any easier than a proper median.
- $n/2$ elements of $A$ qualify as approximate medians and hence a random element is good with probability $1/2!$
Part IV

Randomized Quick Sort
QuickSort

### Deterministic QuickSort

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.
QuickSort

Deterministic QuickSort

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.
Randomized Quicksort

Recall: Deterministic 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.
Randomized Quicksort

Recall: Deterministic 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.
Randomized Quicksort

Recall: Deterministic 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.
**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.
What events to count?
- Number of Comparisions.
Analysis

What events to count?
- Number of Comparisons.

What is the probability space?
- All the coin tosses at all levels and parts of recursion.
What events to count?
- Number of Comparisons.

What is the probability space?
- All the coin tosses at all levels and parts of recursion.

Too Big!!
Analysis

What events to count?
- Number of Comparisons.

What is the probability space?
- All the coin tosses at all levels and parts of recursion.

Too Big!!

What random variables to define?
What are the events of the algorithm?
1. Given array $A$ of $n$ distinct numbers.

2. $Q(A)$: number of comparisons of randomized QuickSort on $A$. Note that $Q(A)$ is a random variable.
Given array \( A \) of \( n \) distinct numbers.

\( Q(A) \) : number of comparisons of randomized QuickSort on \( A \). Note that \( Q(A) \) is a random variable.

\( X_i \) : Indicator random variable, which is set to 1 if pivot is of rank \( i \) in \( A \), else zero.

Let \( A^i_{\text{left}} \) and \( A^i_{\text{right}} \) be the corresponding left and right subarrays.
Given array $A$ of $n$ distinct numbers.

$Q(A)$: number of comparisons of randomized QuickSort on $A$. Note that $Q(A)$ is a random variable.

$X_i$: Indicator random variable, which is set to 1 if pivot is of rank $i$ in $A$, else zero.

Let $A_{\text{left}}^i$ and $A_{\text{right}}^i$ be the corresponding left and right subarrays.

$$Q(A) = n + \sum_{i=1}^{n} X_i \cdot \left( Q(A_{\text{left}}^i) + Q(A_{\text{right}}^i) \right).$$
Analysis via Recurrence

1. Given array $A$ of $n$ distinct numbers.

2. $Q(A)$: number of comparisons of randomized QuickSort on $A$. Note that $Q(A)$ is a random variable.

3. $X_i$: Indicator random variable, which is set to 1 if pivot is of rank $i$ in $A$, else zero.

Let $A_{\text{left}}^i$ and $A_{\text{right}}^i$ be the corresponding left and right subarrays.

$$Q(A) = n + \sum_{i=1}^{n} X_i \cdot \left( Q(A_{\text{left}}^i) + Q(A_{\text{right}}^i) \right).$$

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

$$E[X_i] = \Pr[\text{pivot has rank } i] = 1/n.$$
Independence of Random Variables

### Lemma

Random variables $X_i$ is independent of random variables $Q(A_{i \text{ left}}^i)$ as well as $Q(A_{i \text{ right}}^i)$, i.e.

$$E[X_i \cdot Q(A_{i \text{ left}}^i)] = E[X_i] E[Q(A_{i \text{ left}}^i)]$$

$$E[X_i \cdot Q(A_{i \text{ right}}^i)] = E[X_i] E[Q(A_{i \text{ right}}^i)]$$

### Proof.

This is because the algorithm, while recursing on $Q(A_{i \text{ left}}^i)$ and $Q(A_{i \text{ right}}^i)$ uses new random coin tosses that are independent of the coin tosses used to decide the first pivot. Only the latter decides value of $X_i$. 

□
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 $n$. 
Analysis via Recurrence

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

We have, for any \( A \):

\[
Q(A) = n + \sum_{i=1}^{n} X_i \left( Q(A_{\text{left}}^i) + Q(A_{\text{right}}^i) \right)
\]
Let $T(n) = \max_{A:|A|=n} \mathbb{E}[Q(A)]$ be the worst-case expected running time of randomized QuickSort on arrays of size $n$.

We have, for any $A$:

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

By linearity of expectation, and independence random variables:

$$\mathbb{E}[Q(A)] = n + \sum_{i=1}^{n} \mathbb{E}[X_i] \left( \mathbb{E}[Q(A^i_{\text{left}})] + \mathbb{E}[Q(A^i_{\text{right}})] \right).$$
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 have, for any \( A \):

\[
Q(A) = n + \sum_{i=1}^{n} X_i \left( Q(A^i_{\text{left}}) + Q(A^i_{\text{right}}) \right)
\]

By linearity of expectation, and independence random variables:

\[
E[Q(A)] = n + \sum_{i=1}^{n} E[X_i] \left( E\left[ Q(A^i_{\text{left}}) \right] + E\left[ Q(A^i_{\text{right}}) \right] \right).
\]

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

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

\[
\mathbb{E}[Q(A)] \leq n + \sum_{i=1}^{n} \frac{1}{n} (T(i - 1) + T(n - i)).
\]
Let $T(n) = \max_{A:|A|=n} \mathbb{E}[Q(A)]$ be the worst-case expected running time of randomized QuickSort on arrays of size $n$. We derived:

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

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

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

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

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

\[ T(n) = O(n \log n) \]
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.
Part V

Slick analysis of QuickSort
A Slick Analysis of QuickSort

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

1. For $1 \leq i < j < n$ let $R_{ij}$ be the event that rank $i$ element is compared with rank $j$ element.
A Slick Analysis of **QuickSort**

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

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

2. $X_{ij}$ is the indicator random variable for $R_{ij}$. That is, $X_{ij} = 1$ if rank $i$ is compared with rank $j$ element, otherwise $0$. 

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

and hence by linearity of expectation,

$$E[Q(A)] = \sum_{1 \leq i < j \leq n} E[X_{ij}] = \sum_{1 \leq i < j \leq n} Pr[R_{ij}]$$
A Slick Analysis of **QuickSort**

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

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

2. $X_{ij}$ is the indicator random variable for $R_{ij}$. That is, $X_{ij} = 1$ if rank $i$ is compared with rank $j$ element, otherwise 0.

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

and hence by linearity of expectation,

$$E\left[ Q(A) \right] = \sum_{1 \leq i < j \leq n} E\left[ X_{ij} \right] = \sum_{1 \leq i < j \leq n} \Pr\left[ R_{ij} \right].$$
A Slick Analysis of QuickSort

\[ R_{ij} = \text{rank } i \text{ element is compared with rank } j \text{ element.} \]

**Question:** What is \( \Pr[R_{ij}] \)?
A Slick Analysis of QuickSort

\[ R_{ij} = \text{rank } i \text{ element is compared with rank } j \text{ element.} \]

**Question:** What is \( Pr[R_{ij}] \)?

<table>
<thead>
<tr>
<th>7</th>
<th>5</th>
<th>9</th>
<th>1</th>
<th>3</th>
<th>4</th>
<th>8</th>
<th>6</th>
</tr>
</thead>
</table>

With ranks: 6 4 8 1 2 3 7 5
$R_{ij} = \text{rank } i \text{ element is compared with rank } j \text{ element.}$

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

With ranks: 6 4 8 1 2 3 7 5

As such, probability of comparing 5 to 8 is $\text{Pr}[R_{4,7}]$. 
A Slick Analysis of QuickSort

\[ R_{ij} = \text{rank } i \text{ element is compared with rank } j \text{ element.} \]

**Question:** What is \( \text{Pr}[R_{ij}] \)?

With ranks: 6 4 8 1 2 3 7 5

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

Decision if to compare 5 to 8 is moved to subproblem.
A Slick Analysis of QuickSort

\[ R_{ij} = \text{rank } i \text{ element is compared with rank } j \text{ element.} \]

**Question:** What is \( \Pr[R_{ij}] \)?

With ranks: 6 4 8 1 2 3 7 5

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

\[
\begin{array}{cccccccc}
7 & 5 & 9 & 1 & 3 & 4 & 8 & 6 \\
1 & 3 & 7 & 5 & 9 & 4 & 8 & 6
\end{array}
\]

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

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

\[
\begin{array}{cccccccc}
7 & 5 & 9 & 1 & 3 & 4 & 8 & 6 \\
7 & 5 & 1 & 3 & 4 & 8 & 6 & 9
\end{array}
\]

Decision if to compare 5 to 8 moved to subproblem.
A Slick Analysis of QuickSort

**Question:** What is \( \Pr[R_{i,j}] \)?

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

If pivot is 5 (rank 4). Bingo!

\[
\begin{array}{cccccccc}
7 & 5 & 9 & 1 & 3 & 4 & 8 & 6 \\
6 & 4 & 8 & 1 & 2 & 3 & 7 & 5
\end{array}
\]

\[
\begin{array}{cccccccc}
1 & 3 & 4 & 5 & 7 & 9 & 8 & 6
\end{array}
\]
A Slick Analysis of QuickSort

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

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

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

2. If pivot is 8 (rank 7). Bingo!
A Slick Analysis of QuickSort

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

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

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

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

3. If pivot in between the two numbers (say 6 [rank 5]): 5 and 8 will never be compared to each other.
A Slick Analysis of QuickSort

**Question:** What is \( \text{Pr}[R_{i,j}] \)?

**Conclusion:**

\( R_{i,j} \) happens if and only if:

- \( i \)th or \( j \)th ranked element is the first pivot out of
- \( i \)th to \( j \)th ranked elements.

\[
\text{Pr}[R_{i,j}] = \text{Pr}[i\text{th or } j\text{th ranked element is the pivot } | \text{ pivot has rank in } \{i, i+1\ldots, j-1, j\}]
\]
**Conclusion:**

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

- $i$th or $j$th ranked element is the first pivot out of $i$th to $j$th ranked elements.

Therefore,

$$Pr[R_{i,j}] = Pr[i\text{th or } j\text{th ranked element is the pivot } | \text{ pivot has rank in } \{i, i+1 \ldots, j-1, j\}]$$

There are $k = j - i + 1$ relevant elements.

$$Pr\left[R_{i,j}\right] = \frac{2}{k} = \frac{2}{j - i + 1}.$$
A Slick Analysis of QuickSort

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

**Lemma**

$$\Pr[R_{ij}] = \frac{2}{j-i+1}.$$
Question: What is $\Pr[R_{ij}]$?

Lemma

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

Proof.

Let $a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n$ be elements of $A$ in sorted order. Let $S = \{a_i, a_{i+1}, \ldots, a_j\}$.
Question: What is $\Pr[R_{ij}]$?

Lemma

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

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.
**A Slick Analysis of QuickSort**

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

**Lemma**

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

**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.
A Slick Analysis of QuickSort

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

**Lemma**

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

**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...
Lemma

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

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.
A Slick Analysis of QuickSort

Continued...

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

**Lemma**

\[ \Pr[R_{ij}] = \frac{2}{j-i+1}. \]
A Slick Analysis of **QuickSort**

Continued...

**Lemma**

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

\[
\mathbb{E}[Q(A)] = \sum_{1 \leq i < j \leq n} \Pr[R_{ij}] = \sum_{1 \leq i < j \leq n} \frac{2}{j-i+1}
\]
Lemma

\[ \text{Pr}[R_{ij}] = \frac{2}{j-i+1}. \]

\[ \text{E}[Q(A)] = \sum_{1 \leq i < j \leq n} \frac{2}{j - i + 1} \]
A Slick Analysis of QuickSort

Continued...

Lemma

\[ \Pr[R_{ij}] = \frac{2}{j-i+1} \cdot \]

\[ \mathbb{E}[Q(A)] = \sum_{1 \leq i < j \leq n} \frac{2}{j-i+1} \]

\[ = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \]
A Slick Analysis of QuickSort

Continued...

**Lemma**

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

\[ \mathbb{E}[Q(A)] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \]
A Slick Analysis of **QuickSort**

Continued...

**Lemma**

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

\[ \mathbb{E}[Q(A)] = 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j - i + 1} \]
Lemma

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

\[ E[Q(A)] = 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j - i + 1} = 2 \sum_{i=1}^{n-1} \sum_{\Delta=2}^{n-i+1} \frac{1}{\Delta} \]
Lemma

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

\[
E[Q(A)] = 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j - i + 1} = 2 \sum_{i=1}^{n-1} \sum_{\Delta=2}^{n-i+1} \frac{1}{\Delta}
\leq 2 \sum_{i=1}^{n-1} (H_{n-i+1} - 1) \leq 2 \sum_{1 \leq i < n} H_n
\]

\[ H_k = \sum_{i=1}^{k} \frac{1}{i} = \Theta(\log k) \]
Lemma

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

\[
\mathbb{E}[Q(A)] = 2 \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j-i+1} = 2 \sum_{i=1}^{n-1} \sum_{\Delta=2}^{n-i+1} \frac{1}{\Delta} \\
\leq 2 \sum_{i=1}^{n-1} (H_{n-i+1} - 1) \leq 2 \sum_{1 \leq i < n} H_n \\
\leq 2nH_n = O(n \log n)
\]

\[
H_k = \sum_{i=1}^{k} \frac{1}{i} = \Theta(\log k)
\]
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.