Randomized Algorithms

Lecture 9
September 24, 2013
Part I

Randomized Algorithms
Definition (Informal)

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

Definition

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

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

Equivalent to

$$
\Pr[(X = x) \cap (Y = y)] = \Pr[X = x \, | \, Y = y] \ast \Pr[Y = y].
$$
**Definition (Informal)**

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

**Definition**

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

$$
Pr\left[ X = x \big| Y = y \right] = \frac{Pr\left[ (X = x) \cap (Y = y) \right]}{Pr\left[ Y = y \right]}.
$$

Equivalent to

$$
Pr\left[ (X = x) \cap (Y = y) \right] = Pr\left[ X = x \big| Y = y \right] \ast Pr\left[ Y = y \right].
$$
Definition (Informal)

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

Definition

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

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

Equivalent to

$$
\Pr[(X = x) \cap (Y = y)] = \Pr[X = x \mid Y = y] \times \Pr[Y = y].
$$
Definition

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

$$
\Pr[X = x \cap Y = y] = \Pr[X = x] \cdot \Pr[Y = y].
$$

Definition

The **expectation** of a random variable $X$ is its average value:

$$
E[X] = \sum_x x \cdot \Pr[X = x],
$$
The events $X = x$ and $Y = y$ are independent, if

$$\Pr[X = x \cap Y = y] = \Pr[X = x] \cdot \Pr[Y = y].$$

$$\equiv \Pr[X = x \mid Y = y] = \Pr[X = x].$$

The expectation of a random variable $X$ its average value:

$$E[X] = \sum_x x \cdot \Pr[X = x],$$
Lemma (Linearity of expectation.)

\[ \forall \text{ random variables } X \text{ and } Y: \ E[X + Y] = E[X] + E[Y]. \]

Proof.

Use definitions, do the math. See notes for details.
Definition

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

$$E[X | Y] = E[X | Y = y] = \sum_x x \cdot \Pr[X = x | Y = y].$$

$E[X]$ is a number.

$f(y) = E[X | Y = y]$ is a function.
Lemma

\( \forall X, Y \) (not necessarily independent): \( E[X] = E[E[X \mid Y]] \).

\[
E[E[X \mid Y]] = E_y[E[X \mid Y = y]]
\]

Proof.

Use definitions, and do the math. See class notes.
Conditional Expectation

**Lemma**

∀ X, Y (not necessarily independent): \( E[X] = E[E[X \mid Y]] \).

\[
E[E[X \mid Y]] = E_y[E[X \mid Y = y]]
\]

**Proof.**

Use definitions, and do the math. See class notes.
Conditional Expectation

**Lemma**

∀ X, Y (not necessarily independent): \( E[X] = E[E[X | Y]] \).

\[
E[E[X | Y]] = E_y[E[X | Y = y]]
\]

**Proof.**

Use definitions, and do the math. See class notes.
Problem (Sorting Nuts and Bolts)

1. **Input:** Set $n$ nuts + $n$ bolts.
2. Every nut have a matching bolt.
3. All diff sizes.
4. **Task:** Match nuts to bolts. (In sorted order).
5. **Restriction:** You can only compare a nut to a bolt.
6. **Q:** How to match the $n$ nuts to the $n$ bolts quickly?
Problem (Sorting Nuts and Bolts)

1. Input: Set $n$ nuts + $n$ bolts.
2. Every nut have a matching bolt.
3. All diff sizes.
4. Task: Match nuts to bolts.
   (In sorted order).
5. Restriction: You can only compare a nut to a bolt.
6. Q: How to match the $n$ nuts to the $n$ bolts quickly?
Problem (Sorting Nuts and Bolts)

1. **Input:** Set $n$ nuts + $n$ bolts.

2. Every nut have a matching bolt.

3. All diff sizes.

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

5. Restriction: You can only compare a nut to a bolt.

6. Q: How to match the $n$ nuts to the $n$ bolts quickly?
Problem (**Sorting Nuts and Bolts**)

1. **Input:** Set $n$ nuts + $n$ bolts.
2. Every nut have a matching bolt.
3. All diff sizes.
4. **Task:** Match nuts to bolts. (In sorted order).
5. **Restriction:** You can only compare a nut to a bolt.
6. **Q:** How to match the $n$ nuts to the $n$ bolts quickly?
Problem (Sorting Nuts and Bolts)

1. Input: Set $n$ nuts + $n$ bolts.
2. Every nut have a matching bolt.
3. All diff sizes.
4. Task: Match nuts to bolts. (In sorted order).
5. Restriction: You can only compare a nut to a bolt.
6. Q: How to match the $n$ nuts to the $n$ bolts quickly?
Sorting nuts & bolts...

Algorithm

1. Naive algorithm...
2. ...better algorithm?
Sorting nuts & bolts...

Algorithm

1. Naive algorithm...
2. ...better algorithm?
MatchNutsAndBolts($N$: nuts, $B$: bolts)

Pick a random nut $n_{\text{pivot}}$ from $N$

Find its matching bolt $b_{\text{pivot}}$ in $B$

$B_L \leftarrow$ All bolts in $B$ smaller than $n_{\text{pivot}}$

$N_L \leftarrow$ All nuts in $N$ smaller than $b_{\text{pivot}}$

$B_R \leftarrow$ All bolts in $B$ larger than $n_{\text{pivot}}$

$N_R \leftarrow$ All nuts in $N$ larger than $b_{\text{pivot}}$

MatchNutsAndBolts($N_R, B_R$)

MatchNutsAndBolts($N_L, B_L$)

QuickSort style...
**MatchNutsAndBolts**($N$: nuts, $B$: bolts)

Pick a random nut $n_{pivot}$ from $N$
Find its matching bolt $b_{pivot}$ in $B$

$B_L \leftarrow$ All bolts in $B$ smaller than $n_{pivot}$
$N_L \leftarrow$ All nuts in $N$ smaller than $b_{pivot}$
$B_R \leftarrow$ All bolts in $B$ larger than $n_{pivot}$
$N_R \leftarrow$ All nuts in $N$ larger than $b_{pivot}$

**MatchNutsAndBolts**($N_R,B_R$)
**MatchNutsAndBolts**($N_L,B_L$)

**QuickSort** style...
What is running time for randomized algorithms?

Definitions

**Definition**

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

**Definition**

Expected running time \( \mathbb{E}[\mathcal{RT}(U)] \) for input \( U \).

**Definition**

*expected running-time* of algorithm for input size \( n \):

\[
T(n) = \max_{U \text{ is an input of size } n} \mathbb{E}[\mathcal{RT}(U)] .
\]
What is running time for randomized algorithms?

**Definitions**

**Definition**

\[RT(U)\]: random variable – *running time* of the algorithm on input \(U\).

**Definition**

Expected running time \(E[RT(U)]\) for input \(U\).

**Definition**

*expected running-time* of algorithm for input size \(n\):

\[T(n) = \max_{U \text{ is an input of size } n} E[RT(U)].\]
What is running time for randomized algorithms?

Definitions

Definition

\( RT(U) \): random variable – running time of the algorithm on input \( U \).

Definition

Expected running time \( E[RT(U)] \) for input \( U \).

Definition

expected running-time of algorithm for input size \( n \):

\[
T(n) = \max_{U \text{ is an input of size } n} E[RT(U)].
\]
What is running time for randomized algorithms?

More definitions

Definition

\[ \text{rank}(x) : \text{rank} \text{ of element } x \in S = \text{number of elements in } S \text{ smaller or equal to } x. \]
Theorem

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

Proof.

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

$$T(n) = \mathbb{E}_{k=\text{rank}(n_{pivot})} \left[ O(n) + T(k - 1) + T(n - k) \right]$$
Nuts and bolts running time

Theorem

Expected running time $\text{MatchNutsAndBolts (QuickSort)}$ is $T(n) = O(n \log n)$. Worst case is $O(n^2)$.

Proof.

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

$$T(n) = \mathbb{E}_{k=\text{rank}(n_{\text{pivot}})} \left[ O(n) + T(k - 1) + T(n - k) \right]$$

$$= O(n) + \mathbb{E}_k [T(k - 1) + T(n - k)]$$
Theorem

Expected running time \textbf{MatchNutsAndBolts (QuickSort)} is \( T(n) = O(n \log n) \). Worst case is \( O(n^2) \).

Proof.

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

\[ T(n) = O(n) + \mathbb{E}_k[T(k - 1) + T(n - k)] \]
Expected running time $\text{MatchNutsAndBolts (QuickSort)}$ is $T(n) = O(n \log n)$. Worst case is $O(n^2)$.

Proof.

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

$$T(n) = O(n) + \mathbb{E}\left[ T(k - 1) + T(n - k) \right]$$

$$= O(n) + \sum_{k=1}^{n} \Pr[\text{Rank(Pivot)} = k] \ast (T(k - 1) + T(n - k))$$
Nuts and bolts running time

**Theorem**

*Expected running time* \texttt{MatchNutsAndBolts (QuickSort)} *is*

\[ T(n) = O(n \log n). \text{ Worst case is } O(n^2). \]

**Proof.**

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

\[
T(n) = O(n) + \sum_{k=1}^{n} \Pr[\text{Rank(Pivot)} = k] \times (T(k - 1) + T(n - k))
\]
Nuts and bolts running time

**Theorem**

*Expected running time* \( \text{MatchNutsAndBolts (QuickSort)} \) *is* \( T(n) = O(n \log n) \). *Worst case is* \( O(n^2) \).

**Proof.**

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

\[
T(n) = O(n) + \sum_{k=1}^{n} \Pr[\text{Rank}(Pivot) = k] \\
* (T(k - 1) + T(n - k)) \\
= O(n) + \sum_{k=1}^{n} \frac{1}{n} \cdot (T(k - 1) + T(n - k)) ,
\]
Theorem

Expected running time \textbf{MatchNutsAndBolts (QuickSort)} is 
\[ T(n) = O(n \log n). \text{ Worst case is } O(n^2). \]

Proof.

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

\[ T(n) = O(n) + \sum_{k=1}^{n} \frac{1}{n} \cdot (T(k - 1) + T(n - k)), \]
Theorem

Expected running time \emph{MatchNutsAndBolts} (\emph{QuickSort}) is 
\[ T(n) = O(n \log n). \] Worst case is \( O(n^2) \).

Proof.

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

\[ T(n) = O(n) + \sum_{k=1}^{n} \frac{1}{n} \cdot (T(k - 1) + T(n - k)), \]

Solution is \( T(n) = O(n \log n). \)
Alternative intuitive analysis...

Which is not formally correct

1. **MatchNutsAndBolts** is *lucky* if $\frac{n}{4} \leq \text{rank}(n_{pivot}) \leq \frac{3}{4}n$.

2. $\Pr[\text{“lucky”}] = 1/2$.

3. $T(n) \leq O(n) + \Pr[\text{“lucky”}] \times (T(n/4) + T(3n/4)) + \Pr[\text{“unlucky”}] \times T(n)$.

4. $T(n) = O(n) + \frac{1}{2} \times \left( T\left(\frac{n}{4}\right) + T\left(\frac{3}{4}n\right) \right) + \frac{1}{2}T(n)$.

5. Rewriting: $T(n) = O(n) + T(n/4) + T((3/4)n)$.

6. ... solution is $O(n \log n)$.
Alternative intuitive analysis...
Which is not formally correct

1. **MatchNutsAndBolts** is *lucky* if $\frac{n}{4} \leq \text{rank}(n_{\text{pivot}}) \leq \frac{3}{4} n$.
2. $\Pr[\text{“lucky”}] = 1/2$.
3. $T(n) \leq O(n) + \Pr[\text{“lucky”}] \times (T(n/4) + T(3n/4)) + \Pr[\text{“unlucky”}] \times T(n)$.
4. $T(n) = O(n) + \frac{1}{2} \times (T(\frac{n}{4}) + T(\frac{3}{4} n)) + \frac{1}{2} T(n)$.
5. Rewriting: $T(n) = O(n) + T(n/4) + T((3/4)n)$.
6. ... solution is $O(n \log n)$.
Alternative intuitive analysis...

Which is not formally correct

1. MatchNutsAndBolts is lucky if \( \frac{n}{4} \leq \text{rank}(n_{pivot}) \leq \frac{3}{4}n \).

2. Pr[“lucky”] = 1/2.

3. \( T(n) \leq O(n) + \text{Pr}[/“lucky”] * (T(n/4) + T(3n/4)) + \text{Pr}[/“unlucky”] * T(n). \)

4. \( T(n) = O(n) + \frac{1}{2} * \left(T\left(\frac{n}{4}\right) + T\left(\frac{3}{4}n\right)\right) + \frac{1}{2} T(n). \)

5. Rewriting: \( T(n) = O(n) + T(n/4) + T((3/4)n). \)

6. ... solution is \( O(n \log n). \)
Alternative intuitive analysis…
Which is not formally correct

1. **MatchNutsAndBolts** is *lucky* if \( \frac{n}{4} \leq \text{rank}(n_{pivot}) \leq \frac{3}{4}n \).

2. \( \Pr[\text{“lucky”}] = 1/2 \).

3. \( T(n) \leq O(n) + \Pr[\text{“lucky”}] \ast (T(n/4) + T(3n/4)) + \Pr[\text{“unlucky”}] \ast T(n) \).

4. \( T(n) = O(n) + \frac{1}{2} \ast \left( T\left(\frac{n}{4}\right) + T\left(\frac{3}{4}n\right) \right) + \frac{1}{2} T(n) \).

5. Rewriting: \( T(n) = O(n) + T(n/4) + T((3/4)n) \).

6. … solution is \( O(n \log n) \).
Alternative intuitive analysis...

Which is not formally correct

1. MatchNutsAndBolts is *lucky* if $\frac{n}{4} \leq \text{rank}(n_{pivot}) \leq \frac{3}{4}n$.

2. $\Pr[\text{“lucky”}] = 1/2$.

3. $T(n) \leq O(n) + \Pr[\text{“lucky”}] \ast (T(n/4) + T(3n/4)) + \Pr[\text{“unlucky”}] \ast T(n)$.

4. $T(n) = O(n) + \frac{1}{2} \ast (T(n/4) + T((3/4)n)) + \frac{1}{2} T(n)$.

5. Rewriting: $T(n) = O(n) + T(n/4) + T((3/4)n)$.

6. ... solution is $O(n \log n)$. 

Sariel (UIUC) CS573 Fall 2013 14 / 32
Alternative intuitive analysis...
Which is not formally correct

1. **MatchNutsAndBolts** is *lucky* if $\frac{n}{4} \leq \text{rank}(n_{pivot}) \leq \frac{3}{4}n$.

2. $\Pr["lucky"] = 1/2$.

3. $T(n) \leq O(n) + \Pr["lucky"] \times (T(n/4) + T(3n/4)) + \Pr["unlucky"] \times T(n)$.

4. $T(n) = O(n) + \frac{1}{2} \times \left( T\left(\frac{n}{4}\right) + T\left(\frac{3}{4}n\right) \right) + \frac{1}{2} T(n)$.

5. **Rewriting:** $T(n) = O(n) + T(n/4) + T((3/4)n)$.

6. ... solution is $O(n \log n)$. 
Worst case vs. average case

Expected running time of a randomized algorithm is

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

Worst case running time of deterministic algorithm:

\[ T(n) = \max_{U \text{ is an input of size } n} R\mathcal{T}(U), \]
High Probability running time...

**Definition**

Running time $\text{Alg}$ is $O(f(n))$ with *high probability* if

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

$$\Rightarrow \Pr\left[\mathcal{RT}(\text{Alg}) > c \cdot f(n)\right] \to 0 \text{ as } n \to \infty.$$

Usually use weaker def:

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

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

---

Sariel (UIUC) CS573 16 Fall 2013 16 / 32
Definition

Running time $\text{Alg}$ is $O(f(n))$ with *high probability* if

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

$$\implies \Pr\left[\mathcal{RT}(\text{Alg}) > c \cdot f(n)\right] \to 0 \text{ as } n \to \infty.$$

Usually use weaker def:

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

Technical reasons... also assume that $\mathbb{E}[\mathcal{RT}(\text{Alg}(n))] = O(f(n))$. 
High Probability running time...

### Definition

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

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

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

Usually use weaker def:

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

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

---

Sariel (UIUC)
CS573 16
Fall 2013 16 / 32
Definition

Running time \( \text{Alg} \) is \( O(f(n)) \) with \textit{high probability} if

\[
\Pr\left[ \mathcal{RT}(\text{Alg}(n)) \geq c \cdot f(n) \right] = o(1).
\]

\[\implies \Pr\left[ \mathcal{RT}(\text{Alg}) > c \cdot f(n) \right] \to 0 \text{ as } n \to \infty.\]

Usually use weaker def:

\[
\Pr\left[ \mathcal{RT}(\text{Alg}(n)) \geq c \cdot f(n) \right] \leq \frac{1}{n^d},
\]

Technical reasons... also assume that \( \mathbb{E}[\mathcal{RT}(\text{Alg}(n))] = O(f(n)). \)
Part II

Slick analysis of QuickSort
A Slick Analysis of \textbf{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. $X_{ij}$: \textit{indicator random} variable for $R_{ij}$. 
   $X_{ij} = 1 \iff \text{rank } i \text{ element compared with rank } j \text{ element, otherwise } 0$.

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

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**

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

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

and hence by linearity of expectation,

\[
E[Q(A)] = \sum_{1 \leq i < j \leq n} E[X_{ij}] = \sum_{1 \leq i < j \leq n} \Pr[R_{ij}].
\]
$R_{ij} = \text{rank } i \text{ element is compared with rank } j \text{ element.}$

**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 \( \Pr[R_{ij}] \)?

With ranks: 6 4 8 1 2 3 7 5
A Slick Analysis of QuickSort

Definition: 

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

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

With ranks: 6 4 8 1 2 3 7 5

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

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

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

With ranks: 6 4 8 1 2 3 7 5

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

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

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

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

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

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

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

2. 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!

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

1. 7 5 9 1 3 4 8 6
2. ⇒ 1 3 4 5 7 9 8 6
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!
A Slick Analysis of **QuickSort**

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

![Array and Comparison Diagram]

1. If pivot is 5 (rank 4). Bingo!
   - Original array: \( \begin{array}{ccccccc} 7 & 5 & 9 & 1 & 3 & 4 & 8 & 6 \\ 6 & 4 & 8 & 1 & 2 & 3 & 7 & 5 \end{array} \)
   - Compared: \( \begin{array}{ccccccc} 7 & 5 & 9 & 1 & 3 & 4 & 8 & 6 \end{array} \)
   - Sorted: \( \begin{array}{ccccccc} 1 & 3 & 4 & 5 & 7 & 9 & 8 & 6 \end{array} \)

2. If pivot is 8 (rank 7). Bingo!
   - Original array: \( \begin{array}{ccccccc} 7 & 5 & 9 & 1 & 3 & 4 & 8 & 6 \\ 6 & 4 & 8 & 1 & 2 & 3 & 7 & 5 \end{array} \)
   - Compared: \( \begin{array}{ccccccc} 7 & 5 & 9 & 1 & 3 & 4 & 8 & 6 \end{array} \)
   - Sorted: \( \begin{array}{ccccccc} 7 & 5 & 1 & 3 & 4 & 6 & 8 & 9 \end{array} \)

3. If pivot in between the two numbers (say 6 [rank 5]):
   - Original array: \( \begin{array}{ccccccc} 7 & 5 & 9 & 1 & 3 & 4 & 8 & 6 \\ 6 & 4 & 8 & 1 & 2 & 3 & 7 & 5 \end{array} \)
   - Compared: \( \begin{array}{ccccccc} 7 & 5 & 9 & 1 & 3 & 4 & 8 & 6 \end{array} \)
   - Sorted: \( \begin{array}{ccccccc} 5 & 1 & 3 & 4 & 6 & 7 & 8 & 9 \end{array} \)

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

5 and 8 will never be compared to each other.
A Slick Analysis of QuickSort

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

**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.

**How to analyze this?**

Thinking acrobatics!

1. Assign every element in the array a random priority (say in $[0, 1]$).
2. Choose pivot to be the element with lowest priority in subproblem.
3. Equivalent to picking pivot uniformly at random (as QuickSort do).
How to analyze this?

Thinking acrobatics!

1. Assign every element in the array a random priority (say in $[0, 1]$).
2. Choose pivot to be the element with lowest priority in subproblem.

$\Rightarrow R_{i,j}$ happens if either $i$ or $j$ have lowest priority out of elements rank $i$ to $j$.

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

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

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

### How to analyze this?

**Thinking acrobatics!**

1. Assign every element in the array a random priority (say in $[0, 1]$).
2. Choose pivot to be the element with lowest priority in subproblem.

$\implies R_{i,j}$ happens if either $i$ or $j$ have lowest priority out of elements rank $i$ to $j$,
There are $k = j - i + 1$ relevant elements.

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

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

Lemma

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

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...
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...
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...
A Slick Analysis of QuickSort

Continued...

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...

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

**Lemma**

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

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

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

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

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

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

Continued...

Lemma

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

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

\[
= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j - i + 1}.
\]
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**

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

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

Continued...

Lemma

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

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

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

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

Continued...

Lemma

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

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

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

\[
E\left[Q(A)\right] = 2 \sum_{i=1}^{n-1} \sum_{i<j} \frac{1}{j-i+1} \leq 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)
Part III

Quick Select
Randomized Quick Selection

**Input**  Unsorted array $A$ of $n$ integers

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

---

**Randomized Quick Selection**

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. Return pivot if rank of pivot is $j$.
4. Otherwise recurse on one of the arrays depending on $j$ and their sizes.
Algorithm for Randomized Selection

Assume for simplicity that $A$ has distinct elements.

QuickSelect($A$, $j$):

Pick pivot $x$ uniformly at random from $A$
Partition $A$ into $A_{\text{less}}$, $x$, and $A_{\text{greater}}$ using $x$ as pivot
if ($|A_{\text{less}}| = j - 1$) then
    return $x$
if ($|A_{\text{less}}| \geq j$) then
    return QuickSelect($A_{\text{less}}$, $j$)
else
    return QuickSelect($A_{\text{greater}}$, $j - |A_{\text{less}}| - 1$)
QuickSelect analysis

1. $S_1, S_2, \ldots, S_k$ be the subproblems considered by the algorithm. Here $|S_1| = n$.

2. $S_i$ would be **successful** if $|S_i| \leq (3/4) |S_{i-1}|$

3. $Y_1 = \text{number of recursive calls till first successful iteration.}$
   Clearly, total work till this happens is $O(Y_1 n)$.

4. $n_i = \text{size of the subproblem immediately after the } (i-1)\text{th successful iteration.}$

5. $Y_i = \text{number of recursive calls after the } (i-1)\text{th successful call, till the } i\text{th successful iteration.}$

6. Running time is $O(\sum_i n_i Y_i)$. 
QuickSelect analysis

Example

\( S_i \) = subarray used in \( i \)th recursive call

\(|S_i|\) = size of this subarray

Red indicates successful iteration.

<table>
<thead>
<tr>
<th>Inst'</th>
<th>( S_1 )</th>
<th>( S_2 )</th>
<th>( S_3 )</th>
<th>( S_4 )</th>
<th>( S_5 )</th>
<th>( S_6 )</th>
<th>( S_7 )</th>
<th>( S_8 )</th>
<th>( S_9 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>S_i</td>
<td>)</td>
<td>100</td>
<td>70</td>
<td>60</td>
<td>50</td>
<td>40</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>Succ'</td>
<td>( Y_1 = 2 )</td>
<td>( Y_2 = 4 )</td>
<td>( Y_3 = 2 )</td>
<td>( Y_4 = 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n_i ) =</td>
<td>( n_1 = 100 )</td>
<td>( n_2 = 60 )</td>
<td>( n_3 = 25 )</td>
<td>( n_4 = 2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. All the subproblems after \((i - 1)\)th successful iteration till \(i\)th successful iteration have size \( \leq n_i \).
2. Total work: \( O(\sum_i n_i Y_i) \).
QuickSelect analysis

Example

\( S_i = \) subarray used in \( i \)th recursive call
\( |S_i| = \) size of this subarray
Red indicates successful iteration.

<table>
<thead>
<tr>
<th>Inst’</th>
<th>( S_1 )</th>
<th>( S_2 )</th>
<th>( S_3 )</th>
<th>( S_4 )</th>
<th>( S_5 )</th>
<th>( S_6 )</th>
<th>( S_7 )</th>
<th>( S_8 )</th>
<th>( S_9 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>S_i</td>
<td>)</td>
<td>100</td>
<td>70</td>
<td>60</td>
<td>50</td>
<td>40</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>Succ’</td>
<td>( Y_1 = 2 )</td>
<td>( Y_2 = 4 )</td>
<td>( Y_3 = 2 )</td>
<td>( Y_4 = 1 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n_i = )</td>
<td>( n_1 = 100 )</td>
<td>( n_2 = 60 )</td>
<td>( n_3 = 25 )</td>
<td>( n_4 = 2 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. All the subproblems after \( (i - 1) \)th successful iteration till \( i \)th successful iteration have size \( \leq n_i \).
2. Total work: \( O(\sum_i n_i Y_i) \).
QuickSelect analysis

Total work: \( O(\sum_i n_i Y_i) \).

We have:

1. \( n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1}n \).
2. \( Y_i \) is a random variable with geometric distribution
   Probability of \( Y_i = k \) is \( 1/2^i \).
3. \( E[Y_i] = 2 \).

As such, expected work is proportional to

\[
E\left[\sum_i n_i Y_i\right] = \sum_i E[n_i Y_i] \leq \sum_i E[(3/4)^{i-1} n Y_i]
\]

\[
= n \sum_i (3/4)^{i-1} E[Y_i] = n \sum_{i=1} E[Y_i] \leq 8n.
\]
QuickSelect analysis

Theorem

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