Randomized Algorithms: QuickSort and QuickSelect

Lecture 15
March 12, 2015
Part I

Slick analysis of QuickSort
A Slick Analysis of **QuickSort**

1. Let $Q(A)$ be number of comparisons done on input array $A$:
   - $R_{ij}$: event that rank $i$ element is compared with rank $j$ element, for $1 \leq i < j \leq n$.
   - $X_{ij}$ is the indicator random variable for $R_{ij}$. That is, $X_{ij} = 1$ if rank $i$ is compared with rank $j$ element, otherwise $0$.

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

3. By linearity of expectation,

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

$$
= \sum_{1 \leq i < j \leq n} \Pr[R_{ij}].
$$
A Slick Analysis of QuickSort

1. Let $Q(A)$ be number of comparisons done on input array $A$:
   1. $R_{ij}$: event that rank $i$ element is compared with rank $j$ element, for $1 \leq i < j \leq n$.
   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 $0$.

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

3. By linearity of expectation,

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

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

1. Let $Q(A)$ be number of comparisons done on input array $A$:
   1. $R_{ij}$: event that rank $i$ element is compared with rank $j$ element, for $1 \leq i < j \leq n$.
   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.

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

3. By linearity of expectation,

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

   $$
   = \sum_{1 \leq i < j \leq n} Pr[R_{ij}].
   $$
A Slick Analysis of **QuickSort**

1. Let $Q(A)$ be number of comparisons done on input array $A$:
   - $R_{ij}$: event that rank $i$ element is compared with rank $j$ element, for $1 \leq i < j \leq n$.
   - $X_{ij}$ is the indicator random variable for $R_{ij}$. That is, $X_{ij} = 1$ if rank $i$ is compared with rank $j$ element, otherwise $0$.

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

3. By linearity of expectation, 

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

\[
= \sum_{1 \leq i < j \leq n} \Pr\left[R_{ij}\right].
\]
A Slick Analysis of **QuickSort**

1. Let $Q(A)$ be number of comparisons done on input array $A$:
   - $R_{ij}$: event that rank $i$ element is compared with rank $j$ element, for $1 \leq i < j \leq n$.
   - $X_{ij}$ is the indicator random variable for $R_{ij}$. That is, $X_{ij} = 1$ if rank $i$ is compared with rank $j$ element, otherwise $0$.

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

3. By linearity of expectation,

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

   $$
   = \sum_{1 \leq i < j \leq n} \Pr\left[ R_{ij} \right].
   $$
Let $Q(A)$ be the number of comparisons done on input array $A$:

1. $R_{ij}$: event that rank $i$ element is compared with rank $j$ element, for $1 \leq i < j \leq n$.
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 $0$.

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

By linearity of expectation,

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

$$= \sum_{1 \leq i < j \leq n} \Pr[R_{ij}].$$
A Slick Analysis of QuickSort

1. Let $Q(A)$ be number of comparisons done on input array $A$:
   - $R_{ij}$: event that rank $i$ element is compared with rank $j$ element, for $1 \leq i < j \leq n$.
   - $X_{ij}$ is the indicator random variable for $R_{ij}$. That is, $X_{ij} = 1$ if rank $i$ is compared with rank $j$ element, otherwise $0$.

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

3. By linearity of expectation,

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

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

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

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

7 5 9 1 3 4 8 6
$R_{ij} = \text{rank } i \text{ element is compared with rank } j \text{ element.}$

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

With ranks: 6 4 8 1 2 3 7 5
$R_{ij}$ = 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

1. 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:

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

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

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

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

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!

1. 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
   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 $\Pr[R_{i,j}]$?

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

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

   ![Diagram 1](image1.png)

   7 5 9 1 3 4 8 6

   $\Rightarrow$

   1 3 4 5 7 9 8 6

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

   ![Diagram 2](image2.png)

   7 5 9 1 3 4 8 6

   $\Rightarrow$

   7 5 1 3 4 6 8 9

3. If pivot in between the two numbers (say 6 [rank 5]):

   ![Diagram 3](image3.png)

   7 5 9 1 3 4 8 6

   $\Rightarrow$

   5 1 3 4 6 7 8 9

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 $\iff$

1. $i$th or $j$th ranked element is the first pivot out of the elements of rank $i, i+1, i+2, \ldots, j$

**How to analyze this? Thinking acrobatics!**

1. Assign every element in array random priority (say in $[0, 1]$).
2. Choose pivot to be element with lowest priority in subproblem.
3. Equivalent to picking pivot uniformly at random (as QuickSort do).
**A Slick Analysis of QuickSort**

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

**Conclusion:** $R_{i,j}$ happens iff:

- $i$th or $j$th ranked element is the first pivot out of the elements of rank $i, i + 1, i + 2, \ldots, j$.

**How to analyze this? Thinking acrobatics!**

1. Assign every element in array random priority (say in $[0, 1]$).
2. Choose pivot to be element with lowest priority in subproblem.
3. Equivalent to picking pivot uniformly at random (as QuickSort do).
A Slick Analysis of **QuickSort**

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

**Conclusion:**

$R_{i,j}$ happens $\iff$

$i$th or $j$th ranked element is the first pivot out of the elements of rank $i, i + 1, i + 2, \ldots, j$

---

**How to analyze this? Thinking acrobatics!**

1. Assign every element in array random priority (say in $[0, 1]$).
2. Choose pivot to be element with lowest priority in subproblem.
3. Equivalent to picking pivot uniformly at random (as **QuickSort** do).
A Slick Analysis of QuickSort

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

Conclusion: $R_{i,j}$ happens $\iff$:

$i$th or $j$th ranked element is the first pivot out of the elements of rank $i, i + 1, i + 2, \ldots, j$

How to analyze this? Thinking acrobatics!

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

1. Assign every element in array is a random priority (in \([0, 1]\)).
2. pivot = the element with lowest priority in subproblem.

\( R_{i,j} \) happens if either \( i \) or \( j \) have lowest priority out of elements in rank \( i \ldots j \),

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 $\Pr[R_{i,j}]$?

1. **Choosing a pivot using priorities**
   - Assign every element in array is a random priority (in $[0, 1]$).
   - pivot $= \text{the element with lowest priority in subproblem}.$

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

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

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

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

1. **Choosing a pivot using priorities**
   1. Assign every element in array is a random priority (in $[0, 1]$).
   2. $\text{pivot} = \text{the element with lowest priority in subproblem}$.

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

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

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

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

1. **Choosing a pivot using priorities**
   1. Assign every element in array is a random priority (in $[0, 1]$).
   2. $\text{pivot} = \text{the element with lowest priority in subproblem}$.

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

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

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

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

**Lemma**

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

**Proof**

1. $a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n$: elements of $A$ in sorted order. Let $S = \{a_i, a_{i+1}, \ldots, a_j\}$

2. **Observation:** If pivot is chosen outside $S$ then all of $S$ either in left or right recursive subproblem.

3. **Observation:** $a_i$ and $a_j$ separated when a pivot is chosen from $S$ for the first time. Once separated never to meet again. $\implies a_i$ and $a_j$ will not be compared.
A Slick Analysis of QuickSort

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

**Lemma**

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

**Proof**

1. $a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n$: elements of $A$ in sorted order. Let $S = \{a_i, a_{i+1}, \ldots, a_j\}$

2. **Observation:** If pivot is chosen outside $S$ then all of $S$ either in left or right recursive subproblem.

3. **Observation:** $a_i$ and $a_j$ separated when a pivot is chosen from $S$ for the first time. Once separated never to meet again. $\implies a_i$ and $a_j$ will not be compared.
A Slick Analysis of QuickSort

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

**Lemma**

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

**Proof**

1. \( a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n \): elements of \( A \) in sorted order. Let \( S = \{a_i, a_{i+1}, \ldots, a_j\} \)

2. **Observation:** If pivot is chosen outside \( S \) then all of \( S \) either in left or right recursive subproblem.

3. **Observation:** \( a_i \) and \( a_j \) separated when a pivot is chosen from \( S \) for the first time. Once separated never to meet again. \( \implies \) \( a_i \) and \( a_j \) will not be compared.
A Slick Analysis of QuickSort

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

Lemma

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

Proof

1. \( a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n \): elements of \( A \) in sorted order. Let \( S = \{a_i, a_{i+1}, \ldots, a_j\} \)

2. Observation: If pivot is chosen outside \( S \) then all of \( S \) either in left or right recursive subproblem.

3. Observation: \( a_i \) and \( a_j \) separated when a pivot is chosen from \( S \) for the first time. Once separated never to meet again. \( \implies \) \( a_i \) and \( a_j \) will not be compared.
A Slick Analysis of QuickSort

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

**Lemma**

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

**Proof**

1. $a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n$: elements of $A$ in sorted order. Let $S = \{a_i, a_{i+1}, \ldots, a_j\}$

2. **Observation:** If pivot is chosen outside $S$ then all of $S$ either in left or right recursive subproblem.

3. **Observation:** $a_i$ and $a_j$ separated when a pivot is chosen from $S$ for the first time. Once separated never to meet again. $\implies a_i$ and $a_j$ will not be compared.
**A Slick Analysis of QuickSort**

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

**Lemma**

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

**Proof**

1. \( a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n \): elements of \( A \) in sorted order. Let \( S = \{a_i, a_{i+1}, \ldots, a_j\} \)

2. **Observation:** If pivot is chosen outside \( S \) then all of \( S \) either in left or right recursive subproblem.

3. **Observation:** \( a_i \) and \( a_j \) separated when a pivot is chosen from \( S \) for the first time. Once separated never to meet again. \( \implies \) \( a_i \) and \( a_j \) will not be compared.
Lemma

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

Proof.

1. Let \( a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n \) be sort of \( A \).
2. Let \( S = \{a_i, a_{i+1}, \ldots, a_j\} \)
3. Observation: \( a_i \) is compared with \( a_j \) \( \iff \) either \( a_i \) or \( a_j \) is chosen as a pivot from \( S \) at separation.
4. Observation: Given: Pivot 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...

Lemma

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

Proof.

1. Let \( a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n \) be sort of \( A \).
2. Let \( S = \{a_i, a_{i+1}, \ldots, a_j\} \)
3. **Observation:** \( a_i \) is compared with \( a_j \) \( \iff \) either \( a_i \) or \( a_j \) is chosen as a pivot from \( S \) at separation.
4. **Observation:** Given: Pivot chosen from \( S \). The probability that it is \( a_i \) or \( a_j \) is exactly \( \frac{2}{|S|} \) since the pivot is chosen uniformly at random from the array.
Lemma
\[ \Pr[R_{ij}] = \frac{2}{j-i+1}. \]

Proof.
1. Let \( a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n \) be sort of \( A \).
2. Let \( S = \{a_i, a_{i+1}, \ldots, a_j\} \)
3. **Observation:** \( a_i \) is compared with \( a_j \) \( \iff \) either \( a_i \) or \( a_j \) is chosen as a pivot from \( S \) at separation.
4. **Observation:** Given: Pivot 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...

Lemma

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

Proof.

1. Let $a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n$ be sort of $A$.
2. Let $S = \{a_i, a_{i+1}, \ldots, a_j\}$
3. **Observation:** $a_i$ is compared with $a_j$ either $a_i$ or $a_j$ is chosen as a pivot from $S$ at separation.
4. **Observation:** Given: Pivot 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.
Lemma

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

Proof.

1. Let \( a_1, \ldots, a_i, \ldots, a_j, \ldots, a_n \) be sort of \( A \).
2. Let \( S = \{a_i, a_{i+1}, \ldots, a_j\} \)
3. **Observation**: \( a_i \) is compared with \( a_j \) \( \iff \) either \( a_i \) or \( a_j \) is chosen as a pivot from \( S \) at separation.
4. **Observation**: Given: Pivot chosen from \( S \). The probability that it is \( a_i \) or \( a_j \) is exactly \( \frac{2}{|S|} = \frac{2}{(j - i + 1)} \) since the pivot is chosen uniformly at random from the array.
A Slick Analysis of \texttt{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**

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

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

Continued...

Lemma

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

\[
E\left[ Q(A) \right] = \sum_{1 \leq i < j \leq n} \Pr[R_{ij}] = \sum_{1 \leq i < j \leq n} \frac{2}{j - i + 1}
\]
A Slick Analysis of **QuickSort**

Continued...

**Lemma**

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

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

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

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

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

Continued...

**Lemma**

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

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

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

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

Continued...

Lemma

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

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

Continued...

**Lemma**

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

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

\[
\leq 2 \sum_{i=1}^{n-1} (H_{n-i+1} - 1) \leq 2 \sum_{1 \leq i < n} H_n
\]
A Slick Analysis of **QuickSort**

**Continued...**

**Lemma**

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

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

\[
\leq 2 \sum_{i=1}^{n-1} (H_{n-i+1} - 1) \leq 2 \sum_{1 \leq i < n}^{n} H_{n}
\]

\[
\leq 2nH_{n} = O(n \log n)
\]
Part II

Quick sort with high probability
Yet another analysis of QuickSort

You should never trust a man who has only one way to spell a word

1. Consider element \( e \) in the array.

2. \( S_1, S_2, \ldots, S_k \): subproblems \( e \) participates in during QuickSort execution:

3. **Definition**
   \( e \) is lucky in the \( j \)th iteration if \( |S_j| \leq (3/4) |S_{j-1}| \).

4. **Key observation**: The event that \( e \) is lucky in \( j \)th iteration...

5. ... is independent of the event that \( e \) is lucky in \( k \)th iteration, (if \( j \neq k \))

6. \( X_j = 1 \iff e \) is lucky in the \( j \)th iteration.
Consider element $e$ in the array.

$S_1, S_2, \ldots, S_k$: subproblems $e$ participates in during QuickSort execution:

**Definition**

$e$ is lucky in the $j$th iteration if $|S_j| \leq (3/4)|S_{j-1}|$.

**Key observation**: The event that $e$ is lucky in $j$th iteration...

... is independent of the event that $e$ is lucky in $k$th iteration, (If $j \neq k$)

$X_j = 1 \iff e$ is lucky in the $j$th iteration.
Yet another analysis of QuickSort
You should never trust a man who has only one way to spell a word

1. Consider element $e$ in the array.
2. $S_1, S_2, \ldots, S_k$: subproblems $e$ participates in during QuickSort execution:

Definition

$e$ is lucky in the $j$th iteration if $|S_j| \leq (3/4) |S_{j-1}|$.

4. Key observation: The event that $e$ is lucky in $j$th iteration...
5. ... is independent of the event that $e$ is lucky in $k$th iteration, (If $j \neq k$)
6. $X_j = 1 \iff e$ is lucky in the $j$th iteration.
Yet another analysis of QuickSort
You should never trust a man who has only one way to spell a word

1 Consider element \( e \) in the array.

2 \( S_1, S_2, \ldots, S_k \): subproblems \( e \) participates in during QuickSort execution:

3 **Definition**

\[ e \text{ is lucky in the } j\text{th iteration if } |S_j| \leq \frac{3}{4} |S_{j-1}|. \]

4 **Key observation**: The event that \( e \) is lucky in \( j \)th iteration...

5 ... is independent of the event that \( e \) is lucky in \( k \)th iteration, (If \( j \neq k \))

6 \( X_j = 1 \iff e \text{ is lucky in the } j\text{th iteration.} \)
Yet another analysis of QuickSort
You should never trust a man who has only one way to spell a word

1. Consider element $e$ in the array.

2. $S_1, S_2, \ldots, S_k$: subproblems $e$ participates in during QuickSort execution:

3. **Definition**
   
   $e$ is lucky in the $j$th iteration if $|S_j| \leq (3/4)|S_{j-1}|$.

4. **Key observation**: The event that $e$ is lucky in $j$th iteration...

5. ... is independent of the event that $e$ is lucky in $k$th iteration, (If $j \neq k$)

6. $X_j = 1 \iff e$ is lucky in the $j$th iteration.
Yet another analysis of QuickSort

You should never trust a man who has only one way to spell a word

1. Consider element $e$ in the array.
2. $S_1, S_2, \ldots, S_k$: subproblems $e$ participates in during QuickSort execution:

   **Definition**
   
   $e$ is lucky in the $j$th iteration if $|S_j| \leq \frac{3}{4} |S_{j-1}|$.

4. **Key observation**: The event that $e$ is lucky in $j$th iteration...
5. ... is independent of the event that $e$ is lucky in $k$th iteration, (If $j \neq k$)
6. $X_j = 1 \iff e$ is lucky in the $j$th iteration.
Claim

\[ \Pr[X_j = 1] = 1/2. \]

Proof.

1. \( X_j \) determined by \( j \) recursive subproblem.
2. Subproblem has \( n_{j-1} = |X_{j-1}| \) elements.
3. \( j \)'th pivot rank \( \in [n_{j-1}/4, (3/4)n_{j-1}] \implies e \) lucky in \( j \)'th iter.
4. Prob. \( e \) is lucky \( \geq \frac{|[n_{j-1}/4, (3/4)n_{j-1}]|}{n_{j-1}} = 1/2. \)

Observation

If \( X_1 + X_2 + \ldots X_k = \lceil \log_{4/3} n \rceil \) then \( e \) subproblem is of size one.

Done!
Yet another analysis of QuickSort

Continued...

Claim

\[ \Pr[X_j = 1] = 1/2. \]

Proof.

1. \( X_j \) determined by \( j \) recursive subproblem.
2. Subproblem has \( n_{j-1} = |X_{j-1}| \) elements.
3. \( j \)th pivot rank \( \in [n_{j-1}/4, (3/4)n_{j-1}] \) \( \implies \) e lucky in \( j \)th iter.
4. Prob. e is lucky \( \geq [n_{j-1}/4, (3/4)n_{j-1}]/n_{j-1} = 1/2. \)

Observation

If \( X_1 + X_2 + \ldots X_k = \lceil \log_{4/3} n \rceil \) then e subproblem is of size one. Done!
Yet another analysis of QuickSort

Continued...

Claim

\[ \Pr[X_j = 1] = \frac{1}{2}. \]

Proof.

1. \( X_j \) determined by \( j \) recursive subproblem.
2. Subproblem has \( n_{j-1} = |X_{j-1}| \) elements.
3. \( j \)th pivot rank \( \in [n_{j-1}/4, (3/4)n_{j-1}] \) \( \implies \) \( e \) lucky in \( j \)th iter.
4. Prob. \( e \) is lucky \( \geq |[n_{j-1}/4, (3/4)n_{j-1}]| / n_{j-1} = 1/2. \)

Observation

If \( X_1 + X_2 + \ldots X_k = \lceil \log_{4/3} n \rceil \) then \( e \) subproblem is of size one.

Done!
Yet another analysis of QuickSort

Continued...

Claim

\[ \Pr[X_j = 1] = 1/2. \]

Proof.

1. \( X_j \) determined by \( j \) recursive subproblem.
2. Subproblem has \( n_{j-1} = |X_{j-1}| \) elements.
3. \( j \)th pivot rank \( \in \left[ n_{j-1}/4, (3/4)n_{j-1} \right] \implies e \) lucky in \( j \)th iter.
4. Prob. \( e \) is lucky \( \geq \frac{|\left[ n_{j-1}/4, (3/4)n_{j-1} \right]|}{n_{j-1}} = 1/2. \)

Observation

If \( X_1 + X_2 + \ldots + X_k = \left\lfloor \log_{4/3} n \right\rfloor \) then \( e \) subproblem is of size one.

Done!
Claim

\[ \Pr[X_j = 1] = 1/2. \]

Proof.

1. \( X_j \) determined by \( j \) recursive subproblem.
2. Subproblem has \( n_{j-1} = |X_{j-1}| \) elements.
3. \( j \)th pivot rank \( \in [n_{j-1}/4, (3/4)n_{j-1}] \iff \) \( e \) lucky in \( j \)th iter.
4. Prob. \( e \) is lucky \( \geq \frac{|n_{j-1}/4, (3/4)n_{j-1}|}{n_{j-1}} = 1/2. \)

Observation

If \( X_1 + X_2 + \ldots + X_k = \lceil \log_{4/3} n \rceil \) then \( e \) subproblem is of size one.

Done!
Yet another analysis of QuickSort
Continued...

Claim

\[ \Pr[X_j = 1] = \frac{1}{2}. \]

Proof.

1. \( X_j \) determined by \( j \) recursive subproblem.
2. Subproblem has \( n_{j-1} = |X_{j-1}| \) elements.
3. \( j \)th pivot rank \( \in [n_{j-1}/4, (3/4)n_{j-1}] \implies e \) lucky in \( j \)th iter.
4. Prob. \( e \) is lucky \( \geq \frac{|[n_{j-1}/4, (3/4)n_{j-1}]|}{n_{j-1}} = \frac{1}{2}. \)

Observation

If \( X_1 + X_2 + \ldots X_k = \lceil \log_{4/3} n \rceil \) then \( e \) subproblem is of size one. Done!
Yet another analysis of QuickSort

Continued...

Claim

\[ \Pr[X_j = 1] = 1/2. \]

Proof.

1. \( X_j \) determined by \( j \) recursive subproblem.
2. Subproblem has \( n_{j-1} = |X_{j-1}| \) elements.
3. \( j \)th pivot rank \( \in [n_{j-1}/4, (3/4)n_{j-1}] \implies e \) lucky in \( j \)th iter.
4. Prob. \( e \) is lucky \( \geq |[n_{j-1}/4, (3/4)n_{j-1}]| / n_{j-1} = 1/2. \)

Observation

If \( X_1 + X_2 + \ldots X_k = \lceil \log_{4/3} n \rceil \) then \( e \) subproblem is of size one. Done!
Yet another analysis of QuickSort

Continued...

Observation

Probability \( e \) participates in \( \geq k = 40 \lceil \log_{4/3} n \rceil \) subproblems. Is equal to

\[
\Pr \left[ X_1 + X_2 + \ldots + X_k \leq \lceil \log_{4/3} n \rceil \right] \\
\leq \Pr \left[ X_1 + X_2 + \ldots + X_k \leq k/4 \right] \\
\leq 2 \cdot 0.68^{k/4} \leq 1/n^5.
\]

Conclusion

QuickSort takes \( O(n \log n) \) time with high probability.
Because...

**Theorem**

Let $X_n$ be the number heads when flipping a coin independently $n$ times. Then

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

Randomized selection
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. $Y_1 =$ number of recursive calls till first successful iteration. Clearly, total work till this happens is $O(Y_1 n)$.

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

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

6. Running time is $O(\sum_i n_i Y_i)$.
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 \textbf{successful} if \(|S_i| \leq (3/4)|S_{i-1}|\)

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

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

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

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

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

6. Running time is $O(\sum_i n_i Y_i)$. 
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. \( Y_1 \) = number of recursive calls till first successful iteration. Clearly, total work till this happens is \( O(Y_1 n) \).

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

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

6. Running time is \( O(\sum_i n_i Y_i) \).
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></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>S_i</td>
<td>)</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Succ’</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(n_i)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**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></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>S_i</td>
<td>$</td>
<td>100</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>Succ’</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n_i =$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>S_i</td>
<td>$</td>
</tr>
</tbody>
</table>

Succ’

$n_i =$
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>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>S_i</td>
<td>)</td>
</tr>
<tr>
<td>Succ’</td>
<td>( Y_1 = 2 )</td>
<td></td>
</tr>
<tr>
<td>( n_i = )</td>
<td>( n_1 = 100 )</td>
<td></td>
</tr>
</tbody>
</table>
QuickSelect analysis

Example

$S_i = \text{subarray used in } i\text{th recursive call}$

$|S_i| = \text{size of this subarray}$

Red indicates successful iteration.

<table>
<thead>
<tr>
<th>Inst’</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>S_i</td>
<td>$</td>
<td>100</td>
</tr>
<tr>
<td>Succ’</td>
<td>$Y_1 = 2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n_i = $</td>
<td>$n_1 = 100$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**QuickSelect analysis**

**Example**

\[ S_i = \text{subarray used in } i\text{th recursive call} \]

\[ |S_i| = \text{size of this subarray} \]

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>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>S_i</td>
<td>)</td>
<td>100</td>
<td>70</td>
</tr>
<tr>
<td>Succ’</td>
<td>( Y_1 = 2 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n_i = )</td>
<td>( n_1 = 100 )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**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>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>S_i</td>
<td>$</td>
<td>100</td>
<td>70</td>
<td>60</td>
</tr>
<tr>
<td>Succ’</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n_i$ =</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**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>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>S_i</td>
<td>)</td>
<td>100</td>
<td>70</td>
<td>60</td>
<td>50</td>
</tr>
<tr>
<td>Succ’</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| \( n_i = \) | \( n_1 = 100 \) }
**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>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>S_i</td>
<td>$</td>
<td>100</td>
<td>70</td>
<td>60</td>
<td>50</td>
</tr>
<tr>
<td>Succ’</td>
<td>Y₁ = 2</td>
<td></td>
<td></td>
<td></td>
<td>Y₂ = 4</td>
<td></td>
</tr>
<tr>
<td>$n_i$ =</td>
<td>$n_1 = 100$</td>
<td></td>
<td>$n_2 = 60$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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>
</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>
</tr>
<tr>
<td>Succ'</td>
<td>( Y_1 = 2 )</td>
<td>( Y_2 = 4 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( n_i = \)

| \( n_1 = 100 \) | \( n_2 = 60 \) |
QuickSelect analysis

Example

$S_i = \text{subarray used in } i\text{th recursive call}$

$|S_i| = \text{size of this subarray}$

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>
</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>
</tr>
<tr>
<td>Succ’</td>
<td>$Y_1 = 2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n_i =$</td>
<td>$n_1 = 100$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$n_2 = 60$</td>
</tr>
</tbody>
</table>
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>
</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>
</tr>
<tr>
<td>Succ’</td>
<td>$Y_1=2$</td>
<td></td>
<td></td>
<td>$Y_2=4$</td>
<td></td>
<td></td>
<td>$Y_3=2$</td>
<td></td>
</tr>
<tr>
<td>$n_i =$</td>
<td>$n_1 = 100$</td>
<td></td>
<td></td>
<td>$n_2 = 60$</td>
<td></td>
<td></td>
<td>$n_3 = 25$</td>
<td></td>
</tr>
</tbody>
</table>
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></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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**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></td>
<td>$Y_2 = 4$</td>
<td></td>
<td>$Y_3 = 2$</td>
<td></td>
<td>$Y_4 = 1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$n_i =$</td>
<td>$n_1 = 100$</td>
<td>$n_2 = 60$</td>
<td></td>
<td>$n_3 = 25$</td>
<td></td>
<td>$n_4 = 2$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 \( \leq n_i \).
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>$</td>
<td>S_i</td>
<td></td>
<td>100</td>
<td>70</td>
<td>60</td>
<td>50</td>
<td>40</td>
<td>30</td>
<td>25</td>
</tr>
<tr>
<td>Succ'</td>
<td>Y_1 = 2</td>
<td></td>
<td>Y_2 = 4</td>
<td></td>
<td>Y_3 = 2</td>
<td></td>
<td>Y_4 = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$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

1. Total work: \( O(\sum_i n_i Y_i) \).
2. \( n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1}n \).
3. \( Y_i \) is a random variable with geometric distribution
   Probability of \( Y_i = k \) is \( 1/2^i \).
4. \( E[Y_i] = 2 \).
5. As such, expected work is proportional to

\[
E\left[ \sum_i n_i Y_i \right]
\]
QuickSelect analysis

1. Total work: $O(\sum_i n_i Y_i)$.
2. $n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1}n$.
3. $Y_i$ is a random variable with geometric distribution. Probability of $Y_i = k$ is $1/2^i$.
5. As such, expected work is proportional to

$$E\left[\sum_i n_i Y_i\right]$$
QuickSelect analysis

1. Total work: \( O(\sum_i n_i Y_i) \).
2. \( n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1}n \).
3. \( Y_i \) is a random variable with geometric distribution. Probability of \( Y_i = k \) is \( 1/2^i \).
4. \( E[Y_i] = 2 \).
5. As such, expected work is proportional to

\[
E\left[ \sum_i n_i Y_i \right]
\]
QuickSelect analysis

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

2. $n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1}n$.

3. $Y_i$ is a random variable with geometric distribution
   Probability of $Y_i = k$ is $1/2^i$.


5. As such, expected work is proportional to

   $$E\left[ \sum_i n_i Y_i \right]$$
QuickSelect analysis

1. Total work: \( O(\sum_i n_i Y_i) \).
2. \( n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1}n \).
3. \( Y_i \) is a random variable with geometric distribution
   Probability of \( Y_i = k \) is \( 1/2^i \).
4. \( E[Y_i] = 2 \).
5. As such, expected work is proportional to

\[
E \left[ \sum_i n_i Y_i \right]
\]
QuickSelect analysis

1. Total work: $O(\sum_i n_i Y_i)$.
2. $n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1}n$.
3. $Y_i$ is a random variable with geometric distribution. Probability of $Y_i = k$ is $1/2^i$.
5. As such, expected work is proportional to

$$E\left[\sum_i n_i Y_i\right]$$
QuickSelect analysis

1. Total work: $O(\sum_i n_i Y_i)$.
2. $n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1} n$.
3. $Y_i$ is a random variable with geometric distribution. Probability of $Y_i = k$ is $1/2^i$.
5. As such, expected work is proportional to

$$E\left[\sum_i n_i Y_i\right]$$
QuickSelect analysis

1. Total work: $O(\sum_i n_i Y_i)$.
2. $n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1}n$.
3. $Y_i$ is a random variable with geometric distribution
   Probability of $Y_i = k$ is $1/2^i$.
5. As such, expected work is proportional to

\[
E\left[\sum_i n_i Y_i\right] = \sum_i E\left[n_i Y_i\right]
\]
QuickSelect analysis

1. Total work: $O(\sum_i n_i Y_i)$.
2. $n_i \leq (3/4) n_{i-1} \leq (3/4)^{i-1} n$.
3. $Y_i$ is a random variable with geometric distribution. Probability of $Y_i = k$ is $1/2^i$.
5. As such, expected work is proportional to

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

1. Total work: $O(\sum_i n_i Y_i)$.
2. $n_i \leq (3/4) n_{i-1} \leq (3/4)^{i-1} n$.
3. $Y_i$ is a random variable with geometric distribution. Probability of $Y_i = k$ is $1/2^i$.
5. As such, expected work is proportional to

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

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

2. $n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1}n$.

3. $Y_i$ is a random variable with geometric distribution
   Probability of $Y_i = k$ is $1/2^i$.


5. As such, expected work is proportional to

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

$$= n \sum_i (3/4)^{i-1} E[Y_i] = n \sum_{i=1} E(3/4)^{i-1} 2$$
QuickSelect analysis

1. Total work: $O(\sum_i n_i Y_i)$.
2. $n_i \leq (3/4)n_{i-1} \leq (3/4)^{i-1} n$.
3. $Y_i$ is a random variable with geometric distribution. Probability of $Y_i = k$ is $1/2^i$.
5. As such, expected work is proportional to

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

$$= n \sum_i (3/4)^{i-1} E[Y_i] = n \sum_{i=1} \sum_i (3/4)^{i-1} 2 \leq 8n.$$
QuickSelect analysis

Theorem

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

Analysis via Recurrence

1. Given array $A$ of size $n$ let $Q(A)$ be number of comparisons of randomized selection on $A$ for selecting rank $j$ element.

2. Note that $Q(A)$ is a random variable.

3. Let $A_{\text{less}}^i$ and $A_{\text{greater}}^i$ be the left and right arrays obtained if pivot is rank $i$ element of $A$.

4. Algorithm recurses on $A_{\text{less}}^i$ if $j < i$ and recurses on $A_{\text{greater}}^i$ if $j > i$ and terminates if $j = i$.

$$Q(A) = n + \sum_{i=1}^{j-1} \Pr[\text{pivot has rank } i] Q(A_{\text{greater}}^i) + \sum_{i=j+1}^{n} \Pr[\text{pivot has rank } i] Q(A_{\text{less}}^i)$$
QuickSelect analysis via recurrence

Analysis via Recurrence

1. Given array \( A \) of size \( n \) let \( Q(A) \) be number of comparisons of randomized selection on \( A \) for selecting rank \( j \) element.

2. Note that \( Q(A) \) is a random variable

3. Let \( A^i_{\text{less}} \) and \( A^i_{\text{greater}} \) be the left and right arrays obtained if pivot is rank \( i \) element of \( A \).

4. Algorithm recurses on \( A^i_{\text{less}} \) if \( j < i \) and recurses on \( A^i_{\text{greater}} \) if \( j > i \) and terminates if \( j = i \).

\[
Q(A) = n + \sum_{i=1}^{j-1} \Pr[\text{pivot has rank } i] Q(A^i_{\text{greater}})
\]

\[
+ \sum_{i=j+1}^{n} \Pr[\text{pivot has rank } i] Q(A^i_{\text{less}})
\]

Sariel (UIUC)
Analyzing the Recurrence

As in QuickSort we obtain the following recurrence where $T(n)$ is the worst-case expected time.

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

**Theorem**

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

**Proof.**

(Guess and) Verify by induction (see next slide).
Analyzing the recurrence

**Theorem**

\[ T(n) = O(n). \]

Prove by induction that \( T(n) \leq \alpha n \) for some constant \( \alpha \geq 1 \) to be fixed later.

**Base case:** \( n = 1 \), we have \( T(1) = 0 \) since no comparisons needed and hence \( T(1) \leq \alpha \).

**Induction step:** Assume \( T(k) \leq \alpha k \) for \( 1 \leq k < n \) and prove it for \( T(n) \). We have by the recurrence:

\[
T(n) \leq n + \frac{1}{n} \left( \sum_{i=1}^{j-1} T(n - i) + \sum_{i=j}^{n} T(i - 1) \right) \\
\leq n + \frac{\alpha}{n} \left( \sum_{i=1}^{j-1} (n - i) + \sum_{i=j}^{n} (i - 1) \right) \quad \text{by applying induction}
\]
Analyzing the recurrence

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

\[ \leq n + \frac{\alpha}{n} \frac{(j - 1)(2n - j)}{2} + \frac{(n - j + 1)(n + j - 2)}{2} \]

\[ \leq n + \frac{\alpha}{2n} \left( n^2 + 2nj - 2j^2 - 3n + 4j - 2 \right) \]

above expression maximized when \( j = (n + 1)/2 \): calculus

\[ \leq n + \frac{\alpha}{2n} \left( \frac{3n^2}{2} - n \right) \]

substituting \( (n + 1)/2 \) for \( j \)

\[ \leq n + \frac{3\alpha n}{4} \]

\[ \leq \alpha n \quad \text{for any constant } \alpha \geq 4 \]
Comments on analyzing the recurrence

1. Algebra looks messy but intuition suggest that the median is the hardest case and hence can plug $j = n/2$ to simplify without calculus.

2. Analyzing recurrences comes with practice and after a while one can see things more intuitively.

**John Von Neumann:**
*Young man, in mathematics you don’t understand things. You just get used to them.*