Today

- Randomized Median Finding
Problem. Given a set of numbers $S = \{a_1, \ldots, a_n\}$ the median is the number in the middle if the numbers were sorted.

- If $n$ is odd then $k$th smallest element where $k = (n + 1)/2$.
- If $n$ is even then $k$th smallest element where $k = n/2$. 
Median Find

Problem. Given a set of numbers $S = \{a_1, \ldots, a_n\}$ the median is the number in the middle if the numbers were sorted.

- If $n$ is odd then $k$th smallest element where $k = (n + 1)/2$.
- If $n$ is even then $k$th smallest element where $k = n/2$.

Deterministic algorithm?
Problem. Given a set of numbers $S = \{a_1, \ldots, a_n\}$ the median is the number in the middle if the numbers were sorted.

- If $n$ is odd then $k$th smallest element where $k = (n + 1)/2$.
- If $n$ is even then $k$th smallest element where $k = n/2$.

Deterministic algorithm?

- Sort numbers, take $k$th smallest.
Problem. Given a set of numbers $S = \{a_1, \ldots, a_n\}$ the median is the number in the middle if the numbers were sorted.

- If $n$ is odd then $k$th smallest element where $k = (n + 1)/2$.
- If $n$ is even then $k$th smallest element where $k = n/2$.

Deterministic algorithm?

- Sort numbers, take $k$th smallest.
- $O(n \log n)$. 
More generally

**Problem.** Given a set of numbers $S = \{a_1, \ldots, a_n\}$ and number $k$, return $k$th smallest number. (Assume no duplicates)
More generally

**Problem.** Given a set of numbers $S = \{a_1, \ldots, a_n\}$ and number $k$, return the $k$th smallest number. (Assume no duplicates)

Special cases:

- $k = 1$: minimum element
- $k = n$: maximum element
Problem. Given a set of numbers $S = \{a_1, \ldots, a_n\}$ and number $k$, return $k$th smallest number. (Assume no duplicates)

Special cases:

- $k = 1$: minimum element $O(n)$
- $k = n$: maximum element $O(n)$. 

Why is it $O(n \log n)$ for $k = n/2$?
More generally

**Problem.** Given a set of numbers $S = \{a_1, \ldots, a_n\}$ and number $k$, return $k$th smallest number. (Assume no duplicates)

Special cases:

- $k = 1$: minimum element $O(n)$
- $k = n$: maximum element $O(n)$.

Why is it $O(n \log n)$ for $k = n/2$?
Divide and Conquer Algorithm

- Choose splitter (or pivot) \( a_i \in S \)
Divide and Conquer Algorithm

- Choose splitter (or pivot) $a_i \in S$
- Form sets $S^- = \{a_j : a_j < a_i\}$, $S^+ = \{a_j : a_j > a_i\}$. 
Divide and Conquer Algorithm

- Choose splitter (or pivot) \( a_i \in S \)
- Form sets \( S^- = \{ a_j : a_j < a_i \} \), \( S^+ = \{ a_j : a_j > a_i \} \).

If:
- \( |S^-| = k - 1 \): \( a_i \) is the target.
Divide and Conquer Algorithm

- Choose splitter (or pivot) \( a_i \in S \)
- Form sets \( S^- = \{ a_j : a_j < a_i \} \), \( S^+ = \{ a_j : a_j > a_i \} \).

If:

- \( |S^-| = k - 1 \): \( a_i \) is the target.
- \( |S^-| \geq k \): recurse on \((S^-, k)\).
Divide and Conquer Algorithm

- Choose splitter (or pivot) $a_i \in S$
- Form sets $S^- = \{a_j : a_j < a_i\}$, $S^+ = \{a_j : a_j > a_i\}$.

If:

- $|S^-| = k - 1$: $a_i$ is the target.
- $|S^-| \geq k$: recurse on $(S^-, k)$.
- $|S^-| < k - 1$, recurse on $(S^+, k - (|S^-| + 1))$. 
Pseudocode

\textbf{SELECT}(S,k):

Choose splitter $a_i \in S$.

\textbf{for} each $a_j \in S$ \textbf{do}

- Put $a_j \in S^-$ if $a_j < a_i$.
- Put $a_j \in S^+$ if $a_j > a_i$.

\textbf{end for}

If $|S^-| = k - 1$, then return $a_i$.

If $|S^-| \geq k$, return \textbf{SELECT}($S^-$, $k$).

Else, return \textbf{SELECT}($S^+$, $k - (|S^-| + 1)$).
**Pseudocode**

\[ \text{SELECT}(S, k): \]

Choose splitter \( a_i \in S \).

**for** each \( a_j \in S \) **do**

- Put \( a_j \in S^- \) if \( a_j < a_i \).
- Put \( a_j \in S^+ \) if \( a_j > a_i \).

**end for**

If \( |S^-| = k - 1 \), then return \( a_i \).
If \( |S^-| \geq k \), return \( \text{SELECT}(S^-, k) \).
Else, return \( \text{SELECT}(S^+, k - (|S^-| + 1)) \).

Looks kind of like quicksort...
**Pseudocode**

\[ \text{SELECT}(S,k): \]

Choose splitter \( a_i \in S \).

**for** each \( a_j \in S \) **do**

\( \text{Put } a_j \in S^{-} \text{ if } a_j < a_i. \)

\( \text{Put } a_j \in S^{+} \text{ if } a_j > a_i. \)

**end for**

If \( |S^{-}| = k - 1 \), then return \( a_i \).

If \( |S^{-}| \geq k \), return \( \text{SELECT}(S^{-}, k) \).

Else, return \( \text{SELECT}(S^{+}, k - (|S^{-}| + 1)) \).

Looks kind of like quicksort...

**Fact.** Algorithm is correct.
How to choose splitter?

We want recursive calls to work on much smaller sets.

- Best case, splitter is the median:

  \[ T(n) \leq T(n/2) + cn \Rightarrow O(n) \text{ runtime} \]
How to choose splitter?

We want recursive calls to work on much smaller sets.

- Best case, splitter is the median:
  \[ T(n) \leq T(n/2) + cn \Rightarrow O(n) \text{ runtime} \]

- Worst case, splitter is largest element:
  \[ T(n) \leq T(n - 1) + cn \Rightarrow O(n^2) \text{ runtime} \]
How to choose splitter?

We want recursive calls to work on much smaller sets.

- **Best case, splitter is the median:**
  \[
  T(n) \leq T(n/2) + cn \Rightarrow O(n) \text{ runtime}
  \]

- **Worst case, splitter is largest element:**
  \[
  T(n) \leq T(n - 1) + cn \Rightarrow O(n^2) \text{ runtime}
  \]

- **Middle case, splitter separates }n \text{ elements**}
  \[
  T(n) \leq T((1 - \epsilon)n) + cn
  \]
  \[
  T(n) \leq cn \left[1 + (1 - \epsilon) + (1 - \epsilon)^2 + \ldots\right] \leq \frac{cn}{\epsilon}
  \]

How can we stay close to the best case?
Randomized Splitters

**Idea.** Choose splitter uniformly at random.
Randomized Splitters

**Idea.** Choose splitter uniformly at random.

**Analysis.** Phase $j$ when $n(3/4)^{j+1} \leq |S| \leq n(3/4)^j$.

- **Claim.** Expect to stay in phase $j$ for two rounds.
Randomized Splitters

**Idea.** Choose splitter uniformly at random.

**Analysis.** Phase $j$ when $n(3/4)^{j+1} \leq |S| \leq n(3/4)^j$.

- **Claim.** Expect to stay in phase $j$ for two rounds.
  - Call splitter *central* if separates $1/4$ fraction of elements.
Randomized Splitters

**Idea.** Choose splitter uniformly at random.

**Analysis.** Phase $j$ when $n(3/4)^{j+1} \leq |S| \leq n(3/4)^j$.

- **Claim.** Expect to stay in phase $j$ for two rounds.
  - Call splitter *central* if separates $1/4$ fraction of elements.
  - $\Pr[\text{central splitter}] = 1/2$. 
Randomized Splitters

**Idea.** Choose splitter uniformly at random.

**Analysis.** Phase $j$ when $n(3/4)^{j+1} \leq |S| \leq n(3/4)^j$.

- **Claim.** Expect to stay in phase $j$ for two rounds.
  - Call splitter *central* if separates $1/4$ fraction of elements.
  - $\Pr[\text{central splitter}] = 1/2$.
  - If $X$ is number of attempts until central splitter,
Randomized Splitters

**Idea.** Choose splitter uniformly at random.

**Analysis.** Phase $j$ when $n(3/4)^{j+1} \leq |S| \leq n(3/4)^j$.

- **Claim.** Expect to stay in phase $j$ for two rounds.
  - Call splitter *central* if separates $1/4$ fraction of elements.
  - $\Pr[central \ splitter] = 1/2$.
  - If $X$ is number of attempts until central splitter,
    
    $$E[X] = \sum_{j=1}^{\infty} j \Pr[X = j] = \sum_{j=1}^{\infty} j p(1 - p)^{j-1}$$
    
    $$= \frac{p}{1 - p} \sum_{j=1}^{\infty} j(1 - p)^j = \frac{p}{1 - p} \frac{(1 - p)}{p^2}$$
    
    $$= \frac{1}{p}$$
Analysis

- Let $Y$ be a r.v. equal to number of steps of the algorithm

\[ Y = Y_0 + Y_1 + Y_2 + \ldots \]

where $Y_j$ is steps in phase $j$.

- One iteration in phase $j$ takes $cn(3/4)^j$ steps.

- $E[Y_j] \leq 2cn(3/4)^j$ since expect two iterations.

- $E[Y] = \sum_j E[Y_j] \leq \sum_j 2cn(3/4)^j \leq 8cn$.

Theorem

Expected running time of Select $(n,k)$ is $O(n)$. 
Analysis

- Let $Y$ be a r.v. equal to number of steps of the algorithm
- $Y = Y_0 + Y_1 + Y_2 + \ldots$ where $Y_j$ is steps in phase $j$
Let $Y$ be a r.v. equal to number of steps of the algorithm

- $Y = Y_0 + Y_1 + Y_2 + \ldots$ where $Y_j$ is steps in phase $j$
- One iteration in phase $j$ takes $cn(3/4)^j$ steps.
Let $Y$ be a r.v. equal to number of steps of the algorithm.

$Y = Y_0 + Y_1 + Y_2 + \ldots$ where $Y_j$ is steps in phase $j$

One iteration in phase $j$ takes $cn(3/4)^j$ steps.

$E[Y_j] \leq 2cn(3/4)^j$ since expect two iterations.
Let $Y$ be a r.v. equal to number of steps of the algorithm

$Y = Y_0 + Y_1 + Y_2 + \ldots$ where $Y_j$ is steps in phase $j$

One iteration in phase $j$ takes $cn(3/4)^j$ steps.

$E[Y_j] \leq 2cn(3/4)^j$ since expect two iterations.

\[
E[Y] = \sum_j E[Y_j] \leq \sum_j 2cn(3/4)^j
\]

\[
= 2cn \sum_j (3/4)^j \leq 8cn
\]
Analysis

Let $Y$ be a r.v. equal to number of steps of the algorithm

$Y = Y_0 + Y_1 + Y_2 + \ldots$ where $Y_j$ is steps in phase $j$

One iteration in phase $j$ takes $cn(3/4)^j$ steps.

$\mathbb{E}[Y_j] \leq 2cn(3/4)^j$ since expect two iterations.

$$
\mathbb{E}[Y] = \sum_j \mathbb{E}[Y_j] \leq \sum_j 2cn(3/4)^j
$$

$$
= 2cn \sum_j (3/4)^j \leq 8cn
$$

Theorem

*Expected running time of $\text{SELECT}(n,k)$ is $O(n)$.*
Applications

- Randomized median find in expected linear time
Applications

- Randomized median find in expected linear time

Quicksort (Sketch)
- Choose pivot at random from $S^-, S^+$
- Recursively sort both
- Concatenate together
Applications

- Randomized median find in expected linear time

Quicksort (Sketch)

- Choose pivot at random form $S^-$, $S^+$
- Recursively sort both
- Concatenate together

**Theorem.** Quicksort has expected $O(n \log n)$ time.