Plan

* Problem Description
  - Median & k\textsuperscript{th} element
  - Algorithms to beat

* Announcements

* Random Pivot
  - Failure of Deterministic Pivot
  - Expected Running Time
Finding The Median of a List

Given: a list $L$ of $n$ integers

Task: Return the median of the list

Median: element $m$ s.t. for
- half of $i \in L$, $i \leq m$
- half of $j \in L$, $m < j$
Finding The Median of a List

Given: a list $L$ of $n$ integers
Task: Return the median of the list

Median: element $m$ s.t. for
- half of $i \in L$, $i \leq m$
- half of $j \in L$, $m < j$

If $L$ was sorted
Finding The Median of a List

Given: a list $L$ of $n$ integers.
Task: Return the median of the list.

Median: element $m$ s.t. for

$$|\{i \in L: i \leq m\}| \leq \frac{n}{2}$$
$$|\{j \in L: j > m\}| < \frac{n}{2}$$

Median is $\lceil n/2 \rceil^{th}$ element.
Given: a list $L$ of $n$ integers
Task: Return the $k^{th}$ smallest element
Given: a list $L$ of $n$ integers
Task: Return the $k^{th}$ smallest element

- median: $\left\lceil \frac{n}{2} \right\rceil^{th}$ element
- minimum: $1^{st}$ element
- maximum: $n^{th}$ element
- $80^{th}$ percentile: $\left\lfloor \frac{80 \cdot n}{100} \right\rfloor^{th}$ element
**$k^{th}$ element**.

Given: a list $L$ of $n$ integers

Task: Return the $k^{th}$ smallest element

**First Algorithm**.

Select By Sorting ($L, k$).
- Sort $L$
- Return $k^{th}$ element
**k^{th} element**

Given: a list $L$ of $n$ integers

Task: Return the $k^{th}$ smallest element

**First Algorithm.**

Select By Sorting $(L, k)$.
- Sort $L$
- Return $k^{th}$ element

**Correctness.** By def/construction.

**Running Time.** $O(n \log n)$. 

**Can we do better?**
Announcements.

* Prelim #1 Grades
  → To be released before next lecture
  → Feb break slowed us down. Apologies.

* HW3 Grades returned soon (~Friday)

* RESEARCH NIGHT
  4 March 5-7pm
  Gates 401
  Food provided.
Selection without Sorting

Divide

Choose a "pivot" p.

Partition L into

\[ L_{\leq} = \{ i \in L : i \leq p \} \quad \quad L > = \{ j \in L : j > p \} \]

Conquer

Recurse on correct part.
Selection without Sorting

Divide

Choose a "pivot" \( p \).

\[ \downarrow \text{partition around } p \]

\[ L \leq x \quad x \quad x \quad L > \]
Selection without Sorting

Divide

Choose a "pivot" $p$

↓ partition around $p$

Conquer

To find $k^{th}$ element, consider $l = |L \leq |$ compared to $k$. 
\[ u = 3^{rd} \]

\[ \text{ie } L: i \leq p \]

\[ 6 \quad 21 \quad 3 \]

\[ 2 \quad 7 \quad 9 \quad 10 \quad 12 \quad 24 \]

\[ 24 \quad 9 \quad 10 \quad 12 \quad 7 \]
Select \((L, k)\)

Choose pivot \(p \in L\).
\[L_\leq \leftarrow \{ i \in L : i \leq p \} \]
\[L_\rangle \leftarrow \{ j \in L : j > p \} \]

let \(l = |L_\leq|\)

if \(l = k\) : Return \(p\) // pivot was \(k^{th}\) element

if \(l > k\) : Return Select \((L_\leq, k)\)

else : Return Select \((L_\rangle, k-l)\)
**Select** \((L, k)\)

Choose pivot \(p \in L\).

\[
L_{\leq} \leftarrow \{i \in L : i \leq p\} \\
L_{>} \leftarrow \{j \in L : j > p\}
\]

let \(l = |L_{\leq}|\)

if \(l = k\) : Return \(p\) // pivot was \(k^{th}\) elem

if \(l > k\) : Return Select \((L_{\leq}, k)\)

else : Return Select \((L_{>}, k-l)\)

\[T(|L_{\leq}|) \text{ or } T(|L_{>}|)\]
Which Pivot?

Suppose we choose the pivot to always be $L[1]$.

What is the worst-case running time?
Which Pivot?

Suppose we choose the pivot to always be $L[1]$.

What is the worst-case Running Time?

Select $(L, n)$

$\langle 1, 2, 3, 4, \ldots, n \rangle$

$L \leftarrow \langle 1 \rangle$

$L \rightarrow \langle 2, 3, 4, \ldots, n \rangle$
Which Pivot?

Suppose we choose the pivot to always be $L[1]$.

What is the worst-case running time?

Select $(L, n)$

\[
\langle 1, 2, 3, 4, \ldots, n \rangle
\]

\[
T(n) = c \cdot n + T(n-1)
\]
Which Pivot?

Suppose we choose the pivot to always be L[1].

What is the worst-case Running Time?

Select (L, n)

\[ \langle 1, 2, 3, 4, \ldots, n \rangle \]

\[ T(n) = c \cdot n + T(n-1) \]

\[ = \sum_{j=0}^{n-1} c \cdot (n-j) \]

\[ = \Omega(n^2) \]
Which Pivot?

Suppose we choose the pivot to always be $L[n]$. What is the worst-case Running Time?
Which Pivot?

Suppose we choose the pivot to always be $L[n]$.

What is the worst-case running time?

Select $(L, 1)$

\[
\langle 1, 2, 3, 4, \ldots, n \rangle
\]

\[
\langle 1, 2, \ldots, n-1 \rangle
\]

\[
\langle n \rangle
\]
Which Pivot?

Suppose we choose the pivot to always be $L[n]$.

What is the worst-case Running Time?

Select $(L, 1)$

$$\langle 1, 2, 3, 4, \ldots, n \rangle$$

$$T(n) = c \cdot n + T(n-1)$$

$$= \sum_{j=0}^{n-1} c \cdot (n-j)$$

$$= \Omega(n^2).$$
Theorem: For any deterministic pivot selection (that does not depend on $L$), the worst-case running time of Select is $\Omega(n^2)$. 
Theorem: For any deterministic pivot selection (that does not depend on L), the worst-case running time of Select is $\Theta(n^2)$.

Pivot that depends on L

Ideal: pivot on the median

$$T(n) = c \cdot n + T(n/2) = 2c \cdot n = O(n).$$
Theorem: For any deterministic pivot selection (that does not depend on L), the worst-case running time of Select is \( \Omega(n^2) \).

Pivot that depends on L

- Ideal: pivot on the median
  \[
  T(n) = c \cdot n + T\left(\frac{n}{2}\right)
  = 2c \cdot n = \Theta(n).
  \]

- Actual: median-of-medians
  (Blum, Floyd, Pratt, Rivest, Tarjan 1973)
Theorem: For any deterministic pivot selection (that does not depend on L), the worst-case running time of Select is $\Omega(n^2)$.

Randomized Pivot

- Many pivots are good
- Upper bound the expected RT
Randomized Algorithms.

* Allow algorithm to "flip coins"
Randomized Algorithms

* Allow algorithm to "flip coins"

Basic Randomness Primitives

* Choose random bit $B \in \{0,1\}$ w.p. $1/2$
Randomized Algorithms

* Allow algorithm to "flip coins"

Basic Randomness Primitives

* Choose random bit $b \in \{0,1\}$ w.p. $\frac{1}{2}$

* Given $n$, choose $p \in \{1,\ldots,n\}$ uniformly at random

$$\Pr[p=i] = \frac{1}{n} \quad \forall i \in \{1,\ldots,n\}$$
Deterministic Algorithms

* We design an algorithm A

* Adversary choose input to A

L designed to give worst-case RT
**Deterministic Algorithms**

* We design an algorithm $A$
* Adversary choose input to $A$
  
  $L$ designed to give worst-case $RT$

**Randomized Algorithms**

* We design an algorithm $A$
* Adversary choose input to $A$
* Algorithm "flips coin" while running
  
  $L$ may give improved expected $RT$
Randomized Select \((L, k)\)

Choose pivot \(p\) \underline{Uniformly At Random} from \(L\)

\[
L_\leq \leftarrow \{i \in L : i \leq p\}
\]

\[
L_\geq \leftarrow \{j \in L : j > p\}
\]

let \(l = |L_\leq|\)

if \(l = k\) : Return \(p\) \hspace{1cm} \text{// pivot was \(k^{th}\) element}

if \(l > k\) : Return Select \((L_\leq, k)\)

else : Return Select \((L_\geq, k-l)\)
Expected Running Time Analysis

1. Define a set of “good” pivots.
   - Reduce the problem size significantly

2. Show “good” pivots occur regularly in expectation

3. By linearity of expectation
   
   Expected running time bounded in terms of expected number of pivots.