28 February 2024

Randomized Median Finding

Plan

* Problem Description
  - Median & k^{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 <u>median</u> 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

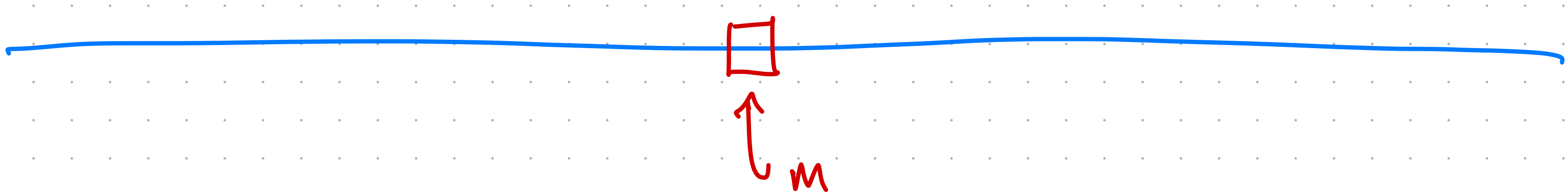
Given: a list L of n integers

Task: Return the median of the list

median: element m s.t. for

- half of $i \in L$, $i \le m$
- 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 <u>median</u> of the list

median: element m s.t. for

$$\left|\{ i \in L : i \leq m \}\right| \leq \frac{n}{2}$$

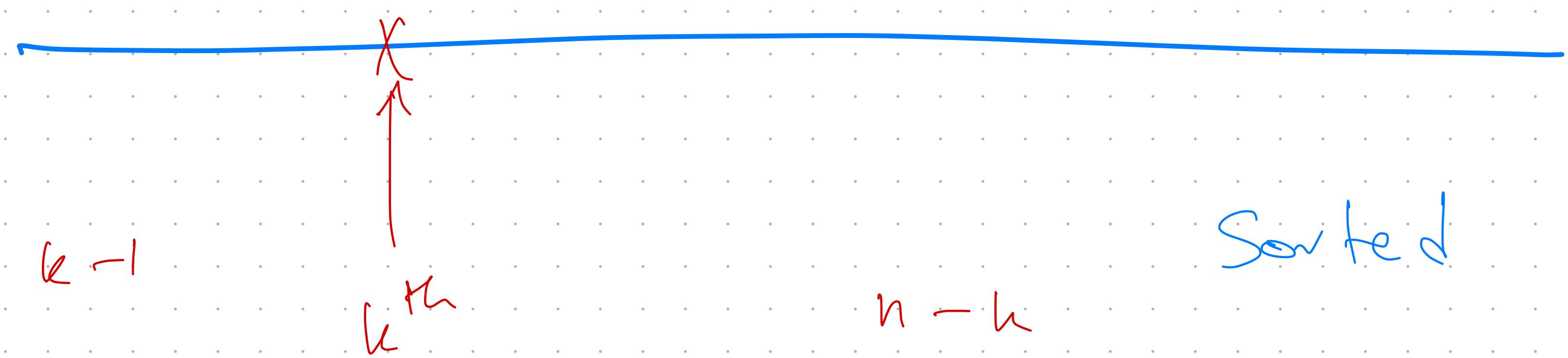
$$\left|\{ j \in L : j > m \}\right| < \frac{n}{2}$$

Median is $\lceil n/2 \rceil^{th}$ element.

k^{th} element.

Given: a list L of n integers

Task: Return the k^{th} smallest element



k^{th} element.

Given: a list L of n integers

Task: Return the k^{th} smallest element

<u>median</u>. $\lceil n/2 \rceil^{th}$ element

<u>minimum</u>. 1^{st} element

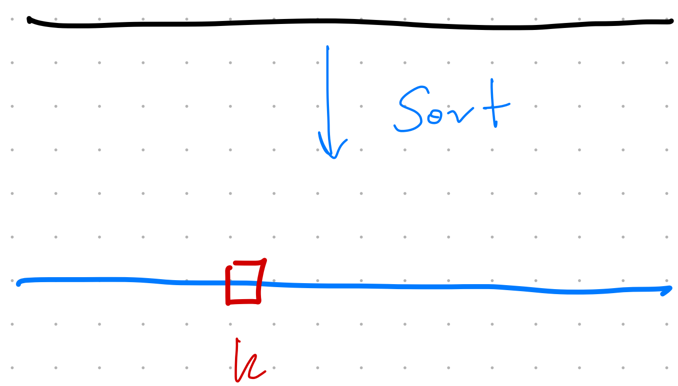
<u>maximum</u>. n^{th} element.

<u>80^{th} percentile</u>. $\left\lceil 80 \cdot n/100 \right\rceil^{th}$ element.

k^{th} element.

Given: a list L of n integers

Task: Return the k^{th} smallest element



First Algorithm.

SelectBySorting (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.

SelectBySorting(L, k).

- Sort L
- Return k^{th} element

Can we do better?

Correctness: By def/construction.

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

Announcements.

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

* HW 3 Grades returned soon (~ Friday)

* RESEARCH NIGHT 4 March 5–7pm
  Gates G01
  Food provided.

Selection without Sorting

Divide

Choose a "pivot" p.

Partition L into

$$L_{\leq} = \langle i \in L : i \leq p \rangle \qquad L_{>} = \langle j \in L : j > p \rangle$$

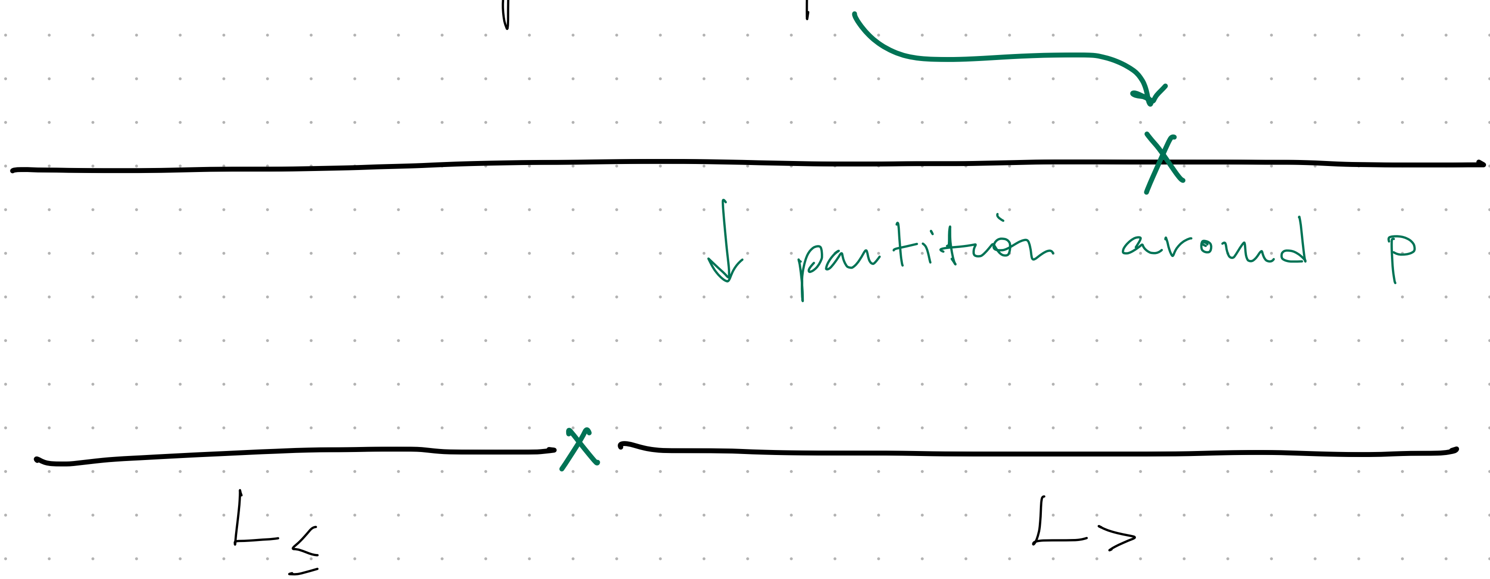
Conquer

Recurse on correct part.

Selection without Sorting

Divide

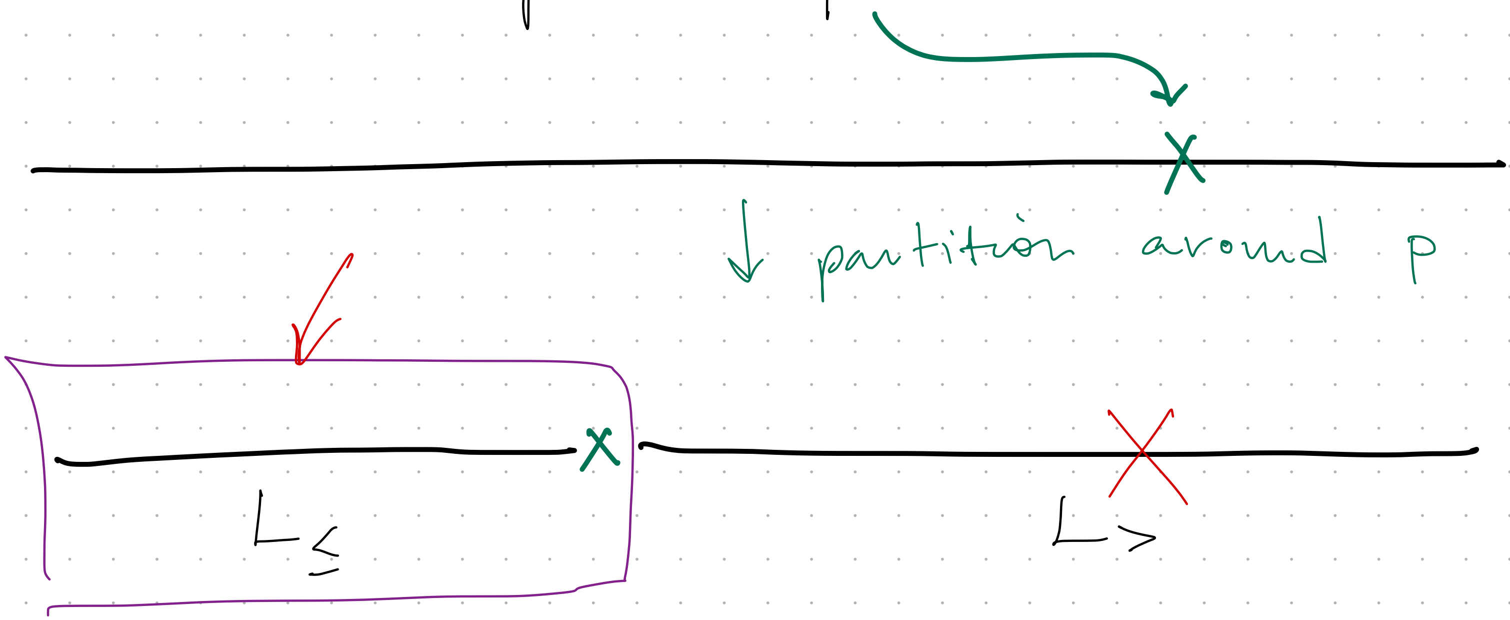
Choose a "pivot" p.



Selection without Sorting

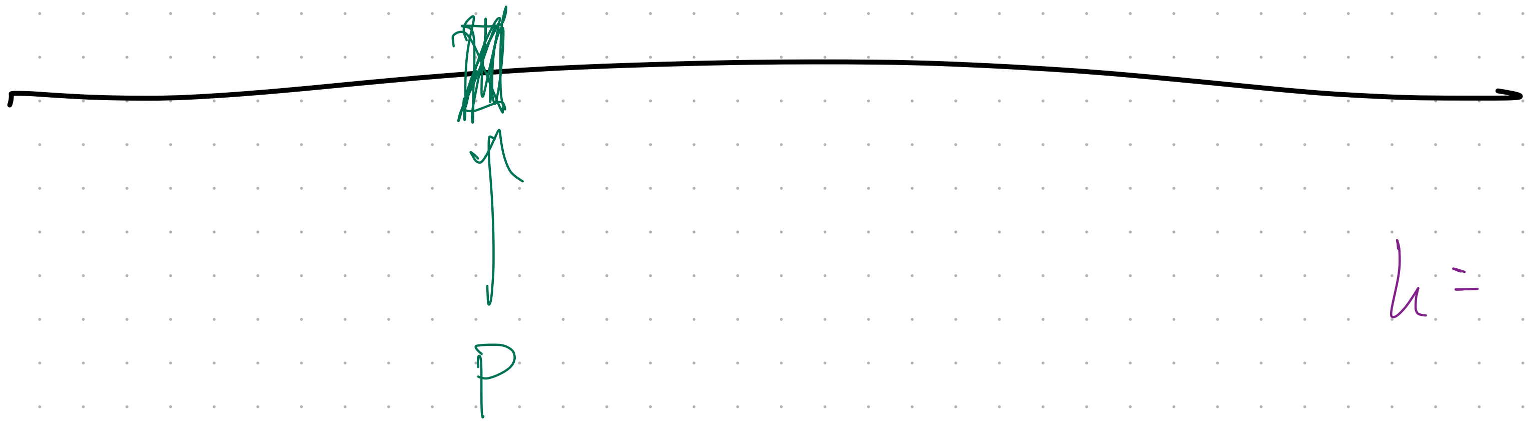
Divide

Choose a "pivot" p.



Conquer

To find k^{th} elem, consider $\ell = |L_{\leq}|$ compared to k.



$k = 3^{rd}$

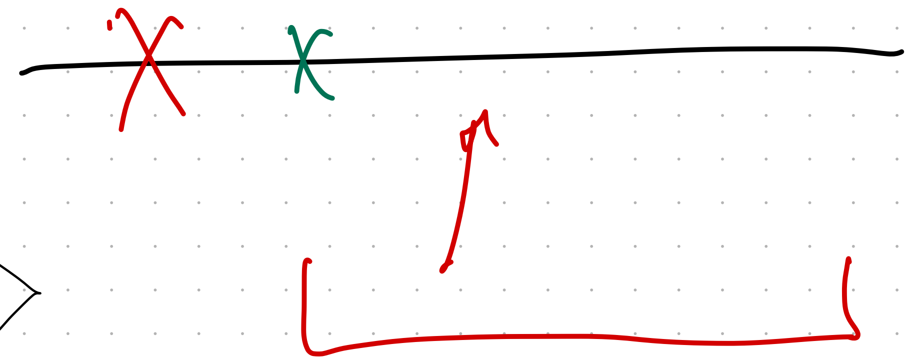
$i \in L : i \leq p$.

6 2 1 3 7 9 10 12 24

6

$\boxed{2 \; 1 \; 3 \; 6}$ 24 9 10 12 7

Select(L, k)



Choose pivot $p \in L$.

$L_{\le} \leftarrow \langle i \in L : i \le p \rangle$

$L_{>} \leftarrow \langle j \in L : j > p \rangle$

let $\ell = |L_{\le}|$

if $\ell = k$: Return p // pivot was k^{th} elem

if $\ell > k$: Return Select$(L_{\le}, k)$

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

Select(L, k)

Running Time?

Choose pivot $p \in L$.

$L_{\leq} \leftarrow \langle i \in L : i \leq p \rangle$

$L_{>} \leftarrow \langle j \in L : j > p \rangle$

Linear scan through L

$O(n)$

let $\ell = |L_{\leq}|$

if $\ell = k$: Return p // pivot was k^{th} elem

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

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

$T(|L_{\leq}|)$ 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?

$\text{Select}(L, n)$

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

$$L_{\leq} = \langle 1 \rangle \qquad L_{>} \langle 2, 3, 4, \ldots, n \rangle$$

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)

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

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

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

$cn + c(n-1) + \ldots$

$$= \Omega(n^2).$$

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$(L, 1)$

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

$$\underline{\langle 1, 2, \ldots\ldots\ldots\ n-1 \rangle} \qquad \langle n \rangle$$

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 <u>deterministic</u> 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

$$T(n) = c \cdot n + T(n/2)$$

$$= 2cn = O(n).$$

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

$$T(n) = c \cdot n + T(n/2)$$
$$= 2cn = O(n).$$

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. $1/2$

* 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
  ↳ designed to give worst-case RT

Deterministic Algorithms

* We design an algorithm A
* Adversary choose input to A

  ↳ designed to give worst-case RT

Randomized Algorithms

* We design an algorithm A
* Adversary choose input to A
* Algorithm "flips coin" while running

  ↳ may give improved expected RT

Randomized Select (L, k)

Choose pivot p Uniformly At Random from L

$L_{\leq} \leftarrow \langle i \in L : i \leq p \rangle$

$L_{>} \leftarrow \langle j \in L : j > p \rangle$

let $\ell = |L_{\leq}|$

if $\ell = k$: Return p // pivot was k^{th} elem

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

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

Expected Running Time Analysis

1. Define a set of "good" pivots.
   ↳ Reduce the problem size significantly

2. Show "good" pivots occur regularly
   <u>in expectation</u>

3. By linearity of expectation

   Expected running time bounded
   in terms of expected number
   of pivots.