Lecture: Analysis of Algorithms (CS583 - 004)

Amarda Shehu

Spring 2019

Order Statistics

Some Order Statistics We Know

Select the i^{th} smallest of n elements (the element with rank i):

- $i = 1$: minimum
- $i = n$: maximum
- $i = (n+1)/2$: median

Order Statistics

Some Order Statistics We Know

Select the i^{th} smallest of n elements (the element with rank i):

- $i = 1$: minimum
- $i = n$: maximum
- $i = (n+1)/2$: median

Design a simple algorithm to find the element with rank i

Order Statistics

Some Order Statistics We Know

Select the i^{th} smallest of n elements (the element with rank i):

- $i = 1$: minimum
- $i = n$: maximum
- $i = (n+1)/2$: median

Design a simple algorithm to find the element with rank i

Order Statistics

Some Order Statistics We Know

Select the i^{th} smallest of n elements (the element with rank i):

- $i = 1$: minimum
- $i = n$: maximum
- $i = (n+1)/2$: median

Design a simple algorithm to find the element with rank i

Naive algorithm: Sort and index i^{th} element.

Order Statistics

Some Order Statistics We Know

Select the i^{th} smallest of n elements (the element with rank i):

- $i = 1$: minimum
- $i = n$: maximum
- $i = (n+1)/2$: median

Design a simple algorithm to find the element with rank i

Naive algorithm: Sort and index i^{th} element.

$$\begin{aligned}\text{Worst-case running time} &\in \theta(n \cdot lgn) + \theta(1) \\ &\in \theta(n \cdot lgn)\end{aligned}$$

Order Statistics

Some Order Statistics We Know

Select the i^{th} smallest of n elements (the element with rank i):

- $i = 1$: minimum
- $i = n$: maximum
- $i = (n+1)/2$: median

Design a simple algorithm to find the element with rank i

Naive algorithm: Sort and index i^{th} element.

$$\begin{aligned} \text{Worst-case running time} \quad &\in \quad \theta(n \cdot lgn) + \theta(1) \\ &\in \quad \theta(n \cdot lgn) \end{aligned}$$

...use mergesort or heapsort (not quicksort)

Order Statistics

Some Order Statistics We Know

Select the i^{th} smallest of n elements (the element with rank i):

- $i = 1$: minimum
- $i = n$: maximum
- $i = (n + 1)/2$: median

Design a simple algorithm to find the element with rank i

Naive algorithm: Sort and index i^{th} element.

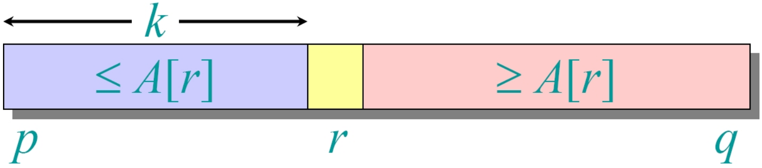
$$\begin{aligned}\text{Worst-case running time} \quad &\in \quad \theta(n \cdot lgn) + \theta(1)\\ &\in \quad \theta(n \cdot lgn)\end{aligned}$$

...use mergesort or heapsort (not quicksort)

Randomized Divide and Conquer Algorithm

RAND-SELECT(array A, p, q, i) $\triangleright i^{\mathrm{th}}$ smallest of $A[p \ldots q]$

if $p = q$ **then**
    return $A[p]$
$r \leftarrow$ RAND-PARTITION(A, p, q)
$k \leftarrow r - p + 1$ $\triangleright k = \mathrm{rank}(\mathrm{A[r]})$
if $i = k$ **then**
    return $A[r]$
if $i < k$ **then**
    return RAND-SELECT$(A, p, r-1, i)$
else return RAND-SELECT$(A, r+1, q, i-k)$



Randomized Select: Trace

Select the $i = 7^{\mathrm{th}}$ smallest element from the array below:

6	10	13	5	8	3	2	11

pivot

$i = 7$

Partition:

2	5	3	6	8	13	10	11

$k = 4$

Now select the $7 - 4 = 3^{\mathrm{rd}}$ smallest element recursively.

Randomized Select: Running Time Analysis

- Analysis follows closely that of quicksort
- For simplicity, we will assume that all elements are distinct
- We will first gain intuition through lucky/unlucky scenarios

Lucky: [assume a 1 : 9 partition after RAND-PARTITION]

$$\begin{aligned} T(n) &= T(9n/10) + \theta(n) && n^{log_{10/9}(1)} = n^0 = 1 \\ &= \theta(n) && \text{CASE 3 of master theorem} \end{aligned}$$

Unlucky: [assume one side of the partitioned array is empty]

$$\begin{aligned} T(n) &= T(n-1) + \theta(n) && \text{arithmetic series} \\ &= \theta(n^2) && \text{worse than sorting!!!} \end{aligned}$$

Randomized Select: Analysis of Expected Time

- Analysis similar to randomized quicksort
- Let $T(n)$ be the random variable for the running time of RAND-SELECT on an input of size n, assuming random numbers are independent
- To obtain upper bound, assume the i^{th} smallest element always falls on the larger side of the partition:

$$T(n) = \begin{cases} T(max\{0, n-1\}) + \theta(n) & \text{if } 0{:}n-1 \text{ split} \\ T(max\{1, n-2\}) + \theta(n) & \text{if } 1{:}n-2 \text{ split} \\ \ldots & \\ T(max\{n-1, 0\}) + \theta(n) & \text{if } n-1{:}0 \text{ split} \end{cases}$$

- Summing up we have:

$$E[T(n)] = \frac{1}{n} \sum_{k=0}^{n-1} E[T(max\{k, n-k-1\}) + \theta(n)]$$

Randomized Select: Those pesky expectations...

$$
\begin{aligned}
E[T(n)] &= \tfrac{1}{n}\textstyle\sum_{k=0}^{n-1} E[T(max\{k, n-k-1\})] + \theta(n) \\
&\qquad\qquad\qquad\qquad \textit{get } \theta(n) \textit{ outside} \\
&= \tfrac{2}{n}\textstyle\sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \theta(n) \\
&\qquad\qquad\qquad\qquad \textit{upper terms appear twice}
\end{aligned}
$$

Randomized Select: Those pesky expectations...

$$E[T(n)] = \frac{1}{n}\sum_{k=0}^{n-1} E[T(max\{k, n-k-1\})] + \theta(n)$$

get $\theta(n)$ outside

$$= \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \theta(n)$$

upper terms appear twice

Prove: $E[T(n)] \leq c \cdot n$ for $c > 0$ (c large enough for base cases)

Randomized Select: Those pesky expectations...

$$
\begin{aligned}
E[T(n)] &= \tfrac{1}{n}\sum_{k=0}^{n-1} E[T(max\{k, n-k-1\})] + \theta(n) \\
&\qquad\qquad\qquad\qquad\qquad \textit{get } \theta(n) \textit{ outside} \\
&= \tfrac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \theta(n) \\
&\qquad\qquad\qquad\qquad\qquad \textit{upper terms appear twice}
\end{aligned}
$$

Prove: $E[T(n)] \leq c \cdot n$ for $c > 0$ (c large enough for base cases)

Assume: $E[T(k)] \leq c \cdot k$, where $k < n$

Randomized Select: Those pesky expectations...

$$E[T(n)] = \frac{1}{n}\sum_{k=0}^{n-1} E[T(max\{k, n-k-1\})] + \theta(n)$$

get $\theta(n)$ outside

$$= \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \theta(n)$$

upper terms appear twice

Prove: $E[T(n)] \leq c \cdot n$ for $c > 0$ (c large enough for base cases)

Assume: $E[T(k)] \leq c \cdot k$, where $k < n$

Then: $E[T(n)] \leq \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} c \cdot k + \theta(n)$

Randomized Select: Those pesky expectations...

$$E[T(n)] = \frac{1}{n}\sum_{k=0}^{n-1} E[T(max\{k, n-k-1\})] + \theta(n)$$

get $\theta(n)$ outside

$$= \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} E[T(k)] + \theta(n)$$

upper terms appear twice

Prove: $E[T(n)] \leq c \cdot n$ for $c > 0$ (c large enough for base cases)

Assume: $E[T(k)] \leq c \cdot k$, where $k < n$

Then: $E[T(n)] \leq \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} c \cdot k + \theta(n)$

$\sum_{k=\lfloor n/2\rfloor}^{n-1} k \leq \frac{3}{8}n^2$ (exercise: show it)

Randomized Select: Those pesky expectations...

$$E[T(n)] = \frac{1}{n}\sum_{k=0}^{n-1} E[T(max\{k, n-k-1\})] + \theta(n)$$

get $\theta(n)$ outside

$$= \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \theta(n)$$

upper terms appear twice

Prove: $E[T(n)] \leq c \cdot n$ for $c > 0$ (c large enough for base cases)

Assume: $E[T(k)] \leq c \cdot k$, where $k < n$

Then: $E[T(n)] \leq \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} c \cdot k + \theta(n)$

$\sum_{k=\lfloor n/2 \rfloor}^{n-1} k \leq \frac{3}{8}n^2$ (exercise: show it)

So: $E[T(n)] \leq cn - (\frac{cn}{4} - \theta(n)) \leq cn$ if $\frac{cn}{4} - \theta(n) \geq 0$

Randomized Select: Those pesky expectations...

$$E[T(n)] = \frac{1}{n}\sum_{k=0}^{n-1} E[T(max\{k, n-k-1\})] + \theta(n)$$

get $\theta(n)$ outside

$$= \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \theta(n)$$

upper terms appear twice

Prove: $E[T(n)] \leq c \cdot n$ for $c > 0$ (c large enough for base cases)

Assume: $E[T(k)] \leq c \cdot k$, where $k < n$

Then: $E[T(n)] \leq \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} c \cdot k + \theta(n)$

$\sum_{k=\lfloor n/2 \rfloor}^{n-1} k \leq \frac{3}{8}n^2$ (exercise: show it)

So: $E[T(n)] \leq cn - (\frac{cn}{4} - \theta(n)) \leq cn$ if $\frac{cn}{4} - \theta(n) \geq 0$

Randomized Select: Those pesky expectations...

$$E[T(n)] = \frac{1}{n}\sum_{k=0}^{n-1} E[T(max\{k, n-k-1\})] + \theta(n)$$

get $\theta(n)$ outside

$$= \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \theta(n)$$

upper terms appear twice

Prove: $E[T(n)] \leq c \cdot n$ for $c > 0$ (c large enough for base cases)

Assume: $E[T(k)] \leq c \cdot k$, where $k < n$

Then: $E[T(n)] \leq \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} c \cdot k + \theta(n)$

$\sum_{k=\lfloor n/2 \rfloor}^{n-1} k \leq \frac{3}{8}n^2$ (exercise: show it)

So: $E[T(n)] \leq cn - (\frac{cn}{4} - \theta(n)) \leq cn$ if $\frac{cn}{4} - \theta(n) \geq 0$

Easy to find a large value of c such that $\frac{cn}{4}$ dominates $\theta(n)$

Randomized Select: Those pesky expectations...

$$E[T(n)] = \frac{1}{n}\sum_{k=0}^{n-1} E[T(max\{k, n-k-1\})] + \theta(n)$$

get $\theta(n)$ outside

$$= \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \theta(n)$$

upper terms appear twice

Prove: $E[T(n)] \leq c \cdot n$ for $c > 0$ (c large enough for base cases)

Assume: $E[T(k)] \leq c \cdot k$, where $k < n$

Then: $E[T(n)] \leq \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} c \cdot k + \theta(n)$

$\sum_{k=\lfloor n/2 \rfloor}^{n-1} k \leq \frac{3}{8}n^2$ (exercise: show it)

So: $E[T(n)] \leq cn - (\frac{cn}{4} - \theta(n)) \leq cn$ if $\frac{cn}{4} - \theta(n) \geq 0$

Easy to find a large value of c such that $\frac{cn}{4}$ dominates $\theta(n)$

Randomized Select: Summary

- Works fast in the average case: linear expected time
- Very simple and fast algorithm in practice
- *But,* worst-case behavior is $\theta(n^2)$
- **Question:** Is there an algorithm that runs in linear time even in the worst case?
- **Answer:** Yes - in 1973, Blum, Floyd, Pratt, and Rivest designed such an algorithm
- **Basic Idea:** Generate good pivots recursively to guarantee a good split

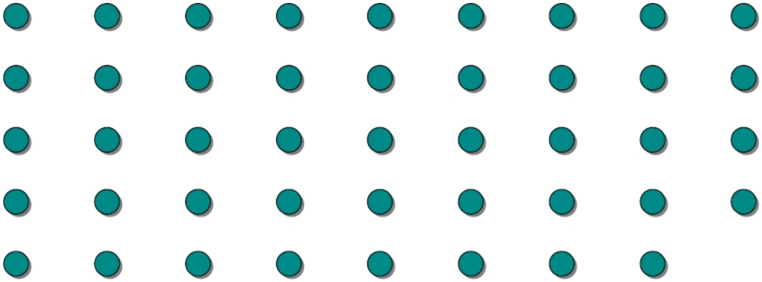
Worst-case Linear-time Order Statistics

SELECT(i,n)

1: Divide the n elements into groups of 5. Find the median of each 5-element group by rote.
2: Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot
3: Partition around the pivot. Let $k = rank(x)$
4: **if** $i = k$ **then**
5:     **return** x
6: **if** $i < k$ **then**
7:     recursively SELECT i^{th} smallest element in lower part
8: **if** $i > k$ **then**
9:     recursively SELECT $(i-k)^{\text{th}}$ smallest element in upper part

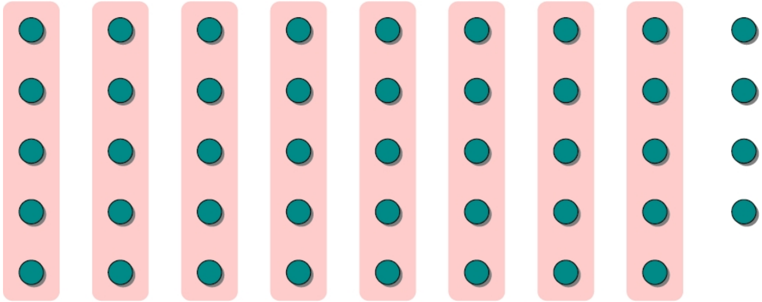
Note: lines 3.-9. are the same as in RAND-SELECT

SELECT: Choosing the Pivot



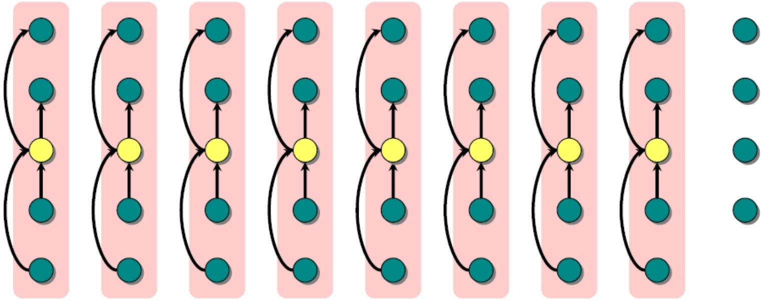
Here is the input: n elements.

SELECT: Choosing the Pivot

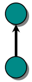


1. Divide the n elements into groups of 5.

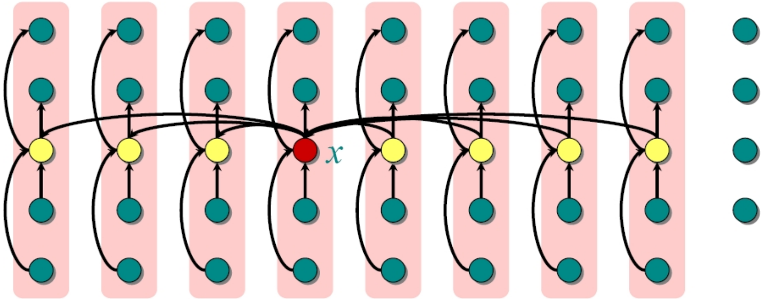
SELECT: Choosing the Pivot



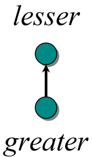
1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.



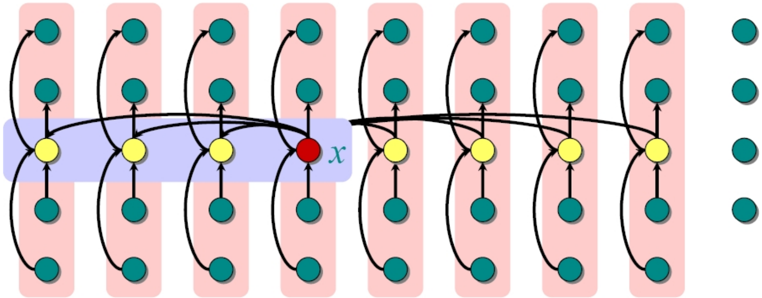
SELECT: Choosing the Pivot



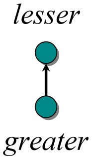
1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.
2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.



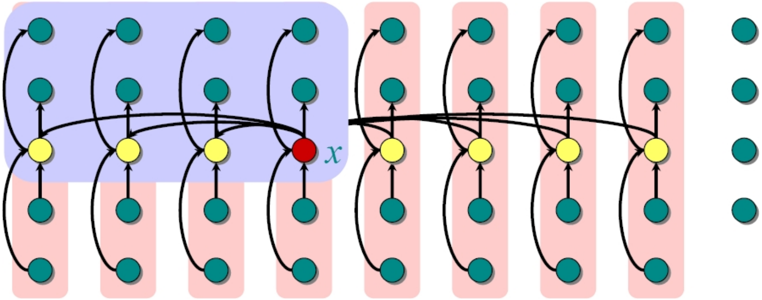
Select: Running Time Analysis



At least half of the group medians are $\leq x$,
which is at least $\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ elements.

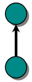


Select: Running Time Analysis

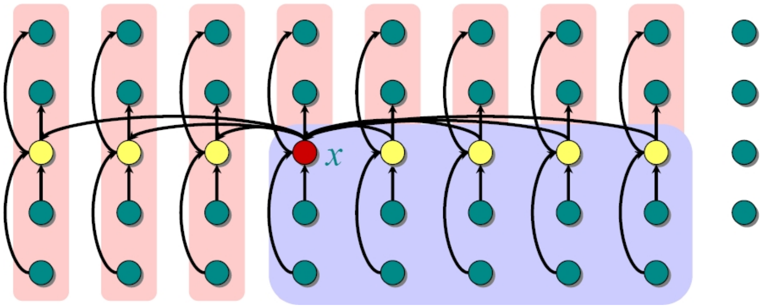


At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ group medians.

- If we assume that all elements are distinct, then there are $3\lfloor n/10 \rfloor$ elements $\leq$ x.



Select: Running Time Analysis



At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ group medians.

- If we assume that all elements are distinct, then there are $3\lfloor n/10 \rfloor$ elements $\leq$ x.
- Similarly, at least $3\lfloor n/10 \rfloor$ elements are $\geq$ x

Select: Running Time Analysis

- For $n \geq 50$, we have $3\lfloor n/10 \rfloor \geq n/4$. So, the call to SELECT in lines 4 and on is executed recursively on at most $3n/4$ elements
- The recurrence for the running time can assume that lines 4 and on takes $T(3n/4)$ in the worst case
- For $n < 50$, we know that the worst-case time is $T(n) \in \theta(1)$

The recurrence is: $T(n) = T(n/5) + \theta(n) + T(3n/4)$

Breakdown:

- **Line 1**: $\theta(n)$
- **Line 2:** $T(n/5)$
- **Line 3:** $\theta(n)$
- **Lines $\geq$ 4:** $T(3n/4)$

Substitution:

$$\begin{aligned} T(n) &\leq \tfrac{1}{5}c \cdot n + \tfrac{3}{4}c \cdot n + \theta(n) \\ &= \tfrac{19}{20}c \cdot n + \theta(n) \\ &= c \cdot n - (\tfrac{1}{20}c \cdot n - \theta(n)) \\ &\leq c \cdot n \end{aligned}$$

if c is large enough to dominate $\theta(n)$

Order Statistics: Conclusions

- Since the work at each level of recursion is a constant fraction (19/20) smaller, the work per level is a geometric series dominated by the linear work at the root
- In practice, this algorithm runs slowly, because the constant in front of n is large
- The randomized algorithm is far more practical and simpler to implement
- **Exercise:** Why not divide into groups of 3?