CMPS 6610 – Fall 2018

Order Statistics

Carola Wenk

Slides courtesy of Charles Leiserson with additions by Carola Wenk

Order statistics

Select the ith smallest of n elements (the element with ***rank*** i).

- $i = 1$: ***minimum***;
- $i = n$: ***maximum***;
- $i = \lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$: ***median***.

Naive algorithm: Sort and index ith element.

Worst-case running time $= \Theta(n \log n + 1)$

$= \Theta(n \log n)$,

using merge sort (*not* quicksort).

Randomized divide-and-conquer algorithm

RAND-SELECT(A, p, q, i) $\rhd$ i-th smallest of $A[p\,..\,q]$

if $p = q$ **then return** $A[p]$

$r \leftarrow$ RAND-PARTITION(A, p, q)

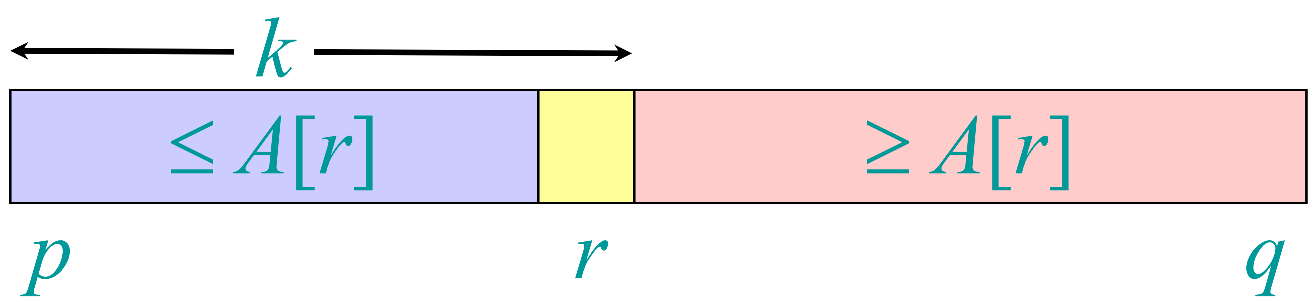
$k \leftarrow r - p + 1$ $\rhd$ $k = \mathrm{rank}(A[r])$

if $i = k$ **then return** $A[r]$

if $i < k$

then return RAND-SELECT$(A, p, r-1, i)$

else return RAND-SELECT$(A, r+1, q, i-k)$



Example

Select the $i = 7$th smallest:

6	10	13	5	8	3	2	11	$i = 7$

pivot

Partition:

2	5	3	6	8	13	10	11	$k = 4$

Select the $7 - 4 = 3$rd smallest recursively.

Intuition for analysis

(All our analyses today assume that all elements are distinct.)

Lucky:

$T(n) = T(3n/4) + dn$ — for RAND-PARTITION

$= \Theta(n)$

$n^{\log_{4/3} 1} = n^0 = 1$

CASE 3

Unlucky:

$T(n) = T(n-1) + dn$

$= \Theta(n^2)$

arithmetic series

Worse than sorting!

Analysis of expected time

- Call a pivot ***good*** if its rank lies in $[n/4, 3n/4]$.
- How many good pivots are there? $n/2$

  $\Rightarrow$ A random pivot has 50% chance of being good.
- Let $T(n,s)$ be the runtime random variable

$$T(n,s) \le \underbrace{T(3n/4,s)}_{\text{time to reduce array size to } \le 3/4n} + \underbrace{X(s)}_{\text{\#times it takes to find a good pivot}} \cdot \underbrace{dn}_{\text{Runtime of partition}}$$

Analysis of expected time

Lemma: A fair coin needs to be tossed an expected number of 2 times until the first "heads" is seen.

Proof: Let $E(X)$ be the expected number of tosses until the first "heads" is seen.

- Need at least one toss, if it's "heads" we are done.
- If it's "tails" we need to repeat (probability ½).

$$\Rightarrow E(X) = 1 + \tfrac{1}{2} E(X)$$

$$\Rightarrow E(X) = 2 \qquad \square$$

Analysis of expected time

time to reduce array size to $\leq 3/4n$

$$T(n,s) \leq T(3n/4,s) + \mathrm{X(s)}\cdot dn$$

#times it takes to find a good pivot

Runtime of partition

$$\Rightarrow E(T(n,s)) \leq E(T(3n/4,s)) + E(\mathrm{X(s)}\cdot dn)$$ *Linearity of expectation*

$$\Rightarrow E(T(n,s)) \leq E(T(3n/4,s)) + E(\mathrm{X(s)})\cdot dn$$

$$\Rightarrow E(T(n,s)) \leq E(T(3n/4,s)) + 2\cdot dn$$ *Lemma*

$$\Rightarrow T_{exp}(n) \leq T_{exp}(3n/4) + 2d\cdot n$$

$$\Rightarrow T_{exp}(n) \in \Theta(n)$$ *Master Thm (case 3)*

Summary of randomized order-statistic selection

- Works fast: linear expected time.
- Excellent algorithm in practice.
- But, the worst case is ***very*** bad: $\Theta(n^2)$.

Q. Is there an algorithm that runs in linear time in the worst case?

A. Yes, due to Blum, Floyd, Pratt, Rivest, and Tarjan [1973].

IDEA: Generate a good pivot recursively.

This algorithm has large constants though and therefore is not efficient in practice.

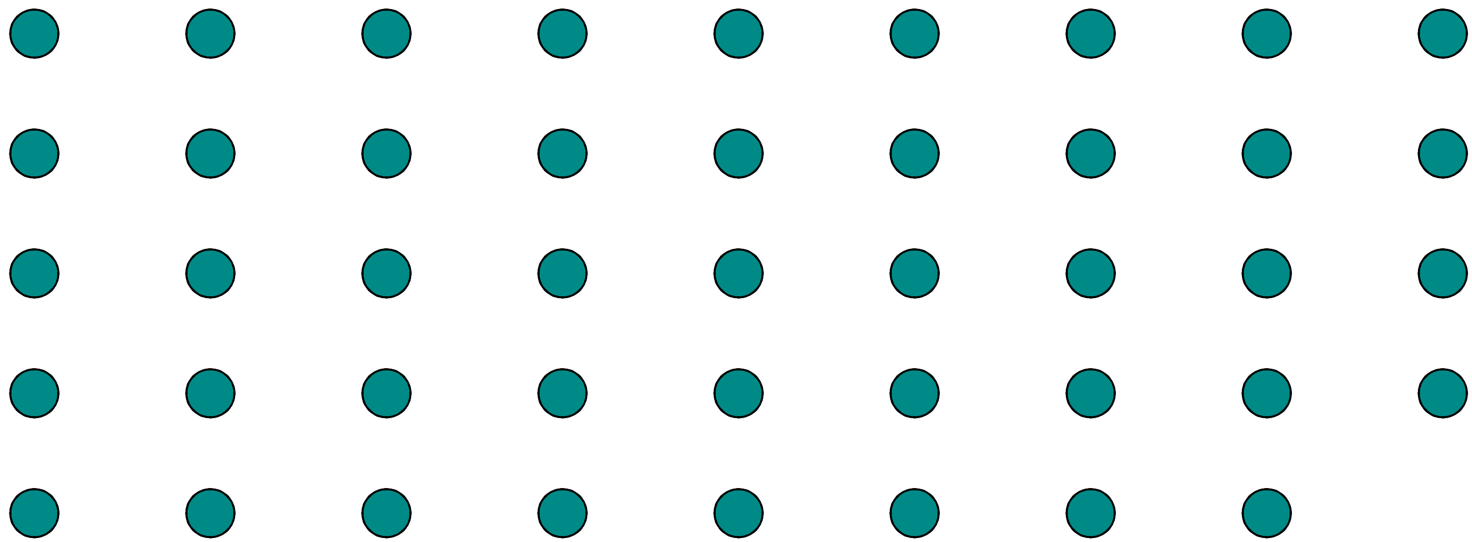
Worst-case linear-time order statistics

SELECT(i, n)

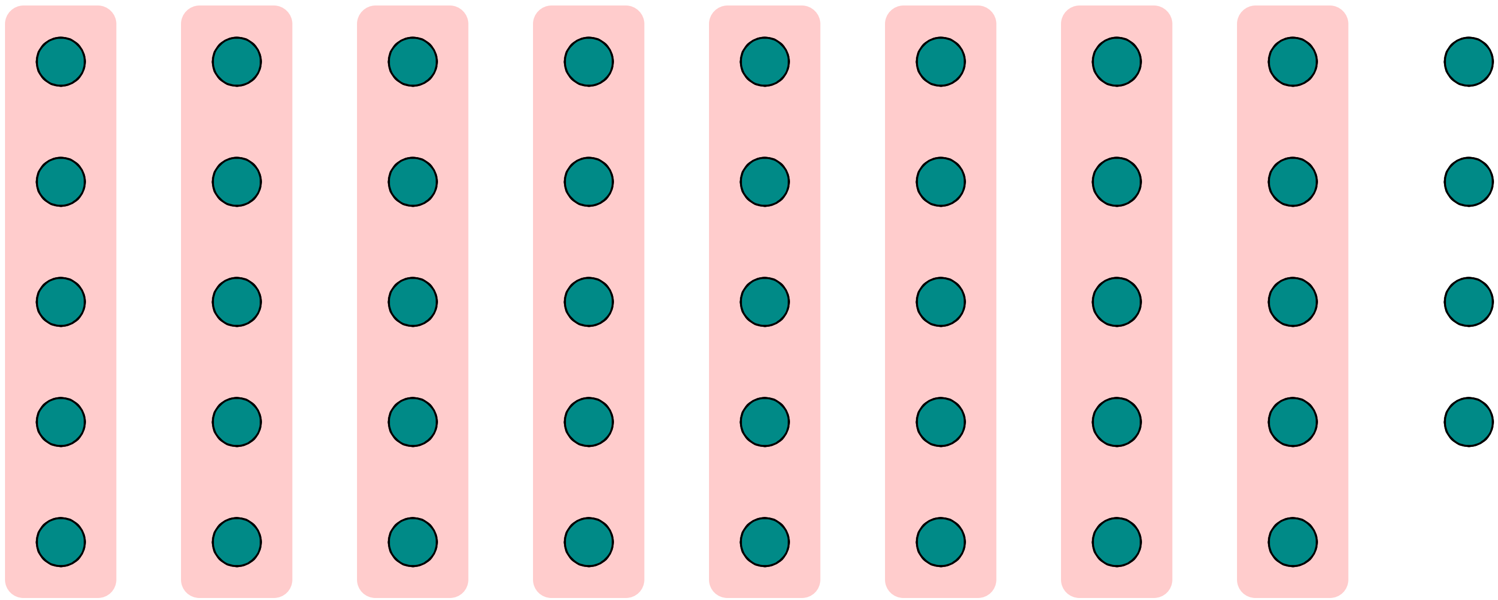
1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.
2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.
3. Partition around the pivot x. Let $k = \mathrm{rank}(x)$.
4. **if** $i = k$ **then return** x
   elseif $i < k$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

Same as RAND-SELECT

Choosing the pivot

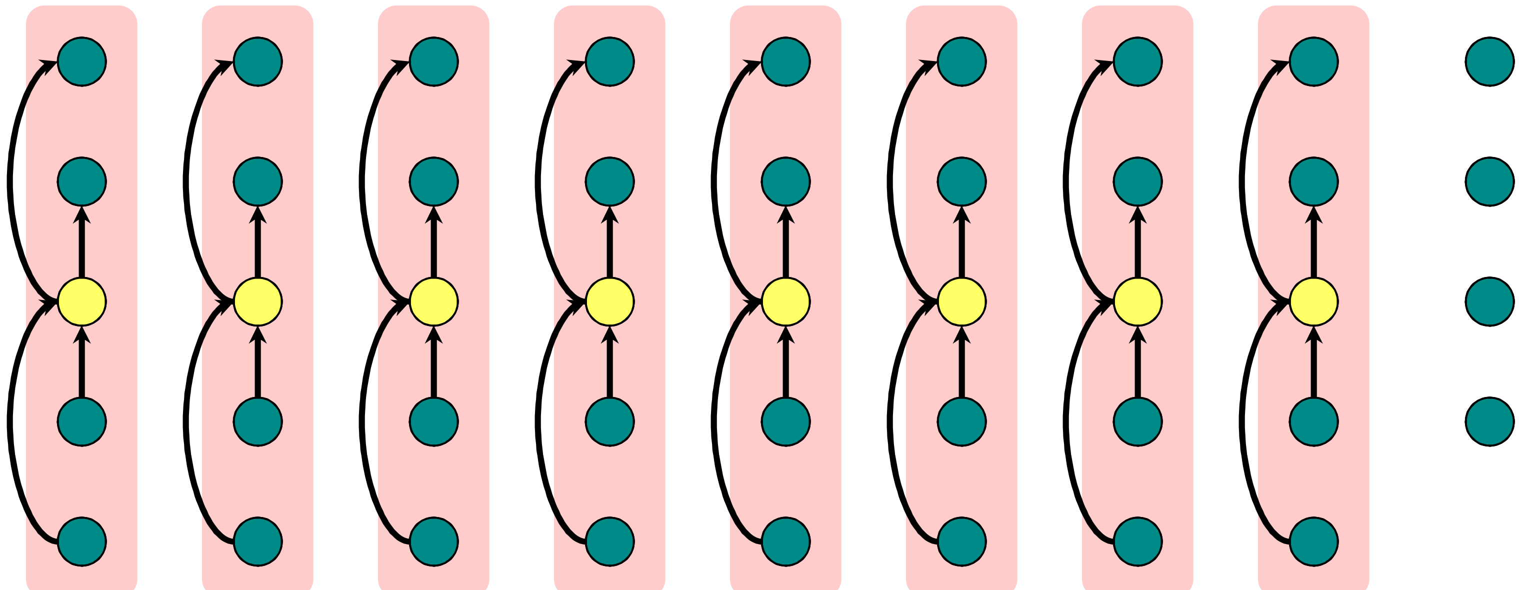


Choosing the pivot

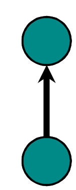


1. Divide the n elements into groups of 5.

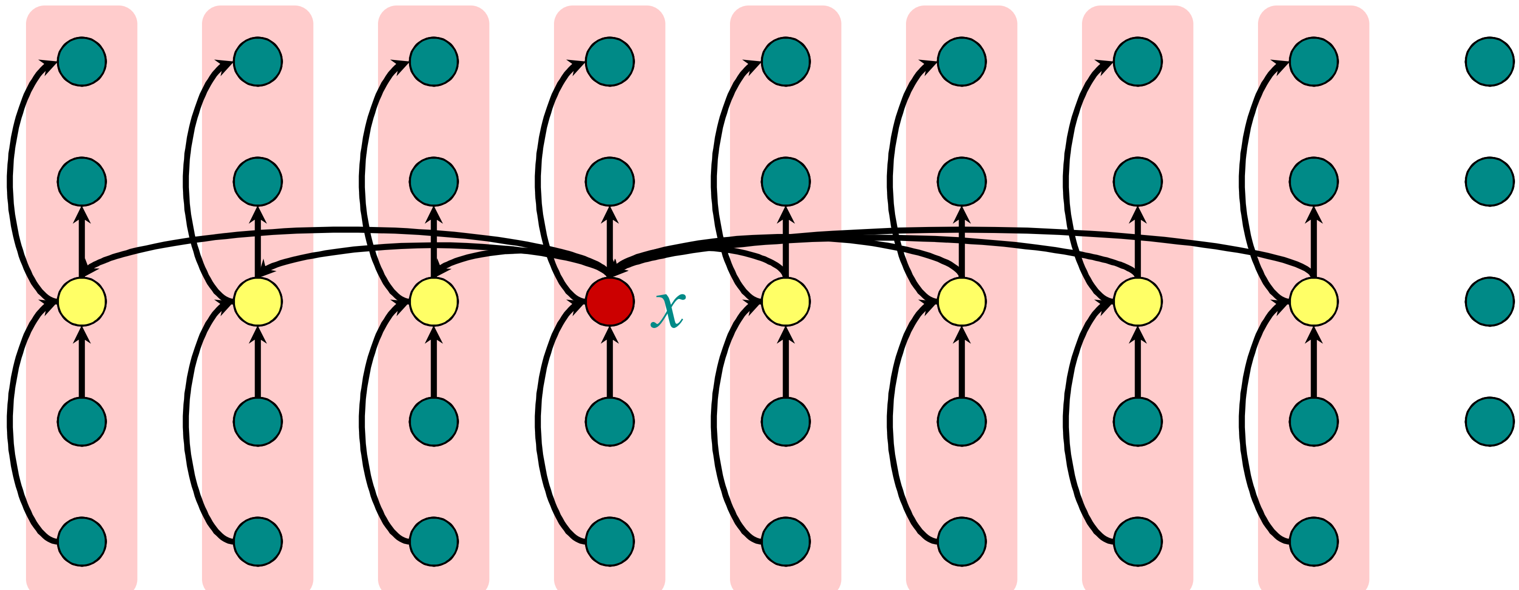
Choosing the pivot



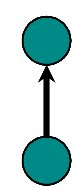
1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.



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.



Developing the recurrence

$T(n)$ — SELECT(i, n)

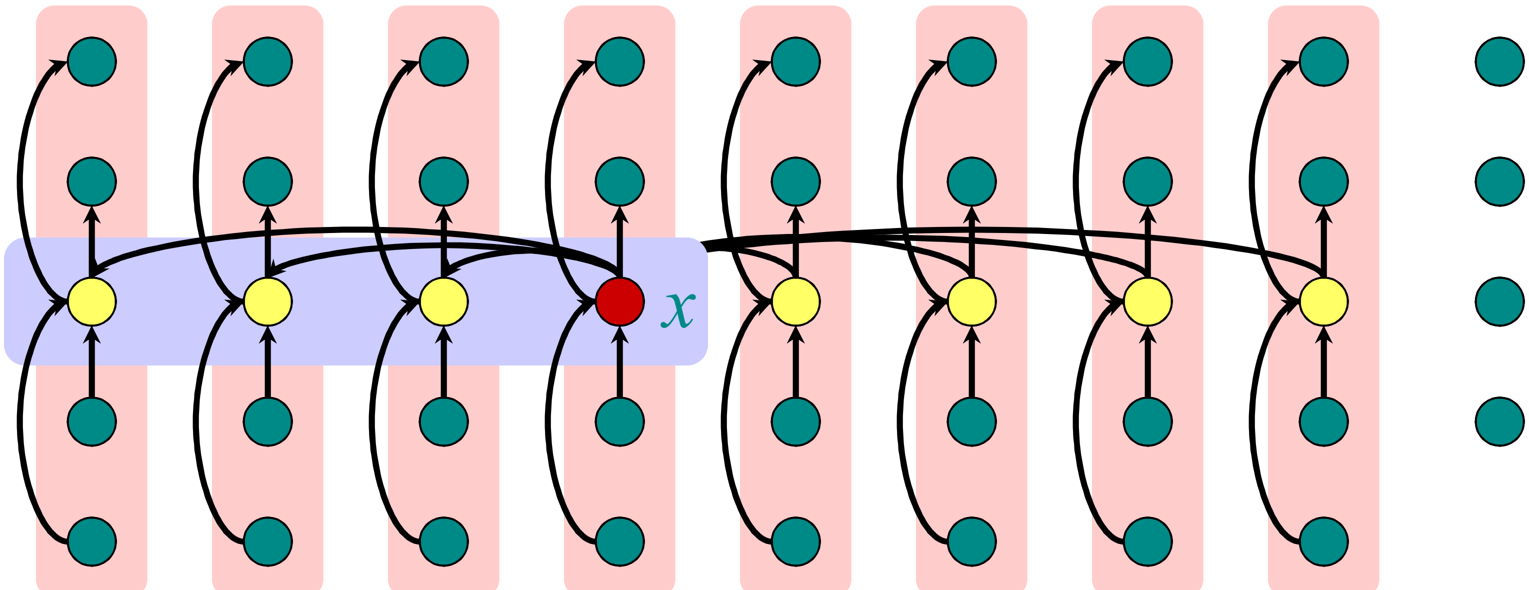
$\Theta(n)$ — 1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.

$T(n/5)$ — 2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.

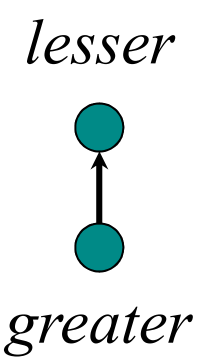
$\Theta(n)$ — 3. Partition around the pivot x. Let $k = \text{rank}(x)$.

$T(\ ?\ )$ — 4. **if** $i = k$ **then return** x
elseif $i < k$
then recursively SELECT the ith smallest element in the lower part
else recursively SELECT the $(i–k)$th smallest element in the upper part

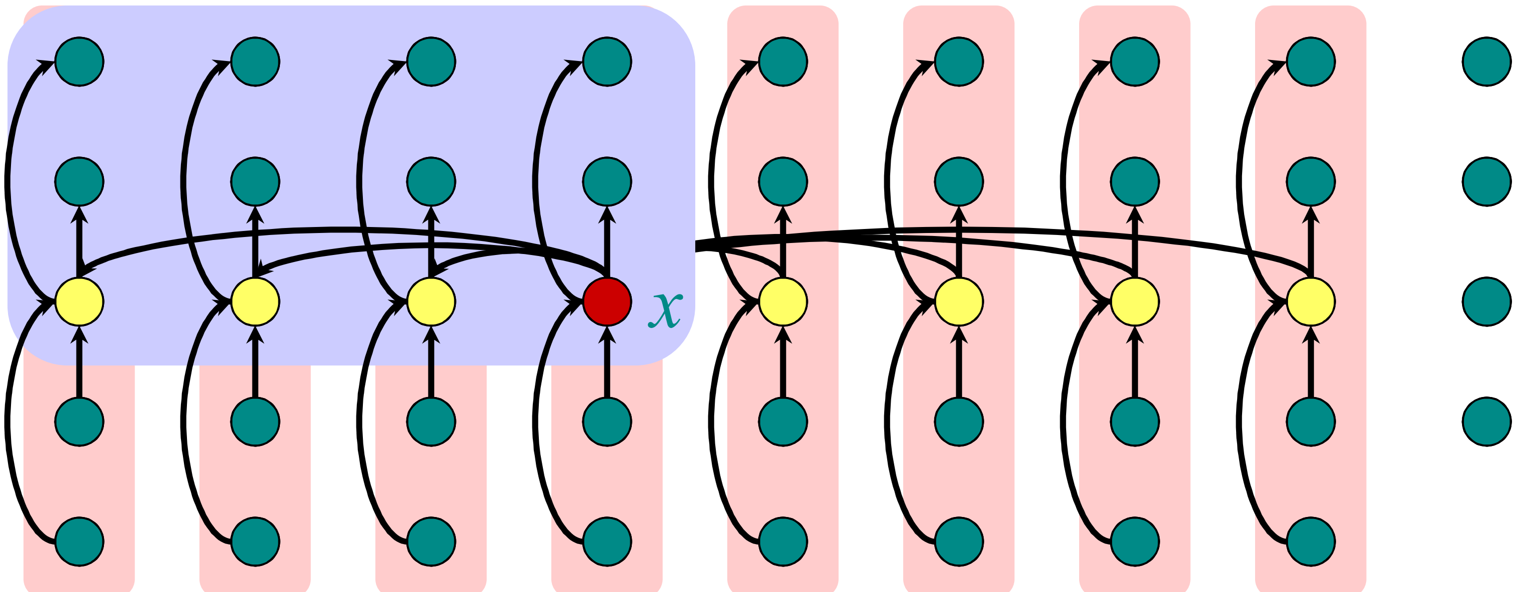
Analysis (Assume all elements are distinct.)



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.

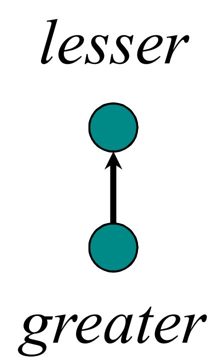


Analysis (Assume all elements are distinct.)

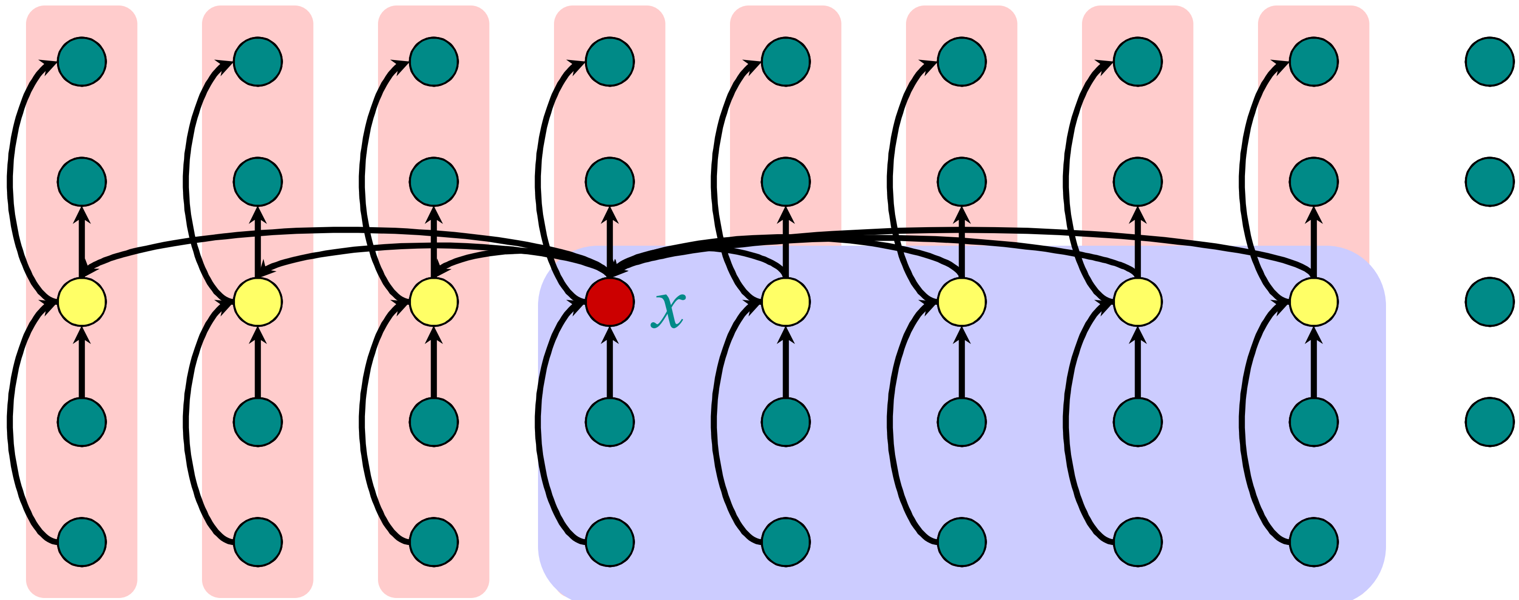


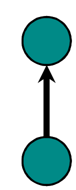
At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ group medians.

- Therefore, at least $3\lfloor n/10 \rfloor$ elements are $\leq x$.



Analysis (Assume all elements are distinct.)





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.

- Therefore, at least $3\lfloor n/10 \rfloor$ elements are $\leq x$.
- Similarly, at least $3\lfloor n/10 \rfloor$ elements are $\geq x$.

Analysis (Assume all elements are distinct.)

Need "at most" for worst-case runtime

- At least $3\lfloor n/10 \rfloor$ elements are $\leq x$
  $\Rightarrow$ at most n-$3\lfloor n/10 \rfloor$ elements are $\geq x$
- At least $3\lfloor n/10 \rfloor$ elements are $\geq x$
  $\Rightarrow$ at most n-$3\lfloor n/10 \rfloor$ elements are $\leq x$
- The recursive call to SELECT in Step 4 is executed recursively on n-$3\lfloor n/10 \rfloor$ elements.

Analysis (Assume all elements are distinct.)

- Use fact that $\lfloor a/b \rfloor \geq (a-(b-1))/b$ (page 51)
- $n-3\lfloor n/10 \rfloor \leq n-3\cdot(n-9)/10 = (10n - 3n + 27)/10 \leq 7n/10 + 3$
- The recursive call to SELECT in Step 4 is executed recursively on at most $7n/10+3$ elements.

Developing the recurrence

$T(n)$	SELECT(i, n)
$\Theta(n)$	1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.
$T(n/5)$	2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.
$\Theta(n)$	3. Partition around the pivot x. Let $k = \text{rank}(x)$.
$T(7n/10 +3)$	4. **if** $i = k$ **then return** x **elseif** $i < k$ **then** recursively SELECT the ith smallest element in the lower part **else** recursively SELECT the $(i–k)$th smallest element in the upper part

Solving the recurrence

for $\Theta(n)$

$$T(n) = T\left(\frac{1}{5}n\right) + T\left(\frac{7}{10}n + 3\right) + dn$$

Big-Oh Induction:

$T(n) \leq c(n - 3)$

Technical trick. This shows that $T(n) \in O(n)$

$$T(n) \leq c(\frac{1}{5}n - 3) + c(\frac{7}{10}n + 3 - 3) + dn$$

$$\leq \frac{9}{10}cn - 3c + dn$$

$$= c(n-3) - \frac{1}{10}cn + dn$$

$$\leq c(n-3) \quad ,$$

if c is chosen large enough, e.g., $c=10d$

Conclusions

- Since the work at each level of recursion is basically a constant fraction (9/10) 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.

Exercise: *Try to divide into groups of 3 or 7.*