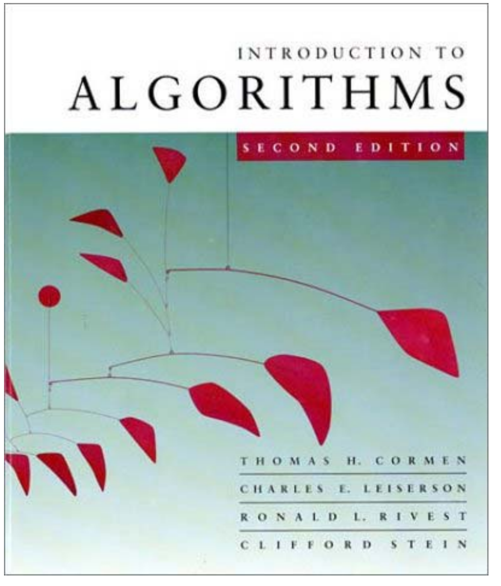
Introduction to Algorithms

6.046J/18.401J

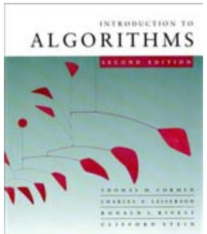


Lecture 5

Prof. Piotr Indyk

Today

- Order statistics (e.g., finding median)
- Two O(n) time algorithms:
  - Randomized: similar to Quicksort
  - Deterministic: quite tricky
- Both are examples of divide and conquer



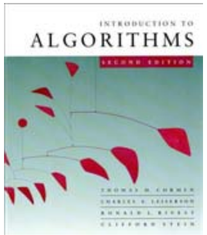
Order statistics

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***.

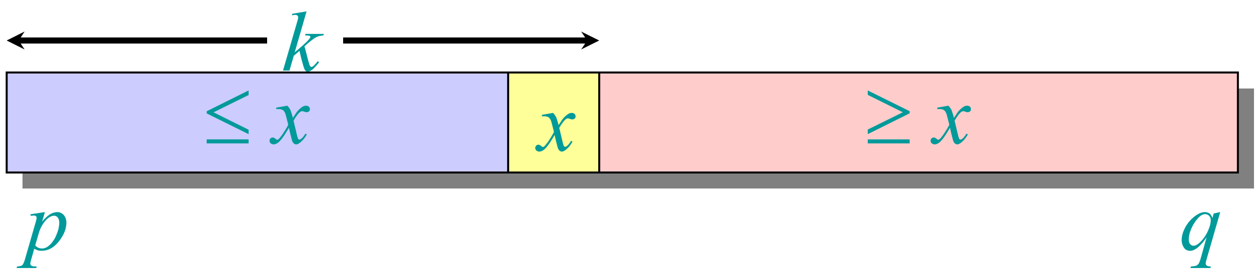
How fast can we solve the problem ?

- Min/max: O(n)
- General i : O(n log n) by sorting
- We will see how to do it in O(n) time

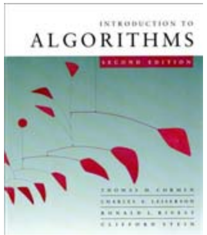


Randomized Algorithm for Finding the i^{th} element

- Divide and Conquer Approach
- Main idea: PARTITION



- If $i<k$, recurse on the left
- If $i>k$, recurse on the right
- Otherwise, output x



Randomized Divide-and-Conquer

RAND-SELECT(A, p, r, i)

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

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

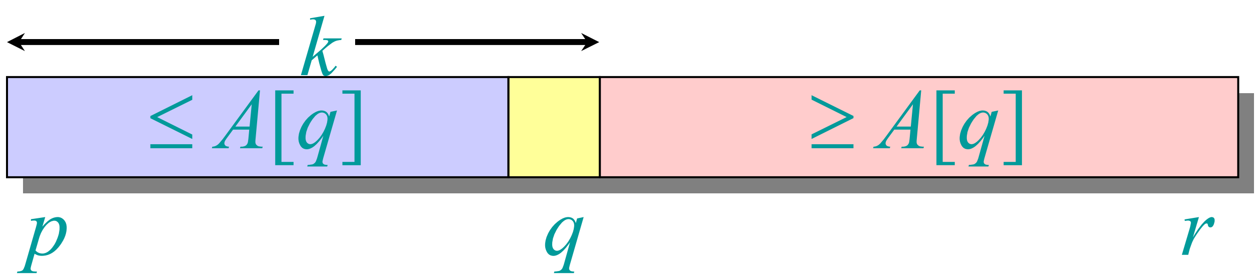
  $k \leftarrow q - p + 1$ $\quad \triangleleft\ k$=rank$(A[q])$

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

  if $i < k$

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

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



Example

Select the $i = 7$th smallest:

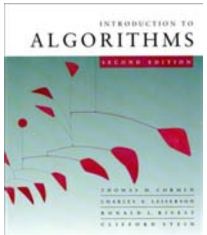
6	10	13	5	8	3	2	11	$i = 7$

pivot

Partition:

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

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



Analysis

- What is the worst-case running time ?

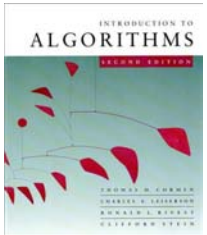
Unlucky:

$$T(n) = T(n-1) + \Theta(n) \qquad \text{arithmetic series}$$
$$= \Theta(n^2)$$

- Recall that a lucky partition splits into arrays with size ratio at most 9:1
- What if all partitions are lucky ?

Lucky:

$$T(n) = T(9n/10) + \Theta(n) \qquad n^{\log_{10/9} 1} = n^0 = 1$$
$$= \Theta(n) \qquad \text{CASE 3}$$



Expected Running Time

- The probability that a random pivot induces lucky partition is at least 8/10 (Lecture 4)
- Let t_i be the number of partitions performed between the (i-1) -th and the i-th lucky partition
- The total time is at most…

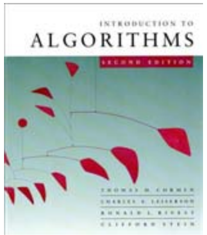
$$T = t_1\, n + t_2\, (9/10)\, n + t_3\, (9/10)^2\, n + \ldots$$

- The total ***expected*** time is at most:

$$E[T]=E[t_1]\, n + E[t_2]\, (9/10)\, n + E[t_3]\, (9/10)^2\, n + \ldots$$

$$= 10/8 * [n + (9/10)n + \ldots]$$

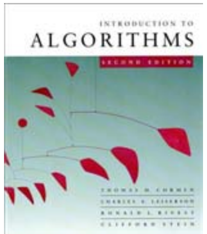
$$= O(n)$$



Digression: 9 to 1

- Do we need to define the lucky partition as 9:1 balanced ?
- No. Suffices to say that both sides have size $\geq \alpha n$, for $0 < \alpha < ½$
- Probability of getting a lucky partition is

$$1\text{-}2\alpha$$



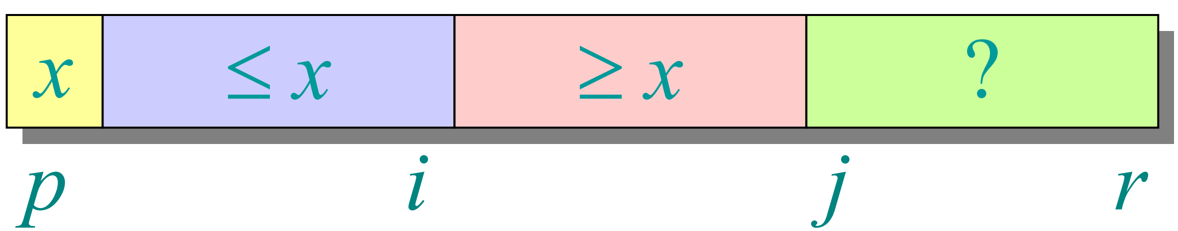
How Does it Work In Practice?

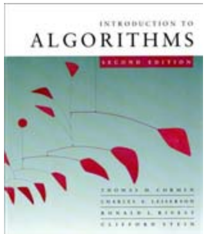
- Need 7 volunteers (a.k.a. elements)
- Will choose the median according to height

Partitioning subroutine

```
PARTITION(A, p, r)    ⊳ A[p . . r]
    x ← A[p]          ⊳ pivot = A[p]
    i ← p
    for j ← p + 1 to r
        do if A[j] ≤ x
              then i ← i + 1
                   exchange A[i] ↔ A[j]
    exchange A[p] ↔ A[i]
    return i
```

Invariant:





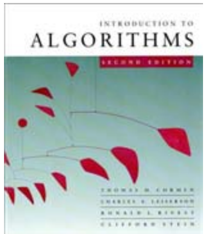
Summary of randomized order-statistic selection

- 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-Tarjan'73].

IDEA: Generate a good pivot recursively.



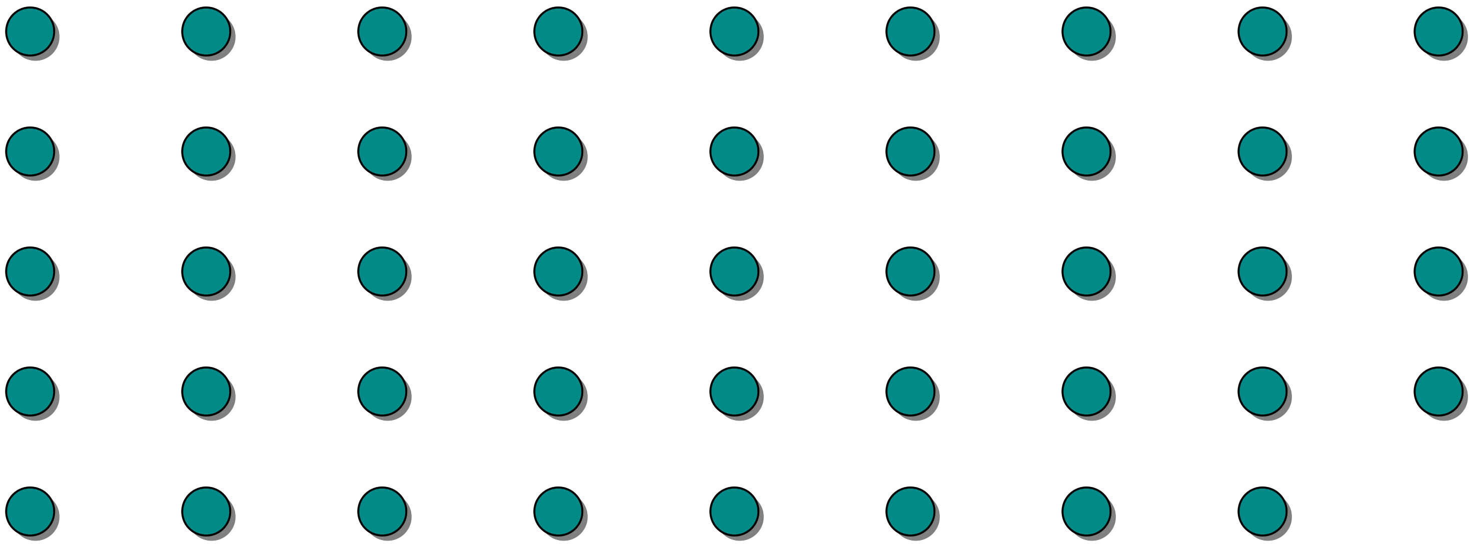
Worst-case linear-time order statistics

SELECT(i, n)

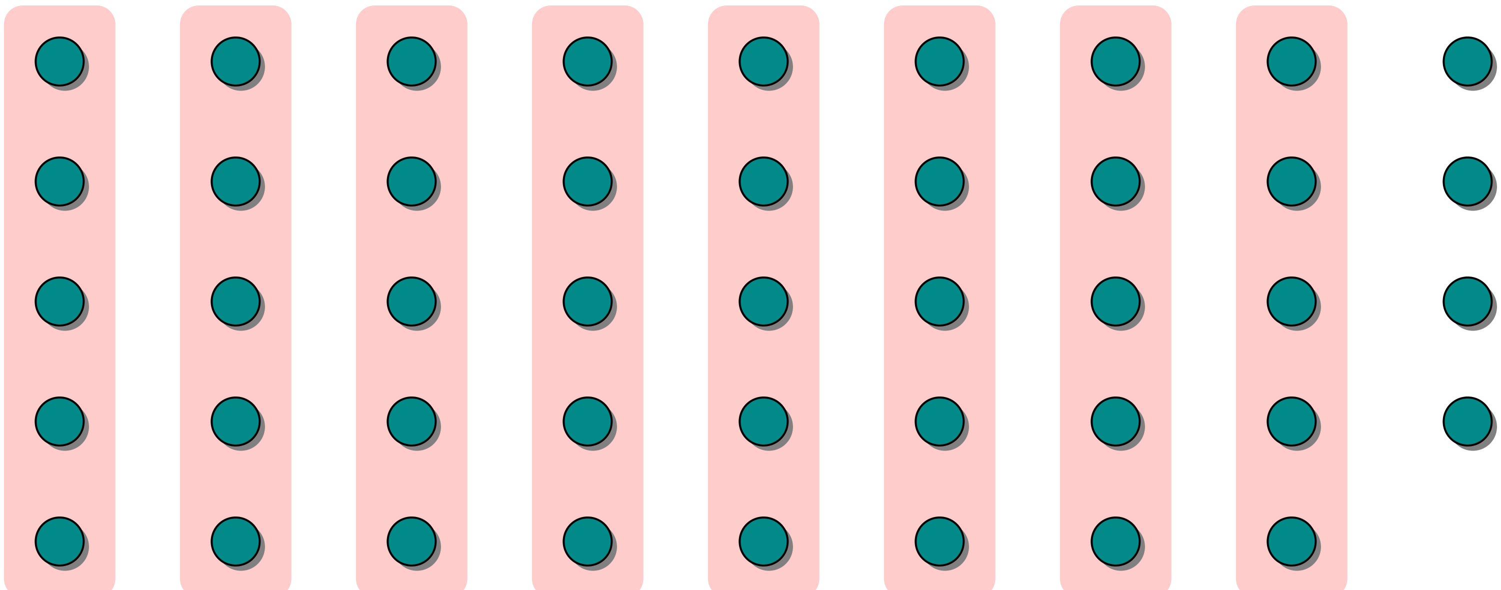
1. Divide the n elements into groups of 5. Find the median of each 5-element group by hand.
2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.
3. Partition around the pivot x. Let $k = \text{rank}(x)$.
4. **if** $i = k$ **then return** x
   elseif $i < k$
   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

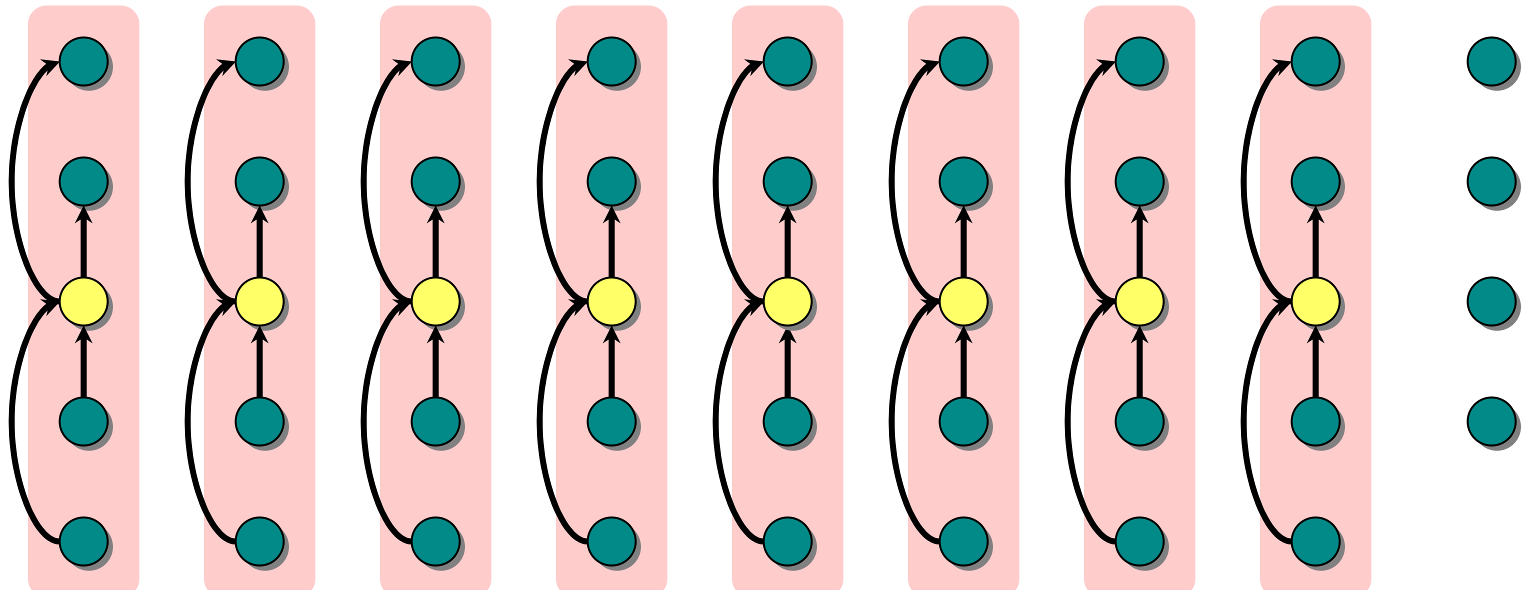


Choosing the pivot

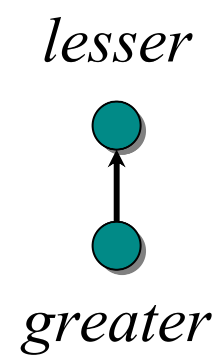


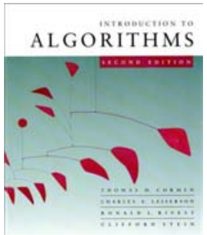
1. Divide the n elements into groups of 5.

Choosing the pivot

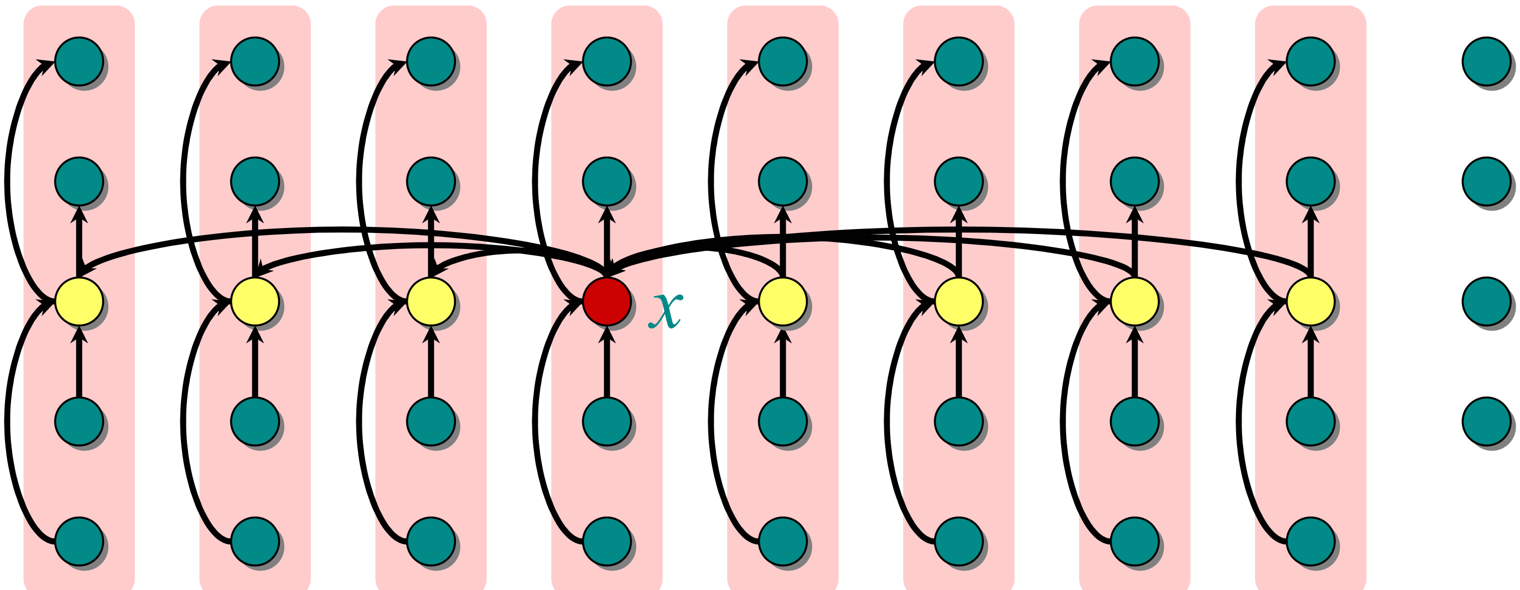


1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.

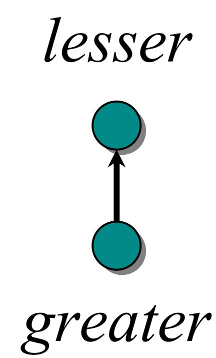


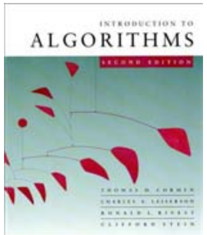


Choosing the pivot

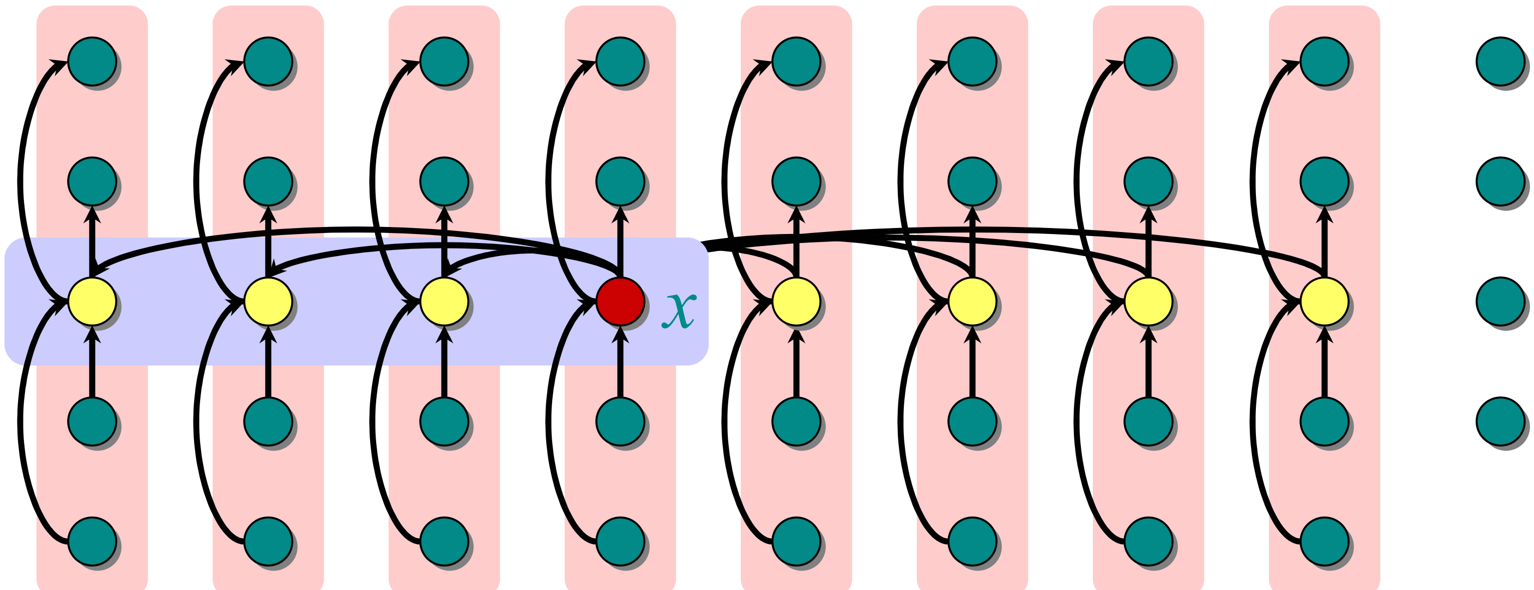


1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.
2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.

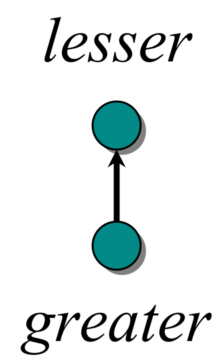


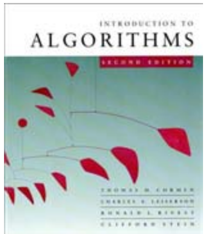


Analysis

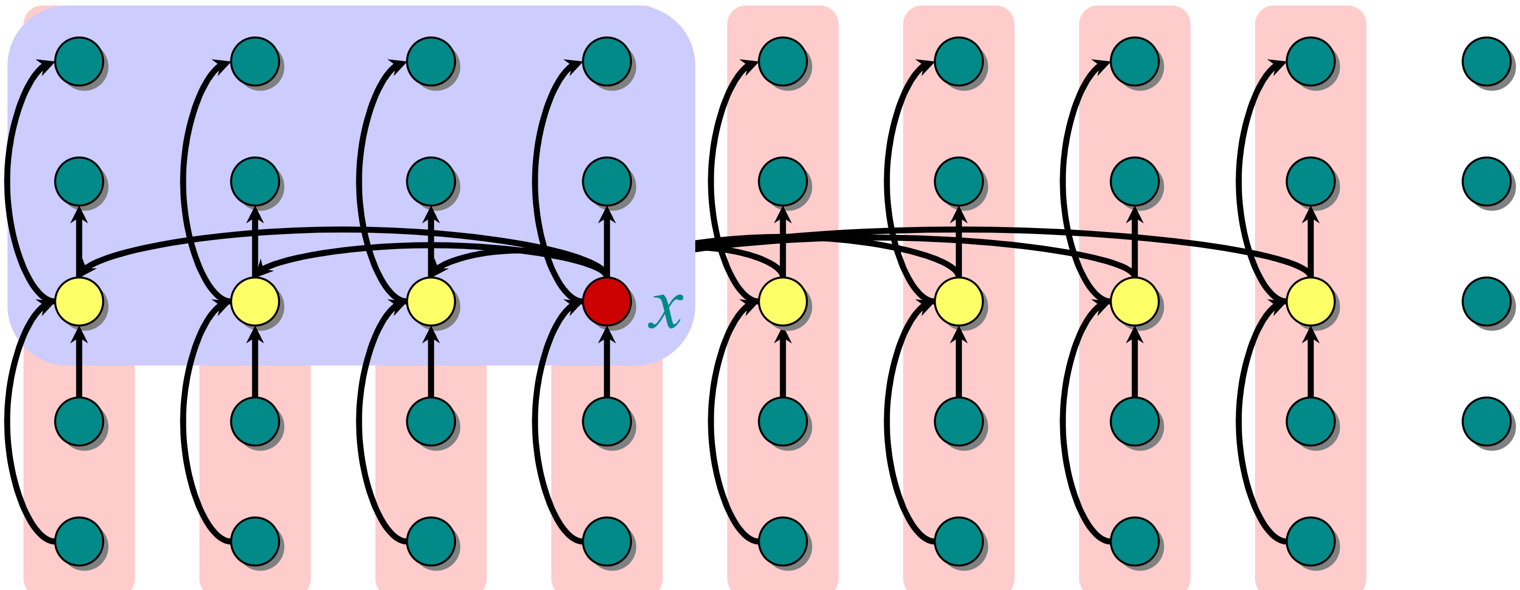


At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ group medians.





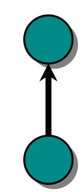
Analysis



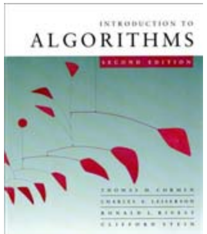
At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ group medians.

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

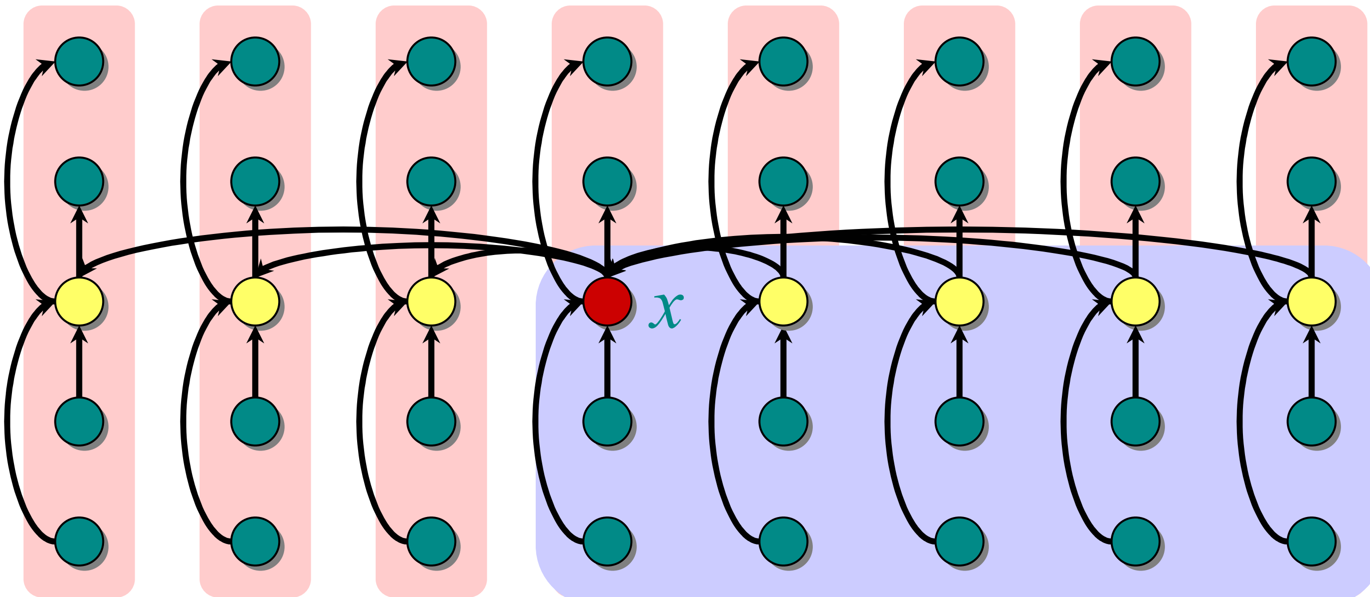
lesser



greater

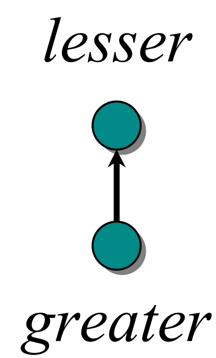


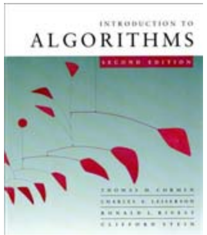
Analysis



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

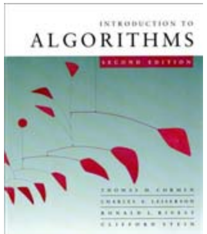
- Therefore, at least $3\lfloor n/10 \rfloor$ elements are $\le x$.
- Similarly, at least $3\lfloor n/10 \rfloor$ elements are $\ge x$.





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 $k = \text{rank}(x)$.
$T(7n/10)$	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

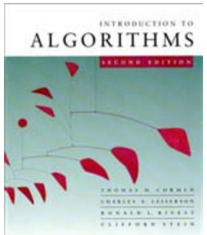
$$T(n) = T\left(\frac{1}{5}n\right) + T\left(\frac{7}{10}n\right) + \Theta(n)$$

Substitution:

$T(n) \leq cn$

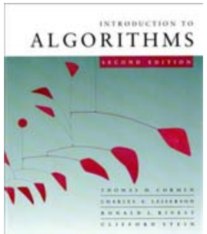
$$\begin{aligned} T(n) &\leq \frac{1}{5}cn + \frac{7}{10}cn + \Theta(n) \\ &= \frac{18}{20}cn + \Theta(n) \\ &= cn - \left(\frac{2}{20}cn - \Theta(n)\right) \\ &\leq cn \end{aligned}$$

if c is chosen large enough to handle the $\Theta(n)$.



Minor simplification

- For $n \geq 50$, we have $3\lfloor n/10 \rfloor \geq n/4$.
- Therefore, for $n \geq 50$ the recursive call to SELECT in Step 4 is executed recursively on $\leq 3n/4$ elements.
- Thus, the recurrence for running time can assume that Step 4 takes time $T(3n/4)$ in the worst case.
- For $n < 50$, we know that the worst-case time is $T(n) = \Theta(1)$.



Conclusions

- Since the work at each level of recursion is a constant fraction (18/20) 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.

Exercise: *Why not divide into groups of 3?*