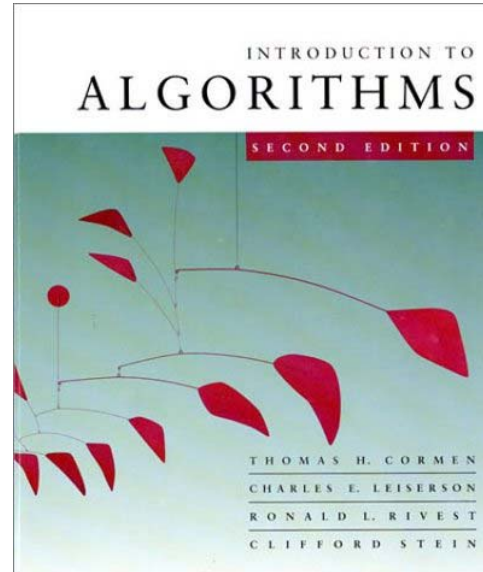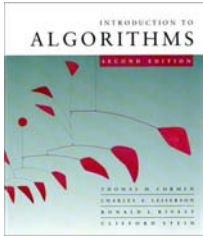# CS 5633 -- Spring 2008

Order Statistics

Carola Wenk

Slides courtesy of Charles Leiserson with small changes by Carola Wenk

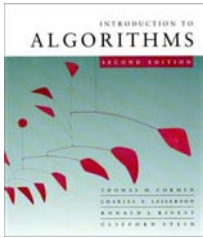# Order statistics

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 $= \Theta(n \log n) + \Theta(1)$
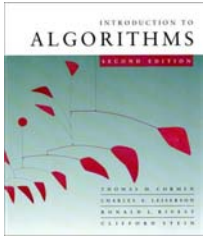$$= \Theta(n \log n),$$
using merge sort or heapsort (*not* quicksort).
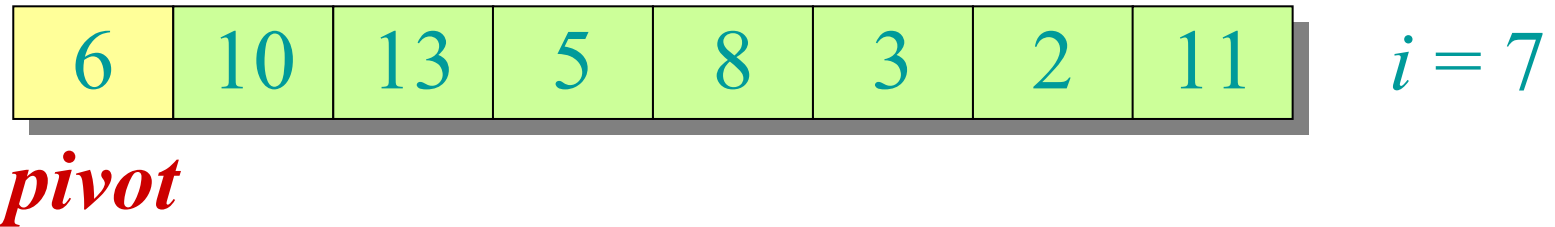
Randomized divide-and-conquer algorithm

RAND-SELECT(A, p, q, i)   ▷ ith smallest of $A[p\,..\,q]$
   if $p = q$ **then return** $A[p]$
   $r \leftarrow$ RAND-PARTITION(A, p, q)
   $k \leftarrow r - p + 1$ ▷ $k = \text{rank}(A[r])$
   if $i = k$ **then return** $A[r]$
   if $i < k$
      then return RAND-SELECT(A, p, $r - 1$, i)
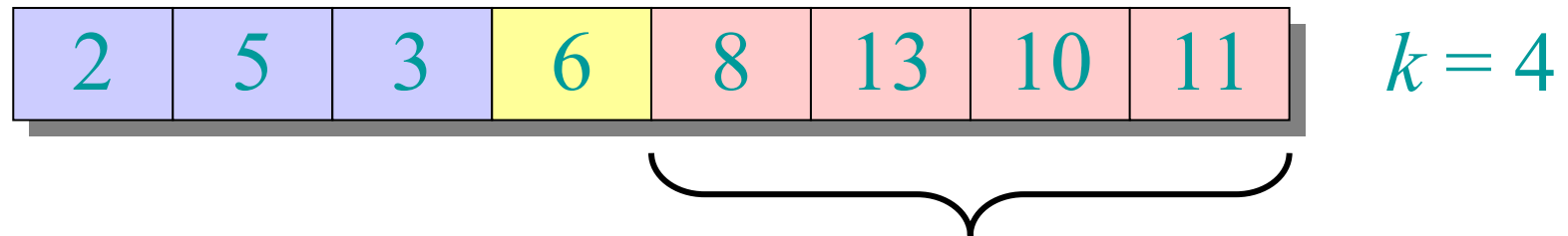   **else return** RAND-SELECT(A, $r + 1$, q, $i - k$)

CS 5633 Analysis of Algorithms
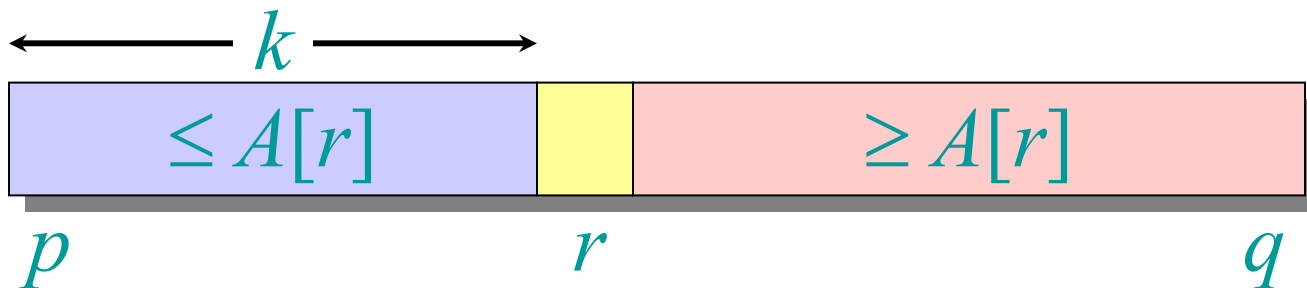
Example

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:

$$T(n) = T(9n/10) + \Theta(n)$$
$$= \Theta(n)$$

$$n^{\log_{10/9} 1} = n^0 = 1$$

CASE 3

Unlucky:

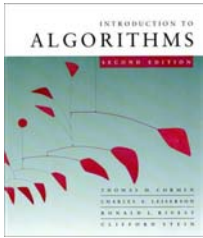$$T(n) = T(n-1) + \Theta(n)$$
$$= \Theta(n^2)$$

arithmetic series

Worse than sorting!

Analysis of expected time

The analysis follows that of randomized quicksort, but it's a little different.

Let $T(n)$ = the random variable for the running time of RAND-SELECT on an input of size n, assuming random numbers are independent.

For $k = 0, 1, \ldots, n-1$, define the ***indicator random variable***
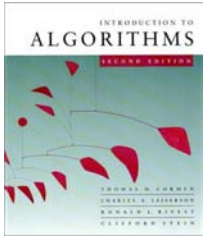
$$X_k = \begin{cases} 1 & \text{if PARTITION generates a } k : n-k-1 \text{ split,} \\ 0 & \text{otherwise.} \end{cases}$$
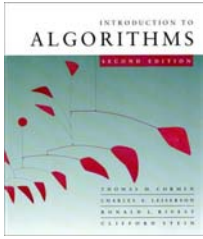
Analysis (continued)

To obtain an upper bound, assume that the *i* th element always falls in the larger side of the partition:

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

$$= \sum_{k=0}^{n-1} X_k \big(T(\max\{k, n-k-1\}) + \Theta(n) \big)$$

$$\leq 2 \sum_{k=\lfloor n/2 \rfloor}^{n-1} X_k \big(T(k) + \Theta(n) \big)$$
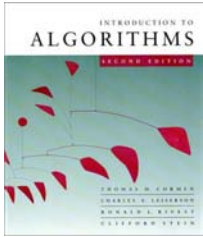
.

Calculating expectation

$$E[T(n)] = E\left[2 \sum_{k=\lfloor n/2 \rfloor}^{n-1} X_k \big(T(k) + \Theta(n) \big) \right]$$
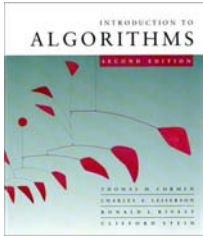
Take expectations of both sides.

Calculating expectation

$$E[T(n)] = E\left[2\sum_{k=\lfloor n/2\rfloor}^{n-1} X_k\big(T(k)+\Theta(n)\big)\right]$$

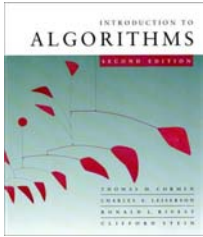$$= 2\sum_{k=\lfloor n/2\rfloor}^{n-1} E\big[X_k\big(T(k)+\Theta(n)\big)\big]$$

Linearity of expectation.

Calculating expectation

$$E[T(n)] = E\left[2 \sum_{k=\lfloor n/2 \rfloor}^{n-1} X_k \big(T(k) + \Theta(n) \big) \right]$$

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

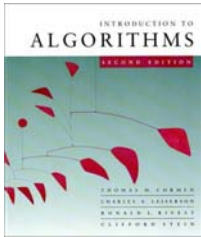$$= 2 \sum_{k=\lfloor n/2 \rfloor}^{n-1} E\big[X_k \big] \cdot E\big[T(k) + \Theta(n) \big]$$

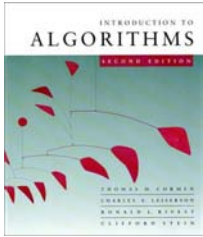Independence of X_k from other random choices.

Calculating expectation

$$E[T(n)] = E\left[2 \sum_{k=\lfloor n/2 \rfloor}^{n-1} X_k \big(T(k) + \Theta(n)\big) \right]$$

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

$$= 2 \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[X_k] \cdot E\big[T(k) + \Theta(n) \big]$$

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

Linearity of expectation; $E[X_k] = 1/n$.

Calculating expectation

$$E[T(n)] = E\left[2 \sum_{k=\lfloor n/2 \rfloor}^{n-1} X_k \big(T(k) + \Theta(n)\big) \right]$$

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

$$= 2 \sum_{k=\lfloor n/2 \rfloor}^{n-1} E\big[X_k \big] \cdot E\big[T(k) + \Theta(n) \big]$$

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

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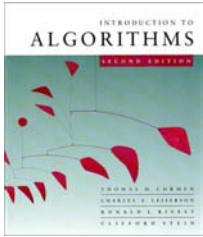
Hairy recurrence

(But not quite as hairy as the quicksort one.)

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

Prove: $E[T(n)] \leq cn$ for constant $c > 0$.

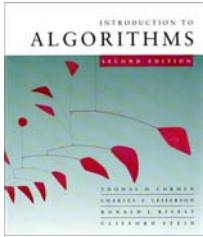- The constant c can be chosen large enough so that $E[T(n)] \leq cn$ for the base cases.

Use fact: $\displaystyle\sum_{k=\lfloor n/2 \rfloor}^{n-1} k \leq \frac{3}{8}n^2$ (exercise).

Substitution method

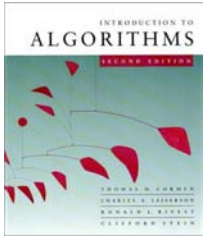$$E[T(n)] \leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + \Theta(n)$$

Substitute inductive hypothesis.

Substitution method

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

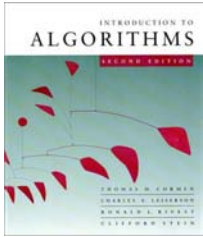$$\leq \frac{2c}{n} \left(\frac{3}{8} n^2 \right) + \Theta(n)$$

Use fact.

Substitution method

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

$$\leq \frac{2c}{n} \left(\frac{3}{8} n^2 \right) + \Theta(n)$$

$$= cn - \left(\frac{cn}{4} - \Theta(n) \right)$$

Express as *desired – residual*.

Substitution method
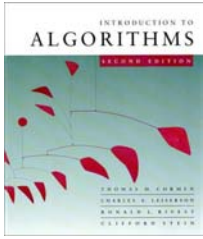
$$E[T(n)] \le \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + \Theta(n)$$

$$\le \frac{2c}{n}\left(\frac{3}{8}n^2\right) + \Theta(n)$$

$$= cn - \left(\frac{cn}{4} - \Theta(n)\right)$$

$$\le cn,$$

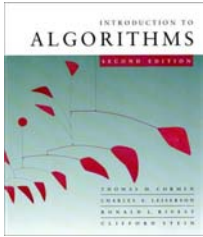if c is chosen large enough so that $cn/4$ dominates the $\Theta(n)$.

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].
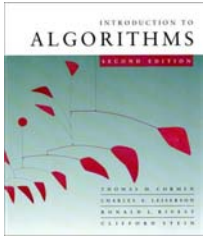
IDEA: Generate a good pivot recursively.
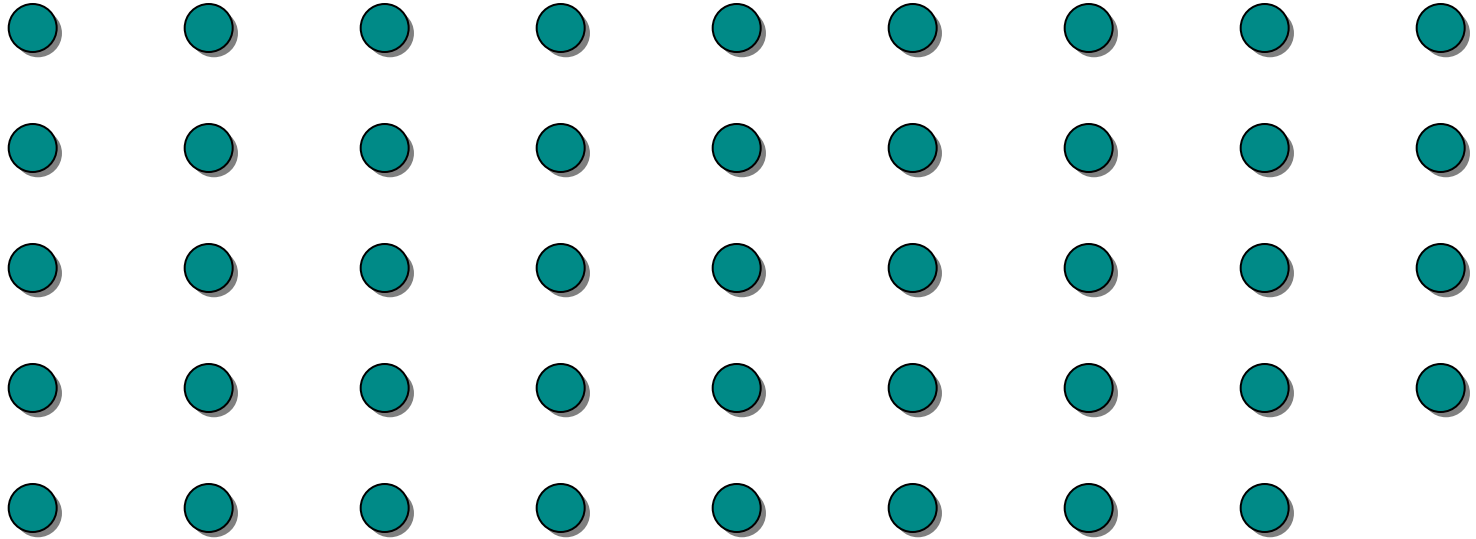
Worst-case linear-time order statistics
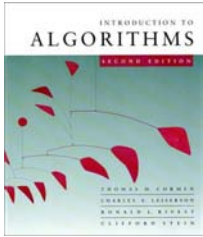
SELECT(i, n)

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 $k = \text{rank}(x)$.

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
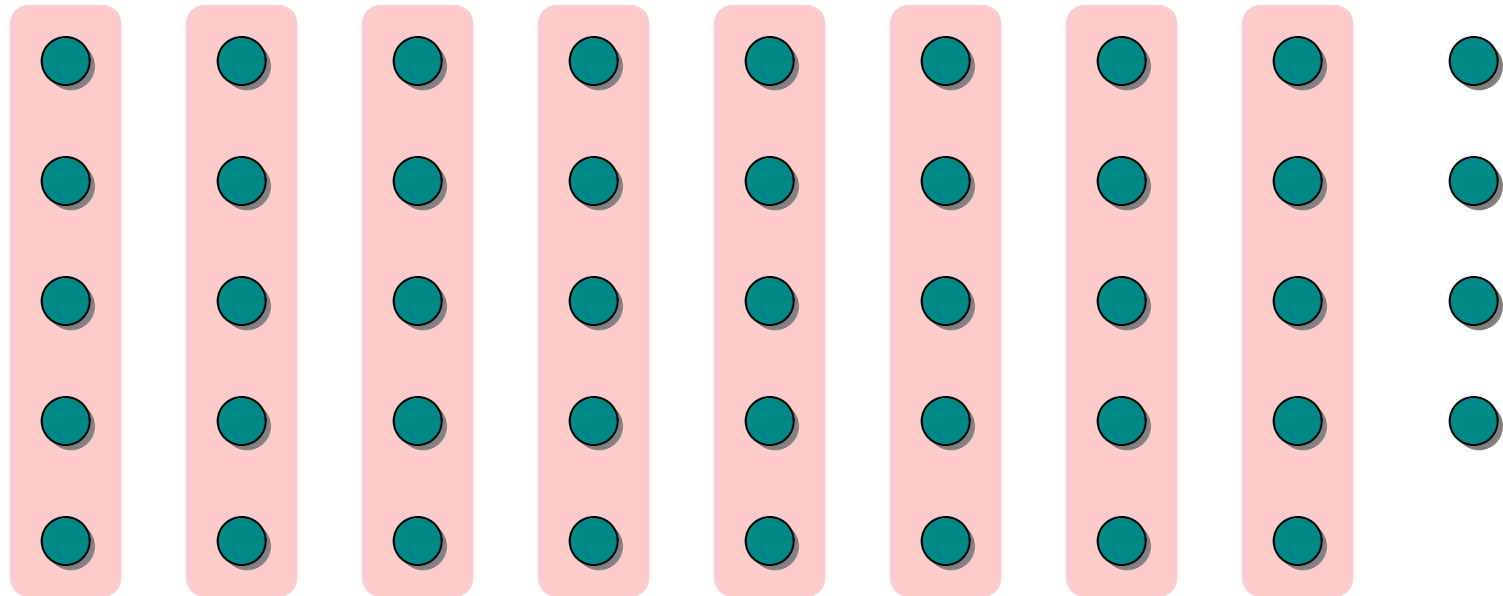
Same as RAND-SELECT
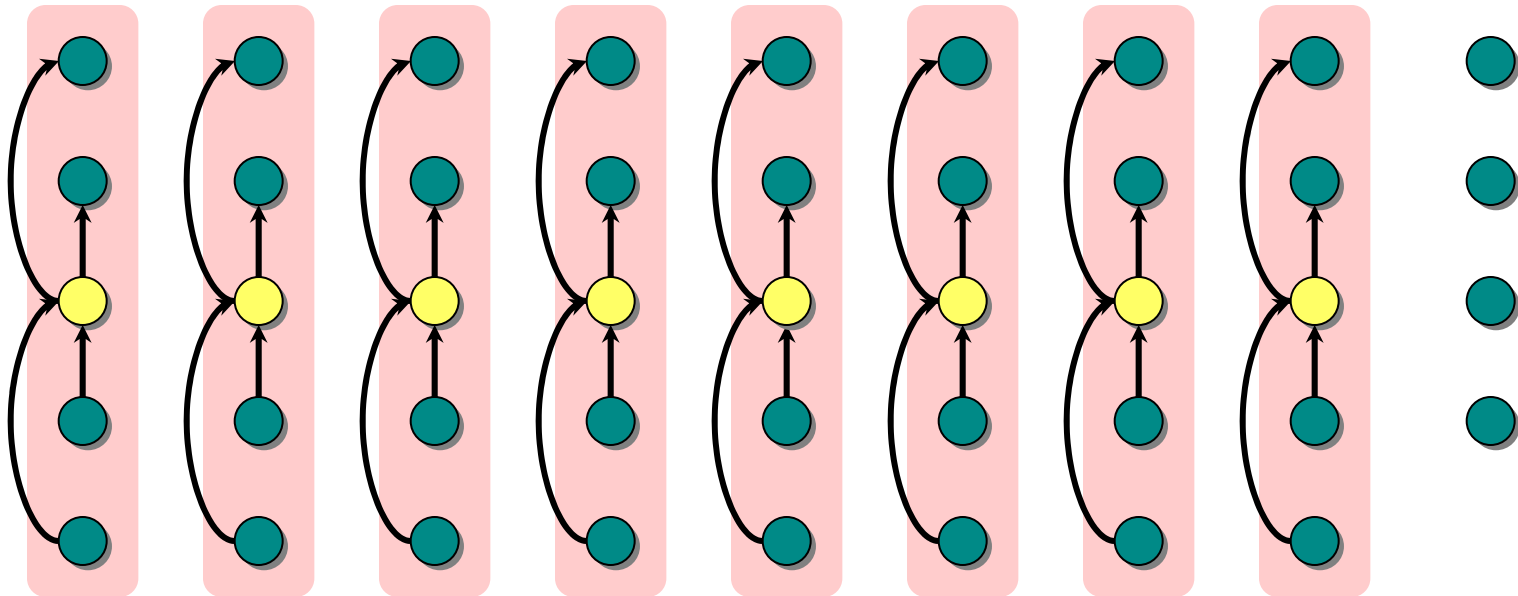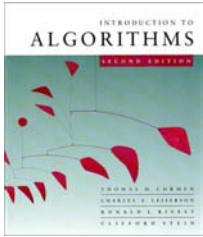
Choosing the pivot

CS 5633 Analysis of Algorithms

Choosing the pivot

1. Divide the *n* elements into groups of 5.
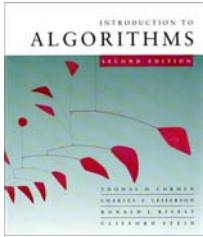
Choosing the pivot

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

lesser

greater

Choosing the pivot

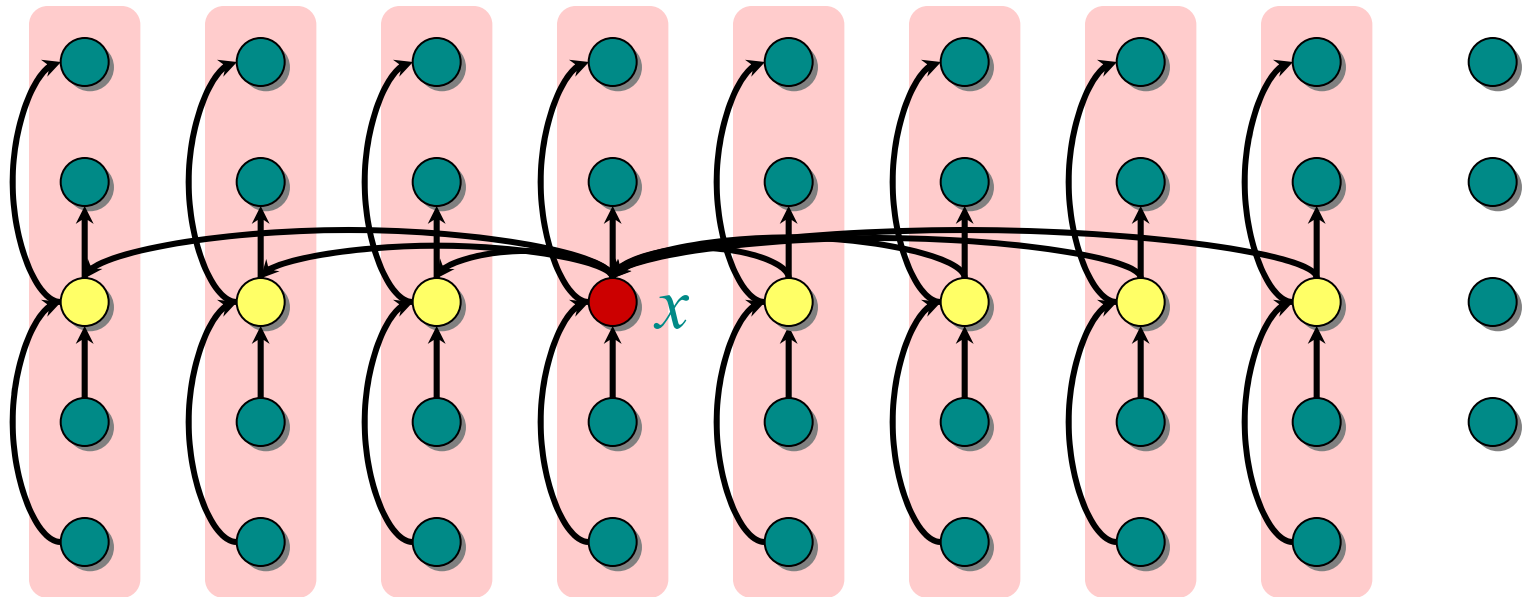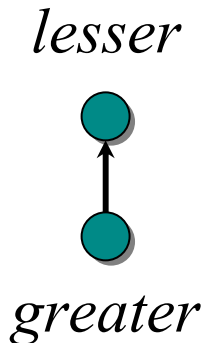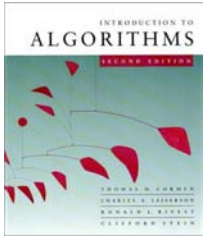

lesser

greater

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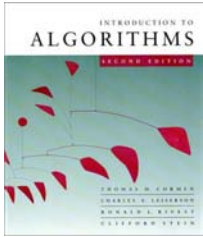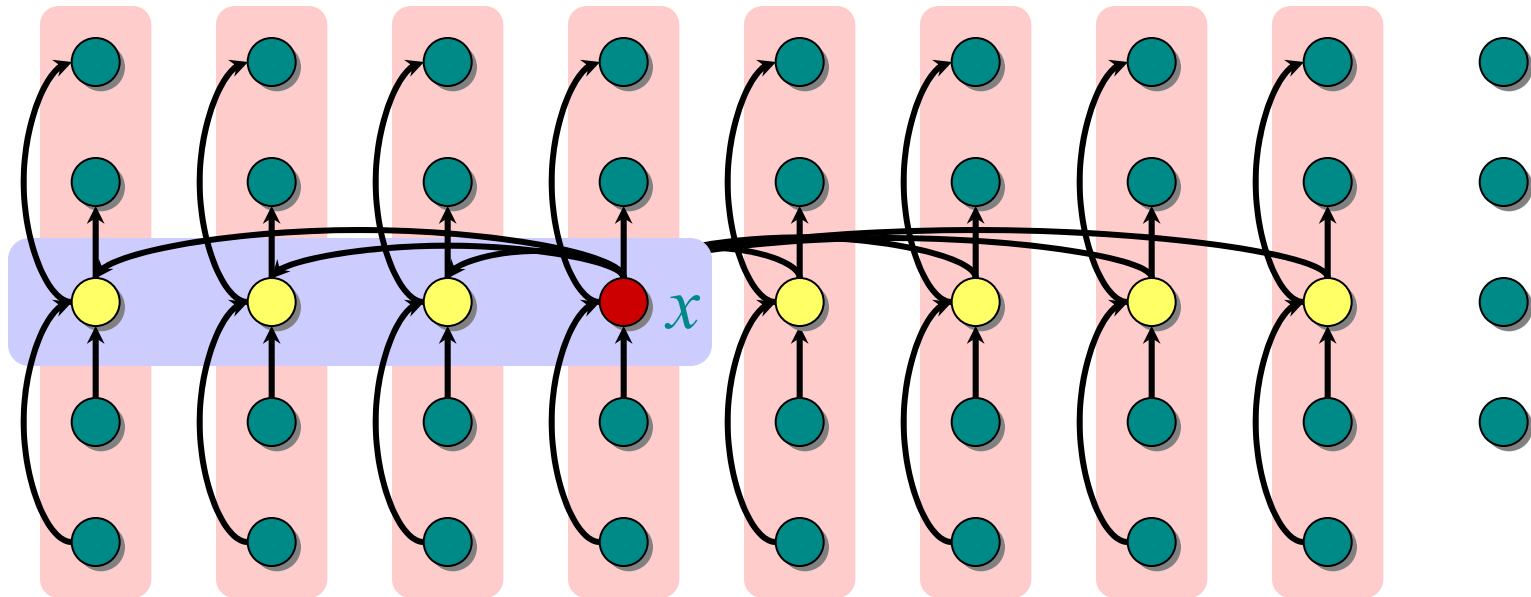2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.

Developing the recurrence

$T(n)$ SELECT(i, n)

$\Theta(n)$ $\left\{\begin{array}{l} \\ \\ \end{array}\right.$ 1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.

$T(n/5)$ $\left\{\begin{array}{l} \\ \\ \end{array}\right.$ 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 $k = \text{rank}(x)$.

$T(\ ?\ )$ $\left\{\begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array}\right.$ 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.
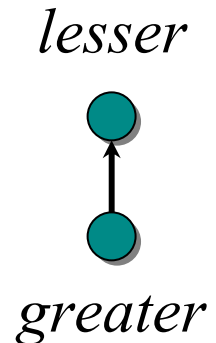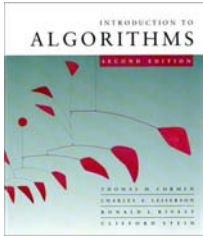
lesser

greater

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 $\leq x$.

lesser

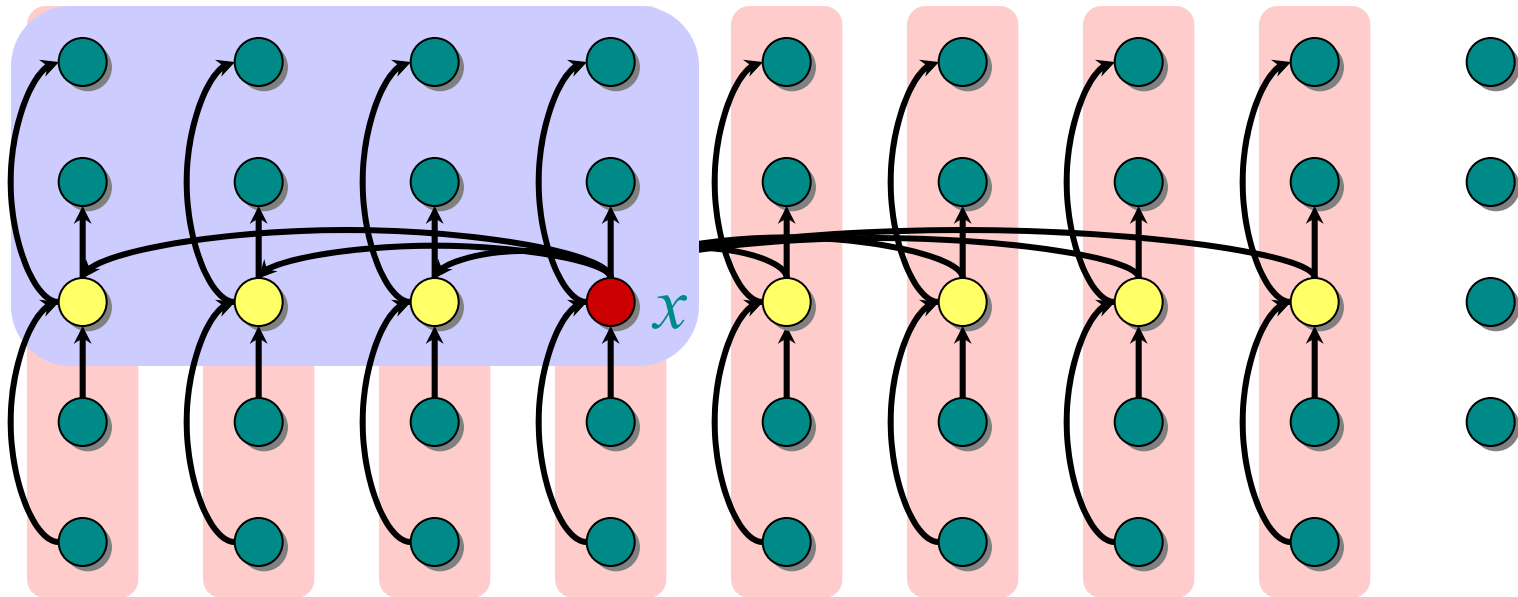*greater*

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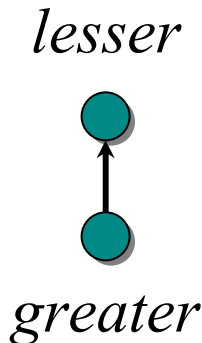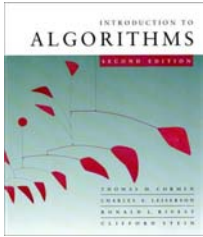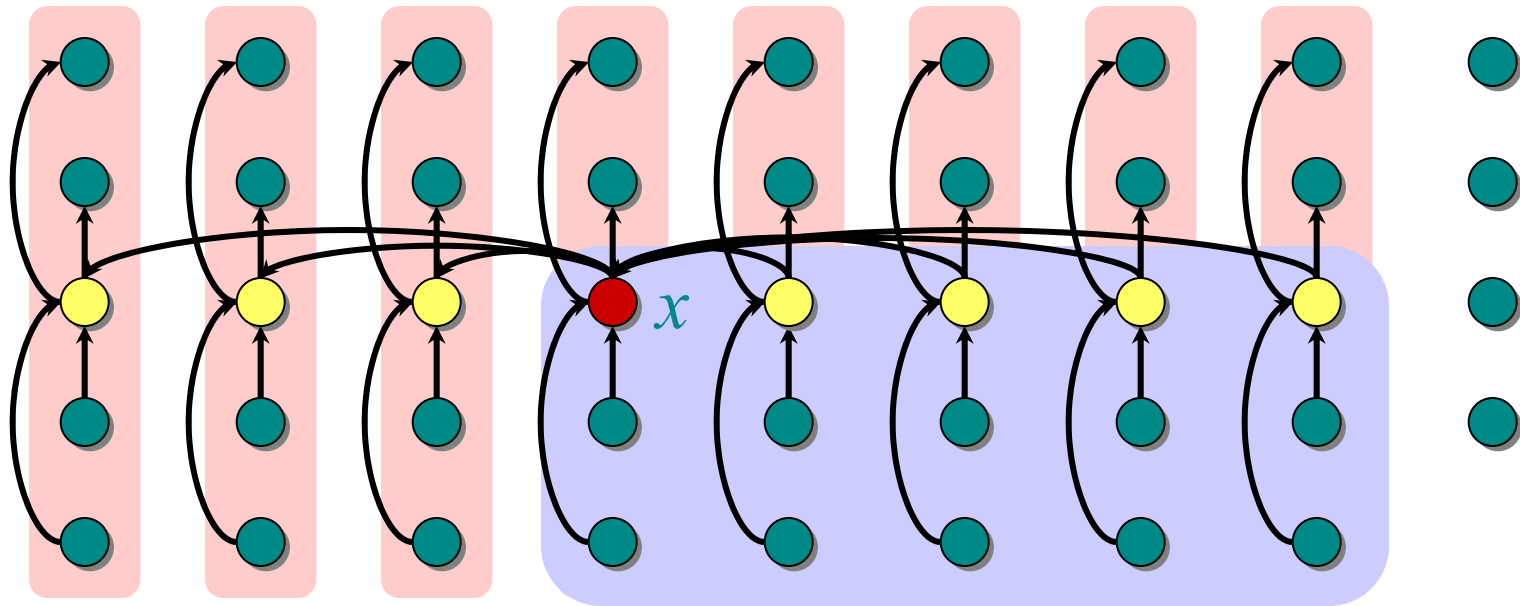
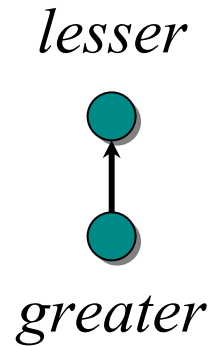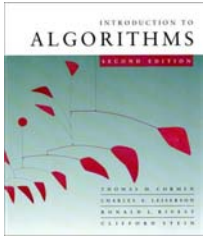- Therefore, at least $3\lfloor n/10 \rfloor$ elements are $\leq x$.
- Similarly, at least $3\lfloor n/10 \rfloor$ elements are $\geq x$.

lesser

greater

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

Analysis (Assume all elements are distinct.)

- Use fact that $\lfloor a/b \rfloor \geq ((a-(b-1))/b$ (page 51)
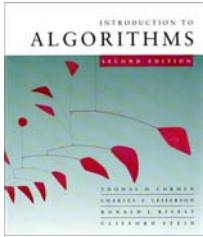- $n-3\lfloor n/10 \rfloor \leq n-3 \cdot (n-9)/10 = (10n-3n+27)/10 \leq 7n/10 + 3$

- The recursive call to SELECT in Step 4 is executed recursively on at most $7n/10+3$ elements.

CS 5633 Analysis of Algorithms

Developing the recurrence

$T(n)$ SELECT(i, n)

$\Theta(n)$ $\left\{\vphantom{\begin{array}{c} a \\ b \end{array}}\right.$ 1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.

$T(n/5)$ $\left\{\vphantom{\begin{array}{c} a \\ b \end{array}}\right.$ 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 $k = \mathrm{rank}(x)$.

$T(7n/10 +3)$ $\left\{\vphantom{\begin{array}{c} a \\ b \\ c \\ d \\ e \end{array}}\right.$ 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

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

Substitution:

$$T(n) \le c(n - 4)$$

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

$$T(n) \le c(\frac{1}{5}n - 4) + c(\frac{7}{10}n + 3 - 4) + dn$$

$$\le \frac{9}{10}cn - 4c + dn$$

$$= c(n - 4) - \frac{1}{10}cn + dn$$

$$\le c(n - 4) \; ,$$

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.*