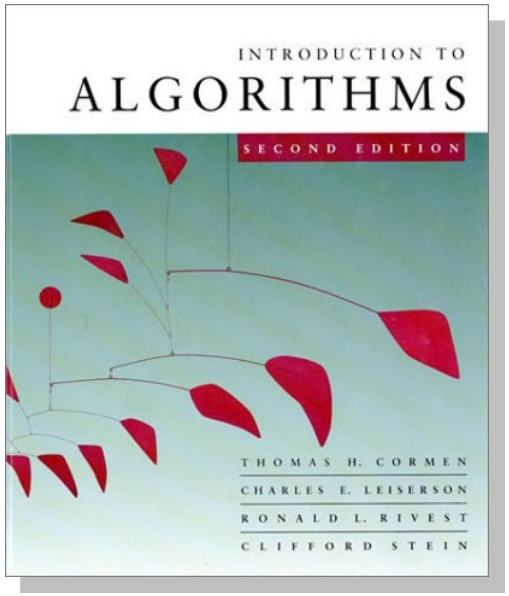
Introduction to Algorithms

6.046J/18.401J

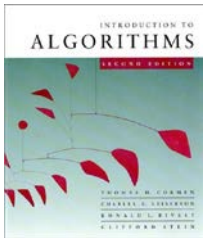


LECTURE 6

Order Statistics

- Randomized divide and conquer
- Analysis of expected time
- Worst-case linear-time order statistics
- Analysis

Prof. Erik Demaine



Order statistics

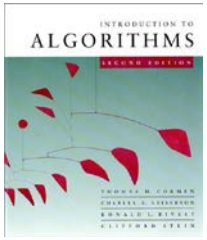
Select the ith smallest of n elements (the element with ***rank*** i).

- $i = 1$: ***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 $= \Theta(n \lg n) + \Theta(1)$
$= \Theta(n \lg n)$,

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)

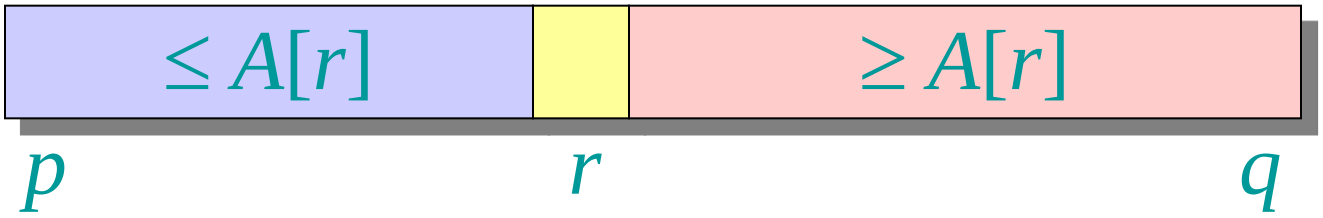
$k \leftarrow r - p + 1$ ▷ $k = \text{rank}(A[r])$

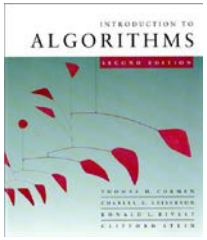
if $i = k$ **then return** $A[r]$

if $i < k$

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

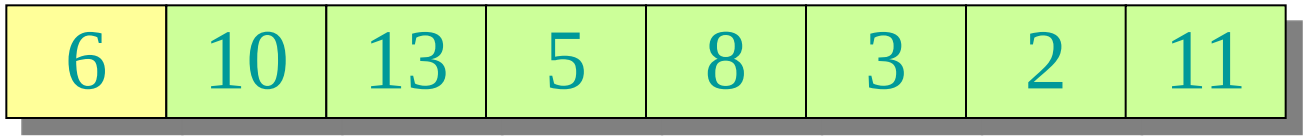
else return RAND-SELECT$(A, r + 1, q, i - k)$





Example

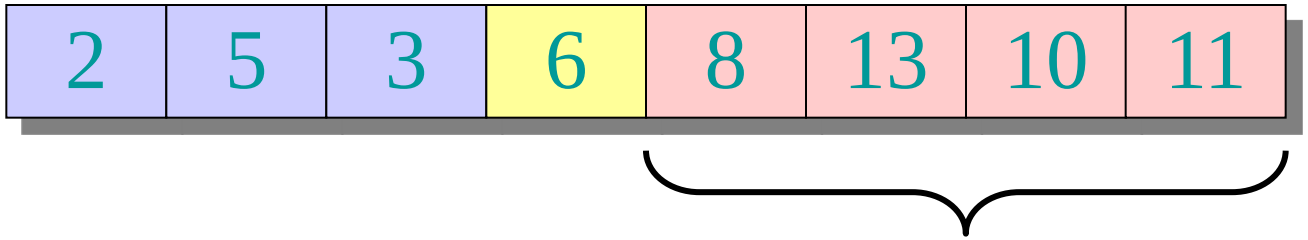
Select the $i = 7$th smallest:



$i = 7$

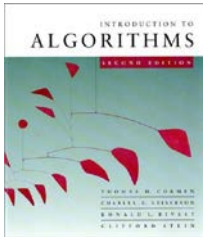
pivot

Partition:



$k = 4$

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



Intuition for analysis

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

Lucky:

$T(n) = T(9n/10) + \Theta(n)$ $\quad n^{\log_{10/9} 1} = n^0 = 1$

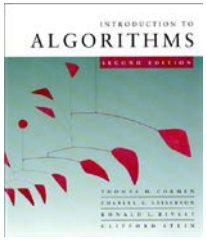
$= \Theta(n)$ CASE 3

Unlucky:

$T(n) = T(n-1) + \Theta(n)$ arithmetic series

$= \Theta(n^2)$

Worse than sorting!



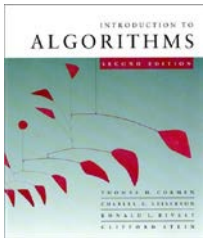
Analysis of expected time

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***

$$X_k = \begin{cases} 1 & \text{if PARTITION generates a } k : n-k-1 \text{ split,} \\ 0 & \text{otherwise.} \end{cases}$$

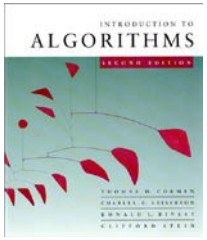


Analysis (continued)

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

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

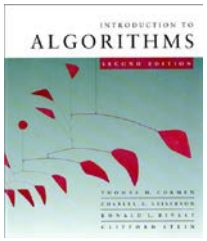
$$= \sum_{k=0}^{n-1} X_k \left(T(\max\{k, n-k-1\}) + \Theta(n)\right).$$



Calculating expectation

$$E[T(n)] = E\left[\sum_{k=0}^{n-1} X_k \left(T(\max\{k, n-k-1\}) + \Theta(n)\right)\right]$$

Take expectations of both sides.

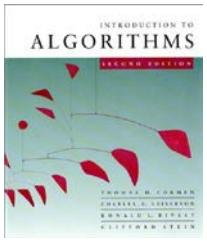


Calculating expectation

$$E[T(n)] = E\left[\sum_{k=0}^{n-1} X_k \left(T(\max\{k, n-k-1\}) + \Theta(n)\right)\right]$$

$$= \sum_{k=0}^{n-1} E\left[X_k \left(T(\max\{k, n-k-1\}) + \Theta(n)\right)\right]$$

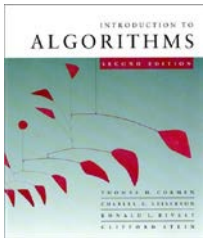
Linearity of expectation.



Calculating expectation

$$
\begin{aligned}
E[T(n)] &= E\left[\sum_{k=0}^{n-1} X_k \left(T(\max\{k, n-k-1\}) + \Theta(n)\right)\right] \\
&= \sum_{k=0}^{n-1} E\left[X_k \left(T(\max\{k, n-k-1\}) + \Theta(n)\right)\right] \\
&= \sum_{k=0}^{n-1} E\left[X_k\right] \cdot E\left[T(\max\{k, n-k-1\}) + \Theta(n)\right]
\end{aligned}
$$

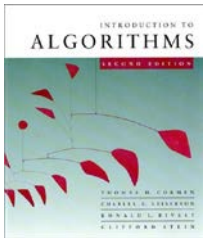
Independence of X_k from other random choices.



Calculating expectation

$$
\begin{aligned}
E[T(n)] &= E\left[\sum_{k=0}^{n-1} X_k \left(T(\max\{k, n-k-1\}) + \Theta(n)\right)\right] \\
&= \sum_{k=0}^{n-1} E\left[X_k \left(T(\max\{k, n-k-1\}) + \Theta(n)\right)\right] \\
&= \sum_{k=0}^{n-1} E\left[X_k\right] \cdot E\left[T(\max\{k, n-k-1\}) + \Theta(n)\right] \\
&= \frac{1}{n}\sum_{k=0}^{n-1} E\left[T(\max\{k, n-k-1\})\right] + \frac{1}{n}\sum_{k=0}^{n-1} \Theta(n)
\end{aligned}
$$

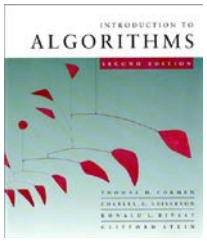
Linearity of expectation; $E[X_k] = 1/n$.



Calculating expectation

$$
\begin{aligned}
E[T(n)] &= E\left[\sum_{k=0}^{n-1} X_k \left(T(\max\{k, n-k-1\}) + \Theta(n)\right)\right] \\
&= \sum_{k=0}^{n-1} E\left[X_k \left(T(\max\{k, n-k-1\}) + \Theta(n)\right)\right] \\
&= \sum_{k=0}^{n-1} E\left[X_k\right] \cdot E\left[T(\max\{k, n-k-1\}) + \Theta(n)\right] \\
&= \frac{1}{n}\sum_{k=0}^{n-1} E\left[T(\max\{k, n-k-1\})\right] + \frac{1}{n}\sum_{k=0}^{n-1} \Theta(n) \\
&\le \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} E\left[T(k)\right] + \Theta(n)
\end{aligned}
$$

Upper terms appear twice.



Hairy recurrence

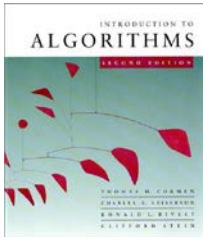
(But not quite as hairy as the quicksort one.)

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

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

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

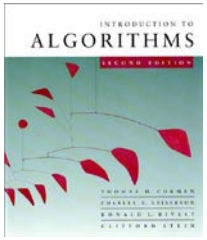
Use fact: $\displaystyle\sum_{k=\lfloor n/2 \rfloor}^{n-1} k \le \frac{3}{8}n^2$ (exercise).



Substitution method

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

Substitute inductive hypothesis.

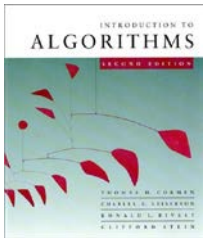


Substitution method

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

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

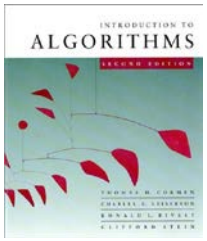
Use fact.



Substitution method

$$
\begin{aligned}
E[T(n)] &\le \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + \Theta(n) \\
&\le \frac{2c}{n}\left(\frac{3}{8}n^2\right) + \Theta(n) \\
&= cn - \left(\frac{cn}{4} - \Theta(n)\right)
\end{aligned}
$$

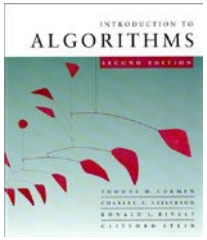
Express as ***desired – residual***.



Substitution method

$$
\begin{aligned}
E[T(n)] &\le \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + \Theta(n) \\
&\le \frac{2c}{n}\left(\frac{3}{8}n^2\right) + \Theta(n) \\
&= cn - \left(\frac{cn}{4} - \Theta(n)\right) \\
&\le cn,
\end{aligned}
$$

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



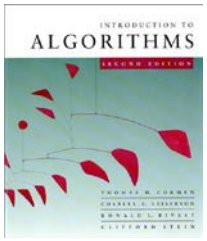
Summary of randomized order-statistic selection

- 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.

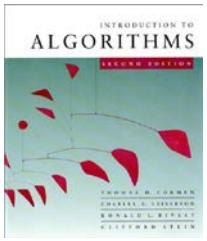


Worst-case linear-time order statistics

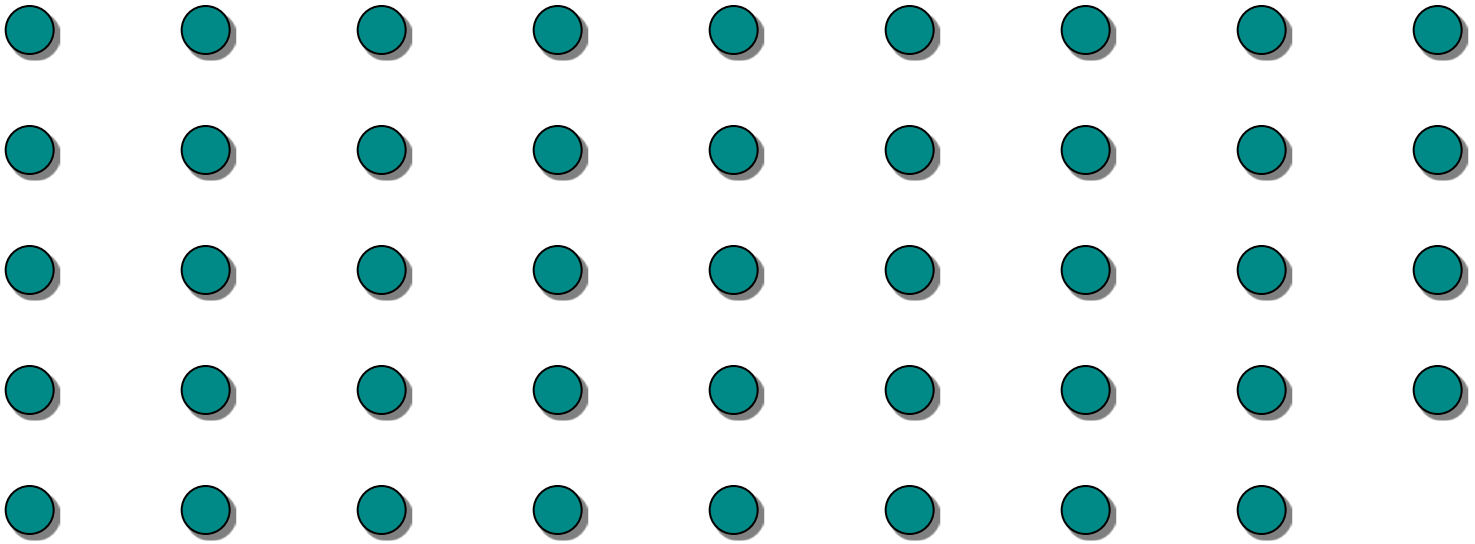
SELECT(i, n)

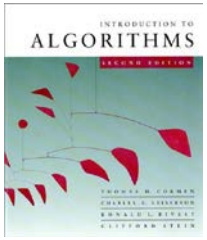
1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.
2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.
3. Partition around the pivot x. Let $k = \text{rank}(x)$.
4. **if** $i = k$ **then return** x
   elseif $i < k$
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

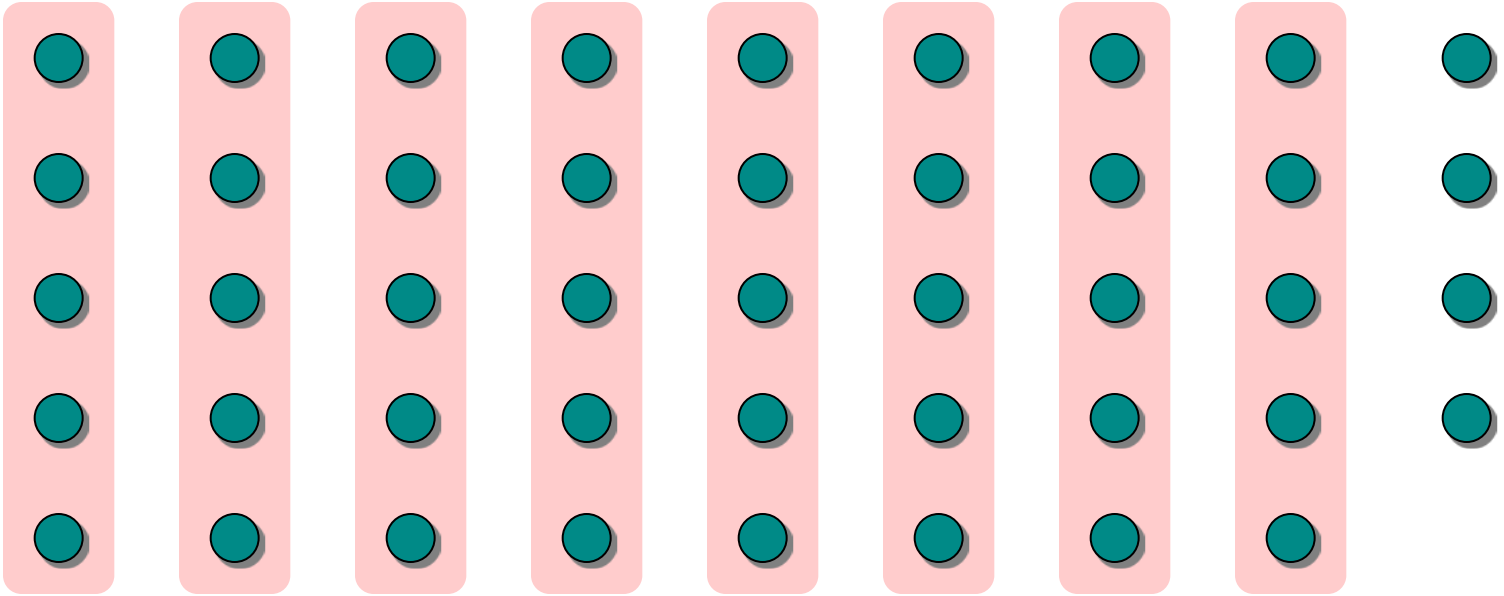


Choosing the pivot

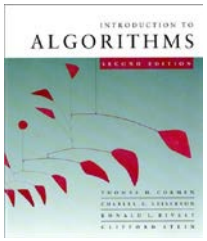




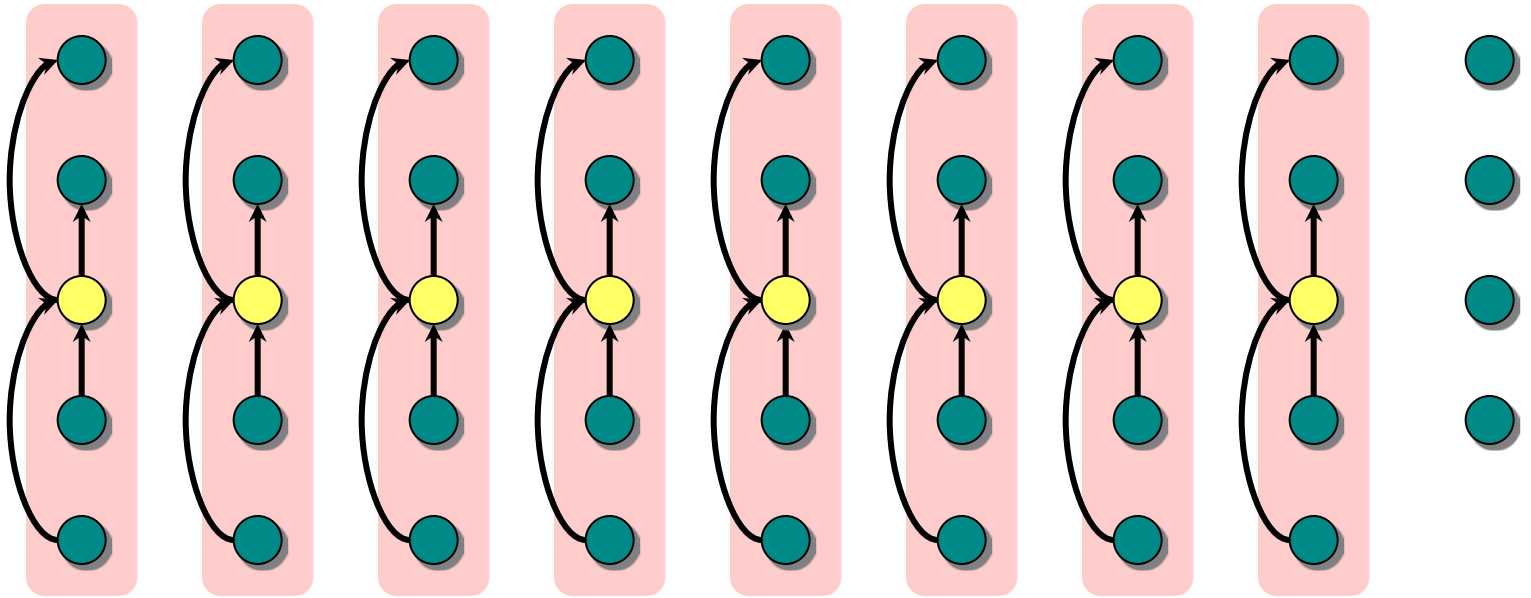
Choosing the pivot



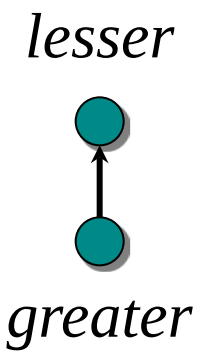
1. Divide the n elements into groups of 5.

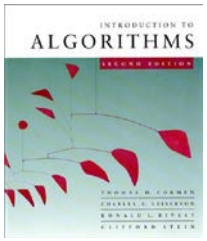


Choosing the pivot

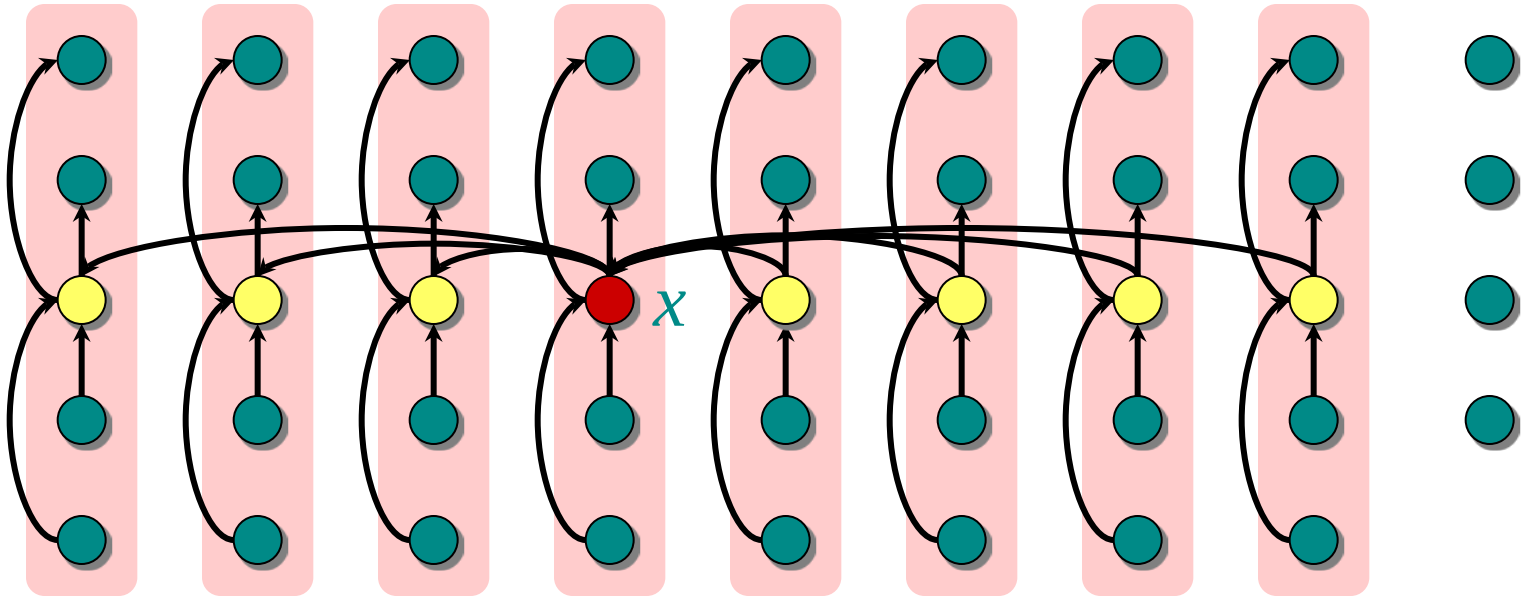


1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.

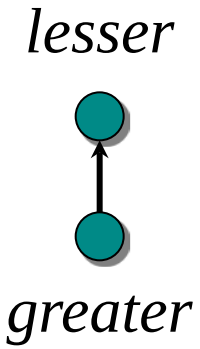


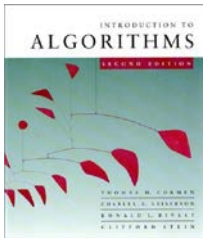


Choosing the pivot

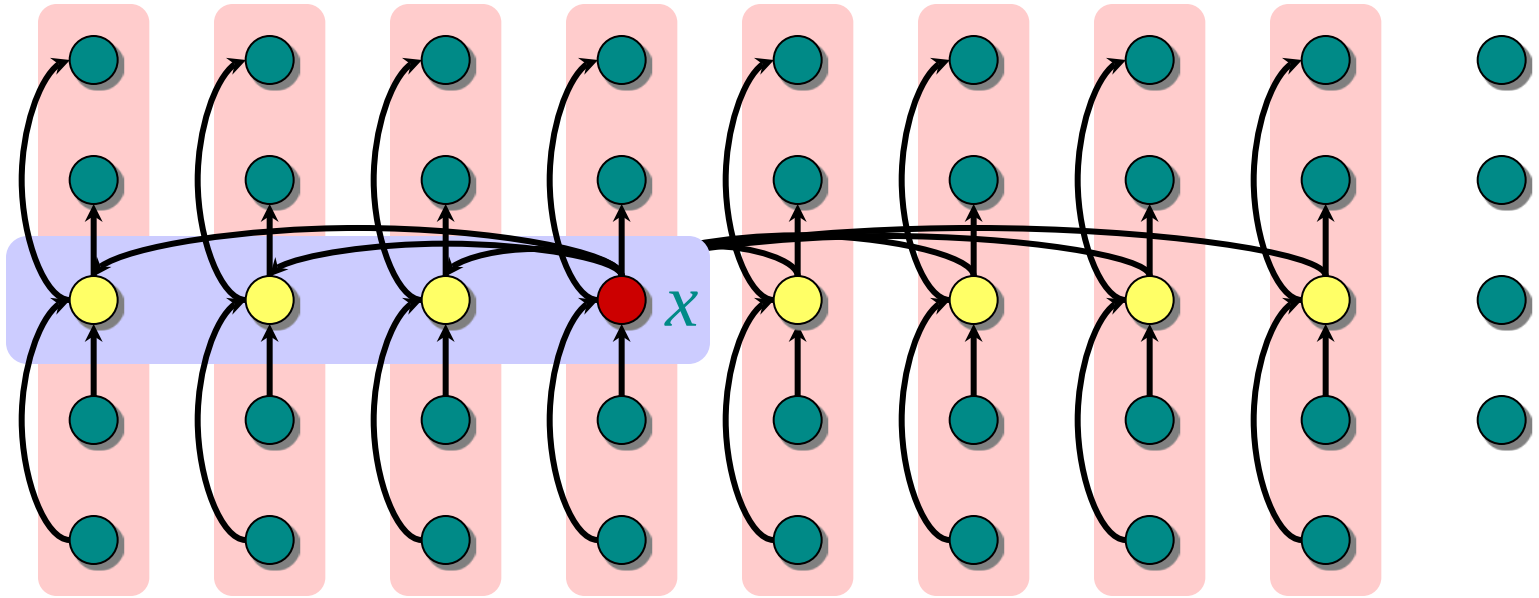


1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.
2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.

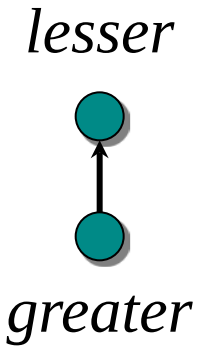


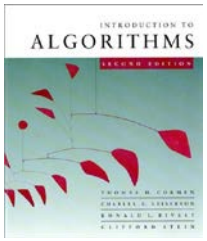


Analysis

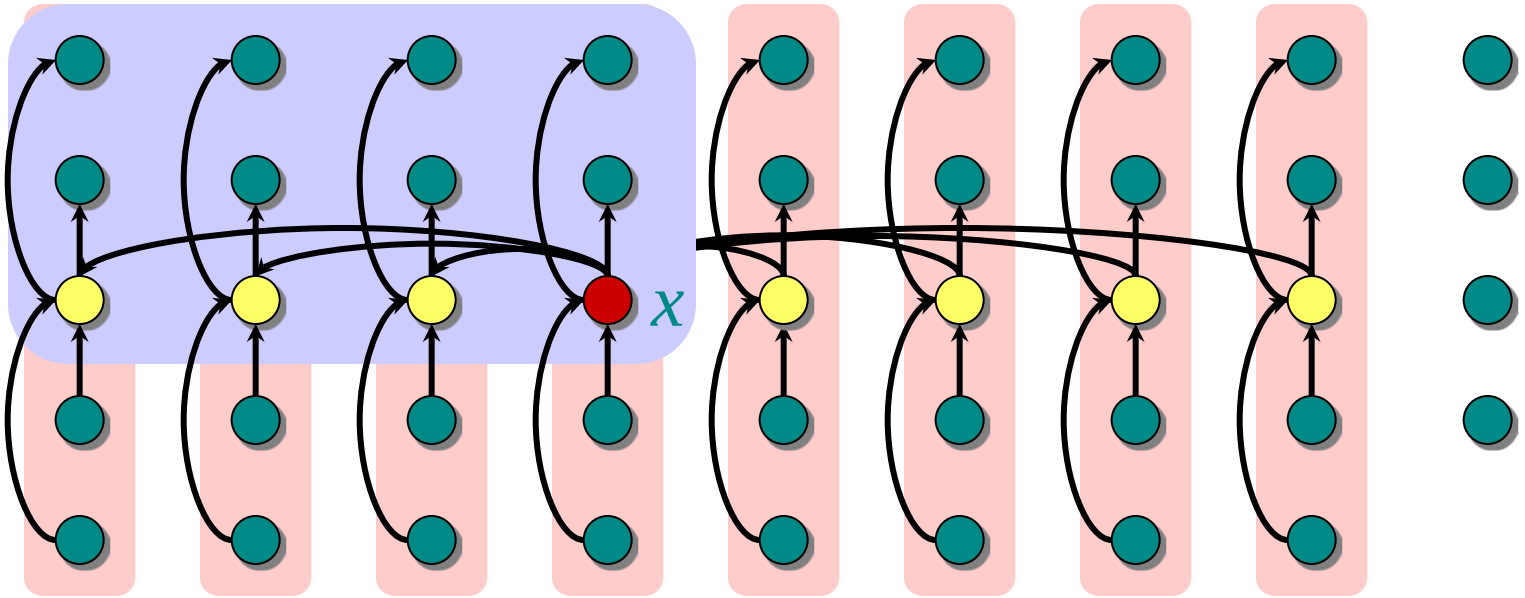


At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ group medians.



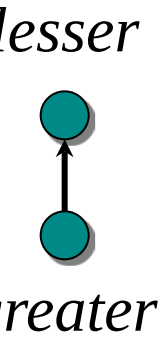


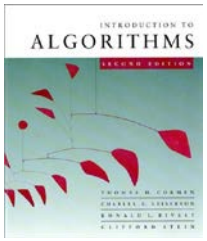
Analysis (Assume all elements are distinct.)



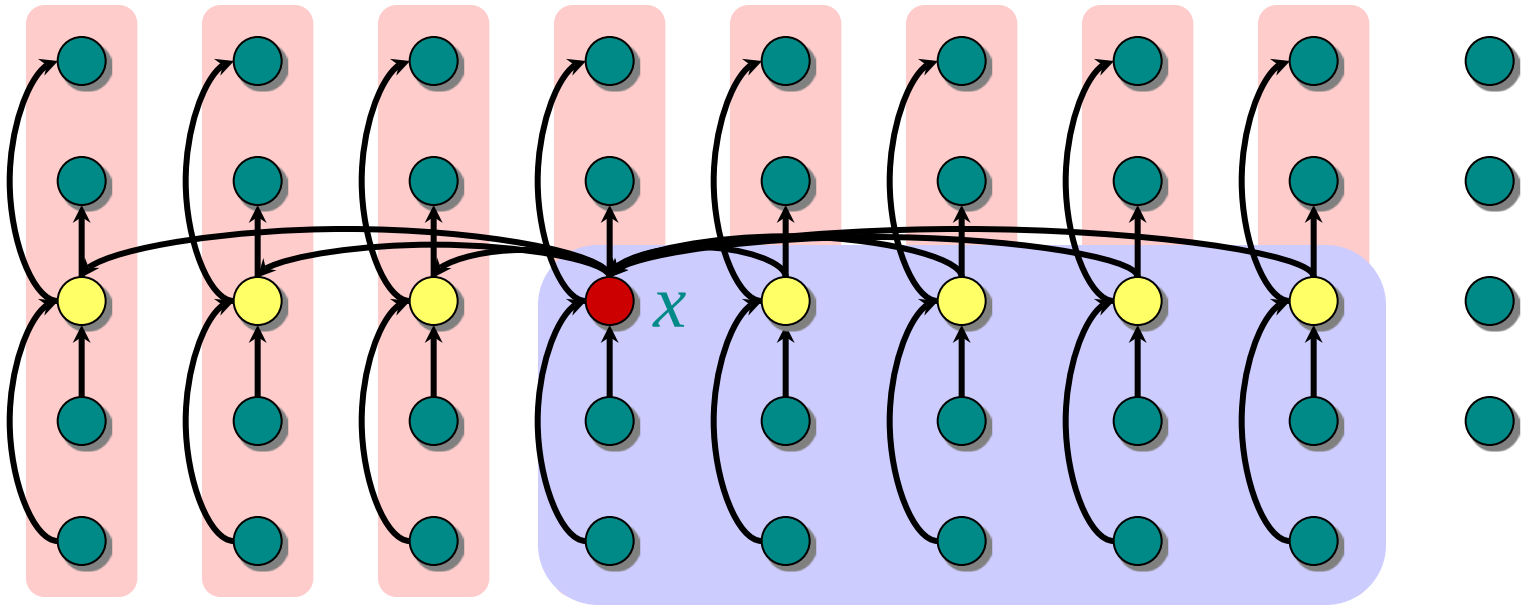
At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ group medians.

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



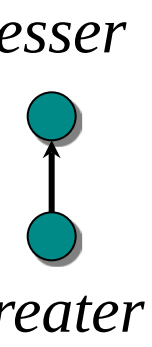


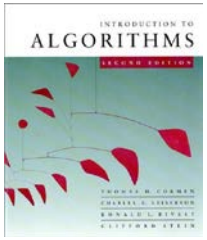
Analysis (Assume all elements are distinct.)



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.

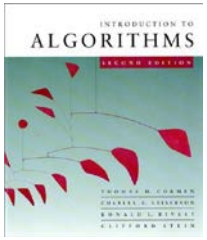
- Therefore, at least $3\lfloor n/10 \rfloor$ elements are $\leq x$.
- Similarly, at least $3\lfloor n/10 \rfloor$ elements are $\geq x$.





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 $T(n) = \Theta(1)$.



Developing the recurrence

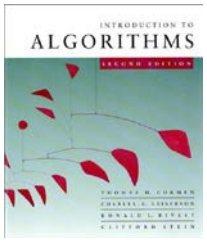
$T(n)$ SELECT(i, n)

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

$T(3n/4)$ 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

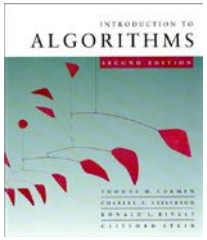
$$T(n) = T\left(\frac{1}{5}n\right) + T\left(\frac{3}{4}n\right) + \Theta(n)$$

Substitution:

$T(n) \le cn$

$$\begin{aligned} T(n) &\le \frac{1}{5}cn + \frac{3}{4}cn + \Theta(n) \\ &= \frac{19}{20}cn + \Theta(n) \\ &= cn - \left(\frac{1}{20}cn - \Theta(n)\right) \\ &\le cn \quad , \end{aligned}$$

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



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.

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

