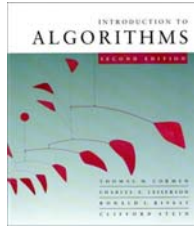




## CS 3343 -- Spring 2009



### Order Statistics

Carola Wenk

Slides courtesy of Charles Leiserson with small changes by Carola Wenk



## Order statistics

Select the  $i$ th 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  $i$ th element.

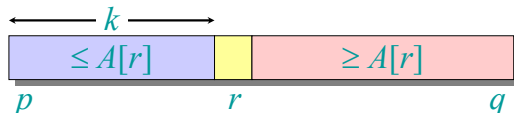
Worst-case running time =  $\Theta(n \log n) + \Theta(1)$   
=  $\Theta(n \log n)$ ,

using merge sort or heapsort (*not* quicksort).



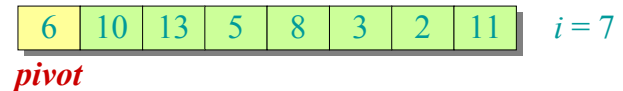
## Randomized divide-and-conquer algorithm

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

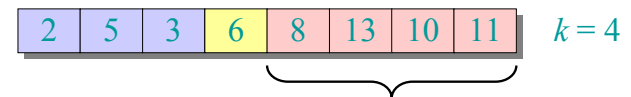


## Example

Select the  $i = 7$ th smallest:



Partition:



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



## Intuition for analysis

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

**Lucky:**

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

**Unlucky:**

$$\begin{aligned}
 T(n) &= T(n-1) + \Theta(n) & & \text{arithmetic series} \\
 &= \Theta(n^2)
 \end{aligned}$$

**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  $k = 0, 1, \dots, n-1$ , 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:

$$\begin{aligned}
 T(n) &= \begin{cases} T(\max\{0, n-1\}) + \Theta(n) & \text{if } 0 : n-1 \text{ split,} \\ T(\max\{1, n-2\}) + \Theta(n) & \text{if } 1 : n-2 \text{ split,} \\ \vdots \\ T(\max\{n-1, 0\}) + \Theta(n) & \text{if } n-1 : 0 \text{ split,} \end{cases} \\
 &= \sum_{k=0}^{n-1} X_k (T(\max\{k, n-k-1\}) + \Theta(n)) \\
 &\leq 2 \sum_{k=\lfloor n/2 \rfloor}^{n-1} X_k (T(k) + \Theta(n))
 \end{aligned}$$



## Calculating expectation

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

Take expectations of both sides.



## Calculating expectation

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

Linearity of expectation.



## Calculating expectation

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

Independence of  $X_k$  from other random choices.



## Calculating expectation

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

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



## Calculating expectation

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



## 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:**  $\sum_{k=\lfloor n/2 \rfloor}^{n-1} k \leq \frac{3}{8}n^2$  (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

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

Use fact.



## Substitution method

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

Express as **desired – residual**.



## Substitution method

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

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

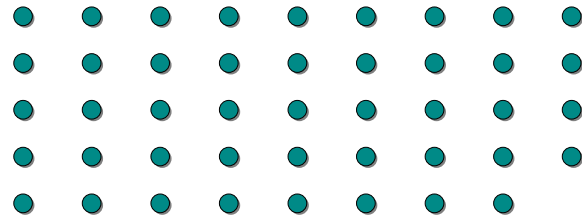
SELECT( $i, n$ )

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  $i$ th smallest element in the lower part  
**else** recursively SELECT the  $(i-k)$ th smallest element in the upper part

} Same as  
RAND-  
SELECT

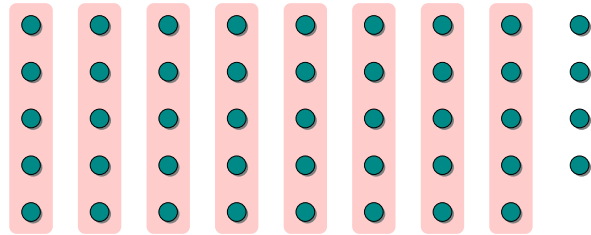


## Choosing the pivot





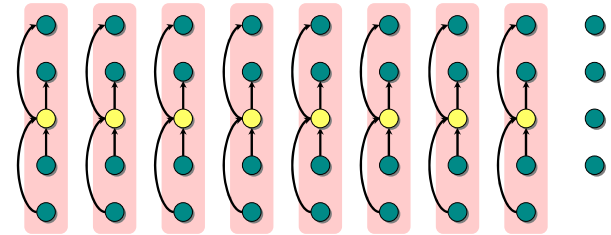
## Choosing the pivot



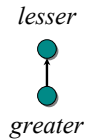
1. Divide the  $n$  elements into groups of 5.



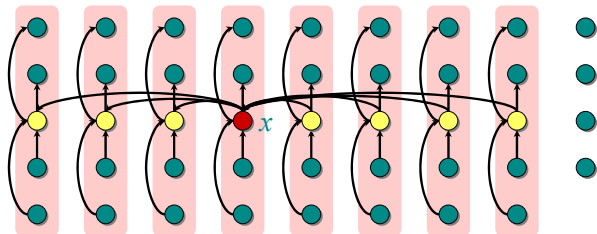
## Choosing the pivot



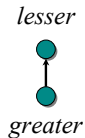
1. Divide the  $n$  elements into groups of 5. Find the median of each 5-element group by rote.



## 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.

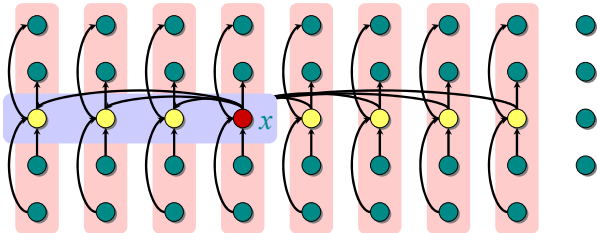


## Developing the recurrence

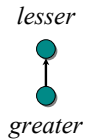
- $T(n)$  SELECT( $i, n$ )
- $\Theta(n)$  {
1. Divide the  $n$  elements into groups of 5. Find the median of each 5-element group by rote.
- $T(n/5)$  {
2. Recursively SELECT the median  $x$  of the  $\lfloor n/5 \rfloor$  group medians to be the pivot.
- $\Theta(n)$  {
3. Partition around the pivot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



## 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.



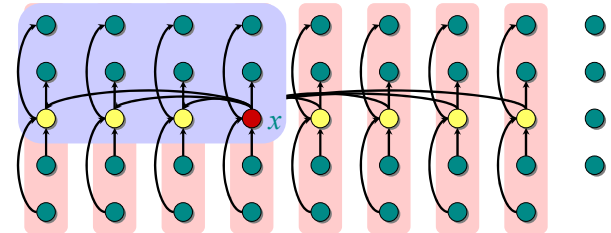
2/25/09

CS 3343 Analysis of Algorithms

25

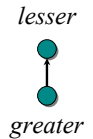


## Analysis (Assume all elements are distinct.)



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  $\leq x$ .



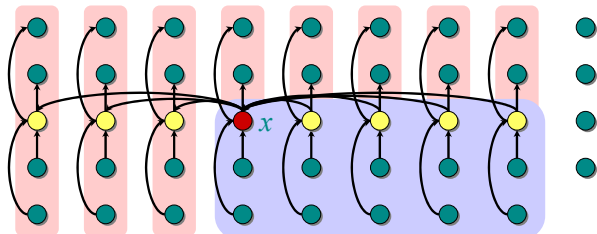
2/25/09

CS 3343 Analysis of Algorithms

26

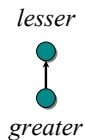


## Analysis (Assume all elements are distinct.)



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  $\geq x$ .



2/25/09

CS 3343 Analysis of Algorithms

27



## 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.

2/25/09

CS 3343 Analysis of Algorithms

28



## 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( $i, n$ )
$\Theta(n)$	<ol style="list-style-type: none"> <li>1. Divide the <math>n</math> elements into groups of 5. Find the median of each 5-element group by rote.</li> <li>2. Recursively SELECT the median <math>x</math> of the <math>\lfloor n/5 \rfloor</math> group medians to be the pivot.</li> </ol>
$T(n/5)$	
$\Theta(n)$	<ol style="list-style-type: none"> <li>3. Partition around the pivot <math>x</math>. Let <math>k = \text{rank}(x)</math>.</li> <li>4. <b>if</b> <math>i = k</math> <b>then return</b> <math>x</math>  <b>elseif</b> <math>i &lt; k</math>  <b>then</b> recursively SELECT the <math>i</math>th smallest element in the lower part  <b>else</b> recursively SELECT the <math>(i-k)</math>th smallest element in the upper part</li> </ol>
$T(7n/10 + 3)$	



## Solving the recurrence

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

**Substitution:**  $T(n) \leq c\left(\frac{1}{5}n - 4\right) + c\left(\frac{7}{10}n + 3 - 4\right) + dn$

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

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

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

$$\leq c(n - 4),$$

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

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



## 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.