

**Find K'st Smallest Element /
Order Istatistic**

Order Statistics:



Randomized Divide And Conquer



Analysis Of Expected Time



Worst-case Linear-time Order Statistics



Analysis

Order statistics



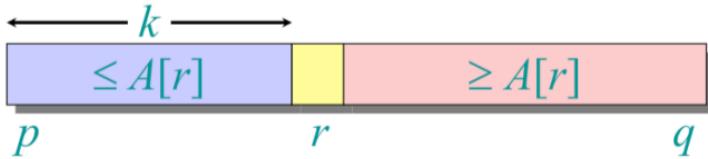
- Select the i th smallest of n elements (the element with rank i).
 - $i = 1$: minimum;
 - $i = n$: maximum;
 - $i = \lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$: median.
- **Naive algorithm:**

Sort and index i th element.

Worst-case running time = $\Theta(n \lg n) + \Theta(1) = \Theta(n \lg n)$,
using merge sort or heapsort (not quicksort).

Randomized Divide-and-Conquer Algorithm

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

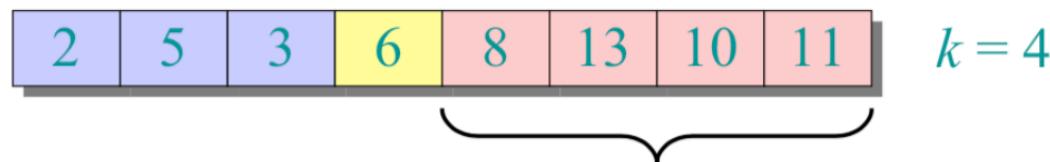


Example:

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:

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

Unlucky:

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

Worse than sorting!

Remember:

- Time Complexity: $O(n^2)$

Sum Of An Arithmetic Series

$$S_n = a + a + d + a + 2d + \dots + l - 2d + l - d + l$$

$$\underline{S_n = l + l - d + l - 2d + \dots + a + 2d + a + d + a}$$

$$\begin{aligned} 2S_n &= a + l + a + l + a + l + \dots + a + l + a + l + a + l \\ &= n(a + l) \end{aligned}$$

$$S_n = \frac{n}{2}(a + l) \quad \text{if we know } l$$

otherwise;

$$S_n = \frac{n}{2}\{a + a + (n-1)d\}$$

$$S_n = \frac{n}{2}\{2a + (n-1)d\}$$

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:

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

Calculating Expectation

- Take expectations of both sides.

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

Calculating Expectation

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

Linearity of expectation.

Calculating Expectation

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

Independence of X_k from other random choices.

Calculating Expectation

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

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

Calculating Expectation

$$\begin{aligned} E[T(n)] &= E\left[\sum_{k=0}^{n-1} X_k (T(\max\{k, n-k-1\}) + \Theta(n))\right] \\ &= \sum_{k=0}^{n-1} E[X_k (T(\max\{k, n-k-1\}) + \Theta(n))] \\ &= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(\max\{k, n-k-1\}) + \Theta(n)] \\ &= \frac{1}{n} \sum_{k=0}^{n-1} E[T(\max\{k, n-k-1\})] + \frac{1}{n} \sum_{k=0}^{n-1} \Theta(n) \\ &\leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \Theta(n) \quad \text{Upper terms appear twice.} \end{aligned}$$

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.

Substitution Method

Substitution Method

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

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

Substitute inductive hypothesis.

Substitution Method

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

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

SUMMARY OF RANDOMIZED ORDER-STATISTIC SELECTION

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

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

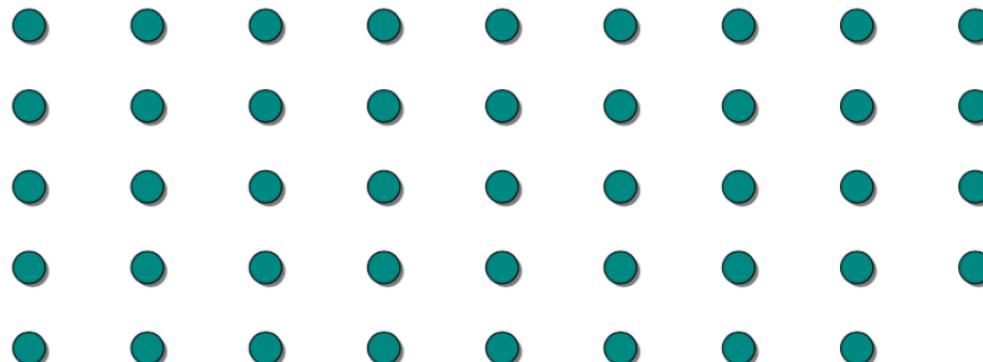
4. if $i=k$ then return x

elseif $i < k$

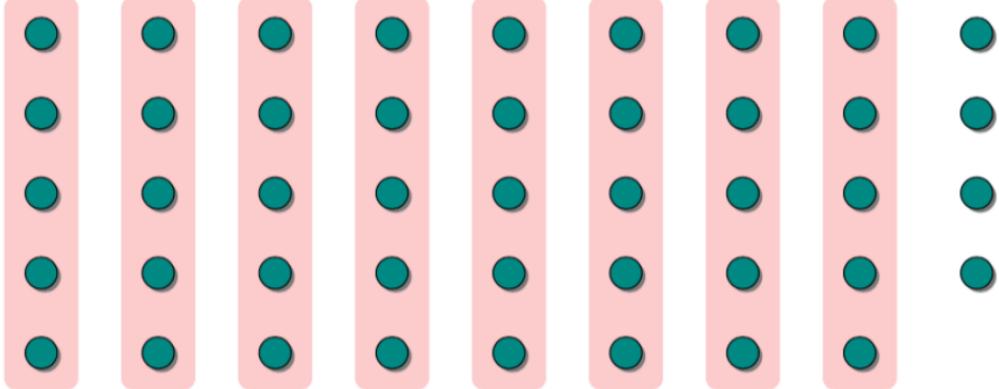
part
then recursively SELECT the ith smallest element in the lower

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

Choosing The Pivot

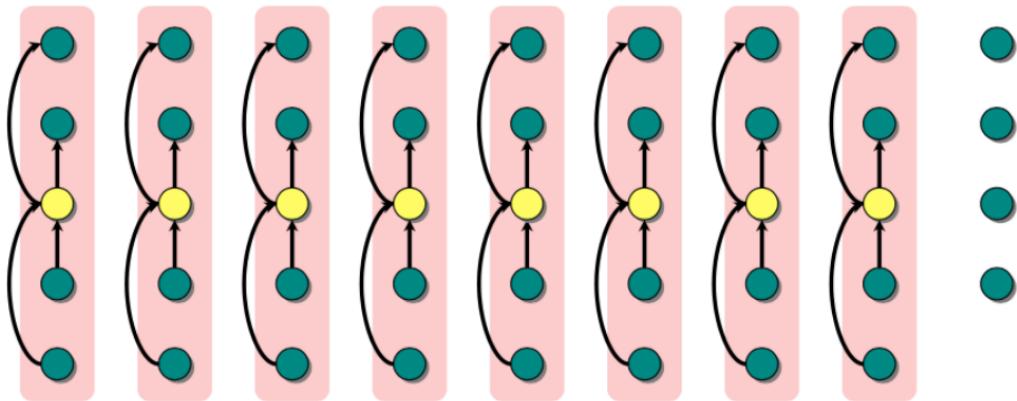


Choosing The Pivot



1. Divide the n elements into groups of 5.

Choosing The Pivot

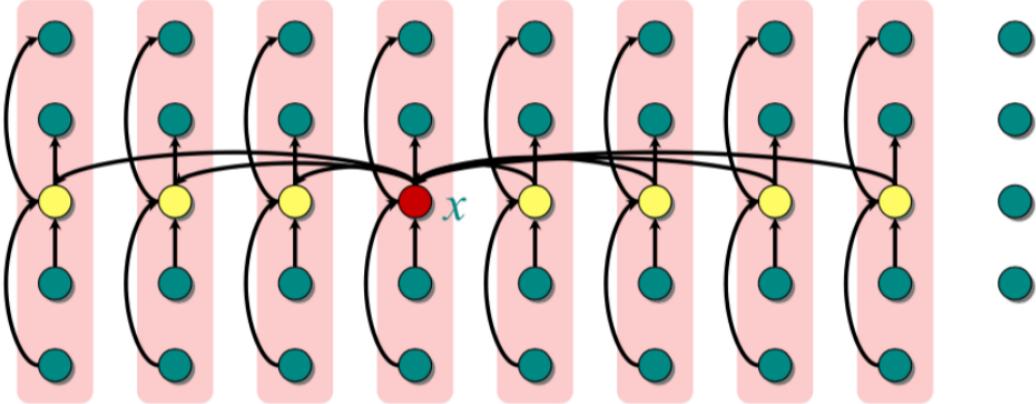


1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.

Choosing The Pivot

lesser

greater

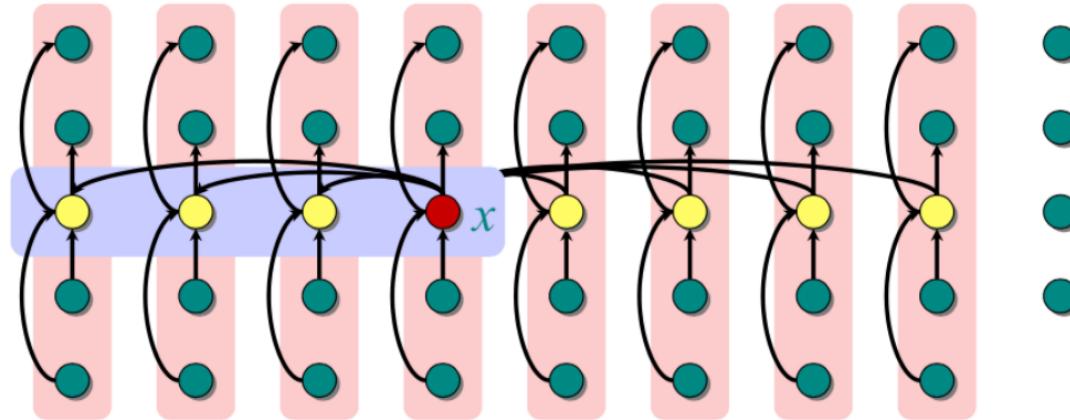


1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.
2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.

Analysis



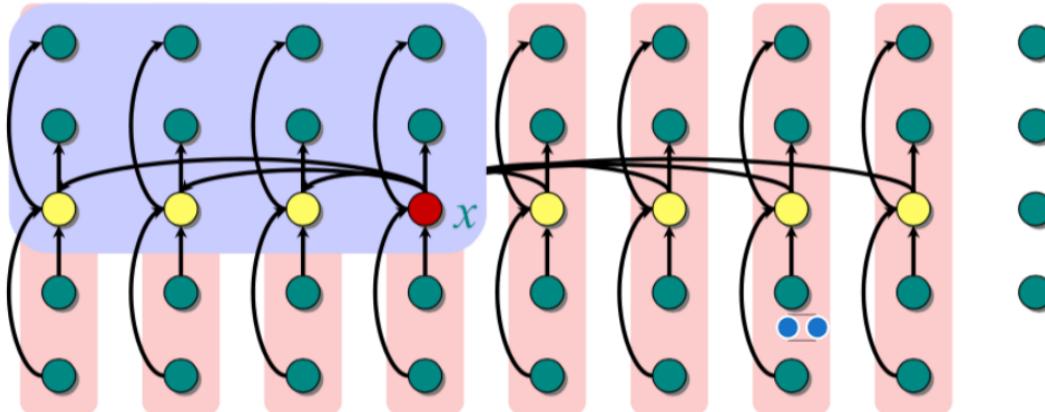
Analysis



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

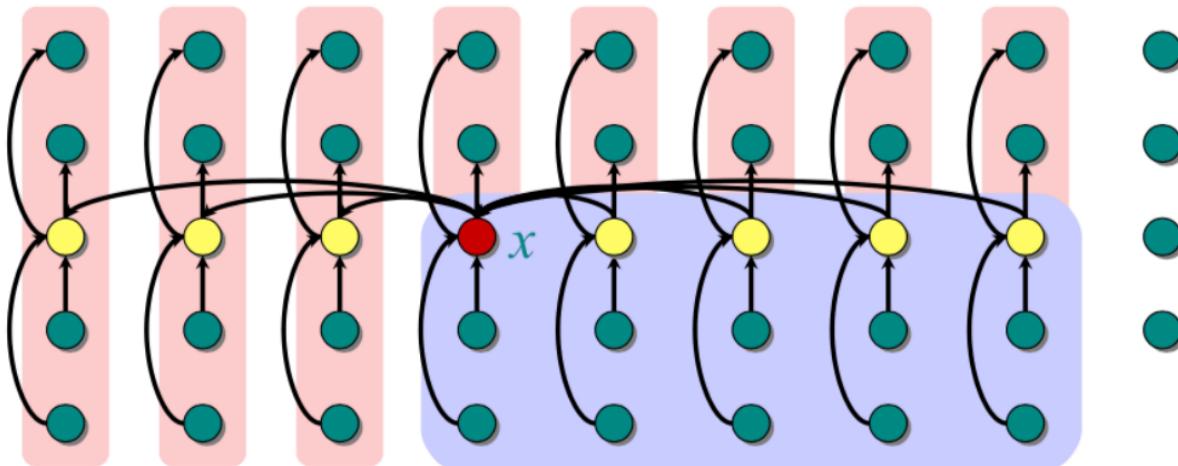
- Therefore, at least $3 \lfloor n/10 \rfloor$ elements are $\leq x$.



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 $\leq x$. •
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 i th smallest element in the lower part else recursively SELECT the $(i-k)$ th smallest element in the upper part



Solving The
Recurrence

Solving The Recurrence

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

Substitution:

$$T(n) \leq cn$$

$$T(n) \leq \frac{1}{5}cn + \frac{3}{4}cn + \Theta(n)$$

$$= \frac{19}{20}cn + \Theta(n)$$

$$= cn - \left(\frac{1}{20}cn - \Theta(n) \right)$$

$$\leq cn ,$$

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

Conclusions

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.