CS 430 Lecture 9 Activities

Opening Questions

1. Order Statistics: Select the ith smallest of n elements (the element with rank i)
   - $i = 1$: minimum; 1^{st} order statistic, $O(n)$
   - $i = n$: maximum; n^{st} order statistic, $O(n)$
   - $i = \lfloor \frac{n+1}{2} \rfloor$ or $\lceil \frac{n+1}{2} \rceil$: median Sort: $O(n \lg n)$, then take $A\left[\frac{n}{2}\right]$ for the median, $O(1)$

How fast can we solve the problem for various i values?

Randomized Algorithm for finding the ith Element

1. Think about partition (with a random choice of the pivot"median of 3") from quicksort. Can you think of a way to use that and comparing the final location of the pivot to i, and then divide and conquer?

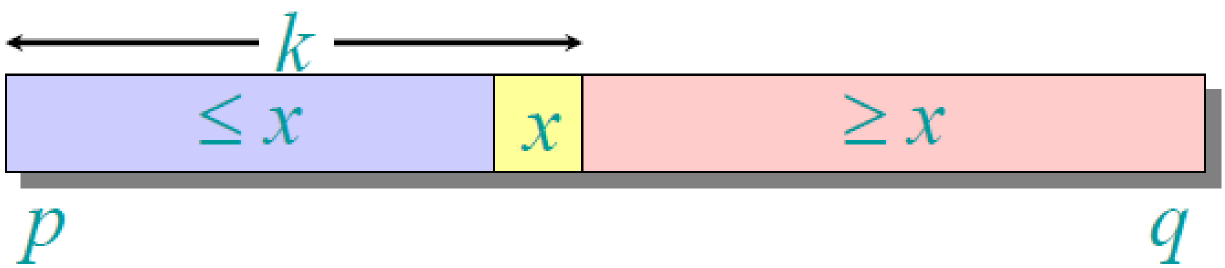


Figure 9.1:

Most cases $O(n)$, $A[k] = x$, the pivot. Three possibilities:

$i = k$ Done, $A[k]$ is the i smallest value

$i < k$ Redo Partition from indexes p to $k - 1$, because we know that the ith smallest must be smaller than the pivot at index k, still looking for the ith smallest.

$i > k$ Redo Partition from indexes $k + 1$ to q, not looking for the $i - k$ smallest, since we threw out k elements that we know are smaller than the ith element.

2. Demonstrate on this array to find i =th smallest element

6	10	13	5	8	3	2	11

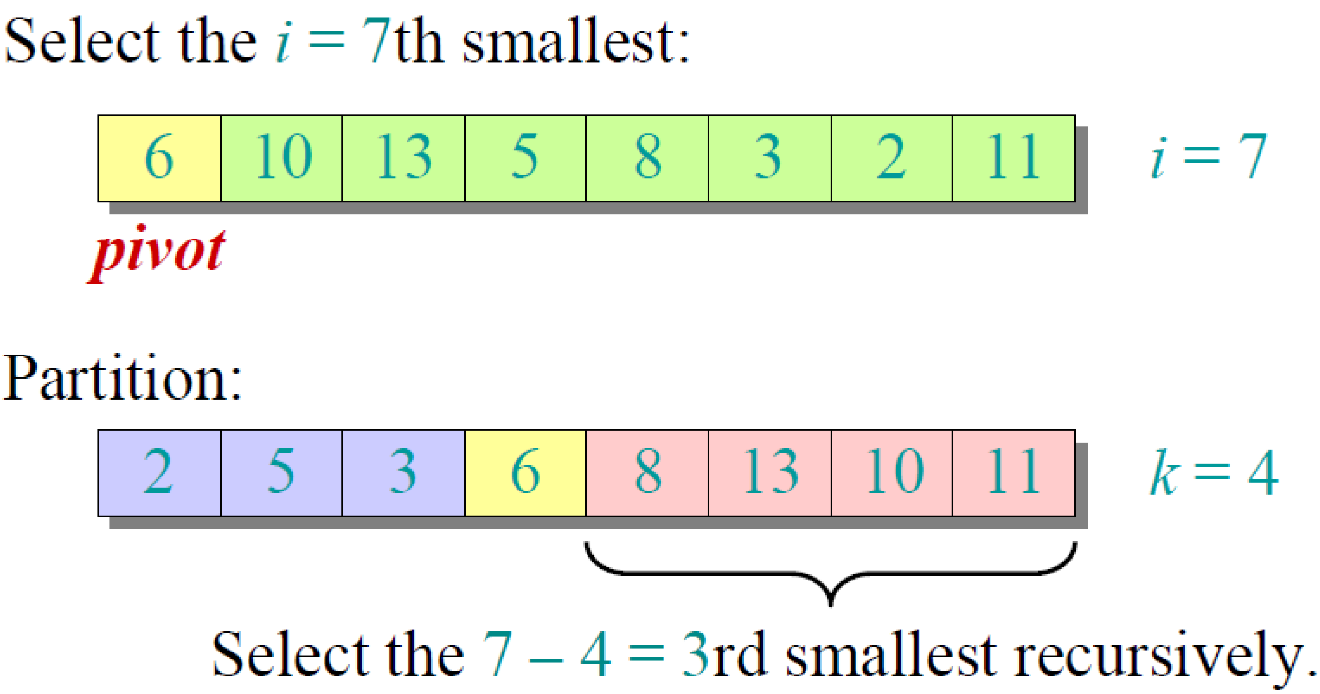


Figure 9.2:

Algorithm 9.1 Statistical Order to find the Smallest Element in Linear Time

function ORDERSTAT(A, p, q, i) ▷ Initial call: ORDERSTAT$(A, 1, n, i)$
 $k \leftarrow$ PARTITION(A, p, q) ▷ $O(\text{size } A \text{ from } p \to q)$
 if $i = k$ **then**
  return $A[k]$
 else if $i < k$ **then**
  return ORDERSTAT$(A, p, k-1, i)$
 else ▷ $i > k$
  return ORDERSTAT$(A, k+1, q, i-k)$
 end if
end function

Runtime Analysis:

$$\begin{aligned}
T(\text{size of subproblem}) &= T(n) \\
&= O(n) + T\left(\text{size of subproblem: } \frac{n}{2} \to n-1\right) \\
T(1) &= O(1) \\
\text{Master method} : T(n) &= aT\left(\frac{n}{b}\right) + f(n) \\
a &= 1 \\
b &= 2 \\
f(n) &= n^{\log_2(1)} = 0 \\
&= 1 \\
\Theta(f(n)) &= \Theta(n)
\end{aligned}$$

Worst case:

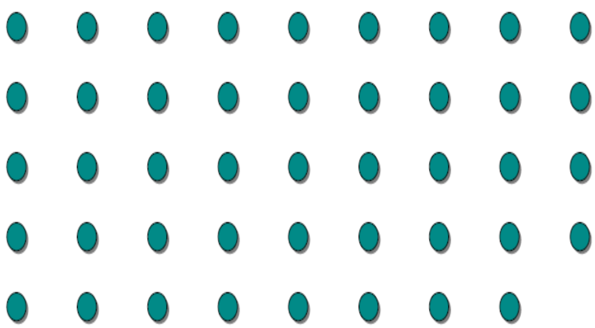
$$\begin{aligned}
T(n) &= T(n-1) + O(1) \\
&= O\left(\frac{n(n-1)}{2}\right) \\
&= O\left(n^2\right)
\end{aligned}$$

3. What is the worst case running time if you find the ith smallest element?
   Is there an algorithm to find the ith smallest element that runs in linear time in the worst case?

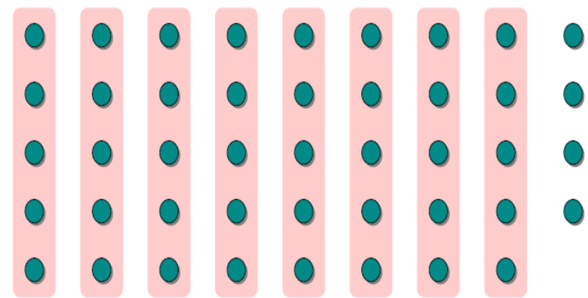
Algorithm 9.2 Select Smallest Element in Linear Time

```
function SELECT(i, n)
    Divide the n elements into groups of 5. Find the median of each 5-element group by hand.
    Recursively SELECT the median x of the ⌊n/5⌋ group medians to be the pivot.
    Partition around the pivot x. Let k =RANK(k).
    if i = k then
        return x
    else if 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
    end if
end function
```

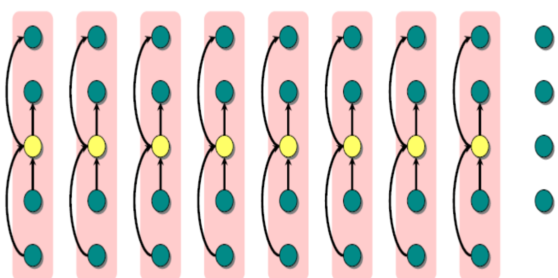
Choosing the pivot

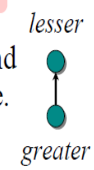


Choosing the pivot

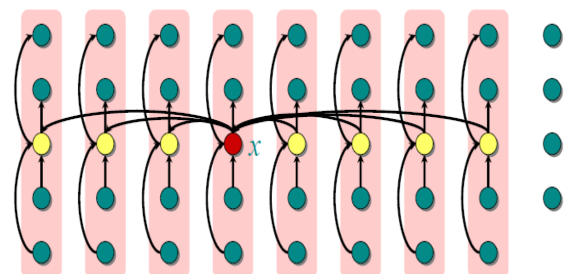


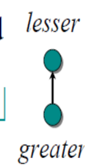
1. Divide the n elements into groups of 5.





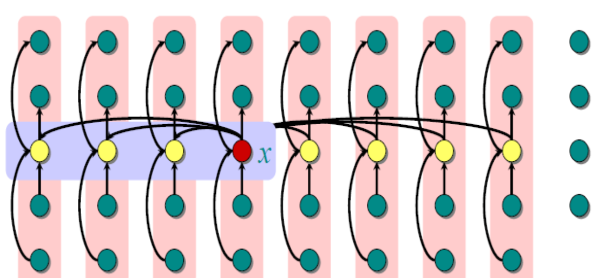
1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.





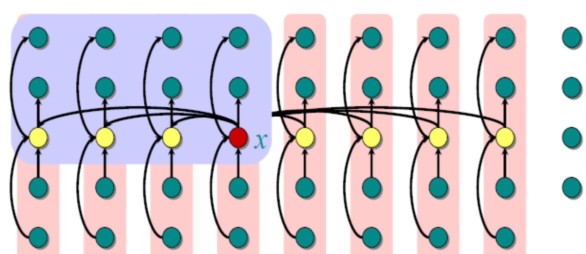
1. Divide the n elements into groups of 5. Find the median of each 5-element group by rote.
2. Recursively SELECT the median x of the $\lfloor n/5 \rfloor$ group medians to be the pivot.

Analysis



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

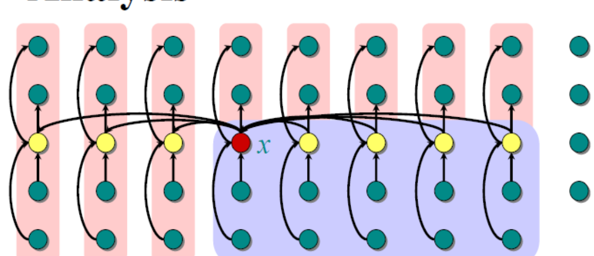
Analysis



At least half the group medians are $\le x$, 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 $\le x$.

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 **Substitution:** $T(n) \leq cn$

$$\begin{aligned}
T(n) &= T\left(\frac{1}{5}n\right) + T\left(\frac{7}{10}\right) + \Theta(n) \\
T(n) &\leq \frac{1}{5}cn + \frac{7}{10}cn + \Theta(n) \\
&= \frac{9}{10}cn + \Theta(n) \\
&= cn - \left(\frac{1}{10}cn - \Theta(n)\right) \\
&\leq cn
\end{aligned}$$

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

In practice, this algorithm runs slowly, because the constant in front of n is large.

Would we use this approach to find the median to partition around in Quicksort, and achieve in worst-case $\Theta(n \log n)$ time? **No**, too many calls to $O(cn)$ Select with a big c. We can do this if we need to find the median once or a couple of times, but not inside of Partition.