ORDER STATISTICS - MEDIAN FINDING, RANK SELECTION

Given $n$ unsorted elements, find the $k$-th smallest.

We will assume distinct elements.

$\Rightarrow$ easy $O(n)$ if $k = o(1)$ or $n = o(n)$.

In some sense median is hardest

Begin with finding the median

Assume data is in an array
Initially we are looking for the element with rank $r = \lceil \frac{n}{2} \rceil$

\[ \text{Select}(r, 1\ldots n) \quad \text{// find } r^{th} \text{ smallest # within array[1\ldots n]} \]

we will choose an element $x$

Compare all elements to $x \rightarrow$ compute $\text{rank}[x] = p$

if $\text{rank}[x] = p = r$, done, else use $x$ as pivot to partition input (set up binary search)

if $p > r$ // rank[$x$] > r, so search lower

\[ \text{Select}(r, 1\ldots p-1) \]

else // $p < r$, so search higher

\[ \text{Select}(r-p, p+1\ldots n) \]
Initially we are looking for the element w/ rank $r = \lceil \frac{n}{2} \rceil$

Select($r, 1,...n$)  // find $r$th smallest # within array[1,...n]

1) Form $\frac{n}{5}$ groups of 5 elements   // the last group can have $\leq 5$
2) Find median in each group   // brute force.
3) Recursively find $x =$ median-of-mediants   // don't worry about why, for now
4) Compare all elements to $x$  $\rightarrow$ compute rank[$x$] = $p$
5) if rank[$x$] = $p = r$, DONE, Else use $x$ as pivot to partition input (set up binary search)
6) if $p > r$  // rank[$x$] $> r$, so search lower
   Select($r, 1...p-1$)
else  // $p < r$, so search higher
   Select($r-p, p+1...n$)
1) Form $\frac{n}{5}$ groups of 5 elements $\Theta(n)$

really, just ignore these.
Lots of ways to handle.

e.g. could add three elements = $\infty$
OR
remove MAX & MAX-1
1) Form \( \frac{n}{5} \) groups of 5 elements \( \Theta(n) \)
2) Find median in each group (and re-organize) \( \frac{n}{5} \cdot \Theta(1) = \Theta(n) \)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>23</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td></td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td></td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

>12
>12
<12
<12
1) Form \( \frac{n}{5} \) groups of 5 elements \( \Theta(n) \)

2) Find median in each group (and re-organize) \( \frac{n}{5} \cdot \Theta(1) = \Theta(n) \)

3) Recursively find \( x = \text{median-of-medians} \) (and re-organize) \( T \left( \frac{n}{5} \right) \)

Re-organizing is not part of the algorithm. It's part of the proof. (although we could afford it)

That's the algorithm. Now to find \( T(n) \)
Let $x \rightarrow y$ mean $x > y$
\#"big" = \#"small" \geq 3 \cdot \frac{\frac{n/3}{2}}{\log n} \geq 3 \cdot \frac{n}{\log n}

\frac{n}{L^5} \rightarrow \frac{n}{s} \text{ if we ignore incomplete column}

\frac{n/5}{L^2} \rightarrow \frac{n/5}{2} \text{ if } n: \text{even}
"big" = "small" \geq 3 \cdot \frac{n^{5/3}}{2} \geq 3 \cdot \frac{n}{10} \geq \frac{1}{4} \cdot n \quad \text{For } n \geq 50

if \Box \text{ is not at the target rank/index, and we need to search lower (i.e., rank}(x) > \text{target), then recurse on all elements except "big" [symmetrically, if searching for target > rank}(x), recurse on all except "small"

\[ T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) + \Theta(n) \]

\text{steps 1\&2: split into groups & find medians of 5 & partition}

\text{find } x \quad \text{reurse if } \text{rank}(x) \neq \text{target}
\[ T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{3n}{4}\right) + \Theta(n) \]

Claim \( T(n) \leq c \cdot n \)

\[
\leq c \cdot \frac{n}{5} + c \cdot \frac{3n}{4} + d \cdot n
\]

\[
= \frac{19}{20} c n + d n = c n - \left(\frac{1}{20} c n - d n\right) \leq c n \text{ if } c > 20d
\]

QED