Cpt_S - 350

Homework 5

Abhilash Ambati

```
1:) void MINMAX ( int[] array) {

    int min = 0;
    int max = 0;

    if(array.length == 1) {
        max = array[0]
        max = array[0].
    }.
    if (array.length >= 2 && array.length % 2 == 0) {.
        if (array.length == 2 && array.[0] > array[1]) {.
            max = array[0];
            min = array[1];
        } else {
            max = array[1];
            min = array[0];
        }.
        for (int i = 2, i < array.length; i += 2) {.
            if (array[i] > array[i+1]) {.
                max = array[i];
            }. else if (array[i+1] < min) {
                min = array.[i+1];
            }
        }
} } }
```

1.

2. Average case complexity of S and A

   T_S = C $\times$ n (Since O(n) if the time complexity of linearselect, so C times n)

   T_A = $n \log n + C$ ($n \log n$ to sort the array and C to select the number)

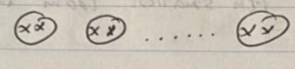
   $n \log n + C \leq$ C*n, which means T_T spends less time than T_S for some small C.

   As we select C more and more T_S is approaching n^2 which is much higher than $n \log n$. However, if C is small then T_S is better.

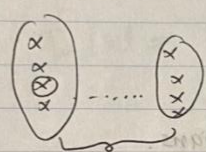
3.) $k=3$.

Suppose we are given an array $A[1,2,3\ldots n]$

Step 1: cut into groups of five:



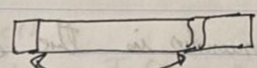
Step 2: Sort each group.



$n/3$ groups. and $n/3$ median.

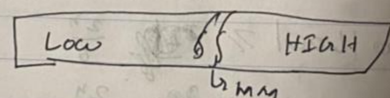
Step 3: Recursively linear select the $n/6$th smallest from of $n/3$ medians.

Step 4: Original array:



Swap mm with the first elmt in A

Step 5: run partition on A.



3.

Step 6: we want to ~~select~~ Select the i-th Samallest.

if $i == r$, return $A[i]$;

if $i < r$, return the result of, recursively using linear select for the i-th smallest from low;

if $i > r$, return the result of $(i-r)$ th Samallest from high

we totally have $\frac{n}{3}$ medians.
there are roughly $\frac{n}{6}$ medians $\le$ MM.

Each median is the middle element in a sorted group of 3 numbers.

there are 2 numbers in the group $\le$ the median

there are at least $2 \cdot \frac{n}{10}$ numbers in the orignal array of numbers that are $\le$ MM.

After partition, each element is low $\le$ MM.

$\Rightarrow |LOW| \geq \frac{2n}{6}$

$|HIGH| \geq \frac{2n}{6}$

note that, $|LOW| + |HIGH| \approx n.$

$\Rightarrow \max \{ |LOW|, |HIGH| \} \leq \frac{4n}{6}.$

<u>proof:</u> (for $T_w(n) = O(n)$)

1. write a formula

$$T_w(n) = T_w\left(\frac{n}{3}\right) + \max \{ T_w(|LOW|), T_w(|HIGH|) \} + O(n)$$

2. $T_w(n) = T_w(n/3) + T_w(\max\{|LOW|, |HIGH|\}) + a.$

guess $T_w(n) = O(n) \leq cn$ for some c.

$$\boxed{\begin{array}{c} I.H \quad \forall i < n \\ T_w(i) \leq c \cdot i \end{array}}$$

3. Check

$T_w(n) = T_w(n/3) + T_w(\max\{|LOW|, |HIGH|\}) + a \cdot n$

$\leq T_w(n/3) + T_w\left(\frac{4n}{6}\right) + a \cdot n$

$\leq c \cdot \frac{n}{3} + c \cdot \frac{4n}{6} + a \cdot n$

$cn + a \cdot n$

$\leq cn$ when $c \gg a.$

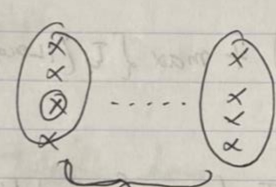
~~k~~ $k = 7$.

Suppose we are given an array. $A[1, 2, 3 \ldots n]$.

Step 1: cut into groups of five.

xx xx xx

Step 2: Sort each group.



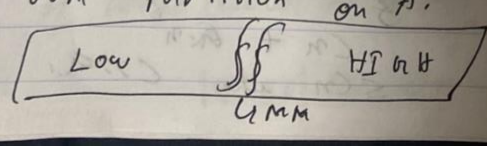
$n/7$ groups ~~&~~ and $n/7$ median.

Step 3: Recursively linear select the $n/14$ th smallest form of ~~n/7~~ $n/7$ median.

Step 4: Original array.

swap MM with the first element in A

Step 5: run partition on A.



step 6: we want to select the ~~i-th smallest~~

i-th Samallest

if $i == r$, return $A[i]$;

if $i < r$, return the result of, recursively using linear select for the i-th smallest from low;

if $i > r$, return the result of $(i-r)$th smallest ~~from~~ from high.

we totally have $n/7$ medians.

there are roughly $n/14$ medians $\leq$ mm

~~each~~ Each median is the middle element in a sorted group of ~~3~~ 7 numbers

there are 7 numbers in each group $\leq$ the median.

there are at least $7 \cdot \frac{n}{10}$ nombers in the original array of number that are $\leq$ MM

After partition each element is low $\leq$ mm

$\Rightarrow |Low| \geq \frac{4n}{14}$

$|HIGH| \geq \frac{4n}{14}$.

note that, $|LOW| + |HIGH| \approx n$.

$$\Rightarrow \max \{ |LOW|, |HIGH| \} \leq \frac{10n}{14}$$

proof : $($for $T_w(n) = o(n))$

1. write a formula

$$T_w(n) = T_w\left(\frac{n}{7}\right) + \max \{ T_w(|LOW|), T_w(|HIGH|) \} + o(n)$$

2. $$T_w(n) = T_w(n/7) + T_w(\max \{ |LOW|, |HIGH| \}) + a.$$

guess $T_w(n) = o(n) \leq Cn$ for some C.

$$\boxed{\begin{array}{l} I.H \ \forall i < n \\ T_w(i) \leq C \cdot i \end{array}}$$

3. check

$$T_w(n) = T_w(n/7) + T_w(\max \{ |LOW|, |HIGH| \}) + an.$$

$$\leq T_w(n/7) + T_w\left(\frac{10}{14} n\right) + a.n$$

$$\leq C \cdot \frac{n}{7} + C \frac{10n}{14} + a.n$$

$$\frac{6}{7} Cn + a.n$$

$$\leq Cn \text{ when } C >> a.$$

4)

	Best case	worst case	Avg
Quick ~~sort~~ select	$O(1)$	$O(n^2)$	$\Theta(n)$
linear select	$O(1)$	$O(n)$	$\Theta(n)$

$$\boxed{\text{low}\,(r-1) \;|\, r \,|\; \text{high}\,(n-r)}$$

worst - case

1. write formula

$$T_w(n) = \Theta(n-r) + O(r-1)^2 + O(n)$$

$$\cancel{T_w(n) = \max_{1 \le r \le n} \{ O(n-r) + O(r-1)^2 + O(n) \}}$$

$$\cancel{\le \max 1 \le r}$$

$$T_w(n) = \max_{1 \le r \le n} \{ O(n-r) + O(r-1)^2 + O(n) \}.$$

$$\le \max_{1 \le r \le n} \{ a(n-r) + a(r-1)^2 + O(n) \}.$$

$$F(r) = a(n-1) + a(r-1)^2 + \Theta(n)$$

$$F(r') = a + 2a(r-1)(1).$$

$$F(r'') = 2a$$

$2a > 0$ for $a > 0$

$\therefore$ $F(r)$ is concave up.

4.

$$\max_{1 \le q \le n} F(q) = \max \{F(1), F(n)\}.$$

$$= \max \{a(n-1) + an,\ a(n-1)^2 + an\}$$

$$\boxed{= a(n-1)^2 + an}$$

$$= a(n^2 - 2n + 1) + an$$

$$= an^2 - 2na + an + a.$$

$$= an^2 - \underbrace{an + a}_{\text{insignificant}}.$$

recursive $O(n^2)$

$$= an^2 \quad \text{where } a > 0.$$

average - case

$$T_{Avg}(n) = O(n-r) + O(r-1) + O(n)$$

$$= \frac{1}{n} \sum_{r=1}^{n} a(n-r) + a(r-1) + a(n)$$

$$\frac{2a}{n} \sum_{r=1}^{n} a(n-r) + \frac{1}{n} \sum_{r-1}^{n} , an$$

$$\frac{2a}{n} \sum_{r=1}^{n} (n-r) + \frac{1}{n} an + an.$$

~~let F(r)~~ = r. let $F(r) = r$.

$$= \frac{2a}{n} \sum_{r=1}^{n} F(n-r) + a_n.$$

$$= \frac{2a}{n} \int_0^n F(r)\, dr + an.$$

$$= \frac{2a}{n} \frac{n^2}{2} + a_n$$

$$= an + a_n$$

$$= 2a_n = O(n).$$

5.

5.) 5com

Before the array is sorted, the information on the ordering is $\log_2 n!$ bits.

each use of 5com decreases the amount by $\log_2$

after the array is sorted, the amount of information o

the ordering is zero.

$\therefore$ number of 5com uses is at least $\left(\log_2 n! / \log_2 5!\right)$