Aritra

MCS202304

for 2 question (All comparison, arithmetic operation are taken to be of constant time and $A = [0]*a$ return an initialized array of length a]

① arr[i] ... arr[n-1]

for question (c) I will be marking the time ~~complexity~~ or steps here will be explained in that part

2 (a) Orderstatistics (A, i): // T(n)

n = length (A) // n

if (n == 1) return A[0]; // 1

pivot = Getpivot (A) // T'(n)

100/100

L,E,G = Partition (A, pivot) , l = length (L), e = length (E), g = length (G) n

if (i < l) return OrderStatistics (L, i) // $T(\frac{7n}{10})$

else if (i < l+e) return pivot // 1

else return orderstatistics (G, i-(l+e)) // $T(\frac{7n}{10})$

Get pivot (A): T'(n)

B = Get Smallerlist (A) // O(b) (b = length of B) = $\lceil \frac{n}{5} \rceil$ = O(n)

No. This function takes $\Theta(n)$ time.

b = length (B) 1

if b is odd

return OrderStatistics (B, $\frac{b-1}{2}$) // $T(\frac{n}{5})$

else

return OrderStatistics (B, $\frac{b}{2}$) // $T(\frac{n}{5})$

~~Get Smallerlist (A):~~

?

if b is odd

[0 1 ... ($\frac{b-1}{2}$) ... b-1]

else, [0 1 ... ($\frac{b}{2}$) b-1]

Next page

```
Get Smallestlist (A):
    a = length (A)                       // 1
    ~~for (i=0, i~~
    B = [0] * [(a+4)/5]                  // a
    ~~A for~~
    ~~for (i=0, ...)~~

    let b = (clip a to multiple of 5.)
    and create a new array C st
        C = A[b : a-1]
    and A = A[0 : b-1]

    for (i=0; i<b; i+=5)                 // 36 b/5
        temp = [0] * 5                   // 5
        temp [0:4] = A [i : i+4]         // 5
        temp = Sort (temp)               // 25
        B[i/5] = temp [2]                // 1
    if length(C) ≠ 0, C = Sort(C), B [length (B)-1] = C [(length(C))/2]    // 4 [at most]
    return B.
```

40/40

modified

```
if length(C)) ≠ 0,
    C = Sort (C)
    B [length (B)-1] = C [length(C)/2]
```

(b) We will prove that our algorithm is correct by stating ② correctness of each of the function.

Orderstatistics (A, i) gives the value of $\hat{A}[i]$.
$\hat{A}$ sorted version of A.

considering $0 \le i \le$ length$(A)-1$

Now, if A has only one element, then A[0] should be the answer.

as, $\hat{A}[i]$ always exists.

Getpivot (A): gives a good candidate for the pivot. We will see that, the value returned by getpivot() will partition the array into two half (L, G) where neither of them are too big [length of E is not of much info in this context]

Now, after getting the pivot, we will partition the array into L, E, G

{when, L stores all the elements less than pivot
E " " " " equal to pivot
G " " " " greater than pivot}

Now, as i is 0-indexed,

(i<l) L has all the elements lesser than pivot and it has length l. So, all the elements L[0: l-1] is less than pivot, if i is less than l i.e [0, l-1] then we will find our ith largest element in L only.

So, we will call OrderStatistics (L, i)

$i < l+e$. So it will handle all the cases when $i = [l, e-1]$

we know all the elements in L are less than pivot and

they will appear in the first (0 : l-1) position.

Then all the pivot elements will take place in the (l : e-1) position

and $i \geq l$ [elseif]

So, if $i < l+e$, then we are guaranteed to get $\hat{A}[i]$ in this part.

So, we are returning the pivot element.

else $\boxed{i \geq e}$

we are now in the greater pivot region.

and we also know that, first $(l+e)$ elements [or $(0 : l+e-1)$]

are less than $\hat{A}[i]$.

So, in the G we will try to find the $i-(l+e)$ elements as

G doesn't have those $(l+e)$ elements.

So, we will call, orderstatistics(G, $i-(l+e)$)

So, Correctness of Orderstatistics done.

30/30

$\exists\ (c, n)$
~~So, T(n) = O~~
we need to prove by induction that $T(n) = O(n)$ ie $T(n) \le cn$ $\forall n \ge n_0$

we will induct on value of n.

Base case will ~~always~~ be ~~T(0)/T(1)~~ $T(1) = t$ and $T(0) = 0$. where t is a constant)

and it will terminate at each step we are dividing to $n/5 + \frac{7n}{10}$

Induction hypothesis:

$T(K) \le cK$ ~~for K>K_0~~

$\forall K > n_0$ ~~and for~~

is true for all $K = (0, n-1)$

30/30

$\therefore T(\frac{n}{5}) \le c\frac{n}{5}$

and $T(\frac{7n}{10}) \le c(\frac{7n}{10})$

$\therefore T(n) = T(\frac{n}{5}) + T(\frac{7n}{10}) + dn + K'$

$\le c\frac{n}{5} + c\frac{7n}{10} + dn + K'$

$\le cn - (n(\frac{c}{10} - d) - K')$

$\therefore T(n) \le cn$ if $n(\frac{c}{10} - d) - K' \ge 0$ for some c

let c be, $100d + 20K'$

$\therefore n(\frac{c}{10} - d) - K'$

$= n(10d + 2K' - d) - K'$ $= dn(9) + K'(2n-1)$

≥ 0.

Hence, $T(n) = O(n)$

So, we proved that,

Getpivot(A) returns a pivot. such that

atleast $3 \cdot \frac{\lceil \frac{n}{5} \rceil - 1}{2} + 2$ element of A are bigger / smaller and equal to ?. and equal to this element.

GOOD

now $\frac{3 \lceil \frac{n}{5} \rceil - 1}{2} + 2$

$= \frac{3 \lceil \frac{n}{5} \rceil + 1}{2}$

Now, GetSmallerList

first makes the array size to multiple of 5 and creating an array (C) from the rest.

By iterating over A by taking 5 elements of A at a time

it sorts that part and put the median in the result B.

It does the same with the (C) that was torn from A.

Then returns B.,

So, all the elements inside B will also be inside (A)

(c)

We know getPivot(A) returns a pivot p that ~~returns~~ is ~~always~~ atleast (greater than / less than) or equal to $\frac{3(\frac{n}{5}+1)}{2}$ elements of a.

So, pivot p will partition the array into three halfs L, E, G such that both L and G will have atmost $n - \frac{3(\frac{n}{5}+1)}{2}$

So, both side will have almost $\frac{7n}{10}$ elements $= n - \frac{3n + 15}{10}$

$10n \quad (3n+15) \quad -15$

Get Smallestlist runs in $O(a)$ time where a = size of input array

from the time complexity analysis of that function

we saw that. it takes steps

$$T''(a) = 1 + a + \frac{36b}{5} + 4$$
$$\leq 1 + \frac{36a}{5} + 4$$
$$= ca + d$$

$$T''(a) = O(a)$$

Let $T(n)$ be the time taken by orderStatistics (A, i) where $n = \text{length}(A)$

$T'(n)$ is the time taken by the get pivot function, n = size of A

$$T'(n) = \overset{O(n)}{O\left(\frac{n}{5}\right)} + 1 + T(n/5)$$ [it will call almost 1 of the orderstatistics with arr length $\frac{n}{5}$]

$$= O(n) + 1 + T(n/5)$$

also, $T(n) = n + 1 + T'(n) + T\left(\frac{7n}{10}\right)$ [it will call atmost one of the orderstatistics with almost array length $\frac{7n}{10}$]

$$\therefore T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n) + k'$$

$$\therefore \boxed{T(n) \leq T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + dn + k'}$$

and we proved that, it i.e. can almost have $\frac{7n}{10}$ elements

A good estimate will be $T(n) = O(n)$

Correctness of Getpivot:

(b)

claim(1):- getpivot return a pivot element that exists in the array

30/30

Proof:- Get Smaller list return a smaller version of the array ie B and each of the element of B will be inside of (A) [we will prove it next]

So, getpivot then takes B and returns the median of B, that is present in the array ✓

claim 2: Getpivot return a GOOD pivot.

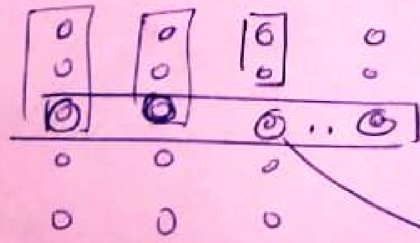
Proof

Getsmallerlist(A) return the smaller version of A by parting it into block of 5 and then returning the median of those each blocks.

and, getpivot return the median of ~~the part~~ B by calling orderStatistics (B, b/2) or (B, $\frac{b-1}{2}$)

Now. $(\lceil \frac{n}{5} \rceil - 1)/2$ such blocks + 2 elements

A



$\lceil \frac{n}{5} \rceil$ such blocks

let this be returned from (pivot) the Getpivot function returns pivot

But, $3\left(\frac{\lceil \frac{n}{5} \rceil - 1}{2}\right) + 2$ are atleast less than or equal to this

at-least ✓

3 elements

blocks

Similarly, we can claim that $3\lceil \frac{n}{5} \rceil$ ...

(1) (a)

⑤

```
ADD (X, Y):
    x = length (X)
    y = length (Y)

    ~~Z = [0] x (x+y)~~ [initialize

    ~~for (i=0, i<x+y, i++)~~
    reverse (X), reverse(Y)  [reverses the array in place]
    ~~x = max~~
    z = max (x, y)

    Z = [0] * (z)

    ~~for (i=0, i<z~~
        carry = 0
    for (i = 0, i<z, i++)

        current = 0
        if i < x,
            current += x

        if i < y
            current += y

        current += carry
        if (current > 1)
            current = 0
            carry = 1
        else
            carry = 0
        Z [i] = current
```

PTO

~~Reverse (z)~~

If carry = 1 :

n = z+1

N = [0] * n

N [0:z] = z [0:z]

N [z] = 1

reverse(N)

return N

else.

reverse (z)

return |z|

reverse (A):

n = length (A)

for. (i=0, i < $\frac{n}{2}$, i++):

A [i] = A [n-1-i].