Name : Pradeep. P

Roll No : 20161145

Assignment → Algorithms

1.

For the given problem,

It is similar to mergesort but instead the problem is divided into 3 parts rather than '2'.

$\Rightarrow T(n) = K + T(n/3)$

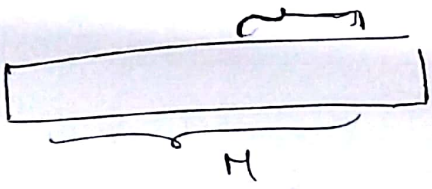
$a = 1 \quad b = 3$

$n^{\log_b a} = 1 = \theta(1) = n^0$

$\Rightarrow$ By applying masters theorem,

$T(n) = \theta(n^0 \log_3 n) = \theta(\log_3 n)$

2. The given list is of size M, given that among them 's' elements are not sorted.



Let n_i be the number belonging to the set 's' which is arbitarily choosen.

While running the loop the maximum number of element should be checked with n_i during insertion sort = (M-1)

[If n_i is at last in unsorte

array and is least among all elements.]

Since n_i is arbitarily choosen element,

$$T(n) = s \times O(n)$$

$$T(n) = O(sn)$$

3.

1. $T(n) = n! \times O(n)$

since $f(n)$ is to be calculated everytime while running the for loop.

$n! \leq n \cdot \ldots \cdot n$ (n times)

$\Rightarrow n! = O(n^n)$

$\Rightarrow T(n) \leq n! \times kn$

$\Rightarrow T(n) \leq k(n+1)! \leq k n^{n+1}$

2. Similar to above,

$T(n) = O(n) \cdot n \leq kn^2 = O(n^2)$

3. $T(n) = O(n^2) \cdot n \leq kn^3 = O(n^3)$

4. $T(n) = O(1)$, the loop will not run.

4.

1. $T(n) = \sqrt{n}\, T(\sqrt{n}) + 100n$

divide with n

$$\frac{T(n)}{n} = \frac{T(\sqrt{n})}{\sqrt{n}} + 100$$

$n = 2^m \Rightarrow m = \log n$

$$\frac{T(2^m)}{2^m} = \frac{T(2^{m/2})}{2^{m/2}} + 100 \qquad \text{let } f(m) = \frac{T(2^m)}{2^m}$$

$$\rightarrow f(m) = f(m/2) + 100$$

$$\Rightarrow f(m) = \theta(100 \log_2 m) \quad [\because \text{By master theorem}]$$

$$\Rightarrow T(2^m) = \theta(2^m 100 \log_2 m)$$

$$\Rightarrow T(n) = \theta(n \log_2 \log_2 n)$$

$$\Rightarrow T(n) = \theta(n \log\log n)$$

3.

$T(n) = T(n-2) + \log n$

$T(n) \rightarrow \log n$

$T(n-2) \rightarrow \log(n-2)$

$T(n-4)$ ⋮ } number of terms $= n/2 - 1$

$\log(2)$

$\Rightarrow T(n) = \log n + \log n-2 + \cdots + (n/2 - 1)$ terms

$\Rightarrow T(n) = \log(n \cdot n-2 \cdot n-4 \cdots)$ $(n/2 - 1)$ terms

we know that

$n \cdot n-2 \cdot n-4 \cdots \leq n \cdot n \cdot \cdots$

$\Rightarrow T(n) \leq k \log(n \cdots (n/2-1) \text{ times})$

$\Rightarrow T(n) \leq k \log(n^{n/2-1})$

$\Rightarrow T(n) \leq k_1 n \log n$

$\Rightarrow T(n) = O(n \log n)$ ---(1)

Also,

$n \cdot n-2 \cdot n-4 \cdots \geq k(n/2 \cdots)$ $(n/4 - 1)$ times

$\Rightarrow n \cdot n-2 \cdot n-4 \cdots \geq k\left(n/2\right)^{n/4}$

$\left(\Rightarrow n \cdot n-2 \cdots = \Omega\left((n/2)^{n/4}\right)\right.$

Apply log on both sides

$\Rightarrow \log(n \cdot n-2 \cdots) \geq \log\left((n/2)^{n/4}\right)$

$\Rightarrow \log(\log(n \cdots)) \geq (n/4)\log n/2$

$\Rightarrow T(n) \geq k\, n/4 \log n/2$

$\Rightarrow T(n) \geq k_1\, n\log n$

$\Rightarrow T(n) = \Omega(n\log n) \cdots (2)$

from (1) and (2)

$T(n) = \Theta(n\log n)$

5.

1. Sudo code:

from the given pt is obvious that 'm' is largest elemen

in the array. lowerIndex ↑ , ↑ largerIndex

get_max(A[], l_1, l_2):

$mid = l_1 + \frac{l_2 - l_1}{2}$

if A[mid] > A[mid+1] && A[mid] > A[mid-1]:

return mid

```
else if A[mid] > A[mid-1]:
    get-max(A[], mid+1, h2)
else
    get-max(A[], l1, mid-1)
return.
```

The above algorithm will give the required answer.

2.

Since the above problem is a divide and conquer algorithm,

$T(n) = T(n/2) + \theta(1)$ [∵ only half of the array was checked that of previous one]

$a=1 \quad b=2$

$n^{\log_2 1} = n^0 = 1 = \theta(1)$

By applying masters theorem

$T(n) = \theta(\log n)$

3.

Loop Invariant:

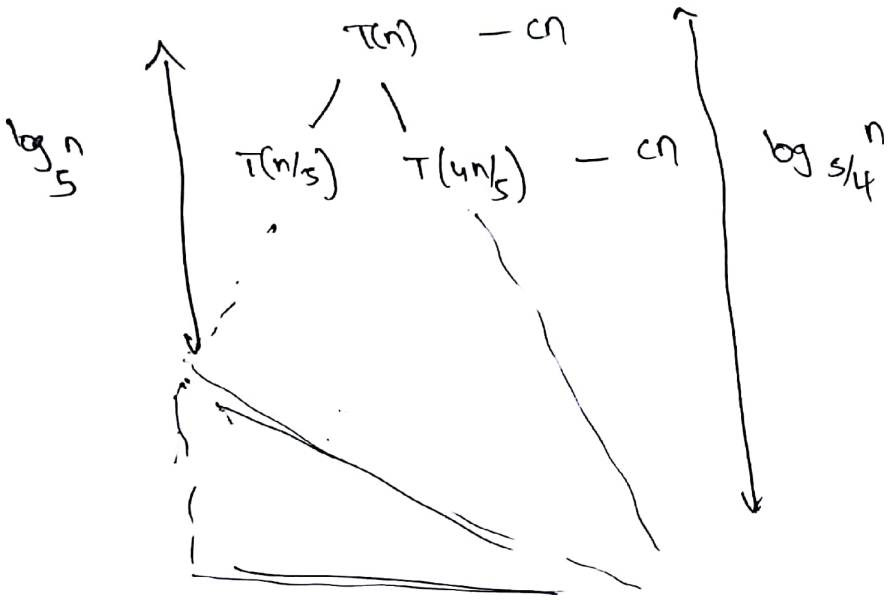
In the recursion tree at any stage the maximum element lies between the indices l_1 and l_2.

In every recursion if the middle element is larger than its neighbourhood elements then that is the required answer.

4.

2. $T(n) = T(n/5) + T(4n/5) + \theta(n)$

solving by using reccurence tree method



at every level sum is cn.

$\therefore$ $T(n) \leq cn \times \log_{5/4} n$ [∵ considering height as $\log_{5/4} n$ for Big O]

$\Rightarrow T(n) = O(n \log n)$ — ①

similarly if we take height as $\log_5 n$

then

$cn \log_5 n \leq T(n)$

$\Rightarrow T(n) = \Omega(n \log n)$ — ②

from ① and ②

$T(n) = \theta(n \log n)$