ID: 20101266
Sec - 10

1. Time complexity

a. sorted from smaller to greater,

$\log(\log(n)) < \log(n) < \sqrt{n} < n < n\log n < n^{3/2} < n^2$

$n^2 < n^2 \log n < n^3 < 2^n < e^{(n+1)} < n!$.

~~b. Writing all the conditions,~~
~~$f(n) = O(g(n))$ if~~

b i, ii. conditions for Θ,

$f(n) = \Theta(g(n))$ iff, $\exists$ +v constants c_1, c_2, n_0

such that, $c_1 * g(n) \leq f(n) \leq c_2\, g(n)$; $n_0 \geq 1$

PTO

1b i) $n^2 + 15n - 3 = \Theta(n^2)$ for $n \geq 1$

$1 \times n^2 \leq n^2 + 15n - 3 \leq n^2 + 15n^2 - 3n^2$

$\underbrace{1}_{c_1}\,\underbrace{n^2}_{g(n)} \leq n^2 + 15n - 3 \leq \underbrace{13}_{c_2}\,\underbrace{n^2}_{g(n)}$

$\therefore \Theta(n^2)$. [using theory].

$4n^3 - 7n^2 + 15n - 3 = \Theta(n^3)$ for $n \geq 1$.

1b ii) $4n^3 - 7n^2 + 15n - 3 \leq 4n^3 + 15n^3$

$1 \times n^3 \leq 4n^3 - 7n^2 + 15n - 3 \leq 19n^3$

$\underbrace{1}_{c_1}\,\underbrace{n^3}_{g(n)} \leq 4n^3 - 7n^2 + 15n - 3 \leq \underbrace{19}_{c_2}\,\underbrace{n^3}_{g(n)}$

$\therefore \Theta(n^3)$. [using theory] biii) $T(n) = 4T(n/2) + n$

using masters theorem for ~~decreasing~~ dividing functions,

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \quad , \quad f(n) = \theta(n^k \log^p n)$$

$a = 4$ $\qquad$ $n^k \log^p n = n$

$b = 2$ $\qquad$ $\therefore p = 0 \,,\ K = 1.$

$$\log_b a = \log_2 4 = 2 > K.$$

So, θ is $n^{\log_b a}$

$$\theta\left(n^{\log_2 4}\right) \Rightarrow \theta(n^2)$$

b iv). $T(n) = 2T(n/2) + n^3 = \Theta(n^3)$

Using masters theorem for dividing functions,

$\log_b^a = \log_2^2 = 1$ | $K = 3, P = 0$.

$\log_b^a < K$ and $p = 0$,

thus,

Θ is $n^K \log^P n$

$\Theta(n^3 \log^0 n) = \Theta(n^3)$.

b v) $T(n) = T(n/4) + T(5n/8) + n = O(n)$.

Computing both $T(n/4)$ and $T(5n/8)$ separately,

Time for $T(n/4) + n$,

$a = 1, b = 4$ $f(n) = n$

$K = 1, P = 0$

$\log_4^1 = 0 < K$,

So, Θ is $n^K \log^P n$

$\Theta(n^1 \log^0 n) = \Theta(n)$ —— (1).

Scanned with CamScanner

Time for $T\left(\frac{n}{8/5}\right) + n$, $f(n) = n$

Here, $a = 1$, $b = 8/5$ $\quad \therefore k = 1, p = 0$.

$\log_{8/5} 1 = 0 < k,$

So θ is also $n^k \log^p n$

$= \theta(n)$.

For $T(n)$ we considered $f(n)$ twice so

$T(n)$ is $O(n)$.

$= O(n)$.

vi) $T(n) = T(n/3) + T(4n/9) + n$

for, $T(n/3) + n$, $\quad f(n) = n$

$K = 1, p = 0$.

$a = 1$ $\quad b = 3$

$\log_3 1 < K$ and $p = 0$

So, θ is $n^k \log^p n \rightarrow \theta(n)$.

for $T\left(\frac{4n}{9}\right) + n$, $f(n) = n$

$a = 1, b = 9/4$ $\quad$ $K = 1, p = 0$

$\log_{9/4} 1 < K$ and $p = 0$,

So, θ is $n^k \log^p n = \theta(n)$.

As for $T(n)$ we considered $f(n)$ twice, so

$T(n)$ has $O(n)$.

1.e 1)

```
count = 0;
for (i=1, i<=n; i*=2)      —— log_2 n.
    for (j=1, j<=i ; j++)
        count ++           —— 2^(log_2 n+1) - 1 => nlogn.
```

i	j
1	1
2	2
2^2	2^2
2^3	2^3
⋮ K times.	⋮
2^K	2^K

count ++ runs for,

$j \Rightarrow 1 + 2^1 + 2^2 + 2^3 + \cdots + 2^K$ times

$\Rightarrow 2^0 + 2^1 + 2^2 + \cdots 2^K$

$\Rightarrow 2^{K+1} - 1$ times.

Assume, $i > n$

$2^K > n$

$K = \log_2 n$.

$2^K - 1$.

$2 \cdot 2$

$2n - 1$

so, $j \Rightarrow 2^{\log_2 n + 1} - 1$

$j \Rightarrow 2 \cdot 2^{\log_2 n} - 1$

$\Rightarrow 2n - 1 \quad [2^{\log_2 n} = n]$

so, $\underline{O(n\log_2 n)} \Rightarrow n$

1 c. 2)

p = 3 —— 1

while (p<n) —— $\log_2 \log_3 n$.

p = p * p.

p	
3	$= 3^{2^0}$
9	$= 3^{2^1}$
81	$= 3^{2^2}$
k times. ⋮	⋮
	$3^{(2^k)}$

Assum,

$p \geq n$

$3^{(2^k)} = n$

$2^k = \log_3 n$

$k = \underline{\log_2 \log_3 n}$.

$$\underbrace{1}_{c_2} \times \underbrace{\log_2 \log_3 n}_{g} \leq \log_2 \log_3 n \leq \underbrace{10}_{c_2} \underbrace{\log_2 \log_3 n}_{g(n)}$$

So $\underline{\theta(\log_2 \log_3 n)}$.

1d. 1)

```
int ternary-search (int l, int r, int x)  ——— T(n)
{
  if (r >= l)  ——————> 1 unit
  { int mid 1 = l + (r-l)/3;  ———> 1 unit
    int mid 2 = r - (r-l)/3;  ———> 1 unit
    if (ar [mid 1] == x)  ———> 1 unit
          return mid 2;
    if (x < ar [mid 1])
         return ternary-search (l, mid1-1, x);      \
    else if (x > ar [mid 2])                         |
         return ternary-search (mid 2+1, r, x);      |— T(n/3)
                                                     |
    else                                             |
    return ternary search (mid+1, mid 2-1, x);      /
  }
}
```

$T(n) = T(n/3) + 4$

$T(n) = T(n/3) + 1$ [taking the constant part as 1].

d ii) $T(n) = T\left(\frac{n}{3}\right) + 1$ —— (1)

$T(n) = T\left(\frac{n}{3^2}\right) + 2$ —— (2)

$T(n) = T\left(\frac{n}{3^3}\right) + 3$ —— (3)

⋮ K times later

$T(n) = T\left(\frac{n}{3^K}\right) + K.$

Assume, $\frac{n}{3^K} = 1.$

$3^K = n$

$K = \log_3 n.$

$T(n) = T(1) + \log_3 n$

$\therefore \; \Theta(\log_3 n).$

Scanned with CamScanner

Searching.

1a)

```
a1, b1 = input().split('') # length of ar1, ar2.
ar1 = input('give ar1').split('')
ar2 = input('give ar2'). split('')
def khujo(ar1, l, r, key):
    if l > r:
        return -1

    else:
        mid = (l+r)//2
        if (ar1[mid]<=key) and (ar1[mid+1]>key):
            return mid+1
        elif (ar1[mid]>key) and (ar1[mid-1]<=key):
            return mid
        elif (ar1[mid]<key) and (ar1[mid+1]<key):
            return khujo(ar1, mid+1, r, key)

        else:
            return khujo(ar1, l, mid-1, key).
```

```
store = [ ]
a1 = int (a1)
b1 = int (b1)
for a in range (b1):
        key1 = ar2 [a]
        if key1 = = ar1 [a1 - 1]:
                store.append (a1)

        else :
                x = khujo (ar1 , 0, a1 - 1, key = key1)
                store.append (x).
```

```
print (store)
```

O/P → 4 , 2 , 4 , 2 , 5

1b. Considering the search part of my code:

```
def khujo(arr1, l, r, key):  —— T(n)
    if l > r:  ——— 1
        return -1
    else:
        mid = (l+r)//2  ——— 1
        if (arr1[mid] <= key) and (arr1[mid+1] > key):  — 1
            return mid+1
        elif (arr1[mid] > key) and (arr1[mid-1] <= key):  — 1
            return mid
        elif (arr1[mid] < key) and (arr1[mid+1] < key):
            return khujo(arr1, mid+1, r, key)
        else:
            return khujo(arr1, l, mid-1, key).  → T(n/2)
```

$$T(n) = T(n/2) + 4$$
$$= T(n/2) + 1.$$

Solving $T(n)$,

$T(n) = T(n/2) + 1.$, $f(n) = 1.$

Using master theorem,

$a = 1$, $b = 2$ $K = 0$, $P = 0$

$\log_2 1 = 0 = K$ and $P = 0$,

So, θ is $n^K \log^{P+1} n$

$n^0 \log^1 n$

$\Rightarrow \theta(\log n)$ Showed.

Now, our for loop runs for n times, so time complexity for my code is,

$n \times \log n \Rightarrow \theta(n \log n)$.

here n is the number of elements in arr 2.