2401331530118
KESHAV KRISHNA

Data Structure & Algorithm

DATE

Q1

i)
```
void func (int n) {
    int i = 1, s = 1;
    while (s <= n) {
        i++;
        s = s + i;
        print ("*");
    }
}
```

start
i = 1 , s = 1
i = 2 ⇒ s = 3
i = 3 ⇒ s = 6
i = 4 ⇒ s = 10
⋮

$s = \frac{(k+1)(k+2)}{2} \leq n \Rightarrow O(\sqrt{n})$

$\frac{k^2}{2} \approx n \Rightarrow k^2 = 2n$

$\Rightarrow k = \sqrt{n}$

ii)
```
for (i=1; i*i <= n; i++) {
    ≡
}
```

$i * i \geq n$

$i^2 \geq n$

$i = \sqrt{n} \Rightarrow O(\sqrt{n})$

iii)
```
void func(int n) {
    int i, j, k, count = 0;
    for (i = n/2; i <= n; i++)          O(n)
        for (j = 1; j + n/2 <= n; j++)  O(n)
            for (k = 1; k <= n; k = k * 2)   O(log2 n)
                count++;
}
```

1 → $i = n/2 + 0$
2 → $i = n/2 + 1$
3 → $i = \frac{n}{2} + 2$
⋮
R → $i = \frac{n}{2} + k$
$= n$

$\frac{n}{2}, R - 1 = n$

$R = n/2 + 1 \Rightarrow O(1)$

for overall ⇒ $n \times n \times \log_2 n$

(iv)

```
func c (int n) {
    if (n==1) return ;                    ⟶ 1
    for (i=1; i≤n ; i++) {                ⟶ (n+1)
        for (j=1 ; j≤n; j++) {            ⟶ n(n+1)
            printf ⟶ (1)                  ⟶ n²+n+n+3
    }}
```

↳ $O(n^2)$

(v)

```
func (int n) {
    if (n==1) return;
    for (int i=1 ; i<=n; i++)
        for (int j=1 ; j<=n ; j++)
            printf ("*");
    func (n-3) ;
```

first loop → O(n)
2nd loop → O(n)

$T(n) = T(n-3) + O(n^2)$
$= n^2 + (n-3)^2 + T(n-6)$
$= n^2 + (n-3)^2 + (n-6)^2$

$T(n) = \sum_{i=0}^{n} (n-3)^2$

$O(n^3)$

Q2

(a) Given $\phi(n)$ is $O(f(n))$ means there exist five constant c and n_0 such that

$0 \le \phi(n) \le c \cdot f(n) \quad \forall \quad n \ge n_0$

If we multiply $\phi(n)$ by a constant the inequality become.

$d \cdot \phi(n) \le a \cdot c f(n)$

deriving c new const $\Rightarrow c' = ac$.

$d(n) \le c' f(n)$, still holds then proves that constant multiple doesnot changes BigO.

(b) If $d(n) = O(f(n))$ & $d(n) = O(g(n))$
then $d(n) = c(n) \longrightarrow O(\max(f(n), g(n)))$

$d(n) = O(f(n)) \longrightarrow d(n) \leq c_1 f(n) \; \forall \; n \geq n_1$
$c(n) = O(g(n)) \longrightarrow e(n) \leq c_2 f(n) \; \forall \; n \geq n_2$
$\max(n_1, n_2)$ & letting $c_3 = \max(c_1, c_2)$
then sum $d(n) + c(n)$
can be bounded as
$d(n) + c(n) \leq c_3 f(n) + c_3 g(n)$
$= c_3 (f(n) + g(n))$

(c) given $d(n) = O(f(n)) \longrightarrow d(n) \leq c_1 f(n) \; \forall \; n \geq n_1$ (i)
$e(n) = O(g(n)) \longrightarrow e(n) \leq c_2 (g(n)) \; \forall \; n \geq n_2$ (ii)
$e(n) \times$ (i)
$\hookrightarrow d(n) \cdot e(n) \leq c_1 f(n)) \cdot c_2 (g(n))$
$d(n) \cdot e(n) \leq (c_1 c_2)(f(n) \cdot g(n))$
let $c_3 = c_1 \cdot c_2$ also a constant
$n \geq \max(n_1, n_2)$ we have
$d(n) \cdot e(n) \leq c_3 f(n) \cdot g(n)$.

Q3
(i) $f(n) = 3n + 8$
$3n + 8 \leq 3n + 8n$
$\leq 11n \longrightarrow O(n)$

(ii) $f(n) = n^2 + 1$
$n^2 + 1 \leq n^2 + n^2$
$\leq 2n^2 \longrightarrow O(n^2)$

3. $f(n) = n^4 + 100n^2 + 50$
$\leq n^4 + 100n^4 + 50n^4$
$\leq 151 n^4 \longrightarrow O(n^4)$

4. $f(n) = 2n^3 - n^2$
$\leq 2n^3 - n^3$
$\leq n^3 \longrightarrow O(n^3)$

5. $f(n) = n$

Q4

(i) $f(n) = 5n^2 \longrightarrow \Omega(n^2)$

(ii) $f(n) = 100n + 5 \longrightarrow \Omega(n)$

(iii) $f(n) = \frac{n^2}{2} - \frac{n}{2} \Rightarrow \frac{n^2}{2} \gg \frac{n}{2}$ (upper)

upper $\rightarrow O(n^2)$
lower $\rightarrow \Omega(n^2)$
$\theta = \theta(n^2)$

(iv) $n \neq \theta(n^2)$

If $n = \theta(n^2) \Rightarrow c_1 \cdot (n^2) \leq n \leq c_2 \cdot n^2 \quad \forall n \geq n_2$

eqn $\Rightarrow n^2 \leq n \Rightarrow c_1 n \leq 1 \Rightarrow n \leq \frac{1}{c_1}$

↳ ~~not~~ true <<

now, no c_1 exist for $c_1 n^2 \leq n$.

(v) $6n^3 \neq O(n^2)$

If $6(n^3)$.

3. $f(n) = n^4 + 100n^2 + 50$

$\leq n^4 + 100n^4 + 50n^4$

$\leq 151 n^4 \longrightarrow O(n^4)$

4. $f(n) = 2n^3 - n^2$

$\leq 2n^3 - n^3$

$\leq n^3 \longrightarrow O(n^3)$

5. $f(n) = n$

Q4

(i) $f(n) = 5n^2 \longrightarrow \Omega(n^2)$

(ii) $f(n) = 100n + 5 \longrightarrow \Omega(n)$

(iii) $f(n) = \frac{n^2}{2} - \frac{n}{2} \Rightarrow \frac{n^2}{2} \gg \frac{n}{2}$ (upper)

upper $\rightarrow O(n^2)$

lower $\rightarrow \Omega(n^2)$

$\theta = \theta(n^2)$

(iv) $n \neq \theta(n^2)$

If $n = \theta(n^2) \Rightarrow c_1 \cdot (n^2) \leq n \leq c_2 \cdot n^2 \quad \forall n \geq n_2$

eqn $\Rightarrow n^2 \leq n \Rightarrow c_1 n \leq 1 \Rightarrow n \leq \frac{1}{c_1}$

↳ not true $\ll n_0$

now, no c_1 exist for $c_1 n^2 \leq n$.

(v) $6n^3 \neq O(n^2)$

If $6(n^3)$.

Q: i) $T(n) = 2T(n/2) + n \log n$

$T(n) = a \cdot T(n/b) + n^k \log^p n$

$a = 2, \ b = 2, \ k = 1, \ p = 1$

$a \geq 1, \ b \geq 1, \ k \geq 0, \ p \rightarrow$ real

$a = 2 \Rightarrow b^k = 2$

$T(n) = O(n^{\log_b a} \log^{p+1} n)$

$= O(n^{\log_2 2} \log^{-1} n) = O(n \log^2 n)$

(ii) $T(n) = 6T(n/3) + n^2 \log n$

$a = 6, \ b = 3, \ k = 2, \ p = 1$

$a < b^k \Rightarrow p \geq 0$

$O(n^k \log^p n) \Rightarrow O(n^2 \log n)$

(iii) $T(n) = 64T(n/8) - n^2 \log n$

$a = 64, \ b = 8, \ k = 2, \ p = 1$

$a = b^k, \ p \geq -1$

$T(n) = O(n^{\log_8 64} \log^{p+1} n)$

$= O(n^2 \log^2 n)$

(iv) $T(n) = \sqrt{2}\, T(n/2) + \log n$

$a = \sqrt{2}, \ b = 2, \ k = 0, \ p = 1$

$a > b^k, \ p = 1$

$T(n) = O(n^{\log_2 \sqrt{2}}) = O(n^{\sqrt{2}})$

(v) $T(n) = 3T(n/3) + \sqrt{n}$

$a = 3, \ b = 3, \ k = 1/2, \ p = 0$

$a > b^k, \ p \geq 0$

$O = (n \log_3^3) = O(n)$

6. $T(n) = T(1/n) + 1$.

$a = 1$, $b = 1$ — master theorem not applicable

Q6

i) $$T(n) = \begin{cases} 3T(n-1) & n > 0 \\ 1 & \text{otherwise} \end{cases}$$

$T(n) = 3T(n-1)$.

$T(n-1) = 3T(n-2)$

$T(n) = 3[3T(n-2)] = 9T(n-2)$ — (2)

$T(n-2) = 3T(n-3)$

$T(n) = 9[3T(n-3)] = 27T(n-3)$ — (3)

⋮

till k^{th} term

↳ $3^k T(n-k)$

$n - k = 0$

$n = k$

$O(3^n)$

ii) $$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ T(n-1) + n(n-1) & n \geq 2 \end{cases}$$

$T(n) = T(n-1) + n(n-1)$

$T(n-1) = T(n-2) + (n-1)(n-2)$

$T(n) = T(n-2) + n(n-1) + (n-1)(n-2)$

$T(n-2) = T[n-3] + (n-2)(n-3)$

$T(n) = T(n-3) + [n-3](n-1) + (n-2)(n-1) + n(n-1)$

$T(n) = T(n-k) + (n-k)(n-k+1) + (n-k+1)(n-k+2) + \cdots + n(n-1)$

$T(n) = 1$ if $n = 1 \Rightarrow T(1) = 1$

$n - k = 1 \Rightarrow k = n-1$

putting $k = n-1$

$T(n) = T(1) + 1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 + \cdots + n(n-1)$

$= 1 + \sum_{i=1}^{n} i(i-1)$

$= 1 + \sum_{i=1}^{n} i^2 - \sum_{i=1}^{n} i$

$= 1 + \frac{n(n+1)(2n+1)}{6} - \frac{n(n+1)}{2}$

$\Rightarrow T(n) = O(n^3)$

(iii)

$T(n) = T(n-1)$ & $T(1) = O(1)$

$T(n) = T(n-1)$

$T(n-1) = T(n-2)$

$T(n-2) = T(n-3)$

$T(n) = T(n-k)$

$n - k = 1$

$k = n-1$

$T(n) = T(1)$

$T(n) = O(1)$

(iv)

$T(n) = T(n-1) + \gamma_n$

$T(n-1) = T(n-2) + \gamma_{n-1}$

$T(n) = \left[T(n-2) + \frac{1}{n-1}\right]_{k-1} + \gamma_n$

$T(n) = T(n-k) + \sum_{i=0} \frac{1}{n-1}$

$\because n - k = 1$

$k = n-1$

$T(n) = T[n-(n-1)] + \sum_{i=0}^{n-2} \frac{1}{n-i}$

$= T(1) + \frac{1}{2} + \frac{1}{2} + \cdots \frac{1}{n}$

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

(4) $T(n) = \begin{cases} 2T(n/2) + 2 & \text{if } n > 2 \\ b & \text{if } n = 2 \end{cases}$

$T(n) = 2T(n/2) + 2$

$T(n/2) = 2T(n/2^2) + 2$

$T(n) = 2T(n/2^2) + 2 + 2$

$T(n/2^2) = 2T(n/2^3) + 2$

$\therefore T(n) = 8T(n/2^3) + 8 + 4 + 2$

$\vdots$ k

$= 2^k \cdot T(n/2^k) + 2^k + 2^{k-1} + \cdots + 2$

$\frac{n}{2^k} = 2 \Rightarrow n = 2^{k+1}$

$= 2^k T(2) + 2^k + 2^{k-1} + \cdots + 2^3 + 2^2 + 2$

$= 2^k \cdot b + \underbrace{2^k + 2^{k-1} + \cdots + 2^2 + 2}_{\text{GP}}$

$= a\frac{(r^{k}-1)}{r-1} = 2\frac{(2^{k}-1)}{2-1}$

$= 2^{k+1} - 2$

$= n - 2$

$\therefore n = 2^{k+1}$

$n = 2^k \cdot 2$

$2^k = n/2$

$T(n) = 2^k \cdot b + n - 2$

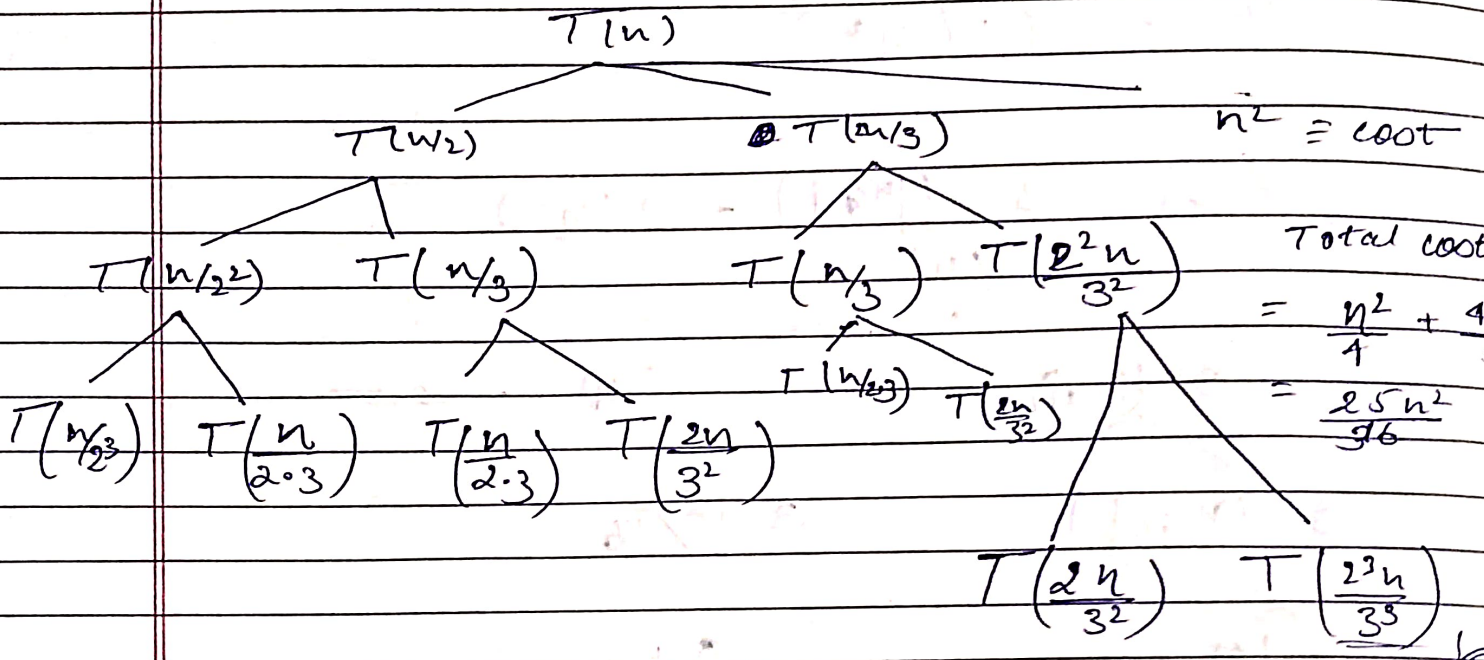
we know $n = 2^{k+1}$ $\log n = k+1$

$T(n) = \frac{n}{2} n + n-2$

$\hookrightarrow O(n)$.

Q5

i) $T(n) = T(n/2) + 2T(2n/3) + n^2$



Total cost

$= \frac{n^2}{1} + \frac{4n^2}{9}$

$= \frac{25n^2}{16}$

Now total cost)

$\frac{n^2}{16} + \frac{n^2}{9} + \frac{n^2}{9} + \frac{16n}{81}$

$= \frac{81n^2 + 144n^2 + 144n^2 + 256n}{81 \times 16} = \frac{625}{1296}$

$= \frac{5^4 n^2}{6^4}$

for k^{th} level.

$\left(\frac{2}{3}\right)^k n = 1$

$n = \left(\frac{3}{2}\right)^k \Rightarrow k = \log_{3/2} n$

Total cost $= n^2 + \frac{5^2}{6^2} n^2 + \frac{5^1}{6^1} n^1 + \cdots k'$

$= n^2 \left(1 + \frac{5^2}{6^2} + \frac{5^1}{6^1} + \cdots + \right)$

$= \left[\frac{a(r^n - 1)}{(r-1)} = @\, n^2 \left(\frac{1 - (5^2/6^2)^{\log_3 n}}{1 - 5/6} \right) \right.$

$\downarrow$ approx.

$= n^2$

$T(n) = O(n^2)$

(ii) $T(n) = 3T(|n/4|) + O(n^2)$

$T(n)$

$3T(|n/4|)$ $\qquad$ $n^2 \rightarrow$ cost

$3T(|n/16|)$ $\qquad$ $3n^2/16$

$3T(|n/43|)$ $\qquad$ $\frac{9n^2}{16}$

∞ Depth $\Rightarrow \left(\frac{1}{4}\right)^k n = 1 \Rightarrow k = \log_4 n$

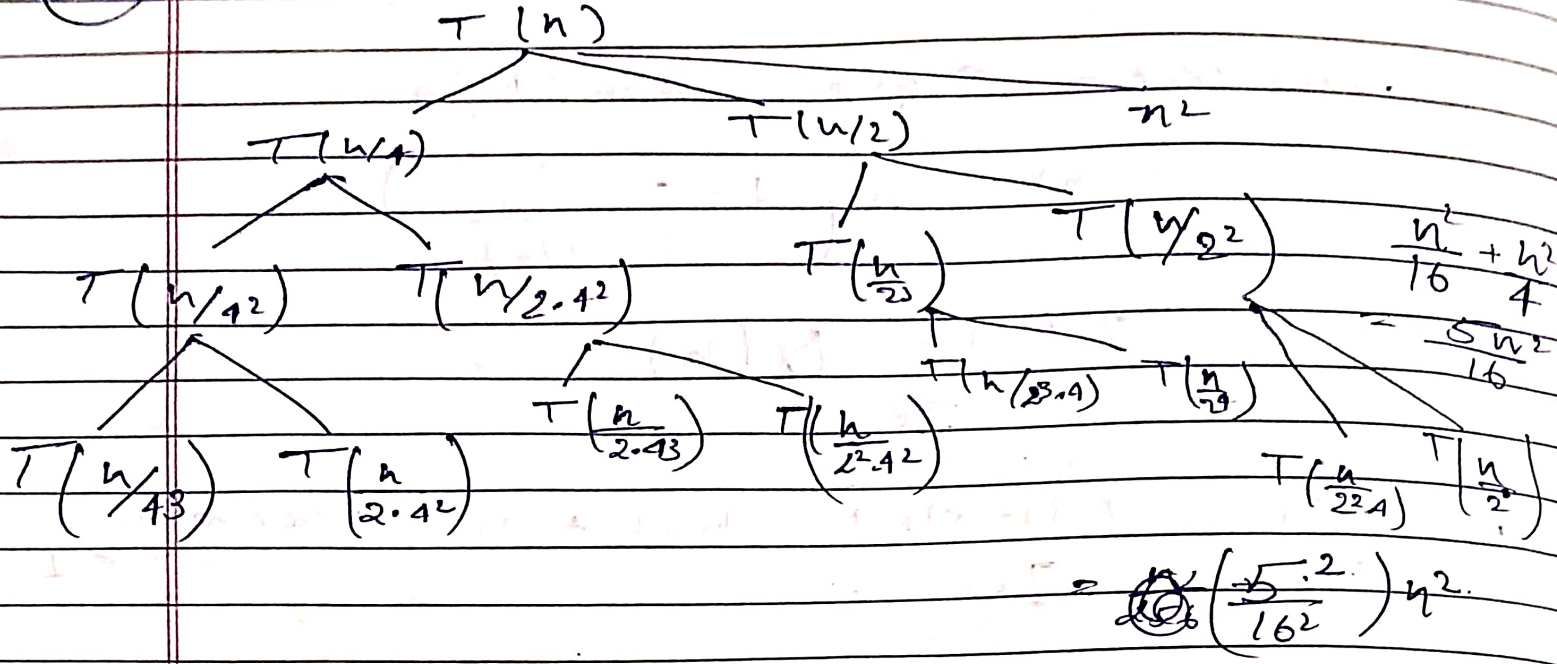
Total cost $\Rightarrow T(n) = n^2 + \frac{3n^2}{4^2} + \frac{3^2 n^2}{4^4} + \cdots^k$

$= n^2 \left(\frac{1 - (3/4^2)^{\log_4 n}}{1 - 3/4^2} \right)$

$\approx n^2$

$T(n) = O(n^2)$

(iii) $T(n) = T(n/4) + T(n/2) + n^2$



$= \left(\frac{5}{16}\right)^2 n^2$

depth :- $\left(\frac{1}{2}\right)^k n = 1 \Rightarrow k = \log_2 n$

Total cost

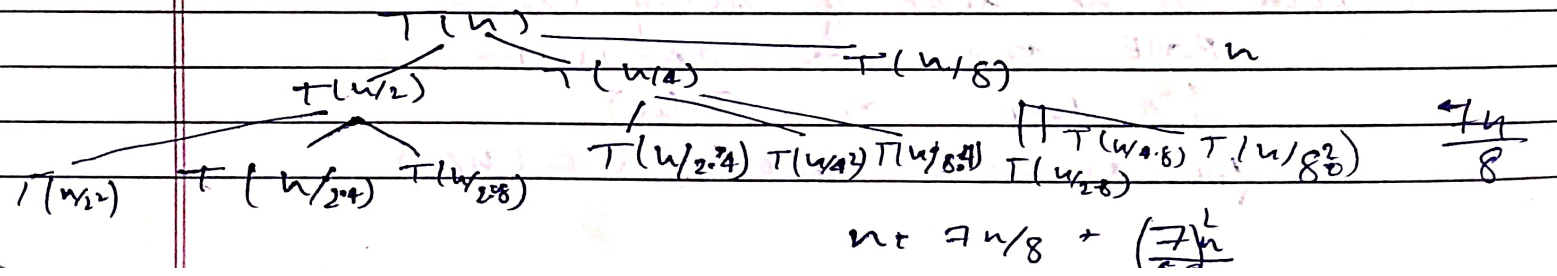
$$T(n) = n^2 + \frac{5n^2}{16} + \frac{5^2}{16^2} n^2 \cdots \text{---} R$$

$$T(n) = n^2 \left(\frac{1 - (5/16)^{\log_2 n}}{1 - 5/16} \right)$$

$$\downarrow$$

$$T(n) = O(n^2)$$

(iv) $T(n) = T(n/2) + T(n/4) + T(n/8) + n$



$n + 7n/8 + \left(\frac{7}{8}\right)^2 n$

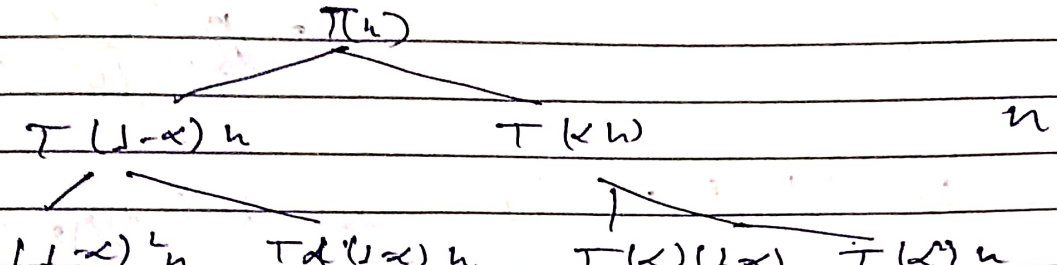
Depth $\Rightarrow \left(\frac{1}{2}\right)^R n = 1 \Rightarrow R = \log_2 n$

Total cost $= n + \frac{7}{8}n + \left(\frac{7}{8}\right)^2 n + \cdots R$

$$T(n) = n\left(\frac{1-(7/8)^{\log_2 n}}{1-7/8}\right)$$

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

(N) $T(n) = T(1-\alpha)n + T(\alpha n) + n$ where $\alpha > 0$ & $\alpha < 1$



Depth $\Rightarrow \alpha^R n = 1$

$\Rightarrow R = \log_{\alpha} n$

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

Q8

(a) A recursive call on the left half or right half both to be size of array $n/2$

Base case $\Rightarrow n = 1$

for loop $\rightarrow O(n)$

for recursive call $= 2T(n/2)$

putting together $T(n) = 2T(n/2) + O(n)$

$a=2, b=2, k=1, p=0$

case 2 :- Hence $T(n) = O(n \log_2^2 \cdot \log^{0+1} n)$

$= O(n \log n)$

(b) → 3 recursive calls

Base case when $n \leq 1$

$T(n) = T(n/2) + T(n/3) + T(n/4) + O(n)$

cost is n

$\downarrow \quad \frac{n}{2}, n/3, n/4 \Rightarrow n/2 + n/3 + n/4 = \frac{13}{12} n$

$\frac{13}{12} > 1$ increases at each level

n is being dominant term, then complexity is $T(n) = O(n)$

Q9

(a) Declare 3 variable to hold play (current) & 3 variable for to hold their index

[top1, top2, top3] & [Topin1, topin2, topin3]

iterate through the array. For each song in play count, compare to top2 if $A[i] > top1$,

top2 = top1

top3 = top2

top1 = A[i]

if A[i] > top1] but A[i] > top2
top3 = top2
top2 = A[i]
if [A[i] > top1] & ![A[i] > top2] but A[i] > top3

(b) Normal time complexity = O(n)
here, n=50 ⇒ T = O[1]
Space complexity = same no. of variable through out, so O(1)

Q10

Sorted array of size $N = 2^{20}$

(a) Max. no. of comparison required in Binary Search
Binary search requires at most $[\log_2 N] + 1$ comparision :-
$\log_2 20 = 20$
worst case comparison ⇒ 20 + 1 = 21

(b) Time if each comparison = 1 us
Best case :- 1 comparison → 1 us
worst case :- 21 comparison → 21 us
Avg case → 20 us.

(c) Taking worst case for both
linear search = 2^{20} us
binary search = 21 us
speed up factor = $\frac{2^{20}}{21}$ = 50080

Q11 A 2D array B[100][150] stored in row major order. Base address 5000 each element occupy 6 bytes. Lower bound for rows & cols $l_1 = 5$ & $l_2 = 10$

i) $B[i][j] = BA + [(i - lb) \times N_1 + (j - lb_2)] w$

$= 5000 + [(i - 5) \times 150 + j - 10] 6$

$B[55, 120] = 5000 + [(155 - 5 \times 5] \times 150 + 120 - 10] \times 6$

$= 50,660$

$B[55][100] = 5000 + [(120 - 10) 100 + 55 \cdot 5] \times 6$

$= 71300$