NAME ⇒ ASHUTOSH FARSWAN ROLL NO ⇒ 20

SECTION ⇒ DS

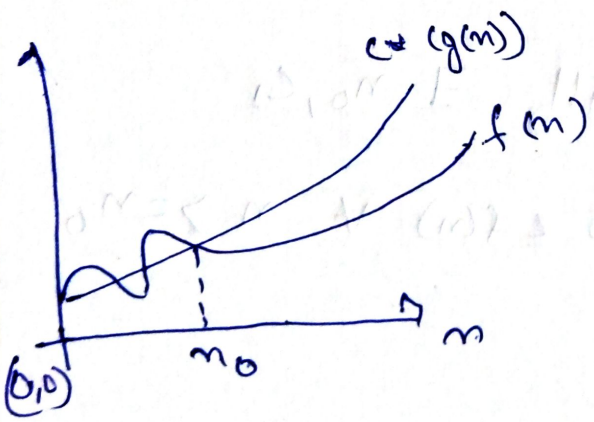
Q.1 ⇒ Asymptotic Notation: There are language to express the required time & space by an algorithm to solve a given problem.

(1) Big-O Notation: It is notation for the worst case analysis of an algorithm. (Upper bound)

According to it for a two func $f(n)$ & $g(n)$

$f(n) = O(g(n))$, if and only if there exist n_0 & c such that.

$0 \leq f(n) \leq c * g(n)$ for all $n \geq n_0$



Ans ⇒ $n+n^2 = O(n^2)$

here $f(n) = n+n^2$, $g(n) = n^2$.

$n+n^2 \leq n^2+n^2$ ($\because n < n^2$, $n^2 = n^2$)

$n+n^2 \leq 2n^2$ (here $c=2$) for $n_0=1$

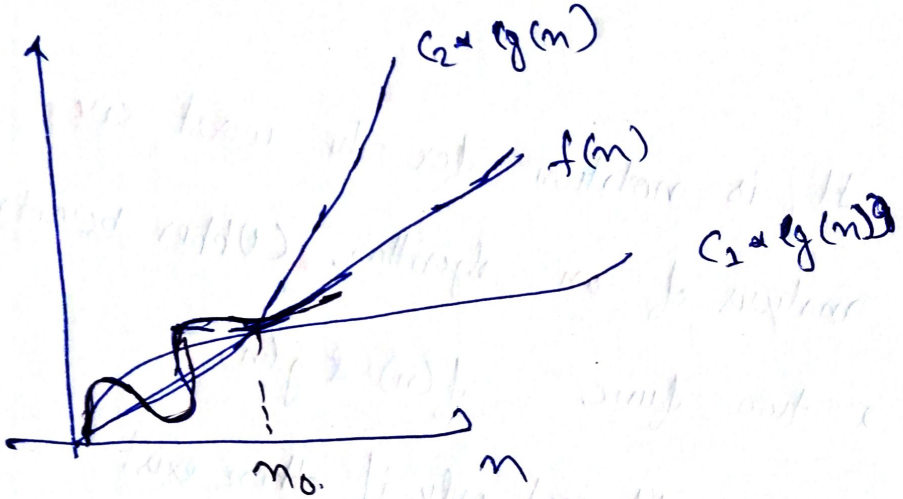
so $f(n) = O(g(n))$

or $n+n^2 = O(n^2)$

⊙ Big theta (Θ): For avg case time complexity (tightly bound)

for any two function $f(n)$ & $g(n)$

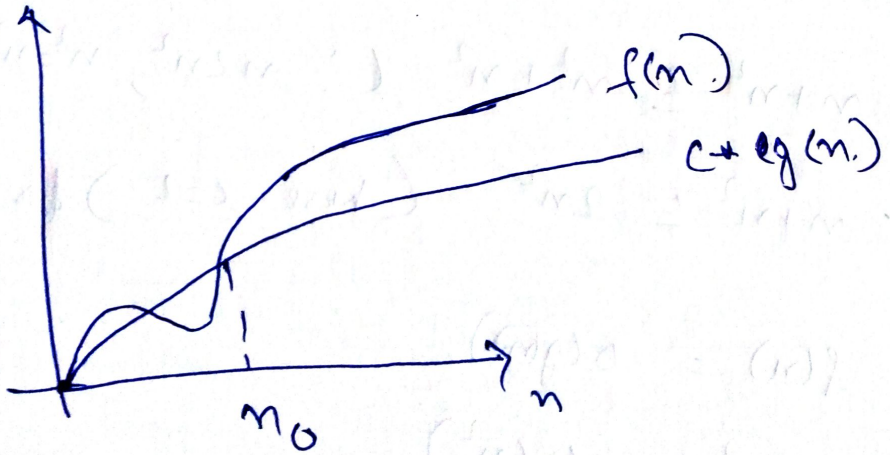
$f(n) = \Theta(g(n))$ if and only if there exists n_0, c_1, c_2 such that $0 \leq c_1 * g(n) \leq f(n) \leq c_2 * g(n)$ [for $n \geq n_0$]



⊙ Big Omega (Ω): for best case complexity (lower bound)

$f(n) = \Omega(g(n))$ iff $\exists\ n_0, c_1$

$\exists\ 0 \leq c_1 * g(n) \leq f(n)$ $\forall\ n \geq n_0$



Q.2 T.C. of for (i=1 to n) { i=i*2 }

Series → 1, 2, 4, 8, 16 ---- n (G.P.)

$a = 1$, $r = 2$

$t_k = ar^{k-1} \Rightarrow n = a*2^{k-1}$

$\Rightarrow n = 2^{k-1}$

$\Rightarrow 2^k = 2n$

$\Rightarrow k = 2\log_2 n$

so T.C. → $O(\log_2 n)$

Q.3 → $T(n) = \{ 3T(n-1) \}$ if $n>0$, otherwise 1 }

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

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

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

or $T(n) = 3^3 T(n-3)$

or $T(n) = 3^n T(n-n)$

$T(n) = 3^n T(0) = 3^n$

so T.C. → $O(3^n)$

Q.4) $T(n) = \begin{cases} 2T(n-1) - 1 & \text{if } n > 0, \\ 1 & \text{otherwise} \end{cases}$

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

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

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

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

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

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

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

or

$T(n) = 2^k T(n-k) - 2^{k-1} - 2^{k-2} - \cdots - 2^1 - 2^0$

$T(0) = 1$, let $n-k = 0$ so $k = n$

$T(n) = 2^n T(n-n) - 2^{n-1} - 2^{n-2} - \cdots - 2^1 - 2^0$

$= 2^n - 2^{n-1} - 2^{n-2} - \cdots - 2^1 - 2^0$

$= 2^n - (2^{n-1} + 2^{n-2} + \cdots + 2^1 + 2^0)$ } G.P

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

so T.C. ⇒ $O(1)$

Q.5 ⇒

```
int i=1, s=1;
while (s<=n){
        i++; s= s+i;
        printf("#");
}
```

Series ⇒ 1, 3, 6, 10, 15, 21, 28 ----- n

1st iteration ⇒ s = s+1

2nd iteration ⇒ s = s+1+2

till ⇒ $1+2+3+\dots+k <= n$

$$\frac{k*(k+1)}{2} <= n$$

or $O(k^2) <= n$

or $k = O(\sqrt{n})$

so T.C. = $O(\sqrt{n})$

Q6 ⇒

```
for (i=1 ; i*i <= n ; i++)
    count ++
```

let loop run till k i=k

$k^2 <= n$

$k <= \sqrt{n}$

no T.C ⇒ $O(\sqrt{n})$

Q7 ⇒

```
for (i=n/2 ; i<=n ; i++)                    O(n)
    for (j=1 ; j<=n ; j=j*2)                O(log n)
        for (k=1 ; k<=n ; k=k*2)            O(log n)
```

no T.C ⇒ $O(n \log^2 n)$

Q8 ⇒

```
function (int n) {
    if (n==1) return;
    for (i=1 to n) {
        for (j=1 to n) {
            print (n+n);
        }
    }
    function (n-3);
}
```

Recurrence Relation ⇒ $T(n) = T(n-3) + n^2$

or $T(n) = T(n-6) + 2*n^2$

$T(n) = T(n-9) + 3n^2$

or $T(n) = T(n-3k) + kn^2$

$T(1) = 0$, $n-3k = 1$ $\Rightarrow k = \frac{n-1}{3}$

so $T(n) = T(1) + \frac{(n-1)}{3} n^2$

so T.C. ⇒ $O(n^3)$

Q.9 ⇒

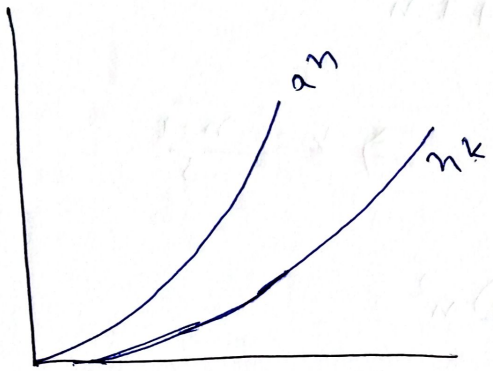
```
for (i=1 to n){
    for (j=1 ; j<=n ; j=j+i)
        printf ("*");
}
```

i	j	times
1	1→n	n
2	1→n	n/2
3	1→n	n/3
⋮	⋮	⋮
n	1→n	n/n

T.C. = ~~$O(n^2)$~~ $O(n \log n)$

Q.10⇒ Find asymptotic relation btw n^k & a^n, $k \geq 1$ & $a > 1$ are constants. Find c & n_0 for which relations holds.

Sol⇒



$n^k = O(a^n)$

$n^k \leq a^n$, c ∀ $c > 0$ & $n \geq n_0$

let $n = n_0$

$n_0^k \leq c \cdot a^{n_0}$

[so let $k = a = 3$]

[$n_0^3 \leq c \cdot 3^{n_0}$ so $c \geq 1$ & $n_0 \geq 1$]