Ans.1 Asymptotic Notation: Asymptotic Notation are used to represent the complexities of algorithms for asymptotic analysis.

There are 3 types of notations.

1. O big-o upper bound.
2. Ω big-omega lower baund
3. θ theta Average bound.

Big O: The fuction $f(n) = O(g(n))$ iff for all +ve constants c and no.

Such that $f(n) \leq c * g(n) \; \forall \; n \geq n_0$

eg. $f(n) = 2n+3$

$$\underbrace{2n+3}_{f(n)} \leq \underbrace{10}_{c}\,\underbrace{n}_{g(n)} \qquad n \geq 1$$

$\therefore F(n) = O(n)$

Omega: The fuction $f(n) = \Omega(g(n))$ iff for all +ve constant c and no.

such that $f(n) \geq c * g(n) \; \forall \; n \geq n_0$

E.g $F(n) = 2n+3$

$$\underbrace{2n+3}_{f(n)} \geq \underbrace{1}_{c} \times \underbrace{n}_{g(n)} \qquad \forall\; n \geq 1$$

$F(n) = \Omega(n)$

or $$\underbrace{2n+3}_{f(n)} \geq \underbrace{1}_{c} \times \underbrace{\log(n)}_{g(n)}$$

$f(n) = \Omega(\log(n))$

Theta Notation: The function $F(n) = \theta(g(n))$ iff for all +ve constant c_1, c_2 and n_0.

Such that $c_1 * g(n) \leq F(n) \leq c_2 * g(n)$

e.g $f(n) = 2n+3$

$$\underbrace{1}_{c_1} \times \underbrace{n}_{g(n)} \leq 2n+3 \leq \underbrace{5}_{c_2} \times \underbrace{n}_{g(n)}$$

$F(n) = \theta(n)$

Que. 2

```
for (i=1 to n)
  { i = i*2;
  }
```

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

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

Sol: $T(n) = 3T(n-1) \quad -(1)$

Put $n = n-1$ in eq 1

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

put $T(n-1)$ from 2 to 1

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

$T(n) = 9T(n-2) \quad -(3)$

put $n = n-2$ in eq 1

$T(n-2) = 3T(n-3) \quad -(4)$

Put $T(n-2)$ from (4) to (3)

$T(n) = 27\,T(n-3)$

for k^{th} term

$T(n) = 3^k\,T(n-k) \quad -(5)$

Scanned with CamScanner

$n-k=1$

$k=(n-1)$

Put value of k in (5)

$T(n)=3^{n-1}\cdot T(n-(n-1))$

$T(n)=3^{n-1}\cdot 1$

$T(n)=\frac{3^n}{3}$

$T(n)=O(3^n)$.

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

~~Then~~ put $n=n-1$ in eq 1

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

Now put $T(n-1)$ from 2 to 1

$T(n)=4T(n-2)-3$ - (3)

put $n=n-2$ in eq 1

$T(n-2)=2T(n-3)-1$ - (4)

Put $T(n-2)$ from 4 to 3

$T(n)=8T(n-3)-7$

similarly for ~~n~~ k^{th} term.

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

Since value of $T(n-1)$ is reducing one by one then lets assume

$n-k=1 \iff n=k$,

$= O(2^n * c - 2^n+1)$

$O(2^n(c-1)+1)$

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

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

5.
```
int i=1, s=1;
 while (s<=n) {
        i++; s=s+i;
        printf("#");
   }.
```
Ans: T.C = $O(\sqrt{n})$.

6.
```
void function(int n){
    int i, j, k, count=0;
       for (i=n/2; i<=n; i++){
            for (j=1; j<=n; j=j*2)
                 for (k=1; k<=n; k=k*2)
                   count++;
       }    }
```
T.C = $O(\sqrt{n})$.

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

8. T.C = $O(n^3)$.

9. T.C = $O(n \log n)$