Assignment

Name- Abhishek Singh

Semester- IV

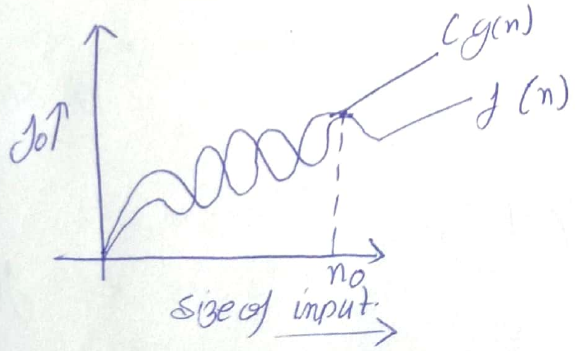
Section- CSTSPL-02

Class Roll no.- 09

Ans. 1) Asymptotic Notation →

They help us to find the complexity of an algorithm when input is very large.

Ex:-> 1) Big O(o)



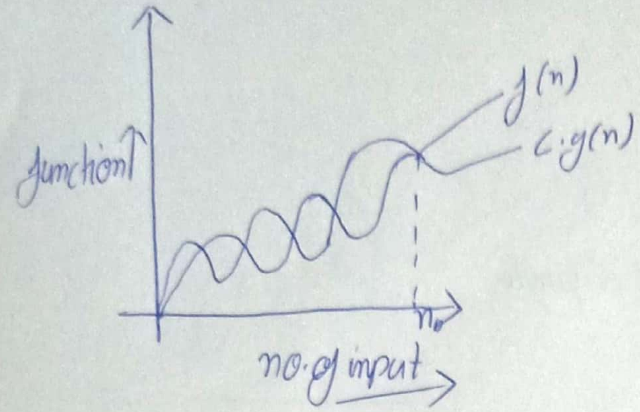
$f(n) = O(g(n))$

iff $f(n) \leq c \cdot g(n) \quad \forall \; n \geq n_0$

for some constant $c > 0$

=> $g(n)$ is tight upper bound of $f(n)$

II) Big Omega (Ω)



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

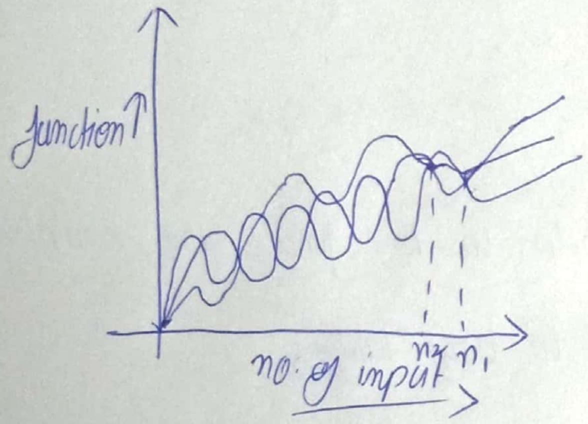
$g(n)$ is tight lower bound of $f(n)$

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

iff $f(n) \geq c \cdot g(n)$

$\forall\ n \geq n_0$ for some constant $c > 0$

III) Theta (Θ)



$f(n) = \Theta(g(n))$

$g(n)$ is both tight uper and lower bound of function $f(n)$

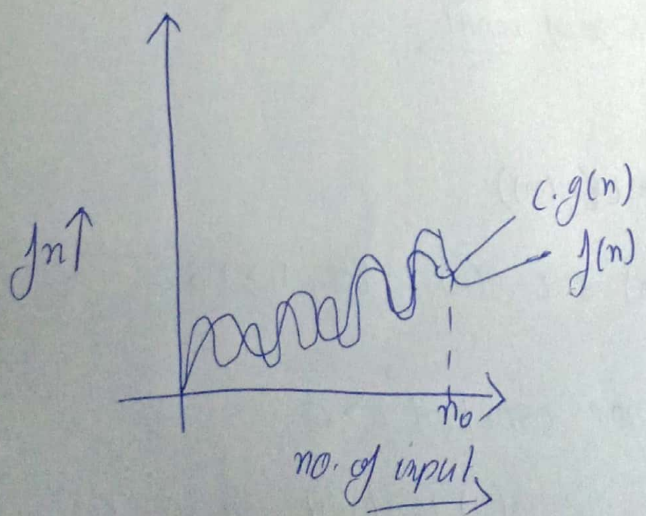
$f(n) = \Theta(g(n))$

iff $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$

$\forall\ n \geq c\ max(n_1, n_2)$

for some constant $c_1 > 0$ & $c_2 > 0$

IV) Small o (o)



$f(n) = o(g(n))$

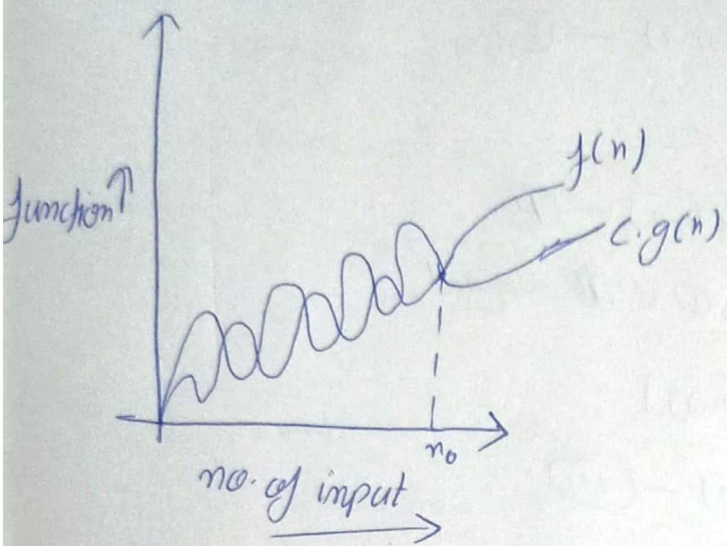
$g(n)$ is upper bound of function $f(n)$

$f(n) = o(g(n))$

when $f(n) < c \cdot g(n)$ $\forall\ n > n_0$

& $\forall\ c > 0$

v) Small omega (ω)



$f(n) = \omega(g(n))$

$g(n)$ is lower bound of function $f(n)$

$f(n) = \omega(g(n))$

when $f(n) > c \cdot g(n)$ $\forall$ $n > n_0$

& $\forall$ $c > 0$

Ans 2> for ($i=1$ to n) { $i = i*2$; }

for ($i=1$ to n) // $i = 1, 2, 4, 8 \ldots n$

{ $i = i*2$; } // $O(1)$

$$\Rightarrow \sum_{i=1}^{n} 1 + 2 + 4 + 8 + \ldots n$$

GP k^{th} value $\Rightarrow T_k = ar^{k-1}$

$= 1 \times 2^{k-1}$

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

$\Rightarrow 2n = 2^k$

$\Rightarrow \log 2n = k \log 2$

$\Rightarrow \log 2 + \log n = k \log 2$

$\Rightarrow \log(n+1) = k$

$O(k) = O(1 + \log n)$

$= \underline{O(\log n)}$

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

$T(n) = 3T(n-1)$ — I

Put $n = n-1$

$T(n-1) = 3T(n-2)$ — II

from equ^n I & II we get

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

$\Rightarrow T(n) = 9T(n-2)$ — III

Put, $n = n-2$ in equ^n I we get

$T(n) = 3(T(n-3))$ — IV

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

$\Rightarrow T(n) = 3^k T(n-k)$

Put, $(n-k) = 0$, we get

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

$\Rightarrow T(n) = 3^n \times 1$

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

Ans. 4> $T(n) = \{2T(n-1) - 1$ if $n>0$, other wise $1\}$

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

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

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

$\vdots$

$T(1) = 2T(0)$

$T(0) = 1$

Put ~~the~~ the value of ~~T(n)~~ T(n-1) in T(n) we get

$T(n) = 2^n \times T(0)$

$\Rightarrow T(n) = 2^n \times 1$

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

Ans. 5>

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

$i = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \ldots$

$s = 1 + 3 + 6 + 10 + 15 \ldots$

sum of $s = 1+3+6+10+\ldots+n$ — (I)

also $s = 1+3+6+10+\ldots T_n + \ldots + n$ — (II)

(II) − (I), we get

$0 = 1+2+3+\ldots+n - T_n$

$\Rightarrow T_k = 1+2+3+4+\ldots k$

$T_k = \frac{1}{2} k(k+1)$

for k iteration

$1+2+3+\ldots+k <= n$

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

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

$$\Rightarrow O(k^2) <= n$$

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

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

Ans. 6)

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

as $i^2 <= n$

$$\Rightarrow i <= \sqrt{n}$$

$$i = 1, 2, 3, 4 \dots, \sqrt{n}$$

$$\sum_{i=1}^{n} 1+2+3+4+ \dots + \sqrt{n}$$

$$\Rightarrow T_n = \frac{\sqrt{n}(\sqrt{n}+1)}{2}$$

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

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

Ans. 7 →

```
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++;
            }
        }
    }
}
```

for $k = k*2$

$k = 1, 2, 4, 8, \ldots n$

$\Rightarrow$ G → $a = 1, r = 2$

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

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

$\Rightarrow n = 2^k$

$\Rightarrow \log n = k$

i	j	k
1	log n	log n * log n
2	log n	log n * log n
⋮	⋮	⋮
n	log n	log n * log n

$\Rightarrow O(n * \log n * \log n)$

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

Ans 2> function(int n)

```
{
    if (n==1) return;    // O(1)
    for (i=1 to n) {     // i=1,2,3 ... n => O(n)
        for (j=1 to n) {    // O(n^2)
            ~~for~~ printf("*");
        }
    }
    function(n-3);    T(n/3)
}
```

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

$\Rightarrow a=1, \quad b=3, \quad f(n) = n^2$

$c = \log_3 1 = 0$

$\Rightarrow n^0 = 1 > (f(n) = n^2)$

$\Rightarrow \underline{T(n) = \theta(n^2)}$

Ans. 9) void function (int n){

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

for $i=1 \Rightarrow j = 1, 2, 3, \ldots n = n$

for $i=2 \Rightarrow j = 1, 3, 5, \ldots n = n/2$

for $i=3 \Rightarrow j = 1, 4, 7 \ldots n = n/3$

⋮

for $i=n \Rightarrow j = 1 \ldots n \Rightarrow 1$

$$\Rightarrow \sum_{j=n}^{1} n + \frac{n}{2} + \frac{n}{3} + \ldots + 1$$

$$\Rightarrow \sum_{j=n}^{1} n \left(1 + \frac{1}{2} + \frac{1}{3} + \ldots \frac{1}{n}\right)$$

$$\Rightarrow \sum_{j=n}^{1} n (\log n)$$

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

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

Ans. 10)

Relation between n^k and c^n is

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

as $n^k \leq ac^n \;\forall\; n \geq n_0$ and some

constant $a > 0$

for $n_0 > 1$

$c = 2$

$\Rightarrow 1^k \leq a2^1$

$n_0 = 1$ and $c = 2$