Design and Analysis of Algorithm.

Tutorial - 1

Ques:- 1. What do you understand by Asymptotic notations. Define different Asymptotic notation with examples.

Ans:- 1. Asymptotic notations are those notations that describing the limiting behaviour of a function There are three different types of notations :-

→ Big Oh (O).

→ Big (Ω).

→ Big (Θ).

→ Big Oh (O).

*Big-Oh (O) notation gives an upper bond for a function $f(n)$ to within a constant factor.

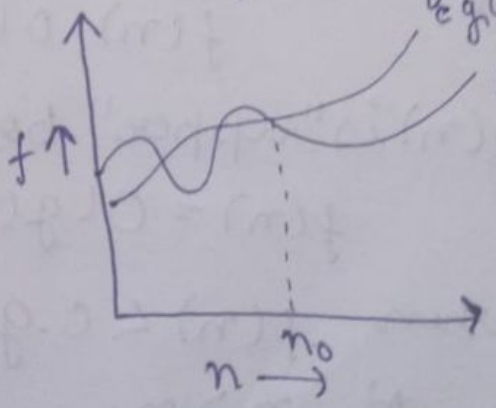
$$f(n) = O(g(n)).$$

$g(n)$ is "tight" upper bound

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

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

$\forall\ n \geq n_0$.



ex:-
```
for (i=1; i≤n; i++).
{
    sum += i
};
```

$\Rightarrow O(1+n+n+n)$

$= O(n)$.

* Big Omega Notation

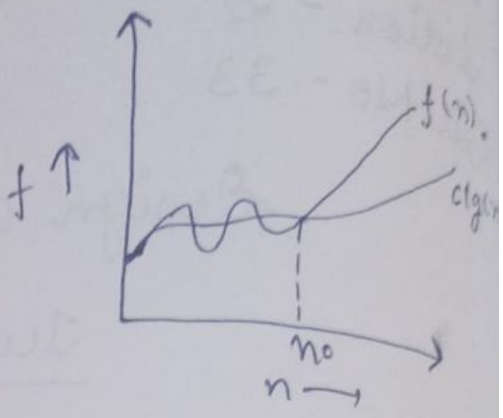
$f(n) = \Omega(g(n))$.

$g(n)$ is "tight" lower bound.

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

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

$\forall \; n \geq n_0$.



* Θ (Theta

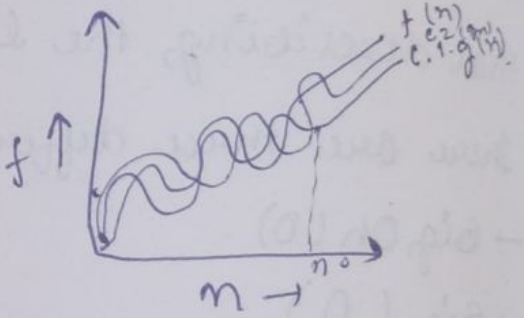
$f(n) = \Theta(g(n))$.

$g(n)$ is both "tight" upper and lower bond of fn.

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

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

$\forall \; n \geq \max(n_1, n_2)$.



* Small Oh (o).

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

$g(n)$ is upper bond of the function $f(n)$.

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

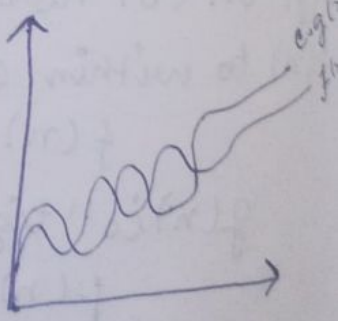
$g(n)$ is "upper" bound of fn

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

when $f(n) < c \cdot g(n)$

$\forall \; n \geq n_0$

$\forall \; c > 0$.

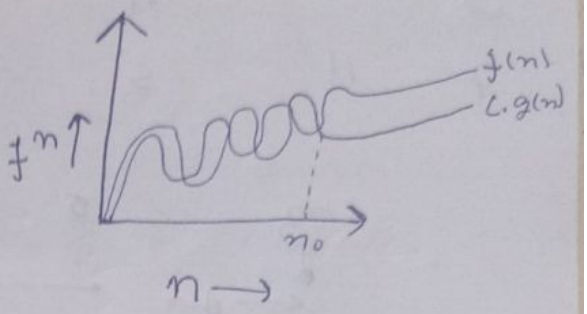


* Small Omega (ω)

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

$g(n)$ is "lower" bound of fn

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



when $f(n) > C \cdot g(n)$.

$\forall\ n > n_0$

$\forall\ C > 0$

Ques:-2 What should be time complexity

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

```
for (i=1 to n)      // i = 1, 2, 4, 8 --- n.
{i = i*2}           // O(1).
```

$$\Rightarrow \sum_{i=1} 1 + 2 + 4 + 8 + \cdots + n.$$

K^{th} term of GP $\Rightarrow$ $T_K = a\, r^{K-1}$.

$n = 1 * 2^{K-1}$.

$n = 2^{K-1}$

$n = \frac{2K}{2}$

$2^n = 2K$

$\log_2(2n) = K(\log_2 2)$

$K = \log_2(2n)$

$K = \log_2 2 + \log_2 n$

$K = 1 + \log_2 n$

$K \neq 1 + \log_2 n$

$O(\log_2 n)$.

$\Rightarrow O(n)$.

Ques:-3

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

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

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

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

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

$\vdots$

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

$= 3^n T(0)$.

$= 3^n$.

$\Rightarrow O(3^n)$.

Ques:-4

$T(n) = \{2T(n-1) - 1$ if $n>0$, otherwise $1\}$.

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

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

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

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

$= 2^3 T(n-3) - 2^2 - 2^1 - 2^0$

$\vdots$

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

$---- 2^2 - 2^1 - 2^0$.

$= 2^n - 2^{n-1} - 2^{n-2} - 2^{n-3}$

$--- 2^2 - 2^1 - 2^0$

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

$T(n) = 1$.

$\Rightarrow O(1)$.

Ques:-5 what should be the time complexity

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

$i = 1, 2, 3, 4, 5, 6, \ldots K$

$s = 2 + 2 + 3 + \ldots K$.

when $s >= n$, then loop will stop at K^{th} iterations

$\Rightarrow s >= n$

$s = n$.

$\rightarrow 2 + 2 + 3 + 4 + \ldots + K = n$

$= 1 + (K * (K+1))/2 = n$

$= K^2 = n$

$K = \sqrt{n}$

$= O(\sqrt{n})$.

Ques:-6 Time complexity of

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

as $i^2 <= n$.

$i <= \sqrt{n}$

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

$\sum_{i=1}^{n} 1 + 2 + 3 + \ldots + \sqrt{n}$.

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

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

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

Ques:-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 $k = k*2$

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

G.P $\Rightarrow a = 1, \; r = 2$

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

$$\Rightarrow \frac{1(2^K-1)}{1}$$

$$n \Rightarrow 2^K$$

$$\log n = K.$$

i	j	K
1	$\log n$	$\log n * \log n$
2	$\log n$	$\log n * \log n$
⋮	⋮	⋮
n	$\log n$	$\log n * \log n$

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

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

Q:- 8.

```
funtion (int n)
{
   int (n==1)
    return ;
for (i=1 to n)
{
  for (j=1 to n).
}
   function (n-3);
}
```

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

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

$$c = \log_3 1 = 0$$

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

$$T(n) = \Theta(n^2).$$

Ques:-9

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

for $i=1 \Rightarrow j = 1,2,3,4,---n.$

for $i=2 \Rightarrow j = 1,3,5,7,---n$

for $i=3 \Rightarrow j = 1,4,7,----n.$

for $i=n \Rightarrow j=1 ---$!

$$\Rightarrow \sum_{i=n}^{1} n+1/2+n/3+n/4+---+1.$$

$$\Rightarrow \sum_{j=n}^{1} n[1+1/2+1/3+1/4+----1/n].$$

$$\Rightarrow \sum_{j=n} n[\log n].$$

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

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

Ques:-10

as given n^k & c^n

relation b/w n^k & c^n is

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

as $n^k \leq c^n$

$\forall\ n \geq n_0$ and some constant $a>0$

for $n_0 = 1$

$C = 2$

$\Rightarrow 1^k \leq a 2^1$

$\Rightarrow n_0 \leq 1$ & $C = 2$.