Tutorial - 1

Name :- Daksh Goel
Section :- F
Roll no. :- 51
Univ. Roll no. :- 2016713

1) what do you understand by Asymptatic notation, define different asymptatic notation with example.

(i) Big O(n)

$f(n) \Rightarrow O(g(n))$

if $f(n) \leq g(n) \times c \quad \forall \; n \geq n_0$

for same constant, $c > 0$

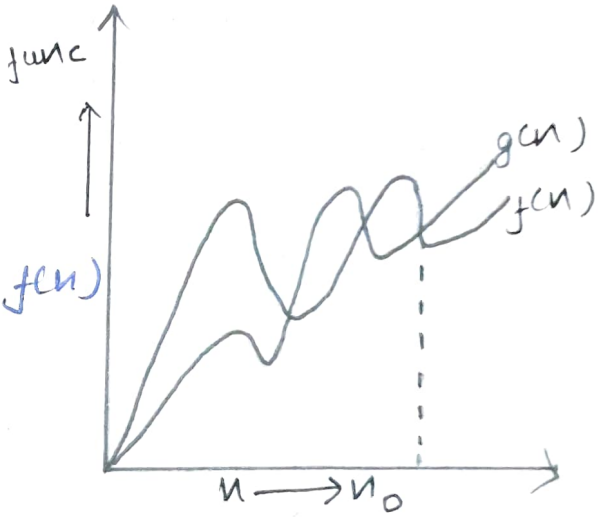
$g(n)$ is 'tight' upper bound of $f(n)$

eg:- $f(n) \Rightarrow n^2 + n$

$g(n) \Rightarrow n^3$

$n^2 + n \leq c * n^3$

$n^2 + n = O(n^3)$



(ii) Big omega (Ω)

when $f(n) = \Omega(g(n))$ means $g(n)$ is "tight" lower bound of $f(n)$ i.e. $f(n)$ can go beyond $g(n)$

i.e. $f(x) = \Omega g(n)$

if and only if

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

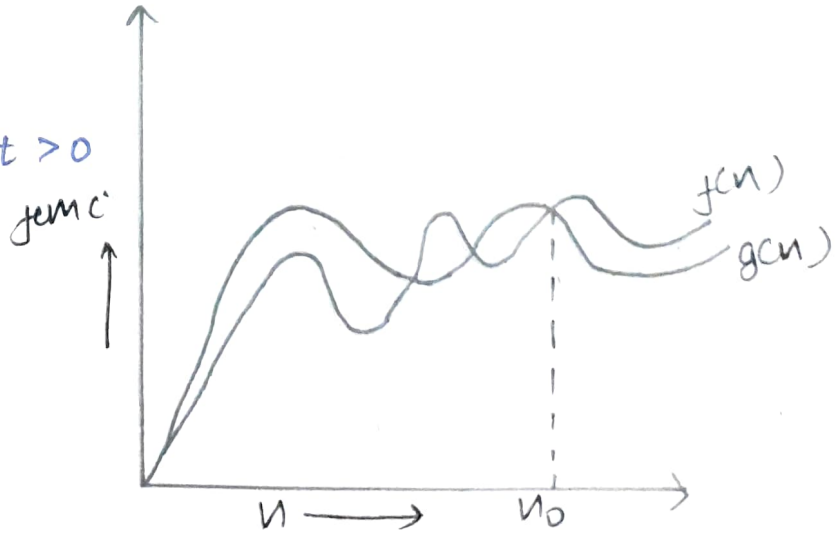
$\forall \; n_2 > n_0$ and c = constant > 0

eg:- $f(n) \Rightarrow n^3 + 4n^2$

$g(n) \Rightarrow n^2$

i.e $f(n) \geq c * g(n)$

$n^3 + 4n^2 = \Omega(n^2)$



Daksh.

(iii) Big Theta (Θ)

when $f(n) = \Theta(g(n))$ gives the tight upper bound and lower bound both

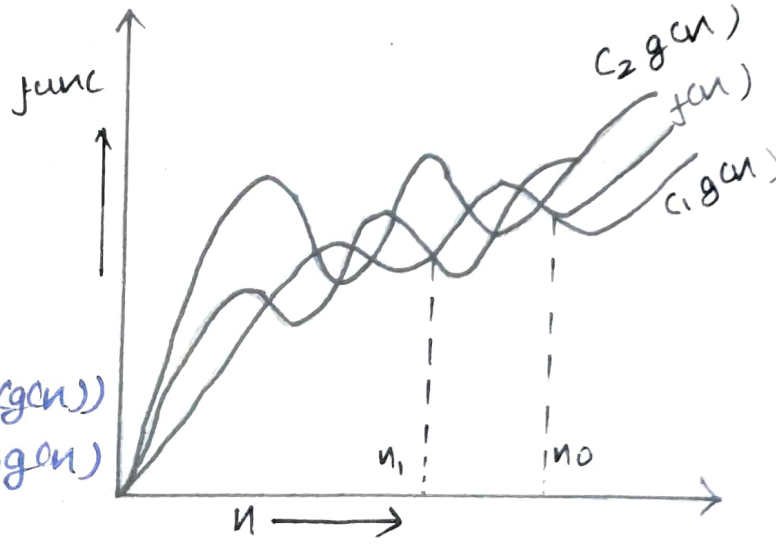
ie $f(n) = \Theta(g(n))$

if and only if

$c_1 * g(n_1) \leq f(n) \leq c_2 * g(n_2)$

for all $n \geq \max(n_1, n_2)$, some constant $c_1 > 0$ & $c_2 > 0$

ie $f(n)$ can never go beyond $c_2(g(n))$ and will never come down of $c_1 g(n)$



Eg:- $3n+2 = \Theta(n)$ as $3n+2 \geq 3n$

$3n+2 \leq 4n$ for n, $c_1 = 3$

$c_2 = 4$ and $n_0 = 2$

(iv) small o (Θ)

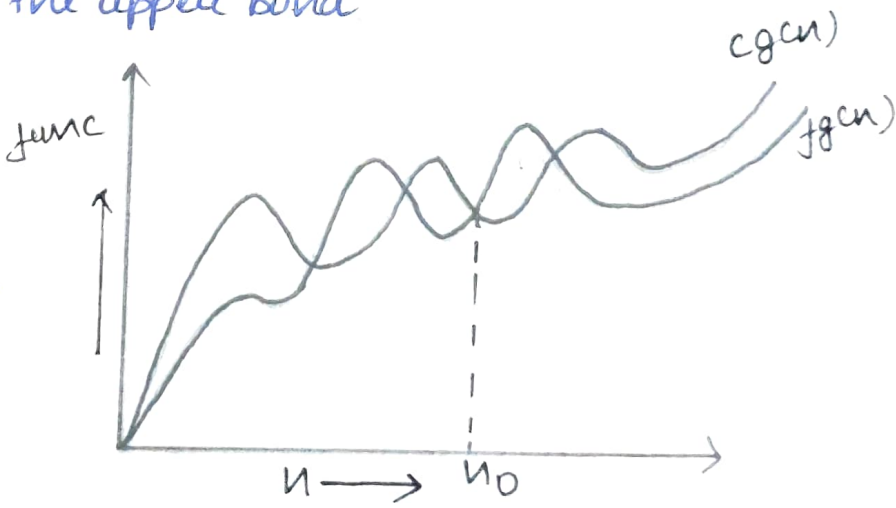
when $f(n) = o(g(n))$ gives the upper bond

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

if and only if

$f(n) < c * g(n)$

$\forall\ n > n_0$ & $n > 0$



Eg:- $f(n) = n^2$; $g(n) = n^3$

$f(n) < c * g(n)$

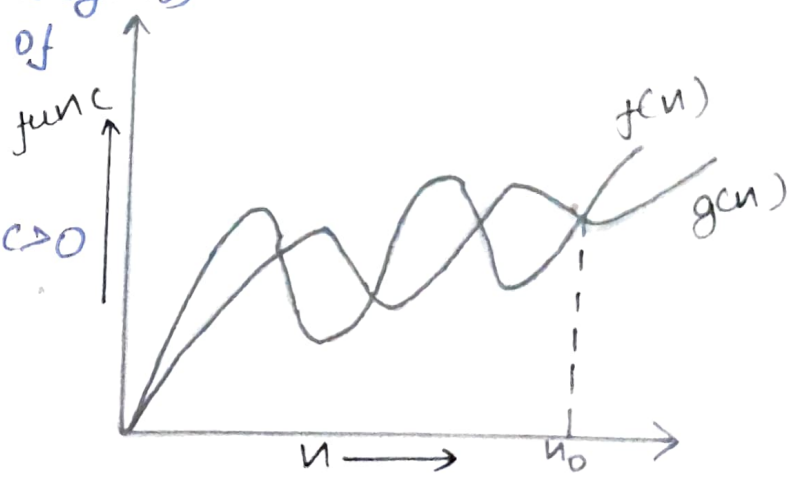
$n^2 = O(n^3)$

(v) small omega (ω)

It gives 'lower bound' ie $f(n) = \omega(g(n))$ where $g(n)$ is lower bound of $f(n)$ if and only if

$f(n) > c * g(n)$

$\forall\ n > n_0$ & some constant, $c > 0$



Daksh.

2) what should be time complexity of:

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

⇒ for i = 1, 2, 4, 6, 8 ... n times

ie series is a GP

so $a=1$, $r=2/1$

k^{th} value of GP:

$t_k = a r^{k-1}$

$t_k = 1(2)^{k-1}$

$2n = 2^k$

$\log_2(2n) = k \log 2$

$\log_2 2 + \log_2 n = k$

$\log_2 n + 1 = k$ (Neglecting '1')

So, Time complexity $T(n) \Rightarrow O(\log_2 n)$

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

⇒ ie $T(n) \Rightarrow 3T(n-1)$ —①

$T(n) = 1$

put $n \Rightarrow n-1$ in ①

$T(n-1) \Rightarrow 3T(n-2)$ —②

put ② in ①

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

$T(n) \Rightarrow 9T(n-2)$ —③

put $n \Rightarrow n-2$ in ①

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

put in ③

$T(n) = 27T(n-3)$ —④

Daksh.

Generalising series,

$T(K) = 3^K T(n-K)$ — (5)

for K^{th} terms, let $n-K=1$ (Base case)

$K = n-1$

put in (5)

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

$T(n) = 3^{n-1}$ [Neglecting 3']

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

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

$\Rightarrow T(n) = 2T(n-1) - 1$ — (1)

put $n = n-1$

$T(n-1) = 2T(n-2) - 1$ — (2)

put in (1)

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

$= 4T(n-2) - 2 - 1$ — (3)

put $n = n-2$ in (1)

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

Put in (1)

$T(n) = 8T(n-3) - 4 - 2 - 1$ — (4)

Generalising series,

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

K^{th} Term

Let $n-K=1$

$K = n-1$

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

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

in series in GP

$a = 1/2$, $r = 1/2$

Darsh.

So,

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

$$= 2^{n-1}(1 - 1 + (1/2)^{n-1})$$

$$= \frac{2^{n-1}}{2^{n-1}}$$

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

5) What should be time complexity of

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

⇒ i = 1, 2, 3, 4, 5, 6 ....

s = 1 + 3 + 6 + 10 + 15 + ...

Sum of s = 1 + 3 + 6 + 10 + ... + n — (1)

Also, s = 1 + 3 + 6 + 10 + ... $T(n-1) + T_n$ — (2)

$0 = 1 + 2 + 3 + 4 + \dots n - T_n$

$T_k = 1 + 2 + 3 + 4 + \dots + k$

$T_k = 1/2\, k(k+1)$

for k iterations

$1 + 2 + 3 + \dots k \le n$

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

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

$O(k^2) <= n$

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

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

Daksh.

6.) Time complexity of

```
void f(int n)
{
    int i, count = 0;
    for(i = 1; i * i <= n; ++i)
}
```

⇒ As $i^2 = n$

$i = \sqrt{n}$

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

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

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

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

$$\boxed{T(n) = O(n)}$$

7.) Time complexity of

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

⇒ Since, for $k = k^2$

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

∵ series is in GP

So, $a = 1, r = 2$

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

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

$n = 2^k - 1$

$n + 1 = 2^k$

$\log_2(n) = k$

Daksh.

i	j	K
1	log(n)	log(n) * log(n)
2	log(n)	log(n) * log(n)
⋮	⋮	⋮
n	log(n)	log(n) * log(n)

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

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

8.) Time complexity of

```
void function(int n)
{
    if(n==1) return;
    for(i=1 to n) {
    for(j=1 to n) {
        printf("*");
    } }
    function(n-3);
}
```

$\Rightarrow$ for(i=1 to n)

we get j=n times every turn

$\therefore i * j = n^2$

K^{th}, Now,

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

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

$T(n-6) = (n-6)^2 + T(n-9)$;

and $T(1) = 1$;

Now, substitute each value in T(n)

$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \cdots + 1$

Let

$K^n - 3K = 1$

$K = (n-1)/3$ total times $= K+1$

Thanks.

$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \cdots + 1$

$T(n) \simeq Kn^2$

$T(n) \simeq (K-1)/3 * n^2$

so, $\boxed{T(n) = O(n^3)}$

9.) Time complexity of :-

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

⇒ for $i=1$ $j = 1+2+\cdots (n \geq j+i)$

$i=2$ $j = 1+3+5 \cdots (n \geq j+i)$

$i=3$ $j = 1+4+7 \cdots (n \geq j+i)$

n^{th} term of AP is

$T(n) = a + d * m$

$T(m) = 1 + d * m$

$(n-1)/d = n$

for $i=1$ $(n-1)/1$ times

$i=2$ $(n-1)/2$ times

$i = n-1$

we get,

$T(n) = i_1 j_1 + i_2 j_2 + i_3 j_3 \cdots i_{n-1} j_{n-1}$

$= \frac{(n-1)}{2} + \frac{(n-2)}{2} + \frac{(n-3)}{3} + \cdots 1$

$= n + n/2 + n/3 + \cdots n/n-1 - n \times 1$

$= n[1 + 1/2 + 1/3 + \cdots 1/n-1] - n \times 1$

$= n \times \log n - n + 1$

since $\int 1/x = \log x$

$\boxed{T(n) = O(n(\log(n)))}$

Daksh.

10.) For Function n^k & c^n, what is asymptotic Relationship b/w these functions?

Assume that $k >= 1$ & $c > 1$ are constants. Find out value of c and no. of which relationship holds.

⇒ As given n^k and c^n

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

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

$$n^k \leq a(c^n)$$

$\forall\ n \geq n_0$ & constant, $a > 0$

for $n_0 = 1$; $c = 2$

$\Rightarrow 1^k < a^2$

$\Rightarrow n_0 = 1$ & $c = 2$

Daksh.