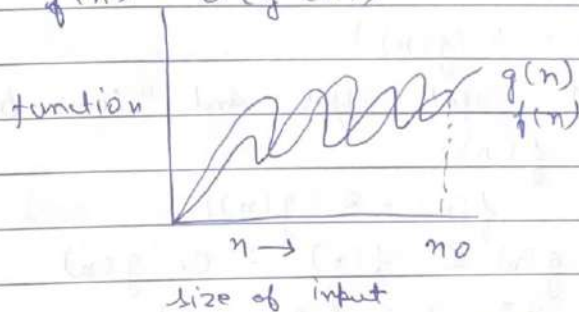Tutorial - 1

1. What do you understand by Asymptotic notations. Define different notation with examples?

Sol: Asymptotic Notations:- They are the mathematical notations used to describe the running time of an algorithm when the input ends towards a particular value or a limiting value.

Different asymptotic notations -

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

function

$n \rightarrow$ n_0
size of input

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

iff $\quad f(n) \leq c\,g(n)$

$\qquad \forall \quad n \geq n_0$

for some constant, $c > 0$

$g(n)$ is "tight" upper bound of $f(n)$.

ex. $f(n) = n^2 + n$

$\qquad g(n) = n^3$

$\qquad n^2 + n \leq c\,n^3$

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

(ii) Big Omega (Ω)

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

$\qquad g(n)$ is "tight" lower bound of function $f(n)$.

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

iff $\quad f(n) \geq c\,g(n)$

$\qquad \forall \quad n \geq n_0$

for some constant $C > 0$

$f(n)$

$g(n)$

↑ function

$n \rightarrow$ n_0
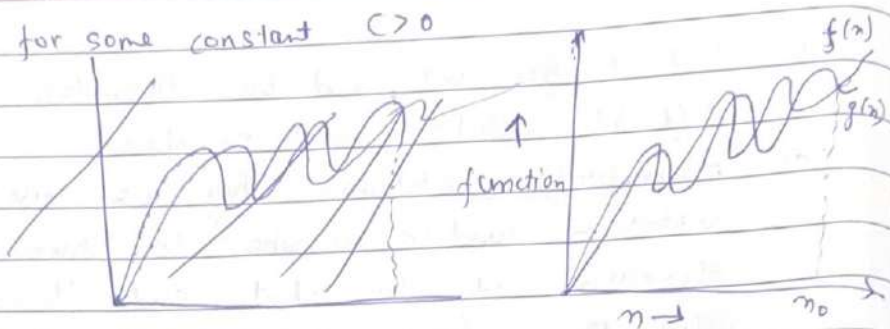
ex. $f(n) = n^3 + 4n^2$

$g(n) = n^2$
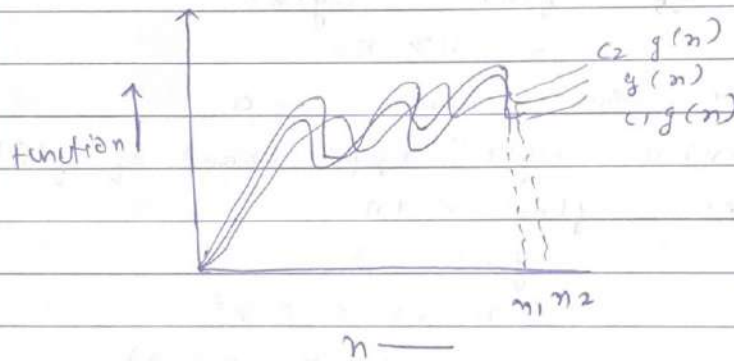
$= n^3 + n^2 - \Lambda(n^2)$

(iii) Big Theta (θ)

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

$g(n)$ is both "tight" upper and "lower bound" of function $f(n)$.

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

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

∀ $n \geq max(n_1, n_2)$

for some constant $c_1 > 0$ and $c_2 \; 0$

function ↑

$c_2 g(n)$
$g(n)$
$c_1 g(n)$

$n_1 n_2$

$n \longrightarrow$

Ex

$3n + 2 = 0(n)$ as $3n + 2 \geq 3n$ and

$3n + 2 \leq t(n)$ for $n, \; K_1 = 3, K_2 = 4$ & $n_0 = 2$

(iv) Small $\theta (\theta)$ -

$$f(n) = 0 (g(n))$$

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

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

when

$$f(n) < cg(n)$$

$$n > n_0$$

and ∀ constant , $c > 0$

function

$cg(n)$
$g(n)$

$n \longrightarrow$ n_0
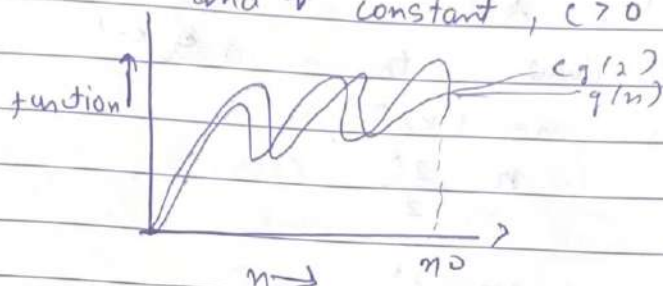
Ex — $f(n) = n^2$

$g(n) = n^3$

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

(v)   Small Omega $\omega(n)$

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

$g(n)$ is lower bound of $f(n)$
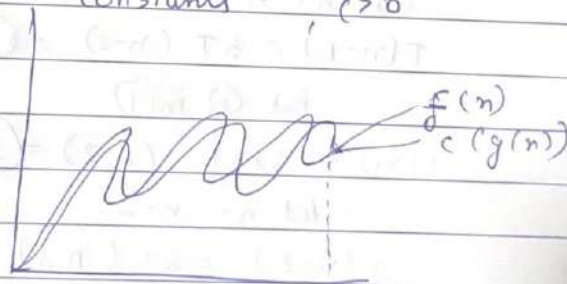
$$f(n) = \omega(g(n))$$

when $f(n) \geq cg(n)$

$n > n_0$

and ∀ constants $c > 0$

$\dfrac{f(n)}{c(g(n))}$

$f(n) = 4n + 6$ $g(n) = (1)$

2. What should be time complexity of
```
for ( i=1 -to n ) { i = i * 2; }
```

Sol: for (i=1   to n)
           ξ i = i * 2 3

i = 1 , 2, 4, 8, 16 , - - - - - n   y.p
$$\rightarrow O(K)$$

a = 1 , $r = 2$,

GP k^{th} value = t_k = $a \, r^{k-1}$

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

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

$$2n = 2^k$$

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

$$k = \log_2 2n$$

$$k = \log_2 2 + \log_2 n$$

$$k = 1 + \log_2 n$$

Time comp = $O(1 + \log_2 n)$

$$\rightarrow O(\log_2 n)$$

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

$$T(n) = 3T(n-1) \rightarrow \text{①}$$

let n = n-1

$$T(n-1) = 3T(n-2) \rightarrow \text{②}$$

Put ② in ①

$$T(n) = 3 \times 3T(n-2) \rightarrow \text{③}$$

let n = n-2

$$T(n-2) = 3T(n-3) \rightarrow \text{④}$$

Put ④ in ③

$$T(n) = 3 \times 3 \times 3 T(n-3) \rightarrow \text{⑤}$$

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

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

$$= 3^n$$

$$= O(3^n)$$

4.
$$T(n) = \{ 2T(n-1) - 1 \quad \text{if } n>0 \text{ , otherwise } 1]$$

$$T(n) = \begin{cases} 2T\left(\frac{n-1}{n-1}\right) - 1 & , \ n>0 \\ 1 & , \ \text{otherwise} \end{cases}$$

$$T(n) = 2T(n-1) - 1 \qquad --- \ ①$$

replacing n with $n-1$

$$T(n-1) = 2T(n-2) - 1 \qquad --- \ ⑪$$

replacing n with $n-2$

$$T(n-2) = 2T(n-3) - 1 \qquad --- \ ⑪⑪$$

from ①, ⑪, ⑪⑪ we get

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

$$T(n) = 2^2 T(n-2) - 2 - 1 \qquad \text{using } ⑪$$

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

$$T(n) = 2^3 T(n-3) - 2^2 - 2 - 1 \qquad \text{, using } ⑪⑪$$

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

we know that

$$T(0) = 1$$

$$n - k = 0$$

$$\boxed{k = n}$$

on putting $k = n$, we get

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

$$= \underbrace{2^n + 2^{n-1} + 2^{n-2} + \cdots + 2 + 1}_{G \cdot P}$$

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

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

$$\boxed{\text{Time complexity} = O(2^n)}$$

5. what should time complexity of

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

3

Sol.
s depends on i, so we make case

At i=1 | 2 | 3 | -4) ... — n
s=1 | 3 | 6 | 101 ... k

at i=n, s=k & k breaks the while cond.
so we can clearly see that s is just
of sum of n natural no's so

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

$$\frac{\boxed{k^2} + k}{2} > n \quad \rightarrow \text{dominating power}$$

$$\rightarrow k^2 \geq n$$

$$\boxed{k = \sqrt{n}}$$

$$\boxed{\text{time complexity} = O(\sqrt{n})}$$

6 - Time complexity of
```
void function (int n) {
    int i, count = 0;
    for (i=1; i * i <= n; i++)
        count ++
}
```

Sol.
$i = 1, 2, 3, \ldots n$

$i^2 = 1, 4, 9, \ldots n$

so $i^2 \leq n$ or $i \leq \sqrt{n}$

$$a_k = a + (k-1)d$$

$$a = 1, \quad d = 1$$

$$a_k \leq \sqrt{n}$$

$$\sqrt{n} = 1 + (k-1)1$$

$$\sqrt{n} = k$$

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

7. Time complexity of
```
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 ++ ; }
```

All   are independent loops,

so   time complexity $= \frac{n}{2}$ * $\log_2 n$ * $\log_2 n$

time complexity $= O(n \log^2 n)$

8. Time complexity of

```
function (int n) {
   if (n==1) return;
   for (i=1 to n) {
      for (j=1 to n) {
         print (" * ");
      }
   }

   function (n-3);
}
```

$T(n) = T(n-3) + n^2$ ——①

$T(1) = 1$ ——②

put $n = n-3$ in ①

$T(n-3) = T(n-6) + (n-3)^2$ ——③

Put ③ in ①

$T(n) = T(n-6) + (n-3)^2 + n^2$ ——④

put $n = n-6$ in ①

$T(n-6) = T(n-9) + (n-6)^2$ ——⑤

Put ⑤ in ④

$T(n) = T(n-9) + (n-6)^2 + (n-3)^2 + n$

Generalizing

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

Let $\quad n-3k=1$

$$\frac{n-1}{3}=k$$

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

$$+\left(n-3\left(n-\frac{1}{3}\right)\right)^2+ \text{---} n^2$$

$$T(n)=T(1)+[n-(n-1)-3]^2+[n-(n-(-6))]^2$$

$$+ (n-(n-1-9))^2 + \text{---} n^2$$

$$T(n)=1+(3+1)^2+(6+1)^2+ \text{---} n^2$$

$$T(n)=1^2+4^2+7^2+ \text{---} n^2$$

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

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

9. Time Complexity of -

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

for $i=1 \quad \longrightarrow \quad j=1$ to $n \quad \longrightarrow \quad n$ times

$i=2 \quad \longrightarrow \quad j=1$ to $n \quad \longrightarrow \quad n/2$ times

$i=3 \quad \longrightarrow \quad j=1$ to $n \quad \longrightarrow \quad n/3$ times

$i=n \quad \longrightarrow \quad j=1$ to $n \quad \longrightarrow \quad 1$ times

So total $= n + n/2 + n/3 + n/4 + \text{---} + 1$

$$= n \underbrace{\left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \text{---} \frac{1}{n}\right)}_{\log n}$$

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

$$\boxed{\text{Time complexity} = O(n \log n)}$$

10. for the function, n^k and c^n, what is the asymptotic relationship b/w these function?

Assume that $k \geq 1$ and $c > 1$ constants. Find out the value of C and $n0$ for which relations holds?

sol As given n^k and c^n

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

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

as $n^k \leq d\, c^n$

∀ $n \geq n_0$ & some constant $a > 0$

for $n_0 > 1$

$c > 2$

$1^k \leq d_2$

$n_0 = 1$ & $c = 2$