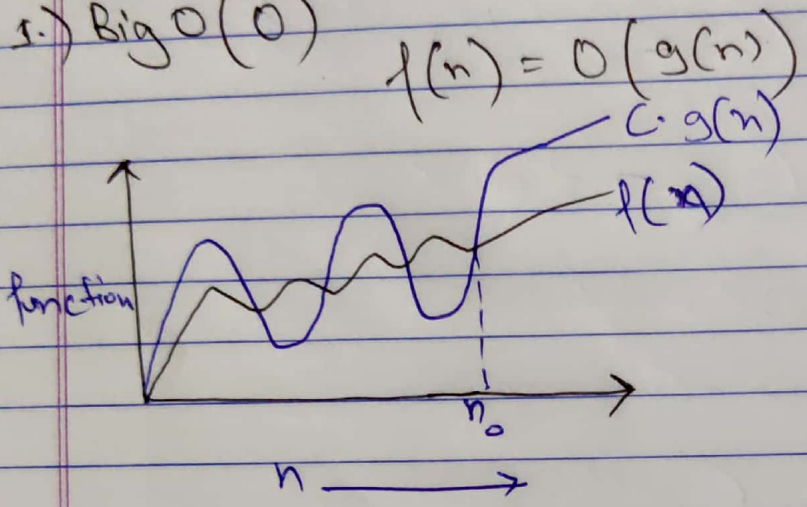
Date
Page No.

Asymptotic notations.

These notations are used to tell the complexity of an algorithm when I/P is very large.

1.) Big O(O)



g(n) is "tight" upper bound.

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

iff

$$f(n) \leq C \cdot g(n)$$

$\forall$ $n \geq n_0$ & some const, $c > 0$

2.) Big Omega Ω

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

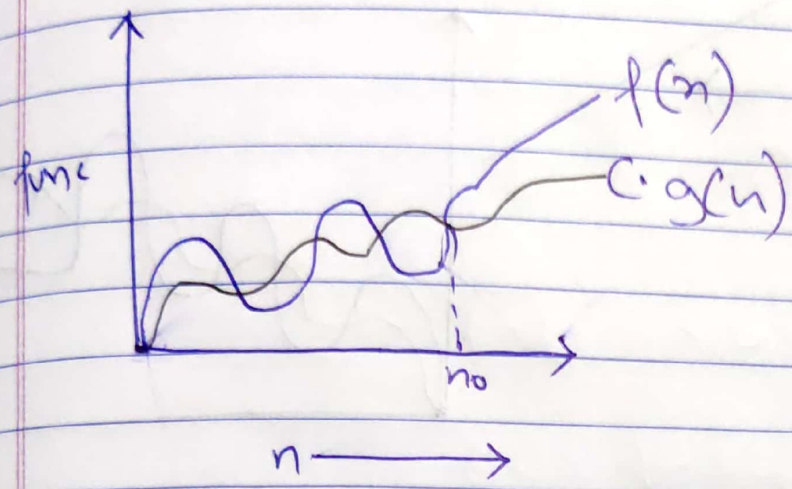
g(n) is "tight" lower bound.

Date.
Page No.

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

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

$\forall$ $n \geq n_0$ and const $c > 0$.



3) Theta (Θ)

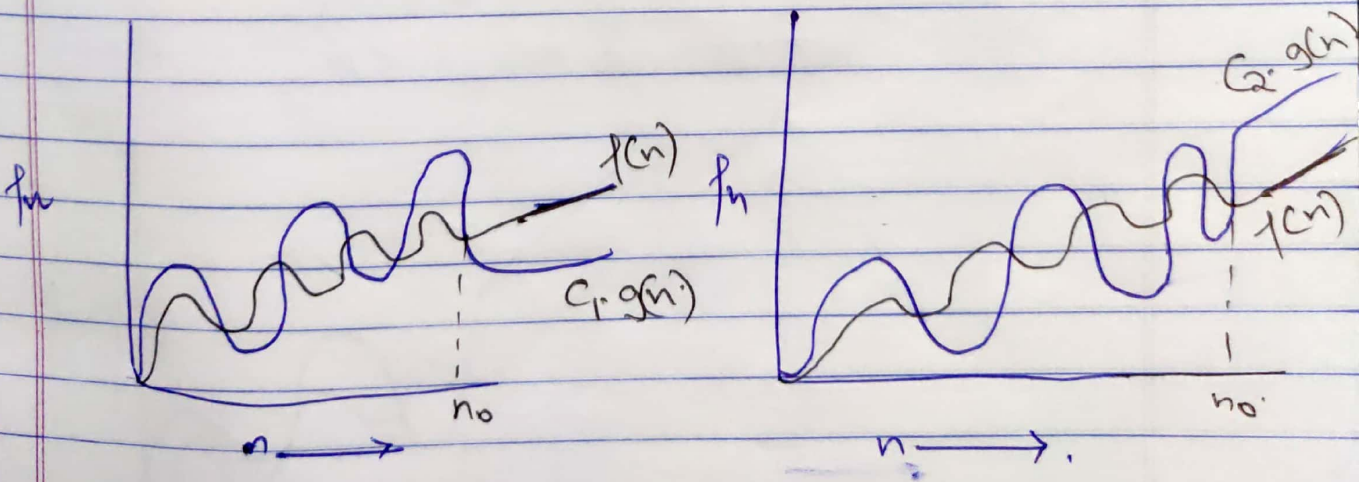
$f(n) = \Theta(g(n))$.

$g(n)$ is both "tight" UB & LB of $f(n)$.

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

iff

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



→ Small o (o)

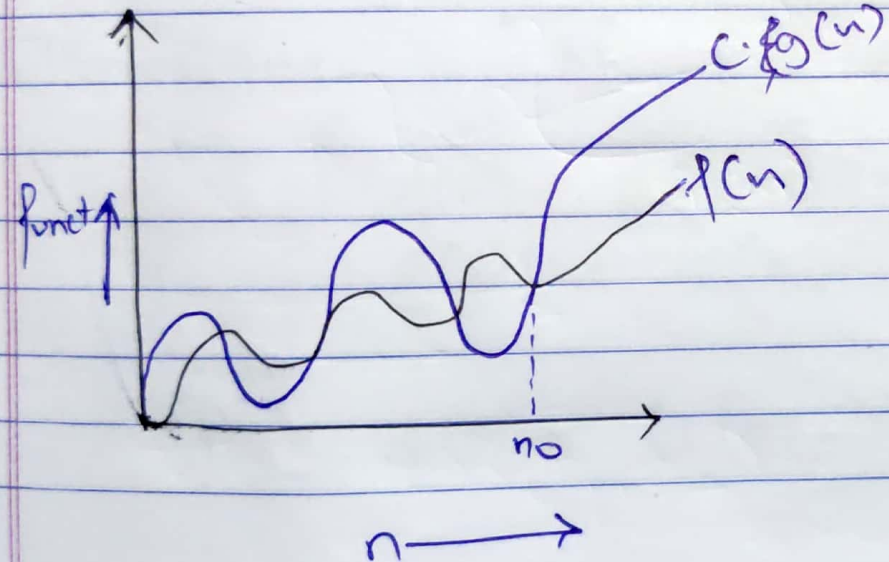
$f(n) = o(g(n))$

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

$f(n) = o(g(n))$. ∀ = forall, every.

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

∀ $n > n_0$ and ∀ every constant $c > 0$



Date
Page No.

5) Small omega (ω)

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

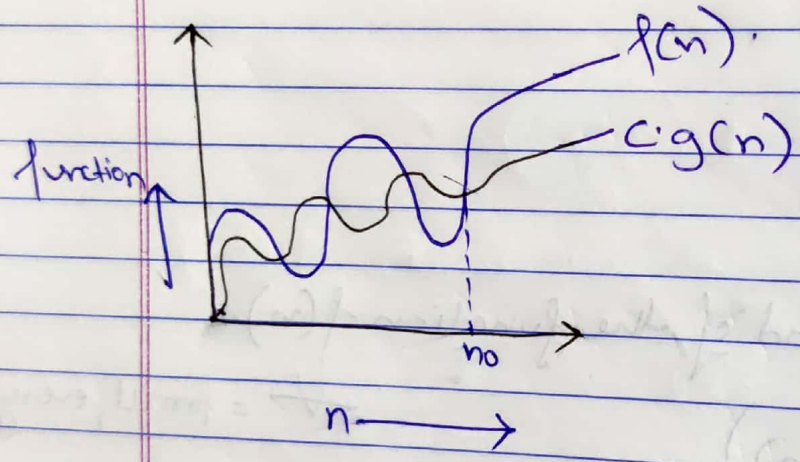
$g(n)$ is lower bound of function $f(n)$.

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

when

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

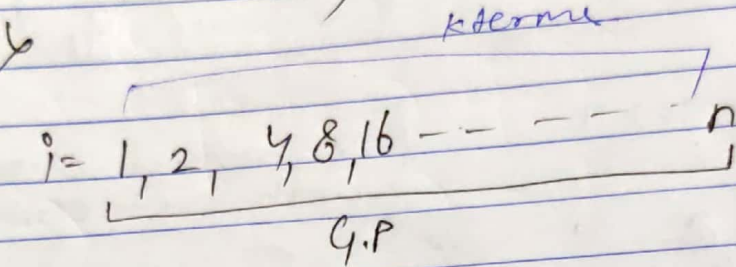
$\forall\ n > n_0$ and $\forall\ c > 0$



Date.
Page No.

Q2) Find Time complexity of.

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



k^{th} term =

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

$n = 1 * 2^{k-1}$

$n = 2^{k-1}$

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

$2n = 2^k$

taking log.

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

$\log_2(2n) = k$

$k = \log 2 + \log n$

$k = \log n$

Time complexity = $O(\log n)$.

Date
Page No.

Q3) $T(n) = [3T(n-1) \text{ if } n>0, \text{ otherwise } 1]$. $T(0)=1$

Put $n=n-1$

$T(n-1) = [3T(n-2)]$ — (B)

Put (B) in (A).

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

$T(n) = [3^2 T(n-2)]$ — (C)

Put $n=n-2$ in eq (A).

$T(n-2) = [3T(n-3)]$ — (D)

Put D in C

$T(n) = 3^2 \cdot [3T(n-3)] = 3^3 \cdot T(n-3)$.

Generalised eq^n

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

putting $n-k=0$

$\therefore T(n) = 3^k \cdot T(0)$

$\because T(0)=1$

$T(n) = 3^k$.

$k=n$

$T(n) = 3^n$

Time complexity $= O(3^n)$ ✱

Q4.) $T(n) = \begin{cases} 2T(n-1)-1 & \text{if } n>0, \text{ otherwise } 1 \end{cases}$ (T(0)=1)

——— (A)

Put $n = n-1$

$T(n-1) = 2T(n-2)-1$ ——— (B)

Put (B) in (A)

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

$T(n) = 4T(n-2)-2-1$ ——— (C)

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

Put $n = n-2$ in (A).

$T(n-2) = \{2T(n-3)-1\}$ ——— (D)

Put (D) in (C).

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

$= 8T(n-3)-4-2-1$ ——— (E)

Generalised form

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

put $n-k = 0$

$n = k$, $T(0) = 1$ [Given]

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

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

Date
Page No.

$$= 2^n - \underbrace{\left[2^{n-1} + 2^{n-2} \; \cdots \; 1\right]}_{k \text{ terms}}$$

$a = 2^{n-1}$, $r = 1/2$.

Sum of G.P series $= \frac{a(1-r^n)}{1-r} = \frac{2^{n-1}\left(1-\left(\frac{1}{2}\right)^{n-1}\right)}{1-\frac{1}{2}}$

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

Sum $= 2^n - 2$

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

$= 2^n - 2^n + 2$

$T(n) = 2$ (constant)

Time complexity $= O(1)$ Ans.

Q5)

Q5) Calc Time complexity ?

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

i	s
1	1
2	3
3	6
4	10
1	1
(	.
.	.
n	n^{th}

k time.

S = 1, 3, 6, 10 ..... n

k terms

k^{th} terms

$t_k = t_{k-1} + k$

$k = t_k - t_{k-1}$

$k = n - t_{k-1}$

$T.C = O(1+1+1+n-t_{k-1})$

$= O(3+n-t_{k-1})$ $\quad t_{k-1} = const.$

$T.C = O(3+n-c)$

$= O(n)$ A

Date
Page No.

Q.) Time Comp. of

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

i*i
1^2
2^2
3^2
4^2
5^2
.
.
n

$i*i = \underbrace{1^2, 2^2, 3^2, 4^2, 5^2 \ldots n}_{k \text{ terms}}$

k^{th} term $= t_k$

$t_k = k^2$

$n = k^2$

$\sqrt{n} = k$

T.C. $= O(1+1+\sqrt{n}) = O(\sqrt{n})$

Q.) Time Complexity of

```
Void fun (int n){          — O(1)
int i, j, k, count = 0;    — O(1)

for(i = n/2; i <= n; i++)          — (
    for(j = 1; j <= n; j *= 2)     — log n
        for(k = 1; k <= n; k *= 2) — log n
            count++;               — O(1)
}
```

Date.
Page No.

$i = \frac{n}{2}, \frac{n+2}{2}, \frac{n+4}{2}, \frac{n+6}{2} \quad \cdots\cdots n$

$= \frac{n+2\times0}{2}, \frac{n+2\times1}{2}, \frac{n+2\times2}{2}, \frac{n+2\times3}{2} \quad \cdots\cdots n$

k terms.

General term

$$t_k = \frac{n+k\times2}{2}$$

total terms $= k+1$

$$t_{(k+1)} = n$$

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

$$= n+2k+2 = 2n$$

$$2k+2 = n$$

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

i	j	k.
$n/2$	$\log_2 n$ time	$(\log n)^2$
$\frac{n+2}{2}$	$\log n$ time	$(\log n)^2$
$\frac{n+4}{2}$	$\log n$	$(\log n)^2$.
1		
1		
n	$\log n$	$(\log n)^2$.

Date
Page No.

$$\left(\frac{n}{2}-1\right)(\log n)^2$$

$$= \left(\frac{n}{2}\log^2 n - \log^2 n\right)$$

$$= \log^2 n \left[n - 1\right]$$

$$= O[n \log^2 n]$$

Q8) Time

```
function(int n){          — O(1)
if(n==2) return;          — O(1)

 for(i=1 to n)            — O(n)
 {  for (j=1 to n)        — O(n)
 {   printf("*");         — O(1)
 }
 }

   function(n-3);
}
```

function is calling

n n-3 n-6 n-9 ---- 1

d = -3

k terms.

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

$$1 = n + (k-1)(-3)$$

$$1 - n = k - 1$$

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

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

$k = \frac{n-1+3}{3} = \frac{n+2}{3}$

function gives a recursive call of $\frac{n+2}{3}$ time

Time compl = $O\left(\left(\frac{n+2}{3}\right) \times n \times n\right)$

$= O(n \times n \times n)$

$= O(n^3)$ ∴

Eg.) Time compl

```
Void fun (int n)
{  for (i=1 to n)
   {  for (j=1 ; j<=n; j=j+i)
         print ("x");
   }
}
```

for i=1 → j = 1, 2, 3, 4 ---- n = n

" i=2 → j = 1, 3, 5, ---- n = n/2

" i=3 → j = 1, 4, 7, 10 ---- n = n/3

for i=n ⇒ j = 1 --- n = 1

Date
Page No.

$$= \sum_{j=n}^{1} n + n/2 + n/3 + n/4 + \cdots\cdots + 1$$

$$= \sum_{j=n}^{1} n\left[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots\cdots \frac{1}{n}\right]$$

$$= \sum_{j=n}^{1} n \log n$$

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

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