Name - Samaridhi
Section - F
Roll no. - 24
University Roll no. - 2016985

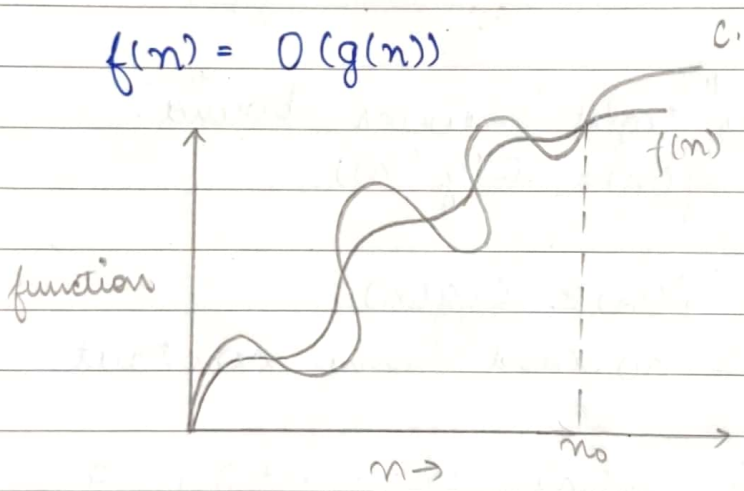
Tutorial - 1

1. Asymptotic Notations:
Asymptotic means tending to infinite or very large. These are used to tell the complexity for very large input.

Diff. Asymptotic Notations

Big O(o):

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



f(n) is "tight" upper bound.

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

if

$f(n) \leq c.g(n)$

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

eg:
```
for( i= 1; i<=n; i++) {
    print(i);        —— O(1)
```

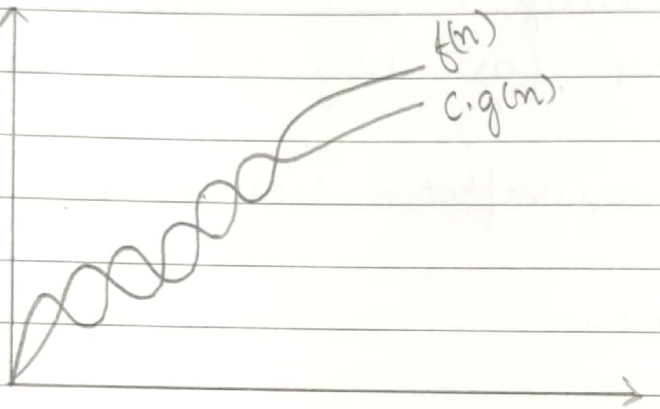
Date

}

$T(n) = O(n)$

ii) Big Omega (Ω):

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



g(n) is "tight" lower bound.

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

iff

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

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

Example: $f(n) = 2n^2 + 3n + 5$, $g(n) = n^2$

$0 \leq c \cdot g(n) \leq f(n)$

$0 \leq c \cdot n^2 \leq 2n^2 + 3n + 5$

$c \leq 2 + \frac{3}{n} + \frac{5}{n^2}$

On putting $n = \infty$ $\quad \frac{3}{n} \to 0$, $\frac{5}{n^2} \to 0$

$c = 2$.

Mohan

Date

$2n^2 \leq 2n^2 + 3n + 5$

On putting $n = 1$

$2 \leq 2 + 3 + 5$

$2 \leq 10$ true

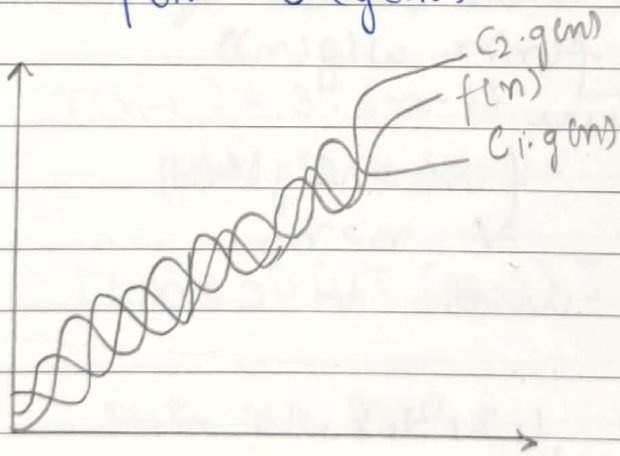
$C = 2, \quad n = n_0 = 1$

$0 \leq 2n^2 \leq 2n^2 + 3n + 5$

$f(n) = \Omega(n^2)$

iii) Bigh Theta (Θ):

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



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

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

if

$C_1 \cdot g(n) \leq f(n) \leq C_2 \cdot g(n)$

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

& some constant $C_1 > 0, \; C_2 > 0$.

Mohan

Date

iv) small oh (o) :

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

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

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

where, $f(n) \leq c \cdot g(n)$

$\forall \quad n > n_0$

and $\forall$ constant, $c > 0$.

v) small Omega (ω) :

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

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

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

where

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

$\forall \quad n > n_0$.

and $\forall \quad c > 0$.

2. $i = \underbrace{1, 2, 4, 8, 16 \; - - \; - - \; n}_{k \text{ terms}}$.

$a = 1$, $r = 2$.

k^{th} term.

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

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

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

Mohan

Date

taking log both sides.

$$\log_2 n = \log_2 2^{k-1}$$

$$\log_2 n = k-1 \log_2 2$$

$$\log_2 n = k-1 \quad [\because \log_a a = 1]$$

$$k = 1 + \log_2 n.$$

$$T(n) = O(k)$$
$$= O(1 + \log_2 n)$$
$$= O(\log_2 n).$$

3. $T(n) = 3T(n-1)$ — ①

put $n = n-1$ in eqn ①

$T(n-1) = 3T(n-2)$ — ②

put this value in ①

$T(n) = 3[3T(n-2)]$ — ③

put $n = n-2$.

$T(n-2) = 3T(n-3)$ — ④

put ④ in ③

$$T(n) = 9[3T(n-3)]$$
$$T(n) = 27\, T(n-3)$$

Mohan

Date

Generalized form:

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

put $n-k=0$

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

$T(0) = 1$

$T(n) = 3^n$

$O(3^n)$.

5.

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

$S = \underbrace{1, 3, 6, 10, 15 - - - - n.}_{K \text{ terms}}$

K^{th} term.

$t_n = t_{n-1} + K$

$K = t_n - K_{K-1}$ — (1)

loop runs K times.

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

but $t_{n-1} = C$ (const.)

$T.C. = O(3 + n.1)$

$= O(n)$

Mohan

Date

6. Time complexity of :

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

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

K^{th} term

$t_k = k^2$

$k^2 = n$

$k = n^{1/2}$

$T.C = O(1 + 1 + 1 + n^{1/2} + 1)$

$= O(n^{1/2}) = O(\sqrt{n})$.

7.

```
void function (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 = j*2)        —— log2 n
            for (k=1; k<=n; k = k*2) - log2 n
                count++;         —— O(1)
    };
```

$T.C = (\log_2 n)^2$

Mohan

Date

8.

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

for function call;

$n, n-3, n-6, n-9 \; -- \; -1$

A.P with $d = -3$

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

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

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

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

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

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

Time complexity $= \left(\frac{n+2}{3}\right)(n)(n)$

$= n3$

$= O(n3)$.

Mohan

Date

$i = \frac{n}{2}, \frac{n+2}{2}, \frac{n+4}{2}, \frac{n+6}{2}$ -- upto n.

$= \frac{n+0\times2}{2}, \frac{n+1\times2}{2}, \frac{n+2\times2}{2}, \frac{n+3\times2}{2}$ -- n

general term:

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

total terms $= k+1$

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

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

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

$$2k = n-2$$

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

i	j	k
$n/2$	$\log_2 n$ times	$(\log_2 n)^2$
$n+2/2$	"	"
$n+4/6$	"	"
⋮	⋮	⋮
$\frac{n}{(\frac{n}{2}-1)}$ times	$\log_2 n$ times	$(\log_2 n)^2$

$= \left(\frac{n}{2} - 1\right)(\log_2 n)^2 = O\left(\frac{n}{2}\log^2 n - \log^2 n\right)$

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

Mohan

Date

9. for $i=1 \rightarrow j = 1,2,3,4 \cdots n = n$
for $i=2 \rightarrow j = 1,3,5 \cdots n = n/2$
for $i=3 \rightarrow j = 1,4,7 \cdots n = n/3$
⋮
for $i=n \rightarrow j = 1 \cdots n = 1$

→ for $i=n \rightarrow j = 1, \cdots n = 1$

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

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

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

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

10. As given n^k & c^n
Relation b/w n^k & c^n is $n^k = O(c^n)$
as $n^k \leq a c^n$ ∀
$n \geq n_0$, for (a) constant ($a \geq 0$)

for ($n_0 = 1$)
∞ $c = 2$
$1^k \leq a 2^1$

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

Mohan