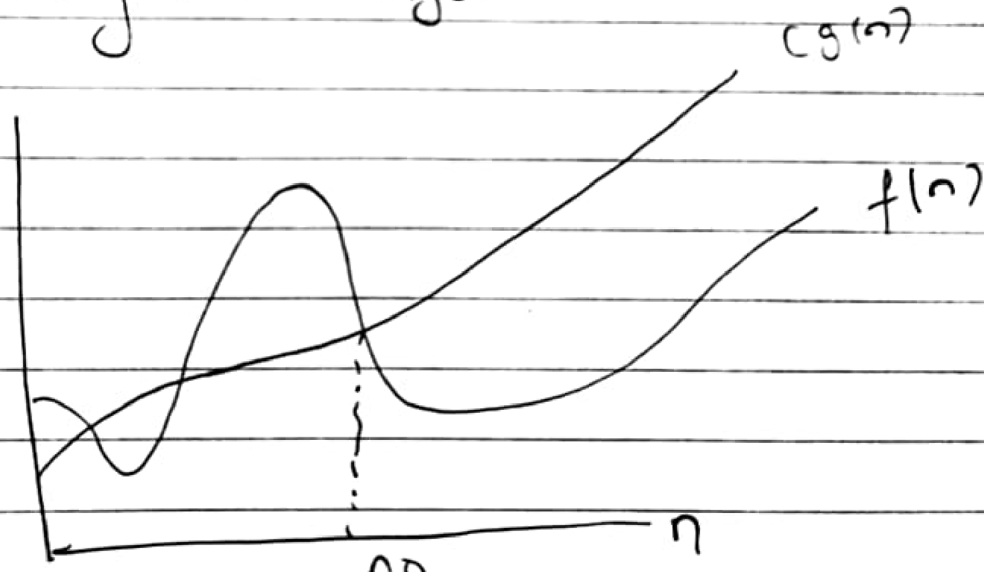
Q1.) Asymptotic Notation :- They are the mathematical notation used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

There are mainly three asymptotic notations :-

(i) Big - O - notation.

- provide worst complexity
- provide upper bound of an running time algo.



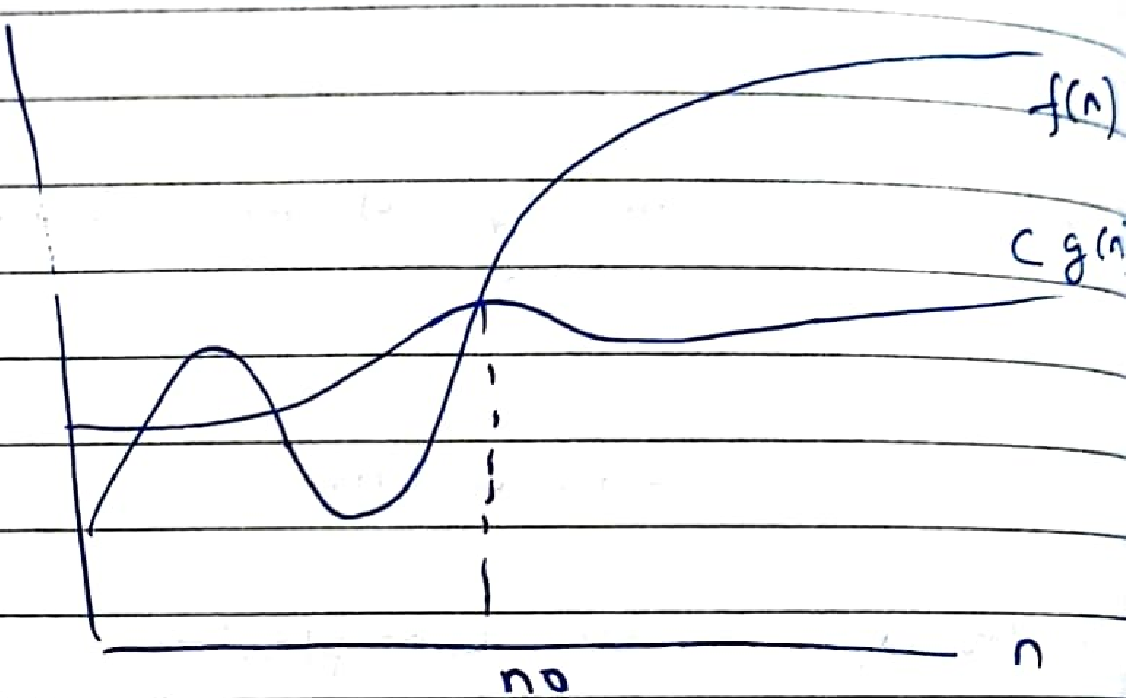
$f(n) = O(g(n))$

$O(g(n)) = \{ f(n)$: there exist positive constant c & n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$

Date:

(ii) omega Notation (Ω)

- provide best case complexity
- ref. lower bound of running time algo.

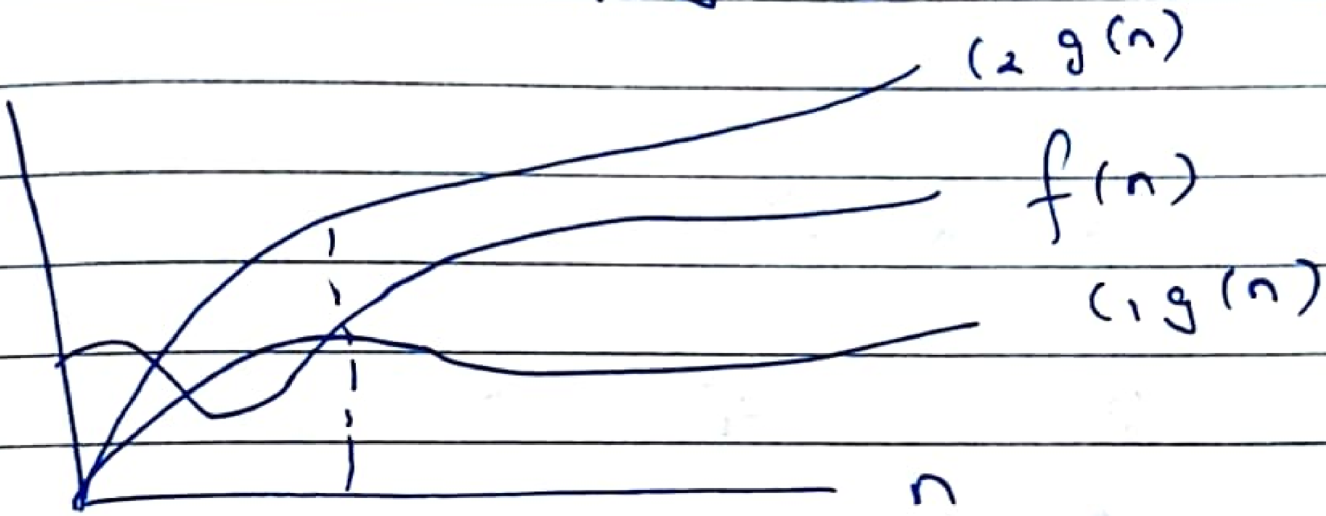


$$f(n) = \Omega(g(n))$$

$\Omega(g(n)) = \{ f(n)$: there exist positive constant c and n_0 such that $0 \le c\,g(n) \le f(n)$ for all $n \ge n_0 \}$

(iii) theta Notation (Θ- notation)

→ used for analysing avg. time complexity.



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

$\Theta(g(n)) = \{ f(n)$: there exist positive constant c_1, c_2, n_0 such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0 \}$

Q2.) for (i = 1 to n)

{

i = i * 2;

}

$i = 1, 2, 2^2 \ldots 2^k$

$2^k \le n \Rightarrow k = \log_2 n$

$\therefore \sum_{i=1}^{2^k} 1 \Rightarrow 1 + 1 + 1 \ldots k$ times

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

Q3 $T(n) = \begin{cases} 3T(n-1), & n > 0 \\ 1, & n \le 0 \end{cases}$

Using forward substitution.

$T(0) = 1$ — $O(1)$

$T(1) = 3T(0)$ "

$T(2) = 3^2 T(0)$ "

$\vdots$ $\vdots$

$T(n) = 3^n \times T(0)$ "

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

Date: / /

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

using forward subs.

$T(1) = 2T(0) - 1$, $\quad T(0) = 1$

$= 1$

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

$\vdots$

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

$\therefore T(n) = O(1)$ Ans

Date: / /

Q5.)

```
int i=1, s=1;

while (s <= n)
{
    i++; s = s+i;
    printf('#');
}
```

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

$S = 1+2;$

for $(i=2)$

$s = 1+2+3$

... for $(i=k)$

$1+2 \ldots + k <= n$

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

$$\Rightarrow O\left(\frac{k^2+k}{2}\right) \leq n$$

$$\Rightarrow O(k^2) <= n \Rightarrow k = O(\sqrt{n})$$

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

Q6.)
```
void function (int n) {
    int i, count = 0;

    for (int i = 1; i * i <= n; i++)
        count ++                       → O(1)
}
```

let 'K' be max +ive value such that

$K^2 \leq n$

$\because K = \sqrt{n}$

$i^2 \leq n$

$\therefore \sum_{i=1} 1 \Rightarrow 1 + 1 + \ldots K$ times

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

Q7

```
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 ++;
    }
  }
```

let 'm' be highest value of 2^m such that

$2^m <= n \quad \therefore \quad m = \log_2 n$

$\Rightarrow$ for $i = n/2 \quad j = \log n \quad k = \log$

$i = (n/2+1)$ " "

$\vdots$

$i = n$ " "

$\therefore \sum_{i=n/2}^{n} j \times k$

$\Rightarrow n/2 (\log n)^2 \quad \therefore$

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

Q8)

```
func (int n)
{
    if (n==1) return;

    for (i=1 to n)
    {
        for (j=1 to n) {
        printf(...)  → O(1)
        }
    }
    func (n-3);
}
```

for :- for (i=1 to n)

we get $j = n$ times every turn

$\therefore \quad i \times j = n^2$

Now,
$$\left.\begin{aligned} T(n) &= n^2 + T(n-3); \\ T(n-3) &= T(n-6) + (n-3)^2 \\ T(n-6) &= T(n-9) + (n-6)^2 \\ &\vdots \end{aligned}\right\} K \text{ terms}$$

$$T(1) = 1;$$

Now subs. each value in $T(n)$

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

Let

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

$$\therefore \quad k = (n-1)/3$$

$\because$ total terms $= k+1$

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

$$T(n) \approx n^2 + n^2 + n^2 \dots (k \text{ times } +1)$$

$$T(n) \approx kn^2$$

$$T(n) \approx \frac{(n-1)}{3} \times n^2$$

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

Q9.) function (int n)
{
for (i = 1 to n)
{
for (j = 1 ; j <= n ; j = j + i)
printf ("*");
}
}

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

$i = 2$ $j = 1 + 3 + 5 \cdots$ "

$i = 3$ $j = 1 + 4 + 7 \cdots$ "

⋮ ⋮

m^{th} term of AP is

$T(m) = a + d \times m$

$T(m) = 1 + d \times m$

$(n - 1)/d = m$

∴ for $i = 1$ — j: $(n-1)/1$ times

$i = 2$ — $(n-1)/2$ times

$i = 3$ — $(n-1)/3$ times

$i = n-1$ — 1

Date: /

We get

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

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

$$= n + n/_2 + n/_3 \cdots + \frac{n}{n-1} - n \times 1 +$$

$$= n\left[1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{n-1}\right] - n + 1$$

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

since $\int \frac{1}{x} = \log x$

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

Q10.) we have given

n^k & c^n

as $k \geq 1$ & $c > 1$.

$\Rightarrow$ for values $k \geq 1$, $c > 1$

we have $c^n \geq n^k$

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

$\forall$ $n \geq n_0$, & some constant $k_0 > 0$

$\Rightarrow$ $k_0 c^n \geq n^k$

for $c > 1$ & $n = 1$

we get.

$\Rightarrow$ $k_0 c \geq 1$

$\therefore$ $\boxed{c > 1 \text{ \& } n_0 = 1}$ Ans