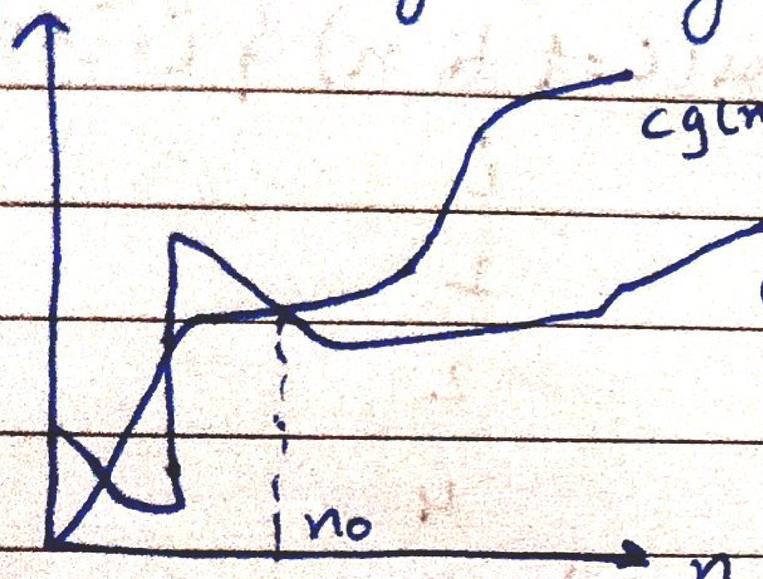
Q1 what do you mean by Asymptotic Notations. Define different Asymptotic notation with examples.

Asymptotic notation are used to tell the complexity of an algorithm when input is very large.

Big-O-Notation

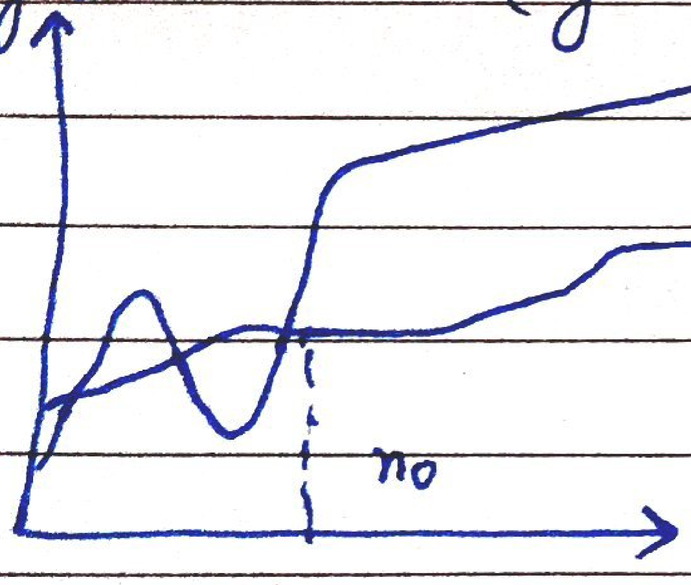
$f(n) = O(g(n))$: $g(n)$ is tight upper bound of $f(n)$



$f(n) = O(g(n))$
if and only if
$f(n) \leq cg(n)$
$\forall n \geq n_0$ and some constants c>0

Big Omega Notation :-

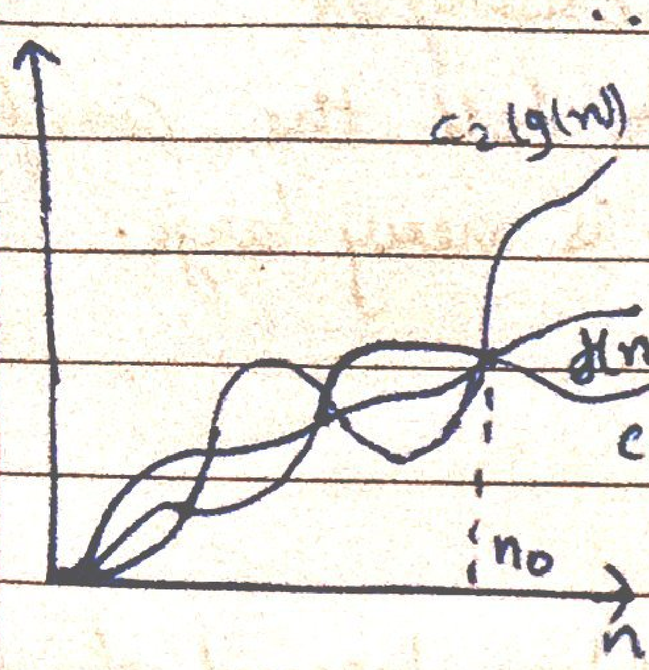
$f(n) = \Omega(g(n))$:. $g(n)$ is tight lower bound of $f(n)$



$f(n) = \Omega(g(n))$
if and only if
$f(n) \geq cg(n)$
$\forall n \geq n_0$ and some constants c>0

Theta Notation:- $f(n) = \Theta(g(n))$

$\therefore \Theta$ gives both 'tight' upper and 'tight' lower.



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

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

$\forall\ n > \max(n_1, n_2)$ & some constraints $c_1, c_2 > 0$

Q2- Sol what is time complexity of - for(i=1 to n) { i=i*2; }

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

i
1
2
4
8
⋮
n → k times

$n = 2^{k-1}$ $(x^m \text{ term} = a_1 x^{m-1})$

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

$2n = 2^k$

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

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

$\log_2(n) + 1 = k$

$O(\log_2(n))$

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

Sol- Using forward substitution

$T(0) = 1$

$T(2) = 3T(1) = 3 \times 1 = 3$

$T(3) = 3T(2) = 3 \times 3 = 9$

$T(4) = 3T(3) = 3 \times 9 = 27$

$T(n) = 1 + 3 + 9 + 27 + \cdots n$

$T(n) = \frac{a(r^n - 1)}{r-1} = \frac{1(3^n - 1)}{2}$ $a r^{n-1}$

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

$= O(3^n)$

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

Sol - using forward substitution

~~$T(0) = 1$~~

$T(1) = 2T(0) - 1$

$= 2 - 1 = 1$

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

$= 2T(1) - 1$

$= 1$

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

$= 2T(2) - 1$

$= 1$

$T(n) = n$

$O(n)$

Q5-

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

Time complexity = $O(n)$

Q6 Time complexity of

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

Sol- $n = i*i$

$n = i^2$

$i = \sqrt{n}$

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

Q7- Time complexity of

```
void function(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++;
        }
    }
}
```

Sol- $i = n/2$

$j = \log n$

$k = \log n$

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

PAGE No.
DATE

qb Time complexity of

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

sol no of asterisks $= n^2 + (n-3)^2 + (n-6)^2 + \cdots$

$= \frac{n(n+1)(2n+1)}{6}$ [∵ Sum of squares]

$= \frac{n^3}{6}\left(1+\frac{1}{n}\right)\left(2+\frac{1}{n}\right)$

$= \frac{n^3}{6}(1)(2)$

$= n^3/3$

Time complexity $= O(n^3)$

Q9- Time complexity of

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

Sol

i=1	j	1 to n
i=2	j	n/2 times
i=3	j	n/3 times
i=n	j	1 times

no. of iterations = $n + n/2 + n/3 + \cdots + 1$

$= \log n$

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