Tutorial - 1

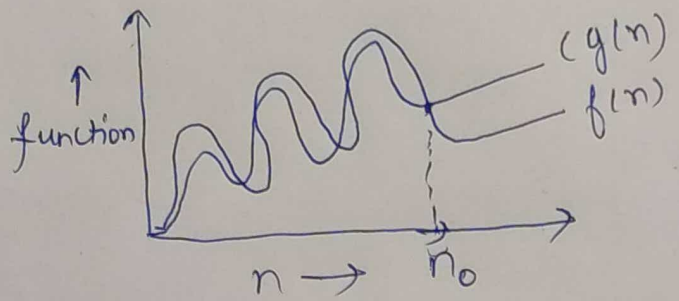
Ques1. What do you understand by Asymptotic Notations. Define different Asymptotic notations with examples.-

Ans. Asymptotic means - tending to infinity.
Asymptotic notations are used to represent the complexities of algorithms for asymptotic analysis. These notations are mathematical tools to represent the complexities.

There are 5 Asymptotic Notations:

1) **Big Oh Notation (O)**

Big-oh (O) notation gives an upper bound for a function $f(n)$ to within a constant factor.



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

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

iff

$f(n) \leq cg(n)$

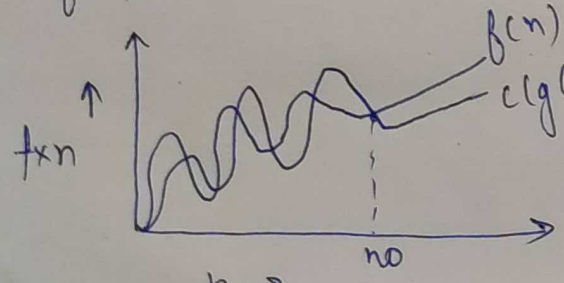
$\forall\ n \geq n_0$

for some constant, $c > 0$.

$g(n)$ is 'tight' upperbound of $f(n)$.

2) **Big Omega Notation (Ω)**

Big-Omega (Ω) Notation gives a lower bound for a function $f(n)$ to within a constant factor.



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

iff

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

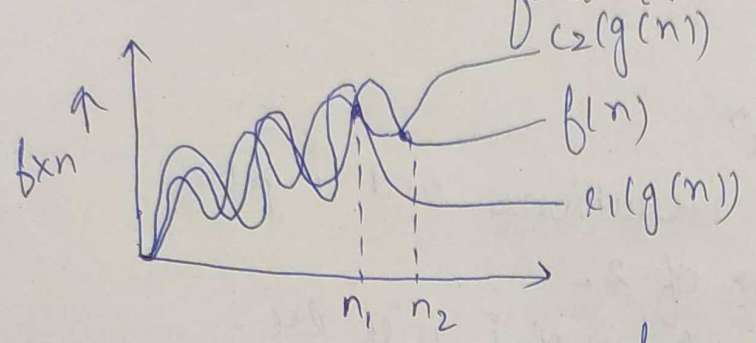
$\forall\ n \geq n_0$

for some constant, $c > 0$.

$g(n)$ is 'tight' lowerbound of $f(n)$.

3) Big Theta Notation (Θ)

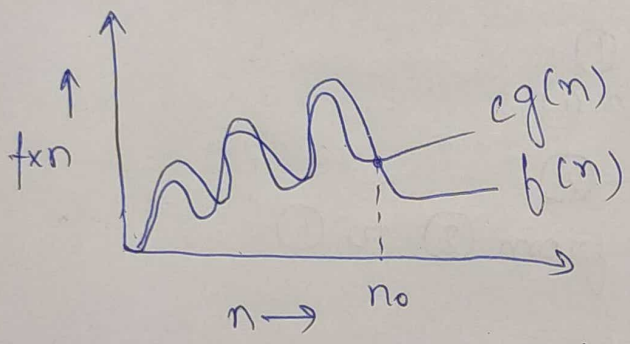
Big-Theta (Θ) Notation gives bound for a function f(n) within a constant factor.



$f(n) = \Theta(g(n))$
iff
$c_1 g(n) \leq f(n) \leq c_2 g(n)$
$\forall n \geq \max(n_1, n_2)$
for some constant $c_2 > 0$

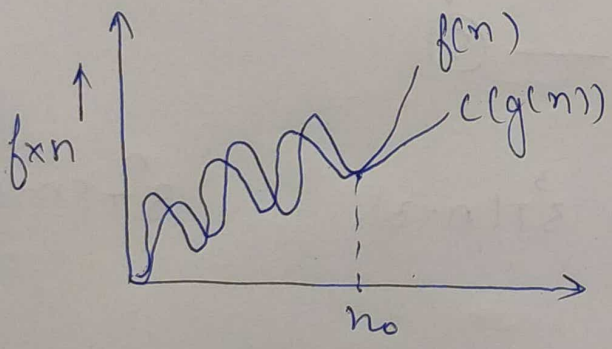
g(n) is both "tight" upper and lower bound of f(n).

4) Small Theta (θ)



$f(n) = \theta(g(n))$
g(n) is upperbound of f(n)
$f(n) = \theta(g(n))$
when
$f(n) < cg(n)$
$\forall n > n_0$
and for all constants, $c > 0$.

5) Small Omega (ω)



$f(n) = \omega(g(n))$
g(n) is lowerbound of f(n)
$f(n) = \omega(g(n))$
when
$f(n) > cg(n)$
$\forall n > n_0$
And for all constants, $c > 0$.

Ques 2. What should be the time complexity of -

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

Time complexity = $\log_2 n$

The loop executes for n iterations and i gets incremented by a factor of 2.

So, the corresponding values of i will be

1 2 4 8 16 32 - - - - n $\Rightarrow t_k = ar^{k-1}$

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

$2n = 2^k$

Ans. $\boxed{TC = \log_2 n.}$ $\log 2n = k \log 2$

$\log 2 + \log n = k \Rightarrow 1 + \log n = k$

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

$T(n) = 3T(n-1)$ — ①

putting $n = n-1$ in eq^n ①

$T(n-1) = 3T(n-2)$ — ②

putting value of $T(n-1)$ from ② to ①

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

$T(n) = 3^2 T(n-2)$ — ③

putting $n = n-2$ in eq^n ①

$T(n-2) = 3T(n-3)$ — ④

putting value of $T(n-2)$ in eq^n ③ from ④

$T(n) = 3^2 [3T(n-3)]$ — ⑤

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

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

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

$= 3^n$

Ans. Time complexity = $O(3^n)$.

Ques 4. $T(n) = \begin{cases} 2T(n-1)-1 & \text{if } n>0, \\ 1 & \text{otherwise} \end{cases}$

$$T(n) = 2T(n-1) - 1 \quad \text{—— (1)}$$

putting $n = n-1$ in eq^n (1)

$$T(n-1) = 2T(n-2) - 1 \quad \text{—— (2)}$$

putting value of $T(n-1)$ from (2) to (1)

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

$$T(n) = 2^2 T(n-2) - 2^1 - 2^0 \quad \text{—— (3)}$$

putting $n = n-2$ in eq^n (1)

$$T(n-2) = 2T(n-3) - 1 \quad \text{—— (4)}$$

putting $T(n-2)$ from (4) to (3)

$$T(n) = 2^2[2T(n-3)-1] - 2^1 - 2^0$$

$$T(n) = 2^3 T(n-3) - 2^2 - 2^1 - 2^0 \quad \text{—— (5)}$$

$$T(n) = 2^n T(n-n) - 2^{n-1} - 2^{n-2} - 2^{n-3} \dots 2^2 - 2^1 - 2^0$$

$$= 2^n - 2^{n-1} - 2^{n-2} - 2^{n-3} \dots 2^2 - 2^1 - 2^0$$

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

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

$$T(n) = O(1) \quad \underline{\text{Ans.}}$$

Ques 5. What should be the time complexity of -

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

We can define the terms 's' according to relation $s_i = s_{i-1} + i$. The value of 'i' increases by one for each iteration.

The value contained in 's' at the i^{th} iteration is the sum of the first 'i' positive integers.

Let k be the total no. of iterations

while loop terminates if : $1+2+3+\cdots+k = [k(k+1)/2] > n$

so $k = O(\sqrt{n})$

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

Ques 6. Time complexity of -

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

$i^2 <= n$

$i <= \sqrt{n}$

$i = 1, 2, 3, 4 \cdots \sqrt{n}$

$\sum_{i=1}^{n} 1+2+3+4+\cdots+\sqrt{n}$

$T(n) = \frac{\sqrt{n} \times (\sqrt{n}+1)}{2}$

$T(n) = \frac{n \times \sqrt{n}}{2}$

$T(n) = O(n)$ Ans.

Ques7. 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++
}
```

for $k = k * 2$

$k = 1, 2, 4, 8 \; - - - - n.$

G.P $\quad a = 1 \quad r = 2$

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

$$n = 2^k$$

$$\log n = k$$

i	j	k
1	log n	log n × log n
2	log n	log n × log n
:	:	:
n	log n	log n × log n

$\Rightarrow \quad n * \log n * \log n$

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

Ques 8. Time complexity of -

```
function ( int n)
{
  if (n==1)
    return;                    ⟶ O(1)
  for (i=1 to n) {             ⟶ O(n)
    for (j=1 to n) {           ⟶ O(n²)
      printf ("* ");
  } }
  function (n-3);              — O(n/3)
}
```

Using Master's Method:

$T(n) = T(n/3) + n^2$

$a = 1 \quad b = 3 \quad f(n) = n^2$

$c = \log_b a \Rightarrow \log_3 1 = 0$

$n^0 = 1 > n^2$

$T(n) = \Theta(n^2)$ Ans.

Ques 9. Time complexity of -

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

for $i=1 \Rightarrow j = 1, 2, 3, 4 \text{ ---- } n$ — n

for $i=2 \Rightarrow j = 1, 3, 5, \text{ ---- } n$ — $n/2$

for $i=3 \Rightarrow j = 1, 4, 7, \text{ ---- } n$ — $n/3$

⋮

for $i=n \Rightarrow j = 1 \text{ ---------- }$ — n

$$\sum_{i=n}^{1} \quad n + \frac{n}{2} + \frac{n}{3} + \frac{n}{4} + \text{ ---- } + 1$$

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

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

Ques 10. for the functions, n^k and c^n, what is asymptotic notations between these functions? Assume that k >= 1 and c > 1 are constants. find out the value of c and no for which relation holds.

Given : n^k, c^n

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

as $n^k \leq ac^n$

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

for $n_0 = 1$

$c = 2$

$\Rightarrow 1^k \leq a2^1$

$n_0 = 1$ and $c = 2$

Ans.