Shubham Lingwal
18-G

Tutorial-1

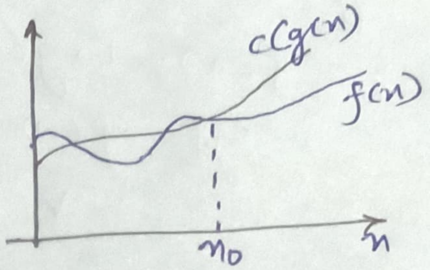
Q-1 What do you understand by Asymptotic notations. Define different asymptotic notations with examples -:

Ans: Asymptotic Notation consist two words i.e Asymptot that means tends to infinity. & Notations are used to represent the complexities by mathematical tools.

There are 5 Asymptotic Notations:

1) Big-Oh Notation (O)

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



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

iff

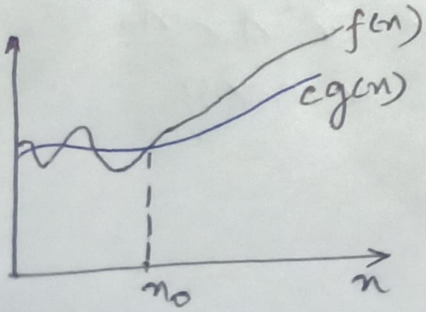
$f(n) \le c g(n) \quad \forall n \ge n_0$

for some constant, $c > 0$

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

2) Big-Omega Notation (Ω)

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



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

iff

$f(n) \ge c g(n)$

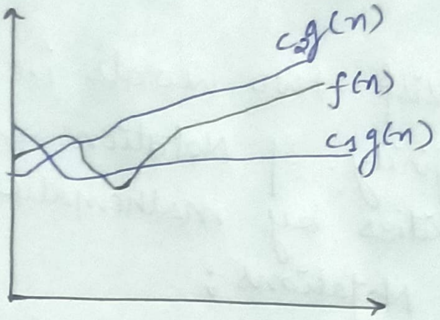
$\forall n \ge n_0$

for some constant, $c > 0$.

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

3. Big-Theta Notation (Θ)

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



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

iff

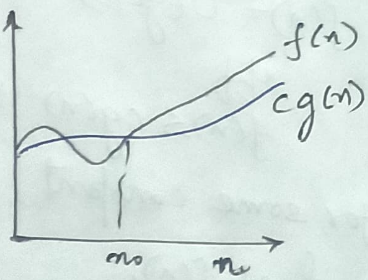
$c_1 g(n) \leq f(n) \leq c_2 g(n)$

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

for some constant $c_2 > 0$

→ $g(n)$ is both tight upper & lower bound of $f(n)$.

4.) Small Omega (ω) : It gives the lower bound of function $f(n)$



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

iff $f(n) > c g(n)$

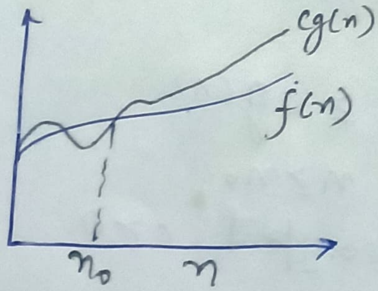
$\forall\, n > n_0,\ c > 0$

5. Small 'O' Notation : It gives the upper bound of function $f(n)$.

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

$f(n) < c g(n)\ \forall\, n > n_0$

$\forall\, c > 0$



Ques-2 What should be the time complexity of

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

Ans. Time complexity = $\log_2 n$
because the loop executes for n iterations & i gets incremented by a factor of 2.
So, the corresponding values of i will be
1 2 4 8 ..... n $\Rightarrow$ $f(n) = ar^{K-1}$

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

$2n = 2^K$

$\boxed{1 + \log n = K}$ $\Leftarrow$ $\log 2n = K \log 2$

$\boxed{TC = \log_2 n}$

Que-3 $T(n) = 3T(n-1)$ if $n > 0$, otherwise 1

Ans: $T(n) = 3T(n-1)$ — ①
Putting, $n = n-1$ in eq①
$T(n-1) = 3T(n-2)$ — ②
Putting value of $T(n-1)$ in eq ①
$T(n) = 3(3T(n-2))$
$T(n) = 3^2 T(n-2)$ — ③
Putting $n = n-2$ in eq ①
$T(n-2) = 3T(n-3)$ — Put in eq ③
$T(n) = 3^3 T(n-3)$ — ④
⋮
$T(n) = 3^n T(0)$
$= 3^n \Rightarrow O(3^n)$ Time complexity.

Que-4 $T(n) = \{2T(n-1) - 1$ if $n > 0$ otherwise $1\}$

Ans: $T(n) = 2T(n-1) - 1$ —①

putting $n = n-1$ in eq ①

$T(n-1) = 2T(n-2) - 1$ ②

putting values of $T(n-1)$ in eq ①

$T(n) = 2(2T(n-2) - 1) - 1$ — ③

$T(n) = 2^2 T(n-2) - 2^1 - 2^0$ — ③

putting $n = n-2$ in eq ①

$T(n-2) = 2T(n-3) - 1$ —④ put in eq ③

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

$T(n) = 2^3 T(n-3) - 2^2 - 2^1 - 2^0$ — ⑧

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

$= 2^n - 2^{n-1} \ldots\ldots 2^1 - 2^0$

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

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

Ques 5 What should be the time complexity of -

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

Ans: 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 contains 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: $1+2+3 \dots K$

$$= [K(K+1)/2] > n$$

So, $K = O(\sqrt{n})$

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

Que-6 Time complexity of :-

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

Ans: $i^2 <= n$

$i <= \sqrt{n}$

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

$\sum_{i=1}^{n} 1+2+3 \dots \sqrt{n} \Rightarrow T(n) = \frac{\sqrt{n} \times (\sqrt{n}+1)}{2}$

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

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

Q-7 Time complexity of :-

```
void function (int n)
{ int i, j, K, count=0;
  for (i= n/2; K<=n ; i++)
    for (j=1 ; j<=n; j=j*2)
      for (K = 1; K<=n; K=K*2)
          count ++;
}
```

Ans: for $K = K*2$

$K = 1, 2, 4, 8 \dots n$

G.P $\Rightarrow a = 1, r = 2$

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

$$n = 2^k$$

i	j	$\log n = k$
1	$\log n$	$\log n \times \log n$
2	$\log n$	:
:	:	:
n	$\log n$	$\log n \times \log n$

$\Rightarrow n \times \log n * \log n$

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

Ques-8 Time Complexity of

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

Ans: Using Master Method:

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

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

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

$n^0 = 1 > n^2 \Rightarrow T(n) = O(n^2)$ Ans

Que-9 Time Complexity of

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

Ans: for $i = 1 \Rightarrow j = 1, 2, 3 \ldots\ldots n$ $\quad n$

for $i = 2 \Rightarrow j = 1, 3, 5 \ldots\ldots n$ $\quad n/2$

for $i = 3 \Rightarrow j = 1, 4, 7 \ldots\ldots n$ $\quad n/3$

$\vdots$

for $i = n \Rightarrow j = 1 \ldots\ldots n$ $\quad n$

$$\sum_{i=1}^{n} \cdot\ n + \frac{n}{2} + \frac{n}{3} \ldots\ldots + 1$$

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

$\Rightarrow n(\log n)$ $\qquad$ $T.C = O(n\log n)$

Que-10 For the functions, n^k & c^n, what is asympto[tic]
notations between these functions?
Assume that $k >= 1$ & $c > 1$ are constants.
Find out the value of c & no. for which relation holds.

Ans: Given : n^k, c^n

$n^k = O(c^n)$ as $n^k \leq ac^n \ \forall \ n \geq n_0$ &
some constant $a > 0$.

for $n_0 = 1$
$c = 2$
$\Rightarrow 1^k \leq a2^1$
$n_0 = 1$ & $c = 2$