DESIGN AND ANALYSIS OF ALGORITHMS

Tutorial - 1

Name- Satyam Kumar
Section :- D
Roll no:- 34
University Roll:- 2017004

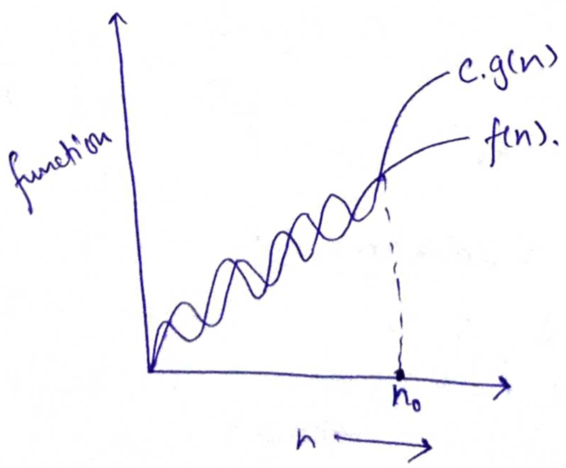
1. What do you understand by Asymptotic Notations. Define different Asymptotic notation with examples.

Ans:- Asymptotic Notation:- These notations are used to tell the complexity of an algorithm, when input is very large. These are mathematical notations used to describe running time of an algorithm when the input tends towards a particular value or a limiting value.

- Different Asymptotic Notations:

i) Big - Oh (O):-

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



g(n) is "tight" upper bound.

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

iff

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

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

E.g:-

```
for (i=1; i<=n; i++)
{
    print (i);      — O(1).
}
```

i
1
2
3
¦

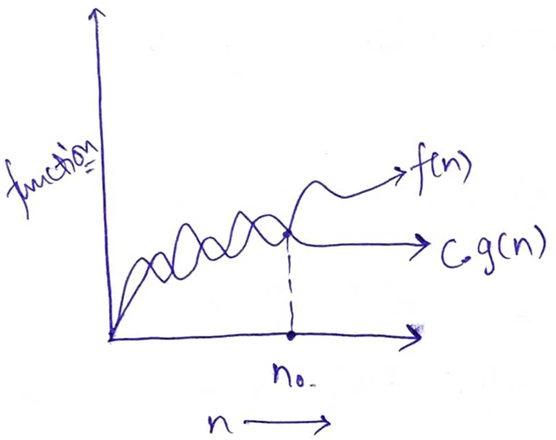
~~n times~~

n times $\Rightarrow O(n)$

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

ii) Big Omega (Ω):-

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



g(n) is "tight" lower bound.

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

iff,

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

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

E.g. $f(n) = 2n^2 + 3n + 5$, $g(n) = n^2$.

$\Rightarrow \because \; 0 \leq c.g(n) \leq f(n)$

$\Rightarrow 0 \leq c.n^2 \leq 2n^2 + 3n + 5$

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

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

$\Rightarrow c = 2$,

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

On putting $n = 1$,

$2 \leq 2 + 3 + 5$

$2 \leq 10$ True.

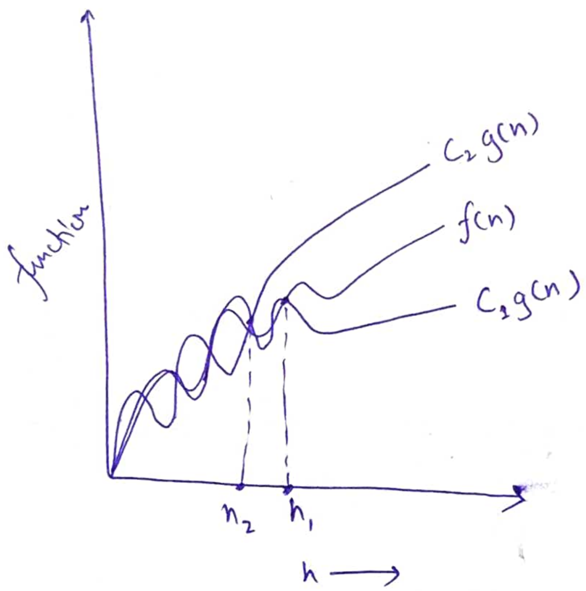
$\Rightarrow \boxed{c = 2, \quad n = n_0 = 1}$

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

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

iii) Big- Theta (Θ):-

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



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

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

iff

$c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$

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

and some constant, $c_1 > 0$, $c_2 > 0$.

Eg:- $f(n) = 10 \log_2 n + 4$. , $g(n) = \log_2 n$.

$\Rightarrow f(n) \le c_2 \cdot g(n)$

$\Rightarrow 10 \log_2 n + 4 \le 10 \log_2 n + \log_2 n$

$10 \log_2 n + 4 \le 11 \log_2 n$

$c_2 = 11$

$\Rightarrow$ $4 \le 11 \log_2 n - 10 \log_2 n$

$4 \le \log_2 n$

$16 \le n$

Here, $\forall\ n \ge 16$

$n_2 = 16$

& $c_2 = 11$

$f(n) \geq c_1 g(n)$

$10 \log_2 n + 4 \geq 2 \log_2 n$

$c_1 = 1$, $n > 0$

$\Rightarrow$ $n_1 = 1$ $\Rightarrow$ $n_0 = \max(n_1, n_2) \Rightarrow$ $n_0 = 16.$

$\Rightarrow$ $\log_2 n \leq 10 \log_2 n + 4 \leq 11 \log_2 n$

$c_1 = 1$

$c_2 = 11.$

$\Rightarrow$ $\theta(\log_2 n)$.

iv) Small oh (o) :-

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

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

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

when, $f(n) < c \cdot g(n)$

$\forall \; n > n_0$

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

v) Small Omega (ω) :-

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

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

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

when

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

$\forall \; n > n_0$

and $\forall \; c > 0$.

3: What should be the complexity of -

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

→

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

⇒ $a = 1, \quad r = 2.$

⇒ k^{th} term :-

$t_k = ar^{k-1}$

⇒ $n = 1 \cdot 2^{k-1}$

$n = 2^{k-1}$

take $\log_2$ both sides,

⇒ $\log_2 n = \log_2 2^{k-1}$

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

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

⇒ $k = 1 + \log_2 n$

⇒ $T(n) = O(k)$

$= O(\cancel{1} + \log_2 n)$

$= O(\log_2 n).$

3. $T(n) = \{3T(n-1)$ if $n>0$, otherwise $1\}$

$\rightarrow$

$\because T(n) = 3T(n-1)$ —— ①

put $n = n-1$ in eqⁿ ①,

$\Rightarrow T(n-1) = 3T(n-2)$ —— ②

put this value in eqⁿ ①,

$T(n) = 3[3T(n-2)]$ —— ③

put $n = n-2$ in eqⁿ ①,

$T(n-2) = 3T(n-3)$ —— ④

put this value in eqⁿ ③,

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

$T(n) = 27T(n-3)$

$\Rightarrow$ Generalised form:-

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

put $n-k=0$

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

but $T(0) = 1$

$\Rightarrow T(n) = 3^n$

$\Rightarrow O(3^n)$.

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

$\Rightarrow$

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

put $n-1$ in equation (1)

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

put this value in eq^n (1)

$\Rightarrow$

$$T(n) = 2\left[2T(n-2) - 1\right] - 1$$

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

put $n = n-2$ in eq^n (1),

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

put this value in eq^n (3),

$$\Rightarrow \quad T(n) = 4\left[2T(n-3) - 1\right] - 2 - 1$$

$$\Rightarrow \quad T(n) = 8T(n-3) - 4 - 2 - 1$$

$\Rightarrow$ Generalised form:-

$$T(n) = 2^K T(n-K) - 2^{K-1} - 2^{K-2} \text{ ------ } - 1$$

put $n - K = 0$

$\Rightarrow n = K$, $T(0) = 1$ (Given).

$\Rightarrow$

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

$$= 2^n - 2^{n-1} - 2^{n-2} \text{ ------- } - 1$$

$$= 2^n - \underbrace{\left[2^{n-1} + 2^{n-2} + \text{ ------- } + 1\right]}_{K \text{ terms}}$$

$$\Rightarrow \quad a = 2^{n-1}, \quad r = \frac{1}{2}.$$

$$\Rightarrow \quad \text{Sum of G.P} = \frac{2^{n-1}\left[1 - \left(\frac{1}{2}\right)^{n-1}\right]}{1 - \frac{1}{2}} = 2^n - 2.$$

$$\Rightarrow T(n) = 2^n - [2^n - 2] = 2$$

$$\Rightarrow O(2) = O(1).$$

5. What should be time complexity of -

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

i	s
1	1
2	3
3	6
⋮	10
n	15
	⋮
	n
	K times.

$\Rightarrow$

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

$\Rightarrow$ K_{th} term,

$$t_K = t_{K-1} + K$$

$$\Rightarrow K = t_K - t_{K-1} \quad \text{--- (1)}$$

Now, from series,

$t_2 - t_1 = 2$

$t_3 - t_2 = 3$

$t_4 - t_3 = 4$

⋮

$\Rightarrow K = n - t_{K-1}$

$\Rightarrow$ loop runs K times.

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

but, $t_{K-1} = c$ (constant)

$\Rightarrow T.C = O(\cancel{3}+n-\cancel{c})$

$= O(n).$

6. Time complexity of -

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

$i*i$
1^2
2^2
3^2
4^2
5^2
⋮
n

$\Rightarrow i*i \Rightarrow \underbrace{1^2, 2^2, 3^2, 4^2, 5^2 ----- n,}_{k \text{ terms.}}$

$\Rightarrow$ K_{th} term:-

$$t_K = K^2$$

$$\Rightarrow K^2 = n$$

$$K = n^{1/2}$$

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

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

7. Time complexity of -

```
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) times
      for (K=1; K<=n; K=K*2)   — log2(n) times
         count++;   ← O(1) .
}
```

$i \Rightarrow n/2, \ \frac{n+2}{2}, \ \frac{n+4}{2}, \ \frac{n+6}{2} \text{ ----- upto } n$

$\Rightarrow \frac{n+0\times 2}{2}, \ \frac{n+1\times 2}{2}, \ \frac{n+2\times 2}{2}, \ \frac{n+3\times 2}{2} \text{ ----upto } n$

$\Rightarrow$ General term:-

$$t_K = \frac{n+K*2}{2}$$

total terms = $K+1$

$$\Rightarrow t_{K+1} = n$$

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

$$n + 2K + 2 = 2n$$

$$2K = n-2$$

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

$\Rightarrow$

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

$$\Rightarrow (\frac{n}{2}-1)(\log_2 n)^2$$

$$\Rightarrow O(\frac{n}{\cancel{2}}\log_2^2 n - \log^2 n)$$

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

Q. 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(n)
    printf ("*");         — O(1)
    }
  }
  function (n-3);
}
```

for function call,

$$\underbrace{n, n-3, n-6, n-9 \ldots\ldots 1}_{K \text{ terms.}}$$

$\Rightarrow$ AP with $d = -3$.

$\therefore \Rightarrow \quad l = a + (K-1)d$

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

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

$\Rightarrow \quad K-1 = \frac{n-1}{3}$

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

$$\boxed{K = \frac{n+2}{3}}$$

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

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

$= n^3$

$\Rightarrow O(n^3)$.

9: Time complexity of -

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

i	j
1	n times
2	$(n+1)/2$ times
3	$(n+2)/3$ times.
4	$n+3/4$ times.
⋮	
n	$\frac{[n+(n-1)]}{n}$ times

$$\Rightarrow T(n) = n + \underbrace{\left(\frac{n+1}{2}\right) + \left(\frac{n+2}{3}\right) + \left(\frac{n+3}{4}\right) + \cdots \left(\frac{2n-1}{n}\right)}_{K \text{ terms}}$$

$\Rightarrow$ General term:-

$$T_K = \frac{n+K}{K+1}$$

$$\Rightarrow \cancel{\frac{2n-1}{n}}$$

Sum of k terms.

$$\Rightarrow S_K = \sum_{m=1}^{K} \left(\frac{n+K}{K+1}\right)$$