DAA

Turtorial -1

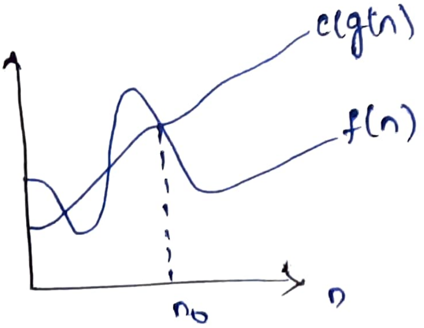
1) what do you understand by Asymptotic notations. Define different Asymptotic notations with examples.

Sol Asymptotic Notations are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or limiting value.

There are 3 asymptotic Notations big O, big Theta Θ, big Omega (Ω)

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

$f(n) \leq c * g(n)$ ∀ n, ~~for~~ some constant $c > 0$



Big O gives the upper bound of a function.

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

Big Theta (Θ)

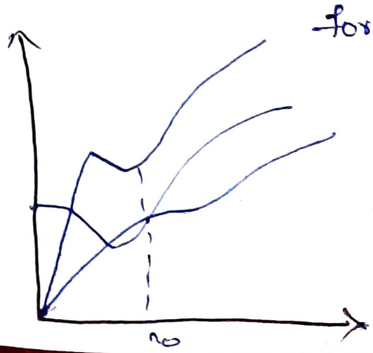
$f(n) = \Theta(g(n))$

g(n) is both "tight" upper bound of function f(θ)

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

$n > \max(n_1, n_2)$

for some const $c_1 > 0$ & $c_2 > 0$



Theta bounds the function with in constant factors.

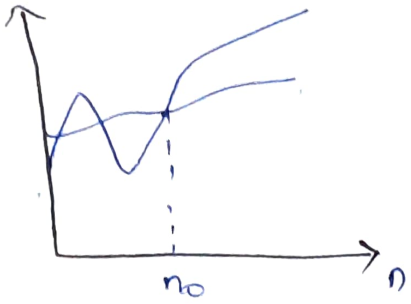
$f(n) = \Theta(g(n)$

8 Big Omega (Ω)

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

iff $f(n) \geq c\,g(n)$

$\forall\ n \geq n_0$ & for some constant $c > 0$



Omega gives the lower bound of a function.

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

② what should be the time complexity of -

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

Sol

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

i	i
1	2
2	4
4	8
8	'
'	'

$n = 2^k$

$\log n = \log 2^k$

$\boxed{k = \log_2 n}$

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

$T(1) = 1$

using subtract & conquer

$T(n) = 3T(n-1) \to$ ①

~~$T(n) = aT(n-b) + f(n)$ → ②~~

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

computing (1&) 2

$a \neq 3$

$b \neq 1$

since $f(n) \neq 0$

$k \neq 0$

$T(n) \neq O(n^k/2)$

$T(n-1) = 3T(n-2) \rightarrow ②$

$T(n-2) = 3T(n-3) \rightarrow ③$

$T(n) = 3 * T(n-1)$

$= 3 * 3 * T(n-2)$

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

$\vdots$

$= 3 * n\, T(n-n)$

$= \underline{\underline{3n}}.$

$O(3n).$

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

$T(n) = \{ 2T(n-1) - 1 \rightarrow ①$

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

$T(n-1) = 2T(n-2) - 1 \rightarrow ②$

$T(n-2) = 2T(n-3) - 1 \rightarrow ③$

$T(n-3) = 2T(n-4) - 1 \rightarrow ④$

$T(n) = 2 (2T(n-2) - 1] - 1 = 4T(n-2) - 2 - 1 = 4T(n-2) - 3$

$T(n) = 4 (2T(n-3) - 1] - 3 = 8T(n-3) - 4 - 3 = 8T(n-3) - 7$

$T(n) = 8 [2T(n-4) - 1] - 7 = 16T(n-4) - 8 - 7 = 16T(n-4) - 15$

$T(n) = 16T(n-4) - 15 \rightarrow ⑤$

<u>Genf</u> $T(n) = 2^k\, T(n-k) - (2^k - 1)$

since $T(n-1)$ is reducing by 1

Let $n - k = 1 \Leftrightarrow n = k$

$T(n) = 2^n\, T(1) - (2^n - 1)$

$O(1)$

$= O(2^n * 1 - 2^n + 1) = O(1) = \underline{\underline{O(1)}}.$

(5) what should be the complexity of

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

Sol:- Let loop executes k times

Loop will execute till $s \leq n$

After 1st

$S = S+1$

After 2nd

$S = S+1+2$

After 3rd

$s = s+1+2+3$

⋮

$S = s+1+2+ - - - k$ (since the loop goes k times)

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

$O(K^2) \leq = n$

$k = O(\text{root}(n))$. = $k = \sqrt{n}$

(6) Time complexity of void function (int n)

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

Sol:-

$i^2 <= n$

$i <= \sqrt{n}$

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

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

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

$$= \frac{n \times \sqrt{n}}{2}$$

$T(n) = O(n)$ // lower order terms can be ignored.

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

7) Time complexity of

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

Sol for $i = n/2$; $i <= n$; $i++$ //run for $n/2$ $\therefore O(n/2) = O(n)$

for $j = 1$, $j <= n$; $j = j*2$ // $k = \log_2 n$ $\therefore O(\log_2 n)$

for $k = 1$; $k <= n$; $k = k*2$ // $x = \log_2 n$ $\therefore O(\log_2 n)$

$$T(n) = O(n) \times O(\log_2 n) \times O(\log n)$$

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

$= O(n . \log^2 n) = O(n \log^2 n)$ Constant can be ignored

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

(8) Time complexity of

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

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

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

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

$n^0 = 1 > f(n) = n^2$

$T(n) = \theta(n^2)$

(9) Time complexity of

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

for $i=1 \Rightarrow j = 1, 2, \ldots ---n = n$

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

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

⋮

for $i=n \Rightarrow j=1 = n/n = 1$

$$\sum_{j=1}^{n} = n + \frac{n}{1} + \frac{n}{2} + --- ,$$

$$n[1 + 1/2 + 1/3 + --- 1/n] \quad \sum_{j=n}^{1} \quad n[\log n]$$

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

(10) for the functions n^k & c^n, what is the asymptotic relationship between these functions?

Assume that k>=1 & c>1 are constants. Findout the value of c & n^o for which relation holds

Sol:- Given n^k & c^n

relation b/w n^k & c^n is

$$n^k = O(c^n) \quad \text{as } n^k \leq c^n$$

$$\forall n \geq n_0 \text{ \& some constant } a>0$$

for $n_0 = 1$

$c = 2$

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

$\Rightarrow n_0 = 1$ & $\underline{\underline{c = 2}}$.