NAME - Astha choudhary ROLL:NO - 30

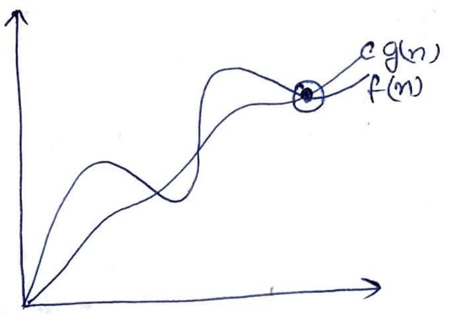
SECTION - DS1

Q1 Asympotic notations are used to tell the Complexity of an algorithm when input is very large.

Type of asymtotic notation :

→ Big O (O) : $f(n) = O\, g(n)$

g(n) is "tight" upper bound of func



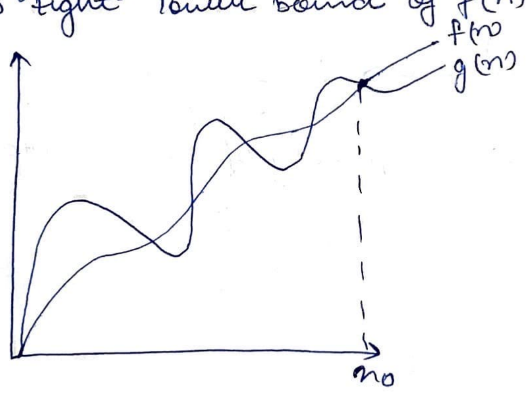
$f(n) = O(g(n))$

if $f(n) \leq c\,g(n)$

$\forall\, n > n_0$ & same constant $c > 0$

→ <u>Big Omega Notation (Ω)</u> : $f(n) = \Omega(g(n))$

g(n) is "tight" lower bound of f(n)



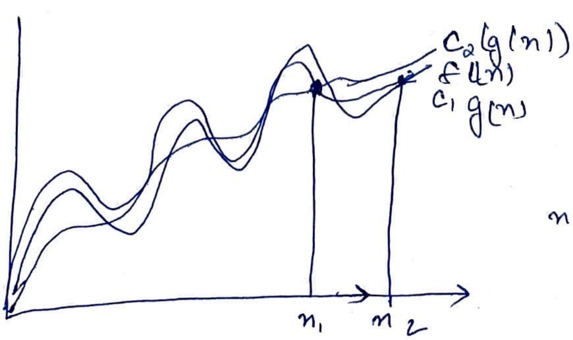
$f(n) \geq c\,g(n)$

$\forall\, n \geq n_0$

& some constants $c > 0$

→ <u>Theta Notation (Θ)</u> : $f(n) = \Theta(g(n))$

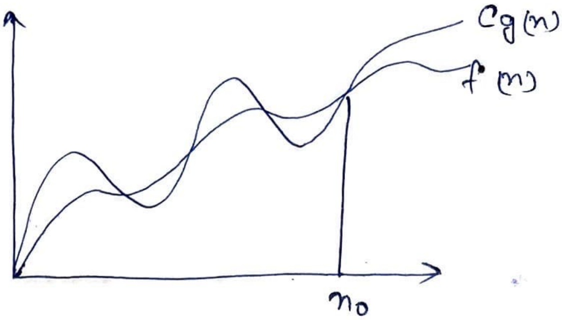
It gives both "tight" upper and tight lower bounds



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

n_1 & n_2 Can be Same also

→ Small theta (θ) : g(n) upper bound of f(n), f(n) = θ(g(n)))

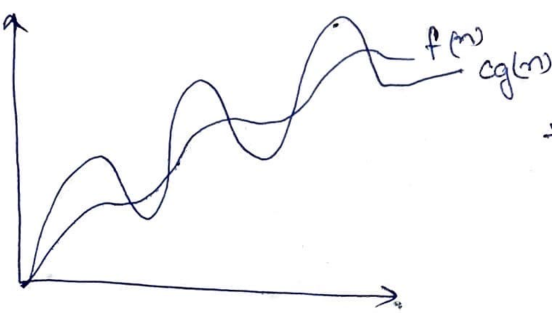


if $f(n) < cg(n)$

$n > n_0$ & Const $c > 0$

→ Small Omega (Ω) : g(n) is lower bond of f(n)

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



if $f(n) > c\,g(n)$

$\forall\ n > n_0$ & $\forall$ Constant $c > 0$

Q2 What should be time complexity of for (i = 1 to n)

{ i = i*2;

}

⇒ Time complexity is O(log n)

for (i = 1 to n)

i = i * 2

$i = \frac{1}{2^0} \cdot \frac{1}{2^1} \cdot \frac{4}{2^2} \cdot \frac{8}{2^3} \cdots \frac{n}{2^K}$

$GP = ar^{k-1}$

$n = 1 \cdot 2^{k-1}$ $\qquad n = \frac{2^k}{2}$

$\log 2n = k \log 2$

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

$$\boxed{T_n = O(\log(n))}$$

Q3 $T(n) = \{ 3T(n-1)$ if $n>0$, otherwise $1 \}$

$T(0) = 1$

$T(1) = 3T(0) = 3$

$T(2) = 3T(1) = 9 = 3^2$

$T(3) = 3T(2) = 27 = 3^3$

$\boxed{T(n) = 3^n = O(3^n)}$

Q4 $T(n) = \{ 2T(n-1)-1$ if $n>0$, otherwise $1 \}$

$T(n) = 2T(n-1)-1$ — ①

put $n = n-1$

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

put $T(n-1)$ in ①

$T(n) = 4T(n-2)-3$ — ②

put $n = n-2$

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

put in ②

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

$= 8T(n-3)-5 = 2^k T(n-k)$

$\therefore$ ~~n~~ $(k \cdot (n-k)) = 1$

$k = n-1$

$T(n) = 2^{n-1} T(n-n+1) - 5 = 2^{n-1} T(1) - 5$

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

$\boxed{T(n) = O(2^n)}$

Q5 what should be the Time Complexity of

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

$i = 1 \Rightarrow i++,\ i = 2$

$s = 3$
$i = 3$
$s = 6$
$i = 4$
$s = 10$
$i = 5$
$s = 15$

$s = s + 1 + 2 + 3 + 4 + \ldots k$

$= k(k+1)/2 \le n$

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

$\doteq k^2 \le n$

$= k < \sqrt{n}$

$\boxed{T(n) = O\sqrt{n}}$

Q6 void fn(int n)

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

$i = 1\ 2\ 3\ 4$

$i^2 = 4\ 9\ 16\ --\ k^2$

$k^2 \le n$

$k \le \sqrt{n}$

$\boxed{T(n) = O\sqrt{n}}$

Q7 void fn(int n)

```
{ int i, j, k , count = 0;
  for( i = n/2 ; i <= n ; i ++ ) .  -> T(n/2)
  { for( j = 1 ; j <= n ; j = j * 2)  -> log(n)
    { for( K = 1 ; K <= n ; K = K * 2);  -> log(n)
        count ++ ;
    }
  }
}
```

$T(n) = T(n/2) * \log(n) * \log(n)$

$= \frac{n}{2} * \log n^2$

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

Q8 fun(int n)

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

n = 6

* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* *
* *

$\boxed{T(n) \Rightarrow O(n^2)}$

Q9 void function (int n)

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

Total iteration $1 + \frac{n}{2} + \frac{n}{3} + \ldots + 1$

which is harmonic sum

$\cancel{T(n)} = O(\log n)$

$$\boxed{T(n) = O(n \log n)}$$

10 For the func n^k and c^n, what is the asymptotic relationship b/w these functions?
Assume that $k >= 1$ & $c > 1$ and constants. Find out the value of c and n_0 for which relation holds.

given n^k c^n

$k >= 1$ $c > 1$

$c = 2$

$k = 1$

$n^1 = 2^n$

$n \cdot 2^n$

This shows that c^n grows faster than n^k.