Date __/__/____
Page _____

Name : Jansi Sabharwal.
Sub : Design and Analysis of algorithm.
Sec : ML
Roll no. : 33

TCS-409

Ques What do you understand by Asympotic notations Define different Asympotic notation with examples.

=) Asympotic notations are the mathematical tools which are used to tell the complexity of an algorithm when the input is very large.

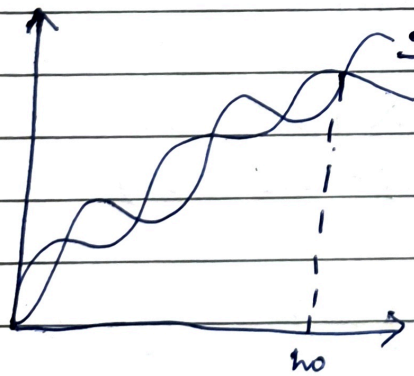
$O(n^2)$ = no. of instructions
where n is number of input.

Different Asympotic notations are —

① **Big O Notation (O)**

It describes the upper bound of an algorithm's time complexity in the worst-case scenario.

eg — $f(n) = O(g(n))$



$g(n)$ is 'right' upper bound of $f(n)$

$f(n) = O\,g(n)$
if $f(n) \leq c\,g(n)$
$\forall\ n \geq no$
and for some constants $c > 0$

- If the algo has time complexity of $O(n^2)$, it means algo's runtime grows quadratically with the size of input.

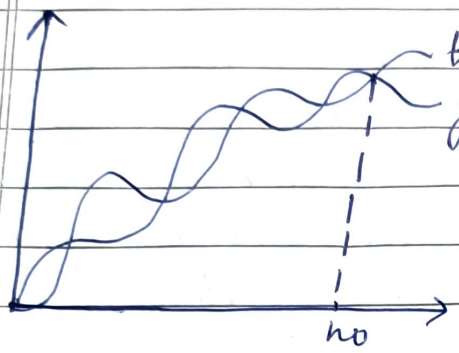
Be Positive

Date __/__/____
Page _____

2) Omega notation (Ω):

Omega notation describes the lower bound of an algorithm's time complexity in the best-case scenario.

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



$f(n) = \Omega g(n)$ if $f(n) \geq c g(n)$

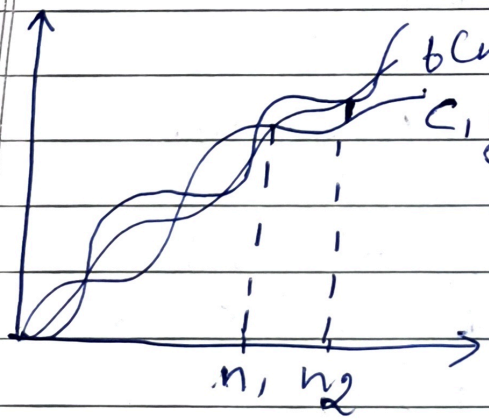
$\forall$ $n \geq no$, and for some constants $c > 0$

- If an algorithm has a time complexity of $\Omega(n)$ it means the algorithm's runtime grows at least linearly with the size of the input

3) Theta notation (Θ):

Theta notation describes both the upper and lower bounds of an algorithm's time complexity, providing a tight bound.

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



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

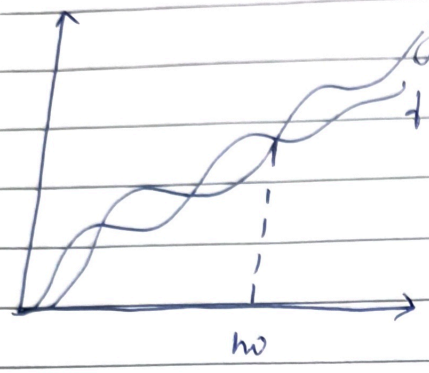
$c_1 g(n) \leq f(n) \leq c_2 g(n)$

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

Be Positive

Date / /
Page

4) Small O notation (o):
It describes an upper bound on a function that is not tight.



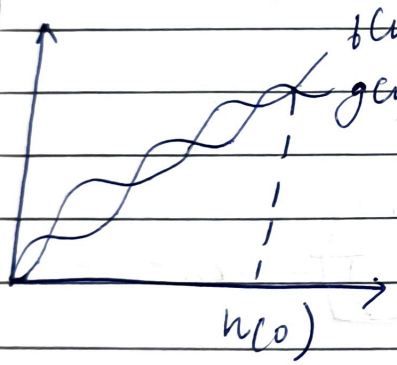
$f(n) < c\,g(n)$

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

g(n) is upper bound of f(n).

5) Small ω notation (ω):

It describe lower bound on a function which is not tight.



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

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

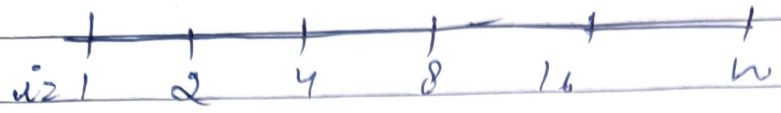
g(n) is the lower bound of f(n)

Ques: what should be time complexity of for (i=1 to n) { i = i * 2 }

```
sum = 0
for (i = 1; i <= n; i *= 2)
{
    sum += i;
}
```

Date __/__/____

Page _____



This is forming a G.P.

$$n = ar^{K-1}$$

where $a = 1$

$$r = \frac{4}{2}, \frac{2}{1}$$

$$r = 2$$

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

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

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

$$2n = 2K$$

taking log on both sides

$$\log_2(2n) = K \log_2 2$$

$$K = \frac{\log_2(2n)}{\log_2(2)}$$ — constant

$$K = \log_2(2n)$$

$$K = \log_2 n + \log_2 2$$

$$K = \log_2 n + 1 \quad (\because \log_2 2 = 1)$$

$$K = \log_2 n \quad (1 \text{ is constant})$$

Time complexity $= O(\log_2 n)$

Date __/__/____
Page _____

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

$T(0) = 1$

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

for $T(1)$

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

$= 3 \times 1$

for $T(2)$

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

$= 3T(1)$

$= 3T(0)$

$= 3 * 3 * 1$

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

$= 3T(2)$

$= 3 * 3 * 3 * 1$

for $T(n)$

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

$= 3 * 3 * 3 * 3$

$= 3^n$.

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

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

$T(0) = 1$

for $T = 1$.

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

$= 2T(0) - 1 = 2-1 = 1$.

Be Positive

Date / Page

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

$= 2T(1) - 1$

$= 2(1) - 1$

$= 2 - 1$

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

$= 2T(2) - 1$

$= 2(2-1) - 1$

$= 4 - 2 - 1$

⋮

for $T(n)$

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

$= 2n - (2n-2) - - - 4 - 2 - 1$

$$\boxed{T(n) = O(1)}$$

Ques 5

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

$S_i = S_{i-1} + i$

when $i = 1$, $S_1 = S_0 + i \Rightarrow S = 1$,

when $i = 2$, $S_2 = S_1 + 1 = 1 + 2 \Rightarrow S = 3$

Be Positive

Date __/__/____
Page _____

When $i=3$, $S_3 = S_2 + 3 = 3+3 = 6$

$\Rightarrow 1+3+6+10 \; - - - - - - \; K = n$

$\Rightarrow \frac{K(K+1)(K+2)}{6} = n$

$O(K^3) = n$

$K = \sqrt[3]{n}$

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

$\frac{K^2+K}{2} = n.$

$O(K^2) = n$

$K = \sqrt{n}$

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

Check $i*i <= n$.

$\because$ $i*i$ should be less than or equal to n

When $i=1$, $1*1 \le n \Rightarrow 1 \le n$

$i=2$, $2\times2 = 4 \le n \Rightarrow 4 \le n$

$i = n$, $\sqrt{n} \times \sqrt{n} = n \le n \Rightarrow$

Be Positive

Date __/__/__
Page ____

So, the loop will be –

1, 2, 3, – – – – $\sqrt{n}$

no. of iteration K is bound by $\sqrt{n}$

So time complexity is $O(\sqrt{n})$

Ques 7
```
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++
```

1. I iterates from $n/2$ to n. Its time complexity is $O(n)$.
2. J iterates from 1 to n with a double increment ($j = j * 2$).
Its, time complexity is $O(\log n)$
3. K iterates from 1 to n with a double increment ($K = K * 2$).
Its, time complexity is $O(\log n)$.

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

Ques 8
```
function (int n){
    if (n == 1) return.
    for (i = 1 to n){       (n times).  ] O(n^2)
    for (j = 1 to n){       (n times)   ]
        print ("*");
    }
}
function (n-3);   T(n-3) times
}
```

Date __/__/____
Page _____

The time complexity of both the inner loops is $O(n^2)$.

$T(n) = T(n-3) + O(n^2)$

as $T(1) = O(1)$

Thus, T.C $= O(n^2)$

Ques9 Time complexity of -

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

for $j := n/1 + n/2 + n/3 + - - - + n/n$

$n = 1 2^{k-1} \Rightarrow n = 2^k/2 \Rightarrow 2n = 2^k$

taking log both side

$\log 2n = \log 2^k$

T. C $= O(\log_2 n)$

Thus, the T.C is $O(n \log_2 n)$

Date __/__/____

Page _____

Ques10. For the functions, n^K and c^n, what is the asymptotic relationship between these function? Assume that K >= 1 and c > 1 are constant. Find the value of c and no for which relation holds.

n^k grows polynomially with n

c^n grows exponentially with n

thus $c^n = n^K$

so, $n^{\wedge K}$ is $O(c^{\wedge n})$

Find the value of c and no

$$\log n^k = \log c\,(c^n)$$

$$\Rightarrow \quad c \geq e \text{ and } n_0 = K$$