SAGAR NEGI SEC - H ROLL NO. - 10

Tutorial - I

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

Sol:- Asymptotic notations are used to describe the running time of an algorithm - how much time an algorithm takes with a given input, n.

(i) Big - oh (o) notation

→ for tight / strict upper bound.

Two functions: $f(n)$ and $g(n)$

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

iff $f(n) \leq c * g(n)$

$\forall n \geq n_0$ and $c > 0$

Example

Binary search - $o(\log n)$

Selection sort - $o(n^2)$

(ii) Big - omega (Ω) notation

→ for tight / strict lower bound

Two functions: $f(n)$ and $g(n)$

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

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

$\forall n \geq n_0$ and $c > 0$

~~Example~~

(iii) Theta (θ) notation

→ It gives tight / strict upper and lower bound both.

Two functions : $f(n)$ and $g(n)$

~~iff c_1~~ then $f(n) = \theta(g(n))$

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

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

and c_1 and $c_2 > 0$

Que 2. What should be time complexity of:

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

Sol:-

$i = 1, 2, 4, 8, 16, \ldots 2^k$

Here $n = 2^k$

taking log both sides

$\log_2 n = \log_2 2^k$

$\log_2 n = k \log_2 2$

$k = \log_2 n$

$\therefore$ Time complexity $= O(\log n)$

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

Sol:- $T(n) = 3T(n-1)$ —(i)

Putting $n = n-1$

$T(n-1) = 3T(n-2)$ —(ii)

Putting (ii) in (i), we get

$T(n) = 3[3T(n-2)]$

$T(n) = 3^2 T(n-2)$ —(iii)

Now putting $n = n-2$

$T(n-2) = 3T(n-3)$ —(iv)

Putting (iv) in (iii), we get

$T(n) = 3^2[3T(n-3)]$

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

Similarly for $n = n$

$3^n T(n-n)$

$3^n T(0)$

$3^n . 1$

3^n

$\therefore$ Time complexity $= O(3^n)$

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

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

for $n = 0$ $T(0) = 1$

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

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

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

$\vdots$

$T(n) = 1$

$\therefore$ Time complexity $= O(1)$.

Que 5. What should be time complexity of:

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

Solution: Initially,
$i=1, s=1$

After 1st iteration
$i=2, s=3$

After 2nd iteration
$i=3, s=6$

After 3rd iteration
$i=4, s=10$

So, here
$1+2+3+\cdots+x <= n$

$(x * (x+1))/2 <= n$

$O(x^2) <= n$

$x = O(\sqrt{n})$

$\therefore$ Time complexity $= O(\sqrt{n})$

Que 6. What should be the time complexity of:

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

Solution: Let 'k' be maximum positive value, such that

$k^2 < n$

$k = \sqrt{n}$

$i^2 \leq n$

$\therefore \sum_{i=1}^{k} 1 = 1+1+\dots k$ times

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

Que 7. What should be the time complexity of:

```
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 ++;
}
```

Solution: i loop → n times.

j loop → log n times,

k loop → log n times,

$\therefore$ Time complexity $= O(n * \log n * \log n)$

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

Que 8. Time complexity of

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

Sol:-

```
function (int n)
{
if (n==1) return;  → 1
for (i=1 to n)  → n  }
{                    } n²
for (j=1 to n)  → n  }
{printf("*");
 }
 }
function (n-3); → T(n-3)
```

$\therefore \boxed{T(n) = T(n-3) + n^2}$

$T(1) = 1$

$T(1) = 1$

$T(4) = T(4-3) + 4^2 = T(1) + 4^2 = 1^2 + 4^2$

$T(7) = T(7-3) + 7^2 = T(4) + 7^2 = 1^2 + 4^2 + 7^2$

So $T(n) = 1^2 + 4^2 + 7^2 + 10^2 + \cdots n^2 = \frac{n(n+1)(2n+1)}{6}$

$\Rightarrow \therefore T.C = O(n^3)$

Que 9. Time complexity of

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

Sol:-

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

$$\Rightarrow \frac{n-1}{1} + \frac{n-1}{2} + \frac{n-1}{3} + \cdots + \frac{n-1}{n-1} + 1$$

$$\frac{n}{1} + \frac{n}{2} + \frac{n}{3} + \cdots + \frac{n}{n-1} - \log(n-1)$$

$$\Rightarrow n\left\{ \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-1} \right\} - \log(n-1)$$

$$\Rightarrow n\log(n-1) - \log(n-1)$$

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

Que 10. for the functions, n^k and c^n, what is the asymptotic relationship between these functions? Assume that k>=1 and c>1 are constants. Find out the value of c and no for which relation holds.

Sol:- $f_1(n) = n^k$, $f_2(n) = c^n$

$k >= 1$, $c > 1$

Asymptotic relationship between f_1 and f_2 is Big O

i.e, $f_1(n) = O(f_2(0)) = O(c^n)$

$n^k \leq c_1 * c^n$ [c_1 is some constant]