TUTORIAL-1

ANSHIKA RATHI
CST-SPL 2
ROLL NO. - 10

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

Ans. Asymptotic notations are languages that allows us to analyze an algorithm's running time by identifying its behaviour as the input size for the algorithm increases.

There are three types of Asymptotic Notations -

(1) Big O

This notation decides an upper bound of an algorithm.
The function $f(n) = O(g(n))$ if and only if $f(n) <= c.g(n)$ for all $n >= n_0$, where c and n_0 are constants.
Here $g(n)$ is k/a upper bound on values of $f(n)$.
Eg: $f(n) = 3n+3$, $g(n) = 4n$.

(2) Big Ω

Ω notation provides an asymptotic lower bound.
The function $f(n) = \Omega(g(n))$ if and only if $f(n) >= c.g(n)$ for all $n >= n_0$ where c and n_0 are constants.
Here, $g(n)$ is k/a lower bound on values of $f(n)$.
Eg: $f(n) = 3n+2$ and $g(n) = 3n$.

(3) Big Θ

The theta notation bounds a function from above and below, so it defines exact asymptotic behaviour. Hence, it is also k/a tightly bound.
The function $f(n) = \Theta(g(n))$ if $c_1.g(n) <= f(n) <= c_2.g(n)$ for all $n >= n_0$, where c_1, c_2 and n_0 are constants.
Eg: $f(n) = 3n+2$, $g(n) = n$, $c_1 = 3$ and $c_2 = 4$

Que2. What should be time complexity of -

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

Ans- $O(n\log n)$

Que3. $T(n) = \{3T(n-1)$ if $n>0$, otherwise $1\}$

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

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

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

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

$\vdots$

$= 3^n T(n-n) = 3^n T(0)$

$= 3^n$

$\therefore$ complexity is $O(3^n)$.

Que4. $T(n) = \{2T(n-1) - 1$ if $n>0$, otherwise $1\}$

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

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

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

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

$= 2^3 T(n-3) - 2^2 - 2^1 - 2^0$

$\vdots$

$= 2^n T(n-n) - 2^{n-1} - 2^{n-2} - 2^{n-3} \dots 2^2 - 2^1 - 2^0$

$= 2^n - 2^{n-1} - 2^{n-2} - 2^{n-3} \dots 2^2 - 2^1 - 2^0$

$= 2^n - (2^n - 1)$ $\{\because 2^{n-1} + 2^{n-2} + \dots 2^0 = 2^n - 1\}$

$= 1$

$\therefore$ complexity is $O(1)$.

Ques5. What should be time complexity -

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

Ans. Let the loop execute x times.

Now, the loop will execute as long as s is less than n

After 1st iteration:

$s = s + 1$

After 2nd iteration:

$s = s + 1 + 2$

As it goes for x iterations,

$1 + 2 + \dots + x <= n$

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

$\Rightarrow O(x^2) <= n$

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

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

Ques6. Time complexity of -

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

Ans. let 'K' be max positive value, such that,

$k^2 \leq n$

$\therefore K = \sqrt{n}$

$i^2 \leq n$

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

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

Que 7. 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++
}
```

Ans. let 'm' be highest possible value such that

$m^2 \leq n \quad \therefore m = \sqrt{n}$

for	i	j	k
	$n/2$	1	1
	$n/2+1$	2	2
	$\vdots$	$\vdots$	$\vdots$
	n	n	n
	$\frac{n}{2}$ times	$\sqrt{n}$ times	$\sqrt{n}$ times

$\therefore \sum_{i=n/2}^{n} j * k$

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

$\Rightarrow T(n) = O(n^2)$

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);
}
```

Ans For :- for (i=1 to n),

we get j = n times

$\therefore i \times j = n^2$

Now, $T(n) = n^2 + T(n-3)$;

$T(n-3) = (n^2-3)^2 + T(n-6)$;

$T(n-6) = (n^2-6)^2 + T(n-9)$;

$\vdots$

$T(1) = 1$;

Now, substitute each value in $T(n)$

$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \dots + 1$

Let $(n-3k) = 1$

$\therefore k = (n-1)/3$

$\because$ Total time $= k+1$

$T(n) = n^2 + (n-3)^2 + (n-6)^2 + \dots 1$

$T(n) \approx n^2 + n^2 + n^2 \dots$ (k times + 1)

$T(n) \approx k n^2$

$T(n) \approx \left(\frac{n-1}{3}\right) \times n^2$

$\therefore T(n) = O(n^3)$

Que 9. Time complexity of -

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

Ans-

$i=1$ $\quad j = 1+2+ \dots (n \geq j+i)$

$i=2$ $\quad j = 1+3+5 \dots (n \geq j+i)$

$i=3$ $\quad j = 1+4+7+ \dots (n \geq j+i)$

$\vdots$ $\quad \vdots$

m^{th} term of AP is

$T(m) = a + d \times m$

$\Rightarrow T(m) = 1 + d \times m$

$\Rightarrow \frac{n-1}{d} = m$

i	j
$i=1$	$(n-1)/1$ times
$i=2$	$(n-1)/2$ times
$i=3$	$(n-1)/3$ times
$\vdots$	$\vdots$
$i=n-1$	1

we get,

$T(n) = i_1 j_1 + i_2 j_2 + \dots i_{n-1} j_{n-1}$

$= \frac{(n-1)}{1} + \frac{(n-2)}{2} + \frac{(n-3)}{3} + \dots + 1$

$= n + \frac{n}{2} + \frac{n}{3} + \dots + \frac{n}{n-1} - n \times 1 + 1$

$= n \left[1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n-1}\right] - n + 1$

$= n \times \log n - n + 1$

since $\int \frac{1}{x} = \log x$

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