TUTORIAL-1

Que1) What do you Understand by Asymptotic notations. Define different Asymptotic notations with example

Ans Asymptotic Notations

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

Types of Asymptotic Notations

i) Big Oh (O) Notation - It defines upper bond of an algorithm. It bounds the function only from above

$f(n) = O(g(n))$ where $g(n)$ is "tight" upper bond of $f(n)$

Example -: $O(1)$ represents the complexity of an algo that always execute in same time or space regardless of the input data.

ii) Small Oh (o) Notation - We denotes o notation to denote an upper bound that is not asymptotically tight.

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

$f(n) < g(n) \quad \forall \ n > n_0$

$c > 0$ there exists a constant

iii) Big Omega (Ω) Notation - It represents the lower bound running time complexity of an algorithm.

$f(n) = \Omega g(n)$ where $g(n)$ is "tight" lower bound of $f(n)$

iv) Small Omega (ω) Notation - It denotes a lower bound that is not asympotically tight.

$f(n) = \omega(g(n))$ where $f(n) > c \cdot g(n) \ \forall \ n > n_0$

$c > 0$

v) Theta (θ) - It bounds the function from above and below. So, it defines exact asymptotic behaviour

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

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

Que2) What should be time Complexity of -

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

Ans $O(\log n)$

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

By using back substitution

$T(n) = 3T(n-1)$ — ①

$T(n-1) = 3T((n-1)-1) = 3T(n-2)$ — ②

$T(n-2) = 3T(n-2-1) = 3T(n-3)$ — ③

Putting eq ③ in ②

$T(n-1) = 3(3T(n-3))$ — ④

Putting eq ④ in ①

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

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

Let $k = n$

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

Que4) $T_n = \begin{cases} 2T(n-1) - 1 & \text{if } n > 0 \\ 1 & n < 0 \end{cases}$

$T(n) = 2T(n-1) - 1$ — ①

Using back substitution

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

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

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

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

After k steps we have

$T(n) = 2^kT(n-k)-2^{k-1}-2^{k-2}-\ldots-2^2-2^1-2^0$ — ①

Considering $T(1) = 1$; let $n-k=1 \Rightarrow k=n-1$

Putting value of k in ①

$T(n) = 2^{n-1}T(1) - [2^0+2^1+2^2+\ldots+2^{n-3}+2^{n-2}]$

$= 2^{n-1} \times 1 - [2^{n-1}-1]$

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

$T(n) = 1$

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(" #");
}
```

Ans $O(n)$

Que 6) Time Complexity of -

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

Ans $O(\sqrt{n})$

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 $O(n \log n \log n)$

Que 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);
}
```

Ans 8) $O(n^2)$

Que9) Time Complexity of -

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

Ans $O(n^2)$

Que10) For the functions, n^k and a^n, what is the asymptotic relationship between these functions?

Assume that $K>=1$ & $a>1$ are constants. Find out the value of c and n_0 for which relation holds.

Ans10) $n^k = O(c^n)$