

## Tutorial - I (DAA)

Ans 1) Asymptotic Notation: Asymptotic Notation are the mathematical notations used to describe the running time of an algorithm.

Different forms of Asymptotic Notation:-

1) Big-O Notation (O):-

It represents upper bound of algorithm

$$f(n) = O(g(n)) \quad \text{if } f(n) \leq c * g(n)$$

2) Omega Notation ( $\Omega$ ):-

It represents lower bound of algorithm

$$f(n) = \Omega(g(n)) \quad \text{if } f(n) > c * g(n)$$

3) Theta Notation ( $\Theta$ ):-

It represents upper and lower bound of algorithm.

$$f(n) = \Theta(g(n)) \quad \text{if } c_1 g(n) \leq f(n) \leq c_2 g(n)$$

Ans 2) for ( $i = 1$  to  $n$ )

{  
   $i = i * 2$   
}

It is forming GP

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

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

$$n = 1 \times (2)^{n-1}$$

$i = 1$   
 $i = 2$   
 $i = 4$   
 $i = 8$   
 $i = 16$   
 $i = n$

$$\begin{pmatrix} a_n = n \\ r = 2 \\ a = 1 \end{pmatrix}$$

$$\log n = \log 2^{n-1}$$

$$\log n = (n-1) \log 2$$

$$\boxed{n = \log n + 1}$$

$$O(\log n)$$

Ans 3]

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

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

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

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

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

if  $n > 0$ , otherwise 1  
[ $T[0] = 1$ ]

$$T(n) = 3 \times 3 \times 3 \dots$$
$$= 3^n = O(3^n)$$

Ans 4]

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

$$T(0) = 1$$

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

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

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

$$= 2 - 1 = 1$$

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

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

$$T(n) = 1 \quad O(1)$$

Ans 5]

int  $i = 1$ ,  $S = 1$

while ( $S \leq n$ )

{  $i++$ ;

$S = S + i$ ;

Printg (" $\#$ ");

?



$i = 1$	$S = 1$
$i = 2$	$S = 1 + 2$
$i = 3$	$S = 1 + 2 + 3$
$i = 4$	$S = 1 + 2 + 3 + 4$

Loop ends when  $S > n$

$$1 + 2 + 3 + 4 \dots k > n$$

$$\frac{k(k+1)}{2} > n$$

$$k^2 > n$$

$$k > \sqrt{n}$$

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

Ans-6

void function (int n)

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

$i = 1$   
 $i = 2$   
 $i = 3$   
 $i = 4$

Loop ends when  $i * i > n$

$$k * k > n$$

$$k^2 > n$$

$$k = \sqrt{n}$$

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

$i = k$

Ans 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<sup>st</sup> Loop  $i = n/2$  to  $n$ ,  $i++$   
 $= O(n/2) = O(n)$

2<sup>nd</sup> nested Loop:

$j = 1$  to  $n$ ,  $j = j \times 2$   
 $j = 1$   
 $j = 2$   
 $j = 4$   
 $j = n$   
 $= O(\log n)$

3<sup>rd</sup> nested Loop :-

$k = 1$  to  $n$ ,  $k = k \times 2$   
 $k = 1$   
 $k = 2$   
 $k = 4$   
 $= O(\log n)$

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

Ans 6

Function (int n)

{ if (n == 1) return; — 1

for (int i = 1 to n)

{ for (int j = 1 to n) —  $n^2$

printf("\*");

}

? ?

function (n-3) —  $T(n-3)$

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

$$T(1) = 1$$



Ans-9

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

i = 1 — j = 1 to n  
i = 2 — j = 1 to n  
i = 3 — j = 1 to n  
i = 4 — j = 1 to n

So, for i upto n it will take  $n^2$   
So,  $T(n) = O(n^2)$ .

Ans 10

$$f_1(n) = n^k$$

$$f_2(n) = c^n$$

$$k \geq 1, c > 1$$

Asymptotic relationship between

is Big O i.e.  $f_1(n) = O(f_2(n)) = O(c^n)$

i.e.  $n^k \leq G \times c^n$  [G is some constant]