

Tutorial-1 (DAA)

Prapti Goel
Sec- I
Roll No. - 66

Answer 1) → Asymptotic Notation: Asymptotic Notation are the mathematical Notations used to describe the running time of an algorithm.

Different type of Asymptotic Notation:

1.) Big-O Notation (O): It represents upper Bound of algorithm.

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

2.) Omega Notation (Ω): It represents upper (Not) and lower Bound of Algorithm.

$$f(n) = \Omega(g(n)) \text{ if } f(n) \geq c * g(n).$$

3.) Theta Notation (Θ): It represents upper and lower Bound of Algorithm.

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

Answer 2) → for ($i = 1$ to n)
 $i = i * 2$
 \nearrow

It is forming $n!$

$$a_n = a * x^{n-1}$$

$$n = a * x^{k-1}$$

$$n = 1 * (2)^{k-1}$$

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

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

$$k = \log n + 1$$

$$i = 1$$

$$i = 2$$

$$i = 4$$

$$i = 8$$

$$i = 16$$

$$i = n$$

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

$$O(\log n).$$

Answer 3)

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

if $n > 0$, otherwise 1

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

$$[T(0) = 1]$$

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

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

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

⋮

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

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

Answer 4)

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

$$T(0) = 1$$

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

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

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

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

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

$$= 2 - 1 = 1$$

⋮

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

Answer 5)

```
int i = 1, s = 1
```

```
while (s <= n)
```

```
{
```

```
    i++;
```

```
    s = s + i;
```

```
    printf("#");
```

```
}
```

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

Loop ends when $S > n$

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

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

$$k^2 > n$$

$$k > \sqrt{n}$$

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

Answer 6) → void function (int n)

┌ int i, count = 0;

for (int i = 1; i * i ≤ n; i++)

count++;

└

Loop ends when $i * i > n$

$$k \times k > n$$

$$k^2 > n$$

$$k > \sqrt{n}$$

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

Answer 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 ($k=1$; $k \leq n$; $k = k * 2$)

Count++;

↳

• 1st Loop: $i = \frac{n}{2}$ to n , $i++$
 $= O\left(\frac{n}{2}\right) = O(n)$

• 2nd Nested Loop: $j = 1$ to n , $j = j * 2$
 $j = 1$
 $j = 2$
 $j = 4$
 $j = n$
 $= O(\log n)$

• 3rd Nested Loop: $k = 1$ to n , $k = k * 2$
 $k = 1$
 $k = 2$
 $k = 4$
 $= O(\log n)$

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

Answer 8) → Function (int n)

↳ If ($n == 1$) return - 1

for (int $i = 1$ to n)

↳ for (int $i = 1$ to n) — n^2

↳ printf ("%x");

↳

↳

↳ function ($n-3$) — $T(n-3)$

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

$T(1) = 1.$

$$\rightarrow T(1) = 1$$

$$\begin{aligned} \rightarrow T(4) &= T(4-3) + 4^2 \\ &= T(1) + 4^2 = 1^2 + 4^2 \end{aligned}$$

$$\begin{aligned} \rightarrow T(7) &= T(7-3) + 7^2 \\ &= 1^2 + 4^2 + 7^2 \end{aligned}$$

$$\begin{aligned} \rightarrow T(10) &= T(10-3) + 10^2 \\ &= 1^2 + 4^2 + 7^2 + 10^2 \end{aligned}$$

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

$$\text{also for terms like } T(2), T(3), T(5) = O(n^3)$$

$$\text{So, } T(n) = O(n^3)$$

Answer 9)

void function (int n)

└ for (int i=1 to n) - n

└ for (j=1; j <= n; j=j+1) - n

└ printf("*");

i=1 — j=1 to n.

i=2 — j=1 to n

i=3 — i=1 to n

i=4 — i=1 to n.

So, for i upto n it will take

$$n^2$$

$$\boxed{\text{So, } T(n) = O(n^2)}$$

Answer 10) $f_1(n) = n^k$

$f_2(n) = C^n$

$k \geq 1, C > 1$

Asymptotic relationship between f_1 and f_2 .

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

is $n^k \leq G \times C^n$ (G is some constant).