AMAN PANDEY
Sec-H
Rollno-25
Univ. Rollno-2016615

Tutorial-1 DAA

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

Types:

1) Big-O Notation(O): It represent upper Bound of algorithm.
   $f(n) = O(g(n))$ if $f(n) \leq c * g(n)$

2) Omega Notation(Ω): It represents lower bound of Algorithm.
   $f(n) = \Omega(g(n))$. if $f(n) \geq c * g(n)$.

3) Theta Notation(Θ): It represent upper and lower bound of algorithm.
   $f(n) = \Theta(g(n))$ if $c_1 g(n) \leq f(n) \leq c_2 g(n)$.

Ans-2)

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

i = 1
i = 2
i = 4
i = 8
⋮
i = n

It is forming a GP:

$a_n = ar^{n-1}$

$a_n = n$
$r = 2$
$a = 1$

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

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

$k = \log n + 1$ $\quad O(\log(n))$

Ans-3:

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

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

$T(1) = 3 \times 1$

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

⋮

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

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

Ans-4: $T(n) = 2T(n-1) - 1$ if $n>0$, otherwise 1

$T(0) = 1$

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

$= 2 - 1 = 1$

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

$= 1$

⋮

$T(n) = 1$

$O(1)$

Ans-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

⋮

Loop ends when s>n

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

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

$k^2 > n$

$k > \sqrt{n}$

$T(c) \Rightarrow O(\sqrt{n})$

Ans-6

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

Loop ends when $i*i > n$

$k \times k > n$

$k^2 > n$

$k > \sqrt{n}$

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

Ans-7.

1st Loop $i = n/2$ to n, $i++$

$O(n/2) = O(n)$

2nd nested Loop: $j=1$ to n, $j = j*2$

$j=1$

$j=2$

$j=3$

$j=4$

$j=n$

$\Rightarrow O(\log n)$

3rd nested loop $k=1$ to n

$k=1$

$k=2$

$k=4$

$O(\log n)$

Time Complexity $T(c) = O(n \log^2 n)$

Ans-8

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

$T(1) = 1$

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

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

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

$$T(10) = T(10-3) + 10^2$$
$$= 1^2 + 4^2 + 7^2 + 10^2$$

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

$$= O(n^3)$$

$$T(c) = O(n^3).$$

Ans-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 — j=1 to n

so, for i upto n it will take n^2

So, $T(n) = O(n^2)$.

Ans-10)

$f_1(n) = n^K$, $f_2(n) = c^n$

Asymptotic relationship between f1 and f2 is Big O. i.e

$f_1(n) = O((f_2(n)) = O(c^n)$

is $n^K \leq G * c^n$ [G is some constant]