Name :- Vishwajeet Patel
Section :- AI & DS
Roll no :- 53
University Roll no :- 2017683

Subject " DAA "

Tutorial - 1 (DAA)

Q1) Solution :- <u>Asymptotic Notation</u> : Asymptotic Notation are the mathematical notations used to describe the running time of an algorithm.

Different types of Asymptotic Notations :

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

2. <u>Omega Notation</u> (Ω) : It represents lower bound of algorithm.
   $f(n) = \Omega(g(n))$ if $f(n) \geq c * g(n)$.

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

Q 2) Solution :-

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

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

It is forming G.P

$a_n = ar^{n-1}$

$n = ar^{k-1}$

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

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

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

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

$\left(\begin{matrix} a_n = n \\ r = 2 \\ a = 1 \end{matrix} \right)$

$O(\log n)$

Q3 Solution :- $T(n) = 3\,T(n-1)$ if $n>0$, otherwise 1

$T(1) = 3\,T(0)$ $[T(0) = 1]$

$T(1) = 3 \times 1$

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

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

$\vdots$

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

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

Q4 Solution :- $T(n) = 2T(n-1) - 1$ if $n>0$, otherwise 1

$T(0) = 1$

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

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

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

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

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

$= 2 - 1 = 1$

$\vdots$

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

Q5 Solution :-

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

$i=1$	$s=1$
$i=2$	$s=1+2$
$i=3$	$s=1+2+3$
$i=4$	$s=1+2+3+4$
$\vdots$	$\vdots$
Loop Ends	when

$s > n$

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

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

$k^2 > n$

$k > \sqrt{n}$

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

Q6> Solution :- Void function (int n)

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

$i = 1$
$i = 2$
$i = 3$
$i = 4$
⋮
$i = K$

Loop Ends when $i * i > n$

$K * K > n$

$K^2 > n$

$K > \sqrt{n}$

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

Q7> Solution :- Void function (int n)

```
{ int i,j,K, Count = 0;
  for ( i=n/2 ; k = n ; i++)
    for ( j=1 ; j<=n ; j=j*2)
      for ( k=1 ; k<=n ; k = k*2)
          Count ++;
}
```

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

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

2nd Nested loop : $j=1$ to n $j = j*2$

$j=1$
$j=2$
$j=4$
⋮
$j=n$

$= O(\log n)$

3rd loop : $K = 1$ to n, $K = K*2$

$K=1$
$K=2$
$K=4$

$= O(\log n)$

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

Q8> Solution :- fun (int n)

```
{ if (n==1) return;            — 1
  for (int i=1; to n)
    for (int j=1 ton)          — n2
      Print ("*");

  fun (n-3)                    — T(n-3)
}
```

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

→ $T(4) = T(4-3) * 4^2 = T(1) + (4\times4) = 1^2 + 4^2 =$

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

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

So, $T(n) = 1^2 + 4^2 + 7^2 + 10^2 \cdots n^2 = \frac{n(n+1)(2n+1)}{6} = O(n^3)$

also for terms Like $T(2)$, $T(3)$, $T(5)$

So, $T(n) = O(n^3)$

Q9) Solution :- Void function (int n)

{ for (int i = 1 to n) — n

for (j = 1 ; j <= n ; j = j+1) — n

Print ("*") ;

}

$i = 1 \rightarrow j = 1$ to n
$i = 2 \rightarrow j = 1$ to n
$i = 3 \rightarrow j = 1$ to n
$i = 4 \rightarrow j = 1$ to n

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

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

Q10) Solution :- $f_1(n) = n^k$ $f_2(n) = c^n$

$k >= 1$, $c > 1$

Asymptotic relationship between f_1 & f_2 is Big O i.e $f_1(n) = O(f_2(n)) = O(c^n)$

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