Day 2 :- DSA

Time Complexity :- Big O Notation (Worst Case)

1) Linear Equation

$y = mx + c$

```
int a = 10;                          1sec ──→ 1 sec ✓
for(int i=0; i<n; i++)               1sec     2sec × n ✓
{                                    ↓  ↑
    a++;  ──→ 1sec ✓                 └──┘
}
```

$2n \times 1$

$1 + 2n$

display ⇒ a

n = 1 ;	3 sec
n = 10 ;	21
n = 100 ;	201
n = 1000 ;	2001

Time Complexity = $2n+1$

$\simeq n$

$O(n)$

Most dominant factor = n

```
for( int i=1 ; i<n ; i=i+2)
{
    a++;
}
```

n = 10

i = 1, 3, 5, 7, 9 ≠ 11

$\frac{n}{2} = 10$

iteration = 5 times

Time Comp = $n/2$

$\cong n$

$O(n)$

H.W.

```
for( int i=1 ; i<n/2 ; i++)
{
    a++;
}
```

i = 1, 2, 3, 4, 5

$\frac{n}{2} = \frac{10}{2} = 5$

Time Complexity = $\frac{n}{2}$

$\cong n$

$O(n)$

---

2) Quadratic Eq^n :-

ax^2+bx+c

Independent Nested Loop

```
int a=10, b=20;                          → 2sec
for(int i=0 ; i<n ; i++)                 (2n+1)   n=5
                 1sec   1sec
{
    a++;                                 → 1sec
    for(int j=0 ; j<n ; j++)             2n
                     1sec   1sec
    {
        b++;                             1sec
```

$(2n+1) \times 2n$

$= 4n^2+2n+0$

$\cong 4n^2$

$\cong n^2$

```
for( 0 → n)
{
    for(0 → n)
    {
    }
}
```

$= n \times n = n^2$

$O(n^2)$

$i=0$; $j = 0, 1, 2, \ldots n$

$i=1$; $j = 0, 1, 2, \ldots n$

String s = "Hello"

```
for(int i=1 ; i<n ; i++)
{
    for( int j=1 ; j<n ; i++)
    {
        S.o.p( s);
    }
}
```

$n = 10$

$n \times n = n^2$

$i=1$ $j=1$ to n $= 10$

$i=2$, $j=1$ to n $= 10$

$i=3$, $j=1$ to n $= 10$

$i=n$; $j=1$ to n $= 10$

$10 \times 10 = 100$

*

```
for( int i=1 ; i< n/2 ; i++)
{
    for( int j=1 ; j<n ; j=j+3)
    {
    }
}
```

$\frac{n}{2} \times \frac{n}{3}$

$n=1$

$= \frac{n^2}{6}$

$\cong n^2$

$O(n^2)$

* `for (int i=1 ; i<=n ; i++)` → Dependant Nested Loop

```
{
    for( int j= 0; j<i ; j++)
    {
        a++;
    }
}
```

i = 1	i = 2	i = 3	-------	i = n
↓	↓	↓		↓
j = 0 to 0	j = 0, 1	j = 0, 1, 2		j = 0, 1, ..n-1
↓	⇓	⇓		⇓
1 times	2 times	3 times		n times

$$1 + 2 + 3 + \text{---------} + n = AP = \frac{n(n+1)}{2}$$

$$\text{Time Complexity} = \frac{n(n+1)}{2} = \frac{n^2+n}{2} = \frac{n^2}{2} + \frac{n}{2}$$

$$= \frac{n^2}{2} = n^2$$

$$O(n^2)$$

```
* for(int i=1 ; i<=n ; i++)
  {
    for(int j=i ; j<=n ; j++)
    {
    }
  }
```

i=1	i=2	i=3	-----	i=n
↓	↓	↓		↓
j=1 to n	j=2 to n	j=3 to n		j=n to n
↓	↓	↓		↓
n times	n-1 times	n-2 times		1 times

Time Complexity $= n+(n-1)+(n-2)+---+3+2+1$

$= A.P.$

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

$\cong \frac{n^2}{2}$

$\cong n^2$

$O(n^2)$

```
for(int i=1 ; i<=n ; i++)      } n
{
  a++;
}
for(int j=1 ; j<=n ; j++)      } n
{
  b++;
}
```

Time Complexity

$= n + n$

$= 2n$

$= O(n)$

Qubic Equation :-

```
* for(int i=1 ; i<=n/2 ; i++)                    --> n/2
  {
    for(int j=1 ; j<=n ; j=j+3)                  --> n/2 x n/3
    {
      for(int k=1 ; k<=n ; k++)                  --> (n/2 x n/3) x n
      {
      }
    }
  }
```

$= \frac{n^3}{6}$

$\approx n^3$

$O(n^3)$

* Logarithmic Eqn :-

```
for(int i=1 ; i<=n ; i=i*2)
{
    S.o.p("Hitesh");
}
```

	1st	2nd	3rd	4th		
i =	1	2	4	8	----	n
	↑	↑	↑	↑		↑
	2^0	2^1	2^2	2^3	---	2^k

*** Whenever cond variable increment by any multiplication constant ao then time complexity goes to log Form.

Time Complexity

$$2^k = n$$

$$\log(2^k) = \log(n)$$

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

$$k = \frac{\log(n)}{\log(2)} = \log_2(n)$$

$$\text{Time Complexity} = \log_2(n) = O(\log_2 n)$$

```
for(int i=n ; i>=1 ; i=i*1/2)
{
    S.o.p("Hello");
}
```

i =	n					1
	$\frac{n}{1}$	$\frac{n}{2}$	$\frac{n}{4}$	$\frac{n}{8}$	---	1
	$\frac{n}{2^0}$	$\frac{n}{2^1}$	$\frac{n}{2^2}$	$\frac{n}{2^3}$	---	$\frac{n}{2k}$

Time Complexity = $\frac{n}{2^k} = 1$

$n = 2^k$

$\log n = \log 2^k$

$k \log 2 = \log n$

$k = \frac{\log n}{\log 2}$

$k = \log_2 n$

$O(\log_2 n)$

* `for(int i=n; i>=1; i*=1/3)` (i = i × 1/3)

```
{

}
```

$i = n \quad \frac{n}{3} \quad \frac{n}{9} \quad \frac{n}{27} \quad \dots \quad 1$

$\frac{n}{3^0} \quad \frac{n}{3^1} \quad \frac{n}{3^2} \quad \dots \quad \frac{n}{3^k}$

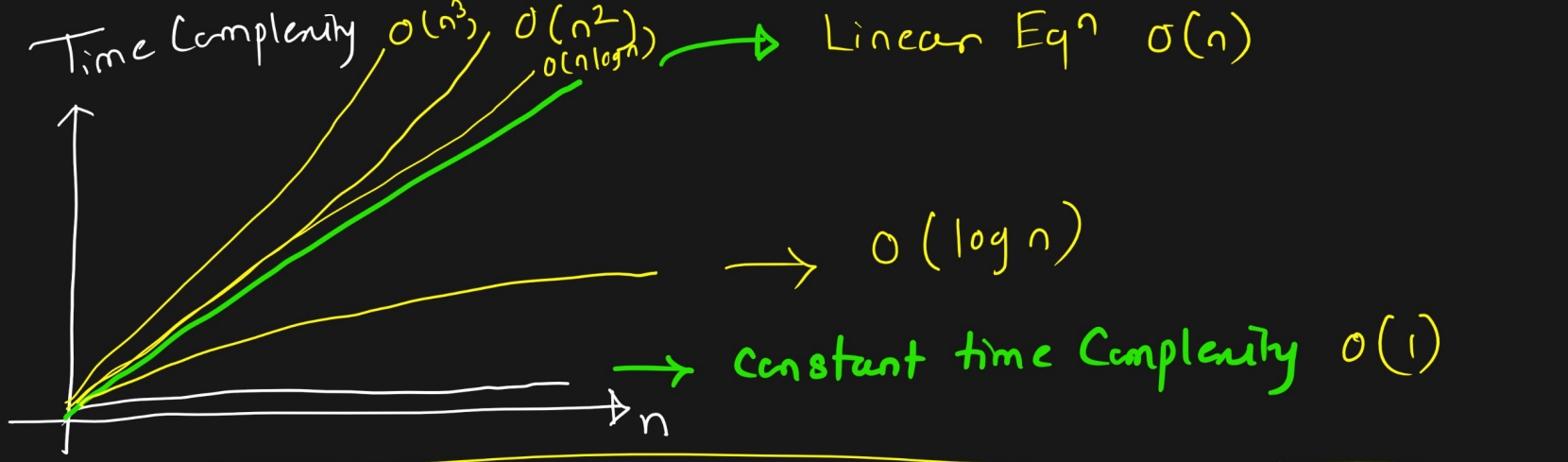
$\frac{n}{3^k} = 1$

Linear Logarithmic Eqn

Task

```
for(int i=0; i<n; i++)          ⟶ n
{
  for(int j=1; j<=n; j=j*4)     ⇒ n × log4 n
  {
  }
}
```

$O(n \log n)$



$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3)$$

* **Function** :- Block of statements to perform the particular task / operation

JAVA

```
public class Main
{
    p.s.v.m (String args[])
    {
        S.o.p("___");
    }
}
```

Python

```
def display ( ):
    print ("Hello")

display()
```

* Factorial of given No:-

num = 5

5! = 5×4×3×2×1 = 120

```
n!      fact = 1
for(int i=1 ; i<=n ; i++)
{
    fact = fact * i;
}
return fact;
```

Time Comp

O(n)

JAVA

```
double factorial (int n)
{
  [ if(num == 1)
      return 1; ]   <- base cond^n
  return (n * factorial(n-1));
}
```

1 2 3 4 5

num = 5 = 120

= 5 × fact(4) = 24

= 4 × fact(3) = 6

= 3 × fact(2) = 2

= 2 × fact(1) = 2×1

fact(1) = 1

Recursive Fun^n

Time Complexity :- O(n) → n times recursive fun^n called

```
int fact = 1;   ⇒ 4 byte
for(int i=1; i<n; i++)
{
    fact = fact * i;
}
return fact;
```

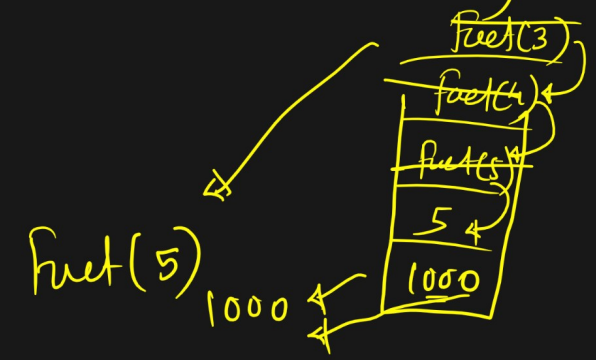
Time Comp :-

$O(n)$

Space Complexity :-

```
int fact (int num)
{
    if(num == 1)
        return 1;
    else
        return num * fact(num-1);
}
```

$O(n)$



H.W.

1)

```
for(int i=0; i*i<n; i++)
{
    display();
}
```

2)

```
for(int i=1; i<=n; i++)
{
    for(int j=1; j*j<=n; j++)
    {
    }
}
```

3)
```
int i=1
while(i<n)
{
    i*=3
    print(i)
}
```

④
```
while(n>0)
{
    n/=2;
    print n
}
```

Leetcode Pbm No:-

1) 509
2) 231
3) 342