## Time Complexity :→ Time Complexity of an algorithm quantifies the amount of time taken by a program to run as a function of length of the input.

Eg

```
int n;
cin >> n;
int a = 0
for (int i = 1; i <= n; i++)
{
    a++;
}
```

Here, in this code, if value of n = 5; then loop will run 5 times.

Hence, we can say that Time Complexity of the given program is linearly proportional to n.

```
int n;
cin >> n;
int a = 0;
for (int i = 1; i <= n; i++)
{
  for (int j = 1; j <= n; j++)
  {
    a++;
  }
}
```

Here, as there are two for loops. Therefore,

i = 1 → j = 1 to n
i = 2 → j = 1 to n
⋮
i = n → j to n.

Hence, time complexity ∝ $n^2$.

## Space Complexity :→

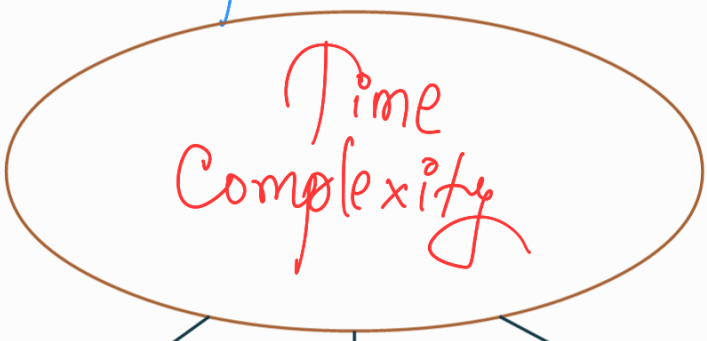
Space Complexity of an algorithm quantifies the amount of time taken by a program to run as a function of length of the input. It is directly proportional to the largest memory your program acquires at any instance during run time.

Eg

```
int n;          → 4 bytes
cin >> n;
int a = 0;      → 4 bytes
for (int i = 1; i < n; i++)   → 4 bytes
{
    a = a + 1;
}
```

Here, Space Complexity of the above code = 12 bytes (irrespective the no. of times the loop is running)

# Types and Representation of Time Complexities




- Worst Case - [O (big Oh) Notation]
- Best Case [Ω (big omega notation)]
- Average Case [Θ (big theta) notation]

```
int n, m;
cin>>n>>m;
for(int i=1; i<=n; i++) → n
  {
    for(int j=1; i<=m; i++) → m
      {
        a = a + rand();
      }
  }
for(int k=1; k<=n; k++) → n
  {
    a = a + rand();
  }
```

Time Complexity : $O(n+n+m)$

```
int n;
cin>>n;
int a=0, i=n; → n
while(i>=1)
  {
    a = a+1;
    i /= 2;
  }
```

Explaination :→

$n \rightarrow \frac{n}{2} \rightarrow \frac{n}{4} \ldots$

$\therefore \frac{n}{2^k} \geq 1$

$\Rightarrow n \geq 2^k \Rightarrow \log n \geq \log 2^k$

$\Rightarrow \log n \geq k$

Time Complexity : $O(\log n)$

☺ Dreams ON ☺

HAPPY LEARNING