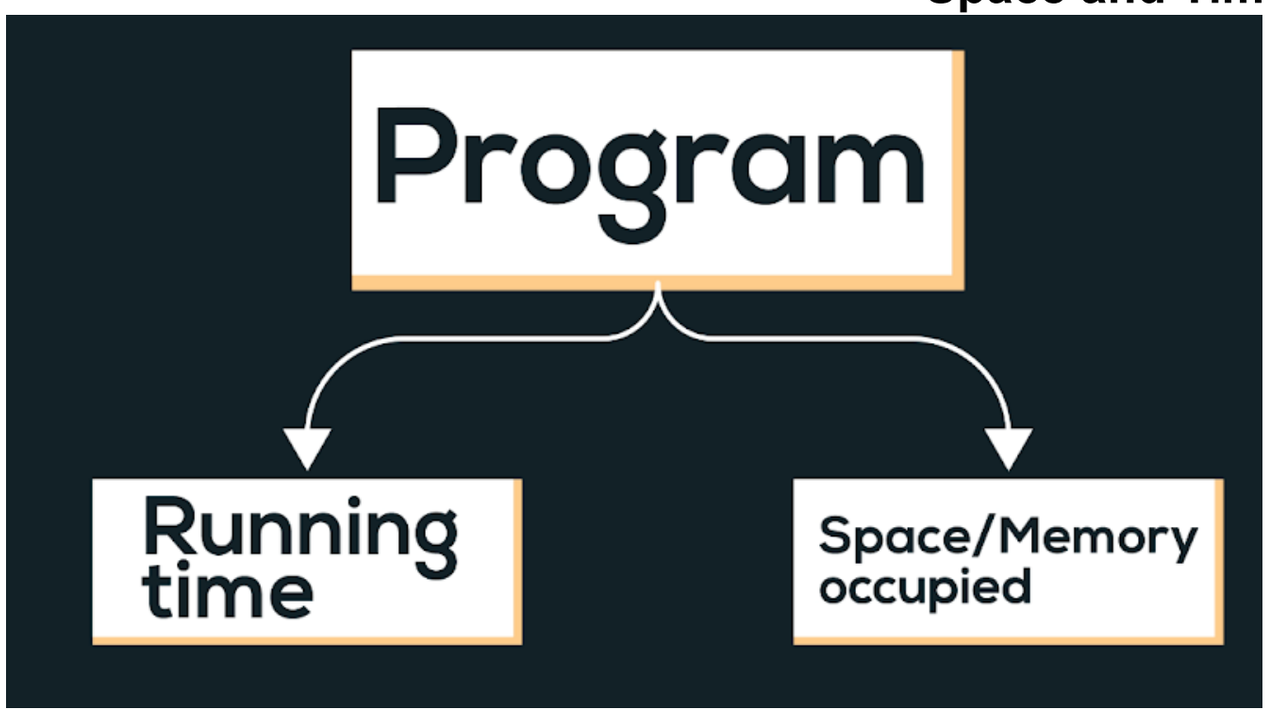
Space and Time Complexity



Time Complexity

Time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the input.

Types of notations

1. O-notation: It is used to denote asymptotic upper bound. For a given function g(n), we denote it by O(g(n)). Pronounced as “big-oh of g of n”. It also known as worst case time complexity as it denotes the upper bound in which algorithm terminates.
2. Ω-notation: It is used to denote asymptotic lower bound. For a given function g(n), we denote it by Ω(g(n)). Pronounced as “big-omega of g of n”. It also known as best case time complexity as it denotes the lower bound in which algorithm terminates.
3. Θ-notation: It is used to denote the average time of a program.

Examples:

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

Linear Time Complexity. O(n)

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

Quadratic time Complexity. $O(n^2)$

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

Apni Kaksha

```
{
    a = a + 1;
}

for (int j = 1; j <= m; j++)
{
    a = a + 1;
}
```

Time Complexity: O(n+m)

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

Time complexity: O(n*m)

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

Time complexity: O(log(n))

Comparison of functions on the basis of time complexity

It follows the following order in case of time complexity:

$O(n^n) > O(n!) > O(n^3) > O(n^2) > O(n.log(n)) > O(n.log(log(n))) > O(n) > O(sqrt(n)) > O(log(n)) > O(1)$

Note: Reverse is the order for better performance of a code with corresponding time complexity, i.e. a program with less time complexity is more efficient.

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.
For example: *int* consumes 4 bytes of memory.