

# Foundation Session - 1

## Agenda

1. Asymptotic Analysis
2. Rules to follow
3. Big-O
4. Big-Theta
5. Big-Omega
6. Best-Case v/s Average-Case v/s Worst-Case Complexity
7. Auxiliary Space v/s Space Complexity
8. String Sorting Analysis
9. Recursive Runtimes

2 rules:

1. Drop-off the constants
2. Drop-off the irrelevant terms

```
For (i =0; i < n; i++) {  
    For (int j = 0; j < m; j++) {  
        Cout << i + j << endl;  
    }  
}
```

TC =>

Aux Space =>

SC =>

```
arr => matrix of strings
For (i =0; i < n; i++) {
    For (int j = 0; j < m; j++) {
        Cout << arr[i][j] << endl;
    }
}
```

TC =>

Aux Space =>

SC =>

Printing an Integer =>  $O(1)$

32-bit integer => [0, 0, 1, 1 .....]

Printing a String =>  $O(\text{length of String})$

Question:

You have an array of Strings.

Array size = n

Length of all the strings = m

TODO: Sort each of the strings in the array individually and then sort the complete array.

Input = ["abd", "dfg", "bca", "ddb"]

Step-1: ["abc", "dfg", "abc", "bdd"]

Step-2: ["abc", "abc", "bdd", "dfg"]

Prerequisites:

- Sorting an array of integer  $\Rightarrow O(n \cdot \log(n))$
- Comparing two strings  $\Rightarrow O(m)$

Recursive Analysis:

```
Int fib(int n) {  
    If (n <= 1) return n;  
    Return fib(n-1) + fib(n-2);  
}
```

TC:

AS:

```
Int fun(int n) {  
    If (n <= 1) return n;  
    For (int i = 0; i < n; i++) { ... do something O(1) ...}  
    Return fun(n-1) + fun(n-2);  
}
```

TC:

AS:

```
Int fun(int n, arr[n]) {  
    If (n <= 1) return n;  
    ... do something with the array ...  
    For (int i = 0; i < n; i++) { ... do something O(1) ...}  
    Return fun(n-1) + fun(n-2);  
}
```

If array is passed by value  $\rightarrow$  AS =

If array is passed by ref  $\rightarrow$  AS =