***TIME COMPLEXITY***

**What is Time Complexity?**

* 1.Time Complexity is the amount of time taken by the algorithm to execute.
* It doesn’t calculate the exact amount of time
* It explains how our algorithm performs based on input variations.
* Here, We don’t use any time metric such as minutes, seconds, milliseconds etc.. to calculate time complexity.
* We use a mathematical function / representation to calculate time complexity with respect to input size.
* It is a graph of input size versus time taken by the algorithm.

**Why Time Complexity?**

* To decide the efficiency of algorithm.
* To make any betterments in the logic.
* Scaling the application.

**Why don’t we calculate time directly?**

1. Because systems are different

Eg : *An 8 GB Ram , i9 processor works faster than the 4 GB Ram,i3 processor.*

1. Effected by internal processes that are learning

Eg : *If there are two systems with same configurations. In one system there are 500 tasks running already. In the second system there are no tasks running. So, if we run our program in the first system and second system at the same time, the first system may take some more extra time to execute our program compared to the second system.*

1. Load on the system.
2. Execution time is different in many programming languages.

Eg : *C++ is faster than Java*

*Java is faster than Python*

**Different Cases:**

1. **Best Case:** What is the input that takes very less time to execute
2. **Worst Case:** What is the input that takes maximum time to execute
3. **Average Case:** Typical behaviour. Anything besides best or worst case.

**Different types of representation:**

1. Big O Notation (Worst Case Time Complexity)
2. Omega Notation (Best Case Time Complexity)
3. Theta Notation

Eg: printArrayValues(int[] arr){

for(int i = 0 ; I < arr.length ; i++){

System.out.println(arr[i]);

}

}

🡺if array has 1 element , the for loop will get executed only once , so the time complexity is 0(1).

This is the best case time complexity

🡺If the array has more elements , for example ‘N’ elements, the loop will be executed n times , so the time complexity is 0(‘N’).

This is the worst case time complexity

**Rules for time Complexity Calculation:**

1. **Sum the time consumed by each line of code.**

Eg: *func(){*

*Line-1; 🡪 time complexity of line 1 = 1*

*Line-2; 🡪 time complexity of line 2 = 2*

*Line-3; 🡪 time complexity of line 3 = n*

*}*

*Overall Time Complexity = 1 + 2 + n = 3 + n*

1. **Declaration, Assignments, Comparision take constant amount of time i.e. O(1)**

Eg: *int a;*

*a = 20;*

*if(a > b)*

*All these operations take O(1) time complexity*

1. **Internal function calls should be considered and calculated separately**

Eg :*int[] arr = {27,36,…..,2051}; 🡪 1*

*int answer; 🡪 1*

*answer = binarySearch(arr); 🡪 log(N)*

*🡪Time complexity = 1 + 1 + log(N) = 2 + log(N)*

1. **Iterative statements take time that is equal to number of times they are running.**

Eg:*for(int i = 0; i < n ; i++){  
 ------------------------------------------*

*}*

*Time complexity of this loop is O(n)*

Eg: *for(int i =0 ; i < m; i++){*

*-------------------------------------*

*}*

*Time complexity of this loop is O(m)*

1. **Neglect the lower order terms.**

Eg: *If overall time complexity is 7N^3 + 3N^2 + N + 7*

*i)Neglect 3N^2 , N , 7*

*🡪Time complexity = 7N^3*

*ii)Neglect Constant*

*🡪Time complexity = O(N^3)*