**DSA 90 Days: Time and Space Complexity**

**Time Complexity:**

Time complexity is the time take by an algorithm which quantifies the amount taken by a program to run based on the input size provided.

**Now,**

**Time complexity can be categorized under 3 section**

**Time Complexity**

**Running time**

**Linearly**

**Quadratic**

**Cubic**

Examples : Text

Description automatically generated

By this we can see this is linearly proportional to n.

Text

Description automatically generated

In first loop i will iterate from i = 1 and enter the next loop and print j =1 till j < n.

**Notations**

1. Worst Case: Big on (O)
2. Best Case: Big Omega (Ω)
3. Average Case: Big Theta(Θ)

**Example:**

Given int :

20 10 5 100 300 17 238

To find: x = 238 and x = 20

**Soln:-** if we have to find 238 in the given array, we will iterate through every array and compare whether it is equal to 238 and then we will conclude it is present at last. It’s **the worst time complexity** because, to find 238 will take more time as it is present at last.

If we have to find 20 in the array we will compare it with the first element and check whether it is equal to 20. Wholaa ! 20 is present at 1st place. It’s the **Best Case time complexity** because we have found It in the beginning, so we don’t have to iterate through every element.

A1 A2 A3 A4 A5…… An

**Worst case** X= iterate through each element and it will n.

**Best Case** X = element found at 1st place. i.e: 1, So it will be **constant** .

**Average Case** X = The Average is said to be total number of Time/ total no of case:

**Formula for Avg Case: N(n+1) / 2**

**N(n+1)/2 =**  this is because be we have to iterate through each element and shift to next and then iterate through next

Eg: 1, 2(it will start with 1 again),3((it will start with 1 again),,,,,,n

So, n(n+1)/2

**So if anyone says, find the time complexity of so and so algorithm we will find the Big O notation.** Because it will be good if we find the worst case of the algorithm.

Big O notation will find the Upper bound of the algorithm.

F(n) = 3**n2** + 5n

Here we will only find the highest power and ignore rest of the variables.

**Time Complexity = O(n2)**

**Wrap Up Time Complexity**

1. **O(n)**

This Scenario is encountered when the time take depends on the n number of inputs.

1. **O(1)**

This Scenario is encountered when n have the constant value or when it doesn’t depend on n.

Example :

int n = 30; // n is constant

for(int i = 0; i<n; i++)

{

Cout<<”Hello”

}

Or in case of simple hello world program.

1. **O(n2)**

The Scenario occurs when an program increases non-linearly with the length of the input.

**This is mostly found in Nested loops.**

Example:

Int main(){

Int n, m;

Cin>>n>>m;

For(int I =0; I <n; i++)

For (int j =0; j<m; j++)

Cout <<”Hello”

}

1. **O(log2n) or O(logn)**

This Scenario occurs when algorithm decrease the magnitude of the input data in each step, it is said to have a logarithimic time complexity.

**Any algorithm uses Divide and Conquer algorithms, such as binary search, or quicksort, in which partition of the array into two sections and finds a pivot element in O(n) time each. As a result, it is O(log2n)**

**A screenshot of a computer

Description automatically generated with medium confidence**

1. **O (n log n)**

The time complexity is popularly known as linearithmic time complexity. It performs slightly slower as compared to linear time complex but is still significantly better than quadratic algos.

**Algo which follows is Merge sort algorithm. This is probably the best time complexity that can be achieved**

**Space Complexity:**

The Term Space Complexity is often being misunderstood for Auxillary Space at many places.

**Definition:** Space Complexity of an algorithm is the total space by the algorithm with respect to input size. It includes both **Auxillary Space** and **space used by input**

Space complexity is a parallel concept to time complexity. If we need to create an array of size n, this will require O(n) space. If we create a two-dimensional array of size n\*n, this will require O(n2) space.

Text

Description automatically generated