MyAnatomy

Time and Space Complexities

What is Time and Space Complexity in Data Structure?

Time complexity is the amount of time taken to run an algorithm. It is the measure of the number of elementary operations performed by an algorithm and an estimate of the time required for that operation. It also depends on external factors such as the compiler, the processor's speed, etc.

If we ask you how much time you can add the first five natural numbers? (you start counting 1+2+3+4+5). Assume that, it took you 3 seconds. But how will you calculate this for the computer? We cannot! And thus, computer scientists have come up with an approach to calculate a rough estimate of the time taken to execute a code, called Time Complexity.

Space complexity is the amount of memory space an algorithm/program uses during its entire execution. It measures the number of variables created to store values, including both the inputs and the outputs.

In simple terms, it is a rough estimation of how much storage your code will take in RAM.

Anyone dreaming of getting into a product-based industry should not just be able to write code but write efficient code which takes the least time and memory to execute. So, let's begin to establish a solid foundation for this concept.

How to calculate Time complexity?

Frequency count method:

Let's look at the algorithms to find the sum of n numbers. There are two ways to get the results.

Code 1:- Frequency in front of code 1, 1, 1, (n+1), n, 1

Sum = 5+2n

Code 2:- Frequency 1,1,1,1,1,1,1,1,1

Sum= 9

According to the frequency count method, we estimate the time by counting the number of times each statement executes and adding them all together.

After this, we remove all the constants and keep only the highest-order term. This gives the time complexity of that algorithm.

For program 1- Sum = 5 + 2n

After removing the constants and keeping only the highest order term we get,

The time complexity for the first program is O(n).

We show complexity using O(), called Big O notation. It describes the complexity of the code using algebraic terms.

For program 2- Sum = 9

After removing the constants and keeping only the highest order term we get,

The time complexity for the first program is O(1).

To develop efficient software, we choose the method with less time complexity. Thus for the above example, we prefer a second method with less time complexity of O(1).

How to calculate Space Complexity?

The space needed by an algorithm is the sum of the fixed space and the variable space required. Different data types take different memory spaces as shown in the table.

Consider an example of the sum of the first N numbers.

Here input value ‘n’ is a constant of type integer and which will take the space of 4 bytes. Similarly ‘i’ and ‘sum’. Thus a total space of 12 bytes.

Now removing the constants and keeping the highest power term we get, Space complexity =O(1).

Consider another example of adding values to an Array.

Here fixed variables are ‘sum’ and ‘i’ of the integer type.

There's also a temporary or extra space used by the algorithm while 'return' is being executed. This temporary space is called auxiliary space and is calculated as a fixed space.

Thus, the fixed part is 3 variables× 4 bytes each= 12 bytes.

The size of the array is variable with integer type each, thus taking the space of 4xN.

Therefore, Total space = 4N+ 12

Removing the constants and keeping the highest power term we get Space complexity = O(N).

Together time and space complexity define the effectiveness of an algorithm/ program. In most cases, there is more than one algorithm for a particular operation. It is always best to use an algorithm with less complexity.