

**time complexity**

The time required by the algorithm to solve given problem is called time complexity of the algorithm.

int i = 1;

do

{

i++;

}while(i<=n); // time complexity of this loop is O(n).

The 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.

Time complexity is often expressed using Big O notation, which provides an upper bound on the time requirements of an algorithm in the worst-case scenario

**Types of Time Complexity:**

Constant Time (O(1)): The execution time remains constant regardless of the input size.

Logarithmic Time (O(log n)): The execution time increases logarithmically with an increase in input size. Binary search is a classic example.

Linear Time (O(n)): The execution time increases linearly with the input size. For instance, finding an item in an unsorted list.

Quadratic Time (O(n²)): The time increases quadratically with the input size. This is common in algorithms with nested loops over the input data.

Exponential Time (O(2n)): The execution time doubles with each addition to the input data set. This is typical of algorithms that solve problems by computing all possible combinations.

**Space Complexity:**

The amount of memory required by the algorithm to solve given problem is called space complexity of the algorithm.

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

To estimate the memory requirement we need to focus on two parts:

(1) A fixed part: It is independent of the input size. It includes memory for instructions (code), constants, variables, etc.

(2) A variable part: It is dependent on the input size. It includes memory for recursion stack, referenced variables, etc.

space complexity is often expressed using Big O notation. This notation provides an upper bound on the space requirements of an algorithm in the worst-case scenario

Types of Space Complexity:

Constant Space (O(1)): The algorithm uses a fixed amount of memory space regardless of the input size. For example, an algorithm that swaps two numbers.

Linear Space (O(n)): The memory required grows linearly with the input size. An example is creating a list of 'n' elements.

Quadratic Space (O(n²)): The space requirement grows quadratically with the input size, commonly seen in algorithms that create two-dimensional arrays based on the input size.