Space & time complexity

Time complexity: the amount of time taken by the algorithm to run as a function of the length of the input.

Big O notation: used to donate asymptotic upper bound. For a given function, g(n) we denote it by O(g(n)) (big O of g of n). It is also known as the worst time case complexity as it denotes the upper bound in which the algorithm terminates.

Omega(Ω) notation: it is used to denote asymptotic lower bound. For a given function, g(n) we denote it by Ω (g(n)) (big omega of g of n). It is also known as the best case complexity as it denotes the lower bound in which the algorithm terminates.

Theta(ɵ) notation: it is used to denote average time of the program. For a given function, g(n) we denote it by ɵ (g(n)) (big theta of g of n). It is also known as the average case complexity as it denotes the average bound in which the algorithm terminates.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Linear time complexity:

O(n): 1 loop of n length

1. Quadratic time complexity:

O(n^2) : nested loops

1. 2 loops (not nested)

O(n+m)

1. 2 loops with different length & nested loops

O(n\*m)

1. Constant

O(1)

Comparison of functions on the basis of time complexity:

O(n^n)> O(n!) > O(n^3) > O(n.log(n)) > O(n log(log(n))) > O(n) > O(sqrt(n))> O(log(n))> O(1)

Space Complexity of an algorithm quantifies the amount of time taken by a program to run as a function of the length of the input.

It is directly proportional to the largest memory of the program acquires any instance during run time.

Int (integer) consumes 4 bytes of memory.

Arrays:

An array is a data structure used to store blocks of information in contiguous memory allocation. Data can be integer, string, characters, class objects, etc.

//Syntax

<data\_type>Name[size]

For example: int myArray[10]; //declaration of an array

Question 1: Find maximum and minimum element from an array of size “n”.

