What is an algorithm? Explain with example the time and space analysis of an algorithm.

Algorithm Analysis

Analysis of efficiency of an algorithm can be performed at two different stages, before implementation and after implementation, as

A priori analysis – This is defined as theoretical analysis of an algorithm. Efficiency of algorithm is measured by assuming that all other factors e.g. speed of processor, are constant and have no effect on implementation.

A posterior analysis – This is defined as empirical analysis of an algorithm. The chosen algorithm is implemented using programming language. Next the chosen algorithm is executed on target computer machine. In this analysis, actual statistics like running time and space needed are collected.

Algorithm analysis is dealt with the execution or running time of various operations involved. Running time of an operation can be defined as number of computer instructions executed per operation.

Algorithm Complexity

Suppose X is treated as an algorithm and N is treated as the size of input data, the time and space implemented by the Algorithm X are the two main factors which determine the efficiency of X.

Time Factor – The time is calculated or measured by counting the number of key operations such as comparisons in sorting algorithm.

Space Factor – The space is calculated or measured by counting the maximum memory space required by the algorithm.

The complexity of an algorithm f(N) provides the running time and / or storage space needed by the algorithm with respect of N as the size of input data.

Space Complexity

Space complexity of an algorithm represents the amount of memory space needed the algorithm in its life cycle.

Space needed by an algorithm is equal to the sum of the following two components

A fixed part that is a space required to store certain data and variables (i.e. simple variables and constants, program size etc.), that are not dependent of the size of the problem.

A variable part is a space required by variables, whose size is totally dependent on the size of the problem. For example, recursion stack space, dynamic memory allocation etc.

Space complexity $S(p)$ of any algorithm p is $S(p) = A + Sp(I)$ Where A is treated as the fixed part and $S(I)$ is treated as the variable part of the algorithm which depends on instance characteristic I. Following is a simple example that tries to explain the concept

Time Complexity

Time Complexity of an algorithm is the representation of the amount of time required by the algorithm to execute to completion. Time requirements can be denoted or defined as a numerical function $t(N)$, where $t(N)$ can be measured as the number of steps, provided each step takes constant time.

For example, in case of addition of two n-bit integers, N steps are taken. Consequently, the total computational time is $t(N) = c*n$, where c is the time consumed for addition of two bits. Here, we observe that $t(N)$ grows linearly as input size increases.