**A DEEP GUIDE: DATASTRUCTURES AND ALGORITHMS**

**INTRODUCTION TO DATASTRUCTURES AND ALGORITHM:**

*Image for post*

**Data Structures :**

Data structure is a particular way of storing and organizing data in a computer so that it can be used efficiently. A data structure is a special format for organizing and storing data. General data structure types include arrays, files, linked lists, stacks, queues, trees, graphs and so on.

Depending on the organization of the elements, data structures are classified into two types:

1) **Linear data structures**: Elements are accessed in a sequential order but it is not compulsory to store all elements sequentially. Examples: Linked Lists, Stacks and Queues.

2) **Non — linear data structures:** Elements of this data structure are stored/accessed in a non-linear order. Examples: Trees and graphs.

*Image for post*

**What is an Algorithm?**

Let us consider the problem of preparing an omelette. To prepare an omelette, we follow the steps given below:

1) Get the frying pan.

2) Get the oil.

a. Do we have oil?

i. If yes, put it in the pan.

ii. If no, do we want to buy oil?

1. If yes, then go out and buy.

2. If no, we can terminate.

3) Turn on the stove, etc…

What we are doing is, for a given problem (preparing an omelette), we are providing a step-by-

step procedure for solving it. The formal definition of an algorithm can be stated as:

An algorithm is the step-by-step unambiguous instructions to solve a given problem.

In the traditional study of algorithms, there are two main criteria for judging the merits of algorithms: correctness (does the algorithm give solution to the problem in a finite number of steps?) and efficiency (how much resources (in terms of memory and time) does it take to execute the).

***Note: We do not have to prove each step of the algorithm.***

*Image for post*

**Why the Analysis of Algorithms?**

To go from city “A” to city “B”, there can be many ways of accomplishing this: by flight, by bus, by train and also by bicycle. Depending on the availability and convenience, we choose the one that suits us. Similarly, in computer science, multiple algorithms are available for solving the same problem (for example, a sorting problem has many algorithms, like insertion sort, selection sort, quick sort and many more). Algorithm analysis helps us to determine which algorithm is most efficient in terms of time and space consumed.

*Image for post*

**Goal of the Analysis of Algorithms:**

The goal of the analysis of algorithms is to compare algorithms (or solutions) mainly in terms of running time but also in terms of other factors (e.g., memory, developer effort, etc.)

*Image for post*

**What is Running Time Analysis?**

It is the process of determining how processing time increases as the size of the problem (input size) increases. Input size is the number of elements in the input, and depending on the problem type, the input may be of different types. The following are the common types of inputs:

• Size of an array

• Polynomial degree

• Number of elements in a matrix

• Number of bits in the binary representation of the input

• Vertices and edges in a graph.

**How to Compare Algorithms?**

To compare algorithms, let us define a few objective measures:

**Execution times?:**

Not a good measure as execution times are specific to a particular computer.

**Number of statements executed?:**

Not a good measure, since the number of statements varies with the programming language as well as the style of the individual programmer.

**Ideal solution?:**

Let us assume that we express the running time of a given algorithm as a function of the input size n (i.e., f(n)) and compare these different functions corresponding to running times. This kind of comparison is independent of machine time, programming style, etc.