Introduction to Data Structures and Algorithms

1. What is Data Structure?

A data structure is a particular way of organizing data in a computer so that it can be used efficiently. Data Structures is about rendering data elements in terms of some relationship, for better organization and storage. For example, we have data of an employee with name "Vineet", age 22 and he is from "iOS" Department. Here "Vineet" & "iOS" is of String data type and 22 is of integer data type.

We can organize this data as a record like Employee record. Now we can store employee's records in a file or database as a data structure.
For example:

```
"Pulkit" 24 "iOS",
"Vishnu" 24 "Android",
"Shorhab" 28 "Android"
```

2. Types of Data Structures

2.1 Primitive Data Structures (Built-In Data Structures)

Primitive data structures are those which are predefined way of storing data by the system. And the set of operations that can be performed on these data are also predefined. Primitive data structures are

- char
- int
- float
- double
- pointer

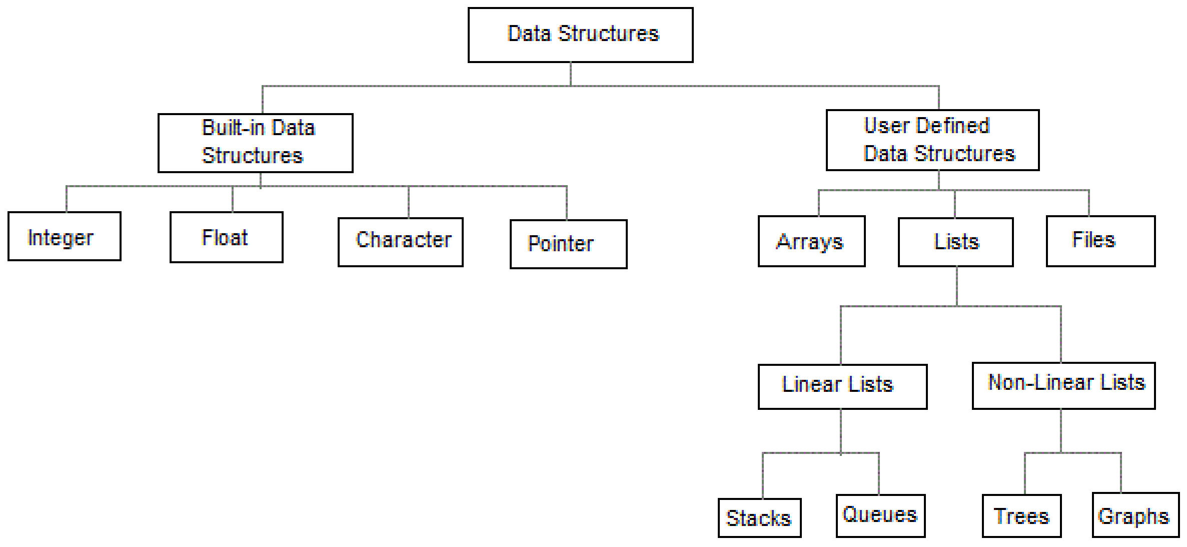
Characters are internally considered as int and floats also falls under double and the predefined operations are addition, subtraction, etc.

2.2 Non-primitive Data Structures (User Defined Data Structures)

Non-primitive data structures are more complicated data structures and they are derived from primitive data structures. Non-primitive data structures are used to store large and connected data. Some example of Non-primitive data structures are :

- Linked List
- Tree
- Graph
- Stack
- Queue etc.

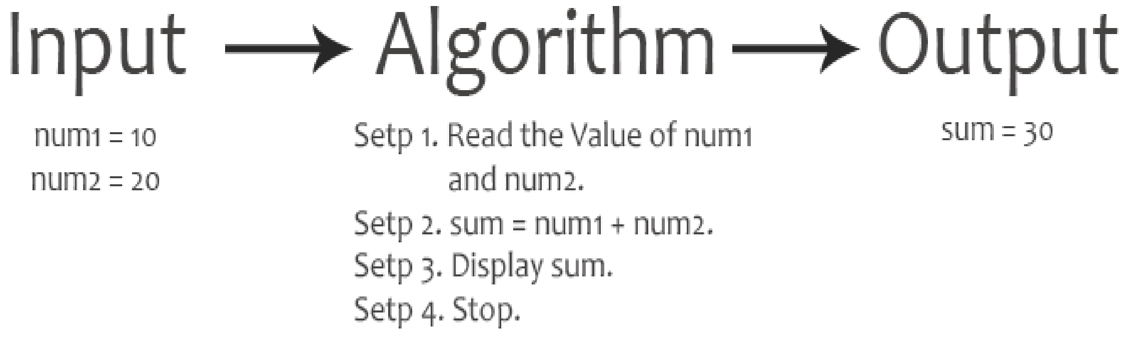
All these data structures allow us to perform different operations on data. We select these data structures based on which type of operation is required.



Types of Data Structures

3. What is Algorithm?

An algorithm is a finite set of unambiguous instructions, written in order, to accomplish a certain predefined task or obtaining a required output for any legitimate input in a finite amount of time. Algorithm is not the complete code or program, it is just the core logic of a problem.



Algorithum for Sum of Two Numbers

4. Algorithmic Analysis

If we want to go from city "A" to city "B", there can be many ways of doing this: by flight, by bus, by train and also by cycle. Depending on the availability and convenience we choose the one which suits us.

Similarly, In Computer Science there can be multiple algorithms for solving the same problem (for example, sorting problem has lot of algorithms like insertion sort, selection sort, quick sort and many more). Algorithm analysis helps us determining which of them is efficient in terms of time and space consumed.

The performance of an algorithm is measured on the basis of following properties:

- Space Complexity
- Time Complexity

5. Space Complexity

Space Complexity is the amount of memory space required by the algorithm, during the execution. Space complexity must be taken seriously for multi-user systems and in situations where limited memory is available.

So, Space complexity deals with finding out how much (extra) space would be required by the algorithm with change in the input size.

An algorithm generally requires space for following components:

5.1 Instruction Space

It’s the space required to store the executable version of the program. This space is fixed, but varies depending upon the number of lines of code in the program.

5.2 Data Space

It’s the space required to store all the constants and variables value.

5.3 Environment Space

It’s the space required to store the environment information needed to resume the suspended function.

6. Time Complexity

Time complexity of an algorithm signifies the total time required by the program to run to completion. The time complexity of algorithms is most commonly expressed using the big O notation. So, Time complexity deals with finding out how the computational time of an algorithm changes with the change in size of the input.