DATA STRUCTURE

DS are the building blocks of any program or the software. Choosing the appropriate DS for a program is the most difficult task for a programmer. Some basic terminology.

1. DATA: It can be defined as an elementary value or the collection of values, for example, student name and its id are data about the student.
2. GROUP ITEMS: Data items which have subordinate data items are called group item, for example, name of a student can have first name and the last name.
3. RECORD: Record can be defined as the collection of various data items, for example, if we talk about the student entity, then its name, address, course and marks can be grouped together to form the record for the student.
4. FILE: Collection of various records of one type of entity, for example, if there are 60 employees in the class, then there will be 20 records in the related file where each record contains the data about each employee.
5. ATTRIBUTE AND ENTITY: An entity represents the class of certain objects. It contains various attributes. Each attribute representing the attribute of an entity.
6. FIELD: Field is a single elementary unit of information representing the attribute of an entity.

NEED OF DATA STRUCURE:

Applications are getting complexed and amount of data is increasing day by day, there may arise the following problems:

1. PROCESSOR SPEED: To handle very large amount of data, high speed processing is required, but as the data is growing day by day to the billions of files per entity, processor may fail to deal with that much amount of data.
2. DATA SEARCH: Consider an inventory size of 106 items in a store, if our application needs to search a particular item, it needs to traverse 106 items every time, result in slowing down the search process.
3. MULTIPLE REQUESTS: If thousands of users are searching the data simultaneously on a web server, then there are the chances that a very large server can be failed during that process.

ADVANTAGES OF DS

1. EFFICIENCY: Efficiency of a program depends upon the choice of data structures. For example: suppose, we have some data and we need to perform the search for a particular record. In that case, if we organize our data in an array, we will have to search for a particular record. In that case, if we organize our data in an array, we will have to search sequentially element by element. Hence, using array may not be very efficient here. There are better DS which can make the search process efficient like ordered array, Binary search tree or hash tables.
2. RESUABILITY: DS are reusable, i.e. once we have implemented a particular DS, we can use it at any other place. Implementation of DS can be compiled into libraries which can be used by different clients.
3. ABSTRACTION: DS is specified by the ADT which provides a level of abstraction. The client program uses the DS through interface only, without getting into implementation details.

DATA STRUCTURRE CLASSIFICATION:

1. PRIMITIVE DS
2. NON-PRIMITIVE DS

NON-PRIMITIVE DS

1. LINEAR DS
2. NON LINEAR

LINEAR DS

1. STATIC -> (Array)
2. DYNAMIC -> (Linked List, Stack, Queue)

NON LINEAR

1. TREE
2. GRAPH

**LINEAR DS:** A DS is called linear if all of its elements are arranged in the linear order. In linear DS, the elements are stored in non-hierarchical way where each element has the successor and predecessors except the first and last element.

1. **Array**: An array is a collection of similar type of data items and each data item is called an element of the array. The data type of the element may be any valid data type like char, int, float, etc. Elements of the array shares same variable name but each one carries a different index number known as subscript.
2. **Linked List**: Linked list is a linear DS which is used to maintain a list in the memory. It can be seen as the collection of nodes stored at non-contiguous memory locations. Each node of the list contains a pointed to its adjacent node.
3. **Stack:** Stack is a linear DS in which insertion and deletions are allowed only at one end, called top. A stack is an abstract data type(ADT), can be implemented in most of the programming languages. It is named as stack because it behaves like a real-world stack, for example: Pile of plates or deck of cards etc.
4. **Queue:** Queue is a linear DS in which elements can be inserted only at one end called rear and deleted only at other end called front. It is an abstract DS, similar to stack. Queue is opened at both end therefore it follows First-In-First-OUT methodology for storing the data items.

**NON LINEAR DS:** This DS does not form a sequence i.e. each item or element is connected with two or more other items in a non-linear arrangement. The data elements are not arranged in sequential structure.

Types of non Linear DS:

1. **TREES:** Trees multilevel DS with a hierarchical relationship among its elements known as nodes. The bottommost nodes in the hierarchy are called **leaf node** while the topmost node is called root node. Each node contains pointers to point adjacent nodes.

**TREES** DS is based on the parent-child relationship among the nodes. Each node in the tree can have more than one children. Except the leaf nodes whereas each node can have at most one parent except the root node. Trees can be classified into many categories.

1. **GRAPHS:** Graphs can be defined as the pictorial representation of the set of elements connected by the links known as edges. A graph is different from tree in the sense that a graph can have cycle while the tree cannot have the one.

**OPERATIONS:**

1. **TRAVERSING:** It means visiting each element of DS in order to perform some specific operation like searching or sorting.
2. **INSERTION:** It can be classified as the process of adding the elements of the data structure at any location.
3. **DELETION:** Process of removing an element from the DS is called Deletion. We can delete an element from DS at any random location.

**NOTE:** If we try to delete an element from an empty DS then **UNDERFLOW** occurs

1. **SEARCHING:** The process of finding the location of an element within the DS is called Searching. There are 2 algorithms to perform searching, Linear Search and Binary Search.
2. **SORTING:** Process of arranging the DS in a specific order is known as sorting. There are many algorithms that can be used to perform sorting.
3. **MERGING:** When 2 lists A and list B of size M and N respectively, of similar type of elements, clubbed or joined to produce the third list, List C of size (M+N), then this process is called merging.