Sri SAI RAM ENGINEERING COLLEGE
INSTITUTE OF TECHNOLOGY
West Tambaram, Chennai - 44

YEAR	SEM
II	III

CS8391

DATA STRUCTURES

UNIT No. 1
LINEAR DATA STRUCTURES – LIST

Version: 1.XX

UNIT I LINEAR DATA STRUCTURES – LIST

- Abstract data types (ADTS)
- List ADT
- Array-based implementation
- Linked list implementation
- Singly linked lists
- Circularly linked lists
- Doubly linked lists
- Applications of lists
- Polynomial manipulation - all operation(insertion, deletion, merge, traversal)

Data: A collection of facts, concepts, figures, observations, occurrences or instructions in a formalized manner.

Information: The meaning that is currently assigned to data by means of the conventions applied to those data (i.e. processed data)

Record: Collection of related fields.

Data type: Set of elements that share common set of properties used to solve a program.

Introduction to DS

- Data structures are generally based on the ability of a computer to fetch and store data at any place in its memory, specified by a pointer

• Thus, the array and record data structures are based on computing the addresses of data items with arithmetic operations

Program:

A Set of Instructions -Data Structures + Algorithms

Data Structure = A Container stores Data

Algorithm = Logic + Control

Need of Data Structures:

As applications are getting complex and amount of data is increasing day by day, there may arise the following problems:

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.

Data Search: Consider an inventory size of 106 items in a store, If our application needs to search for a particular item, it needs to traverse 106 items every time, results in slowing down the search process.

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. In order to solve the above problems, data structures are used. Data is organized to form a data structure in such a way that all items are not required to be searched and required data can be searched instantly.

Operations on data structure

1) **Traversing:** Every data structure contains the set of data elements. Traversing the data structure means visiting each element of the data structure in order to perform some specific operation like searching or sorting.

Example: If we need to calculate the average of the marks obtained by a student in 6 different subject, we need to traverse the complete array of marks and calculate the total sum, then we will devide that sum by the number of subjects i.e. 6, in order to find the average.

2) **Insertion:** Insertion can be defined as the process of adding the elements to the data structure at any location.

If the size of data structure is **n** then we can only insert **n-1** data elements into it.

3) **Deletion:** The process of removing an element from the data structure is called Deletion. We can delete an element from the data structure at any random location.

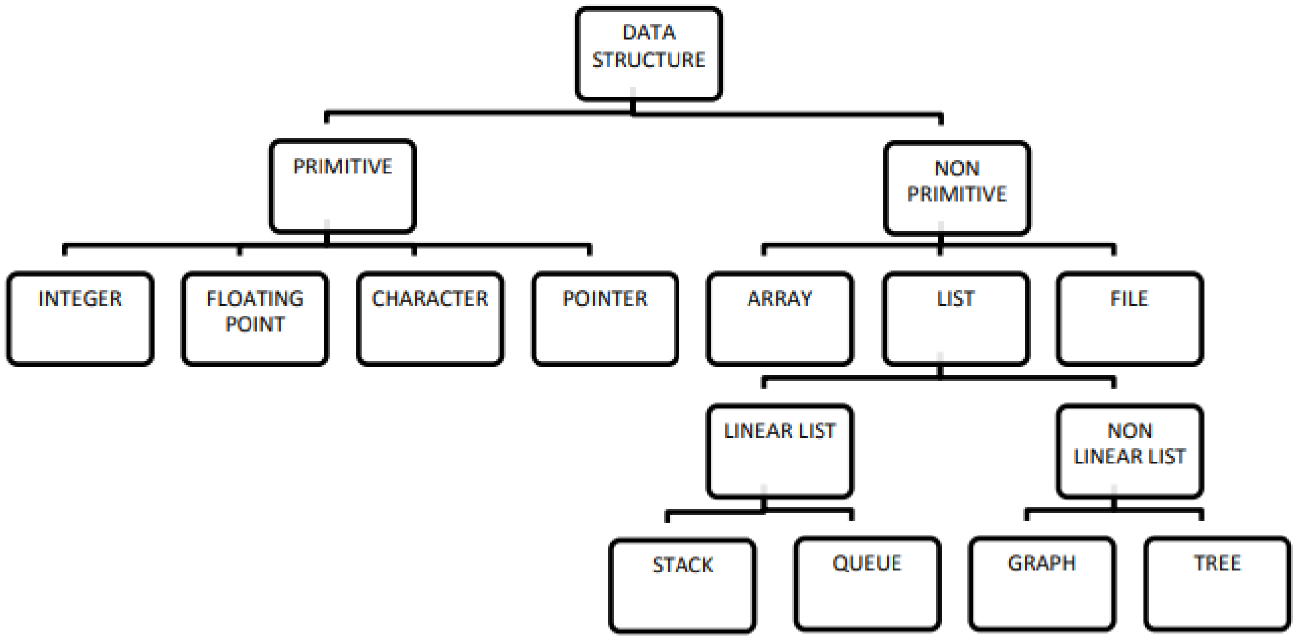
If we try to delete an element from an empty data structure then **underflow** occurs.

4) **Searching:** The process of finding the location of an element within the data structure is called Searching. There are two algorithms to perform searching, Linear Search and Binary Search. We will discuss each one of them later in this tutorial.

5) **Sorting:** The process of arranging the data structure in a specific order is known as Sorting. There are many algorithms that can be used to perform sorting, for example, insertion sort, selection sort, bubble sort, etc.

6) **Merging:** When two lists List 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

CLASSIFICATION OF DATA STRUCTURES



Primary Data Structures/Primitive Data Structures:

Primitive data structures include all the fundamental data structures that can be directly manipulated by machine-level instructions. Some of the common primitive data structures include integer, character, real, boolean etc

Secondary Data Structures/Non Primitive Data Structures:

Non primitive data structures, refer to all those data structures that are derived from one or more primitive data structures. The objective of creating non-primitive data structures is to form sets of homogeneous or heterogeneous data elements.

Linear Data Structures:

Linear data structures are data structures in which, all the data elements are arranged in linear or sequential fashion. Examples of data structures include arrays, stacks, queues, linked lists, etc.

Non Linear Structures:

In non-linear data structures, there is definite order or sequence in which data elements are arranged. For instance, a non-linear data structures could arrange data elements in a hierarchical fashion. Examples of non-linear data structures are trees and graphs.

Application of data structures:

The field of computer science will address the task of storing, accessing & manipulating data. Data structures are widely applied in the following areas:

- Compiler design
- Operating system
- Statistical analysis package
- DBMS
- Numerical analysis
- Simulation
- Artificial intelligence Graphics
- System software design ,Robotics etc.,

ABSTRACT DATA TYPES (ADTS):

An abstract Data type (ADT) is defined as a mathematical model with a collection of operations defined on that model. Set of integers, together with the operations of union, intersection and set difference form a example of an ADT. An ADT consists of data together with functions that operate on that data.

Advantages/Benefits of ADT:

- Modularity
- Reuse
- Code is easier to understand
- Implementation of ADT can be changed without requiring changes to the program that uses the ADT

LIST ADT:

List is an ordered set of elements. The general form of the list is A_1 ,A_2 , ……,A_N

A_1 - First element of the list

N –Size of the list, If the element at position i is A_i, then its successor is A_{i+1} and its predecessor is A_{i-1}

AN : last element If N=0, then empty list

Linearly ordered Ai precedes Ai+1 Ai follows Ai-1