Integrated MSc(IT)

SEMESTER – 3

1601304

DATA STRUCTURES

Unit 1: Introduction of Data Structure

- Introduction
- Abstract Data Types
- Primitive Data Types
- Difference between Abstract Data types, Data types and Data Structures
- Introduction to Linear & Non-Linear Data Structures
- Arrays in C : Representation of single & multidimensional arrays
- Sparse Matrix
  - Introduction
  - Representation through Arrays

INTRODUCTION TO DATA STRUCTURE

DATA STRUCTURE: Data Structure is a way of collecting and organizing data in such a way that we can perform operations on these data in an effective way.

DATA is nothing but a piece of information.

INFORMATION: Meaningful data or processed data. Information is used for data with its attributes.

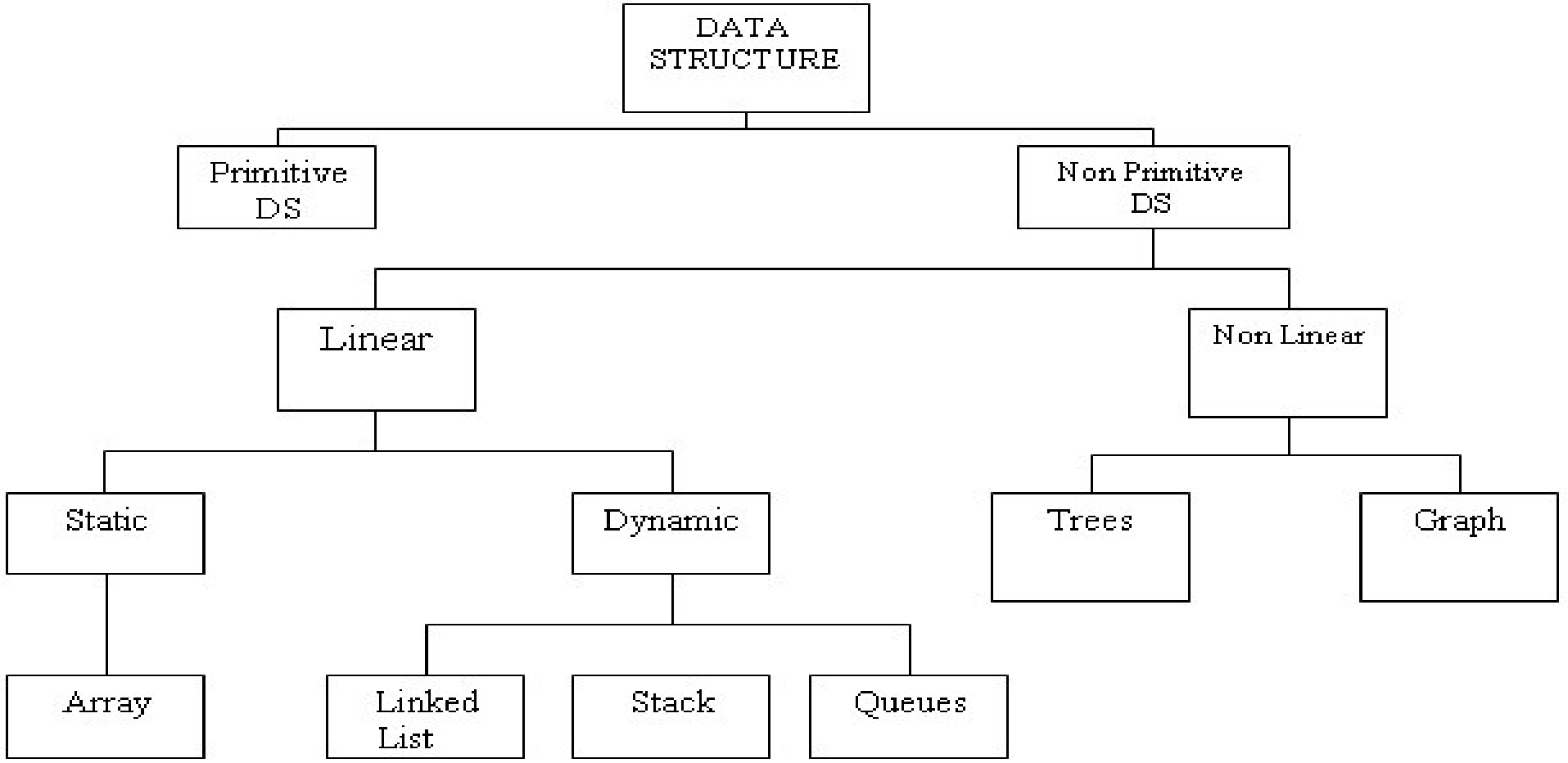
Example:

Data	Meaning
22	Age of person
Herry	Nick name of person
22/4/15	DOB of person

INTRODUCTION TO DATA STRUCTURE

- Data structures are used to store data in a computer in an organized form.
- Data Structure in C Programming Language is a specialized format for organizing and storing data.
- Different types of data structures are; Array, Stack, Queue, Linked List, Tree

Types of Data Structure



Primitive and Non-Primitive data Types

- **PRIMITIVE DATATYPES: The primitive data types are the basic data types that are available in most of the programming languages.** The primitive data types are used to represent single values. Programmers can use these data types when creating variables in their programs.
- **Primitive data types are predefined types of data.**
- For Example -
- **Integer**: This is used to represent a number without decimal point.
- **Float and Double**: This is used to represent a number with decimal point.
- **Character :** This is used to represent single character
- **String:** This is used to represent group of characters.
- **Boolean**: This is used represent logical values either true or false

Primitive and Non-Primitive data Types

- **NON-PRIMITIVE DATATYPES:** The data types that are **derived from primary data types are known as non-Primitive data types.**
- These data types are used to store group of values.
- **Non-primitive data types are not defined by the programming language, but are instead created by the programmer.**
- The non-primitive data types are:
- Arrays, Structure, Union, linked list, Stacks, Queue etc.

Types of Data Structures: Linear & Non-Linear

LINEAR DATA STRUCTURE:

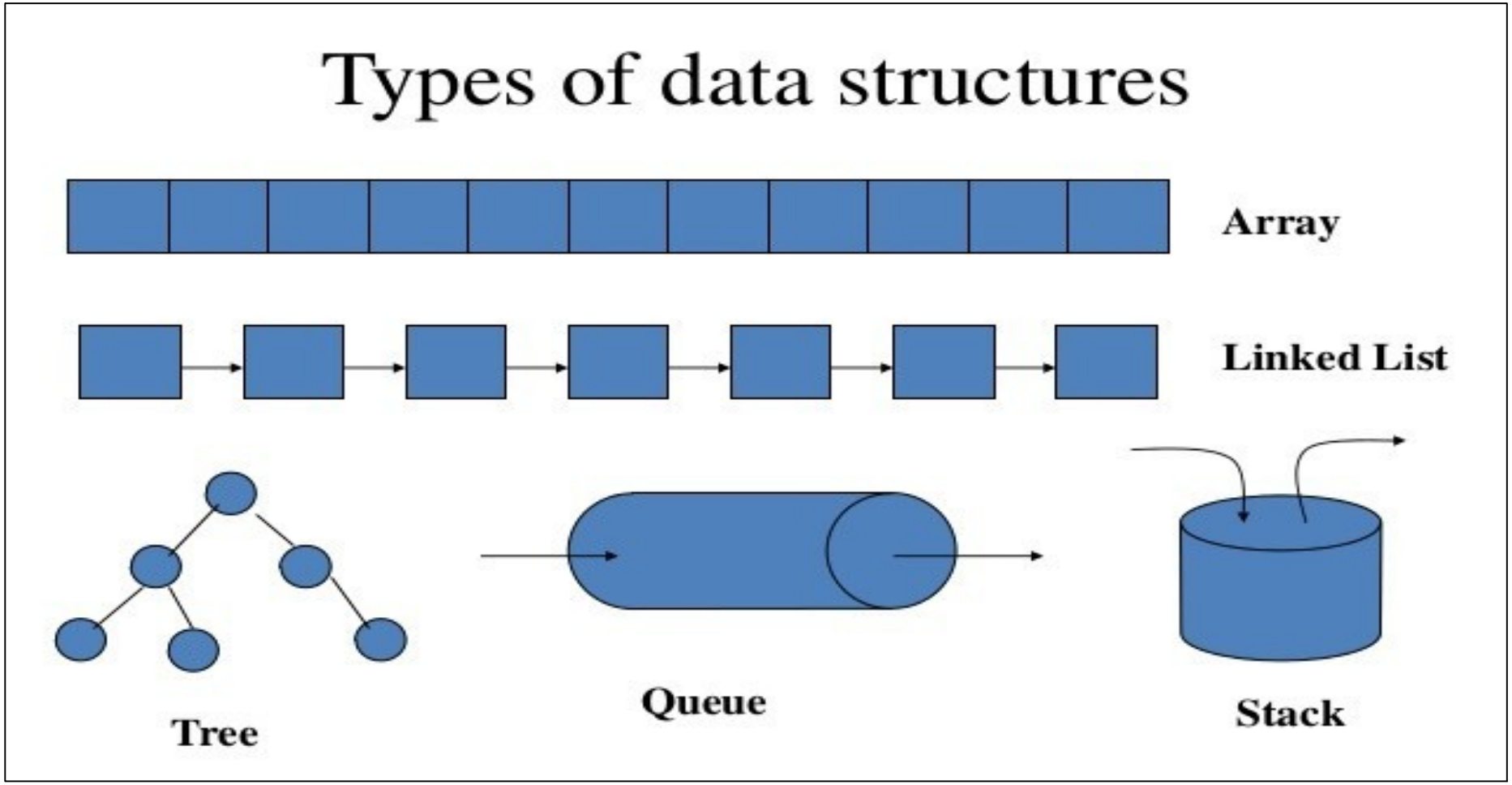
- **A linear data structure traverses the data elements sequentially, in which only one data element can directly be reached.** Ex: Arrays, Linked Lists, etc.
- Linked list stores data **in an organized, linear fashion**. They store data in the form of a list.
- A **stack** is actually a list where data elements can only be added or removed from the top of the list.
- A **queue** is also a list, where data elements can be added from one end of the list and removed from the other end of the list.

Types of Data Structures: Linear & Non-Linear

NON- LINEAR DATA STRUCTURE:

- **Every data item is attached to several other data items in a way that is specific for reflecting relationships**. The data items are not arranged in a sequential structure. Ex: Trees, Graph.
- **Tree** data structure is an example of a non linear data structure. A tree has one node called as root node that is the starting point that holds data and links to other nodes.
- A **graph** is a data structure that is made up of a finite set of edges and vertices. Edges represent connections or relationships among vertices that stores data elements.

Types of Data Structures: Linear & Non-Linear



Abstract Data Type (ADT)

- **Abstract data type are like user defined data type on which we can perform functions without knowing what is there inside the data type and how the operations are performed on them.**
- Stacks and queues are perfect example of an Abstract Data Type.
- We can implement both these ADTs using an array or a linked list.
- Data type of a variable is the set of values that the variable can take. E.g. int, char, float and double
- The word ‘abstract’ in the context of data structures means Considered apart from the detailed specifications or implementation.

Array in C

- Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type.
- Declaring Array:

Datatype arrayName [arraySize];

- Initializing Array:

Double balance [5] = {1000.0,2.0,3.4,7.0,50.0};

Array in C

Two-dimensional Arrays

The simplest form of multidimensional array is the two-dimensional array. A two-dimensional array is, in essence, a list of one-dimensional arrays. To declare a two-dimensional integer array of size [x][y], you would write something as follows:

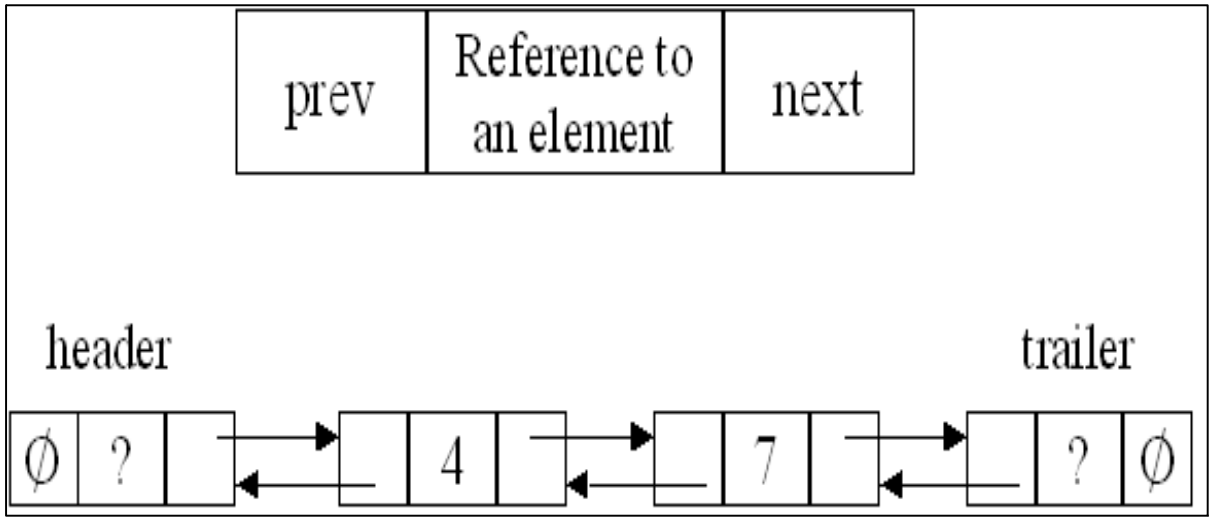
```
type arrayName [ x ][ y ];
```

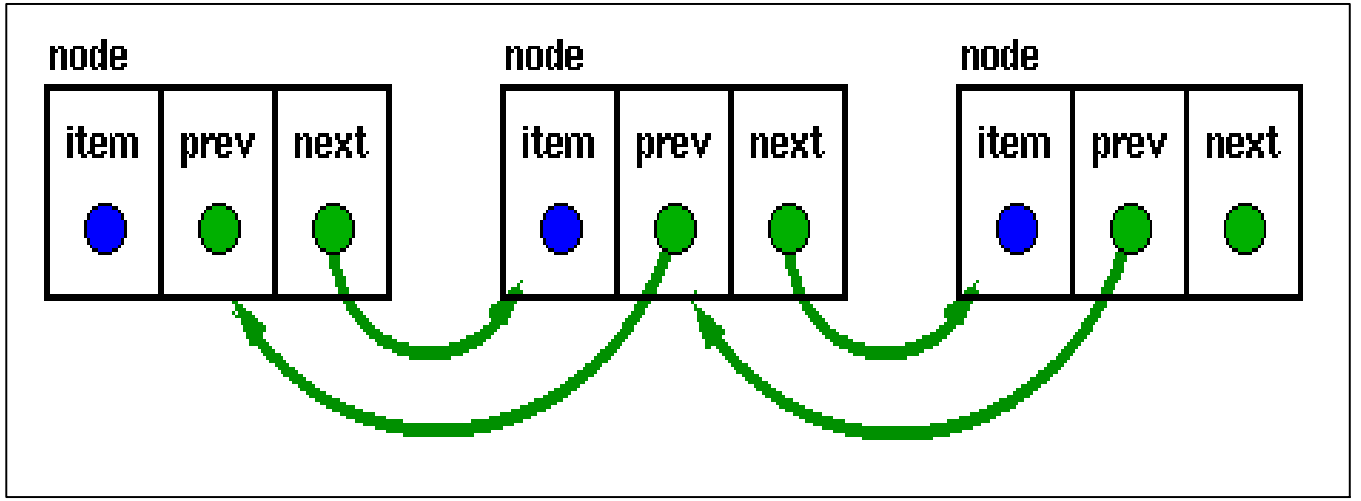
Where **type** can be any valid C data type and **arrayName** will be a valid C identifier. A two-dimensional array can be considered as a table which will have x number of rows and y number of columns. A two-dimensional array **a**, which contains three rows and four columns can be shown as follows:

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

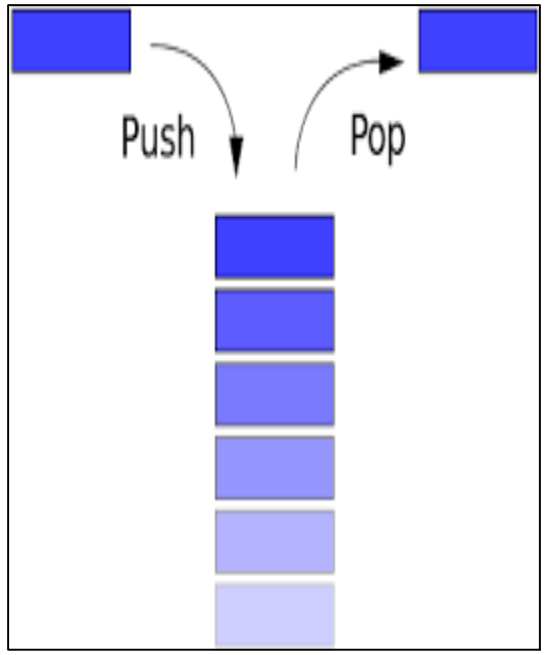
Thus, every element in the array **a** is identified by an element name of the form **a[i][j]**, where 'a' is the name of the array, and 'i' and 'j' are the subscripts that uniquely identify each element in 'a'.

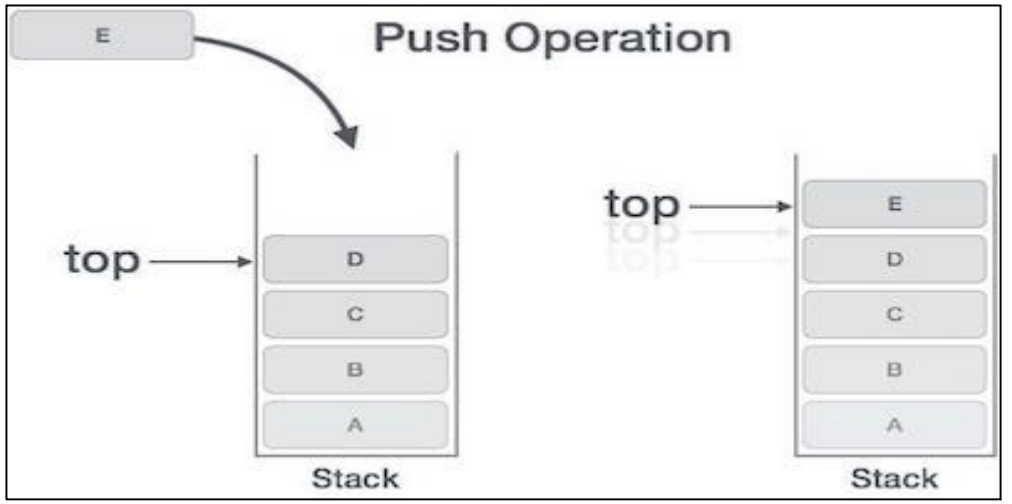
Linear: Linked List

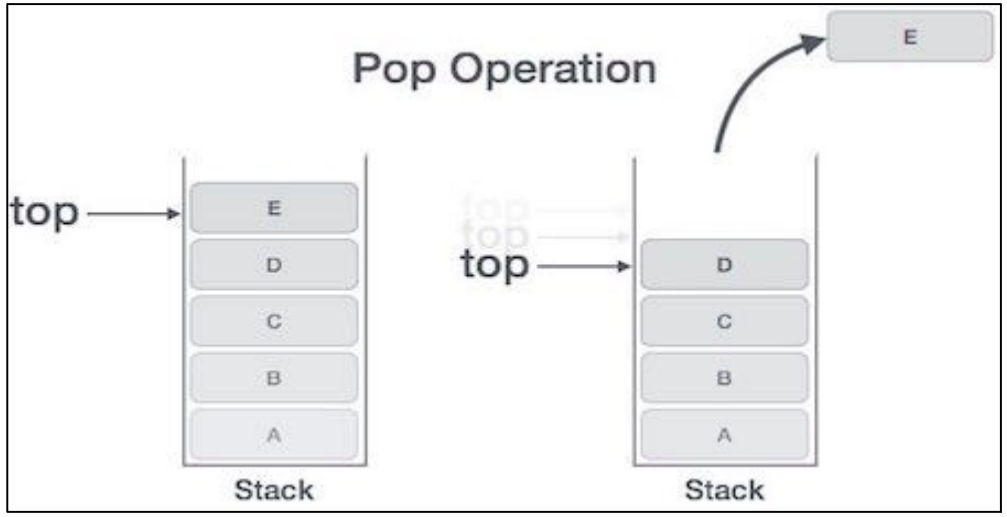




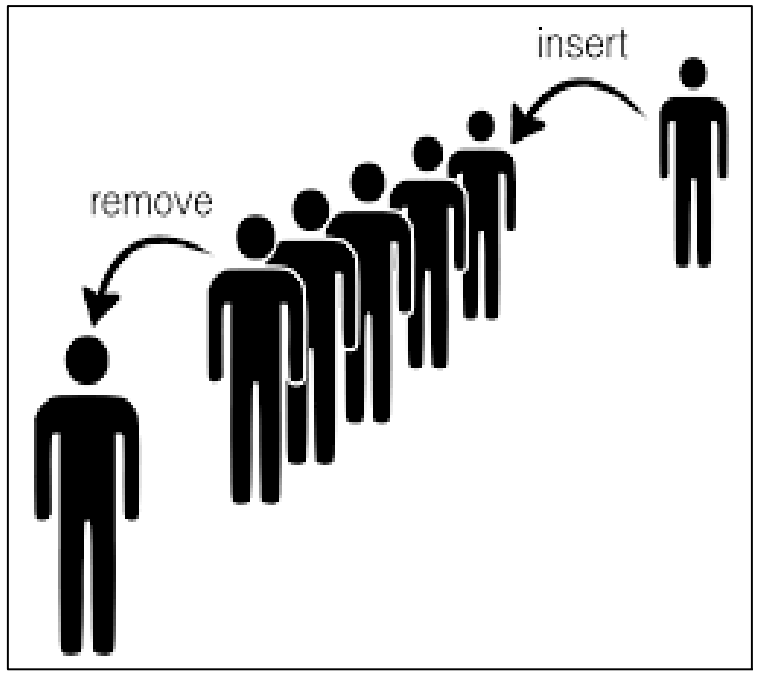
Linear : Stack

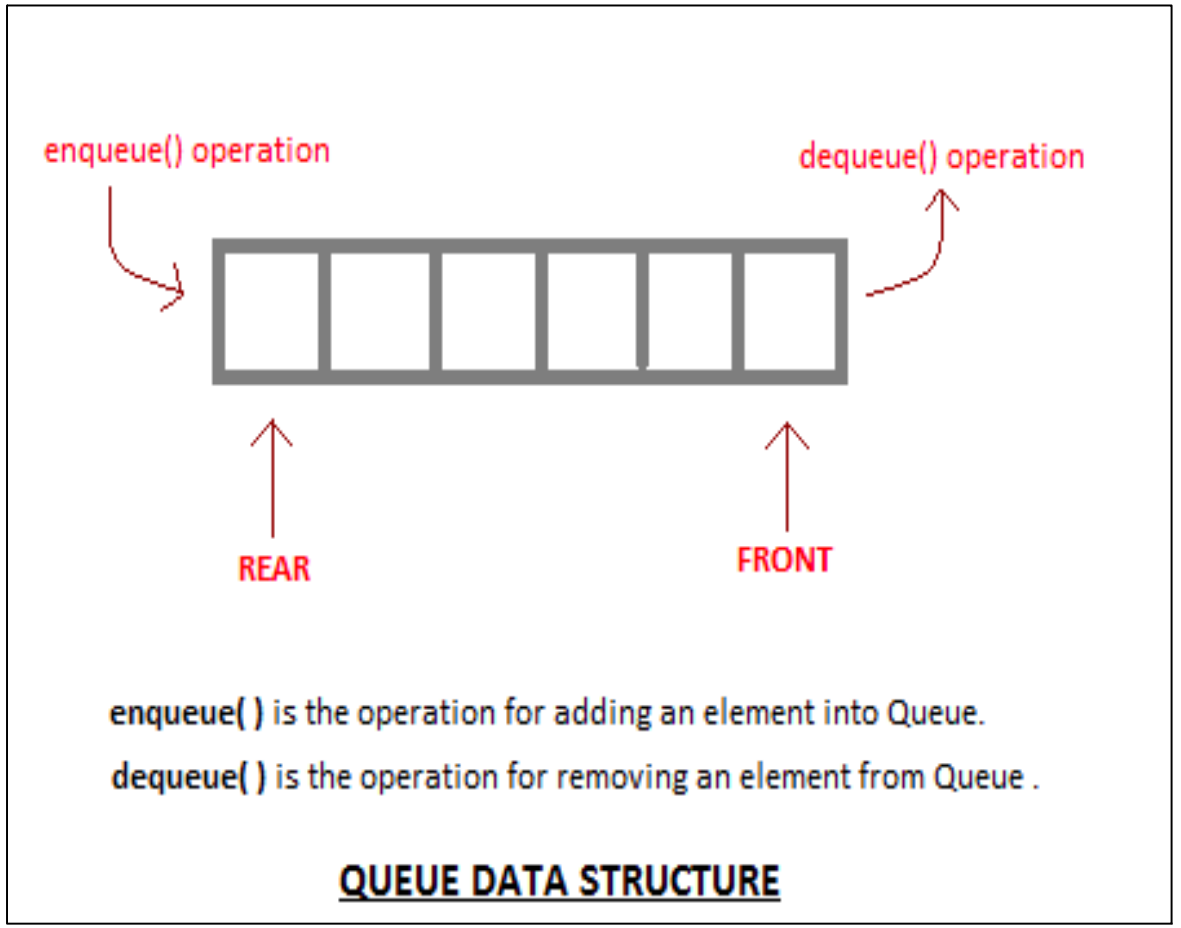






Linear : Queue



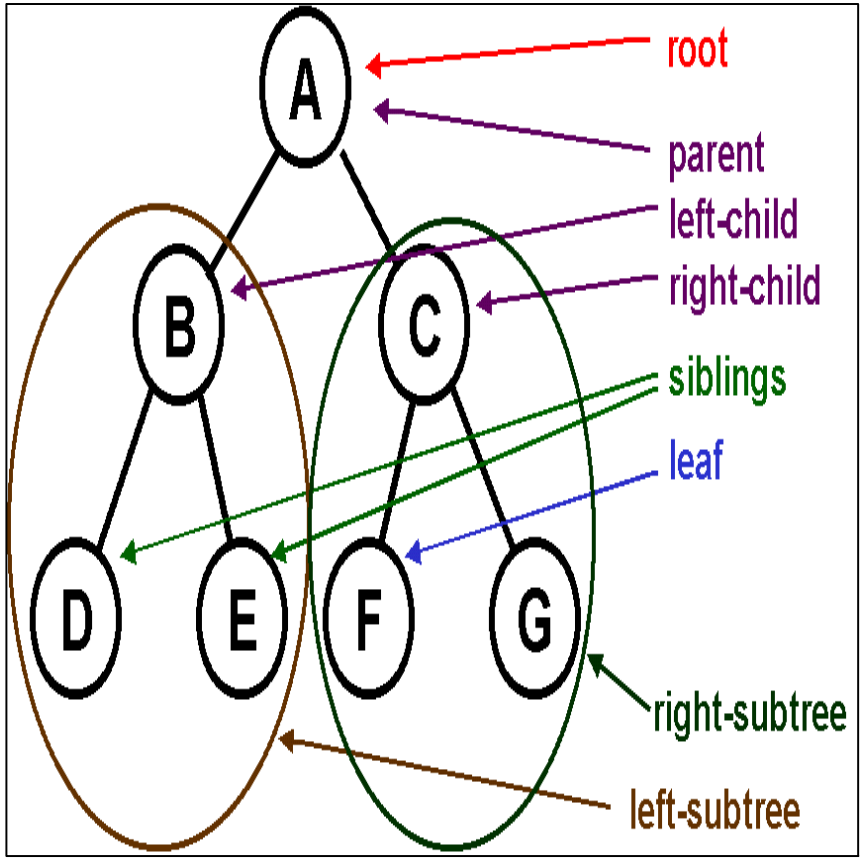


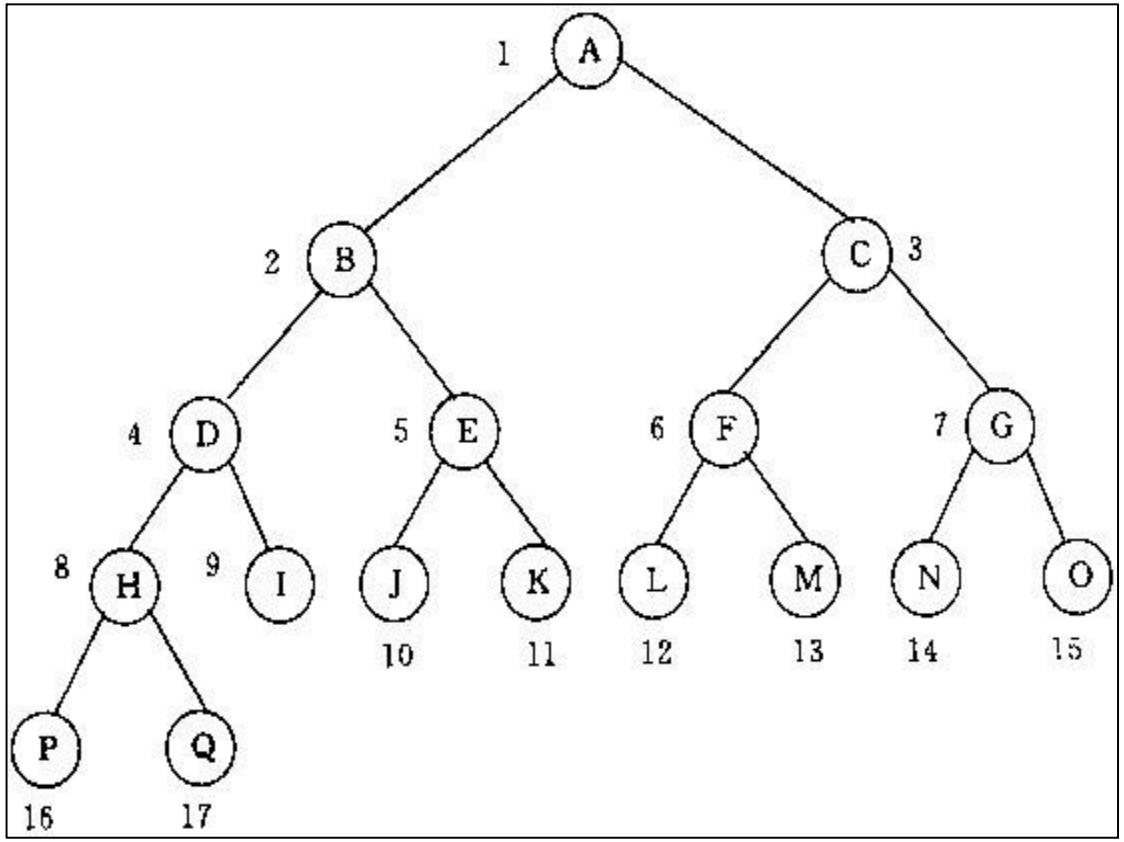
enqueue( ) is the operation for adding an element into Queue.

dequeue( ) is the operation for removing an element from Queue .

QUEUE DATA STRUCTURE

Non Linear : Tree

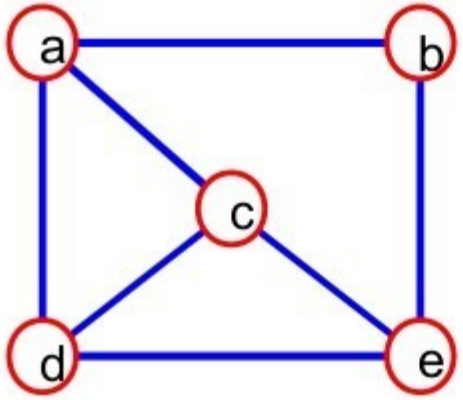




Non Linear : Graph

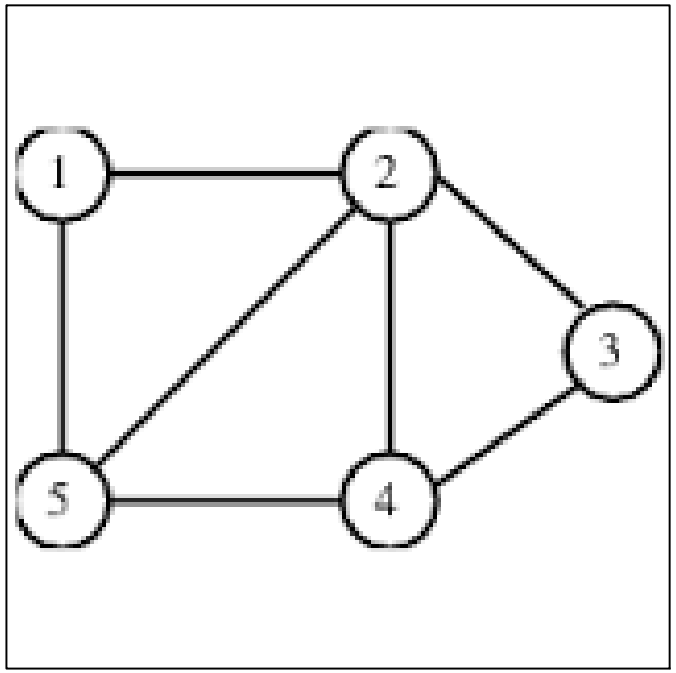
What is a Graph?

- A graph G = (V,E) is composed of:
  V: set of vertices
  E: set of edges connecting the vertices in V
- An edge e = (u,v) is a pair of vertices
- Example:



V= {a,b,c,d,e}

E= {(a,b),(a,c),
(a,d),
(b,e),(c,d),(c,e),
(d,e)}



Sparse Matrix

What is Sparse Matrix?

- In computer programming, a matrix can be defined with a 2-dimensional array. Any array with 'm' rows and 'n' columns represent a m X n matrix. There may be a situation in which a **matrix contains more number of ZERO values than NON-ZERO values. Such matrix is known as sparse matrix.**
- **Sparse matrix is a matrix which contains very few non-zero elements.**
- **Sparse Matrix Representations**
  - Triplet Representation (Array Representation)
  - Linked Representation

Sparse Matrix

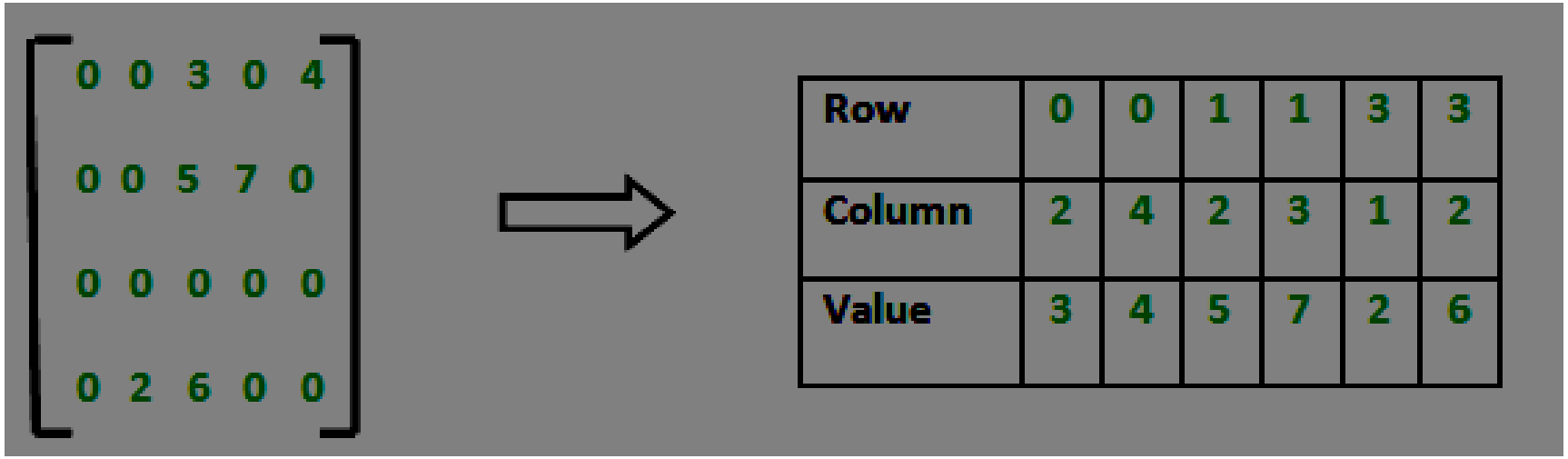
Triplet Representation:

- There are lesser non-zero elements than zeros and thus lesser memory can be used to store only those elements.
- Computing time can be saved by logically designing a data structure traversing only non-zero elements..
- Representing a sparse matrix by a 2D array leads to wastage of lots of memory as zeroes in the matrix are of no use in most of the cases. So, instead of storing zeroes with non-zero elements, we only store non-zero elements. This means storing non-zero elements with triples- (Row, Column, value).

Sparse Matrix

Triplet Representation:

- 2D array is used to represent a sparse matrix in which there are three rows named as :
- **Row**: Index of row, where non-zero element is located
- **Column**: Index of column, where non-zero element is located
- **Value**: Value of the non zero element located at index – (row,column)

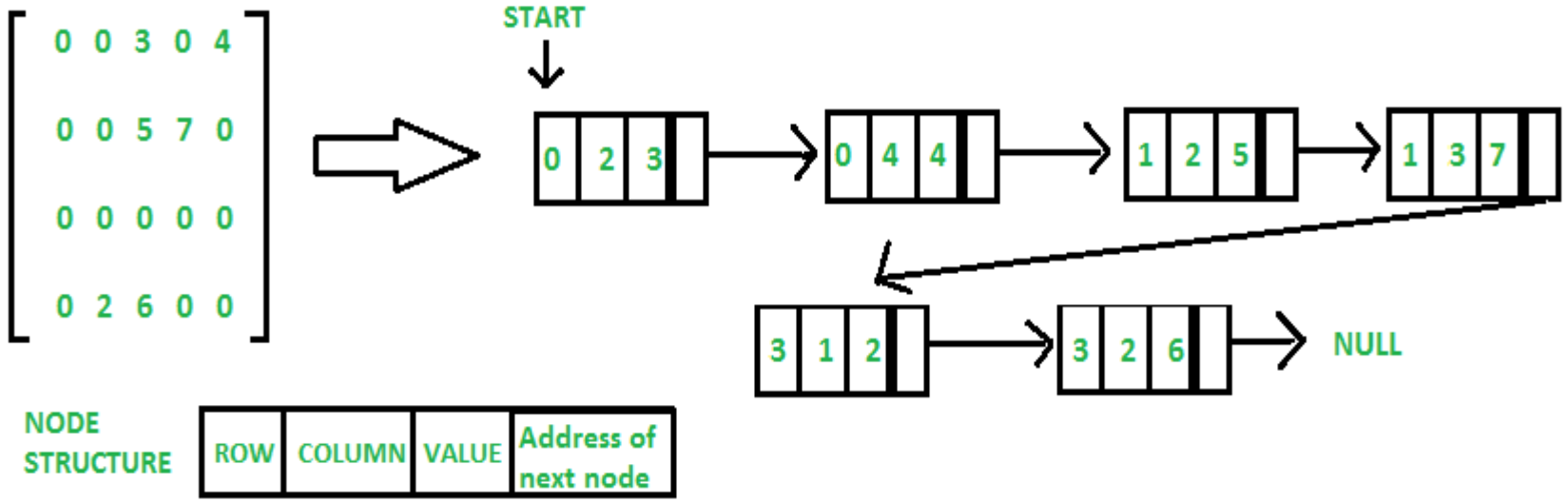


Sparse Matrix

Linked Representation: Using Linked Lists

- In linked list, each node has four fields. These four fields are defined as:
- **Row:** Index of row, where non-zero element is located
- **Column:** Index of column, where non-zero element is located
- **Value:** Value of the non zero element located at index – (row,column)
- **Next node:**

Address of the next node



Diagonal Matrix

- A square matrix has the same number of rows and columns.
- **Diagonal-Matrix:** A matrix is called a Diagonal Matrix, **if all of the non-diagonal elements of the matrix are zero.**
- Diagonal Matrix: $M(i,j) = 0$ for $i \neq j$

$$\begin{matrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 6 \end{matrix}$$

(a) Diagonal