

# Data Structure and Algorithm

## Chapter # 01

Er. Aruna Chhatkuli

Nepal College of Information Technology

Balkumari, Lalitpur

# Introduction to Data Structure

- ❑ A data structure is a **specialized format for organizing, processing, retrieving and storing data.**
- ❑ There are several basic and advanced types of data structures, all designed to arrange data to suit a specific purpose.
- ❑ Data structures make it **easy for users to access and work** with the data they need in appropriate ways

# Introduction to Data Structure

- ❑ **Data Structure** is a way of organizing all data items that considers not only the elements stored but also their relationship to each other.
- ❑ We can also define data structure as a mathematical or logical model of a particular organization of data items.
- ❑ The representation of particular data structure in the main memory of a computer is called as **storage structure**.
- ❑ The storage structure representation in auxiliary memory is called as **file structure**.
- ❑ **Data Structure** is **defined** as the way of storing and manipulating data in organized form so that it can be used efficiently.

# Introduction to Data Structure

- Data Structure mainly specifies the following four things
  - Organization of Data
  - Accessing methods
  - Degree of associatively
  - Processing alternatives for information
- Algorithm + Data Structure = Program
- Data structure study covers the following points
  - Amount of memory require to store.
  - Amount of time require to process.
  - Representation of data in memory.
  - Operations performed on that data.

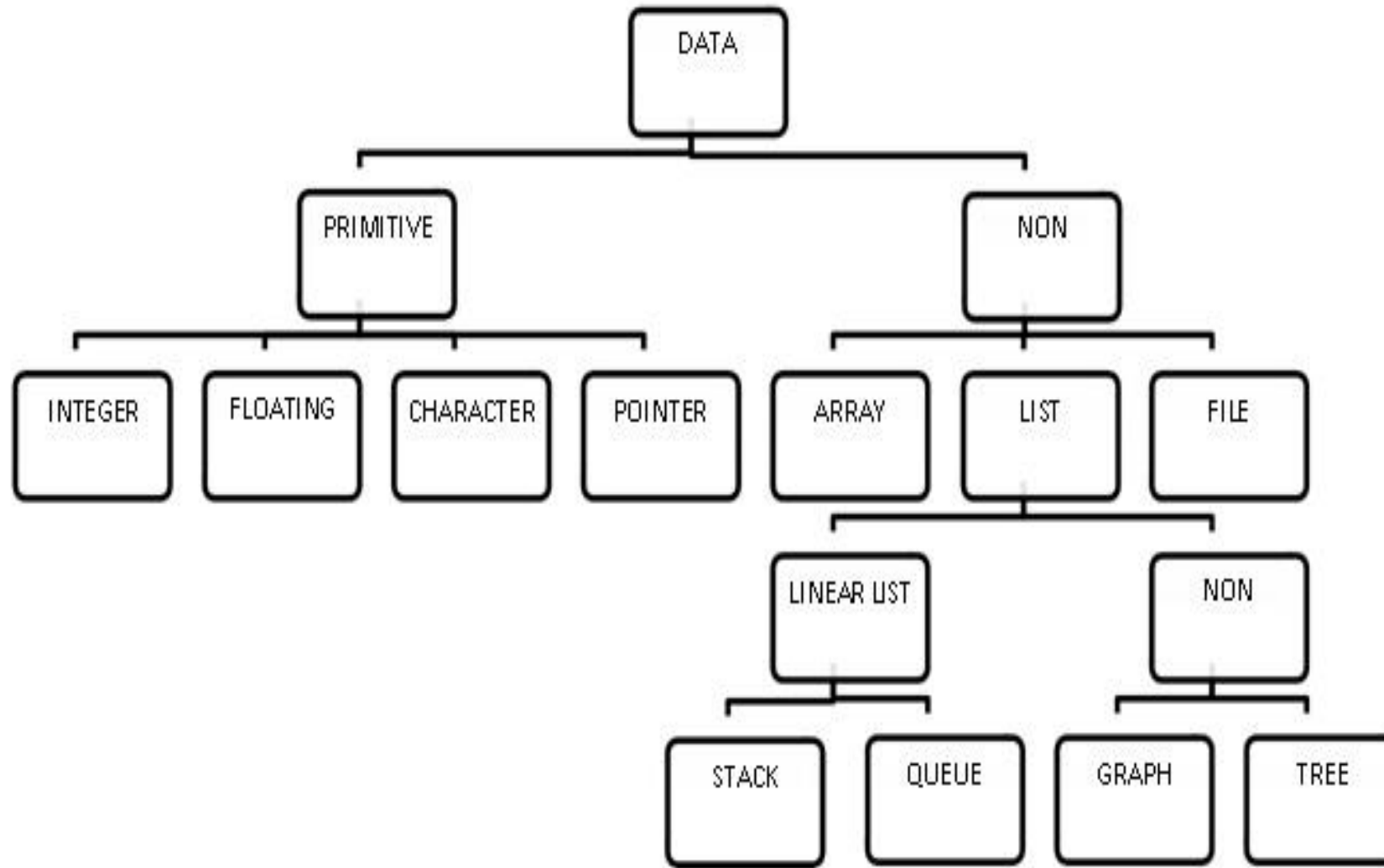
# Example of Data Structure are:

- ❖ arrays,
  - ❖ Linked List,
  - ❖ Stack,
  - ❖ Queue, etc.
- 
- ❖ Data structures are used in all areas of computer science such as Artificial Intelligence, graphics,
  - ❖ For eg: the data structure QUEUE is used by printer for printing its assigned jobs.

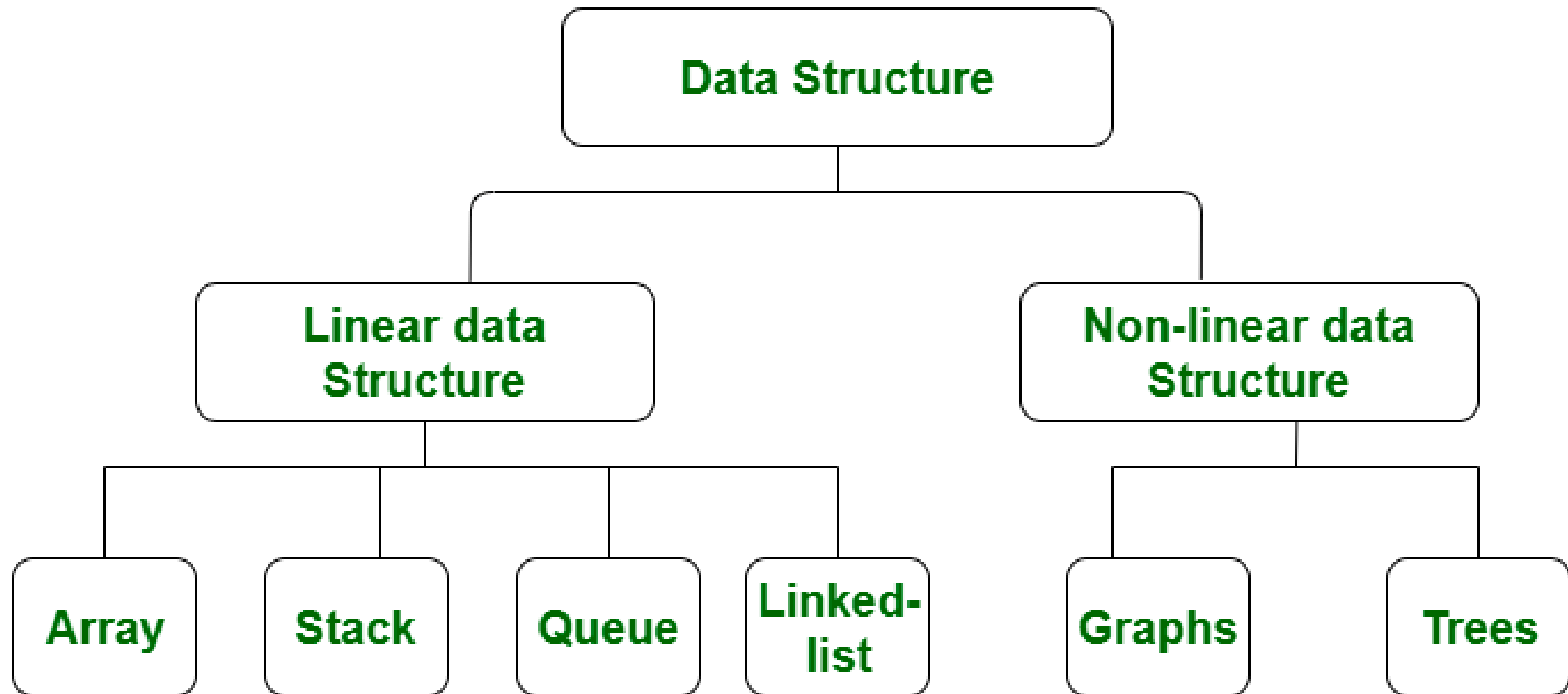
## Choice of data model depends on two considerations:

- ❑ It must be rich enough in structure to mirror the actual relationship of the data in the real world.
- ❑ The structure should be simple enough that one can effectively process the data when necessary.

# Classification of Data Structure



# Types or Classification of Data Structure



# Data Structures

**Data Structures are normally classified into two broad categories**

- Primitive Data Structure
- Non-primitive data Structure

## **Data types**

- A particular kind of data item, as defined by the values it can take, the programming language used, or the operations that can be performed on it.

# What is data types:

Two important things about data types:

1. Defines the certain Domain of the value.
2. Define operation allowed on those values.

Example:

int type

- Take only integer values.
- Operations: addition, subtraction, multiplication....etc.

# Data Structures

## Primitive Data Structure

- Primitive data structures are basic structures and are directly operated upon by machine instructions.
- Primitive data structures have different representations on different computers. **Integers, floats, character and pointers** are examples of primitive data structures.
- These data types are available in most programming languages as built in type.

## Primitive Data Structure cont..

- ❖ **Integer:** It is a data type which allows all values without fraction part. We can use it for whole numbers.
- ❖ **Float:** It is a data type which use for storing fractional numbers.
- ❖ **Character:** It is a data type which is used for character values. X=a
- ❖ **Pointer:** A variable that holds memory address of another variable are called pointer.

## Non Primitive Data Structure cont..

### **Non primitive Data Type**

- These are more sophisticated(develop to the high degree of complexity) data structures.
- These are derived from primitive data structures.
- The non-primitive data structures emphasize on structuring of a group of homogeneous or heterogeneous data items.
- Examples of Non-primitive data type are Array, List, and File etc.
- A Non-primitive data type is further divided into Linear and Non-Linear data structure

## Non primitive Data Type cont..

- ❖ **Array:** An array is a fixed-size sequenced collection of elements of the same data type.
- ❖ **List:** An ordered set containing variable number of elements is called as Lists.
- ❖ **File:** A file is a collection of logically related information. It can be viewed as a large list of records consisting of various fields.

## Non primitive Data Type cont..

### Linear data structures

- A data structure is said to be Linear, if its elements are connected in linear fashion by means of logically or in sequence memory locations.
- There are two ways to represent a linear data structure in memory,
  - ❖ Static memory allocation
  - ❖ Dynamic memory allocation
- The possible operations on the linear data structure are: **Traversal, Insertion, Deletion, Searching, Sorting and Merging.**
- Examples of Linear Data Structure are Stack and Queue .

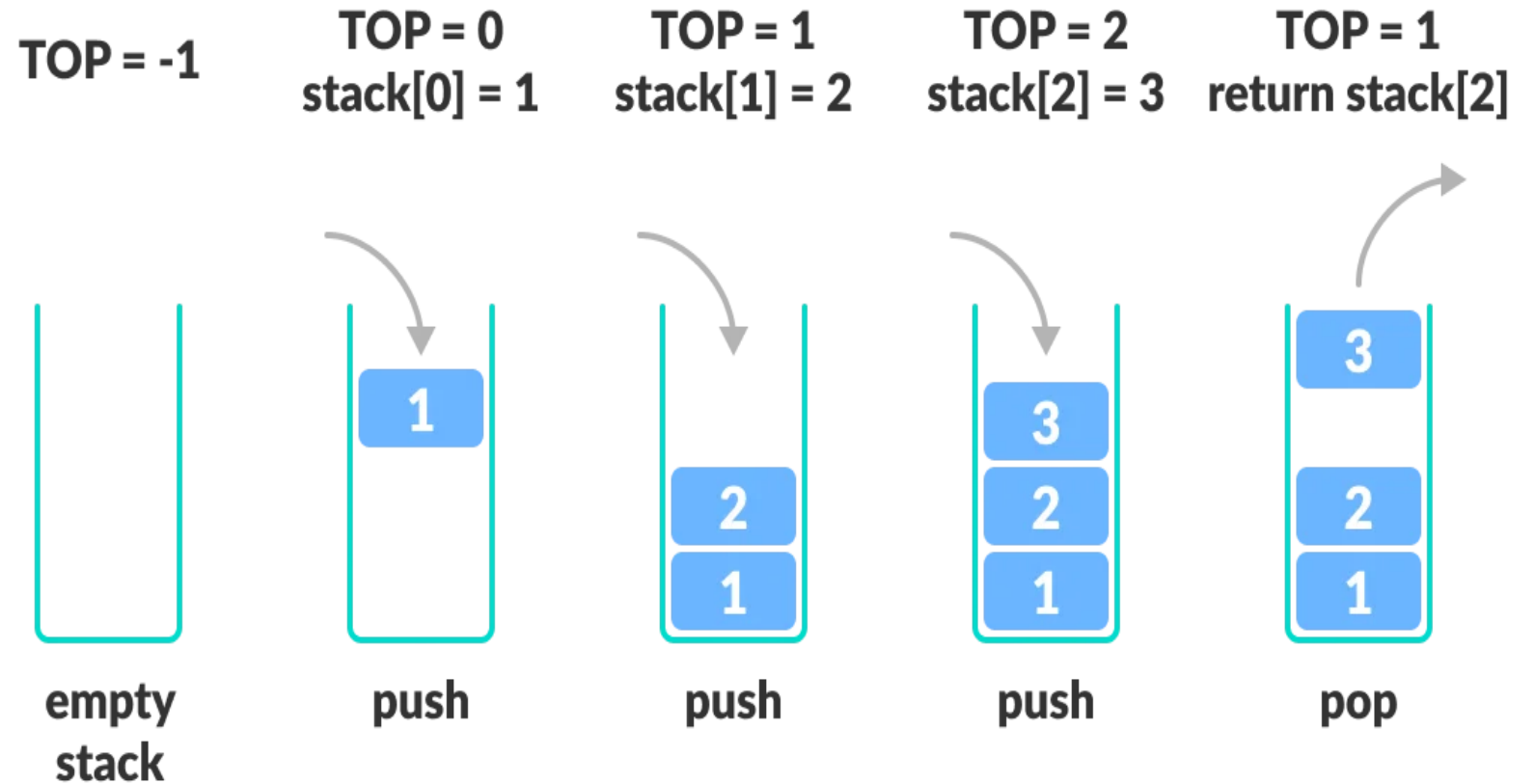
## Linear data structures cont..

**Stack:** Stack is a data structure in which insertion and deletion operations are performed at one end only.

- The insertion operation is referred to as 'PUSH' and deletion operation is referred to as 'POP' operation.
- Stack is also called as Last in First out (LIFO) data structure.

# Linear data structures cont..

## Stack:



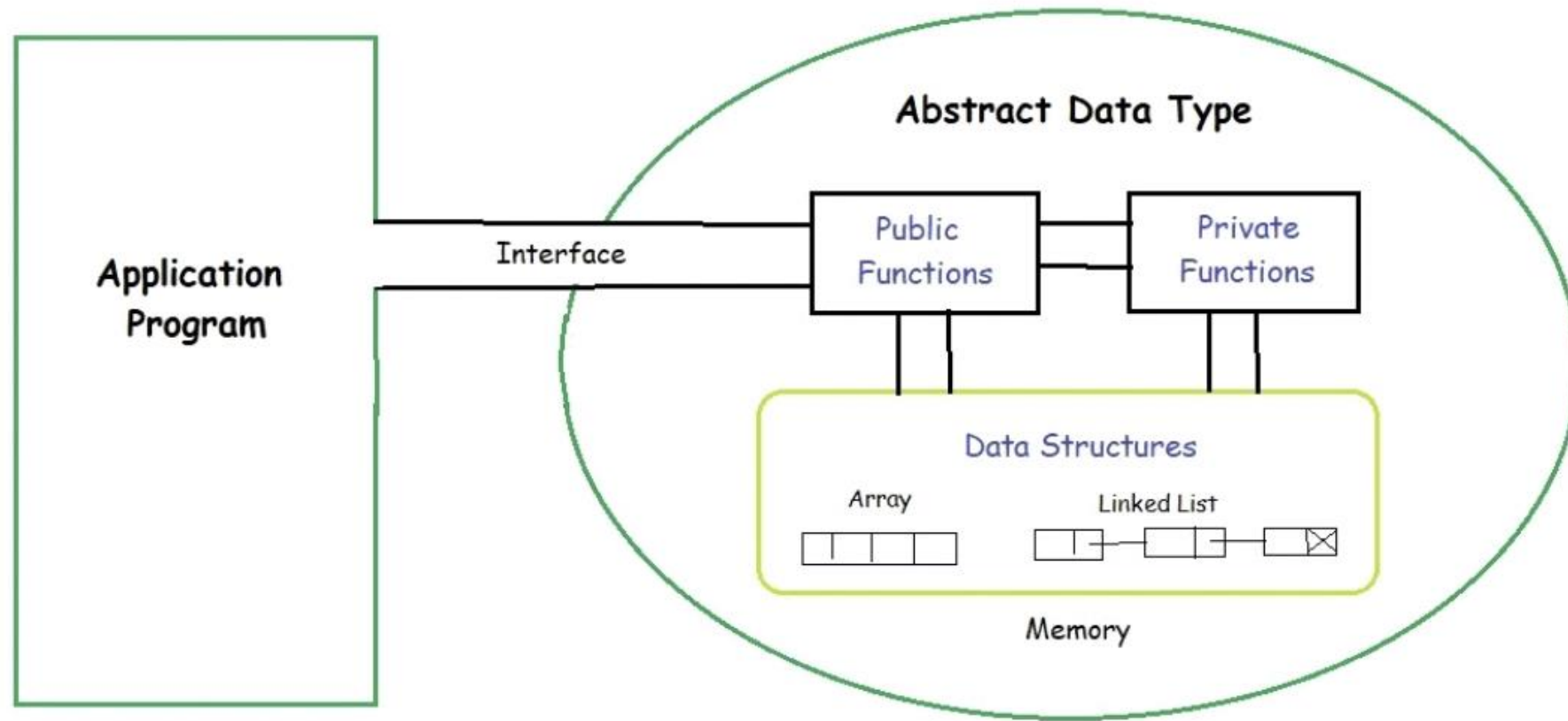
# Abstract Data Type(ADT)

- Abstract Data type (ADT) is useful tool for specifying the logical properties of the data type.
- Abstract Data type is like user define data types which defines operation on values using function without specifying what is there inside the function and how the operation are performed.

# Abstract Data Type(ADT)

- ❑ The definition of ADT only mentions **what operations are to be performed but not how these operations will be implemented.**
- ❑ It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations.
- ❑ It is called abstract because it gives an implementation independent view.
- ❑ The process of providing only the essentials and hiding the details is known as abstraction

# ADT



**Fig : ADT**

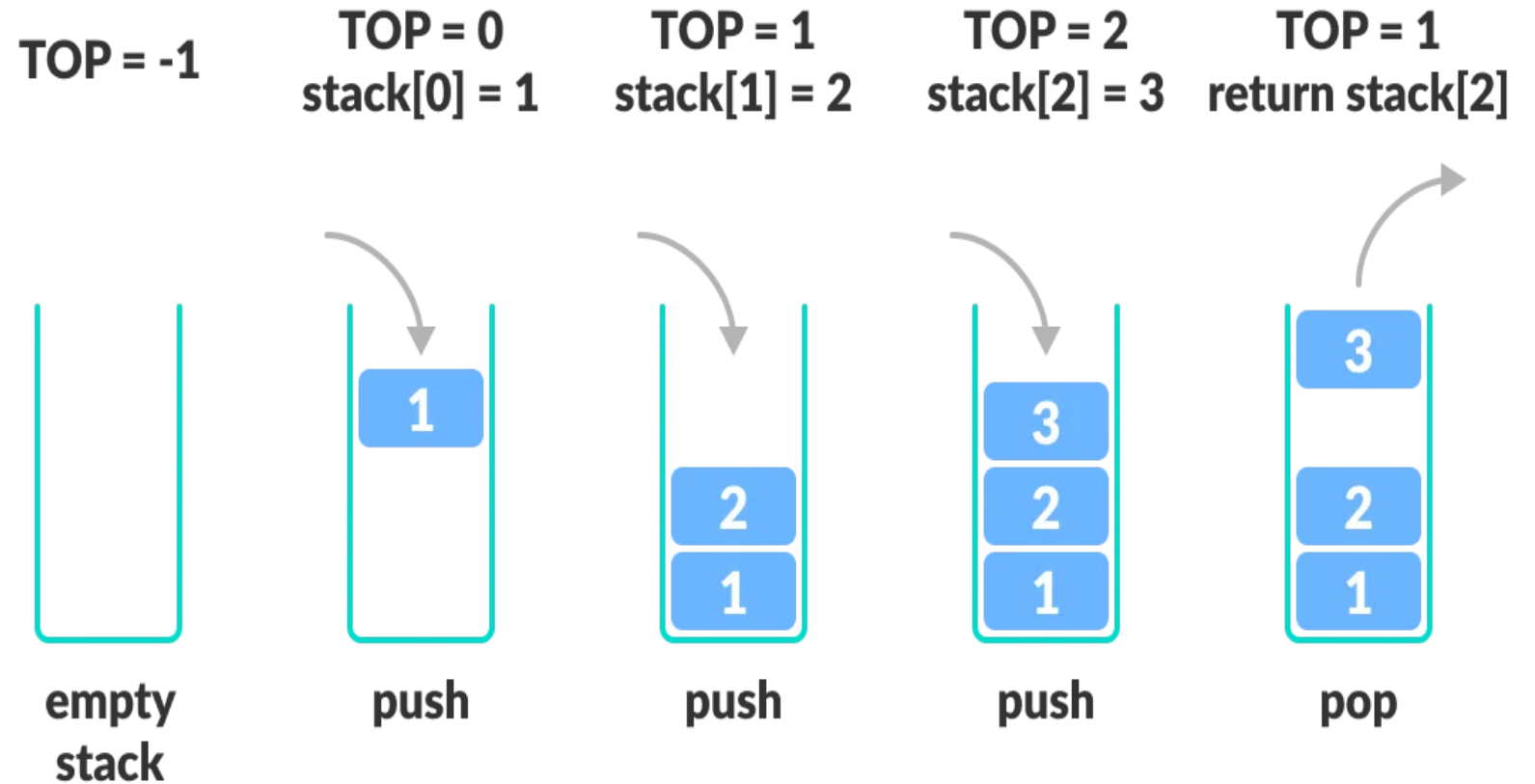
## Linear data structures cont..

**Stack:** Stack is a data structure in which insertion and deletion operations are performed at one end only.

- The insertion operation is referred to as 'PUSH' and deletion operation is referred to as 'POP' operation.
- Stack is also called as Last in First out (LIFO) data structure.

# Linear data structures cont..

## Stack:



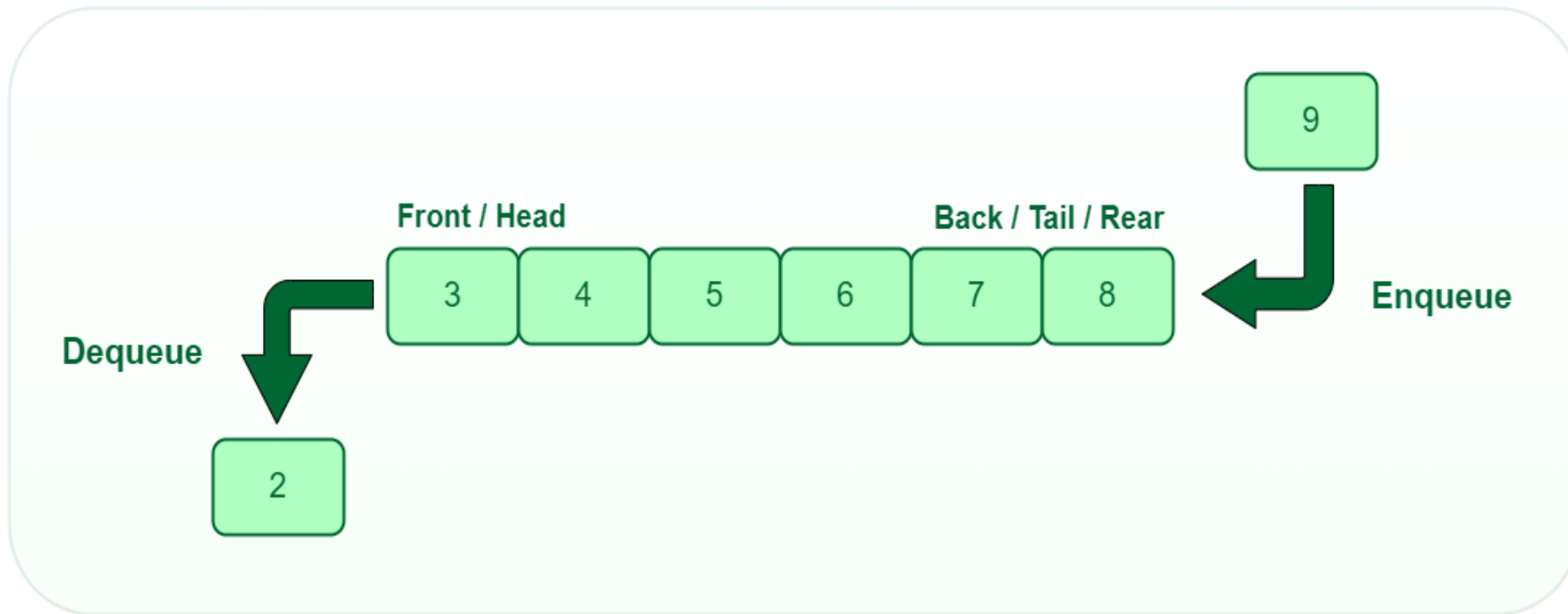
## Linear data structures cont..

**Queue:** The data structure which permits the insertion at one end and Deletion at another end, known as Queue.

- End at which deletion is occurs is known as FRONT end and another end at which insertion occurs is known as REAR end.
- Queue is also called as First in First out (FIFO) data structure.

# Linear data structures cont..

## Queue:



## Nonlinear data structures

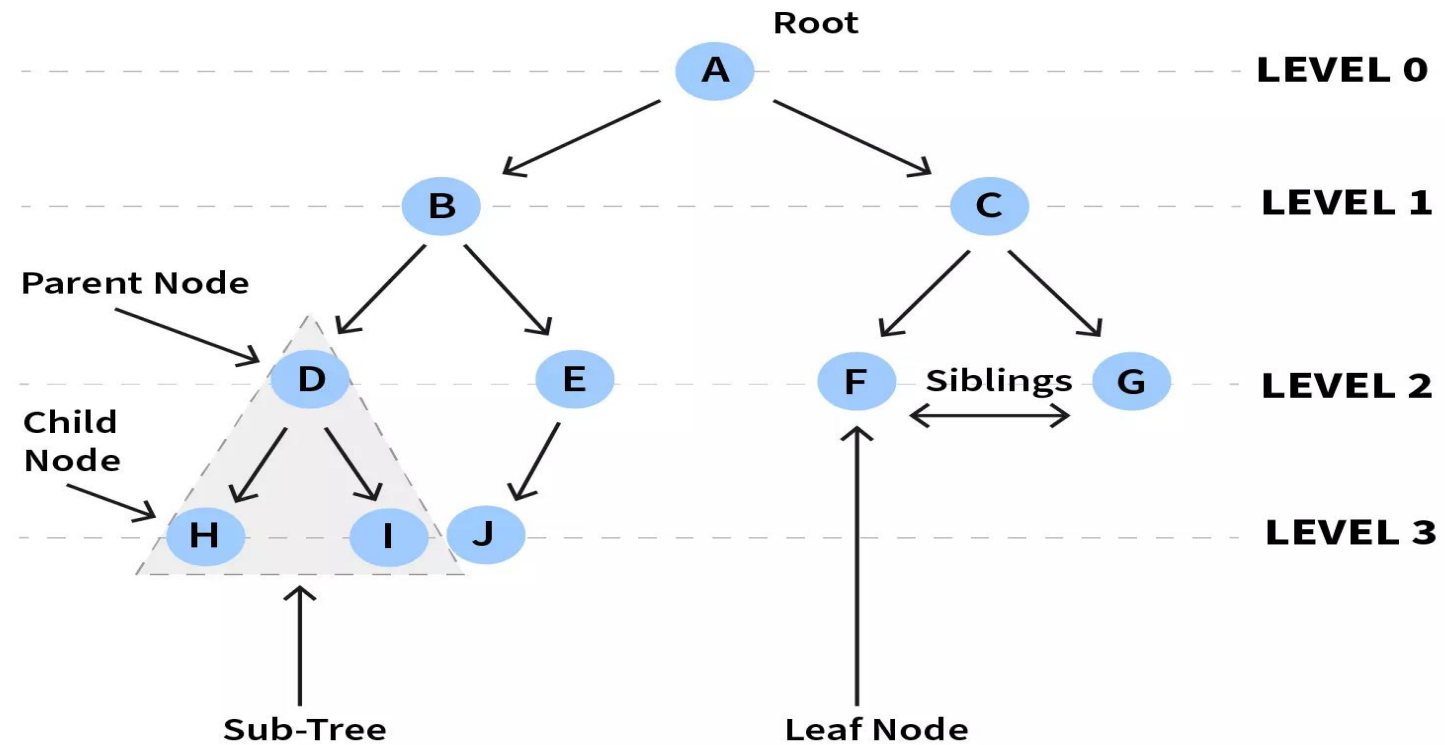
- Nonlinear data structures are those data structure in which data items are not arranged in a sequence.

Examples of Non-linear Data Structure are Tree and Graph.

- **Tree:** A tree can be defined as finite set of data items (nodes) in which data items are arranged in branches and sub branches according to requirement.
- ❖ Trees represent the hierarchical relationship between various elements.
- ❖ Tree consist of nodes connected by edge, the node represented by circle and edge lives connecting to circle.

# Nonlinear data structures

## Tree:



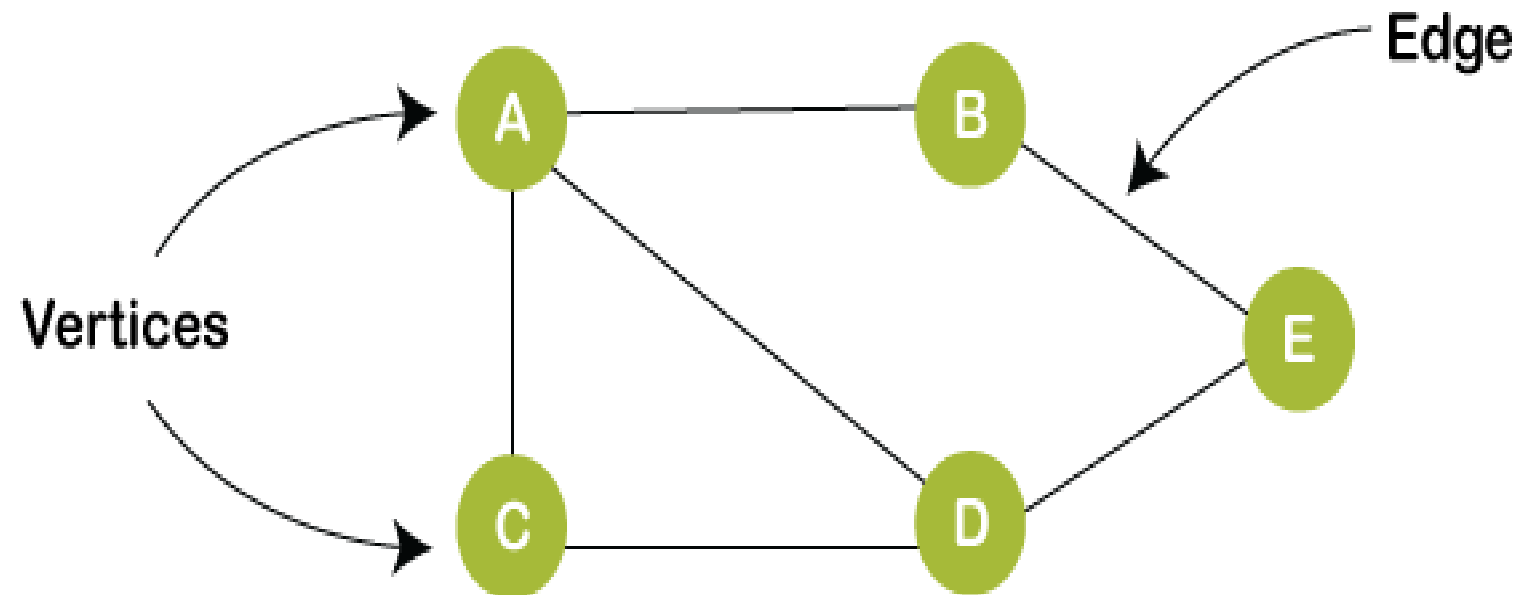
SCALER  
Topics

# Nonlinear data structures

- **Graph:** Graph is a collection of nodes (Information) and connecting edges (Logical relation) between nodes.
  - A tree can be viewed as restricted graph.
  - Graphs have many types:
    - Un-directed Graph
    - Directed Graph
    - Mixed Graph
    - Multi Graph
    - Simple Graph
    - Null Graph
    - Weighted Graph

# Nonlinear data structures

**Graph:**



# Difference between Linear and Non Linear Data Structure

| Linear Data Structure                                | Non-Linear Data Structure                     |
|--|---|
| Every item is related to its previous and next time. | Every item is attached with many other items. |
| Data is arranged in linear sequence.                 | Data is not arranged in sequence.             |
| Data items can be traversed in a single run.         | Data cannot be traversed in a single run.     |
| Eg. Array, Stacks, linked list, queue.               | Eg. tree, graph.                              |
| Implementation is easy.                              | Implementation is difficult.                  |

# Operation on Data Structures

Design of efficient data structure must take operations to be performed on the data structures into account. The most commonly used operations on data structure are broadly categorized into following types

## ***Create***

- ❑ The create operation results in reserving memory for program elements. This can be done by declaration statement.
- ❑ Creation of data structure may take place either during compile-time or run-time. malloc() function of C language is used for creation.

# Operation on Data Structures

## ***Destroy***

- ❑ Destroy operation destroys memory space allocated for specified data structure.
- ❑ `free()` function of C language is used to destroy data structure.

## ***Selection***

- ❑ Selection operation deals with accessing a particular data within a data structure.

# Operation on Data Structures

***Updating:*** It updates or modifies the data in the data structure.

***Searching:*** It finds the presence of desired data item in the list of data items, it may also find the locations of all elements that satisfy certain conditions.

***Sorting:*** *Sorting* is a process of arranging all data items in a data structure in a particular order, say for example, either in ascending order or in descending order

# Operation on Data Structures

**Merging:** Merging is a process of combining the data items of two different sorted list into a single sorted list.

**Splitting:** Splitting is a process of partitioning single list to multiple list.

**Traversal:** Traversal is a process of visiting each and every node of a list in systematic manner.

## *Algorithm*

- ❑ an algorithm is a finite sequence of rigorous instructions, typically used to solve a class of specific problems or to perform a computation.
- ❑ Algorithms are used as specifications for performing calculations and data processing.
- ❑ **An essential aspect to data structures is algorithms.**
- ❑ **Data structures are implemented using algorithms.**
- ❑ **An algorithm is a procedure that you can write as a C function or program, or any other language.**
- ❑ **An algorithm states explicitly how the data will be manipulated.**

# Goals of data structure:-

❑ Data structure involves two complementary goals:-

1. Identify and develop useful mathematical entities and operations and determine what class of problems can be solved by using these entities and operations.
2. Determine representation from those abstract entities and implement the abstract operations on these concrete representations.

# Problem

Write a C program to implement abstract data types on rational numbers and create the function where two rational numbers are added.

# Goals of Data Structure

- ***Identify and develop useful mathematical entities and operation and determine what class of problems can be solved by using these entities and operations.***
- ***Determine representation for those abstract entities and implement the abstract operations on these concrete representations.***

# Properties of Data Structure

- Generalization
  - Entity generalizes the primitive data type (integer, float)
- Encapsulation
  - All the operation of that type can be localized to one section of program.

```
#include <stdio.h>
struct rational{
    int num;
    int den;
};
struct rational sum(struct rational a, struct rational b)
{
    struct rational c;
    c.num= a.num*b.den+b.num*a.den;
    c.den=a.num*b.den;
    return c;
}
int main()
{
    struct rational x,y,z;
    printf("Enter all values:");
    scanf("%d %d %d %d",&x.num,&x.den,&y.num,&y.den);
    z=sum(x,y);
    printf("The sum is %d/%d",z.num,z.den);

    return 0;
}
```

Thank you