Module 1

Introduction to Data Structure

Agenda

– Introduction to Data structure
– Concepts of ADT
– Types of Data Structure
– Operations on Data Structure

Introduction to Data Structure

- A **good program** is defined as a program that
  - runs correctly
  - is easy to read and understand
  - is easy to debug and
  - is easy to modify.

  A program should undoubtedly give correct results, but along with that it should also run efficiently.

Introduction to Data Structure

- A program is said to be efficient when it executes in minimum time and with minimum memory space.
- In order to write efficient programs we need to apply certain data management concepts.
- A data structure is basically a **group of data** elements that are **put together** under **one name**, and which defines a particular way of storing and organizing data in a computer so that it can be used efficiently.

Introduction to Data Structure

- Some common examples of data structures are arrays, linked lists, queues, stacks, binary trees, and hash tables.
- Data structures are widely applied in the following areas:
  - Operating system
  - Statistical analysis package
  - DBMS
  - Numerical analysis
  - Simulation
  - Artificial intelligence
  - Graphics

DATA TYPE

Two important things about data types:

1. Defines a certain domain of values
2. Defines operations allowed on those values

Example:

int type

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

DATA TYPE

Example:

float type
- Takes only floating point values
- Operations: addition, subtraction, multiplication, division etc. (bitwise and % operations are not allowed)

USER DEFINED DATA TYPES

- In contrast to primitive data types, there is a concept of user defined data types.
- The operations and values of user defined data types are not specified in the language itself but is specified by the user.

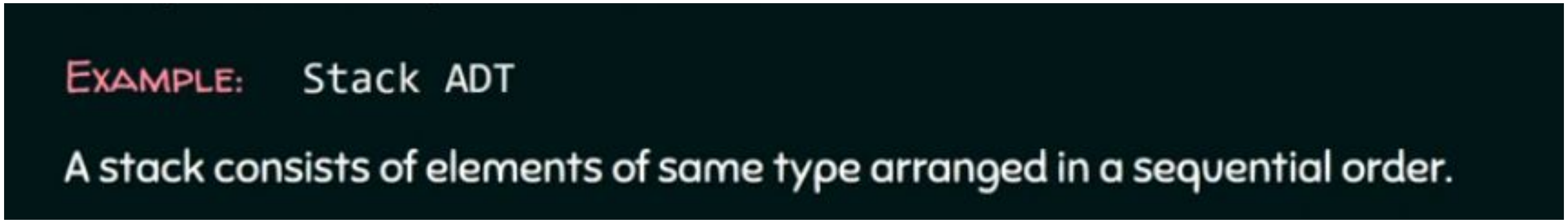
Example:

Structure, union, enumeration

By using structures, we are defining our own type by combining other data types.

Abstract Data types (ADT)

ADTs are like user defined data types which defines operations on values using functions without specifying what is there inside the function and how the operations are performed.



EXAMPLE: Stack ADT

A stack consists of elements of same type arranged in a sequential order.

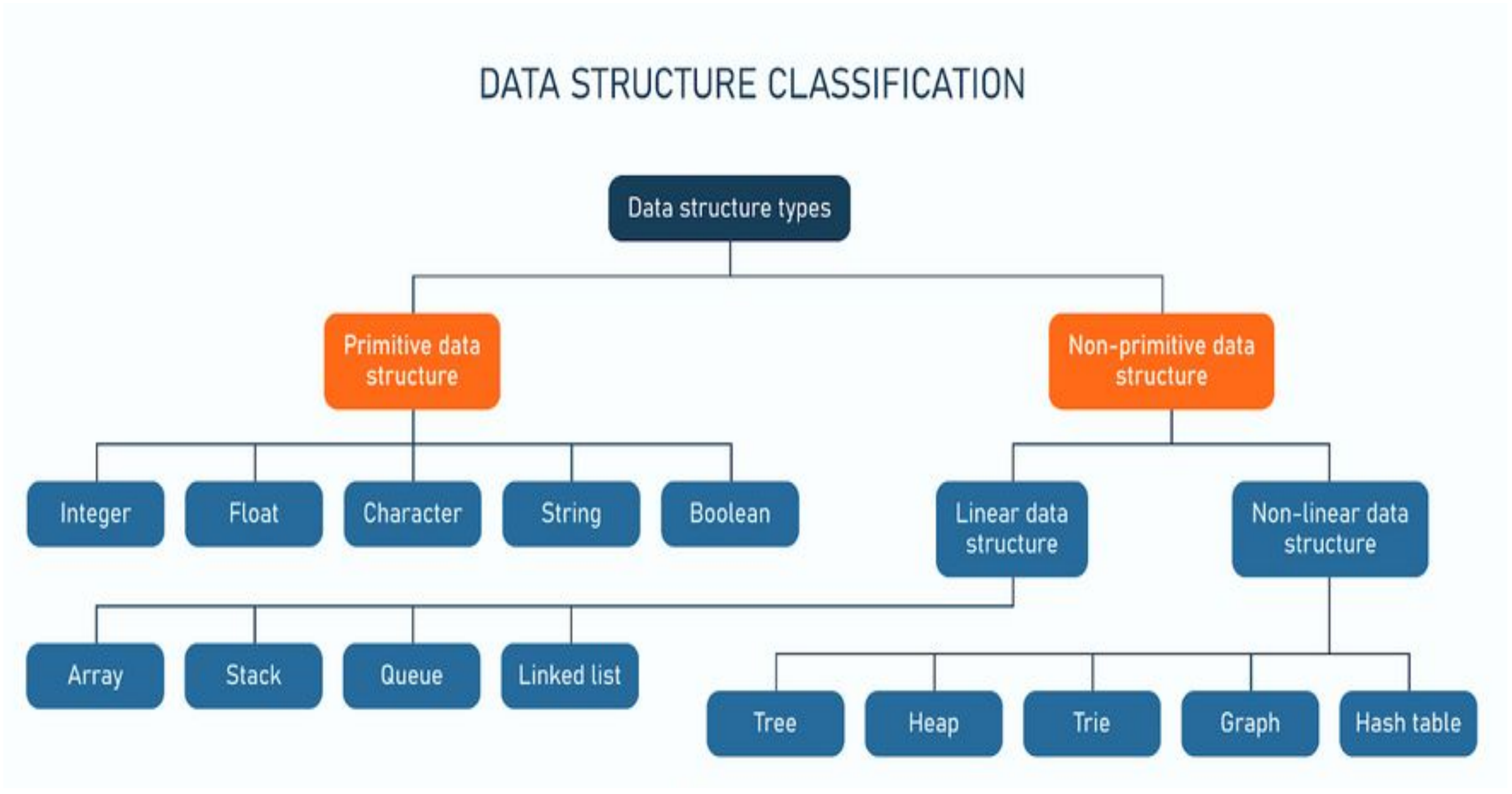
Operations:

Initialize() – initializing it to be empty
Push() – insert an element into the stack
Pop() - delete an element from the stack
isEmpty() – checks if stack is empty
isFull() – checks if stack is full

ADT(Abstract Data Type)

- To process data with a computer, we need to define the data type and the operation to be performed on the data.
- The definition of the data type and the definition of the operation to be applied to the data is part of the idea behind an abstract data type (ADT)
- ADT means to hide how the operation is performed on the data.
- In other words, the user of an ADT needs only to know that a set of operations are available for the data type, but does not need to know how they are applied.

Classification of Data Structure



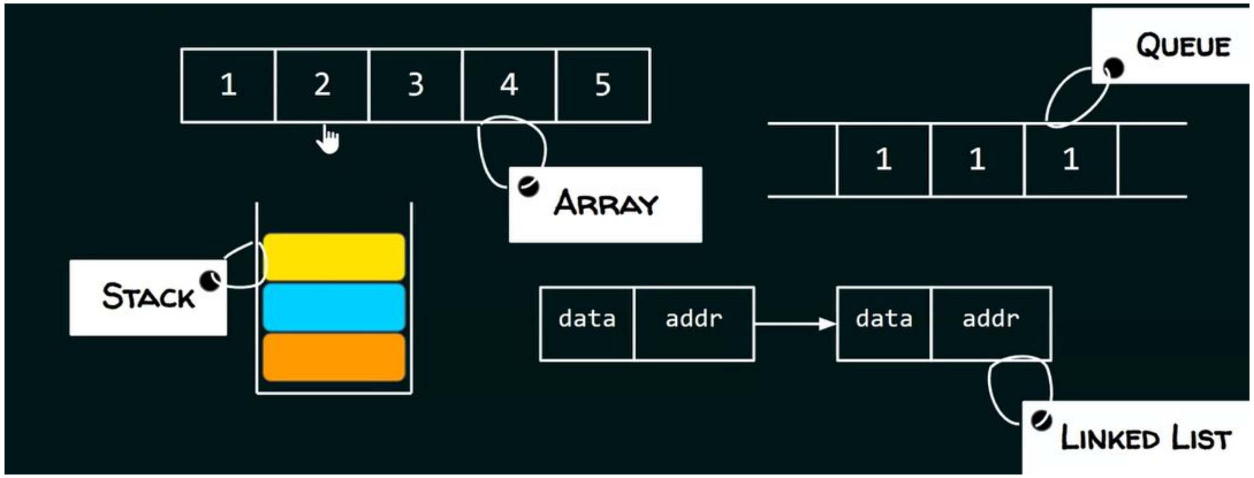
Classification of Data Structure

Primitive Data Structure: Primitive data structure is a fundamental type of data structure that stores the data of only one type whereas the non-primitive data structure is a type of data structure which is a user-defined that stores the data of different types in a single entity.

Non Primitive Data Structure: The non-primitive data structure is a kind of data structure that can hold multiple values either in a contiguous or random location. The non-primitive data types are defined by the programmer. It can be classified as Linear and Non-Linear Data Structure.

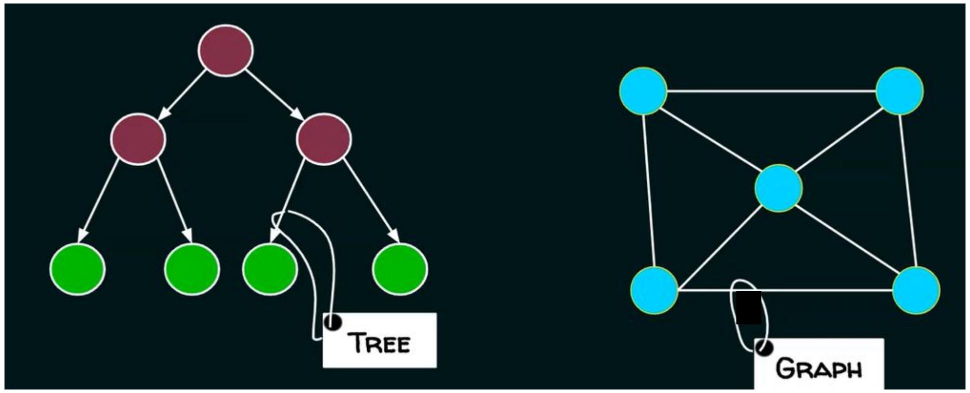
Types of Data Structure

Linear Data Structures: A linear data structure traverses the data elements sequentially, in which only one data element can directly be reached. Ex: Arrays, Linked Lists



Non Linear Data Structures: A data structure is non linear when all the elements are not arranged in a linear (sequential) order.

Ex: Tree, Graph



Operation on Linear/Non-Linear Data Structure

- Add an element
- Delete an element
- Traverse / Display
- Sort the list of elements
- Search for a data element

Types of Data Structure

Array: is commonly used in computer programming to mean a contiguous block of memory locations, where each memory location stores one fixed-length data item. e.g. Array of Integers int ar[10], Array of Character char arr[10]

Array of Integers									
0	1	2	3	4	5	6	7	8	9
5	6	4	3	7	8	9	2	1	2

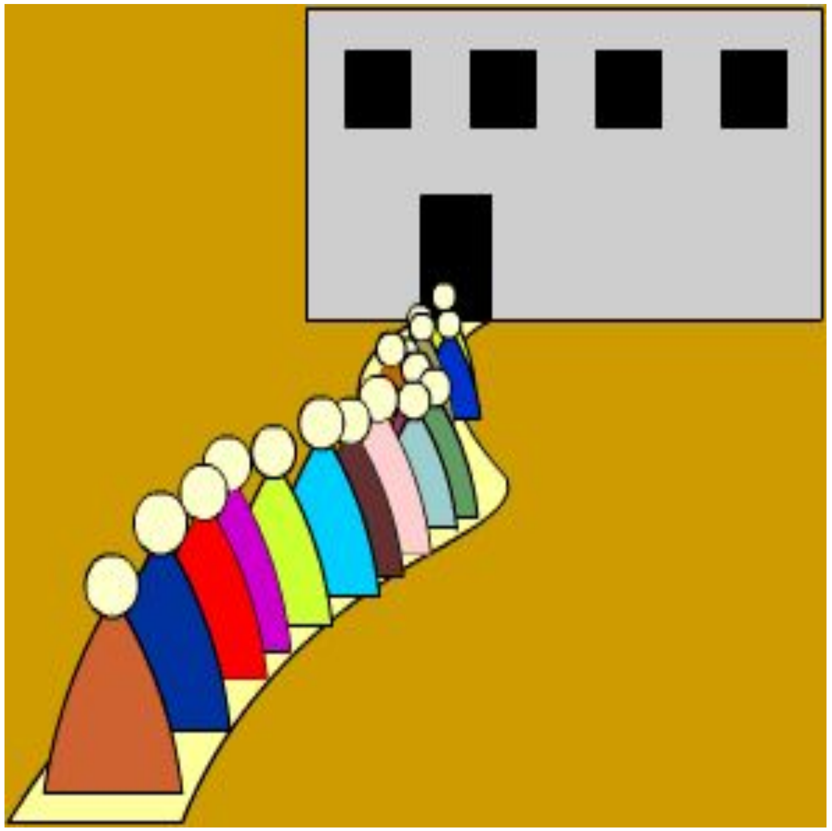
Array of Character									
0	1	2	3	4	5	6	7	8	9
a	b	c	d	e	f	g	h	i	f

Types of Data Structure

Stack: A stack is a data structure in which items can be inserted only from one end and get items back from the same end. There, the last item inserted into stack, is the first item to be taken out from the stack. In short its also called Last in First out [LIFO].

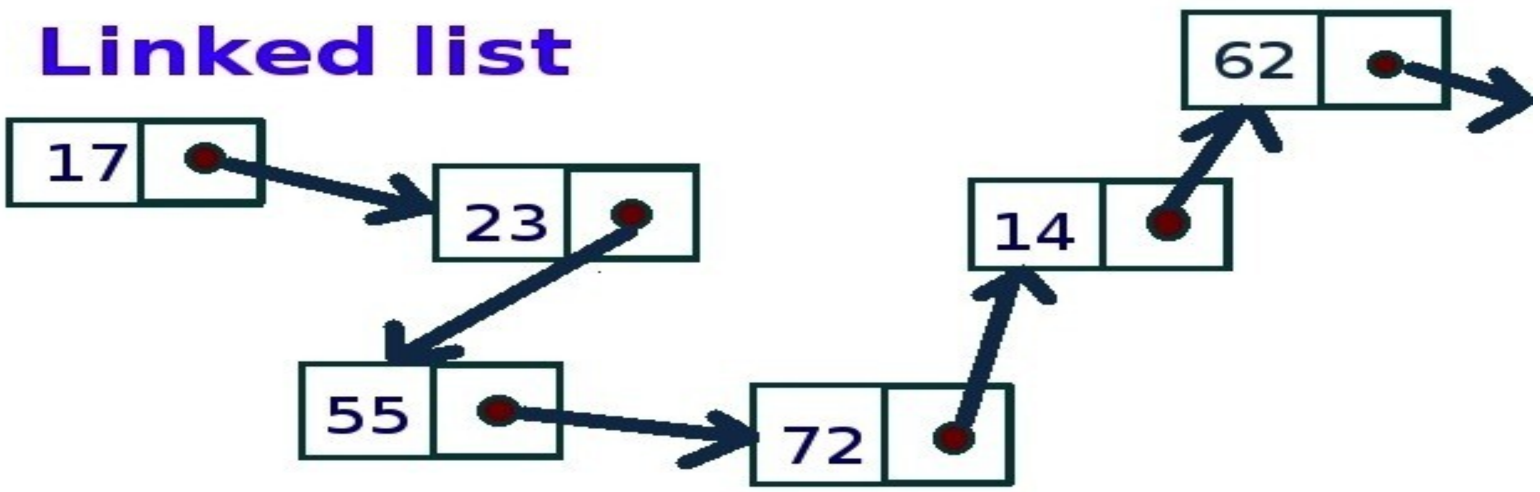
Types of Data Structure

Queue: A queue is a data structure in which items can be inserted from one end and taken out from the other end. Therefore, the first item inserted into queue is the first item to be taken out from the queue. This property is called First in First out [FIFO].



Types of Data Structure

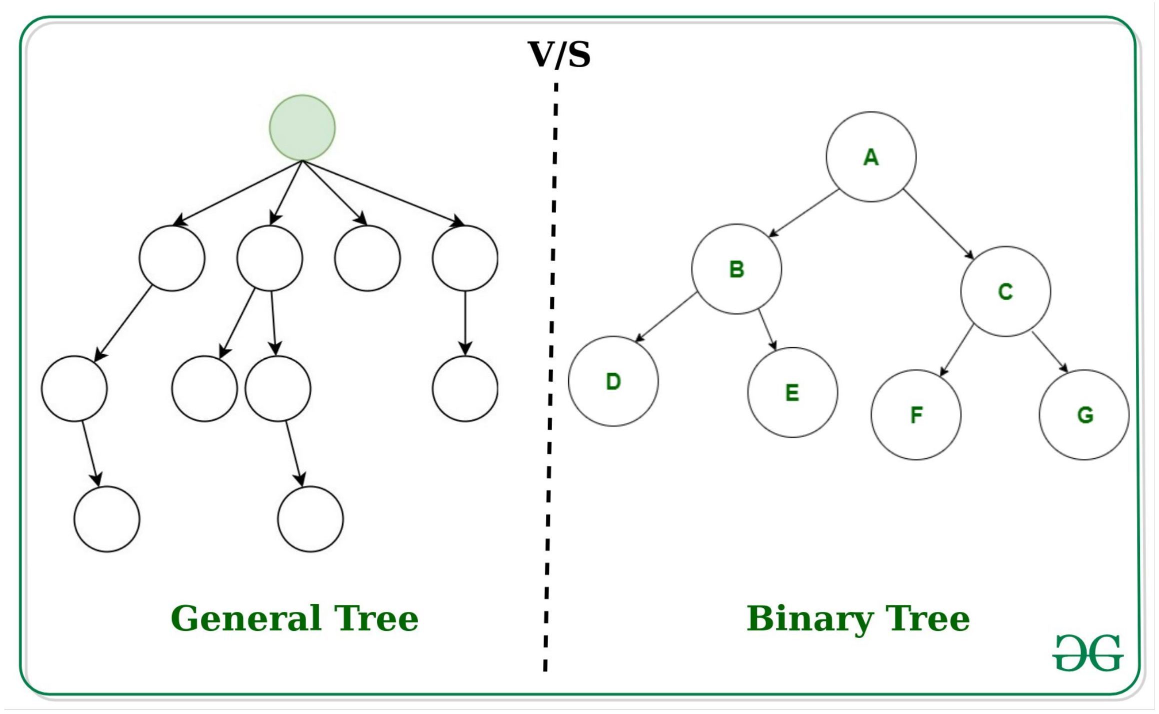
Linked List: Could alternately used to store items. In linked list space to store items is created as it needed and destroyed when space no longer required to store items. Hence linked list is a dynamic data structure space acquire only when need.



Types of Data Structure

Tree: is a non-linear data structure which is mainly used to represent data containing a hierarchical relationship between elements.

Binary Tree: A binary tree is a tree such that every node has at most 2 child and each node is labeled as either left of right child.

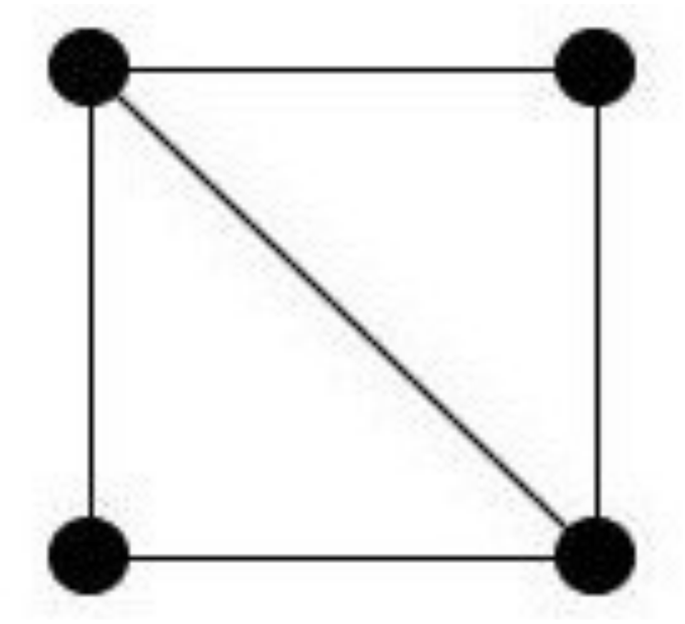


Types of Data Structure

Graph: It is a set of items connected by edges. Each item is called a vertex or node. Trees are just like a special kinds of graphs. Graphs are usually represented by G = (V, E), where V is the set vertices and E is the set of Edges.

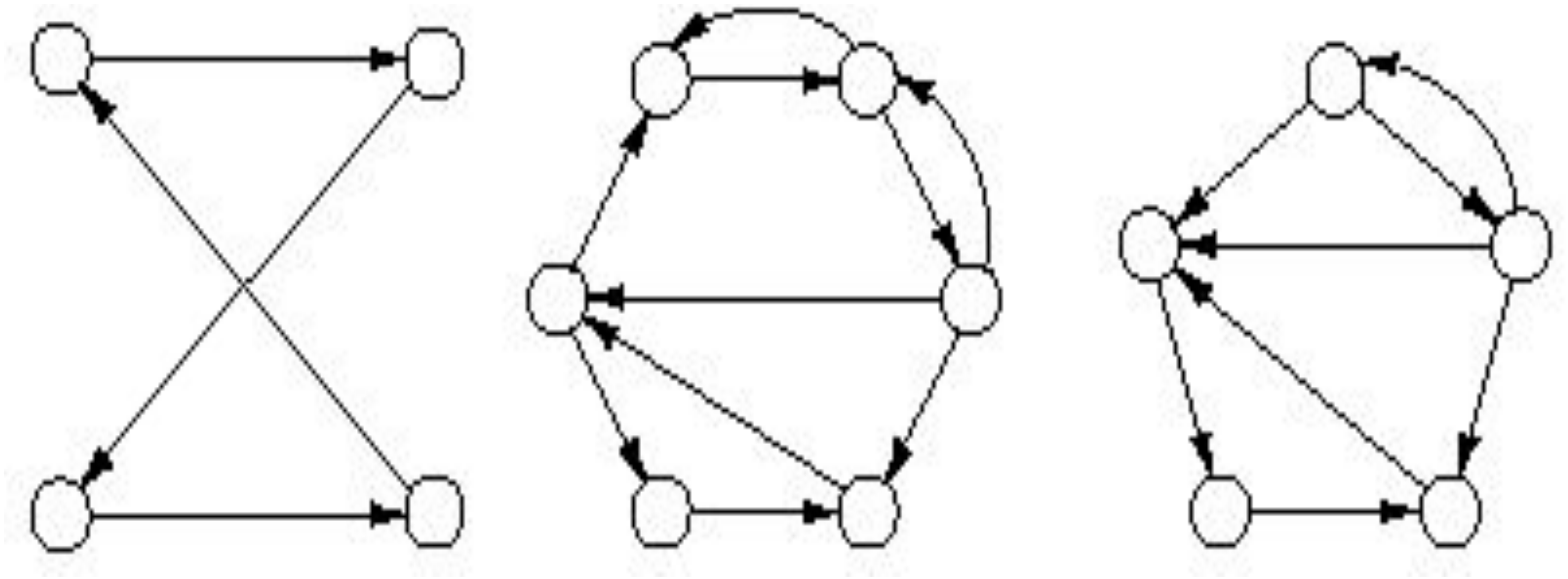
Types of Data Structure

Undirected Graph: A graph whose edges are unordered pair of vertices. That is each edge connects two vertices. In an undirected graph, direction is not important, if the path is available, it can be traversed in any direction.



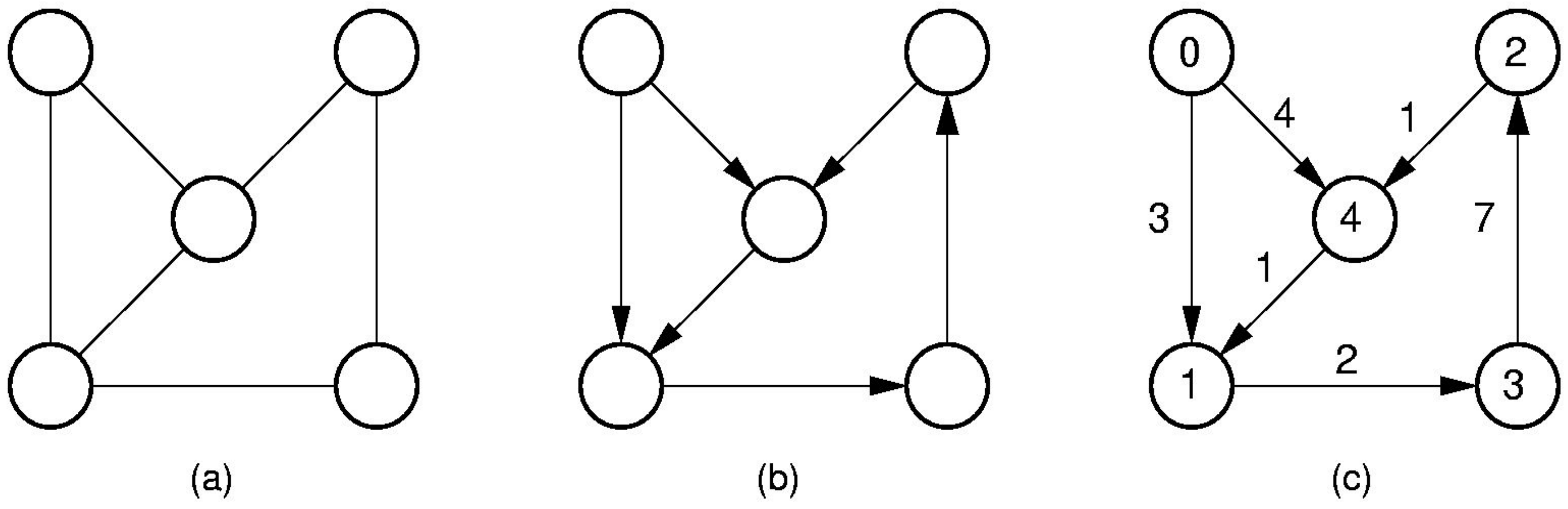
Types of Data Structure

Directed Graph: In directed graph a directional edge connect two node/vertex. If there is one edge from one vertex to other then only this path can be followed.



Types of Data Structure

Weighted Graph: A graph having a weight, or number associated with each edge


0
2
4
1
3
4
1
3
7
1
2
(a)
(b)
(c)