

Advanced Algorithms and Data Structures

Lecture# 03: Concepts of Data Structures

Definition

- Data structure is representation of the logical relationship existing between individual elements of data.
- In other words, a data structure is a way of organizing all data items that considers not only the elements stored but also their relationship to each other.

Introduction

- Data structure affects the design of both structural & functional aspects of a program.

Program = algorithm + Data Structure

- You know that a algorithm is a step by step procedure to solve a particular function.
- That means, algorithm is a set of instruction written to carry out certain tasks & the data structure is the way of organizing the data with their logical relationship retained.

Introduction

- To develop a program of an algorithm, we should select an appropriate data structure for that algorithm.
- Therefore algorithm and its associated data structures form a program.
- The data structure you pick needs to *support* the operations you need.
- Ideally it supports the operations you will use most often in an *efficient* manner.
- Examples of operations:
 - A **List** with operations **insert** and **delete**
 - A **Stack** with operations **push** and **pop**

Terminology

- Abstract Data Type (ADT)
 - Mathematical description of an object with set of operations on the object. Useful building block.
- Algorithm
 - A high level, language independent, description of a step-by-step process.
- Data structure
 - A specific family of algorithms for implementing an abstract data type.
- Implementation of data structure
 - A specific implementation in a specific language

Concepts vs. Mechanisms

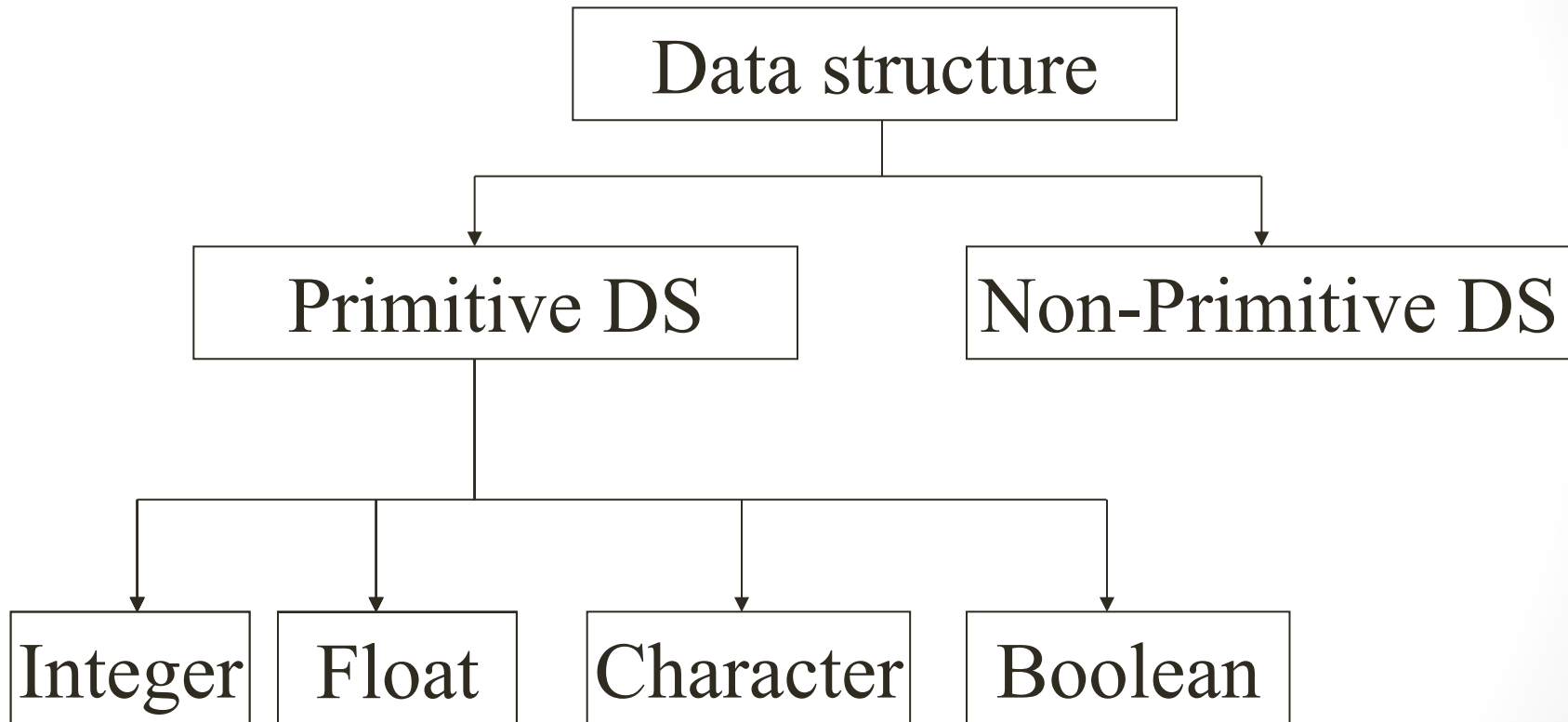
- Abstract
- Pseudocode
- Algorithm
 - A sequence of high-level, language independent operations, which may act upon an abstracted view of data.
- Abstract Data Type (ADT)
 - A mathematical description of an object and the set of operations on the object.

- Concrete
- Specific programming language
- Program
 - A sequence of operations in a specific programming language, which may act upon real data in the form of numbers, images, sound, etc.
- Data structure
 - A specific way in which a program's data is represented, which reflects the programmer's design choices/goals.

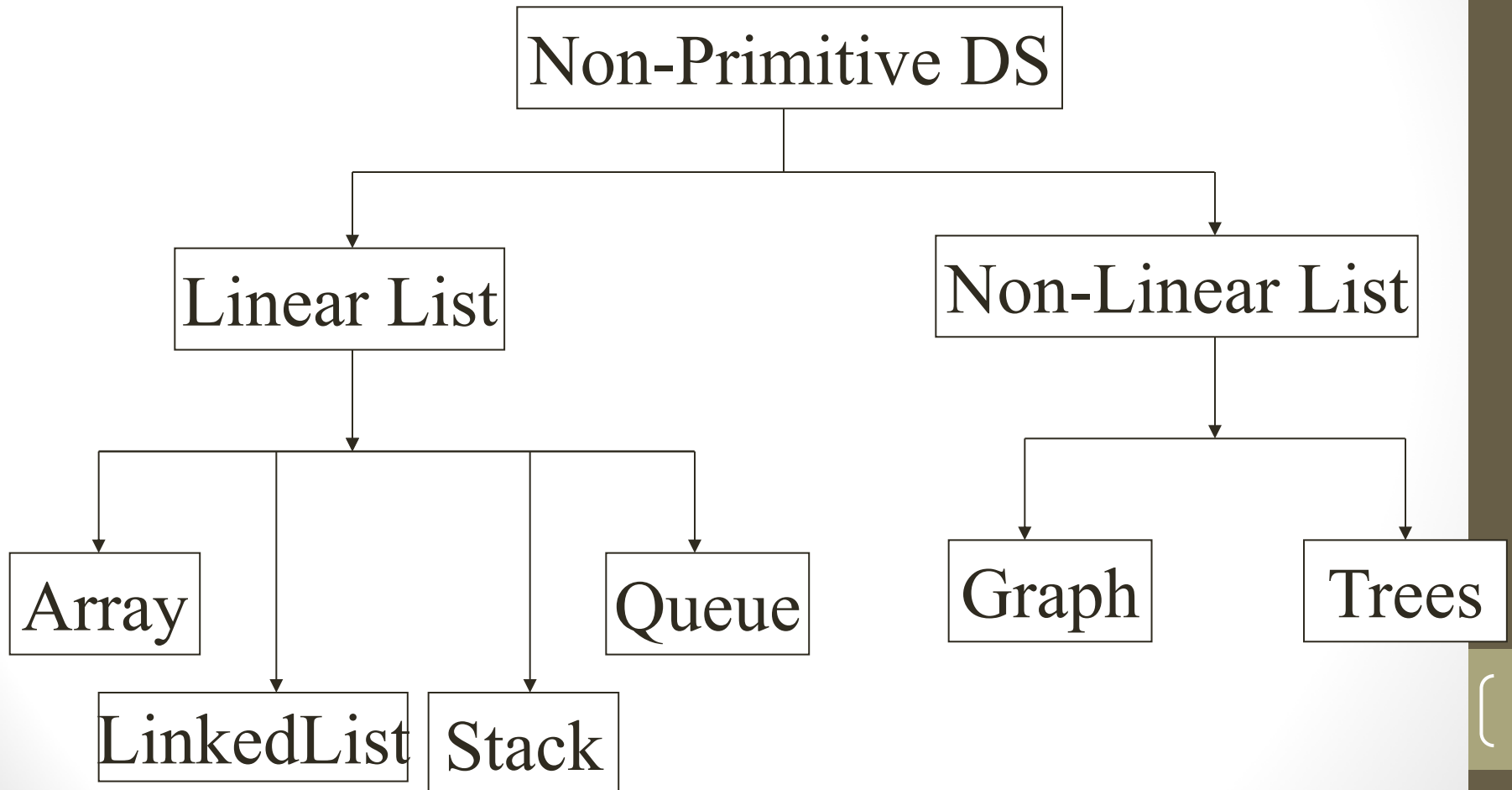
Classification of Data Structure

- Data structure are normally divided into two broad categories:
 - Primitive Data Structure
 - Non-Primitive Data Structure

Classification of Data Structure



Classification of Data Structure



Primitive Data Structure

- There are basic structures and directly operated upon by the machine instructions.
- In general, there are different representation on different computers.
- Integer, Floating-point number, Character constants, string constants, Boolean, pointers etc, fall in this category.

Non-Primitive Data Structure

- There are more sophisticated data structures.
- These are derived from the primitive data structures.
- The non-primitive data structures emphasize on structuring of a group of homogeneous (same type) or heterogeneous (different type) data items.

Non-Primitive Data Structure

- Lists, Stack, Queue, Tree, Graph are example of non-primitive data structures.
- The design of an efficient data structure must take operations to be performed on the data structure.

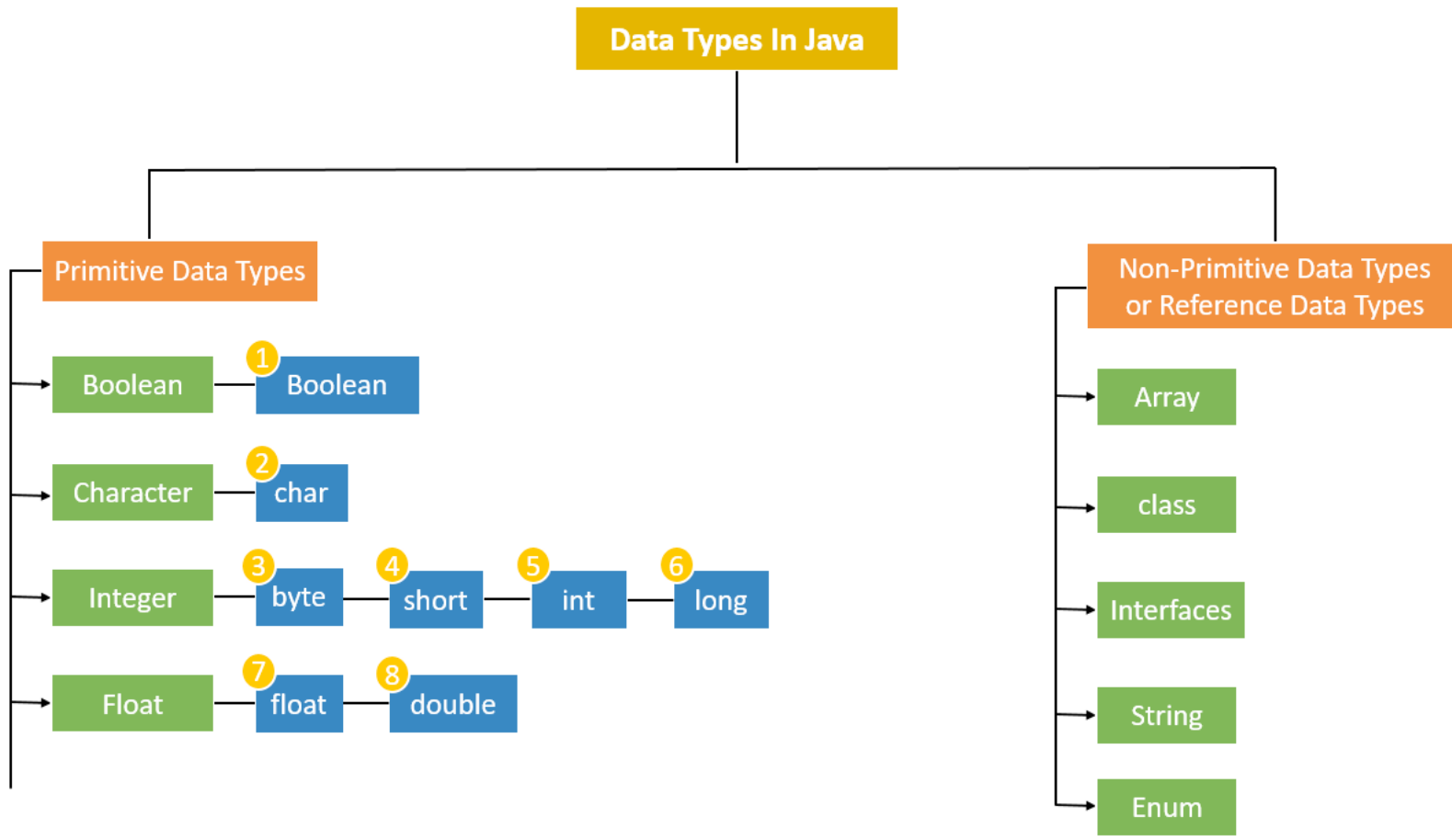
Non-Primitive Data Structure

- The most commonly used operation on data structure are broadly categorized into following types:
 - Create
 - Selection
 - Updating
 - Searching
 - Sorting
 - Merging
 - Destroy or Delete

Different between them

- A primitive data structure is generally a basic structure that is usually built into the language, such as an integer, a float.
- A non-primitive data structure is built out of primitive data structures linked together in meaningful ways, such as a or a linked-list, binary search tree, AVL Tree, graph etc.

Data Types



Data types in Java (Source: Crunchify).

Primitive Data Types

Type	Size (in bits)	Range
byte	8	-128 to 127
short	16	-32,768 to 32,767
int	32	-2^{31} to $2^{31}-1$
long	64	-2^{63} to $2^{63}-1$
float	32	1.4e-045 to 3.4e+038
double	64	4.9e-324 to 1.8e+308
char	16	0 to 65,535
boolean	1	true or false

Abstract Data Types

- ADT users are NOT concerned with how the task is done but rather what it can do.
- An abstract data type is a **data declaration** packaged together with the **operations** that are meaningful for the data type.
- We **encapsulate** the data and the operations on the data, and then hide them from the user.
- All references to and manipulation of the data in a data structure are handled through defined interfaces to the structure.

Data Structures : Arrays

- An array is defined as a set of finite number of homogeneous elements or same data items.
- It means an array can contain one type of data only, either all integer, all float-point number or all character.

Arrays

- Simply, declaration of array is as follows:

```
int arr[] = new int[10];
```

- Where int specifies the data type or type of elements arrays stores.
- “arr” is the name of array & the number specified inside the square brackets is the number of elements an array can store, this is also called sized or length of array.

Arrays

- Following are some of the concepts to be remembered about arrays:
 - The individual element of an array can be accessed by specifying name of the array, following by index or subscript inside square brackets.
 - The first element of the array has index zero[0]. It means the first element and last element will be specified as: `arr[0]` & `arr[9]` respectively.

Arrays

- The elements of array will always be stored in the consecutive (continues) memory location.
- The number of elements that can be stored in an array, that is the size of array or its length is given by the following equation:

$$(\text{Upperbound}-\text{lowerbound})+1$$

Arrays

- For the above array it would be
 - $(9-0)+1=10$, where 0 is the lower bound of array and 9 is the upper bound of array.
- Array can always be read or written through loop. If we read a one-dimensional array it require one loop for reading and other for writing the array.

Arrays

- For example:

```
int[] arr = new int[] {  
    1,2,3,4,5,6,7,8,9,10 };  
for(i=0;i<arr.length;i++){  
    system.out.println(arr[i]);  
}
```

Arrays

- If we are reading or writing two-dimensional array it would require two loops. And similarly the array of a N dimension would required N loops.
- Some common operation performed on array are:
 - Creation of an array
 - Traversing an array

Arrays

- Insertion of new element
- Deletion of required element
- Modification of an element
- Merging of arrays

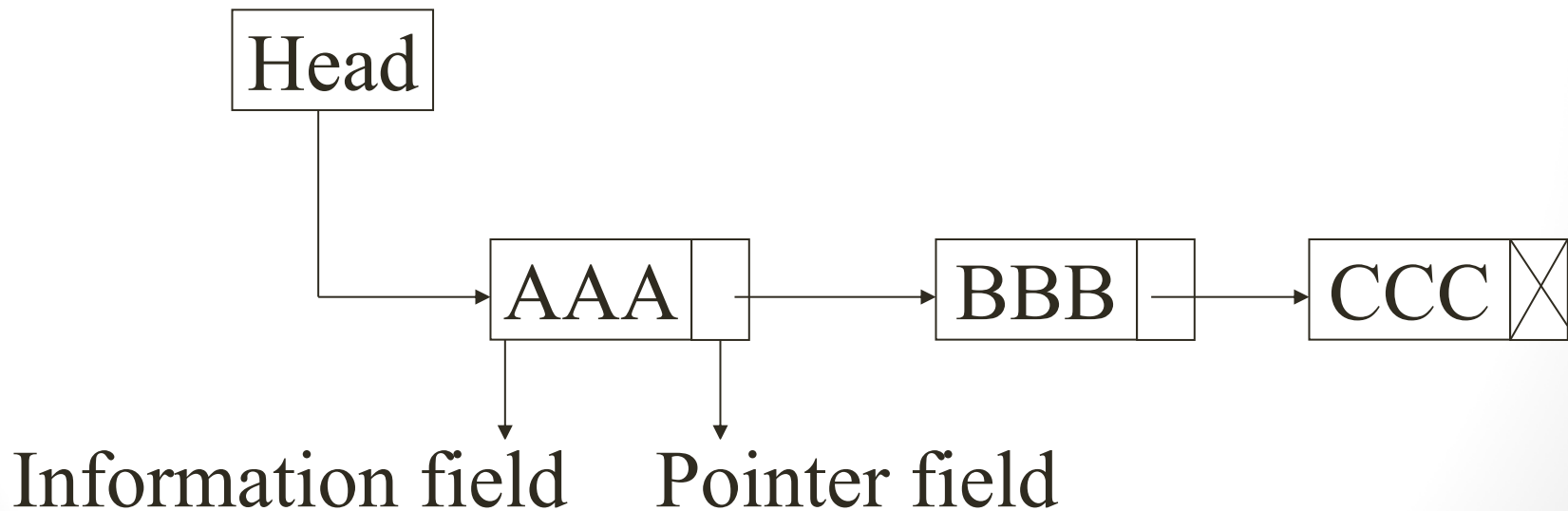
Lists

- A lists (Linear linked list) can be defined as a collection of variable number of data items.
- Lists are the most commonly used non-primitive data structures.
- An element of list must contain at least two fields, one for storing data or information and other for storing address of next element.
- As you know for storing address we have a special data structure of list the address must be pointer type.

Lists

- Technically each such element is referred to as a node, therefore a list can be defined as a collection of nodes as show bellow:

[Linear Liked List]



Lists

- Types of linked lists:
 - Single linked list
 - Doubly linked list
 - Single circular linked list
 - Doubly circular linked list

Stack

- A stack is also an ordered collection of elements like arrays, but it has a special feature that deletion and insertion of elements can be done only from one end called the top of the stack (TOP)
- Due to this property it is also called as last in first out type of data structure (LIFO).

Stack

- It could be thought of just like a stack of plates placed on table in a party, a guest always takes off a fresh plate from the top and the new plates are placed on to the stack at the top.
- It is a non-primitive data structure.
- When an element is inserted into a stack or removed from the stack, its base remains fixed where the top of stack changes.

Stack

- Insertion of element into stack is called PUSH and deletion of element from stack is called POP.
- The bellow show figure how the operations take place on a stack:



Stack

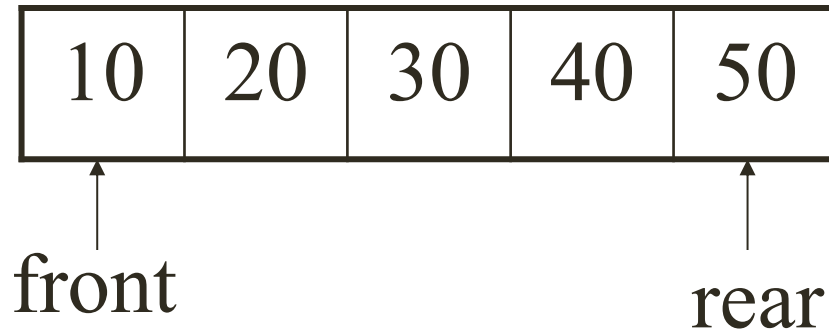
- The stack can be implemented into two ways:
 - Using arrays (Static implementation)
 - Using pointer (Dynamic implementation)

Queue

- Queue are first in first out type of data structure (i.e. FIFO)
- In a queue new elements are added to the queue from one end called REAR end and the element are always removed from other end called the FRONT end.
- The people standing in a railway reservation row are an example of queue.

Queue

- Each new person comes and stands at the end of the row and person getting their reservation confirmed get out of the row from the front end.
- The bellow show figure how the operations take place on a stack:



Queue

- The queue can be implemented into two ways:
 - Using arrays (Static implementation)
 - Using pointer (Dynamic implementation)

Trees

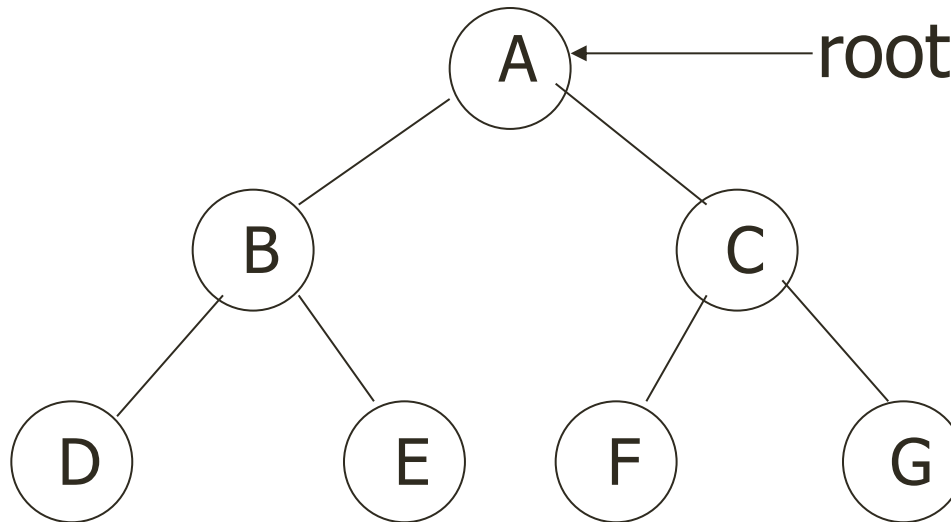
- A tree can be defined as finite set of data items (nodes).
- Tree is non-linear type of data structure in which data items are arranged or stored in a sorted sequence.
- Tree represent the hierarchical relationship between various elements.

Trees

- In trees:
- There is a special data item at the top of hierarchy called the Root of the tree.
- The remaining data items are partitioned into number of mutually exclusive subset, each of which is itself, a tree which is called the sub tree.
- The tree always grows in length towards bottom in data structures, unlike natural trees which grows upwards.

Trees

- The tree structure organizes the data into branches, which related the information.



Graph

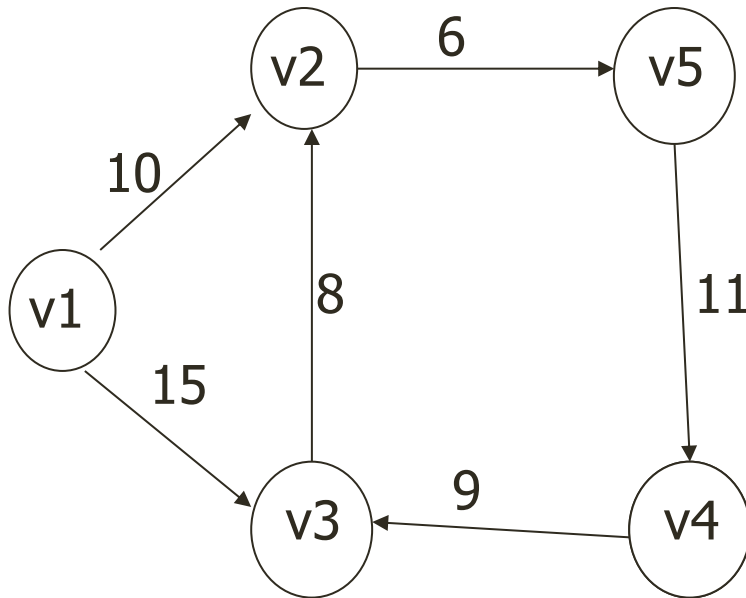
- Graph is a mathematical non-linear data structure capable of representing many kind of physical structures.
- It has found application in Geography, Chemistry and Engineering sciences.
- Definition: A graph $G(V,E)$ is a set of vertices V and a set of edges E .

Graph

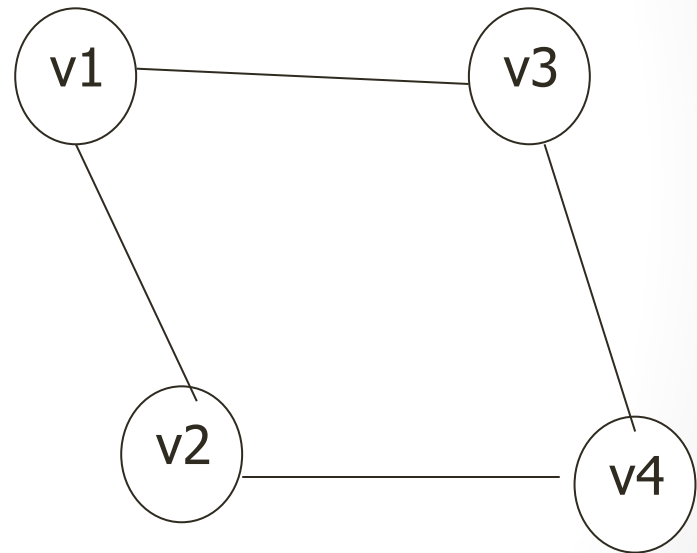
- An edge connects a pair of vertices and many have weight such as length, cost and another measuring instrument for according the graph.
- Vertices on the graph are shown as point or circles and edges are drawn as arcs or line segment.

Graph

- Example of Graph:



[a] Directed & Weighted Graph



[b] Undirected Graph

Graph

- Types of Graphs:
 - Directed graph
 - Undirected graph
 - Simple graph
 - Weighted graph
 - Connected graph
 - Non-connected graph