Data Structures and Algorithms

Dr. Chayan Halder
Assistant Professor,
Ramakrishna Mission Vivekananda Centenary College, Kolkata

Data

- In general, data is any set of characters that is gathered and translated for some purpose, mostly for analysis. It can be any character, including text and numbers, pictures, sound, or video etc.
- If data is not put into context, it doesn't do anything to a human or computer. Within a computer's storage, data is nothing but collection of numbers represented as bytes that are in turn composed of bits.

Data Structure

- 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.

Algorithm

- An algorithm is a step by step recipe for solving an instance of a problem.
- Every single procedure that a computer performs is an outcome of some sort of algorithm.
- An algorithm is a precise procedure for solving a problem in finite number of steps.
- An algorithm states the actions to be executed and the order in which these actions are to be executed.
- An algorithm is a well ordered collection of clear and simple instructions of definite and effectively computable operations that when executed produces a result and stops executing at some point in a finite amount of time rather than just going on and on infinitely.

Properties of Algorithm

An algorithm need to possess these following properties:

- It must be correct.
- It must be composed of a series of concrete steps.
- There can be no ambiguity as to which step will be performed next.
- It must be composed of a finite number of steps.
- It must terminate.
- It takes zero or more inputs.
- It should be efficient and flexible.
- It should use less memory space as much as possible.
- It results in one or more outputs.

Computer Programs

- A computer program is a series of instructions to carry out a particular task written in a language that a computer can understand.
- The process of preparing and feeding the instructions into the computer for execution is referred as programming.
- There are a number of features for a good program:
  - Run efficiently and correctly
  - Have a user friendly interface
  - Be easy to read and understand
  - Be easy to debug
  - Be easy to modify
  - Be easy to maintain

Computer Programs (cont'd)

- Programs consists of two things: Algorithms and data structures.
- A good Program is a combination of both algorithm and a data structure. Program = Algorithms + data structures
- An algorithm is a step by step recipe for solving an instance of a problem.
- A data structure represents the logical relationship that exists between individual elements of data to carry out certain tasks.
- A data structure defines a way of organizing all data items that consider not only the elements stored but also stores the relationship between the elements

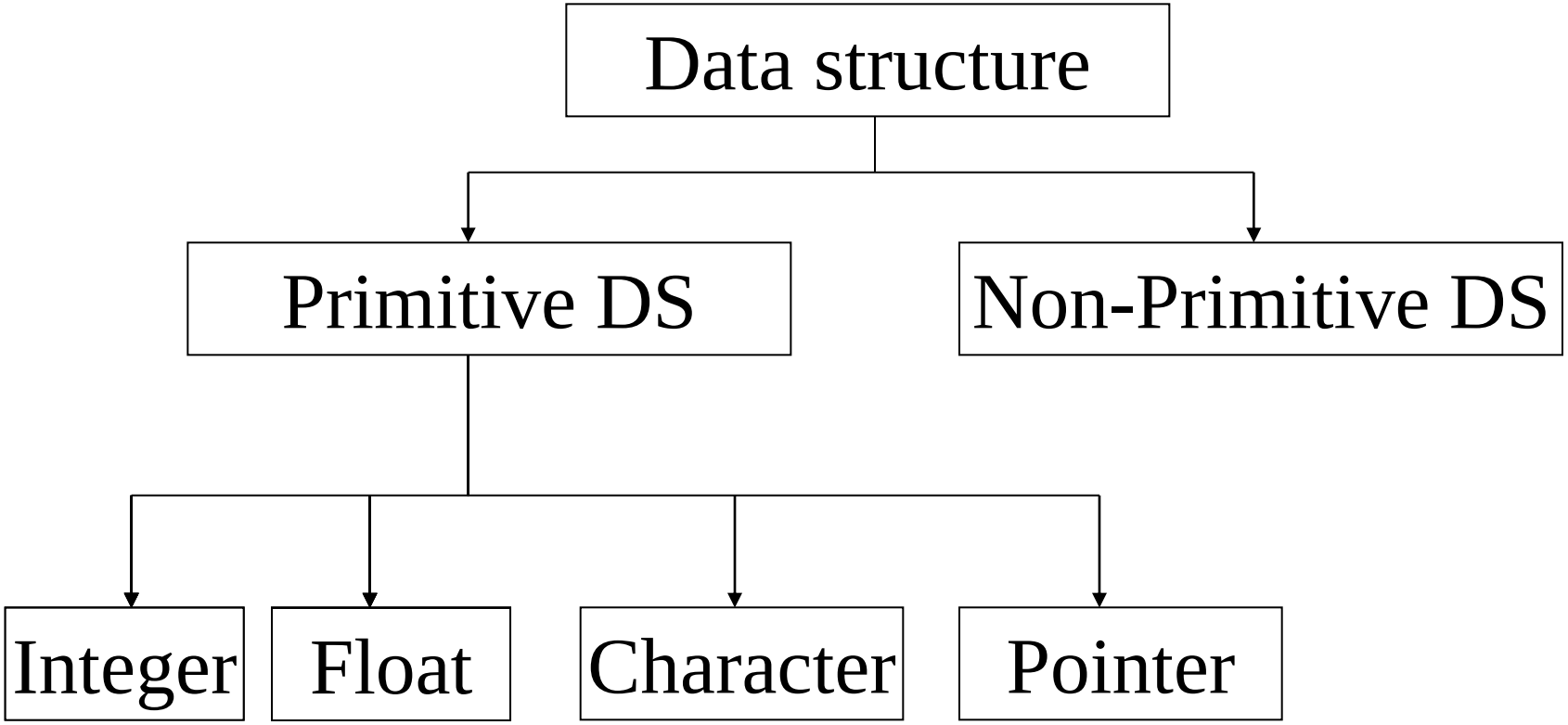
Computer Programs (cont'd)

- 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.
- To develop a program of an algorithm, we should select an appropriate data structure for that algorithm.
- Therefore algorithm and its associated data structures from a program.

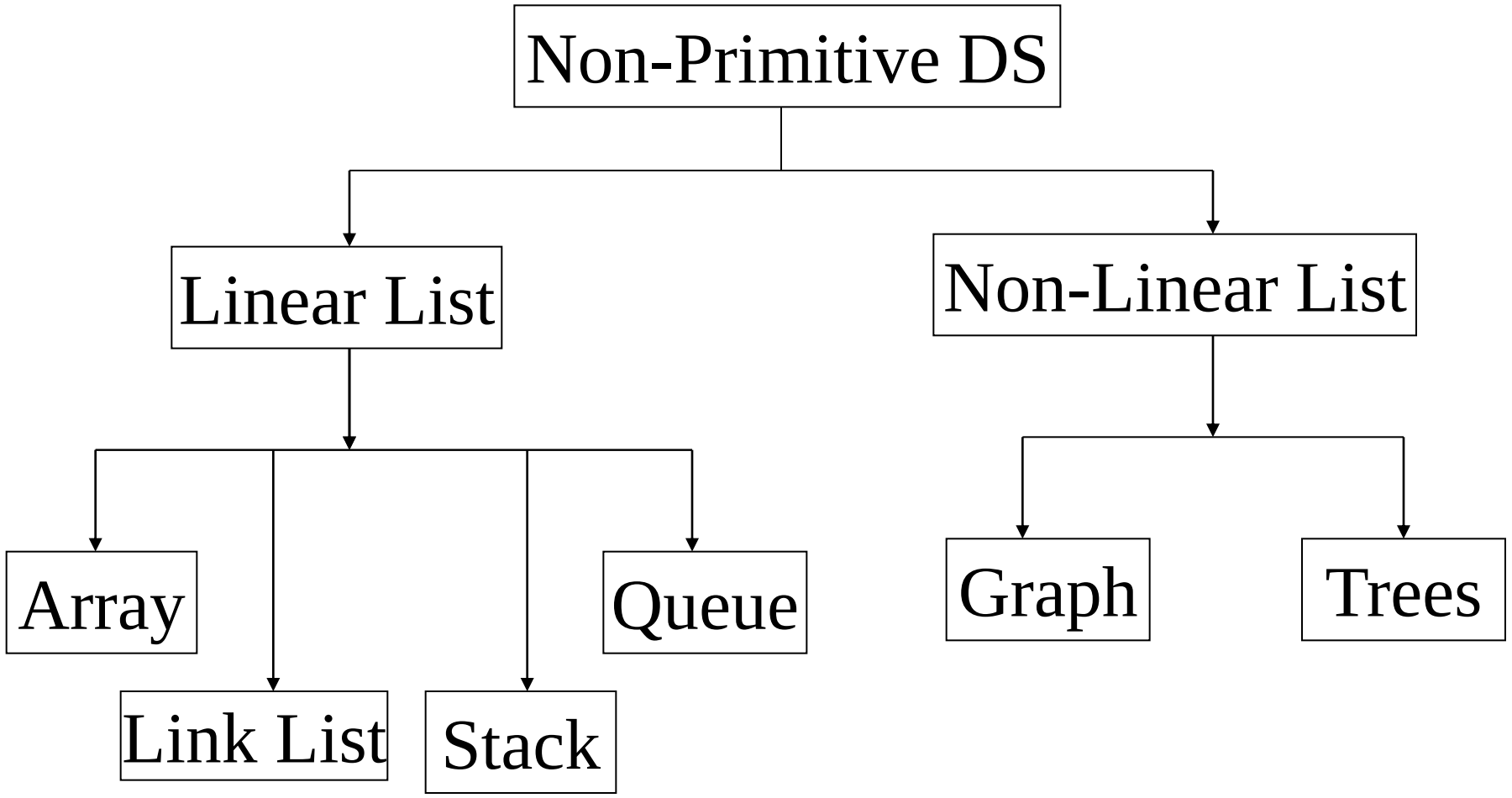
Classification of Data Structure

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

Classification of Data Structure (cont'd)



Classification of Data Structure (cont'd)



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, 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 (cont'd)

- 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 (cont'd)

- 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 array or linked-list, binary search tree, AVL Tree, graph etc.

Description of various 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[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 (cont'd)

- 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 (cont'd)

- 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:

  (Upperbound-lowerbound)+1

Arrays (cont'd)

- 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 (cont'd)

- For example: Reading an array

```
For(i=0;i<=9;i++)
    scanf("%d",&arr[i]);
```

- For example: Writing an array

```
For(i=0;i<=9;i++)
    printf("%d",arr[i]);
```

Arrays (cont'd)

- 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 (cont'd)

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

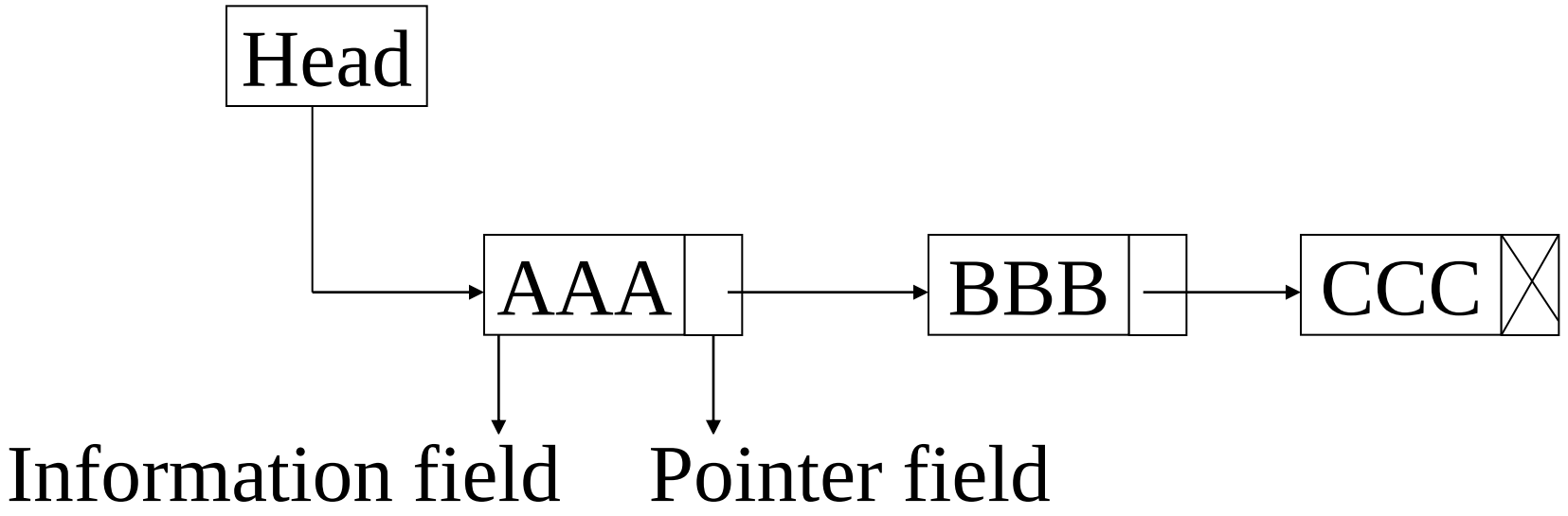
Lists

- 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 (cont'd)

- 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 (cont'd)

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

Stack

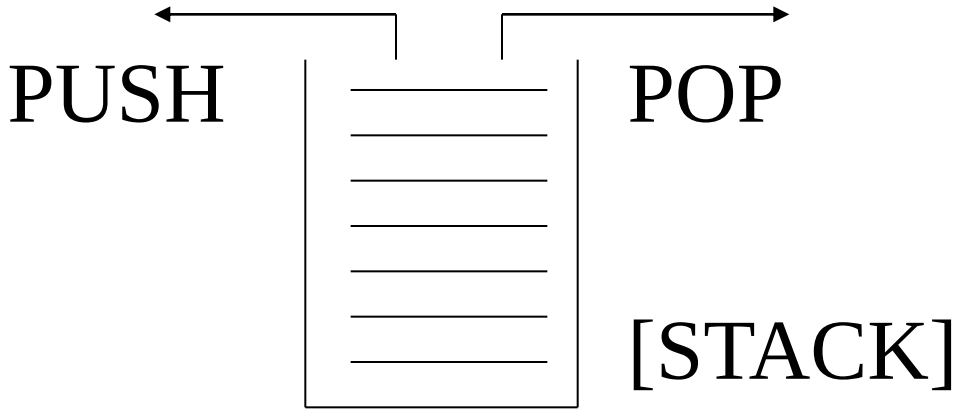
- A stack is also an ordered collection of elements like arrays, but it has a special feature that deletion and insertion of elements can be done only from one end called the top of the stack (TOP)
- Due to this property it is also called as last in first out type of data structure (LIFO).

Stack (cont'd)

- It could be through 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 (cont'd)

- 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 (cont'd)

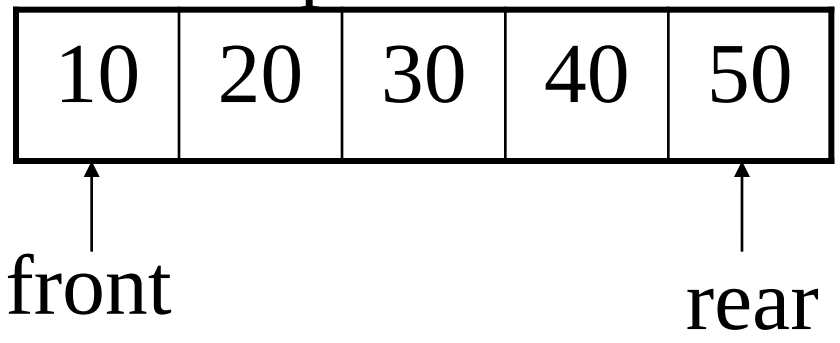
- The stack can be implemented into two ways:
  - Using arrays (Static implementation)
  - Using pointer (Dynamic implementation)

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 (cont'd)

- 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 (cont'd)

- 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.

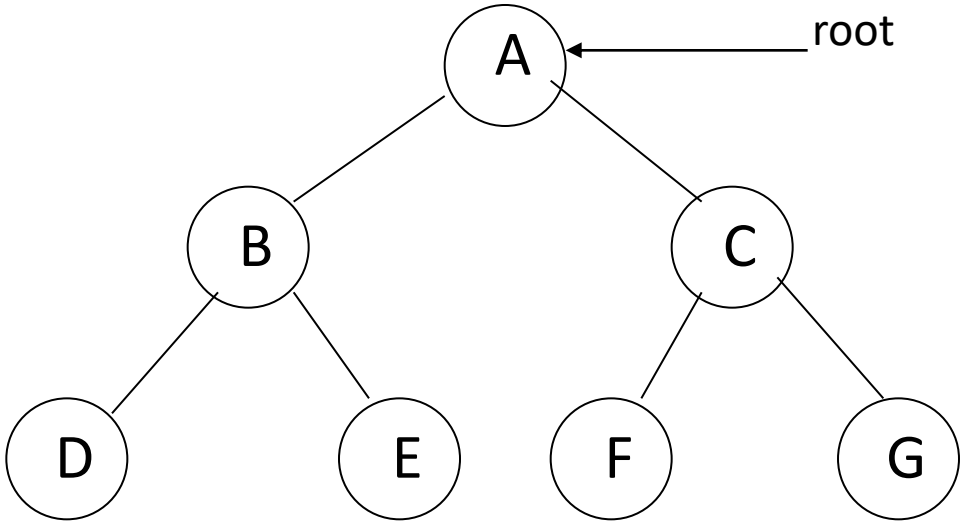
Trees (cont'd)

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 (cont'd)

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



Graph

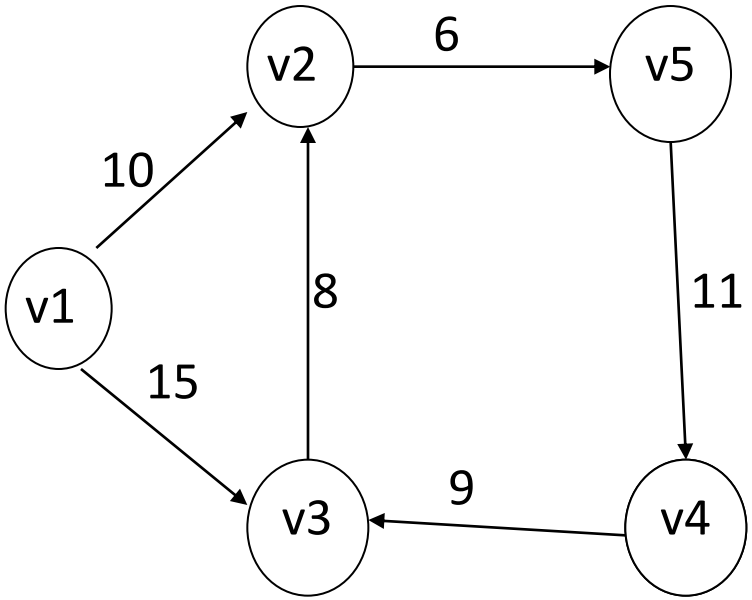
- Graph is a mathematical non-linear data structure capable of representing many kind of physical structures.
- It has found application in Geography, Chemistry and Engineering sciences.
- Definition: A graph G(V,E) is a set of vertices V and a set of edges E.

Graph (cont'd)

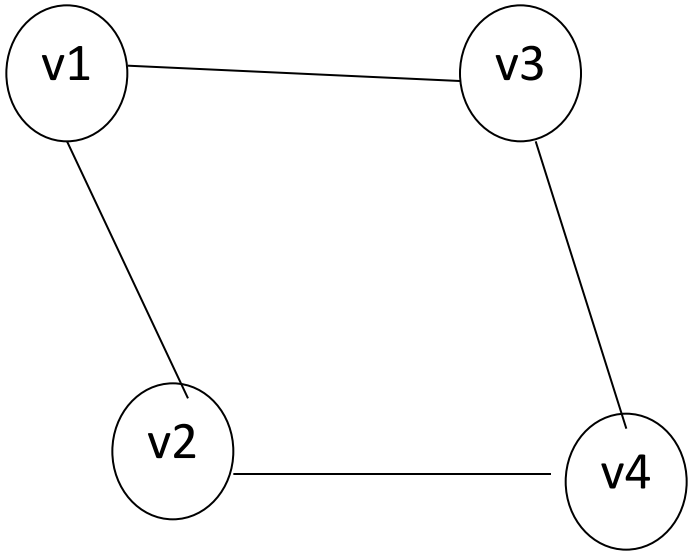
- An edge connects a pair of vertices and many have weight such as length, cost and another measuring instrument for according the graph.
- Vertices on the graph are shown as point or circles and edges are drawn as arcs or line segment.

Graph (cont'd)

- Example of graph:



[a] Directed & Weighted Graph



[b] Undirected Graph

Graph (cont'd)

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