Data Structures

Dr. Anish Soni

Basic Terminology:

- Data – Data are values or set of values.
- Data Item – Data item refers to single unit of values.
- Group Items – Data items that are divided into sub items are called as Group Items.
- Elementary Items – Data items that cannot be divided are called as Elementary Items.
- Attribute and Entity – An entity is that which contains certain attributes or properties, which may be assigned values.
- Entity Set – Entities of similar attributes form an entity set.
- Field – Field is a single elementary unit of information representing an attribute of an entity.
- Record – Record is a collection of field values of a given entity.
- File – File is a collection of records of the entities in a given entity set.

Introduction

- Data Structure is a systematic way to organize data in order to use it efficiently.
- Following terms are the foundation terms of a data structure.
- Interface – Each data structure has an interface. Interface represents the set of operations that a data structure supports. An interface only provides the list of supported operations, type of parameters they can accept and return type of these operations.
- Implementation – Implementation provides the internal representation of a data structure. Implementation also provides the definition of the algorithms used in the operations of the data structure

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.

Introduction

- 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.

Characteristics of a Data Structure:

- Correctness – Data structure implementation should implement its interface correctly.
- Time Complexity – Running time or the execution time of operations of data structure must be as small as possible.
- Space Complexity – Memory usage of a data structure operation should be as little as possible.

Need for Data Structure:

As applications are getting complex and data rich, there are three common problems that applications face now-a-days.

- Data Search − Consider an inventory of 1 million(106) items of a store. If the application is to search an item, it has to search an item in 1 million(106) items every time slowing down the search. As data grows, search will become slower.
- Processor speed − Processor speed although being very high, falls limited if the data grows to billion records.
- Multiple requests − As thousands of users can search data simultaneously on a web server, even the fast server fails while searching the data.

To solve the above-mentioned problems, data structures come to rescue. Data can be organized in a data structure in such a way that all items may not be required to be searched, and the required data can be searched almost instantly

Execution Time Cases:

- There are three cases which are usually used to compare various data structure's execution time in a relative manner.
- Worst Case − This is the scenario where a particular data structure operation takes maximum time it can take. If an operation's worst case time is $f(n)$ then this operation will not take more than $f(n)$ time where $f(n)$ represents function of n.
- Average Case − This is the scenario depicting the average execution time of an operation of a data structure. If an operation takes $f(n)$ time in execution, then m operations will take $mf(n)$ time.
- Best Case − This is the scenario depicting the least possible execution time of an operation of a data structure. If an operation takes $f(n)$ time in execution, then the actual operation may take time as the random number which would be maximum as $f(n)$.

Asymptotic Notations:

- Asymptotic Notations are mathematical tools used to analyze the performance of algorithms by understanding how their efficiency changes as the input size grows.
- These notations provide a brief way to express the behavior of an algorithm's time or space complexity as the input size approaches infinity.
- By using asymptotic notations, we can categorize algorithms based on their worst-case, best-case, or average-case time or space complexities
- Following are the commonly used asymptotic notations to calculate the running time complexity of an algorithm.
- **O Notation (Big Oh)**
- **Ω Notation (Omega)**
- **θ Notation (Theta)**

- Big Oh Notation, O The notation O(n) is the formal way to express the upper bound of an algorithm's running time. It measures the worst case time complexity or the longest amount of time an algorithm can possibly take to complete.

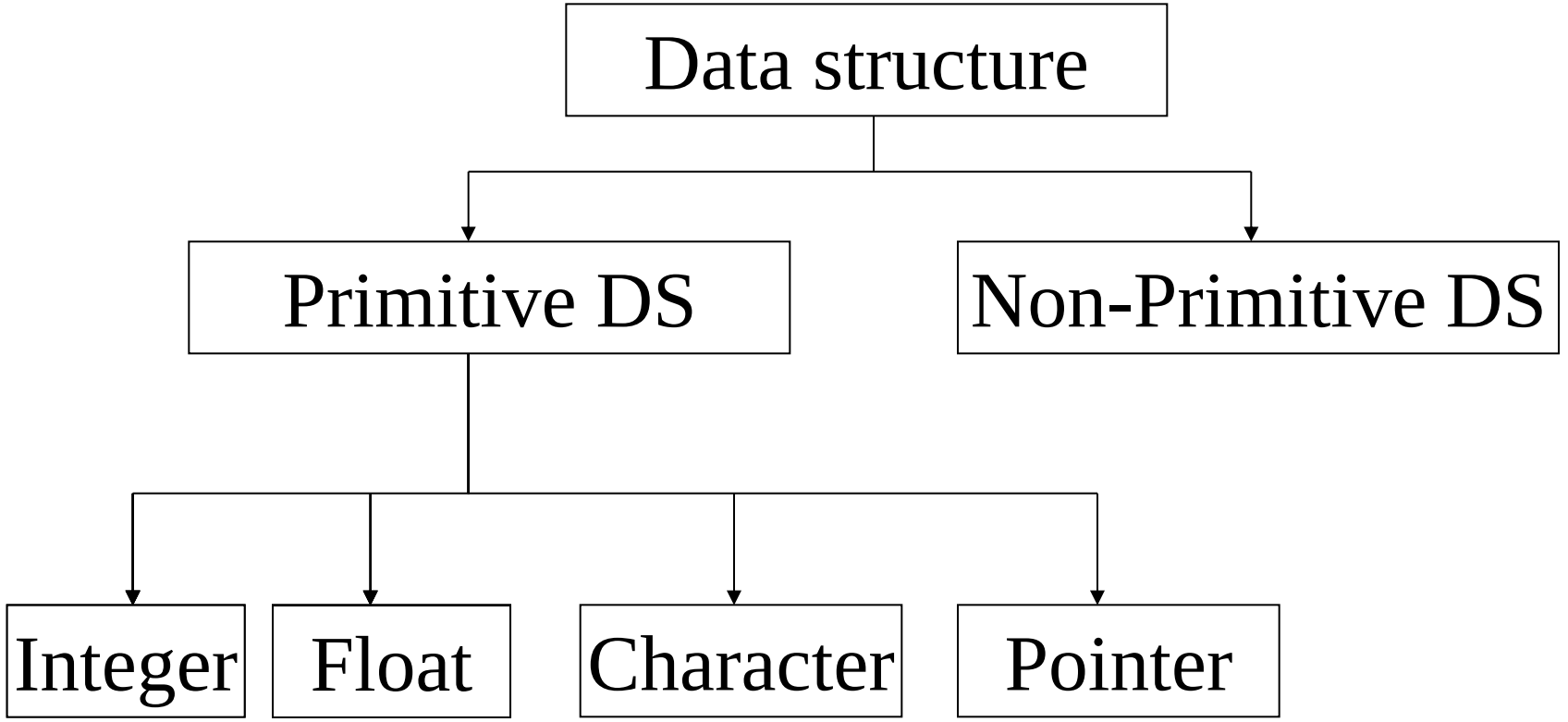
- Omega Notation, Ω The notation Ω(n) is the formal way to express the lower bound of an algorithm's running time. It measures the best case time complexity or the best amount of time an algorithm can possibly take to complete.

- Theta Notation, θ The notation θ(n) is the formal way to express both the lower bound and the upper bound of an algorithm's running time. ie the average case.

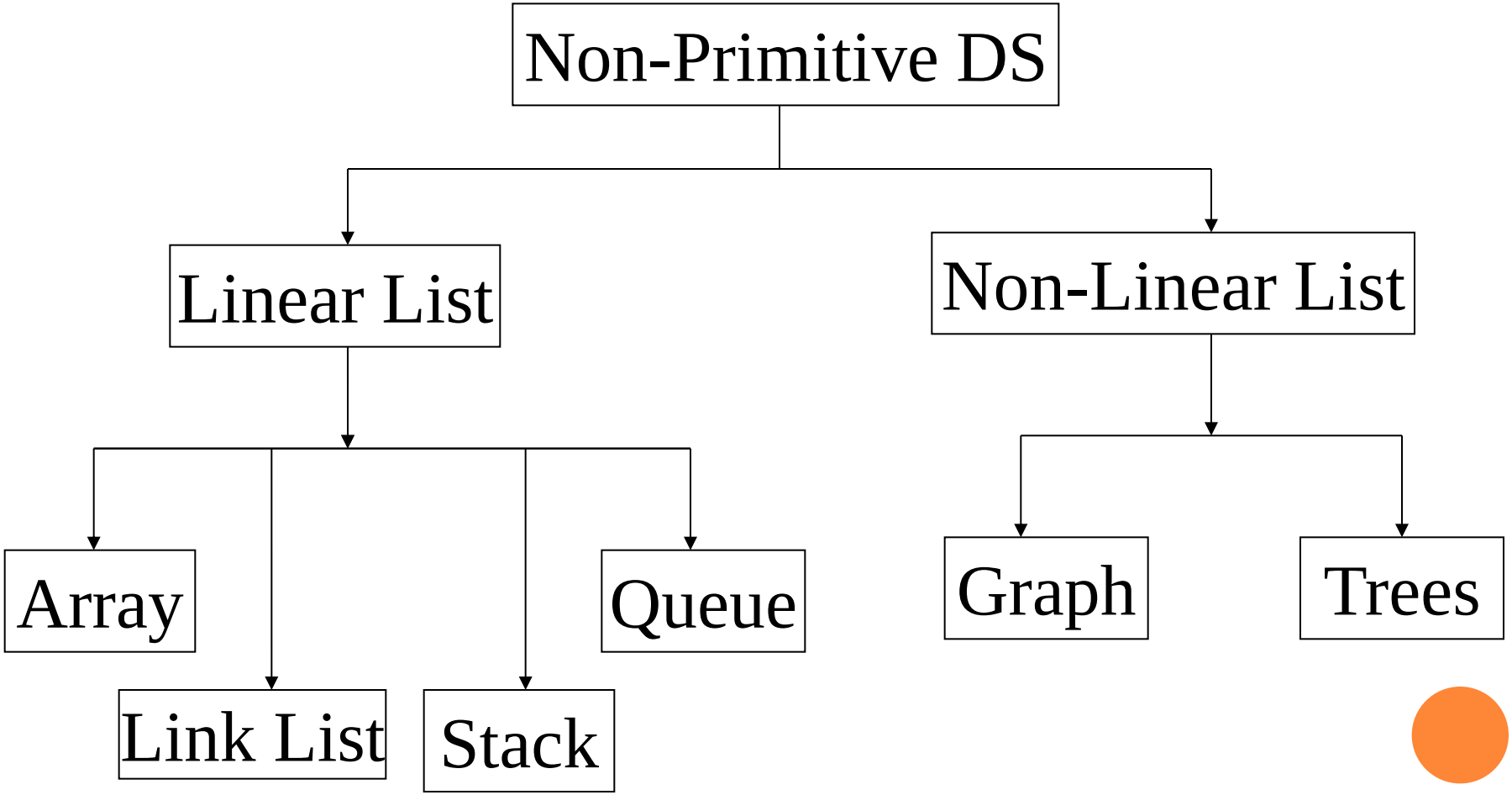
Classification of Data Structure

- 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, 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

Difference 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 a 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

- 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:

  (Upperbound-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: 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

- 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

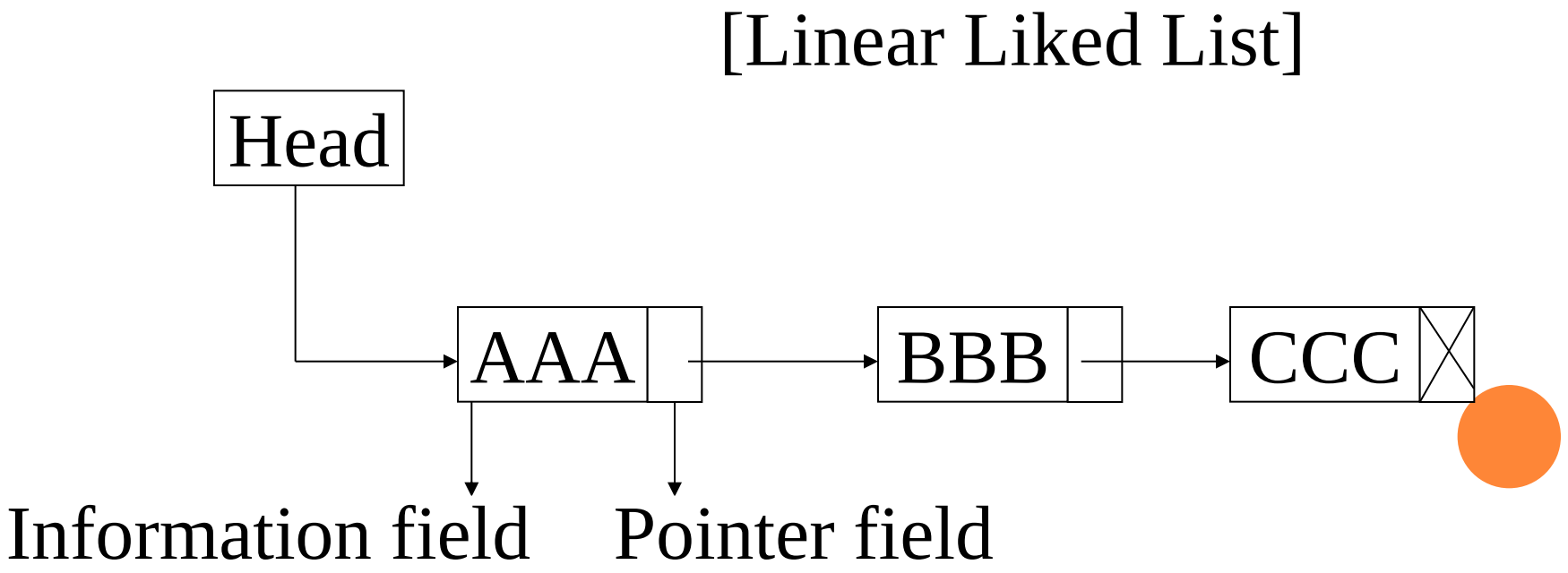
- Insertion of new element
- Deletion of required element
- Modification of an element
- Merging of arrays

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:



Lists

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

Stack

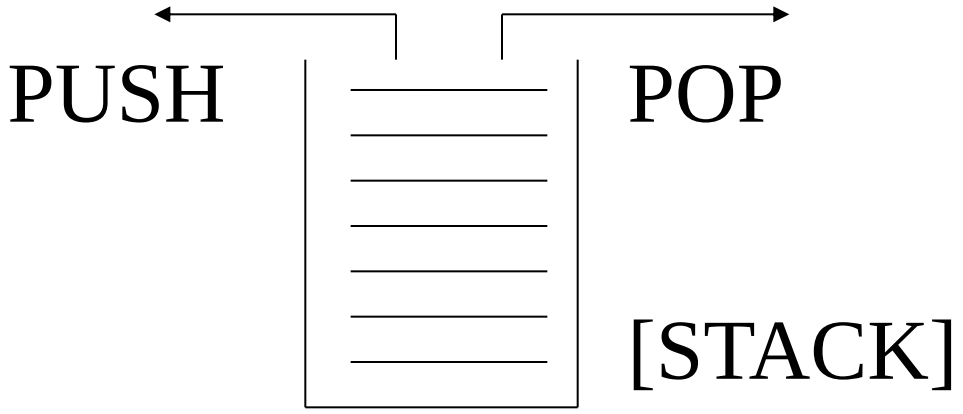
- A stack is also an ordered collection of elements like arrays, but it has a special feature that deletion and insertion of elements can be done only from one end called the top of the stack (TOP)
- Due to this property it is also called as last in first out type of data structure (LIFO).

Stack

- It could be 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

- 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

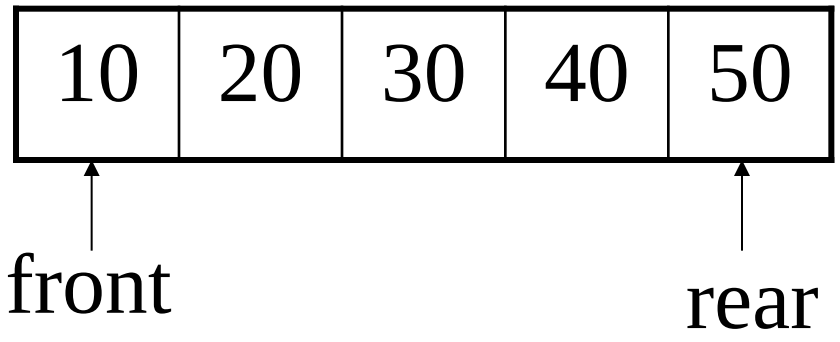
- The stack can be implemented into two ways:
  - Using arrays (Static implementation)
  - Using pointer (Dynamic implementation)

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

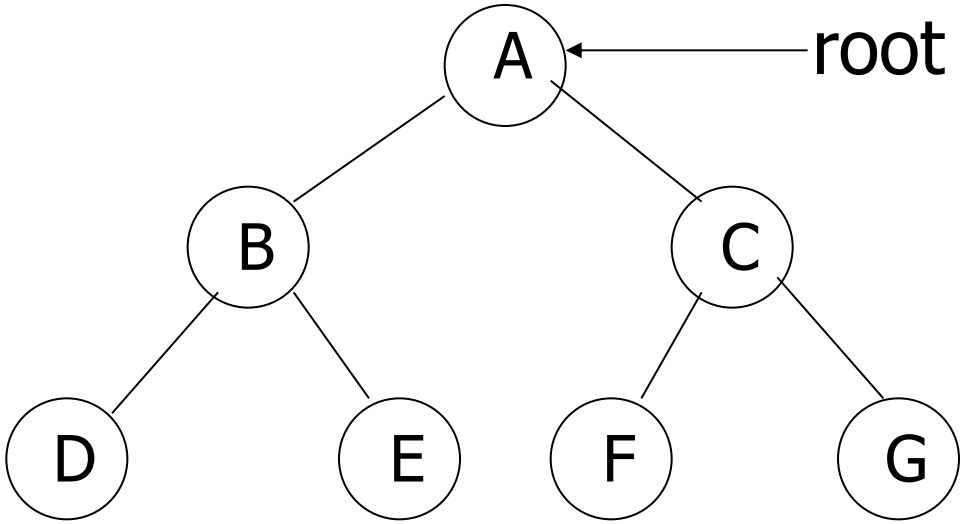
- A tree can be defined as finite set of data items (nodes).
- Tree is non-linear type of data structure in which data items are arranged or stored in a sorted sequence.
- Tree represent the hierarchical relationship between various elements.

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

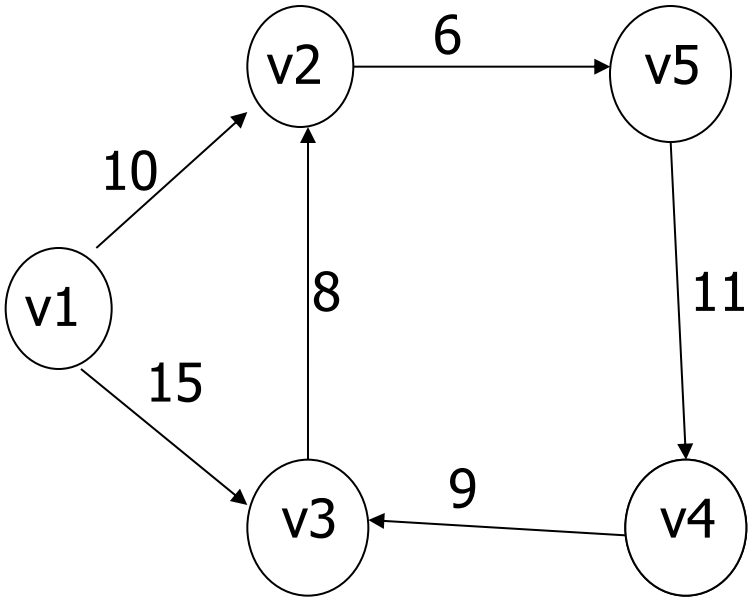
- Graph is a mathematical non-linear data structure capable of representing many kind of physical structures.
- It has found application in Geography, Chemistry and Engineering sciences.
- Definition: A graph G(V,E) is a set of vertices V and a set of edges E.

Graph

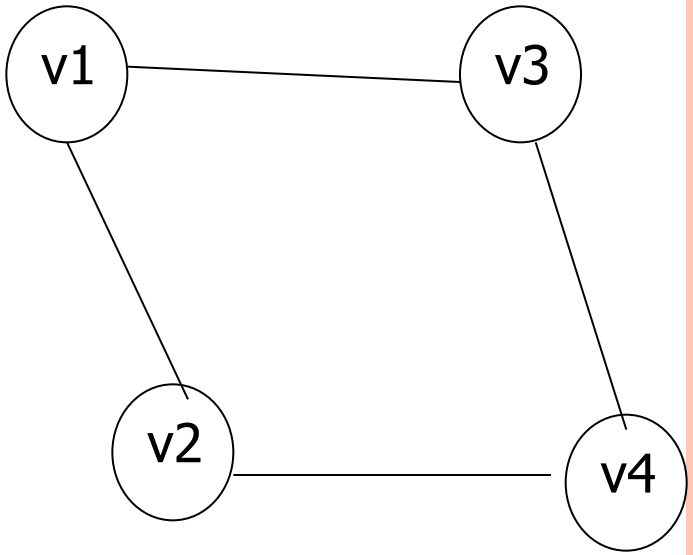
- An edge connects a pair of vertices and many have weight such as length, cost and another measuring instrument for according the graph.
- Vertices on the graph are shown as point or circles and edges are drawn as arcs or line segment.

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