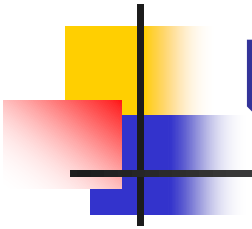
Data Structure and Algorithms

Course Teacher: Prof. Dr. Mostofa Kamal Nasir [MKN1]

E-mail: kamal.mostofa@gmail.com
mostafa.nasir@northsouth.edu



❑RECOMMENDED BOOKS

- C++ plus data Structures, Fifth Edition by Nell Dale
- Data Structures with C++ Schaum's Outline Series

Data and Information

- **Data** can be defined as a representation of facts and concepts by values.
- **Data** is collection of **raw facts**.

Data is represented with the help of characters such as alphabets (A-Z, a-z), digits (0-9) or special characters (+,-,/,*,<,>,= etc.)

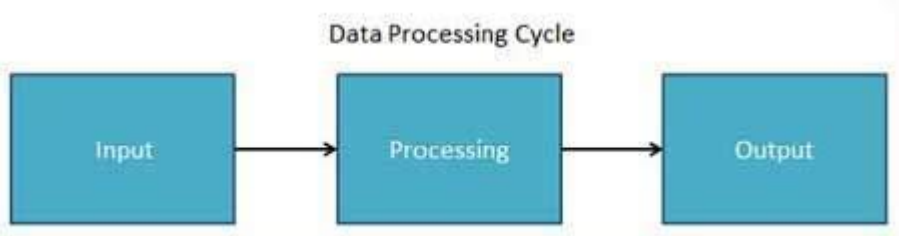
- **Data structure** is representation of the logical relationship existing between individual elements of data.

Information

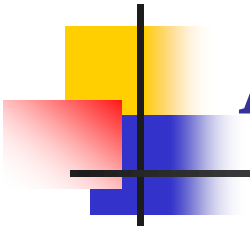
- **Information** is organized or classified data, which has some meaningful values for the receiver. Information is the processed data on which decisions and actions are based.

- The processed data must qualify for the following characteristics –
- **Timely** – Information should be available when required.
- **Accuracy** – Information should be accurate.
- **Completeness** – Information should be complete.

Data Processing Cycle



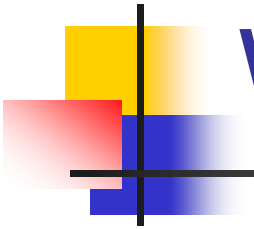
- **Input:** the input data is prepared in some convenient form for processing
- **Processing:** the input data is changed to produce data in a more useful form
- **Output:** the result of the proceeding processing step is collected.



Algorithm

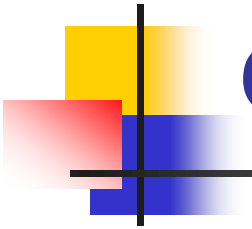
- An Algorithm is a well defined list of steps to solve a problem.
- Data structure is the logical or mathematical relationship of individual elements of data.

- Algorithm + Data Structure= Program



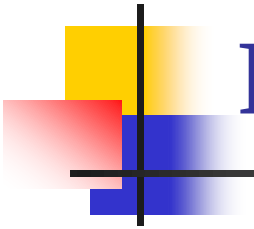
Why Data Structures?

- They are essential ingredients in creating fast and powerful algorithms.
- They help to manage the organize data.
- They make code cleaner and easier to understand.



Classification of Data Structure

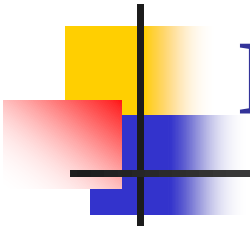
- Two broad categories of data structure are :
  - Primitive Data Structure
  - Non-Primitive Data Structure



Primitive Data Structure

- They are basic structures and directly operated upon by the machine instructions.

- Integer, Floating-point number, Character constants, string constants, pointers etc,



Non-Primitive Data Structure

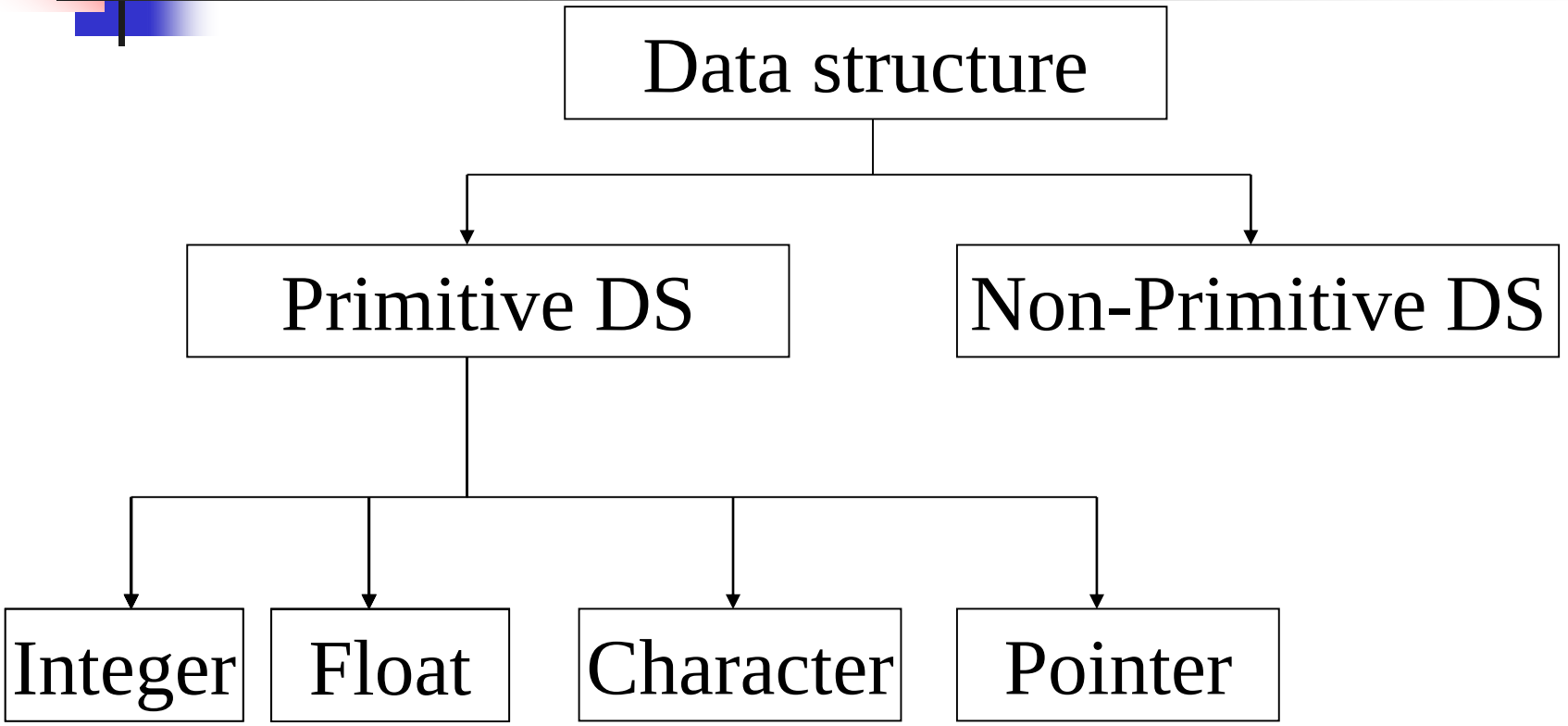
- These are derived from the primitive data structures.
- Example: Array, Lists, Stack, Queue, Tree, Graph

Difference between them

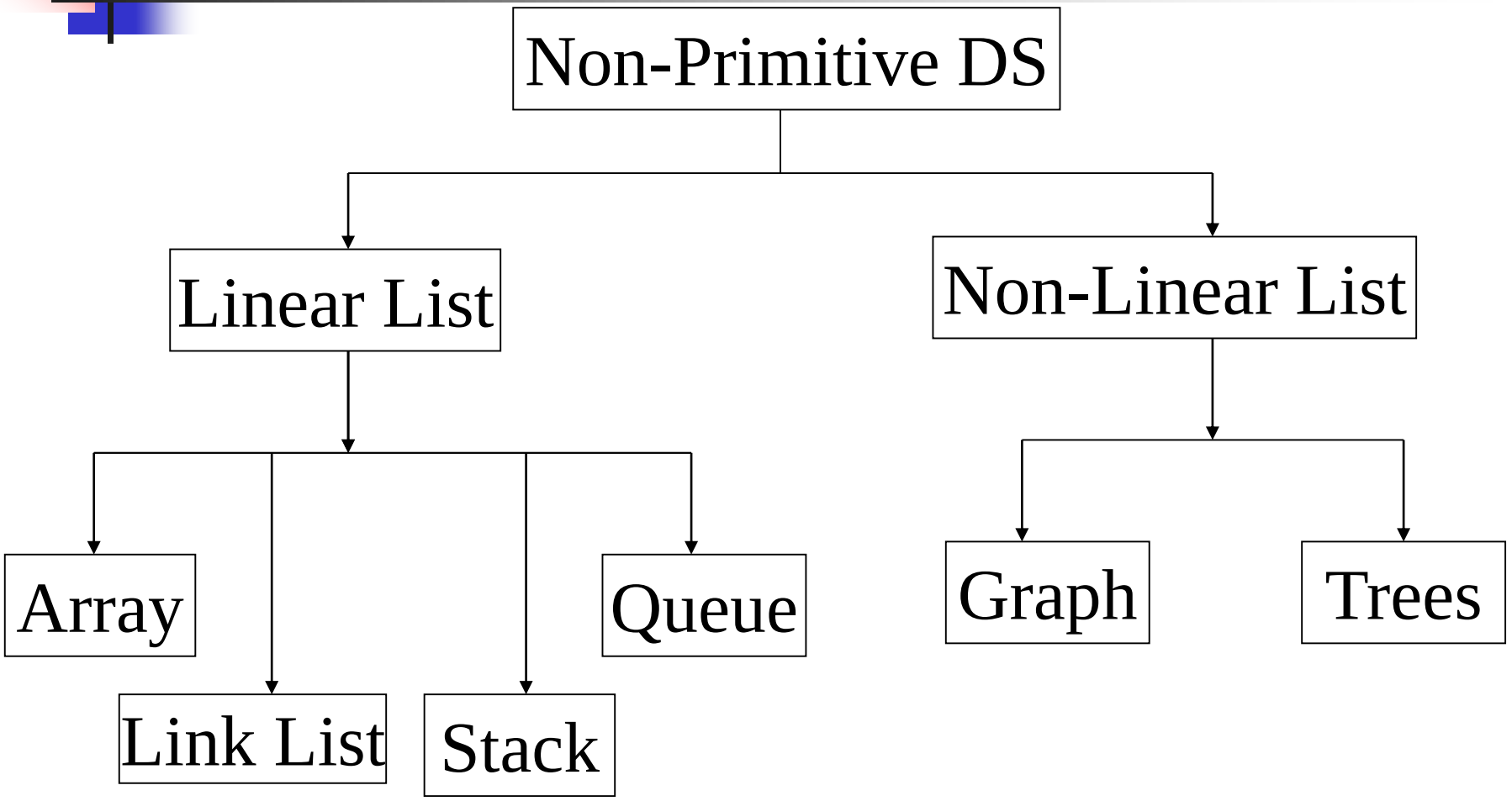
- **A primitive data structure** is generally a basic structure that is usually built into the language, such as an integer, a float.

- **A non-primitive data structure** is built out of primitive data structures linked together in meaningful ways, such as a linked-list, binary search tree, AVL Tree, graph etc.

Classification of Data Structure



Classification of Data Structure



Types of Data

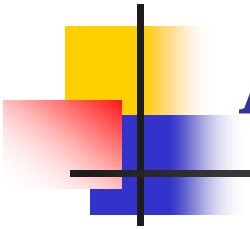
Characterstic	Description
Linear	In Linear data structures, the data items are arranged in a linear sequence. Example: Array
Non-Linear	In Non-Linear data structures, the data items are not in sequence. Example: Tree, Graph
Homogeneous	In homogeneous data structures, all the elements are of same type. Example: Array
Non-Homogeneous	In Non-Homogeneous data structure, the elements may or may not be of the same type. Example: Structures
Static	Static data structures are those whose sizes and structures associated memory locations are fixed, at compile time. Example: Array
Dynamic	Dynamic structures are those which expand or shrink depending upon the program need and its execution. Also, their associated memory locations changes. Example: Linked List created using pointers

Data Structure Operations

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

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.

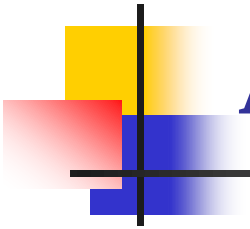


Arrays

- The elements of array will always be stored in the consecutive (continues) memory location.

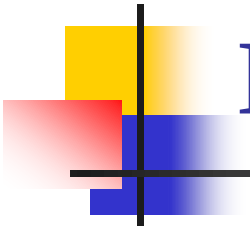
- The number of elements that can be stored in an array, that is the size of array or its length is given by the following equation:

  (Upperbound-lowerbound)+1



Arrays

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



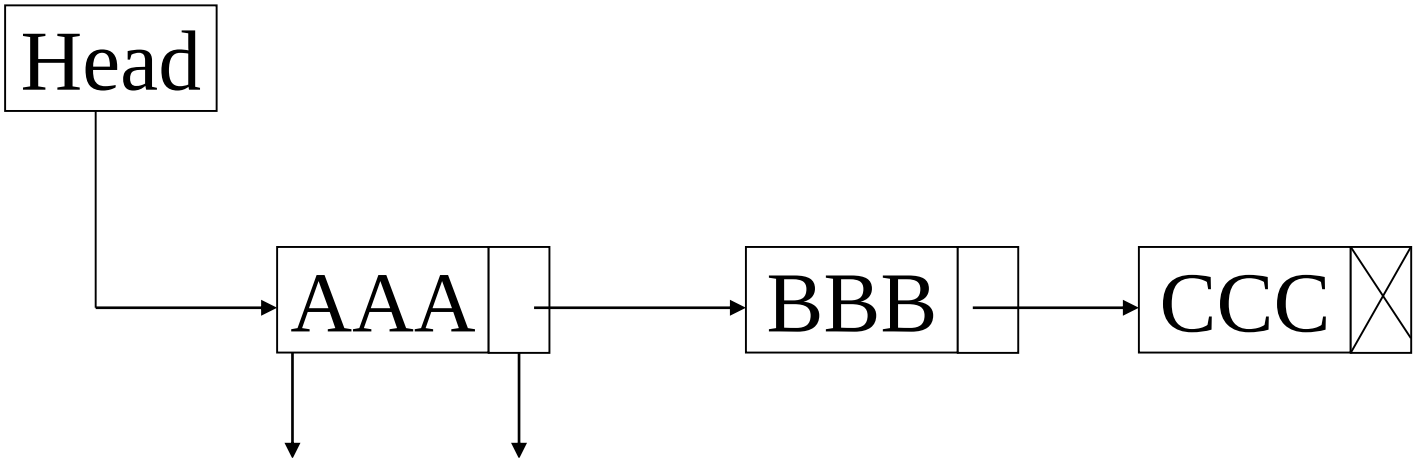
Lists

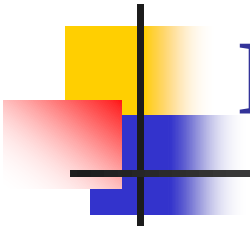
- A lists (Linear linked list) can be defined as a collection of variable number of data items.
- An element of list must contain at least two fields, one for storing data or information and other for storing address of next element.
- For storing address need a special data structure of list that is 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:

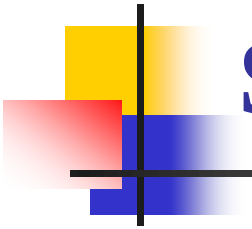
[Linear Liked List]





Lists

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

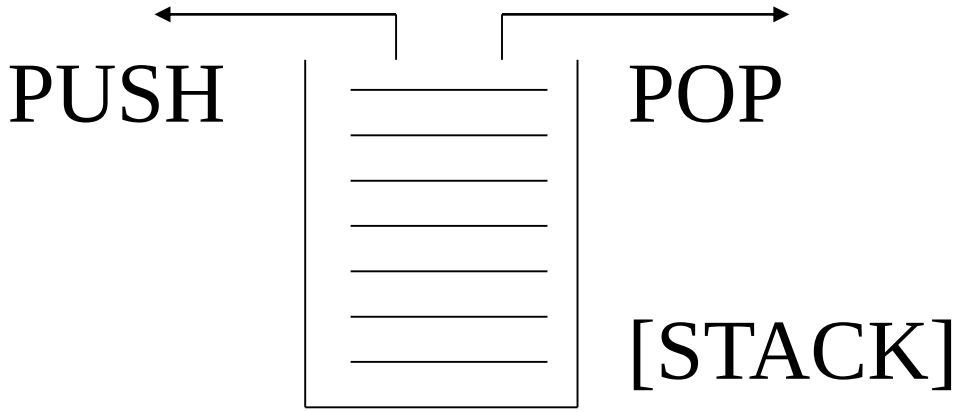


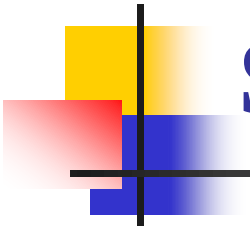
Stack

- 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

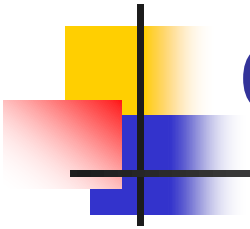
- Insertion of element into stack is called PUSH and deletion of element from stack is called POP.
- The bellow show figure how the operations take place on a stack:





Stack

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

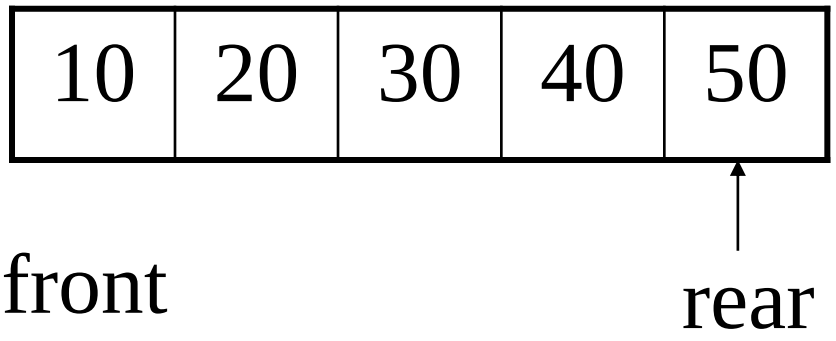


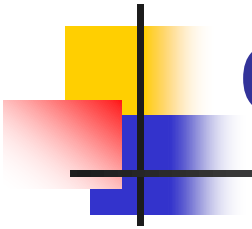
Queue

- 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

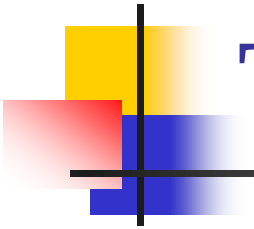
- The bellow show figure how the operations take place on a queue:





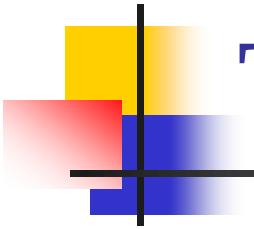
Queue

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



Trees

- Tree is non-linear type of data
- Tree represent the hierarchical relationship between various elements.



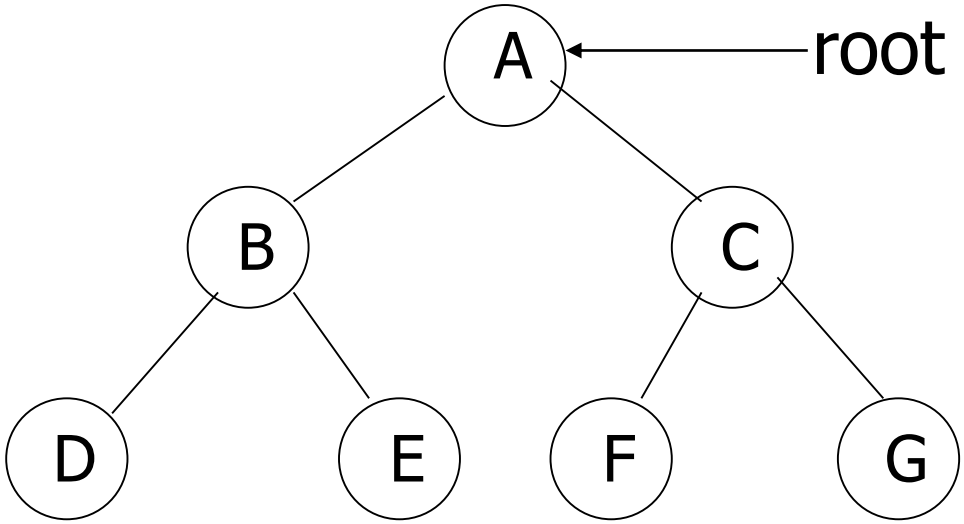
Trees

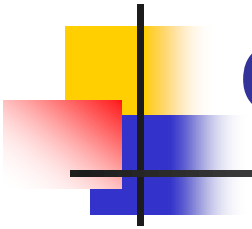
- There is a special data item at the top of hierarchy called the Root of the tree.

- 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.

Trees

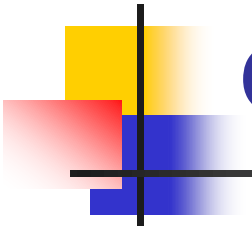
- The tree structure organizes the data into branches, which related the information.





Graph

- Graph is a mathematical non-linear data structure capable of representing many kind of physical structures.
- Definition: A graph G(V,E) is a set of vertices V and a set of edges E.

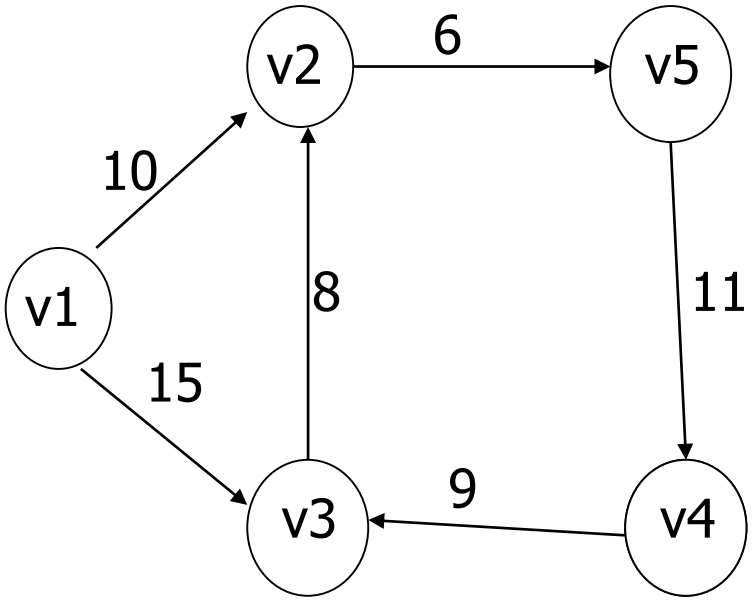


Graph

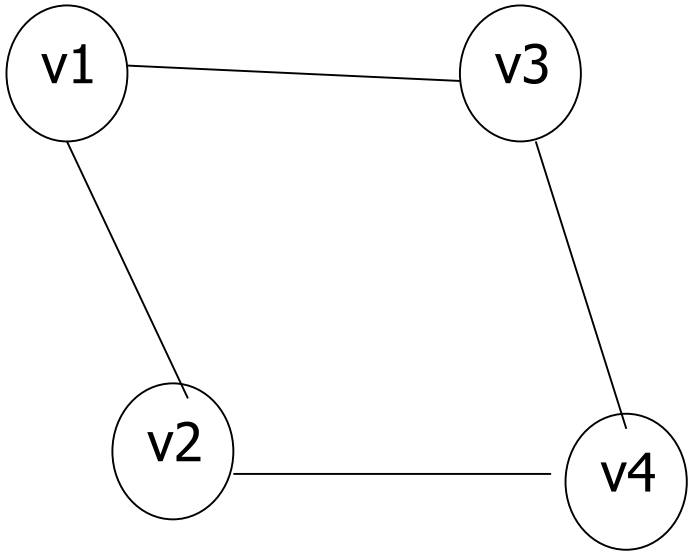
- An edge connects a pair of vertices and many have weight such as length, cost and another measuring instrument for according the graph.
- Vertices on the graph are shown as point or circles and edges are drawn as arcs or line segment.

Graph

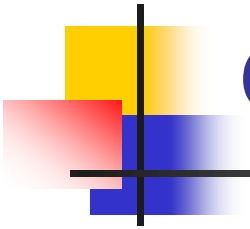
- Example of graph:



[a] Directed & Weighted Graph



[b] Undirected Graph



Graph

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

❑OPERATIONS

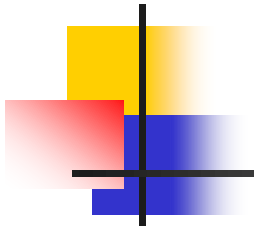
- Data appearing in Data Structure are processed by means of certain operation

- Particular DS one chooses for a given situation depends largely on the frequency with which specific operations are performed

MAJOR OPERATION

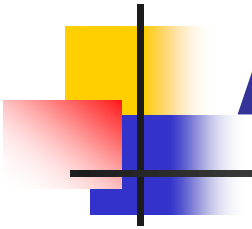
- **Traversing**: Accessing each record exactly once so that certain items in the record may be processed [Also known as Visiting the record]
- **Searching**: Finding the location of the record with a given key value, or finding the locations of all record which satisfy one or more conditions
- **Inserting** : Adding a new record to the structure
- **Deleting** : Removing a record from the structure
- **Sorting**: Arranging a list in some logical order.
- **Merging**: Combing two list in a single list.

Abstract Data Type



- Abstract Data Types (ADT's) are a model used to understand the design of a data structure. Abstract mean an implementation-independent view of the data structure.

- ADTs specify the type of data stored and the operations that support the data

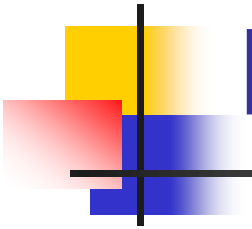


Abstract data type (ADT)

- Abstract data type (ADT) is a specification of a set of data and the set of operations that can be performed on the data. Each operation does a specific task.

Uses of ADT

- 1. It helps to efficiently develop well designed program
- 2. Facilitates the decomposition of the complex task of developing a software system into a number of simpler subtasks
- 3. Helps to reduce the number of things the programmer has to keep in mind at any time
- 4. Breaking down a complex task into a number of earlier subtasks also simplifies testing and debugging



List o f ADT's:

- 1. Insertion at first, middle, last
- 2. Deletion at first, middle, last
- 3. Searching
- 4. Reversing
- 5. Traversing
- 6. Modifying the list,
- 7. Merging the list