1. Define Array
➢ An Array is defined as, an ordered set of similar data items. All the data items of an array are stored in consecutive memory locations. ➢ The data items of an array are of same type and each data items can be accessed using the same name but different index value. ➢ • An array is a set of pairs, , such that each index has a value associated with it. It can be called as corresponding or a mapping

2. Define Structures
➢ In C, a way to group data that permits the data to vary in type. This mechanism is called the structure, for short struct. ➢ A structure (a record) is a collection of data items, where each item is identified as to its type and name.
➢ Self Refers Struct ➢ A self-referential structure is one in which one or more of its components is a pointer to itself

3. Define Union
➢ A union is similar to a structure, it is collection of data similar data type or dissimilar.

4. Define Pointers
➢ A pointer is a variable which contains the address in memory of another variable.

5. DMA
➢ Dynamic memory management refers to the process of allocating, using, and freeing memory at runtime in a program

Data Structures

6. Data Structures
➢ Data may be organized in many different ways. ➢ The logical or mathematical model of a particular organization of data is called a data structure.

7. Data Structures Operations
➢ Traversing: accessing each record/node exactly once so that certain items in the record may be processed
➢ Searching: Finding the location of the desired node with a given key value, or finding the locations of all such nodes which satisfy one or more conditions
➢ Inserting: Adding a new node/record to the structure.
➢ Deleting: Removing a node/record from the structure.
➢ Sorting: Arranging the records in some logical order (e.g., alphabetically according to some NAME key, or in numerical order according to some NUMBER key, such as social security number or account number)
➢ Merging: Combining the records in two different sorted files into a single sorted file.

8. Primitive data Structures
➢ Primitive data structures are the fundamental data types which are supported by a programming language. Basic data types such as integer, real, character and Boolean are known as Primitive data Structures

9. Non- Primitive data Structures:
➢ Non-primitive data structures are those data structures which are created using primitive data structures. Examples of non-primitive data structures is the processing of complex numbers, linked lists, stacks, trees, and graphs.

10.	**Linear Data Structure:** ➢ **A data structure is said to be linear if its elements form a sequence or a linear list.** ➢ **The common examples of linear data structure are Arrays, Queues, Stacks, Linked lists**
11.	**2. Non-linear Data Structure:** ➢ **A data structure is said to be non-linear if the data are not arranged in sequence or a linear.** ➢ **The insertion and deletion of data is not possible in linear fashion.** ➢ **This structure is mainly used to represent data containing a hierarchical relationship between elements. Trees and graphs are the examples of non-linear data structure.**

M-2

1. STACK
➢ **"A stack is an ordered list in which insertions (pushes) and deletions (pops) are made at one end called the top."**
➢ **Since the last element inserted into a stack is the first element removed, a stack is also known as a Last-In-First-Out (LIFO) list.**

➢ **A Recursive function** ➢ **A function is said to be recursively defined if the function definition refers to itself**

2. Queue
➢ **"A queue is an ordered list in which insertions (additions, pushes) and deletions (removals and pops) take place at different ends."** ➢ **The end at which new elements are added is called the rear, and that from which old elements are deleted is called the front.**
➢ **Since the first element inserted into a queue is the first element removed, queues are also known as First-In-First-Out (FIFO) lists.**

3. Circular Queue
➢ **It is "The queue which wrap around the end of the array." The array positions are arranged in a circle**

4. Dequeue
➢ **A deque (double ended queue) is a linear list in which elements can be added or removed at either end but not in the middle.**

5. Priority Queue
➢ **A priority queue is a collection of elements such that each element has been assigned a priority and such that the order**

in which elements are deleted and processed comes from the following rules:

- **(1) An element of higher priority is processed before any element of lower priority.**
- **(2) Two elements with the same priority are processed according to the order in which they were added to the queue.**

M-3

1. LINKED LIST

- **A linked list, or one-way list, is a linear collection of data elements, called nodes, where the linear order is given by means of pointers. That is, each node is divided into two parts:**
- **The first part contains the information of the element, and The second part, called the link field or next pointer field, contains the address of the next node in the list.**

2. DOUBLY LINKED LIST

- **It is a linear collection of data elements, called nodes, where each node N is divided into three parts:**
- **1. An information field INFO which contains the data of N**
- **2. A pointer field LLINK (FORW) which contains the location of the next node in the list**
- **3. A pointer field RLINK (BACK) which contains the location of the preceding node in the list**

M-4

1. Trees

- **A tree is a finite set of one or more nodes such that**
- **There is a specially designated node called root.**
- **The remaining nodes are partitioned into n >= 0 disjoint set T1,…,Tn, where each of these sets is a tree. T1,…,Tn are called the subtrees of the root.**

TERMINOLOGY

- ***Node:*** The item of information plus the branches to other nodes
- ***Degree:*** The number of subtrees of a node
- ***Degree of a tree:*** The maximum of the degree of the nodes in the tree.
- ***Terminal nodes (or leaf):*** nodes that have degree zero or node with no successor
- ***Nonterminal nodes***: nodes that don't belong to terminal nodes.
- ***Parent and Children:*** Suppose N is a node in T with left successor S1 and right successor S2, then N is called the Parent (or father) of S1 and S2. Here, S1 is called left child (or Son) and S2 is called right child (or Son) of N.
- ***Siblings:*** Children of the same parent are said to be siblings.
- ***Edge:*** A line drawn from node N of a T to a successor is called an edge
- ***Path:*** A sequence of consecutive edges from node N to a node M is called a path.
- ***Ancestors of a node:*** All the nodes along the path from the root to that node.
- ***The level of a node:*** defined by letting the root be at level zero. If a node is at level *l*, then it children are at level *l+1*.
- ***Height (or depth):*** The maximum level of any node in the tree

2. BINARY TREES

- **A binary tree T is defined as a finite set of nodes such that,**
- **T is empty or**
- **T consists of a root and two disjoint binary trees called the left subtree and the right subtree**

3. Skewed Tree

- **A skewed tree is a tree, skewed to the left or skews to the right. or It is a tree consisting of only left subtree or only right subtree.**
- **A tree with only left subtrees is called Left Skewed Binary Tree.**
- **A tree with only right subtrees is called Right Skewed Binary Tree.**

4. Complete Binary Tree

- **A binary tree T is said to complete if all its levels, except possibly the last level, have the maximum number node 2 i , i ≥ 0 and if all the nodes at the last level appears as far left as possible.**

5. Full Binary Tree

- **A full binary tree of depth 'k' is a binary tree of depth k having 2 k – 1 nodes, k ≥ 1.**

6. Extended Binary

- **Trees or 2-trees An extended binary tree is a transformation of any binary tree into a complete binary tree.**

7. BINARY TREE TRAVERSALS

- **Visiting each node in a tree exactly once is called tree traversal**

8. Inorder: Inorder traversal calls for moving down the tree toward the left until you cannot go further. Then visit the node, move one node to the right and continue. If no move can be done, then go back one more node.

9. Preorder: Preorder is the procedure of visiting a node, traverse left and continue. When you cannot continue, move right and begin again or move back until you can move right and resume.

10. Postorder: Postorder traversal calls for moving down the tree towards the left until you can go no further. Then move to the right node and then visit the node and continue

11. THREADED BINARY TREE

- **In the linked representation of any binary tree, there are more null links than actual pointers. These null links are replaced by the pointers, called threads, which points to other nodes in the tree**

12. BINARY SEARCH TREE

- **In a Binary search tree, the value of left node must be smaller than the parent node, and the value of right node must be greater than the parent node. This rule is applied recursively to the left and right subtrees of the root.**

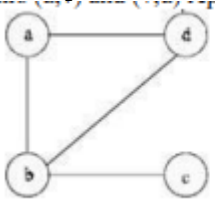
M-5

1. Graph

- **A graph G consist of two sets V and E**
  1. V is a finite nonempty set of vertex and
  2. E is a set of pairs of vertices these pairs are called edges
- **A graph can be represents as G = (V, E). V(G) will represent the set of vertices and E(G) will represent the set of edges of the graph G**

Undirected Graph: In a undirected graph the pair of vertices representing an edge is unordered. thus the pairs (u,v) and (v,u) represent the same edge.

Example:

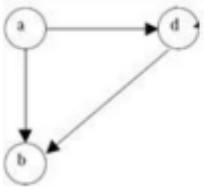


V(G)={a,b,c,d}

E(G)={(a,b),(a,d),(b,d),(b,c)}

Directed Graph (digraph): In a directed graph each edge is represented by a directed pair (u,v), v is the head and u is the tail of the edge. Therefore (v,u) and(<u,v) represent two different edges

Example:

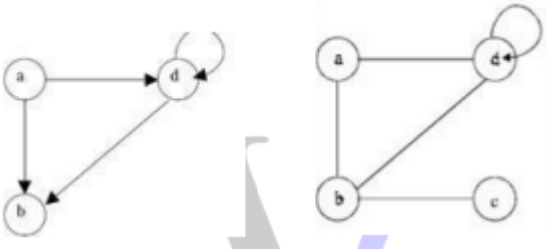


V(G)={a,b,d}

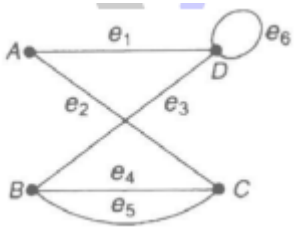
E(G)={(a,d), (a,b), (d,b)}

Self Edges/Self Loops: Edges of the form(v,v) are called self edges or self loops . It is an edge which starts and ends at the same vertex.

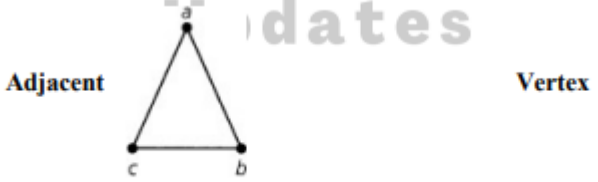
Example:



Mutigraph: A graph with multiple occurrences of the same edge is called a multigraph Example:



Complete Graph: An undirected graph with n vertices and exactly n(n-1)/2 edges is said to be a complete graph. In a graph all pairs of vertices are connected by an edge. Example : A complete graph with n=3 vertices



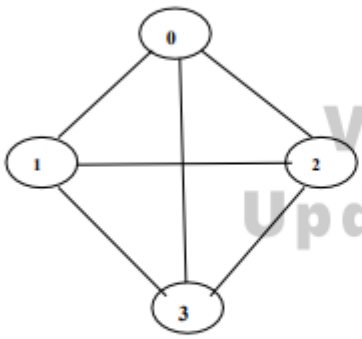
If (u,v) is an edge in E(G), then we say that the vertices u and v are adjacent and the edge(u,v) is incident on vertices u and v.

Connected Graph: An undirected graph G is said to be connected if for every pair of distinct vertices u and v in V(G) there is a path from u to v in G.

Degree of a vertex : In a undirected graph degree of a vertex is the number of edges incident on a vertex.

Adjacency Matrix: Let G=(V,E) be a graph with n vertices, n>=1. The adjacency matrix of G is a two dimensional n*n array for example a, with the property that a[i][j]=1 if there exist ane edge (i,j) (for a directed graph edge <i,j> is in E(G).a[i][j]=0 if no such edge in G.

Example:



Adjacency Matrix

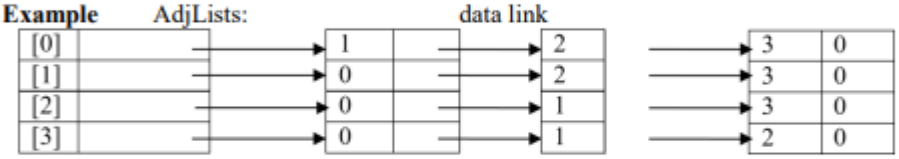
	0	1	2	3
0	0	1	1	1
1	1	0	1	1
2	1	1	0	1
3	1	1	1	0

Figure 5.1 Graph G1

Adjacency list: In adjacency matrix the n rows of the adjacency matrix are represented as n chains. There is one chain for each vertex in G. The nodes in chain i represent the vertices that are adjacent from vertex i. The data field of a chain node stores the index of an adjacent vertex.

Example: the adjacency list of graph G1 in figure 5.1 is shown below

Example AdjLists: data link



- For an undirected graph with n vertices and e edges. The linked adjacency lists representation requires an array of size n and 2e chain nodes.
- The degree of any vertex in an undirected graph may be determined by counting the number of nodes in the adjacency list. □ For a digraph the number of list nodes is only e.