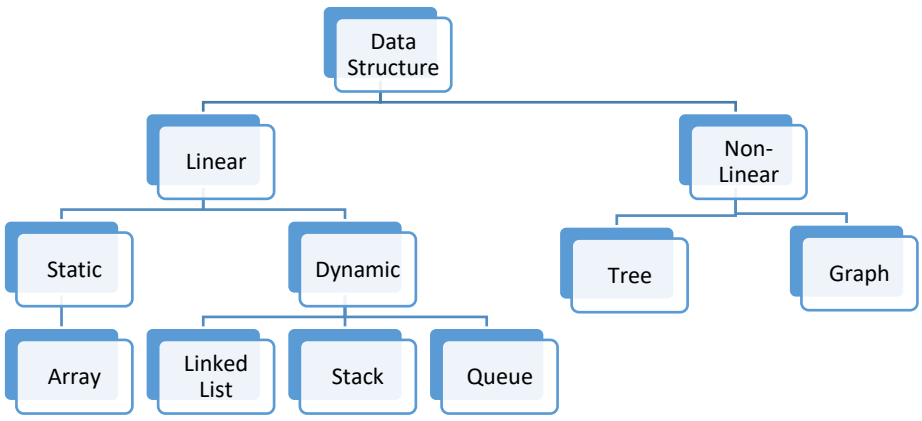
Data Structure :-

1. **What is Data Structure?**
   Ans:- Data structure is a specific way to store & organize data in computers memory so that these data can be used efficiently later.

2. **Why we used data structure, need of data structure?**
   Ans:- As applications are getting complex & data rich, there are three common problems that are " Data Search , process Speed & multiple request". To solve these problems ,data structure is used. Data can be organized in data structure in such way that all items may not required to be searched & required data can be searched almost instantly.

3. **What are applications of data structure?**
   Ans:-
   1. To implement mathematical vectors & matrices.
   2. Navigation of System
   3. Evaluating expressions
   4. Manage resource sharing such as CPU scheduling & disk scheduling.
   5. Finding shortest path in map.
   6. Airline networks.

4. **What are types of Data Structure, explain linear & non-linear data structure?**
   Ans:-



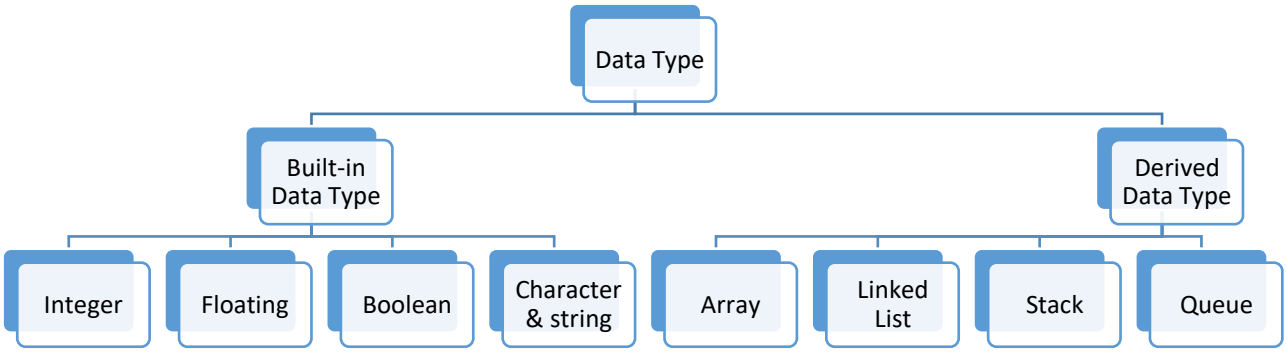
Linear Data Structure :- A data structure is said to be linear if its elements combine to form any specific order. There are two techniques to represent linear structure within memory

a) Linear relationship among all elements represented using linear memory location i.e. Array.
b) Linear relationship among all elements represented by using pointers or links i.e. Linked list.
e.g. Arrays , Linked list, Stack & Queue

Non-linear Data Structure :- this structure is mostly used to represent data that contain a hierarchical relationship among various elements.
e.g. Graph , Tree & Hash table.

5. What are D.S. data types?

Ans:-



6. Which operations we can perform using D.S.?

1) Traversing
2) Searching
3) Insertion
4) Deletion
5) Sorting
6) Merging

7. What are characteristics of D.S.?

Ans:- a) Correctness b) Time Complexity c)Space Complexity

8. **Explain Basic terms of D.S. such as Data,Data Object, Data item, group item, elementary items,attribute & entity,entity set,field,record,file.**

   Ans:-

   1) Data:- Data are values or set of values.
   2) Data Object:- Objects having some data.
   3) Data Item:- Data items refers to single unit of values.
   4) Group Items:-Data items that are divided into subitems are called “Group Items”.
   5) Elementary Items:- Data items that can’t be divided are called “Elementary items”.
   6) Attribute & entity :- An entity is that which contains attributes (properties) which may be assigned values.
   7) Entity Set :- Entities of similar attributes(properties) form an “Entity Set”.
   8) Filed :- Its is single elementary unit of information representing an attribute of an entity.
   9) Record :- Record is collection of field values of given entity.
   10) File :- File is collection of records of entities in given entity set.

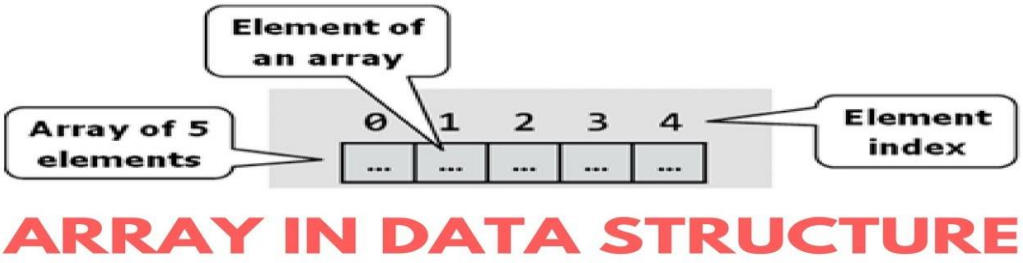
9. **What is Abstract Data Types? Give examples.**

   Ans:- Its type or class for objects whose behaviour is defined by set of value & set of operations.

10.What is Array?

Ans:- Array is fixed-size sequential collection of variables having same data type and name.

Array Representation:-



Element :- Each item stored in an array is called an “Element”

Index :-Each location of an element in an array has numerical index which is used to identify the element. Array index must start with 0.

11.Which operation we can perform on Array in D.S.?

Ans:- Array Operations are as follows :-

1) Insertion
2) Deletion
3) Traversal
4) Searching
5) Update

12.What is multi-dimensional array?

Ans:- Array of array is called “Multidimensional Array”.

Representation of Multidimensional array :-
e.g. Multidimensional array of 3x3 x[3][3].

	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

13.What are application of array?

Ans:-

- ➔ Arrays are widely used to implement mathematical vectors, matrices, and other kinds of rectangular tables.
- ➔ Many databases include one-dimensional arrays whose elements are records.
- ➔ Arrays are also used to implement other data structures such as strings, stacks, queues, heaps,and hash tables.
- ➔ Arrays can be used for sorting elements in ascending or descending order.

14.What is Linked list?

Ans :- A linked list is a sequence of data which are connected together via links.

15.What are types of Linked list?

Ans:- There are three types of linked list ,

1) Single Linked List
2) Double Linked List
3) Circular Linked List

16.Which operation we can perform on linked list.

Ans:- 1) Insertion (at beginning , at end , at specific position).
2) Delete beginning node
3) Delete specific node
4) Length of linked list
5) Traverse the linked list

17.What is Single linked list, give its representation?

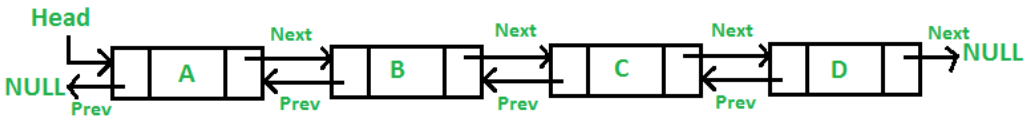
Ans:- In Single linked list there is node and each node contain data and link(pointer) to the next node. In this type of list only forward connection is possible.

Representation:-

18.What is double linked list,give its representation?

Ans:- In this list ,node contain data, left link pointer to hold the address of previous node's left link pointer address & right link pointer to hold the address of next node's left link pointer address. In this list both forward & backward connection is possible.

Representation:-

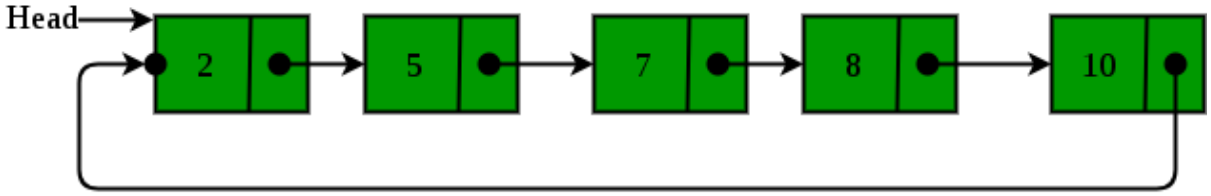


19.What is circular linked list,give its representation?

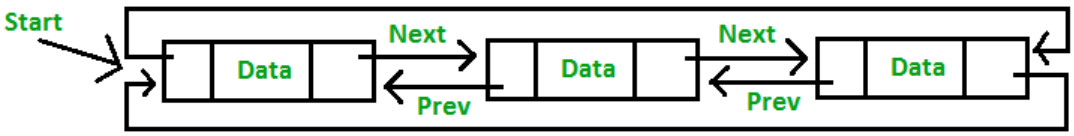
Ans:- Circular linked list is a linked list where all nodes are connected to form a circle. There is no NULL at the end. A circular linked list can be a singly circular linked list or doubly circular linked list.

Representation :-

A) Circular Single Linked List:-



B)Circular Double Linked List :-



20.What are applications of linked list?

Ans :- 1) Implement Stacks & Queue.

2) Implement Graph.

3) Dynamic memory allocation.

21.What is difference between array & linked list?

Ans :-

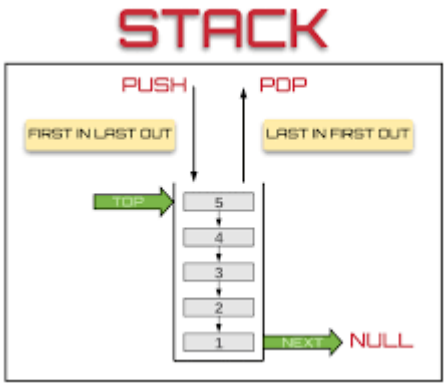
Sr.No.	Points	Array	Linked List
1	**Definition**	Array elements stored in continuous location	Linked list elements stored in random memory location.
2	**Size**	Fixed	Not fixed
3	**Accessing elements**	Randomly	Sequentially
4	**Memory Requirement**	Less	More
5	**Memory allocation**	Static	Dynamic

22.What is stack?

Ans :- Stack is linear data structure in which data operations can be done at only one end and follows “Last In First Out [LIFO]” order.

23.Give stack representation.

Ans:-



24.Which operation we can perform for stack?

Ans :- Operations of Stack are,

1) PUSH - Insert element into stack.
2) POP - Delete element from stack.
3) Traverse - Display all elements if stack.
4) Peek – Display topmost element of stack without deleting it.

25.What is infix, postfix & prefix?

Ans :- 1) Infix :- Operators are used between the operands.

e.g. A + B

2) Postfix :- Operators are used after operands.

e.g. AB +

3)Prefix :- operators are used before operands.

e.g. + AB

26.What are applications of stack?

Ans :- 1) Conversion of infix ,postfix & prefix expressions.

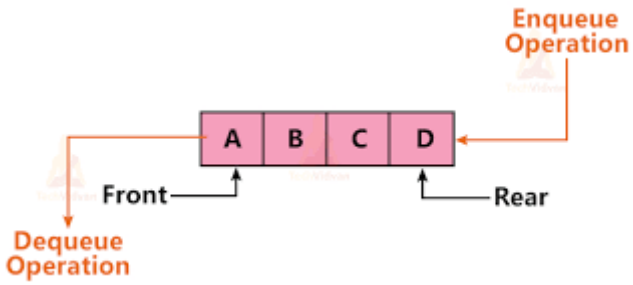
2) Recusrion.

27.What is queue?

Ans :- Queue is Abstract Data Structure to manipulate data. It has two end ,one is “Front“ and another is “Rear” . Data can be deleted from “Front” side and inserted from “Rear” side. Queue operations perform from both side. Queue follows “First In First Out” order.

28.Give queue representation.

Ans:-



29.Which operations we can perform on Queue?

Ans:- Operations on Queue are ,

1) Enqueue :- Insert elements to Queue
2) Dequeue :- Delete elements from Queue
3) Traverse :- Display all elements of Queue
4) Peek :-Display “Front” element without deleting it.

30.What is difference between Stack & Queue?

Ans :-

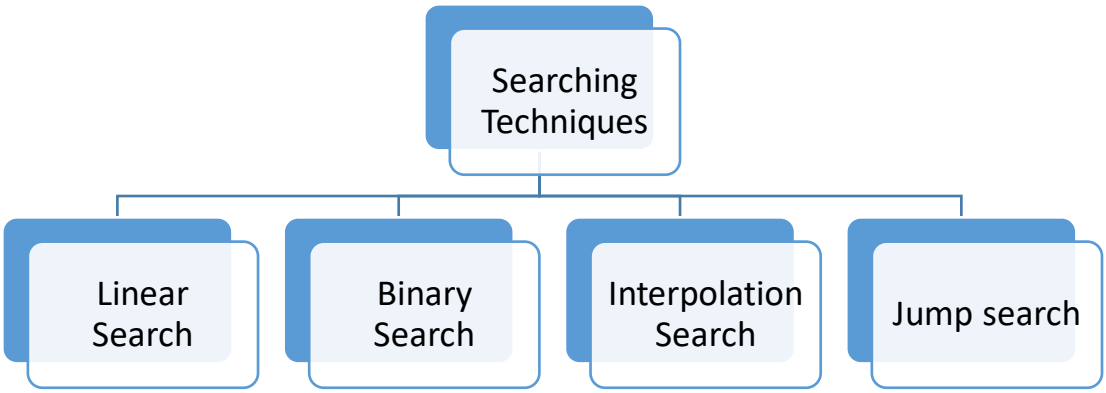
Sr.No.	Stack	Queue
1	Stack follows “Last In First Out [LIFO]“ order.	Queue follows “First In First Out [FIFO] “ order.
2	Stack operations can be performed only from one side.	Queue operations can be performed from both side.

31.What is searching in D.S.?

Ans :- Searching means to find whether a particular value is present in an array or not.

32.What are types of searching in D.S.?

Ans:-

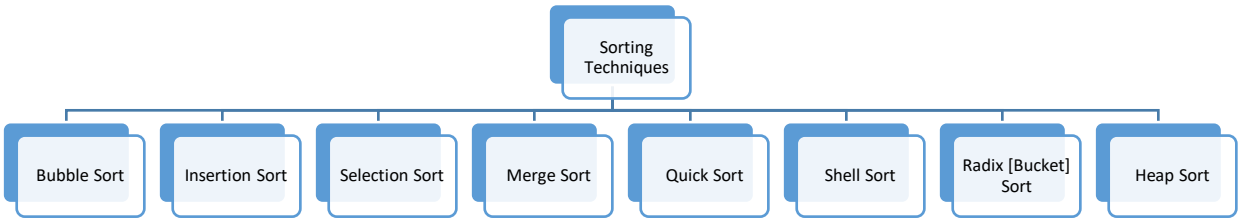


33.What is sorting is D.S.?

Ans :- Sorting means arranging elements of an array so that they are placed in some relevant order which may be ascending or descending order.

34.What are types of sorting?

Ans:-



35.What is graph? Explain in detail.

Ans:-

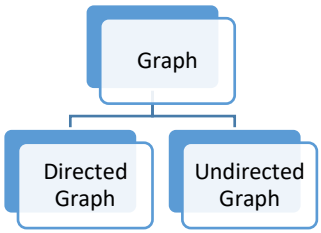
➔ **Definition :-** A graph is an abstract data structure that is used to implement the mathematical concept of graphs. It is basically a collection of vertices (also called nodes) and edges that connect these vertices. A graph G is defined as an ordered set (V, E), where V(G) represents the set of vertices and E(G) represents the edges that connect these vertices.

➔ **Terminology of Graph :-**

1) **Vertices :-** A graph is a set of points, called nodes or **vertices.**
2) **Edge :- Vertices** interconnected by a set of lines called **edges.**
3) **Adjacent Node :- When two nodes are connected by same edge is called Adjacent Node .**
4) **Degree of Node :-** Degree of a node is the total number of edges containing the node.If degree of node is 0, it means that node does not belong to any edge and such a node is known as an isolated node.
5) **Regular Path :-** It is a graph where each vertex has the same number of neighbours. That is, every node has the same degree. A regular graph with vertices of degree k is called a k–regular graph or a regular graph of degree k.
6) **Path :-** Path represents a sequence of edges between the two vertices.
7) **Closed Path:-** A path P is known as a closed path if the edge has the same end-points.

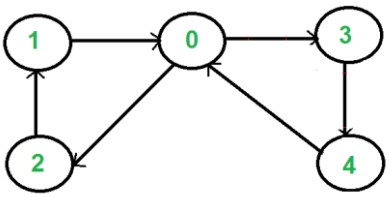
8) **Simple Path :-** A path P is known as a simple path if all the nodes in the path are distinct with an exception that v0 may be equal to vn. If v0 = vn, then the path is called a closed simple path.
Where V0 :-First Vertices & Vn :- nth vertices.
9) **Cycle :-** A path in which the first and the last vertices are same. A *simple cycle* has no repeated edges or vertices (except the first and last vertices).
10) **Connected Graph :- A graph is said to be connected if two vertices are connected through path.**
11) **Complete Graph :-** A graph G is said to be complete if all its nodes are fully connected. That is, there is a path from one node to every other node in the graph. A complete graph has n(n–1)/2 edges, where n is the number of nodes in graph.
12) **Weighted Graph :-** A graph is said to be weighted or labelled if every edge in the graph
is assigned some data. In a weighted graph, the edges of the graph are assigned some weight or length. The weight of an edge denoted by w(e) is a positive value which indicates the cost of traversing the edge.
13) **Multiple Edges :-** Distinct edges which connect the same end-points are called multiple edges.
14) **Loop :-** An edge that has identical end-points is called a loop.
15) **Multi-graph :-** A graph with multiple edges and/or loops is called a multi-graph.
16) **Size of Graph :-** The size of a graph is the total number of edges in it.

➔ **Types of Graphs :-**



A) **Directed Graph :-** A directed graph G, also known as a *digraph*, is a graph in which every edge has a direction assigned to it.

Fig. Directed Graph

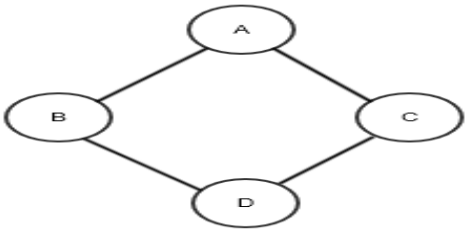


➔ **Terminology of Directed Graph :-**

1) **Out degree of node :-** The out of degree node is number of edges that originates from node i.e. outgoing edges from node.
2) **In degree of node :-** The in degree of node is number of edges that terminates at node i.e. incoming edges of node.
3) **Degree of node :-** Its sum of in degree and out degree of node.
4) **Isolated Vertex :-** Vertex with degree zero is called isolated vertex.
5) **Pendant Vertex :-** A vertex with degree one is called Pendant Vertex .Its also called leaf vertex.
6) **Cut Certex :-** A vertex which when deleted would disconnect the remaining graph.
7) **Source :-** A node u is known as a source if it has a positive out-degree but a zero in-degree.
8) **Sink :-** A node u is known as a sink if it has a positive in-degree but a zero out-degree.
9) **Reachability :-** A node v is said to be reachable from node u, if and only if there exists a (directed) path from node u to node v.
10) **Strongly Connected Graph :-** A digraph is said to be strongly connected if and only if there exists a path between every pair of nodes in graph.
11) **Unilaterally Connected Graph :-** A digraph is said to be unilaterally connected if there exists a path between any pair of nodes u, v in graph such that there is a path from u to v or a path from v to u, but not both .
12) **Weakly Connected Graph :-** Directed graph is said to be weakly connected if it is connected by ignoring the direction of edges. That is, in such a graph, it is possible to reach any node from any other node by traversing edges in any direction (may not be in the direction they point). The nodes in a weakly connected directed graph must have either out-degree or in-degree of at least 1.
13) **Multiple(Parallel) Edges :-** Distinct edges which connect the same end-points are called multiple edges.
14) **Simple Directed Graph :-** A directed graph G is said to be a simple directed graph if and only if it has no multiple edges.

B) **Undirected Graph :-** A undirected graph G, is a graph in which every edge has no direction assigned to it.

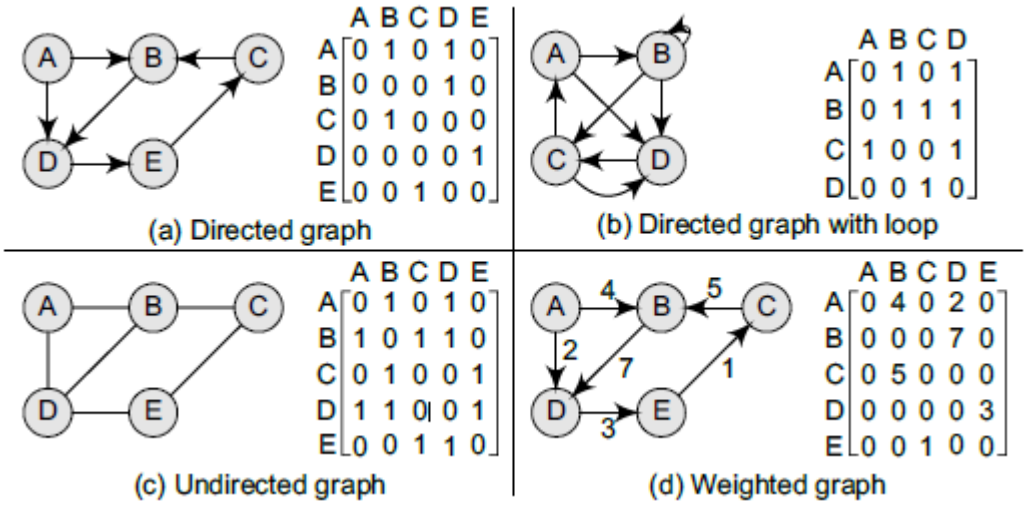
Fig. Directed Graph



➔ **Representation of Graph :-**

A) **Adjacency Matrix Representation :-** An adjacency matrix is used to represent which nodes are adjacent to one another.

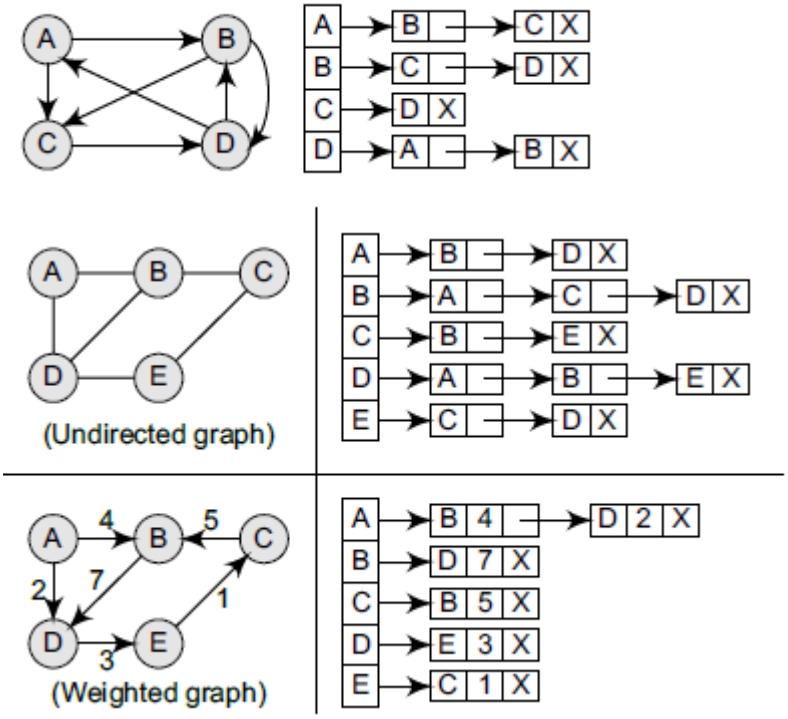
- In undirected graph , if two nodes are adjacent then it represented by 1 else 0 in matrix.
- In directed graph , if there is direction between two nodes then it represented by 1 else 0 in matrix. Also for weighted directed graph matrix is represented by its weight.
- E.g.



B) **Adjacency List Representation :-**In this, Graph represent adjacency nodes using list .

- Undirected graph & Directed graph are represented using single linked list.
- Weighted directed graph are represented using double linked list.
- E.g.

Fig. Directed Graph representation using List



➔ **Graph Traversal Algorithm :-**

A) Breadth First Search Algorithm :-

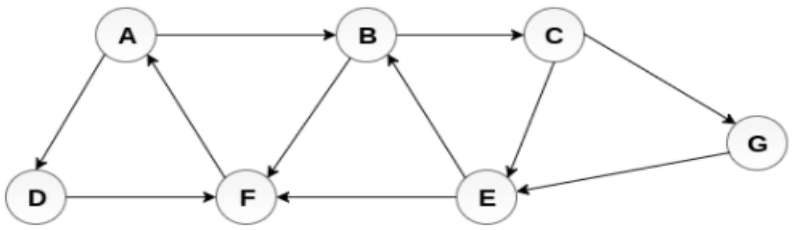
Statement :- Breadth first search is a graph traversal algorithm that starts traversing the graph from root node and explores all the neighbouring nodes. Then, it selects the nearest node and explore all the unexplored nodes. The algorithm follows the same process for each of the nearest node until it finds the goal. The algorithm explores all neighbours of all the nodes and ensures that each node is visited exactly once and no node is visited twice. Queue is used for BFS traversal.

Algorithm:-

Step 1: SET STATUS = 1 (ready state) for each node in G
Step 2: Enqueue the starting node A and set its STATUS = 2 (waiting state)
Step 3: Repeat Steps 4 and 5 until QUEUE is empty
Step 4: Dequeue a node N. Process it and set its STATUS = 3 (processed state).
Step 5: Enqueue all the neighbours of N that are in the ready state (whose STATUS = 1) and set their STATUS = 2 (waiting state) [END OF LOOP]
Step 6: EXIT

Example:-

Q . Consider the graph G shown in the following image, calculate the minimum path p from node A to node E. Given that each edge has a length of 1.



Adjacency Lists

A : B, D
B : C, F
C : E, G
G : E
E : B, F
F : A
D : F

Solution :-

Step 1:- Add A to QUEUE1 and NULL to QUEUE2.

QUEUE1 = {A}
QUEUE2 = {NULL}

Step 2:- Delete the Node A from QUEUE1 and insert all its neighbours. Insert Node A into QUEUE2

QUEUE1 = {B, D}
QUEUE2 = {A}

Step 3:-Delete the node B from QUEUE1 and insert all its neighbours. Insert node B into QUEUE2.

QUEUE1 = {D, C, F}
QUEUE2 = {A, B}

Step 4:- Delete the node D from QUEUE1 and insert all its neighbours. Since F is the only neighbour of it which has been inserted, we will not insert it again. Insert node D into QUEUE2.

QUEUE1 = {C, F}
QUEUE2 = { A, B, D}

Step 5:- Delete the node C from QUEUE1 and insert all its neighbours. Add node C to QUEUE2.

QUEUE1 = {F, E, G}
QUEUE2 = {A, B, D, C}

Step 6:- Remove F from QUEUE1 and add all its neighbours. Since all of its neighbours has already been added, we will not add them again. Add node F to QUEUE2.

QUEUE1 = {E, G}
QUEUE2 = {A, B, D, C, F}

Step 7:- Remove E from QUEUE1, all of E's neighbours has already been added to QUEUE1 therefore we will not add them again. All the nodes are visited and the target node i.e. E is encountered into QUEUE2.

QUEUE1 = {G}
QUEUE2 = {A, B, D, C, F, E}

Now, backtrack from E to A, using the nodes available in QUEUE2.

The minimum path will be **A** → **B** → **C** → **E**.

B) Depth First Search Algorithm

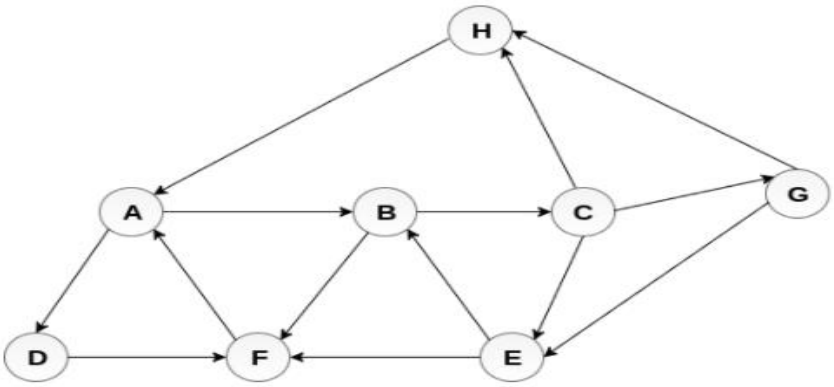
Statement :- Depth first search (DFS) algorithm starts with the initial node of the graph G, and then goes to deeper and deeper until we find the goal node or the node which has no children. The algorithm, then backtracks from the dead end towards the most recent node that is yet to be completely unexplored. The data structure which is being used in DFS is stack. In DFS, the edges that leads to an unvisited node are called “Discovery Edges” while the edges that leads to an already visited node are called “Block Edges”.

Algorithm :-

Step 1: SET STATUS = 1 (ready state) for each node in G
Step 2: Push the starting node A on the stack and set its STATUS = 2 (waiting state)
Step 3: Repeat Steps 4 and 5 until STACK is empty
Step 4: Pop the top node N. Process it and set its STATUS = 3 (processed state)
Step 5: Push on the stack all the neighbours of N that are in the ready state (whose STATUS = 1) and set their STATUS = 2 (waiting state)
[END OF LOOP]
Step 6: EXIT

Example :-

Q. Consider the graph G along with its adjacency list, given in the figure below. Calculate the order to print all the nodes of the graph starting from node H, by using depth first search (DFS) algorithm.



Adjacency Lists

A : B, D
B : C, F
C : E, G, H
G : E, H
E : B, F
F : A
D : F
H : A

Solution :-

Step 1:-Push H onto the stack

STACK : H

Step 2:-POP the top element of the stack i.e. H, print it and push all the neighbours of H onto the stack that are is ready state.

Print H
STACK : A

Step 3:-Pop the top element of the stack i.e. A, print it and push all the neighbours of A onto the stack that are in ready state.

Print A
Stack : B, D

Step 4:-Pop the top element of the stack i.e. D, print it and push all the neighbours of D onto the stack that are in ready state.

Print D
Stack : B, F

Step 5:-Pop the top element of the stack i.e. F, print it and push all the neighbours of F onto the stack that are in ready state.

Print F
Stack : B

Step 6:-Pop the top of the stack i.e. B and push all the neighbours

Print B
Stack : C

Step 7:-Pop the top of the stack i.e. C and push all the neighbours.

Print C
Stack : E, G

Step 8:-Pop the top of the stack i.e. G and push all its neighbours.

Print G
Stack : E

Step 9:-Pop the top of the stack i.e. E and push all its neighbours.

Print E
Stack :

Hence, the stack now becomes empty and all the nodes of the graph have been traversed.

The printing sequence of the graph will be :

H → A → D → F → B → C → G → E

➔ **Shortest Path Algorithm :-**

A) Minimum Spanning Tree
B) Dijkstra's Algorithm
C) Warshall's Algorithm
D) Prim's Algorithm
E) Kruskal's Algorithm

36.What is tree? Explain its terminology.

Ans:-

➔ **Definition :-** A tree is recursively defined as a set of one or more nodes where one node is designated as the root of the tree and all the remaining nodes can be partitioned into non-empty sets each of which is a sub-tree of the root. Tree is non –linear data structure.

➔ **Tree Terminology :-**

A) **Root Node :-** The root node R is the topmost node in the tree. If R = NULL, then it means the tree is empty.

B) **Leaf Node :-** A node that has no children is called the leaf node or the terminal node.

C) **Path :-** A sequence of consecutive edges is called a *path.*

D) **Ancestor Node :-** An ancestor of a node is any predecessor node on the path from root to that
node. The root node does not have any ancestors.

E) **Descendants Node :-** A descendant node is any successor node on any path from
the node to a leaf node. Leaf nodes do not have any descendants.

F) **Level Number :-** Every node in the tree is assigned a *level number* in such a way that the root node is at level 0, children of the root node are at level number 1. Thus, every node is at one level higher than its parent. So, all child nodes have a level number given by parent's level number + 1.

G) **Degree of Node :-** Degree of a node is equal to the number of children that a node has.
The degree of a leaf node is zero.

H) **In degree :-** In-degree of a node is the number of edges arriving at that node.

I) **Out degree :-** Out-degree of a node is the number of edges leaving that node.

J) **Parent :-** If N is any node in tree that has *left successor* S1 and *right successor* S2, then N is called the *parent* of S1 and S2.Correspondingly, S1 and S2 are called the left child and the right child of N. Every node other than the root node has a parent.

K) **Siblings :-** All nodes that are at the same level and share the same parent are called *siblings* (brothers).

L) **Similar Trees :-** Two binary trees T and T ¢ are said to be similar if both these trees have the same structure.

M) **Copies :-** Two binary trees T and T ¢ are said to be *copies* if they have similar structure and if they have same content at the corresponding nodes.

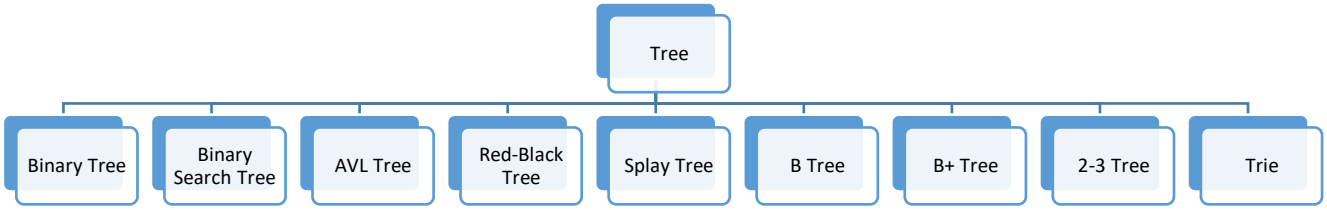
N) **Edge :-** It is the line connecting a node N to any of its successors.

O) **Depth :-** The *depth* of a node N is given as the length of the path from the root R to the node N. The depth of the root node is zero.

P) **Height of Tree :-** It is the total number of nodes on the path from the root node to the deepest node in the tree.

37. What are types of trees?

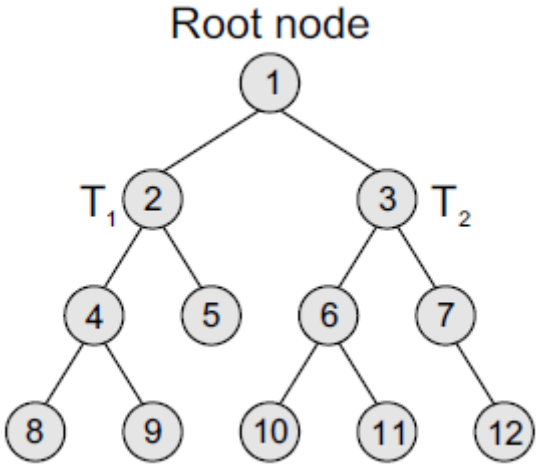
Ans:-



38.What is Binary Tree? Explain

Ans:- Binary tree is a tree which having 0 , 1 & maximum 2 chldren.

Example :-



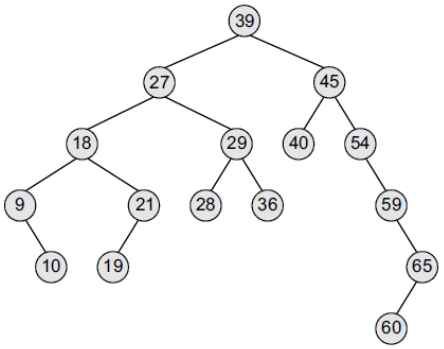
Complete Binary Tree

39. What is Binary Search Ttree? Explain in detail.

Ans:-

➔ **Definition:-** Binary Search Tree (BST) is also called “Ordered Binary Tree”. In BST , all nodes are arranged in order. In BST , all nodes in left sub-tree have values less than “Root Node” & all nodes in right sub-tree have values which is equal to or greater than “Root Node”.

Example :-



Binary Search Tree

➔ **Traversal Technique of BST :-**

A) Pre Order Traversal :-
   Step 1:- Start from Root Node
   Step 2:- Traversing left sub-tree
   Step 3:- Traversing right sub-tree

B) In Order Traversal :-
   Step 1:- Traversing left sub-tree
   Step 2:- Go to Root Node
   Step 3:- Traversing right sub-tree

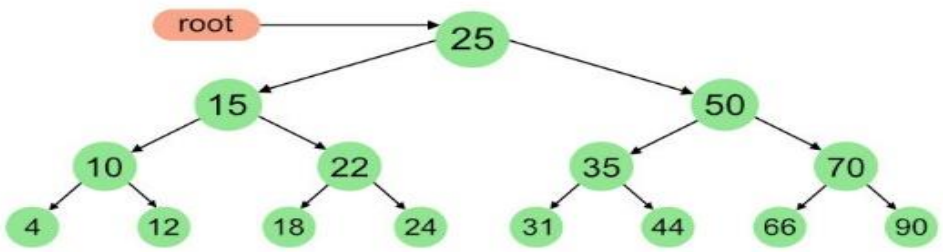
C) Post Order Traversal :-
   Step 1:- Traversing left sub-tree
   Step 2:-Traversing right sub-tree
   Step 3:- Go to Root Node

Example:-

InOrder(root) visits nodes in the following order:
4, 10, 12, 15, 18, 22, 24, 25, 31, 35, 44, 50, 66, 70, 90

A Pre-order traversal visits nodes in the following order:
25, 15, 10, 4, 12, 22, 18, 24, 50, 35, 31, 44, 70, 66, 90

A Post-order traversal visits nodes in the following order:
4, 12, 10, 18, 24, 22, 15, 31, 44, 35, 66, 90, 70, 50, 25



➔ **Operations of Binary Search Tree :-**

1) Insertion
2) Deletion
3) Searching
4) Length of nodes
5) Traversing

40.What is difference between Binary Tree & Binary Search Tree.

Ans:-

Sr.No.	Binary Tree	Binary Search Tree
1.	Binary Tree is a specialized form of tree which represents hierarchical data in tree structure.	Binary Search Tree is type of Binary Tree which keeps data in sorted order.
2.	Binary Tree is unordered tree.	Binary Search Tree is Ordered Tree.

41.Explain AVL tree in detail .

Ans:-

➔ **Definition :-** AVL tree is self-balancing tree , balancing is consider using Balance Factor and it may be 0 , 1 or -.1.

If Balance factor is 0 , then height of left sub-tree is equal to height of right left sub-tree.

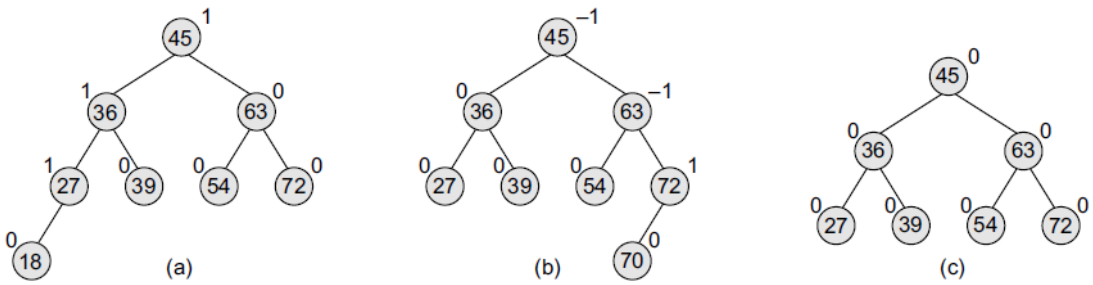
If balance factor is 1,then left sub-tree is one level higher than right sub-tree.

If balance factor is -1 , then left sub-tree is one level lower than right sub-tree.

➔ In an AVL tree, the heights of the two sub-trees of a node may differ by at most one. Due to this property, the AVL tree is also known as a height-balanced tree.

➔ AVL tree invented by G.M. Adelson-Velsky and E.M. Landis in 1962. Tree is named AVL in honour by its inventors.

➔ Example :-



(a) Left-heavy AVL tree, (b) right-heavy tree, (c) balanced tree

➔ **Operations on AVL Tree :-**

1. Insertion
2. Deletion
3. Searching
4. Insertion Rotation

- *LL rotation* The new node is inserted in the left sub-tree of the left sub-tree of the critical node.
- *RR rotation* The new node is inserted in the right sub-tree of the right sub-tree of the critical node.
- *LR rotation* The new node is inserted in the right sub-tree of the left sub-tree of the critical node.
- *RL rotation* The new node is inserted in the left sub-tree of the right sub-tree of the

5. Deletion Rotation

- L 0 Rotation
- L 1 Rotation
- L -1 Rotation
- R 0 Rotation
- R 1 Rotation
- R -1 Rotation

42.Explain Red-Black Tree in detail.

Ans:-

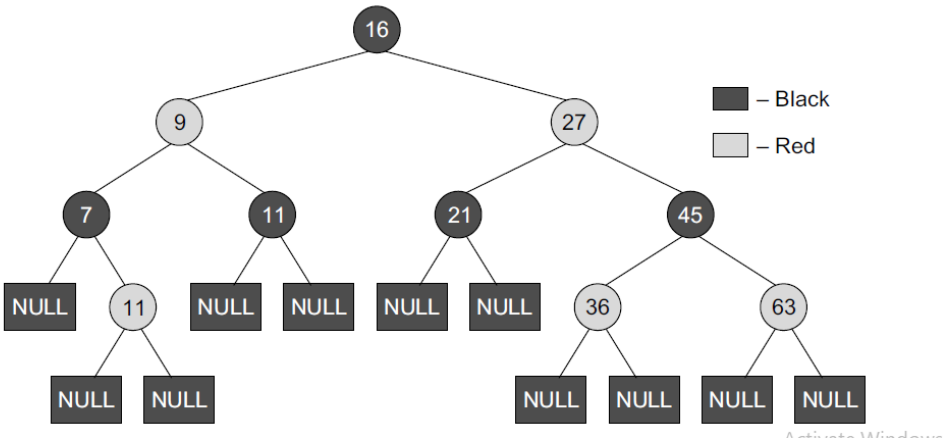
➔ **Definition :-** A Binary Search Tree is called Red-Black Tree if it has following properties.

1. The colour of a node is either red or black.
2. The colour of the root node is always black.
3. All leaf nodes are black.
4. Every red node has both the children coloured in black.
5. Every simple path from a given node to any of its leaf nodes has an equal number of black nodes.
6. It a self balancing tree.

➔ Red-Black Tree is also called “Symmetric Binary B-Tree” .

➔ It was invented by “Rudolf Bayer” in 1972.

➔ Example :-



➔ **Operations :-**

1. Insertion
2. Deletion

43.What is Order of Tree?

Ans:- Maximum number of children possibility is called “Order of Tree”.

44.What is key of Tree ?

Ans:- “Key” is collection of values.

45.Explain B Tree Concept .

Ans:-

➔ **Definition :-** B tree is Binary Search Tree which has following property,

1. It is self balanced tree.
2. Generalization of BST in which node can have more than 2 keys or more than 2 children.
3. All leaf nodes are at same level.
4. Every node in B tree has at most maximum m children (m is order of tree).
5. Every node in the B tree except the root node and leaf nodes has at least (minimum) m/2
   children. This condition helps to keep the tree bushy so that the path from the root node to
   the leaf is very short, even in a tree that stores a lot of data.
6. The root node has at least two children if it is not a terminal (leaf) node.
7. Every node has maximum (m-1) keys .
8. Minimum keys for root node is 1
9. Minimum keys for all other nodes have ⌈m/2⌉-1 keys

➔ **Operations on B Tree**

1. Insertion
2. Deletion
3. Searching
4. Traversal

46.Explain B+ Tree Concept .

Ans:-

➔ **Definition :-** B+ Tree has following properties,

1. All leaves are at same level.
2. The root has at least two children.
3. Each node except root can have maximum m children & at least m/2 children.
4. Each node can contain maximum of m-1 keys and minimum ⌈m/2⌉ -1 keys
5. Data can be stored on leaf nodes & internal nodes can only store key values.
6. Leaf nodes of B+ Tree are linked to each other to form single linked list

➔ **Advantages of B+ Tree over B tree**

1. Records can be fetched in equal number of disk accesses
2. It can be used to perform a wide range of queries easily as leaves are linked to nodes at the upper level
3. Height of the tree is less and balanced
4. Supports both random and sequential access to records
5. Keys are used for indexing

➔ **Operations of B+ Tree**

1. Insertion
2. Deletion
3. Search
4. Traversal

47.Difference between B Tree & B+ Tree.

Ans:-

Sr.No.	B Tree	B+ Tree
1.	Search keys are not repeated	Stores redundant search key
2.	Data is stores in internal or leaf nodes	Data is stored only in leaf nodes
3.	Leaf nodes can't be stores as linked list.	Leaf nodes data can be ordered using sequential linked list.
4.	Searching takes more time as data can be found in leaf as well as non-leaf node.	Searching takes less time as data can be found only at leaf node
5.	Deletion of leaf node is complicated.	Deletion is very simple as data in leaf node.
6.	Structure & operations are complicated	Structure & operations are easy

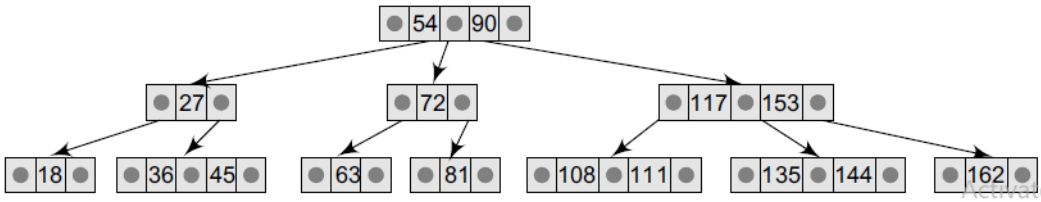
48.Explain 2-3 Trees.

Ans:-

➔ **Definition :-** A tree called 2-3 Tree if it has following property.

- In 2-3 Tree , each interior node has two or three children.
- Nodes with two children are called 2-nodes. The 2-nodes have one data value and two children
- Nodes with three children are called 3-nodes. The 3-nodes have two data values and three children (left child, middle child, and a right child).

➔ Example :-



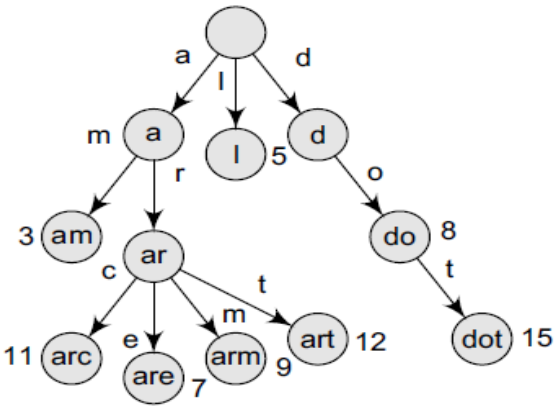
➔ **Operations :-**

1. Insertions
2. Deletions
3. Searching

49.Explain TRIE tree concept.

Ans:- The term trie has been taken from the word 'retrieval'. A trie is an ordered tree data structure, which was introduced in the 1960s by Edward Fredkin. Trie stores keys that are usually strings. It is basically a "k-ary" position tree.

Example :-



50. Give time complexity of each operation of Data Structure.

Ans:-

Algorithm	Average Time Complexity				Worst Time Complexity			
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion
Array	O(1)	O(N)	O(N)	O(N)	O(1)	O(N)	O(N)	O(N)
Stack	O(N)	O(N)	O(1)	O(1)	O(N)	O(N)	O(1)	O(1)
Queue	O(N)	O(N)	O(1)	O(1)	O(N)	O(N)	O(1)	O(1)
Singe L.L.	O(N)	O(N)	O(1)	O(1)	O(N)	O(N)	O(1)	O(1)
Double L.L.	O(N)	O(N)	O(1)	O(1)	O(N)	O(N)	O(1)	O(1)
Hash table	O(1)	O(1)	O(1)	O(1)	O(N)	O(N)	O(N)	O(N)
Binary Search Tree	O(log N)	O(log N)	O(log N)	O(log N)	O(N)	O(N)	O(N)	O(N)
AVL Tree	O(log N)	O(log N)	O(log N)	O(log N)	O(log N)	O(log N)	O(log N)	O(log N)
B Tree	O(log N)	O(log N)	O(log N)	O(log N)	O(N)	O(N)	O(N)	O(N)
Red Black Tree	O(log N)	O(log N)	O(log N)	O(log N)	O(log N)	O(log N)	O(log N)	O(log N)

51.Give time & space complexity of Searching & Sorting Techniques.

Ans:-

Algorithm	Average Time Complexity	Worst Time Complexity	Worst Space Complexity
Linear Search	O(N)	O(N)	O(1)
Binary Search	O(log N)	O(log N)	O(1)
Interpolation Search	log(log N)	O(N)	
Jump Search	O(√N)	O(√N)	
Bubble Sort	$O(N^2)$	$O(N^2)$	O(1)
Selection Sort	$O(N^2)$	$O(N^2)$	O(1)
Insertion Sort	$O(N^2)$	$O(N^2)$	O(1)
Merge Sort	O(N log N)	O(N log N)	O(N)
Quick Sort	O(N log N)	$O(N^2)$	O(log N)
Heap Sort	O(N log N)	O(N log N)	O(N)
Radix Sort	O(NK)	O(NK)	O(N+K)
Shell Sort	$O((N \log N)^2)$	$O((N \log N)^2)$	O(1)

Note :- In Radix sort , N means numbers that has to be sorted and K is the number of digits in largest number.