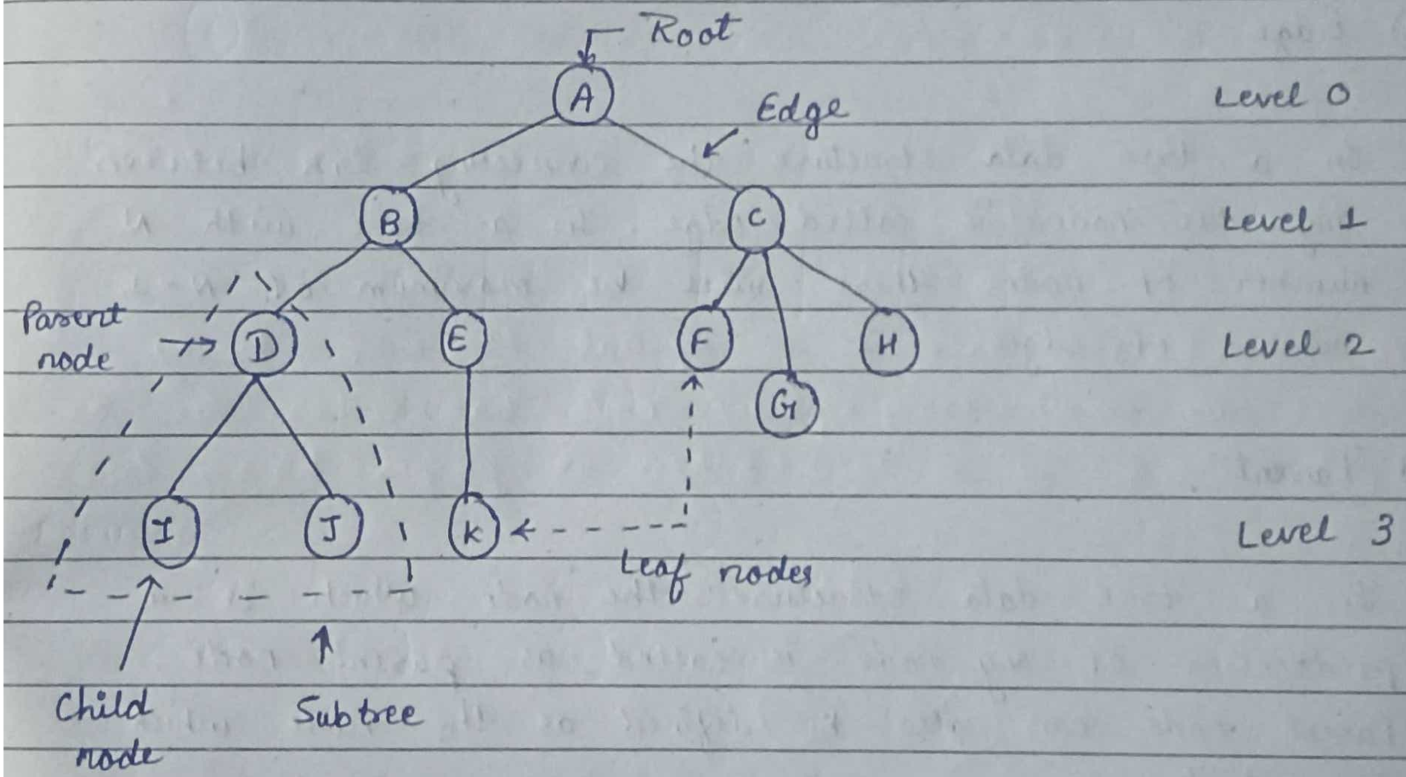
Unit - III

Tree And Graph

Tree data Structure

- Tree data structure is a specialized data structure to store data in hierarchial manner. It is used to organize and store data in computer.

- It consist of central node, structural node and subnodes which are connected via edges. we can also say that tree data structure has roots, branches and leaves connected.



Page No.
Date

Basic terminology

① Root

In a tree data structure the first node is called as Root node. Every tree must have root node. we can say that the root node is the origin of the tree data structure. In any ~~t~~ tree there must be only one root node we can never have multiples root node in a tree.

② Edge

In a tree data structure the connecting link between any two node is called edge. In a tree with N number of nodes there will be maximum of N-1 number of edges.

③ Parent

In a tree data structure the node which is a predecessor of any node is called as parent node. Parent node can also be defined as the node which has child.

④ Child

In a tree data structure the node which is descendent of any node is called as child node. In simple word the node which has a link from its parent node is called as Child node.

⑤ Sibling

In a tree data structure nodes which already belong to same parent are called as sibling.

⑥ Leaf

In a tree data structure the node which does not have a child is called Leaf nodes.

⑦ Degree

In a tree data structure the total number of children of node is called as Degree of that node

⑧ Level

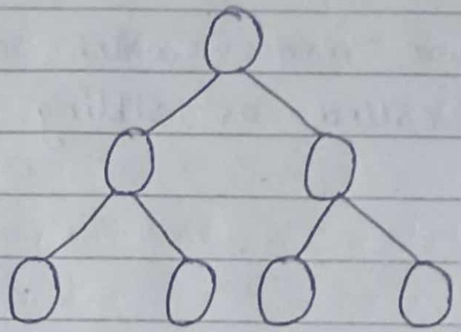
In a tree data structure the root node is said to be at level 0 and the children of root node are at Level 1 and the children of the nodes which are at level 1 will be at level 2 and so on

⑨ Subtree

In a tree data structure each child forms a node forms a subtree.

Binary Trees

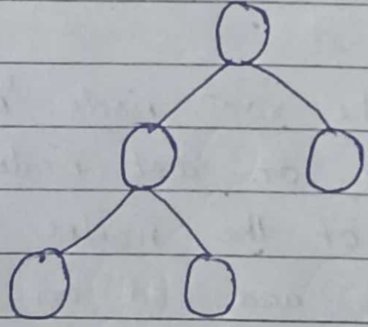
A binary tree data structure is a hierarchial data structure in which each node has at most two children reffered to as the left and right child.



Types :-

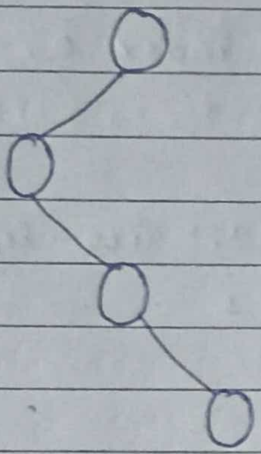
① Full Binary tree

A full binary tree is a special type of binary tree in which every parent node has either two or no children it is also known as proper binary tree.



(2) ~~Tree~~ Degenerate Tree

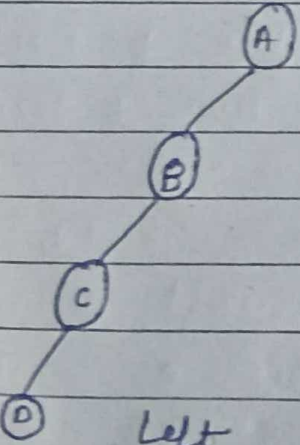
A tree where every parent node has one child. A degenerate tree is a trees having a single child either left or right.



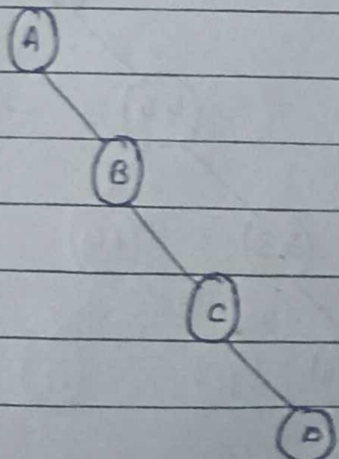
(3) Skewed Binary tree

A skewed binary tree is a degenerate tree in which the tree is either dominated by left node or the right node

(i) Right skewed Binary tree
(ii) Left skewed Binary tree



Left



Right

Page No.

Date

Tree Presentation as Array

Rules

1> Root Element

The root of the tree is stored at the first position in the array. i.e index 0.

2> Left Child

for any node at index 'i' its left child is stored at index 2 * i + 1

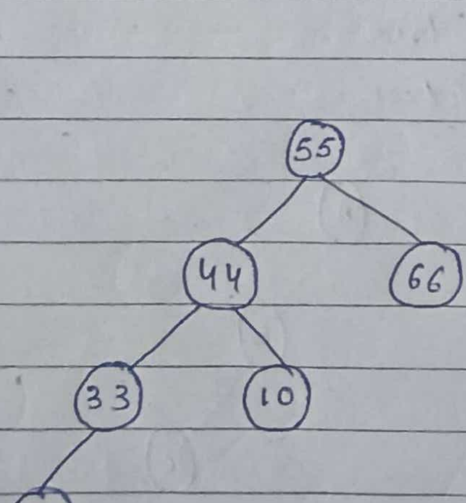
3> Right Child

for any node at index 'i' its right child is stored at index 2 * i + 2

4> Parent node

for any node at index 'i' its parent is located at index (i - 1)//2

Example :-



55	44	66	33	10			20
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]

44 = 2 * 0 + 1 = 1

66 = 2 * 0 + 2 = 2

33 = 2 * 1 + 1 = 3

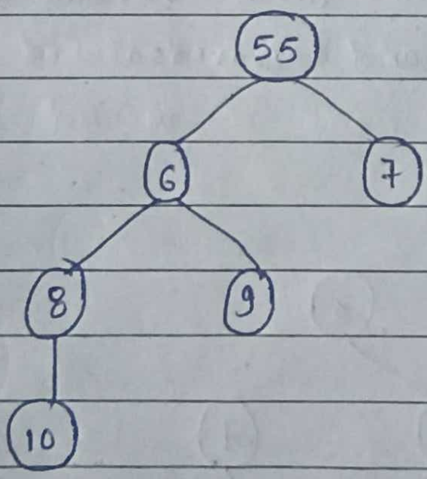
10 = 2 * 1 + 2 = 4

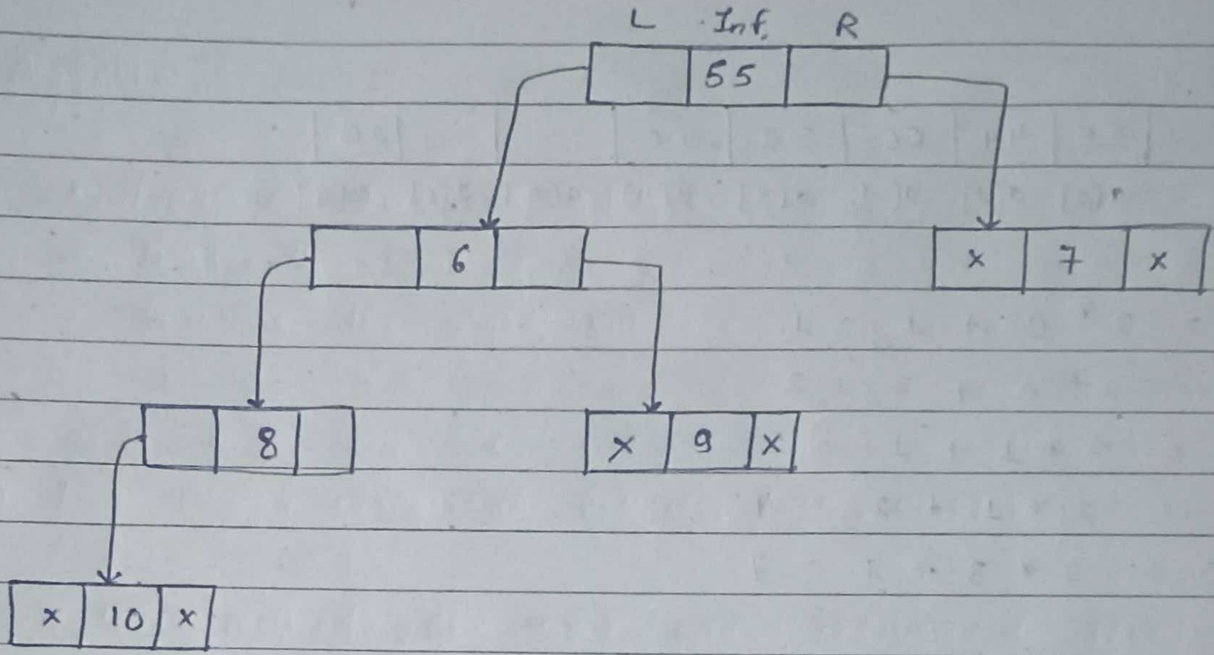
20 = 2 * 3 + 1 = 7

Tree as Linked List

Rule

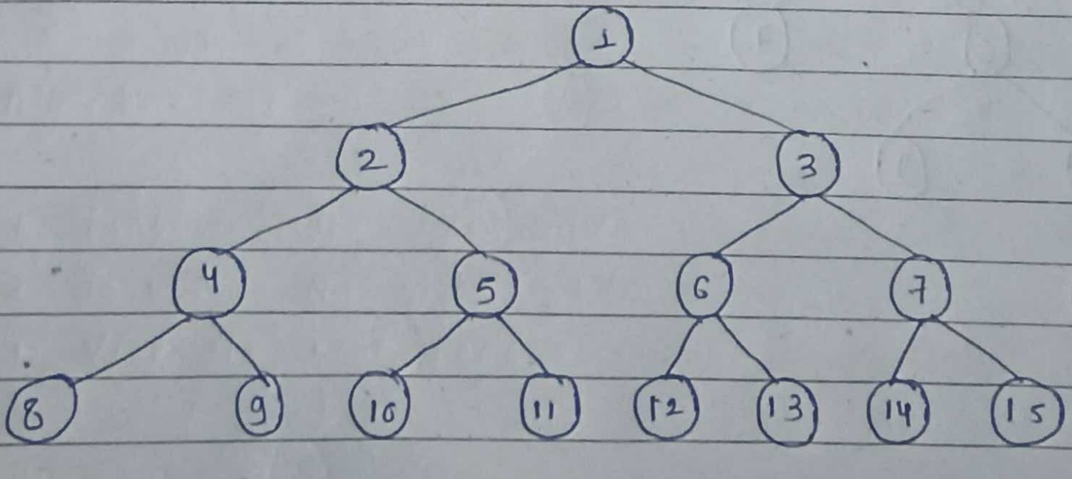
Information	-	data item
Left	-	address of left child.
Right	-	adress of right child.
Root	-	Location of root node.





Binary Tree Representation

- A binary tree data structure is a hierarchial data structure in which each node has at most two children, reffered to as a left or right child ..
- It is commonly used in computer science for efficient storage and retrival of data with various operation such as insertion, deletion and traversal



Application :-

1> Binary trees can be used to implement efficient sorting algorithm.

2> Binary trees can be used to store data in a database system with each node representing a record.

3> Binary trees can be used to implement file system.

4> Binary trees can be used to implement game AI.

5> Binary trees can be used to implement decision trees.

Traversal of Binary tree

Traversal of Binary tree refer to visiting all the nodes of the tree in a specific order. There are several type of traversal method for binary tree are as follows :-

(1) In Order Traversal (Left , Root , Right)

In this traversal method , the left subtree is visited first followed by the root and then right subtree

Algorithm

- Traverse the left subtree.
- Visit the root node.
- Traverse the right sub tree.

② Pre Order Traversal (Root, Left, Right)

In this method the root node is visited first followed by the left subtree and then right subtree.

Algorithm

- Visit the root node.
- Traverse the left subtree.
- Traverse the right subtree.

③ Post Order Traversal (Left, Right, Root)

Here the left subtree is visited first followed by the right subtree and the root node is visited last.

Algorithm

- Traverse the left subtree
- Traverse the right sub tree
- Vist the root node.

Threaded Binary Tree

- A threaded binary tree is a variant of binary tree where null pointer in the nodes are replaced by the pointer to the in-order predecessor or successor.

- This allows for more efficient in-order traversal of tree without using stack or recursion.

Types

1) Single Threaded B.T

Either the right or left null pointer are used to store, reference to inorder predecessor or successor.

2) Double Threaded B.T

Both left and right null pointer are used to store reference to the in-order predecessor and successor respectively.

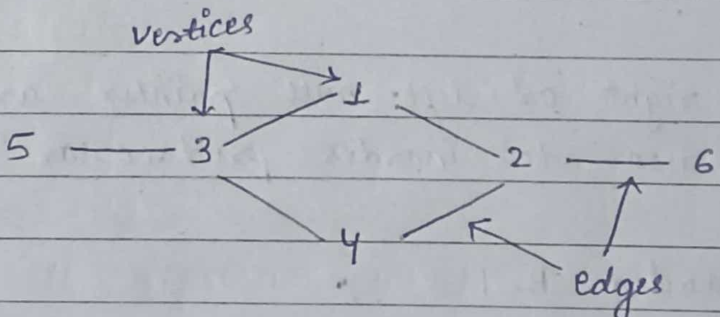
Page No.
Date

Graph

- Graph is a non - linear data structure consisting of vertices and edges.

- The vertices are sometime also reffered to as node and the edges are line or arcs that connect any two node in the graph.

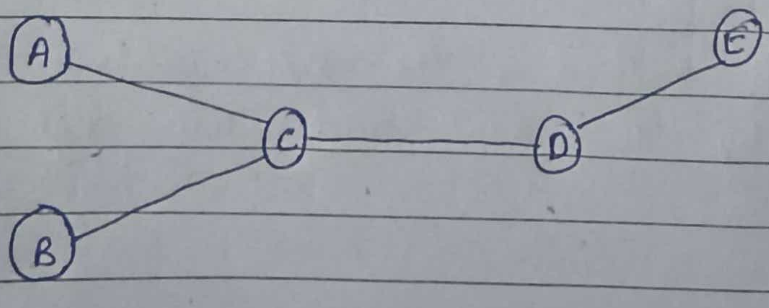
- formally a graph is composed of a set of vertices (V) and set of edges (E). The graph is denoted by G (V, E)



Types

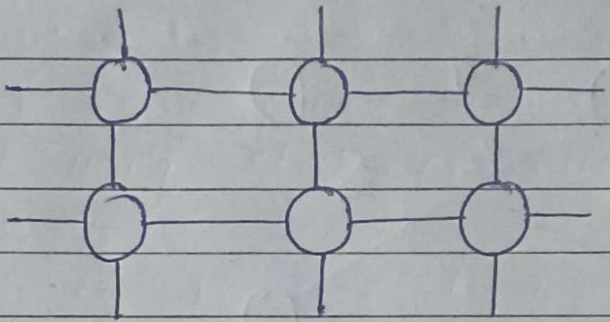
① finite Graph

A graph is said to be finite if it has a finite number of vertices and a finite number of edges.



2) Infinite Graph

A graph is said to be infinite if it has an infinite number of vertices and infinite number of edeges.

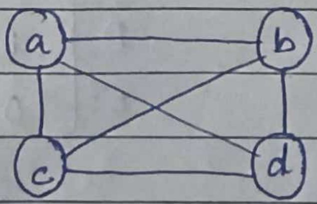


3) Trival Graph

A graph is said to be trival if a finite graph contain only one vertex and no edge also Known as a singleton graph.

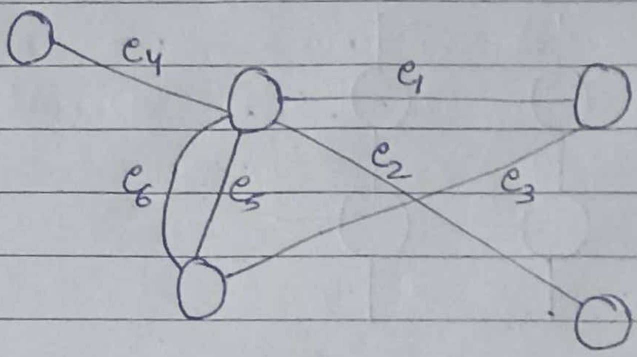
4) Simple Graph

A simple graph is a graph that does not contain more than one edge between the pair of vertices.



⑤ Multiple Graph

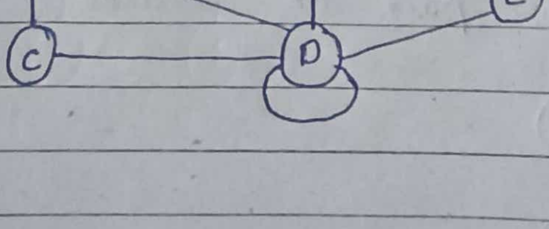
Any graph which contain some parallel edges but doesnot contain any self loop is called multiple gra



Adjacency Matrices

- Adjacency matrices is a sequential representation.
- It is used to represent which nodes are adjacent to each node.

Ex :-

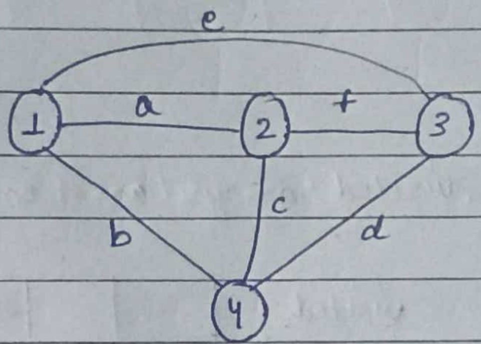


$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 1 & 1 & 0 \\ B & 1 & 0 & 0 & 1 & 1 \\ C & 1 & 0 & 0 & 1 & 0 \\ D & 1 & 1 & 1 & 1 & 1 \\ E & 0 & 1 & 0 & 1 & 0 \end{array}$$

Incident Matrices.

- In this graph can be represented using a matrix of size

- Total no. of vertices by total no. of edges, it means if a graph has 4 vertices and 6 edges then it can be represented using a matrix 4 × 6 class.



$$\begin{array}{c|cccccc} & a & b & c & d & e & f \\ \hline 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 & 1 & 1 & 1 \\ 4 & 0 & 1. & 1 & 1 & 0 & 1 \end{array}$$

Graph Traversal

Graph traversal is a technique for finding a vertex in a graph. It is also used to determine the order in which vertices are visited throughout the search process.

we can traverse a graph in two ways :-

→ BFS (Breadth first Search)

→ DFS (Depth first Search)

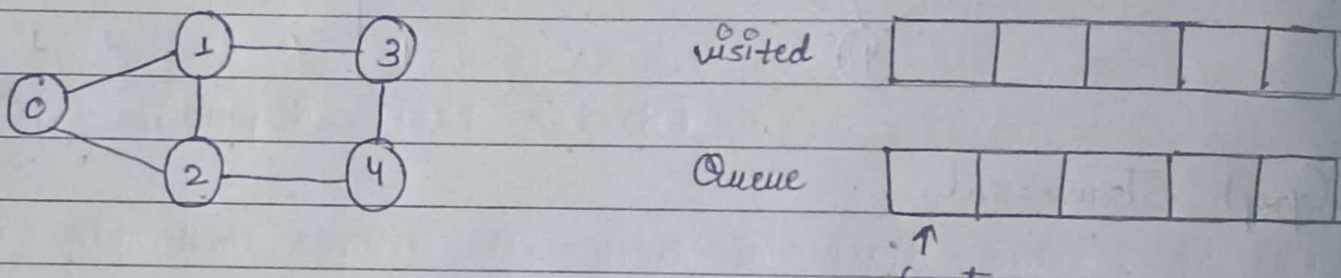
(1) BFS (Breadth first Search)

BFS is a technique for visiting all nodes in a given network. This traversal algorithm selects a node and visit all nearby nodes in order. After checking all nearby vertices, examine another set vertices, then recheck adjacent vertices.

Example :-

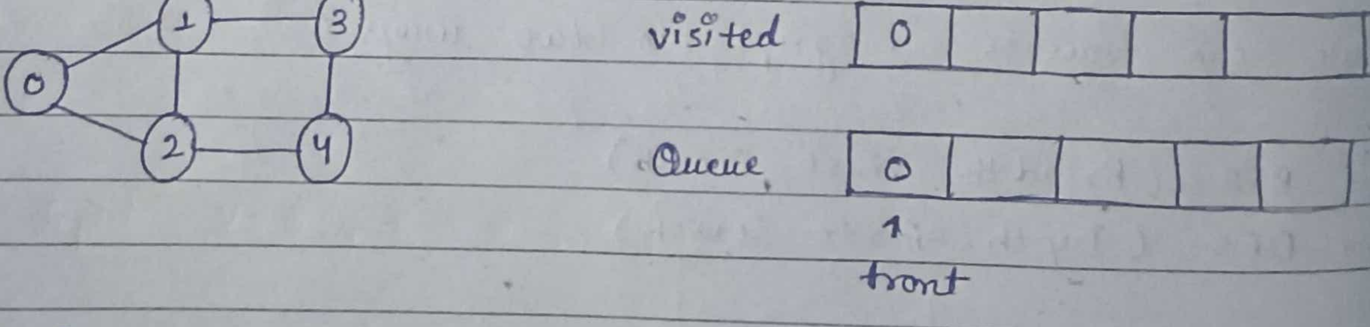
Step 1 :-

Initially queue and visited arrays are empty



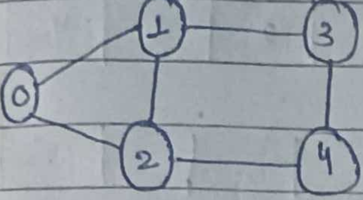
Step 2 :-

Push 0 into the queue and mark it visited



Step 3 :-

Remove 0 from the front of queue and visit the unvisited neighbour and push them into queue.



visited

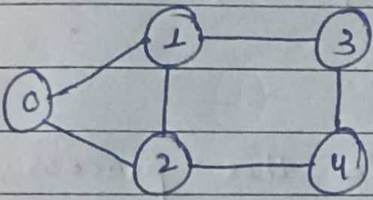
0	1	2		

Queue

1	2			

↑
front

Step 4 :-
Remove 1 from the queue and visit the unvisited neighbour and push them into queue.



visited

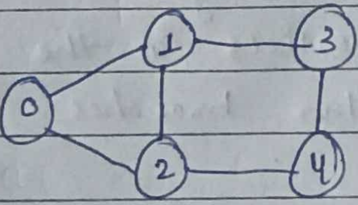
0	1	2	3	

Queue

2	3			

↑
front

Step 5 :-
Remove 2 from the queue and visit the unvisited neighbours and push them into queue



visited

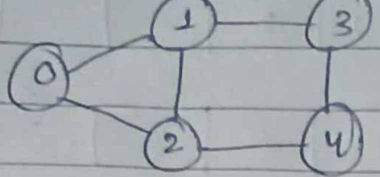
0	1	2	3	4

Queue

3	4			

↑
front

Step 6 :-
~~As we can see~~ Remove 3 from the queue and as we can see every neighbour of 3 is visited so move the next node that are in the front of queue.



visited

0	1	2	3	4

Queue

4				

↑
front

Step 7 :-

Remove 4 from the queue.

Queue

↑
front

Now, Queue becomes empty, so terminate this process of iteration.

② DFS (Depth first Search)

This algorithm explores the graph in depth order, starting with a given source node and then recursively visiting all of its surrounding vertices before backtracking.

DFS will analyze the deepest vertices in the branch of graph before moving on to other branches.

Example :-

Input V = 5, E = 5, ~~edges~~

Edges = {(1,2), (1,0), (0,2), (2,3), (2,4)}

Source 1

Steps :-

- Start at 1

  Mark as visited. Output 1
- Move to 2

  Mark as visited. Output 2
- Move to 0

  Mark as visited. Output 0 (back-track to 2)
- Move to 3

  Mark as visited. Output 3 (back track to 2)
- Move to 4

  Mark as visited. Output 4 (back track to 1)

Output :- 1 2 0 3 4