Unit V

Unit V Graphs: Definition and Terminology, Representation Techniques, Graph Traversal Algorithms: BFS and DFS.

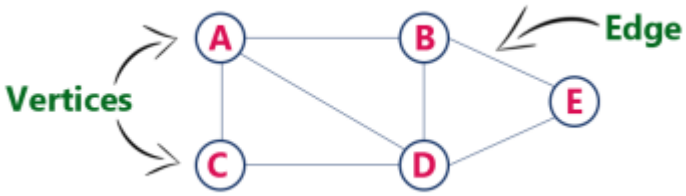
Definition and Terminology of Graph:

- A graph is a non-linear data structure used to model pairwise relationships between objects.
- It consists of a finite set of points (vertices) and a set of lines (edges) connecting them.

A graph is defined as Graph is a collection of vertices and arcs which connects vertices in the graph. A graph G is represented as G = (V , E), where V is set of vertices and E is set of edges.

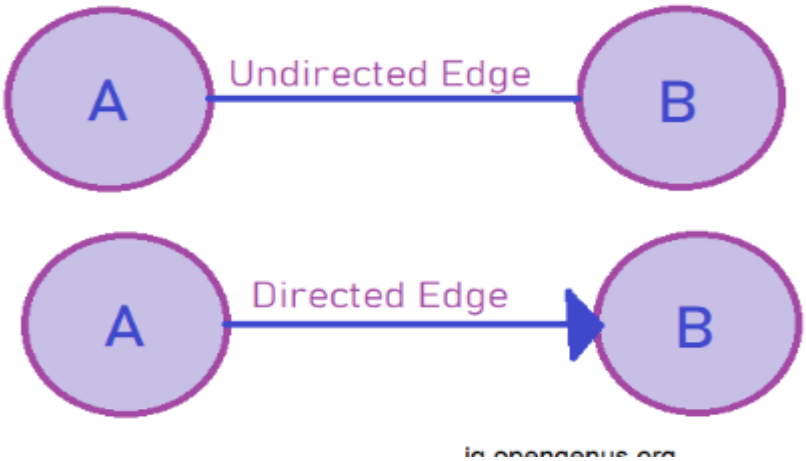
Example: graph G can be defined as G = (V , E) Where V = {A,B,C,D,E} and

E = {(A,B),(A,C)(A,D),(B,D),(C,D),(B,E),(E,D)}. This is a graph with 5 vertices and 7 edges.

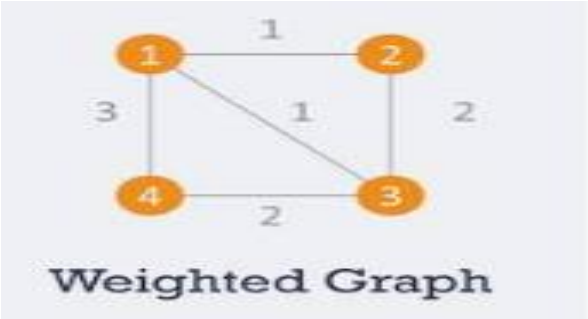


Graph Terminology

1.Vertex : An individual data element of a graph is called as Vertex. Vertex is also known as node. In above
example graph, A, B, C, D & E are known as vertices.
2.Edge : An edge is a connecting link between two vertices. Edge is also known as Arc. An edge is represented as
(starting Vertex, ending Vertex).
In above graph, the link between vertices A and B is represented as (A,B).
Edges are three types:
1.Undirected Edge - An undirected edge is a bidirectional edge. If there is an undirected edge between vertices A
and B then edge (A , B) is equal to edge (B , A).
2.Directed Edge - A directed edge is a unidirectional edge. If there is a directed edge between vertices A and B
then edge (A , B) is not equal to edge (B , A)

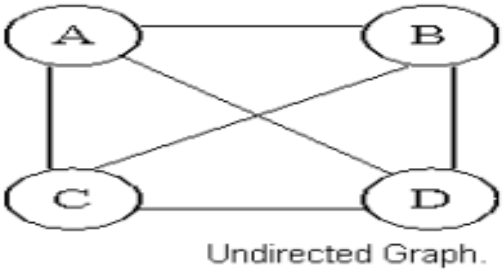


3.Weighted Edge - A weighted edge is an edge with cost on it.

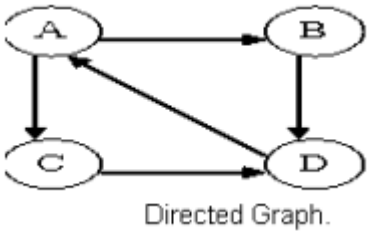


Types of Graphs:

1.Undirected Graph:A graph with only undirected edges is said to be undirected graph.

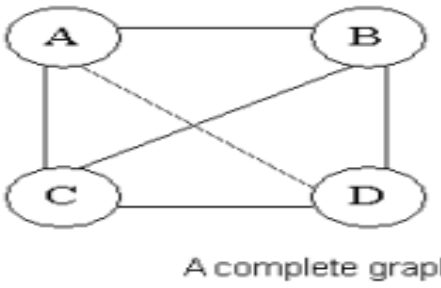


2.Directed Graph: A graph with only directed edges is said to be directed graph.



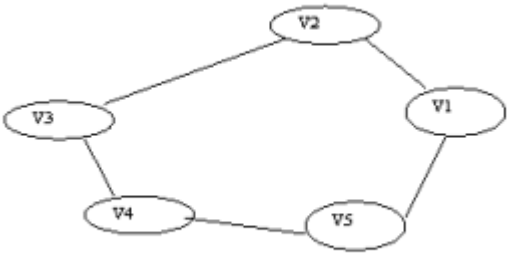
3.Complete Graph: A graph in which any V node is adjacent to all other nodes present in the graph is known as a complete graph.

The following figure shows a complete graph.



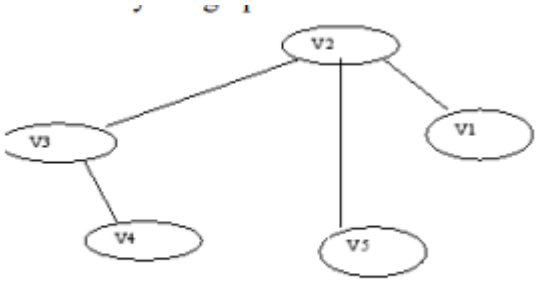
.

4.Cycle Graph: A graph having cycle is called cycle graph. In this case the first and last nodes are the same. A closed simple path is a cycle.



A cycle graph

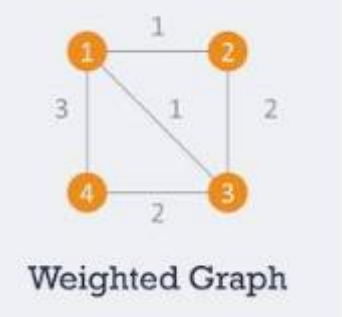
5.Acyclic Graph: A graph without cycle is called acyclic graphs.



A acyclic graph

6. Weighted Graph: A graph is said to be weighted if there are some non negative value assigned to each edges of the graph.

The value is equal to the length between two vertices. Weighted graph is also called a network.



Outgoing Edge: A directed edge is said to be outgoing edge on its origin vertex.

Incoming Edge: A directed edge is said to be incoming edge on its destination vertex.

Degree: Total number of edges connected to a vertex is said to be degree of that vertex.

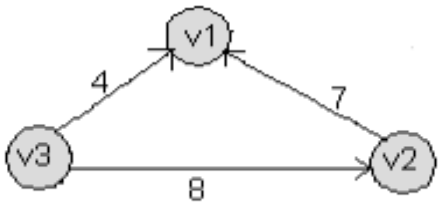
Indegree: Total number of incoming edges connected to a vertex is said to be indegree of that vertex.

Outdegree: Total number of outgoing edges connected to a vertex is said to be outdegree of that vertex.

Parallel edges or Multiple edges: If there are two undirected edges to have the same end vertices, and for two directed edges to have the same origin and the same destination. Such edges are called parallel edges or multiple edges.

Self-loop: An edge (undirected or directed) is a self-loop if its two endpoints coincide.

Simple Graph: A graph is said to be simple if there are no parallel and self-loop edges.



A weighted graph

Adjacent nodes: When there is an edge from one node to another then these nodes are called adjacent nodes.

Applications of Graph:

- Transportation Systems: Google Maps employs graphs to map roads, where intersections are vertices and roads are edges. It calculates shortest paths for efficient navigation.
- Social Networks: Platforms like Facebook model users as vertices and friendships as edges, using graph theory for friend suggestions.
- World Wide Web: Web pages are vertices, and links between them are directed edges, inspiring Google's Page Ranking Algorithm.
- Resource Allocation and Deadlock Prevention: Operating systems use resource allocation graphs to prevent deadlocks by detecting cycles.
- Mapping Systems and GPS Navigation: Graphs help in locating places and optimizing routes in mapping systems and GPS navigation.
- Graph Algorithms and Measures: Graphs are analyzed for structural properties and measurable quantities, including dynamic properties in networks.

****

Representation of Graph /Graph Representation:

A Graph is a non-linear data structure consisting of vertices and edges. The vertices are sometimes also referred to as nodes and the edges are lines or arcs that connect any two nodes in the graph.

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

Representations of Graph

There are two most common ways to represent a graph : For simplicity, we are going to consider only **unweighted graphs** in this post.

1. Adjacency Matrix
2. Adjacency List

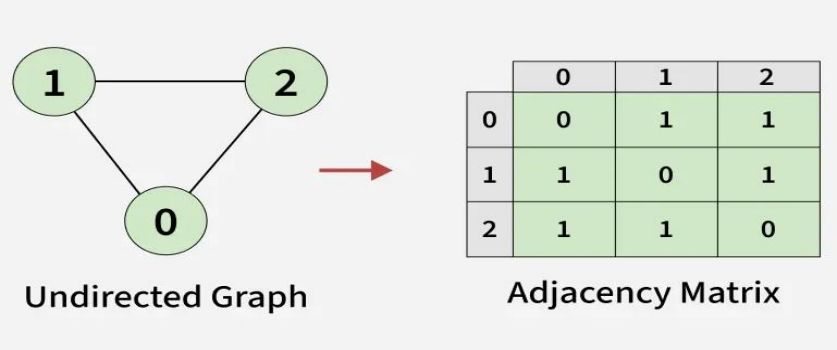
1.Adjacency Matrix Representation

An adjacency matrix is a way of representing a graph as a boolean matrix of (0's and 1's).

Let's assume there are n vertices in the graph So, create a 2D matrix adjMat[n][n] having dimension n x n.

- If there is an edge from vertex i to j, mark adjMat[i][j] as 1.
- If there is no edge from vertex i to j, mark adjMat[i][j] as 0.

Representation of Undirected Graph as Adjacency Matrix:



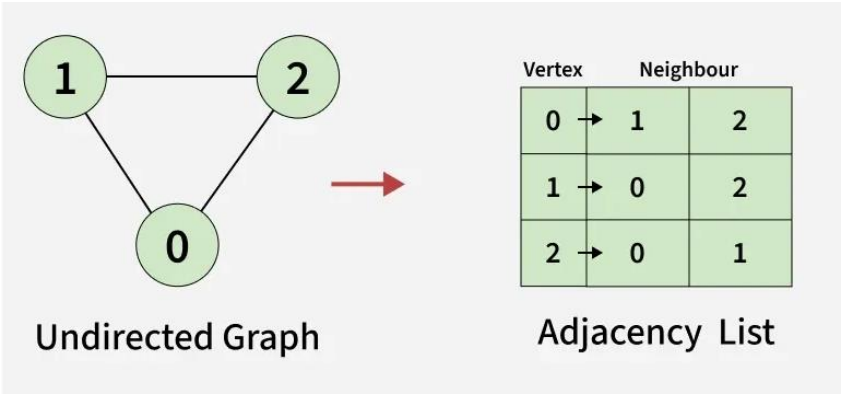
	0	1	2
0	0	1	1
1	1	0	1
2	1	1	0

- We use an adjacency matrix to represent connections between vertices.
- Initially, the entire matrix is filled with 0s, meaning no edges exist.
- There is an edge between vertex 0 and vertex 1,so we set mat[0][1] = 1 and mat[1][0] = 1.
- There is an edge between vertex 0 and vertex 2,so we set mat[0][2] = 1 and mat[2][0] = 1.
- There is an edge between vertex 1 and vertex 2,so we set mat[1][2] = 1 and mat[2][1] = 1.

2.Adjacency List Representation:

- An array of Lists is used to store edges between two vertices. The size of array is equal to the number of vertices (i.e, n).
- Each index in this array represents a specific vertex in the graph.
- The entry at the index i of the array contains a linked list containing the vertices that are adjacent to vertex i.
- Let's assume there are n vertices in the graph So, create an array of list of size n as adjList[n].
  - adjList[0] will have all the nodes which are connected (neighbour) to vertex 0.
  - adjList[1] will have all the nodes which are connected (neighbour) to vertex 1 and so on.

Representation of Undirected Graph as Adjacency list:



- We use an array of lists (or vector of lists) to represent the graph.
- The size of the array is equal to the number of vertices (here, 3).
- Each index in the array represents a vertex.
- Vertex 0 has two neighbours (1 and 2).

- Vertex 1 has two neighbours (0 and 2).
- Vertex 2 has two neighbours (0 and 1).

****

Graph Traversal Algorithms: BFS and DFS:

Graph Traversal is the process of visiting (exploring) all the vertices (nodes) of a graph in a systematic way, starting from a particular node, so that each node is visited at least once.

It is mainly used in data structures to **search, analyze, and process graphs**.

There are two main types:

Type	Description	Data Structure Used
Breadth First Search (BFS)	Visits nodes **level by level** (all neighbours first, then next level)	Queue
Depth First Search (DFS)	Visits nodes **as deep as possible** before backtracking	Stack / Recursion

1. **Breadth First Search (BFS):**

- Breadth First Search (BFS) is a graph traversal algorithm that starts from a source node and explores the graph level by level.
- First, it visits all nodes directly adjacent to the source.
- Then, it moves on to visit the adjacent nodes of those nodes, and this process continues until all reachable nodes are visited**.**

Algorithm of DFS

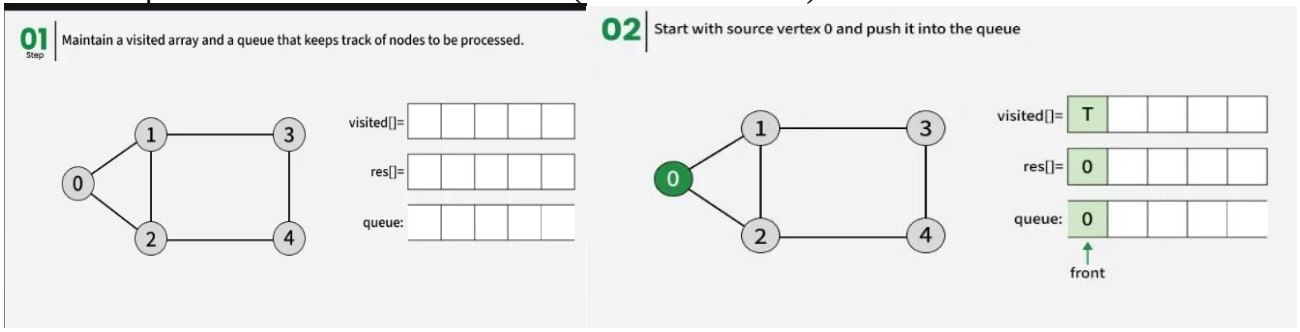
Step 1: Take an Empty Queue.
Step 2: Select a starting node insert it into the Queue.
Step 3: dequeue the element mark it as visiting and insert its unvisited adjacent vertex it into the Queue.(if already in queue ignore it)
Step 4: repeat step 2 and 3 until all vertices are visited.

visited[] Array
Purpose: Keeps track of which nodes have already been explored.
res[] (Result Array)

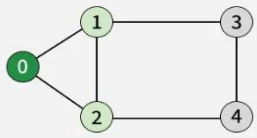
- Purpose: Stores the order in which nodes are visited.
- Initially: Empty ([ ]).

queue

- Purpose: Core data structure of BFS (First-In-First-Out).



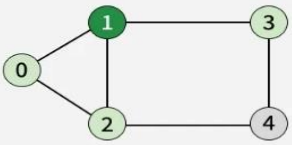
03 Process node 0, and push its unvisited neighbours into the queue.



visited[]=	T	T	T		
res[]=	0				
queue:	1	2			

front

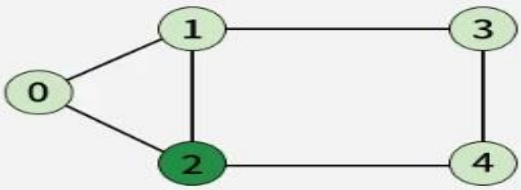
04 Process node 1, and push its unvisited neighbours into the queue.



visited[]=	T	T	T	T	
res[]=	0	1			
queue:	2	3			

front

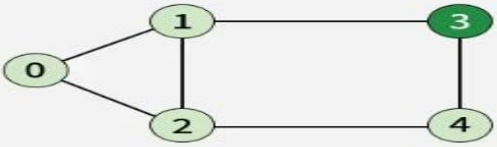
05 Process node 2, and push its unvisited neighbours into the queue.



visited[]=	T	T	T	T	T
res[]=	0	1	2		
queue:	3	4			

front

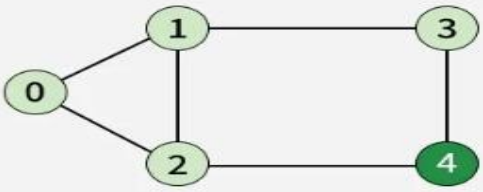
06 Process node 3, and push its unvisited neighbours into the queue.



visited[]=	T	T	T	T	T
res[]=	0	1	2	3	
queue:	4				

front

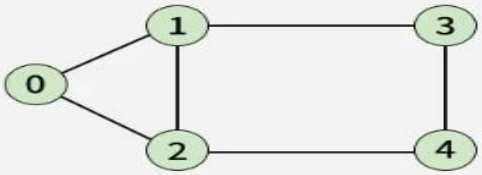
07 Process node 4, and push its unvisited neighbours into the queue.



visited[]=	T	T	T	T	T
res[]=	0	1	2	3	
queue:	4				

front

08 All the neighbours of 4 have been visited, proceed to the next node in the queue.Now that the queue is empty and there are no more nodes to process, we obtain the final BFS traversal.



visited[]=	T	T	T	T	T
res[]=	0	1	2	3	4
queue:					

BFS traversal order is 0, 1, 2, 3, 4.

Applications of BFS in graphs:

Shortest Path → Finds the shortest path in unweighted graphs.
Cycle Detection → Helps check if a graph contains cycles.
Connected Components → Identifies all nodes connected together in a graph.
Network Routing → Used to find efficient paths for data packets in networks.

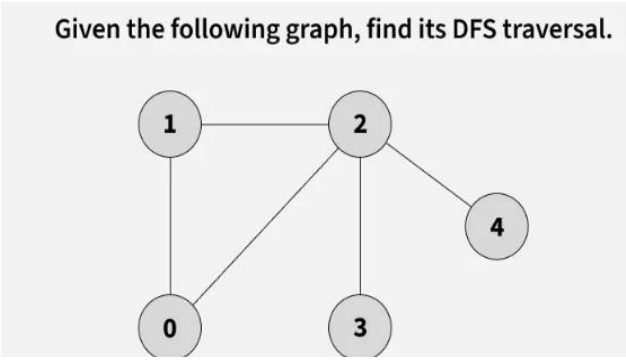
2.Depth First Search (DFS):

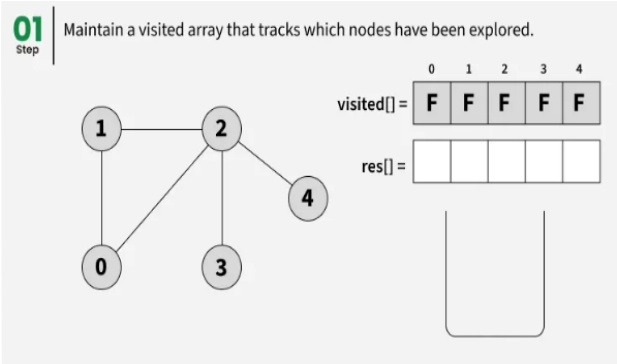
- Depth First Search (DFS) is a graph traversal algorithm that starts from a source node and explores as far as possible along each branch before backtracking.

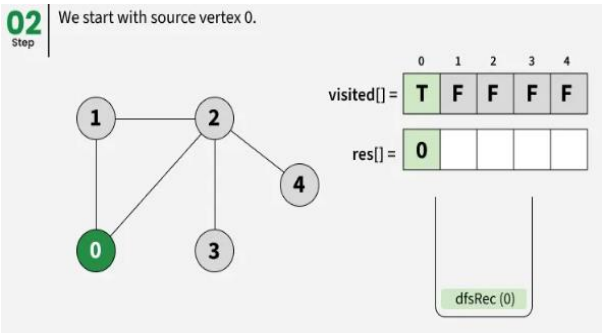
DFS Algorithm :
Step1:Start with a stack
- Take an empty stack (or use recursion).

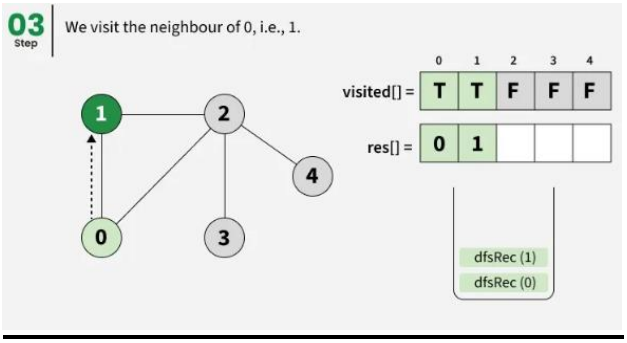
Step2:Pick a starting node
- Push it into the stack and while pop mark it visited Process the node
- Record or display it. and push its adjacent vertices in the stack

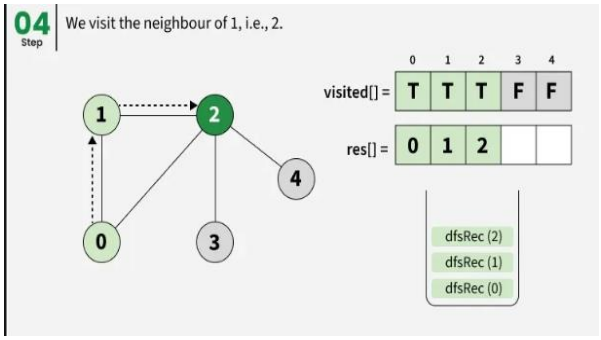
Step3:pop top element and repeat the process of step2.
Step4:Repeat until stack is empty
- Continue until all nodes are visited.

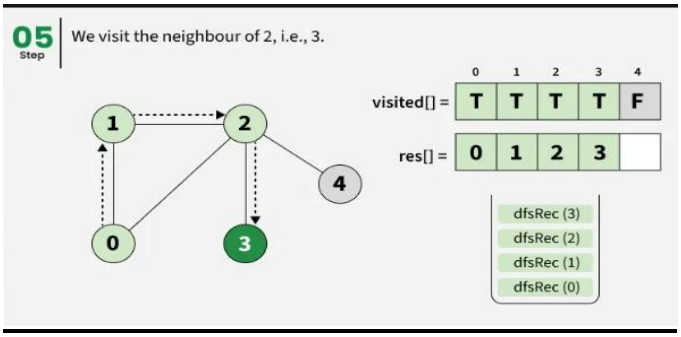


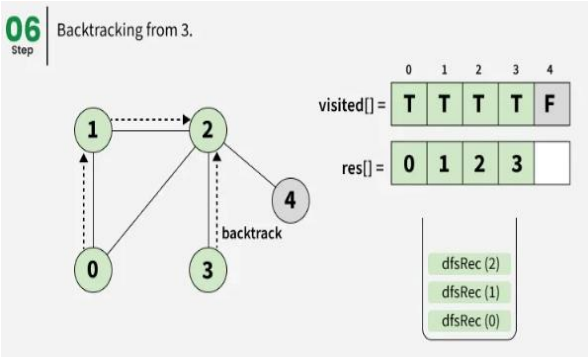


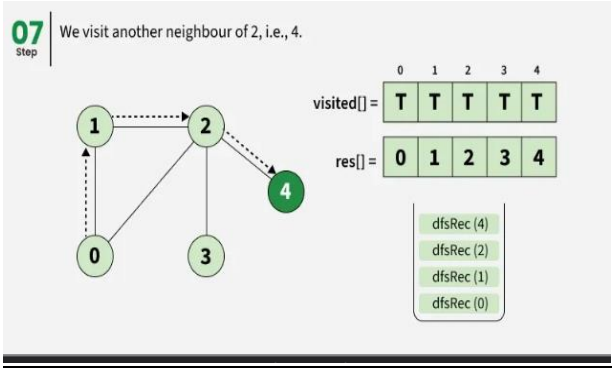












Applications of DFS in graphs:

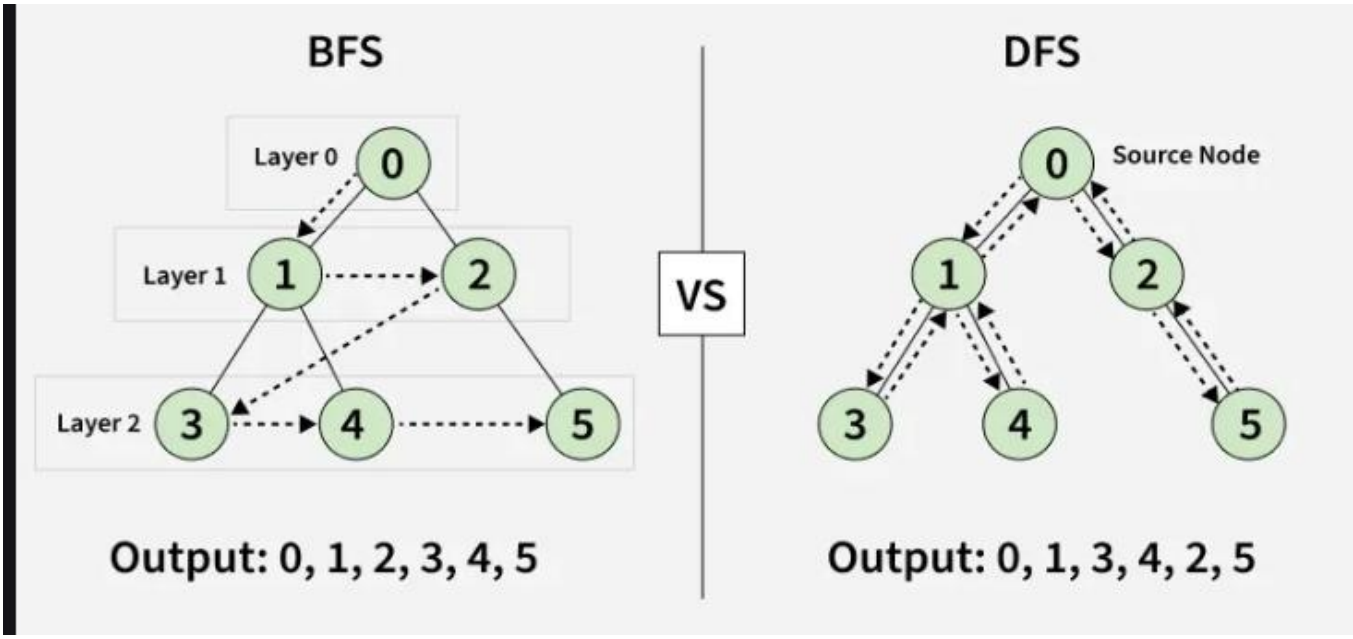
Path Finding → Useful for exploring paths deeply, often used in puzzles and mazes.
Cycle Detection → Helps detect cycles in both directed and undirected graphs.
Topological Sorting → DFS is used to order tasks in dependency graphs.
Connected Components → Finds all connected parts of a graph by exploring deeply.
Strongly Connected Components (SCCs)

****

Difference between BFS and DFS:

Breadth-First Search (BFS) and Depth-First Search (DFS) are two fundamental algorithms used for traversing or searching graphs and trees.

Parameters	BFS (Breadth First Search)	DFS (Depth First Search)
Stands for	Breadth First Search	Depth First Search
Data Structure	Uses **Queue**	Uses **Stack**
Definition	Traverses nodes level by level, visiting all nodes at the same level before moving to the next level	Traverses from root and goes as deep as possible before backtracking
Conceptual Difference	Builds tree **level by level**	Builds tree **sub-tree by sub-tree**
Approach Used	Follows **FIFO (First In First Out)**	Follows **LIFO (Last In First Out)**
Suitable for	Best for searching nodes **closer to the source**	Best for searching nodes **far from the source**
Applications	Shortest path (when edges have equal weight), Bipartite graph checking	Cycle detection, Strongly Connected Components, Topological sorting
Traversal Order	Level-wise traversal	Depth-wise traversal
Memory Usage	Requires more memory (stores all nodes at current level)	Requires less memory (stores only path nodes)



****

UNIT-5

SHORT

1. List the types of Graphs.
2. Define a Graphs.
3. Graph traversal techniques.
4. Define DFS.
5. Define BFS.

LONG

1. Define Graph and explain the various representation of Graph.

   (OR)

   Explain Representation of graph in brief.
2. Explain various types of graphs.
3. Explain about DFS Graph Traversal algorithm with an example.
4. Discuss about BFS algorithm with an example.
5. Write the difference between DFS and BFS