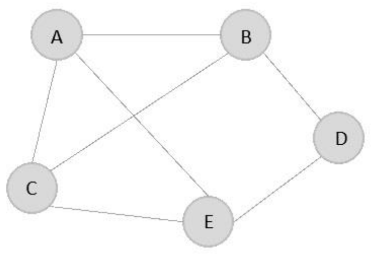
UNIT-5
GRAPHS

Introduction to Graph:-

Graph is a ADT and it is a collection of nodes connected by edges. It's used to represent relationships between different entities. Graph algorithms are methods used to manipulate and analyze graphs, solving various problems like finding the shortest path or detecting cycles.

Definition:-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).**



G(V, E)

V(G)={A,B,C,D,E} Means collection of vertices

E(G)=(A,B),(A,C),(A,E),(B,D),(B,C),(C,E),(D,E)} Means collection of edges

Basic Graph Terminology

1. Graph

A Graph **G** is a non-empty set of vertices (or nodes) **V** and a set of edges **E**, where each edge connects a pair of vertices. Formally, a graph can be represented as **G= (V, E)**. Graphs can be classified based on various properties, such as directedness of edges and connectivity.

2. Vertex (Node)

A Vertex, often referred to as a **Node**, is a fundamental unit of a graph. It represents an **entity** within the graph

3. Edge

An Edge is a **connection between two vertices** in a graph. It can be either directed or undirected

4. Degree of a Vertex

The Degree of a Vertex in a graph is the **number of edges incident** to that vertex. In a directed graph, the degree is further categorized into the in-degree (number of incoming edges) and out-degree (number of outgoing edges) of the vertex.

5. Path

A Path in a graph is a **sequence of vertices** where each adjacent pair is connected by an edge. Paths can be of varying lengths and may or may not visit the same vertex more than once. The shortest path between two vertices is of particular interest in algorithms such as Dijkstra's algorithm for finding the shortest path in weighted graphs.

6. Cycle

A Cycle in a graph is a **path that starts and ends at the same vertex**, with no repetitions of vertices (except the starting and ending vertex, which are the same).

7. Directed Graph (Digraph):

A Directed Graph consists of nodes (vertices) connected by directed edges (arcs). Each edge has a specific direction, meaning it goes from one node to another. Directed Graph is a network where information flows in a specific order.

8. Undirected Graph:

In an Undirected Graph, edges have no direction. They simply connect nodes without any inherent order

9. Weighted Graph:

Weighted graphs assign numerical values (weights) to edges. These weights represent some property associated with the connection between nodes.

10. Acyclic Graph:

An acyclic graph contains no cycles (closed loops). In other words, you cannot start at a node and follow edges to return to the same node.

11. Cyclic Graph:

A cyclic graph has at least one cycle. You can traverse edges and eventually return to the same node.

REPRESENTATION OF GRAPH

While representing graphs, we must carefully depict the elements (vertices and edges) present in the graph and the relationship between them. Pictorially, a graph is represented with a finite set of nodes and connecting links between them. However, we can also represent the graph in other most commonly used ways, like

1. Adjacency Matrix
2. Adjacency list

Adjacency Matrix Graph Representation

- Adjacency matrix is a sequential representation.
- It is used to represent which nodes are adjacent to each other. i.e. is there any edge connecting nodes to a graph.
- In this representation, we have to construct a nXn matrix A. If there is any edge from a vertex i to vertex j, then the corresponding element of A, $a^{i,j}$ = 1, otherwise $a^{i,j}$= 0.

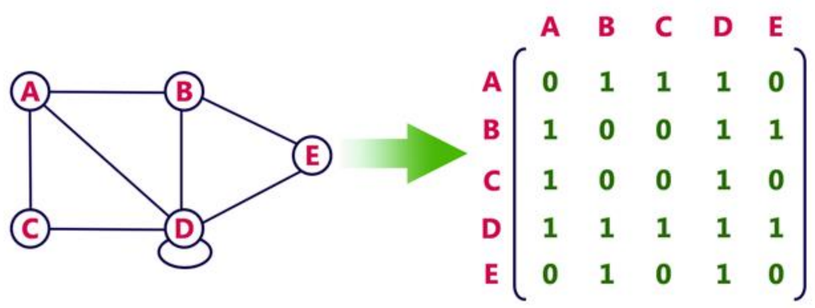
Note, even if the graph on 100 vertices contains only 1 edge, we still have to have a 100x100 matrix with lots of zeroes.

- If there is any weighted graph then instead of 1s and 0s, we can store the weight of the edge.

Example

Consider the following **undirected graph representation**:

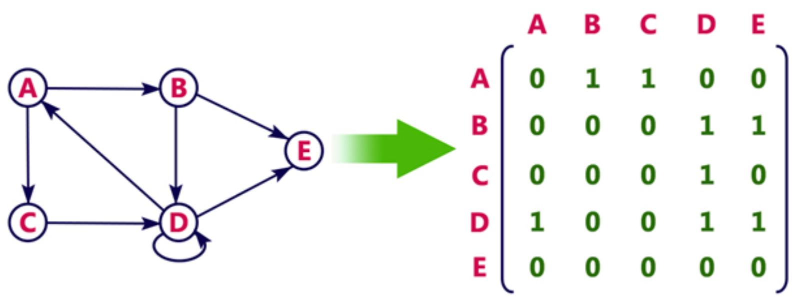
Undirected graph representation



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

Directed graph represenation

See the directed graph representation:



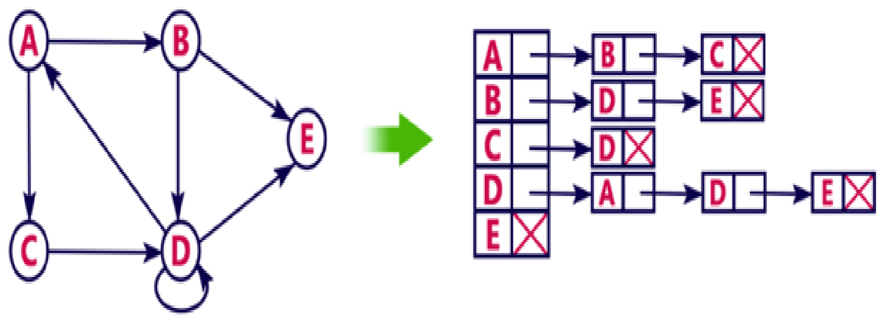
$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 1 & 0 & 0 \\ B & 0 & 0 & 0 & 1 & 1 \\ C & 0 & 0 & 0 & 1 & 0 \\ D & 1 & 0 & 0 & 1 & 1 \\ E & 0 & 0 & 0 & 0 & 0 \end{array}$$

In the above examples, 1 represents an edge from row vertex to column vertex, and 0 represents no edge from row vertex to column vertex.

Adjacency List Graph Representation:

- Adjacency list is a linked representation.
- In this representation, for each vertex in the graph, we maintain the list of its neighbors. It means, every vertex of the graph contains list of its adjacent vertices.
- We have an array of vertices which is indexed by the vertex number and for each vertex v, the corresponding array element points to a **singly linked list** of neighbors of v.

Let's see the following directed graph representation implemented using linked list:

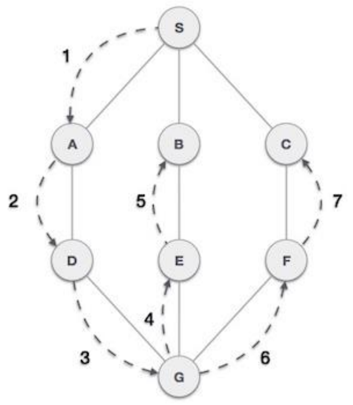


- Adjacency list saves lot of space.
- We can easily insert or delete as we use linked list.
- Such kind of representation is easy to follow and clearly shows the adjacent nodes of node.

GRAPH TRAVERSALS

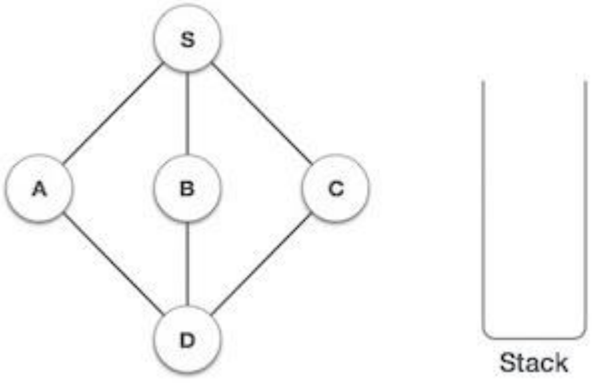
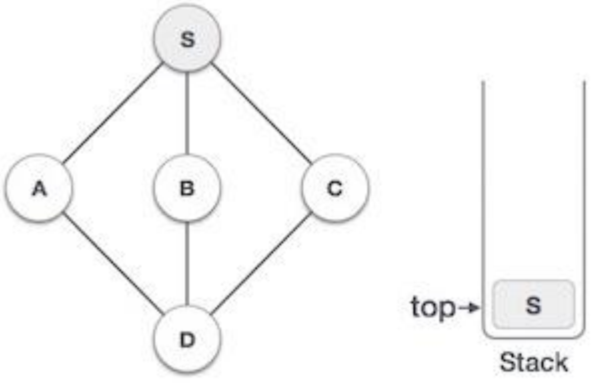
Depth First Search (DFS) Algorithm

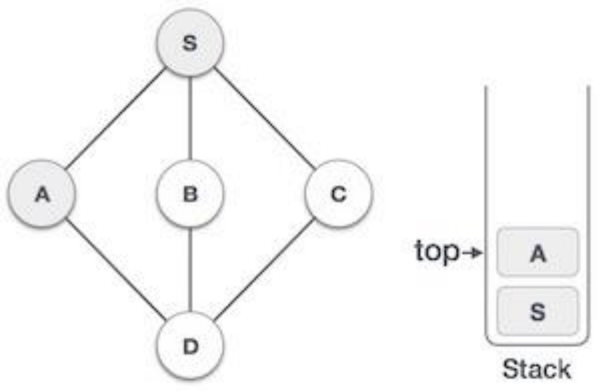
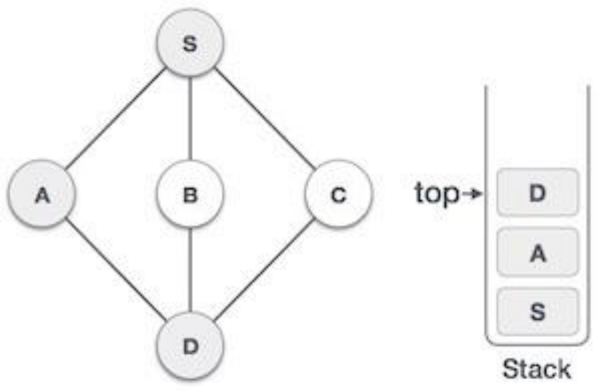
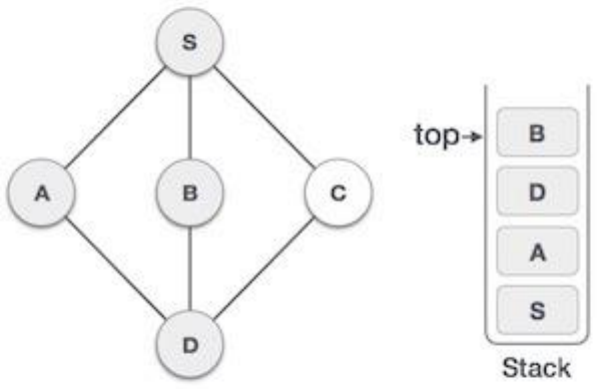
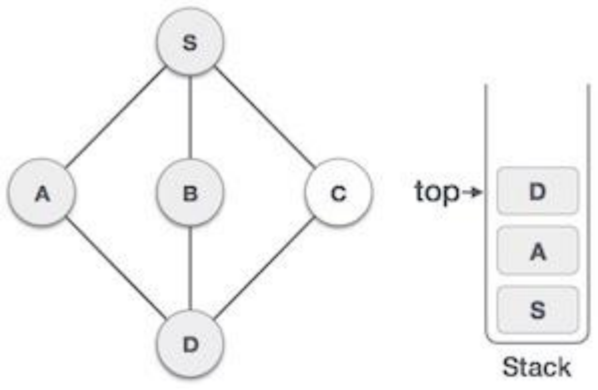
The DFS traversal uses the stack data structure to keep track of the unvisited nodes. Depth First Search (DFS) algorithm is a recursive algorithm for searching all the vertices of a graph or tree data structure. This algorithm traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

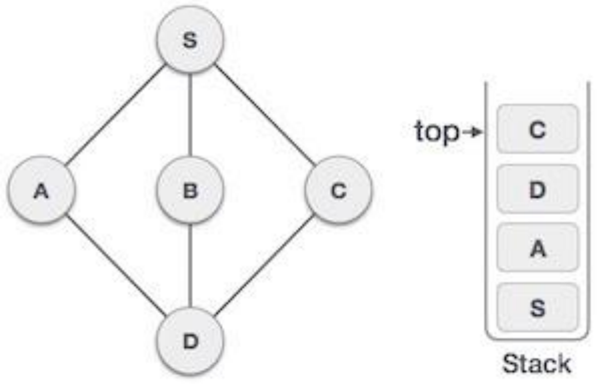


It employs the following rules.

- **Rule 1** – Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack.
- **Rule 2** – If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the vertices from the stack, which do not have adjacent vertices.)
- **Rule 3** – Repeat Rule 1 and Rule 2 until the stack is empty.

Step	**Traversal**	**Description**
1		Initialize the stack.
2		Mark **S** as visited and put it onto the stack. Explore any unvisited adjacent node from **S**. We have three nodes and we can pick any of them. For this example, we shall take the node in an alphabetical order.

3		Mark **A** as visited and put it onto the stack. Explore any unvisited adjacent node from A. Both **S** and **D** are adjacent to **A** but we are concerned for unvisited nodes only.
4		Visit **D** and mark it as visited and put onto the stack. Here, we have **B** and **C** nodes, which are adjacent to **D** and both are unvisited. However, we shall again choose in an alphabetical order.
5		We choose **B**, mark it as visited and put onto the stack. Here **B** does not have any unvisited adjacent node. So, we pop **B** from the stack.
6		We check the stack top for return to the previous node and check if it has any unvisited nodes. Here, we find **D** to be on the top of the stack.

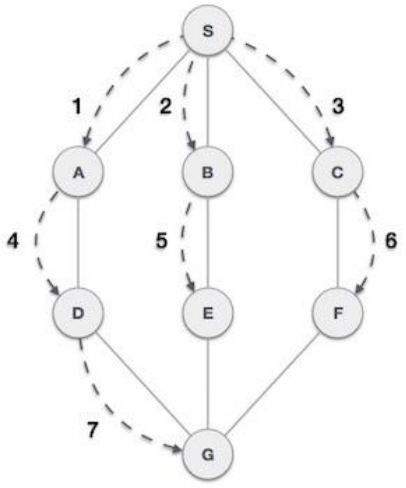
7		Only unvisited adjacent node is from **D** is **C** now. So we visit **C**, mark it as visited and put it onto the stack.

As **C** does not have any unvisited adjacent node so we keep popping the stack until we find a node that has an unvisited adjacent node. In this case, there's none and we keep popping until the stack is empty.

<u>**BREADTH FIRST SEARCH (BFS) ALGORITHM**</u>

The DFS traversal uses the queue data structure to keep track of the unvisited nodes. Breadth First Search (BFS) algorithm traverses a graph in a breadthward motion to search a graph data structure for a node that meets a set of criteria. It uses a queue to remember the next vertex to start a search, when a dead end occurs in any iteration.

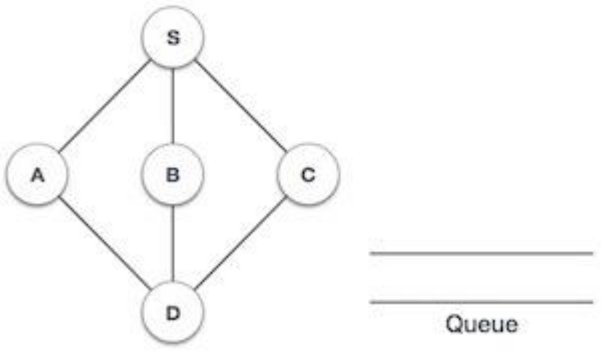
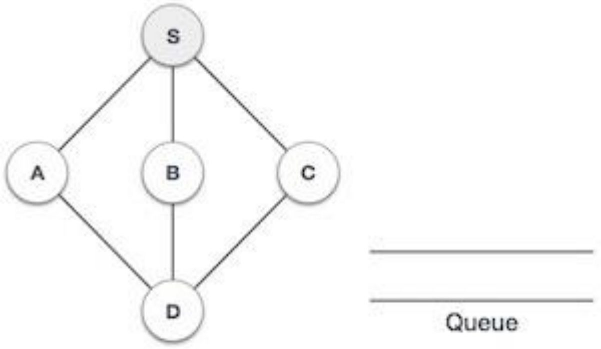
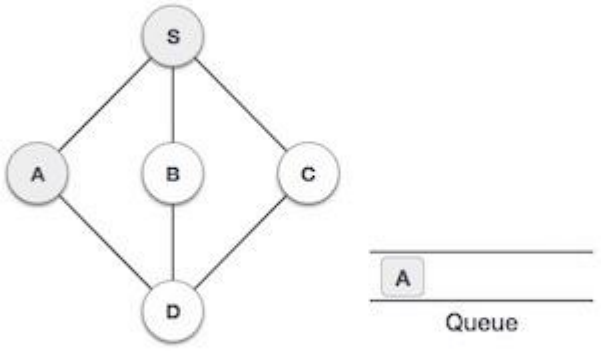
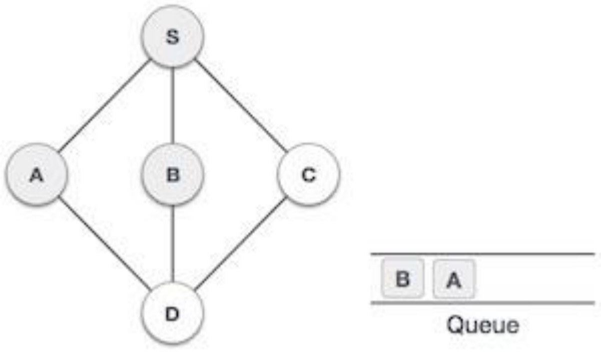
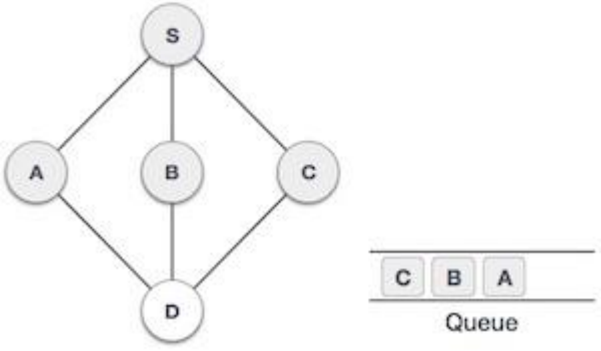
Breadth First Search (BFS) algorithm starts at the tree root and explores all nodes at the present depth prior to moving on to the nodes at the next depth level.

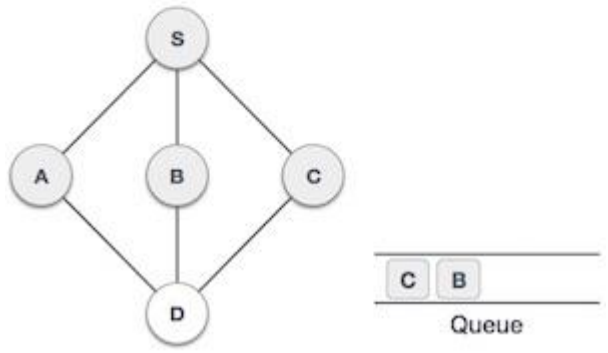
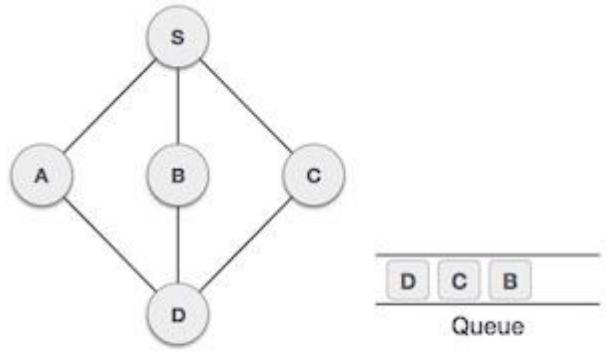


It employs the following rules.

- **Rule 1** − Visit the adjacent unvisited vertex. Mark it as visited. Display it. Insert it in a queue.
- **Rule 2** − If no adjacent vertex is found, remove the first vertex from the queue.
- **Rule 3** − Repeat Rule 1 and Rule 2 until the queue is empty.

Step	**Traversal**	**Description**

1		Initialize the queue.
2		We start from visiting **S** (starting node), and mark it as visited.
3		We then see an unvisited adjacent node from **S**. In this example, we have three nodes but alphabetically we choose **A**, mark it as visited and enqueue it.
4		Next, the unvisited adjacent node from **S** is **B**. We mark it as visited and enqueue it.
5		Next, the unvisited adjacent node from **S** is **C**. We mark it as visited and enqueue it.

6		Now, **S** is left with no unvisited adjacent nodes. So, we dequeue and find **A**.
7		From **A** we have **D** as unvisited adjacent node. We mark it as visited and enqueue it.

At this stage, we are left with no unmarked (unvisited) nodes. But as per the algorithm we keep on dequeuing in order to get all unvisited nodes. When the queue gets emptied, the program is over.

<u>Applications of Graph</u>

- **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.