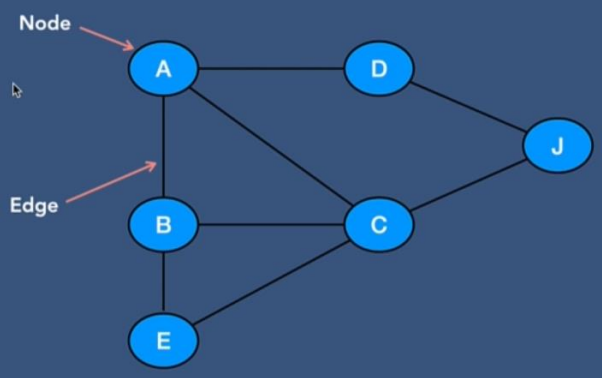
What you will learn

- What is a graph? Why do we need it?
- Graph Terminologies
- Types of graphs. Graph Representation
- Traversal of graphs. (BFS and DFS)
- Topological Sorting
- Single source shortest path (BFS, Dijkstra and Bellman Ford)
- All pairs shortest path (BFS, Dijkstra, Bellman Ford and Floyd Warshall algorithms)
- Minimum Spanning Tree (Kruskal and Prim algorithms)

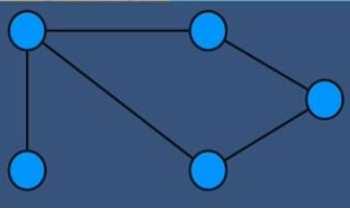
What is a Graph?

Graph consists of a finite set of Vertices(or nodes) and a set of Edges which connect a pair of nodes.



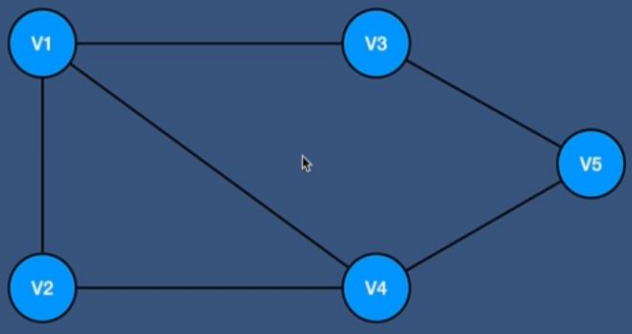
It is mostly used in social media like facebook, LinkedIn for finding friends and connection.

Why Graph?



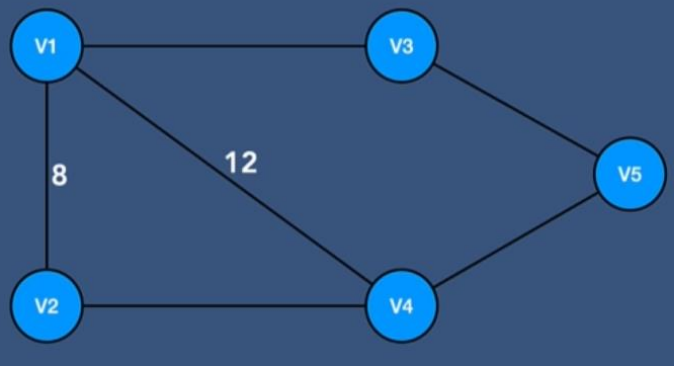
Graph Terminology

- **Vertices** : Vertices are the nodes of the graph
- **Edge** : The edge is the line that connects pairs of vertices
- **Unweighted graph** : A graph which does not have a weight associated with any edge



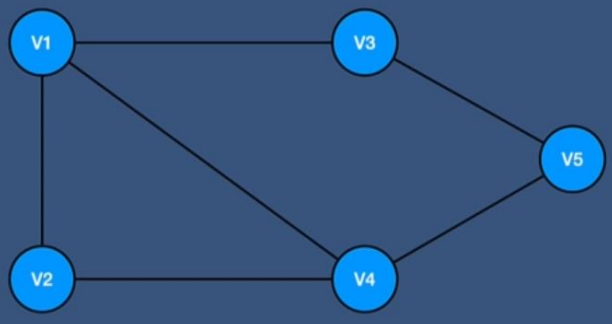
Graph Terminology

- **Vertices** : Vertices are the nodes of the graph
- **Edge** : The edge is the line that connects pairs of vertices
- **Unweighted graph** : A graph which does not have a weight associated with any edge
- **Weighted graph** : A graph which has a weight associated with any edge



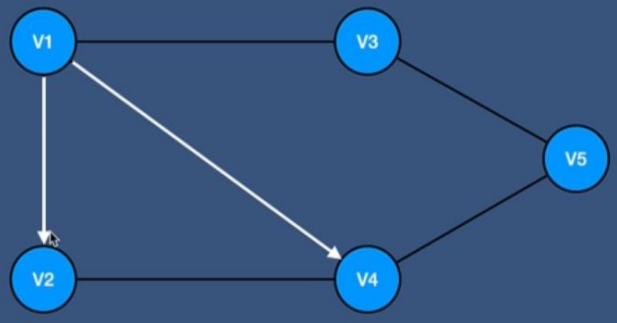
Graph Terminology

- **Vertices** : Vertices are the nodes of the graph
- **Edge** : The edge is the line that connects pairs of vertices
- **Unweighted graph** : A graph which does not have a weight associated with any edge
- **Weighted graph** : A graph which has a weight associated with any edge
- **Undirected graph** : In case the edges of the graph do not have a direction associated with them



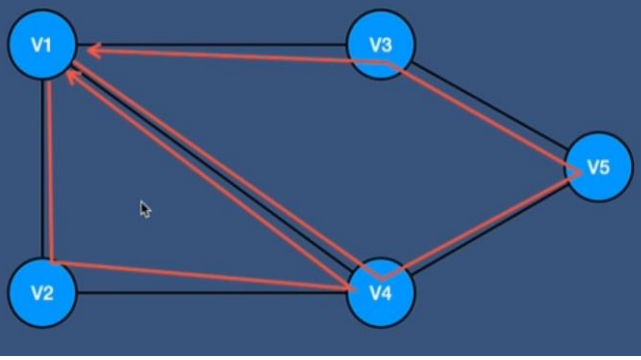
Graph Terminology

- **Vertices** : Vertices are the nodes of the graph
- **Edge** : The edge is the line that connects pairs of vertices
- **Unweighted graph** : A graph which does not have a weight associated with any edge
- **Weighted graph** : A graph which has a weight associated with any edge
- **Undirected graph** : In case the edges of the graph do not have a direction associated with them
- **Directed graph** : If the edges in a graph have a direction associated with them



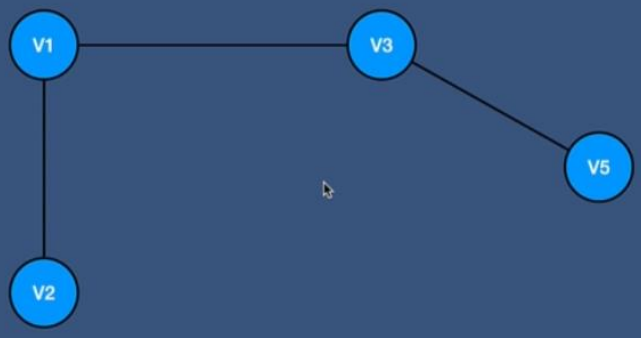
Graph Terminology

- **Vertices** : Vertices are the nodes of the graph
- **Edge** : The edge is the line that connects pairs of vertices
- **Unweighted graph** : A graph which does not have a weight associated with any edge
- **Weighted graph** : A graph which has a weight associated with any edge
- **Undirected graph** : In case the edges of the graph do not have a direction associated with them
- **Directed graph** : If the edges in a graph have a direction associated with them
- **Cyclic graph** : A graph which has at least one loop



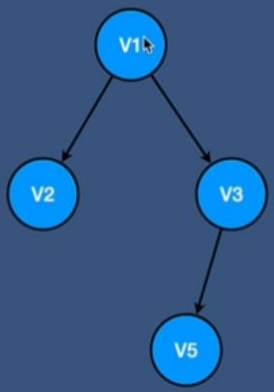
Graph Terminology

- **Vertices** : Vertices are the nodes of the graph
- **Edge** : The edge is the line that connects pairs of vertices
- **Unweighted graph** : A graph which does not have a weight associated with any edge
- **Weighted graph** : A graph which has a weight associated with any edge
- **Undirected graph** : In case the edges of the graph do not have a direction associated with them
- **Directed graph** : If the edges in a graph have a direction associated with them
- **Cyclic graph** : A graph which has at least one loop
- **Acyclic graph** : A graph with no loop

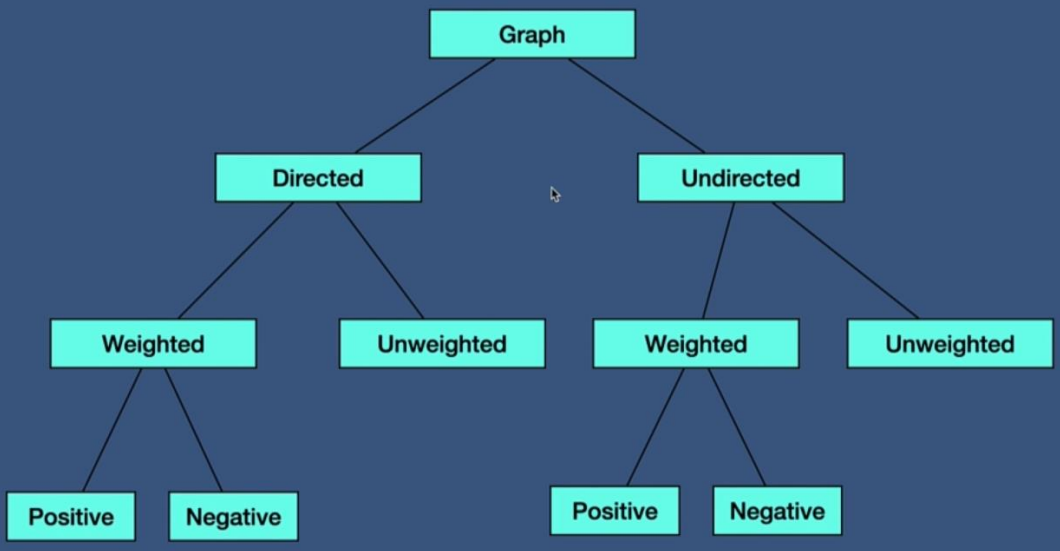


Graph Terminology

- **Vertices** : Vertices are the nodes of the graph
- **Edge** : The edge is the line that connects pairs of vertices
- **Unweighted graph** : A graph which does not have a weight associated with any edge
- **Weighted graph** : A graph which has a weight associated with any edge
- **Undirected graph** : In case the edges of the graph do not have a direction associated with them
- **Directed graph** : If the edges in a graph have a direction associated with them
- **Cyclic graph** : A graph which has at least one loop
- **Acyclic graph** : A graph with no loop
- **Tree**: It is a special case of directed acyclic graphs

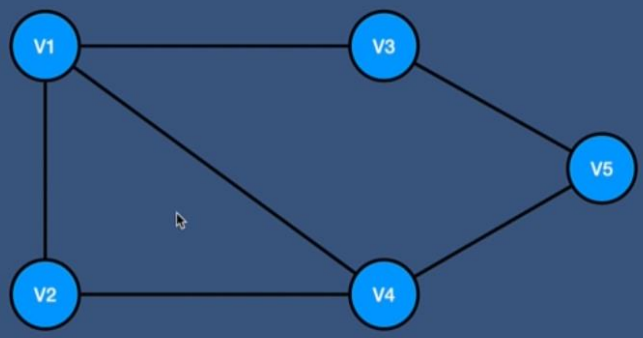


Graph Types



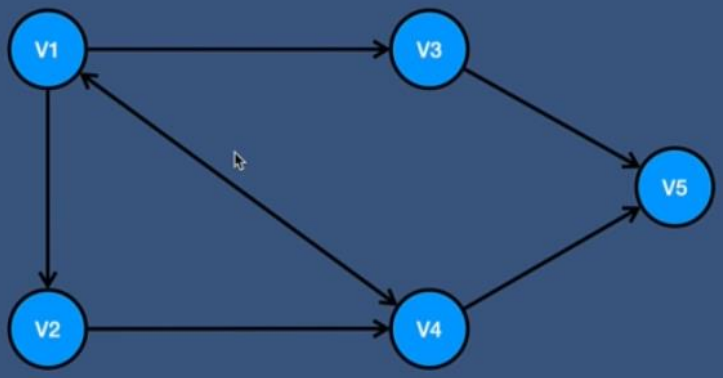
Graph Types

1. Unweighted - undirected



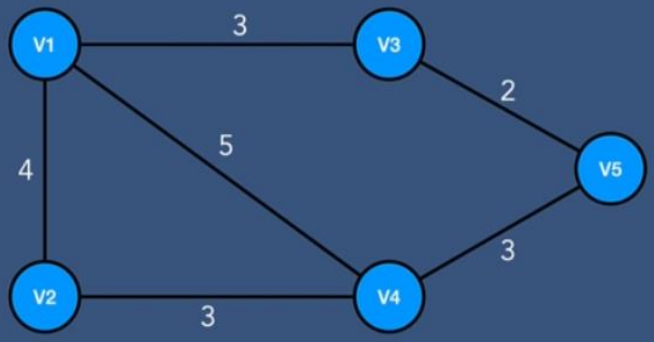
Graph Types

1. Unweighted - undirected
2. Unweighted - directed



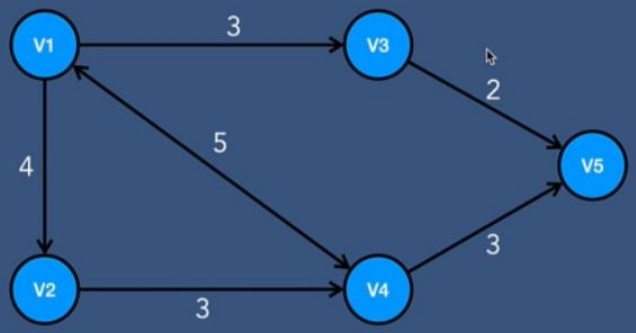
Graph Types

1. Unweighted - undirected
2. Unweighted - directed
3. Positive - weighted - undirected



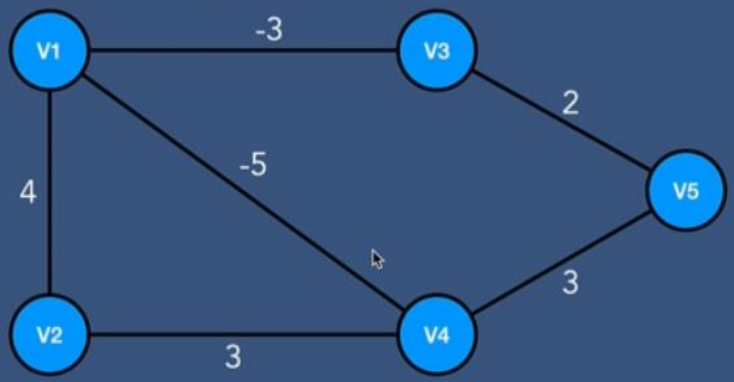
Graph Types

1. Unweighted - undirected
2. Unweighted - directed
3. Positive - weighted - undirected
4. Positive - weighted - directed



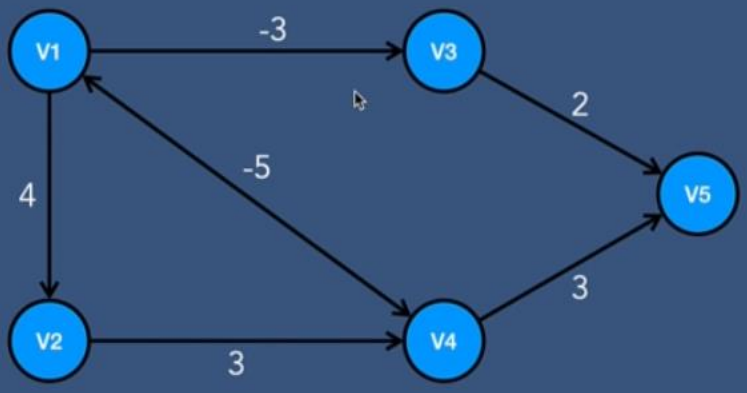
Graph Types

1. Unweighted - undirected
2. Unweighted - directed
3. Positive - weighted - undirected
4. Positive - weighted - directed
5. Negative - weighted - undirected



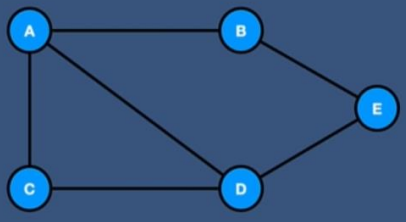
Graph Types

1. Unweighted - undirected
2. Unweighted - directed
3. Positive - weighted - undirected
4. Positive - weighted - directed
5. Negative - weighted - undirected
6. Negative - weighted - directed



Graph Representation

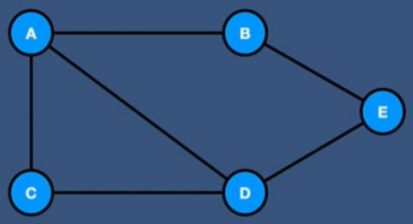
Adjacency Matrix : an adjacency matrix is a square matrix or you can say it is a 2D array. And the elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph

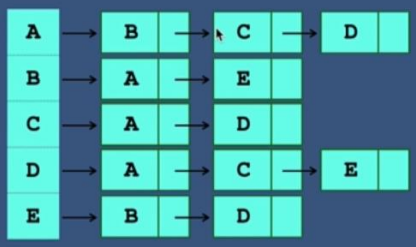


	A	B	C	D	E
A	0	1	1	1	0
B	1	0	0	0	1
C	1	0	0	1	0
D	1	0	1	0	1
E	0	1	0	1	0

Graph Representation

Adjacency List : an adjacency list is a collection of unordered list used to represent a graph. Each list describes the set of neighbors of a vertex in the graph.





Graph Representation

If a graph is complete or almost complete we should use **Adjacency Matrix**

If the number of edges are few then we should use **Adjacency List**

	A	B	C	D	E
A	0	1	1	1	0
B	1	0	0	0	1
C	1	0	0	1	0
D	1	0	1	0	1
E	0	1	0	1	0