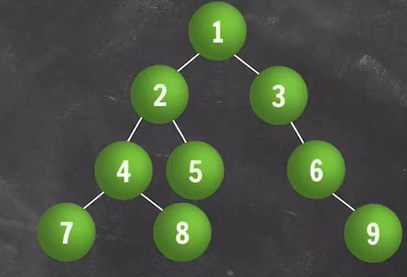
Tree



**Children of 2**: 4 ,5

**No of Leaves**: 4 (7,8,5,9)

**Parent of 3**: 1

**Level of 5**: 3

**Subtree of 2 & 6**:

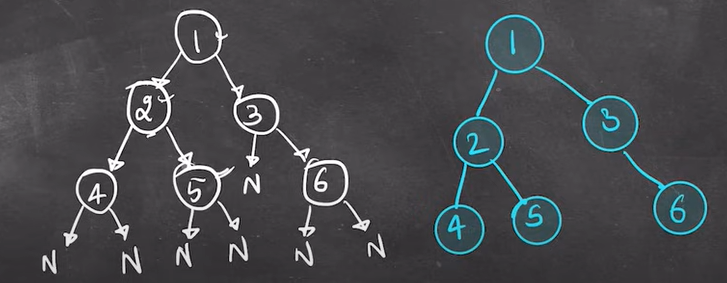
**2**: (left subtree) 4,7,8 , (right sub tree) 5

**6**: (right subtree) 9

**Ancestors of 4**: 2 and 1

Build Tree Preorder

1, 2, 4, -1, -1, 5, -1, -1, 3, -1, 6, -1, -1



Graph

**Graph Data Structure**is a [dat](https://www.geeksforgeeks.org/introduction-to-hierarchical-data-structure)a structure consisting of vertices and edges. It is useful in fields such as social network analysis, recommendation systems, and computer networks.

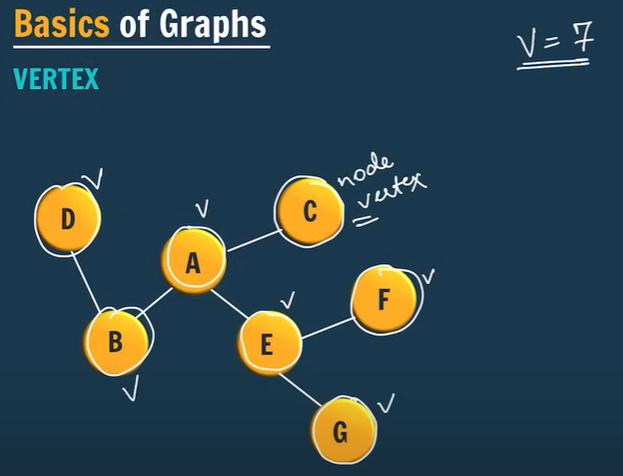
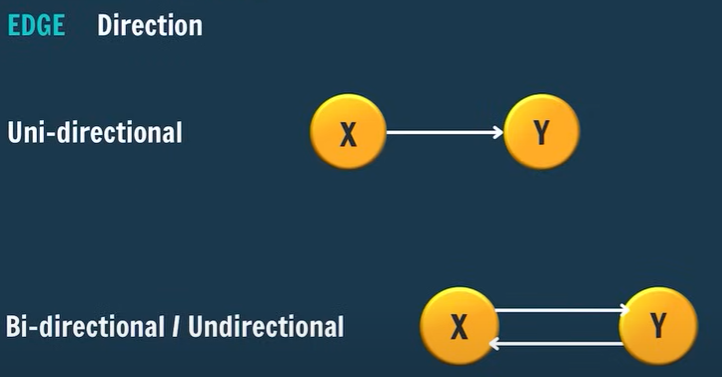
It consists of vertices and edges. The vertices are sometimes also referred to as nodes and the edges are lines that connect any two nodes in the graph. More formally a Graph is composed of a set of vertices (V) and a set of edges (**E**). The graph is denoted by **G (V, E).**

**Application –**

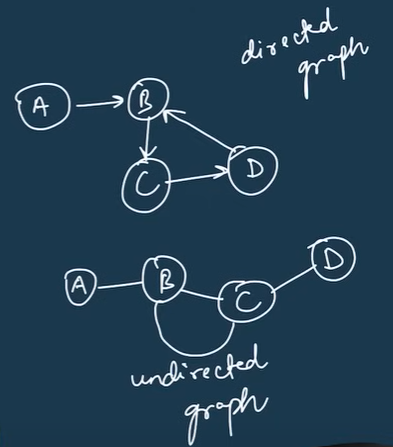
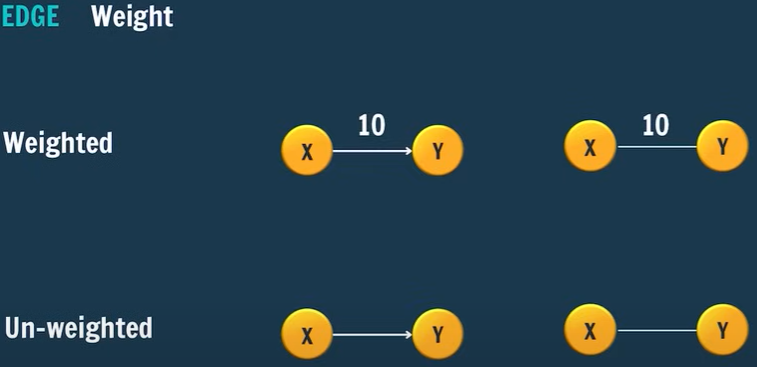
1. Maps
2. Social Network
3. Delivery Network

**Basics**

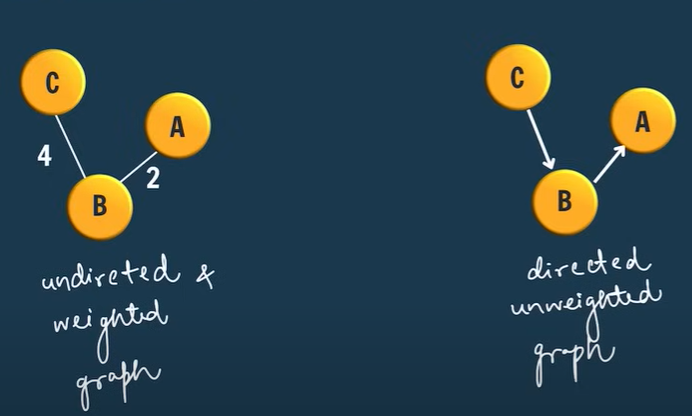
Vertex = node Edge – connecting two vertices

Directed graph and undirected graph

Example –



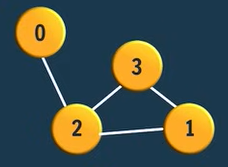
Storing a Graph

1. Adjacency List – Mostly used - List of List
2. Adjacency Matrix
3. Edge List
4. 2D Matrix (Implicit Graph)

1. **Adjacency List**

List of List

Example - Vertex = 4 | Edge = 4 | undirected unweighted graph



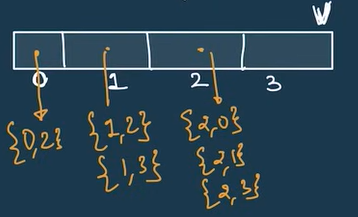
We will store vertex wise,

0-> {0,2}

1-> {1,2}, {1,3}

2-> {2,0}, {2,1}, {2,3}

3-> {3,2}, {3,1}



Representation – Length of arraylist = no of vertex

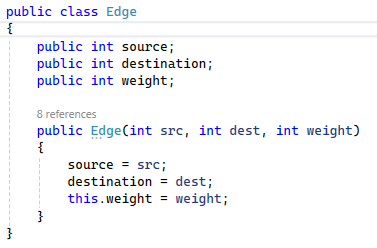
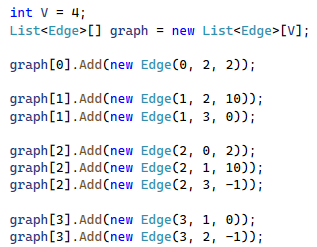
 

Figure / Implementation in class

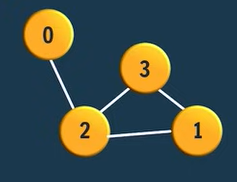
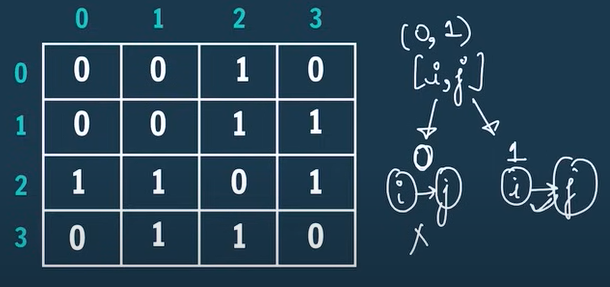
1. **Adjacency Matrix**

It is store in the form of matrix

**If there are no edges, then we store 0**

**If there are edges between vertex, then we store 1 for non-weighted graph**

**If it is weighted graph, then we assign the weight instead of 1.**

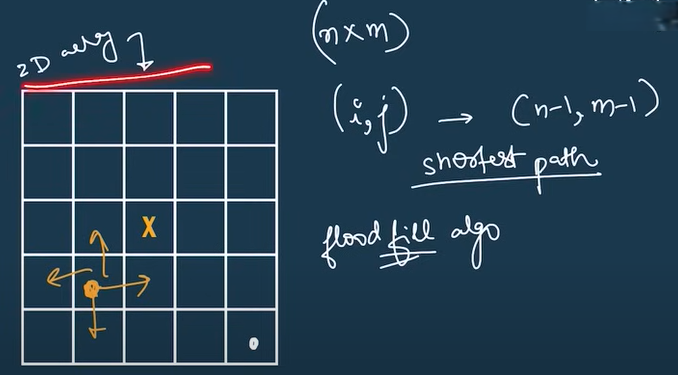
Disadvantages

1. Extra space - o(v2)
2. Neighbour – o(v) but in case of adjacency List – o(x) {x is the no. of neighbours}

3) Edge List

4) Implicit Graph

2D – array

****

Algorithm – MST , Flood Fill Algo

**Graph Traversals**