Introduction to Graph Theory

PRESIDENCY UNIVERSITY
Private University Estd. in Karnataka State by Act No. 41 of 2013

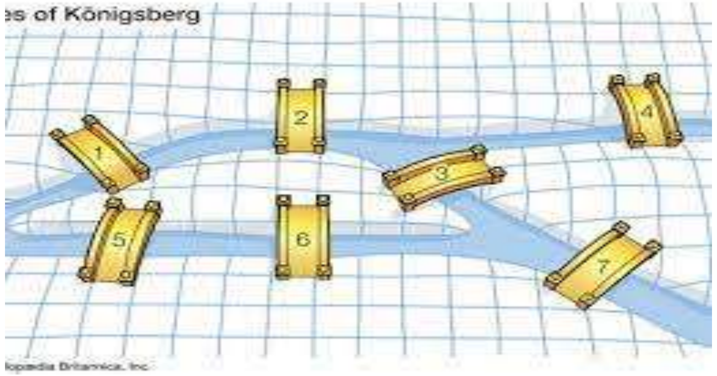
Graph Theory

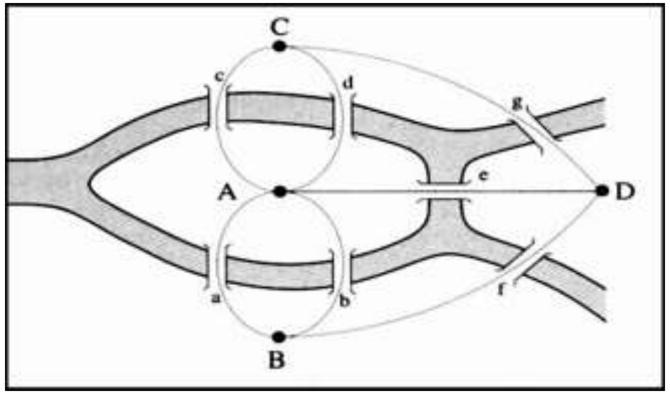
A branch of mathematics that explores the relationships between various entities, represented as nodes, connected by edges.

Origin

Euler's Problem (1736):

Graph theory originated from Euler's Seven Bridges of Königsberg problem, where he aimed to find a walk through the city that would cross each of its seven bridges once and only once.





Basic Concepts

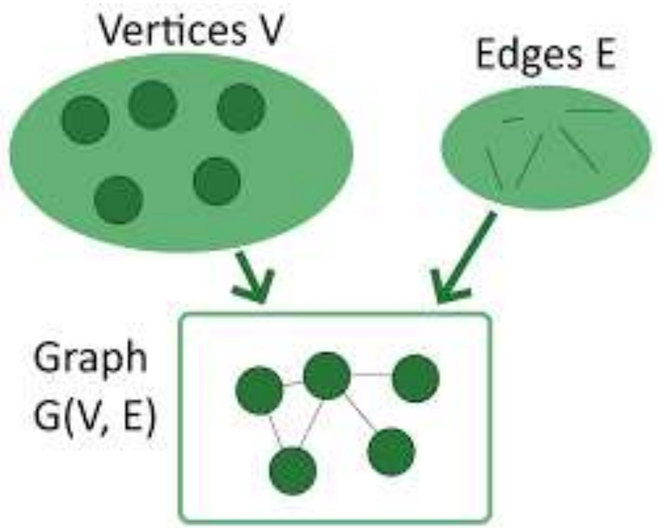
Vertex (Node): Represents an object or entity.

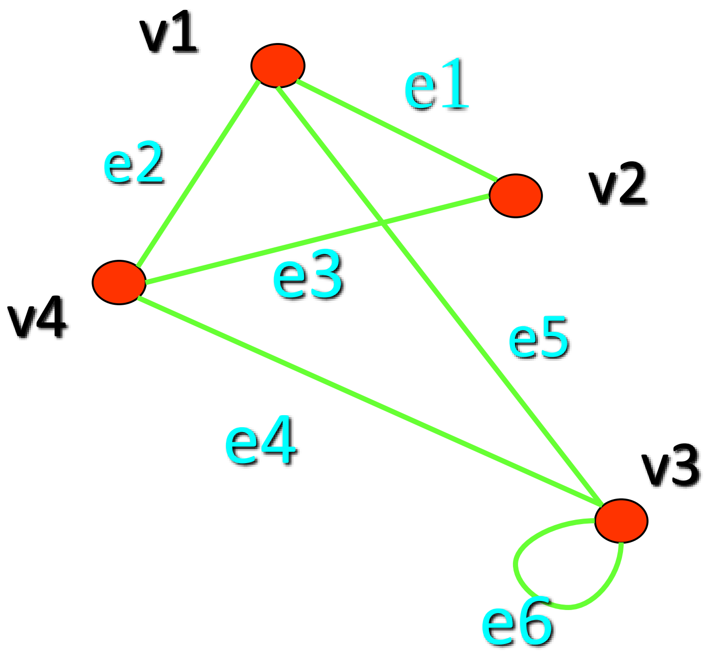
Edge: Represents a connection between vertices.

Graph: A graph G is a pair (V, E). It is collection of vertices and edges.

V: set of vertices

E: set of edges connecting the vertices in V





The graph has 4 vertices namely v1, v2, v3, v4

& 6 edges namely e1, e2, e3, e4, e5, e6

Then e1=(v1, v2)

Similarly for other edges.

In short, we can represent G=(V,E)

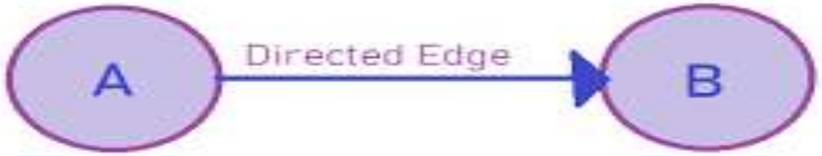
where V=(v1, v2, v3, v4) &

E=(e1, e2, e3, e4, e5, e6)

Definitions

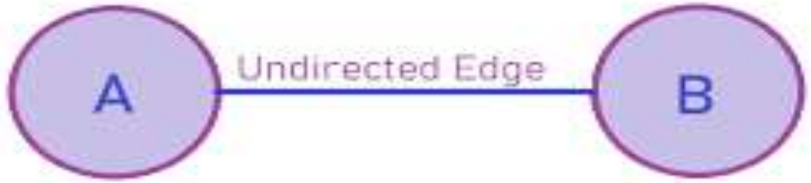
Directed edge:

The edge with arc / directions/ arrow is referred to directed edge.
Each directed edge connects an ordered pair of vertices, indicating a one-way relationship or connection from one vertex to another.



Undirected edge:

The edge with out arc / directions/ arrow is referred to undirected edge.
instead, they simply connect pairs of vertices, indicating a mutual connection or relationship.

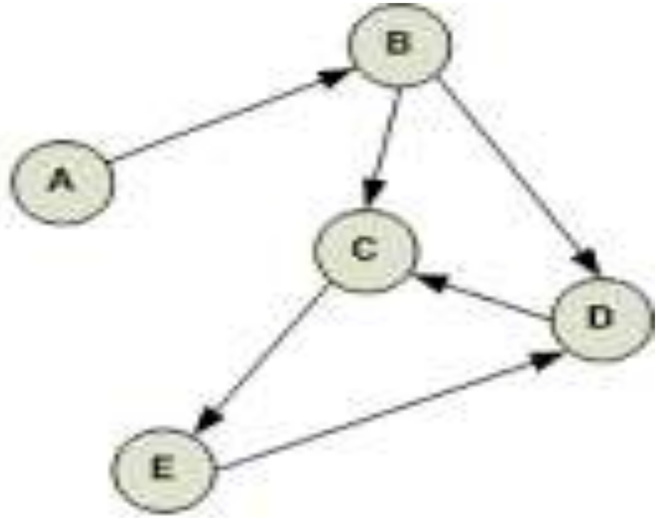


Types of Graphs

- Directed Graph (Digraph)
- Undirected Graph
- Weighted Graph
- Unweighted Graph
- Bipartite Graph

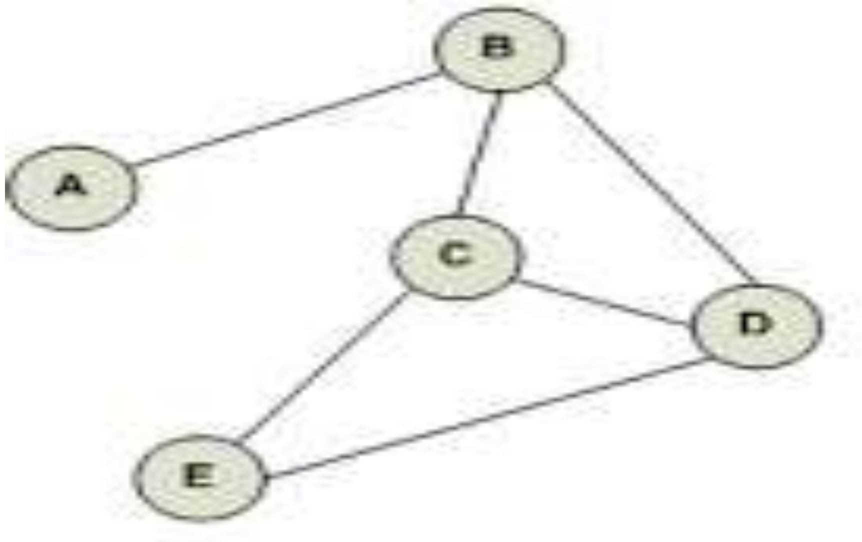
- **Directed Graph (Digraph)**

A graph in which all the edges are directed is called as a directed graph.
In other words, all the edges of a directed graph contain some direction.
Directed graphs are also called as **digraphs**.



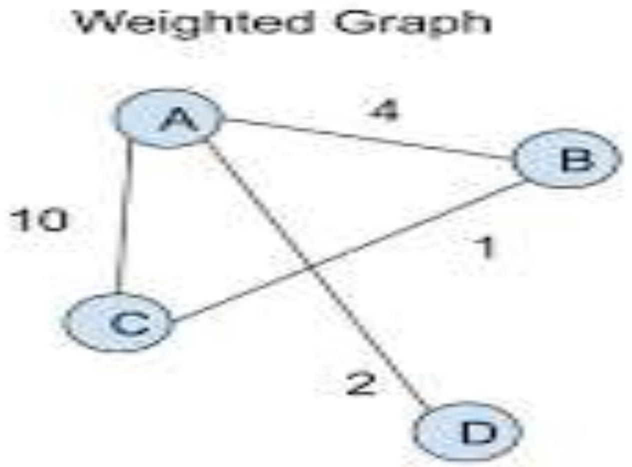
- **Undirected Graph (Simple Graph)**

A graph in which all the edges are undirected is called as a non-directed graph.
In other words, edges of an undirected graph do not contain any direction.



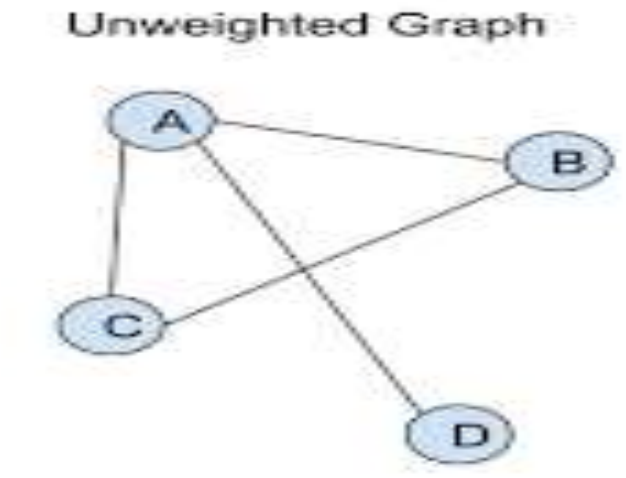
- **Weighted Graph**

A graph in which a number (the weight) is assigned to each edge is called as weighted graph.
Such weights might represent for example costs, lengths or capacities, depending on the problem at hand.



• Unweighted Graph

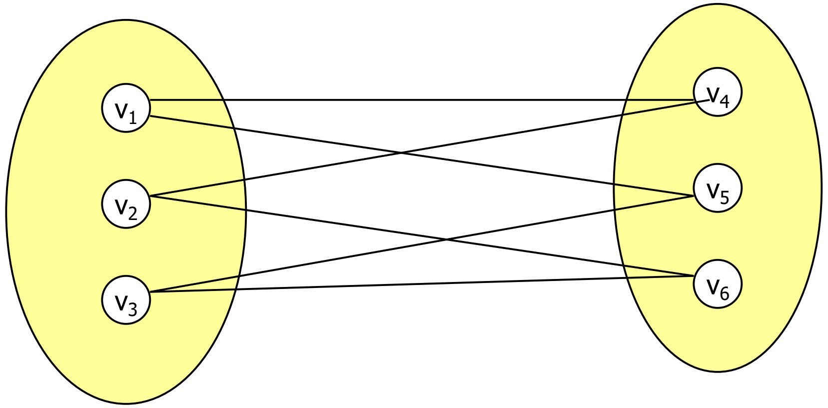
A graph in which no weight is assigned to edges is called as unweighted graph.



- **Bipartite Graph**

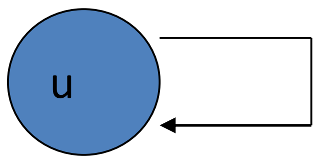
In a simple graph G, if V can be partitioned into two disjoint sets V_1 and V_2 such that every edge in the graph connects a vertex in V_1 and a vertex V_2 (so that no edge in G connects either two vertices in V_1 or two vertices in V_2)

Example: $V_1 = \{v_1, v_2, v_3\}$ and $V_2 = \{v_4, v_5, v_6\}$,



Terminology

Loop: A loop is an edge whose endpoints are equal i.e., an edge joining a vertex to it self is called a loop. Represented as {u, u} = {u}



Multiple Edges: Two or more edges joining the same pair of vertices.

Terminology – Undirected graphs

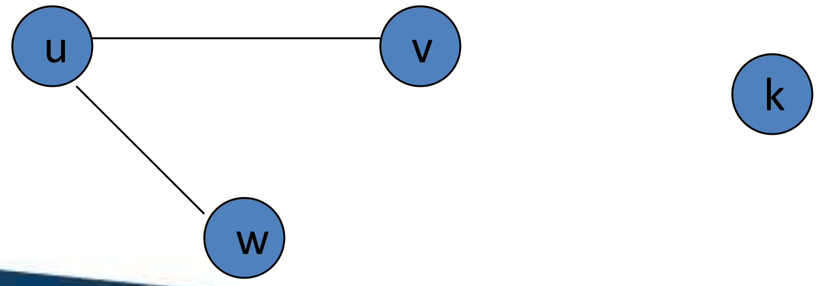
u and v are **adjacent** if {u, v} has an edge. Then, e is called **incident** with u and v. u and v are called **endpoints**.

Degree of Vertex (deg (v)): The number of edges incident on a vertex.

Pendant Vertex: deg (v) =1

Isolated Vertex: deg (k) = 0

Example: For V = {u, v, w} , E = { {u, w}, {u, v} }, deg (u) = 2, deg (v) = 1, deg (w) = 1, deg (k) = 0, w and v are pendant , k is isolated



Terminology – Directed graphs

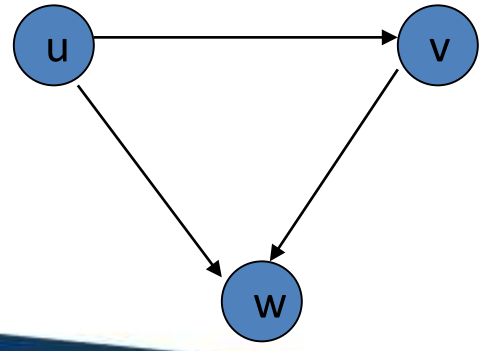
For the edge (u, v), u is **adjacent to** v OR v is **adjacent from** u, u – **Initial vertex**, v – **Terminal vertex**

In-degree (deg^- (u)): number of edges for which u is terminal vertex

Out-degree (deg^+ (u)): number of edges for which u is initial vertex

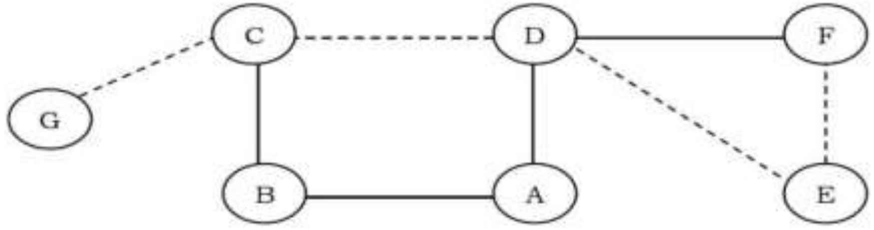
Note: A loop contributes 1 to both in-degree and out-degree (why?)

Example: For V = {u, v, w} , E = { (u, w), (v, w), (u, v) }, deg^- (u) = 0, deg^+ (u) = 2, deg^- (v) = 1, deg^+ (v) = 1, and deg^- (w) = 2, deg^+ (u) = 0

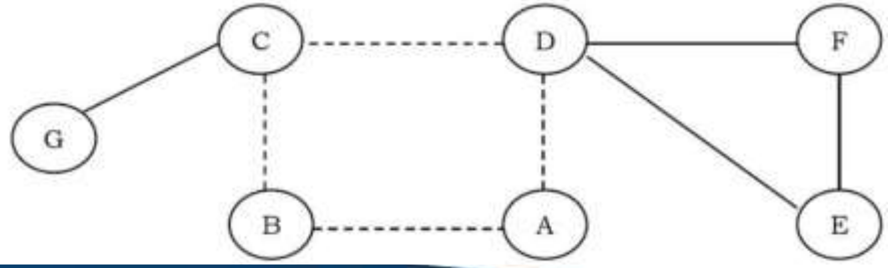


Terminology

Path: A path in a graph is a sequence of adjacent vertices. Simple path is a path with no repeated vertices. In the graph below, the dotted lines represent a path from G to E.

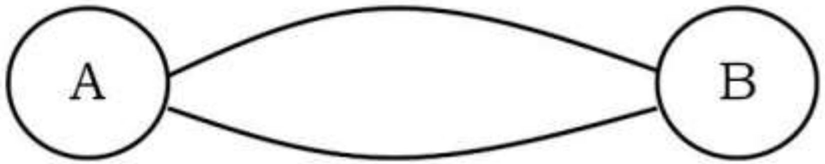


Cycle: A cycle is a path where the first and last vertices are the same. A simple cycle is a cycle with no repeated vertices or edges (except the first and last vertices).

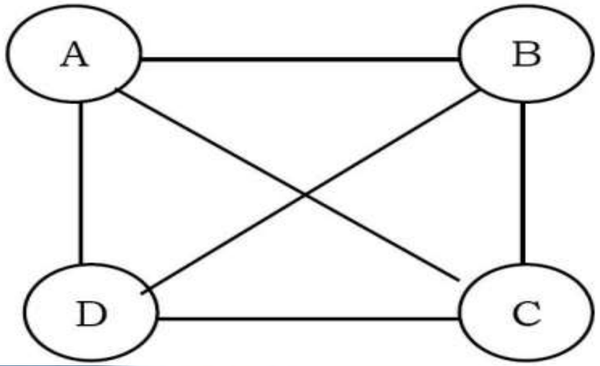


Terminology

Parallel Edges: Two edges are parallel if they connect the same pair of vertices

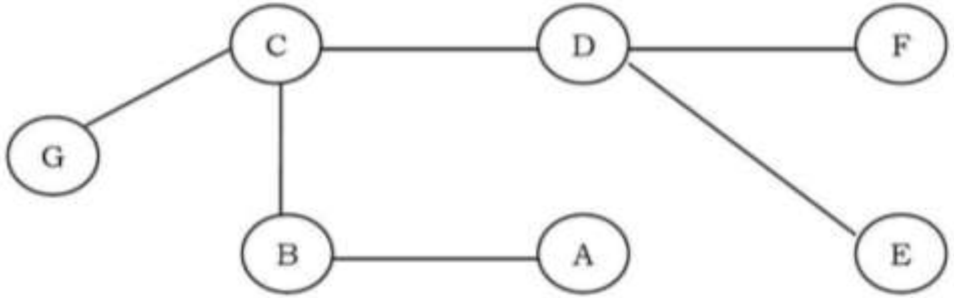


Complete graph: is an undirected graph in which every pair of distinct vertices is connected by a unique edge.

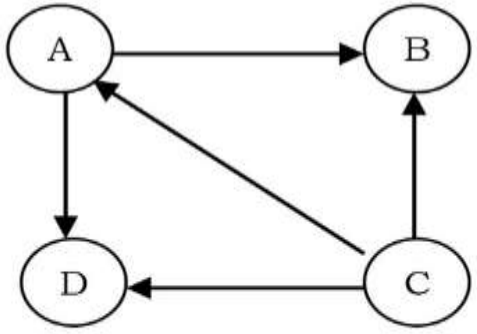


Terminology

Acyclic Graph : A graph with no cycles. A tree is an acyclic connected graph.



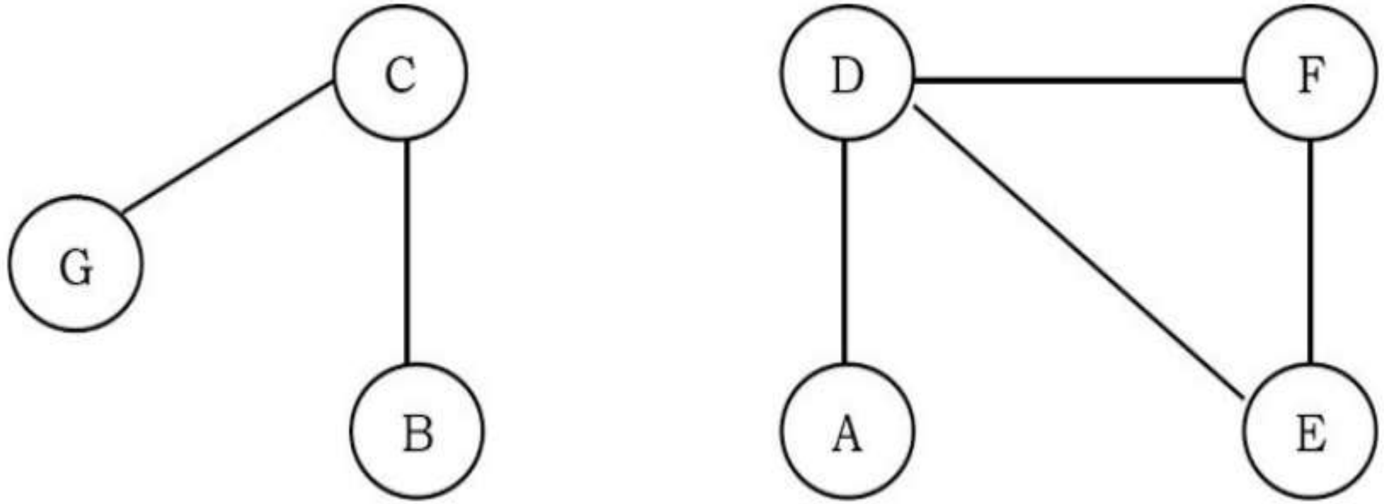
Directed acyclic graph [DAG]: A directed graph with no cycles.



Terminology

Connected Graph: A graph is connected if there is a path from every vertex to every other vertex.

• If a graph is not connected then it consists of a set of connected components.



Graph Representation

To manipulate graphs we need to represent them in some useful form.

There are three ways of doing this:

- Adjacency Matrix
- Adjacency List
- Adjacency Set

Adjacency Matrix

An N x N matrix, where |V| = N , the Adjacenct Matrix (NxN) A = $[a_{ij}]$

For undirected graph

$$a_{ij} = \begin{cases} 1 \text{ if } \{v_i, v_j\} \text{ is an edge of } G \\ 0 \text{ otherwise} \end{cases}$$

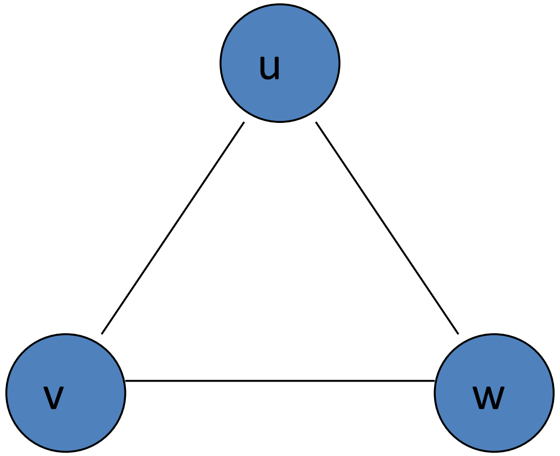
For directed graph

$$a_{ij} = \begin{cases} 1 \text{ if } (v_i, v_j) \text{ is an edge of } G \\ 0 \text{ otherwise} \end{cases}$$

This makes it easier to find subgraphs, and to reverse graphs if needed. The adjacency matrix representation is good if the graphs are dense.

Adjacency Matrix (Contd..)

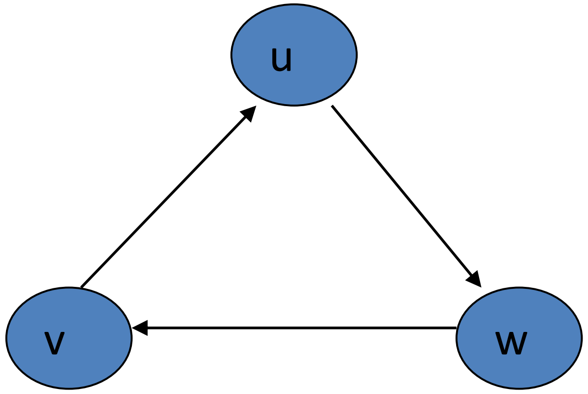
Example: Undirected Graph G (V, E)



	v	u	w
v	0	1	1
u	1	0	1
w	1	1	0

Adjacency Matrix (Contd..)

Example: Directed Graph G (V, E)



	v	u	w
v	0	1	0
u	0	0	1
w	1	0	0

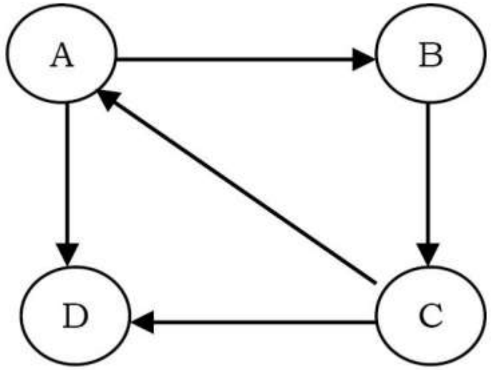
Adjacency List

This representation is implemented with linked lists

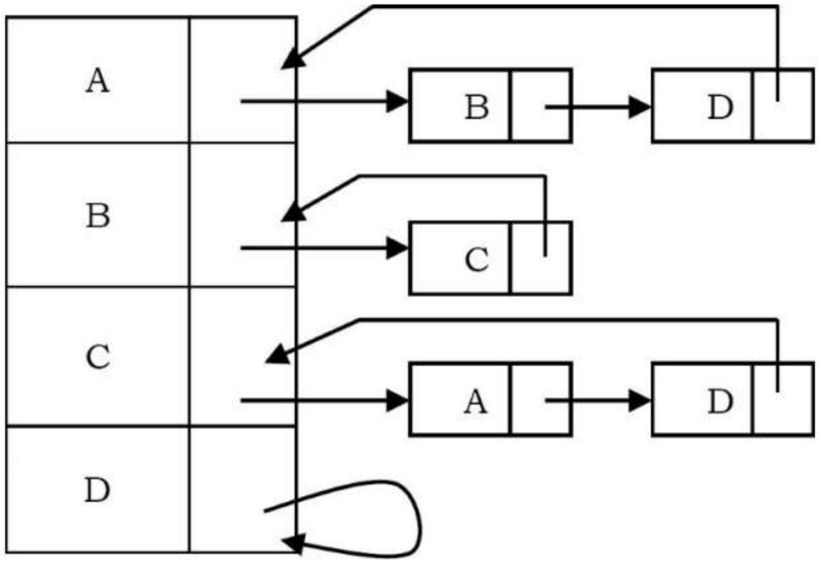
- All the vertices connected to a vertex u are listed on an adjacency list for that vertex u.
- For each vertex u we use a linked list and list nodes represents the connections between u and other vertices to which u has an edge.
- The total number of linked lists is equal to the number of vertices in the graph

Adjacency List(Contd..)

Example



a. Graph



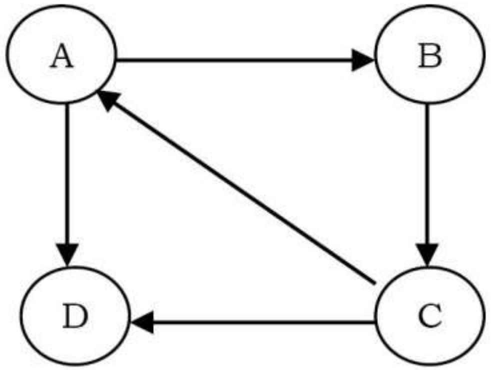
b. Representation Using Adjacency List

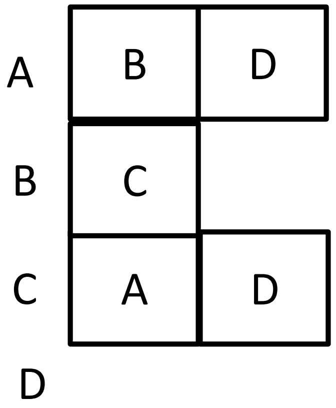
Using adjacency list representation we cannot perform some operations efficiently.

Adjacency Set

It is very much similar to adjacency list but instead of using Linked lists, Disjoint Sets are used.

Example





Applications of Graphs

- Representing relationships between components in electronic circuits
- Transportation networks: Highway network, Flight network
- Computer networks: Local area network, Internet, Web
- Databases: For representing ER (Entity Relationship) diagrams in databases, for representing dependency of tables in databases