# CSCI 2110 Data Structures and Algorithms

# Module 10: Graphs
# Part 1 - Introduction


**Data Structures: Module 10 – Part 1** **Srini Sampalli**

48

# Topics

- Definition and motivation
- Graph terminology
- Graph Representation

Part 1

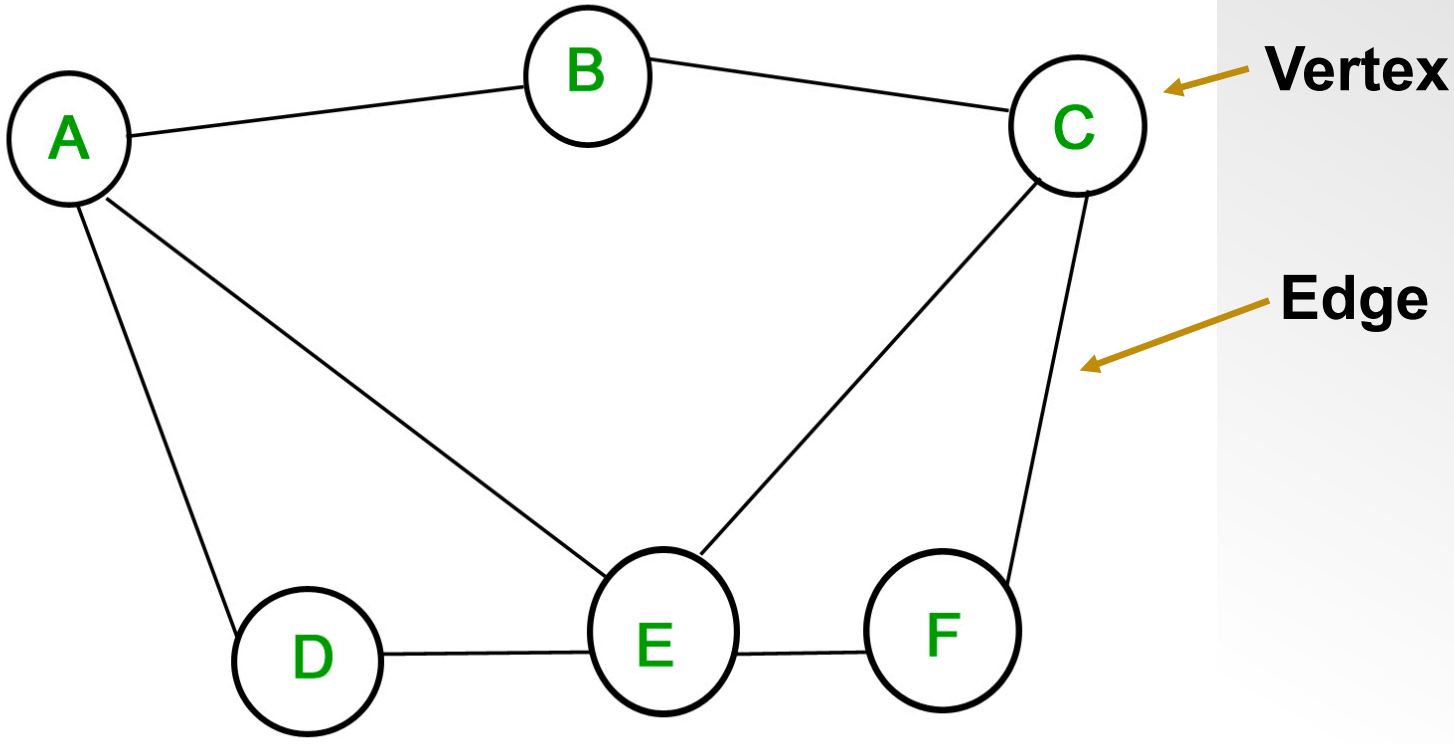
- Graph Algorithms

Part 2

# The Graph Data Structure

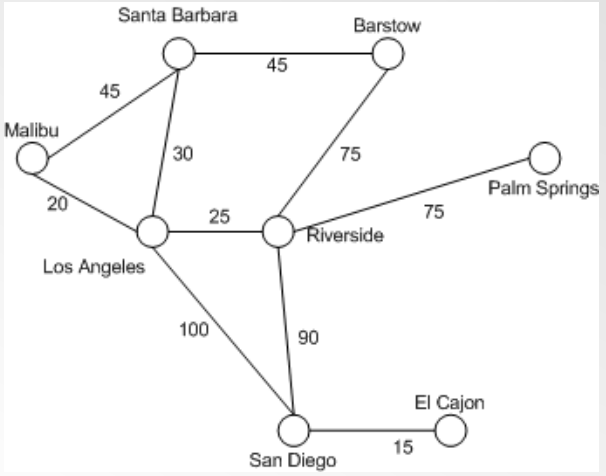
- A graph is a data structure that consists of a set of **nodes** (also called **vertices**) and a set of **edges**.
- Each node or vertex generally holds data.
- Each edge connects a pair of nodes that have some form of relationship between them.
- A graph can be used to represent any geometric pattern – that is, a list, a binary tree, binary search tree, etc. can all be considered as special cases of a graph data structure.
- This makes the graph one of the most widely used data structures.

# The Graph Data Structure

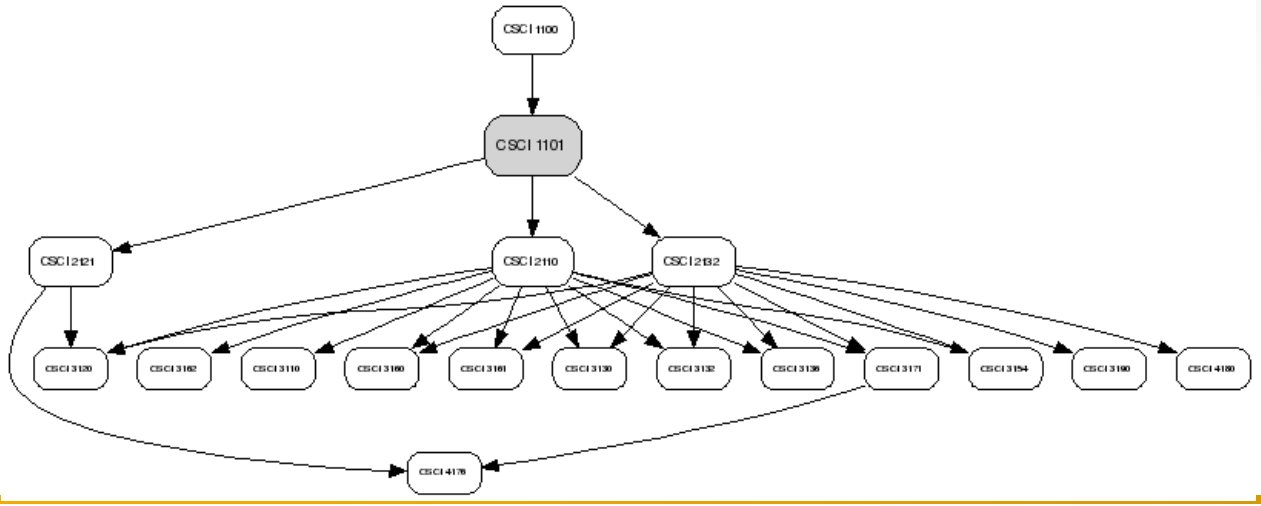


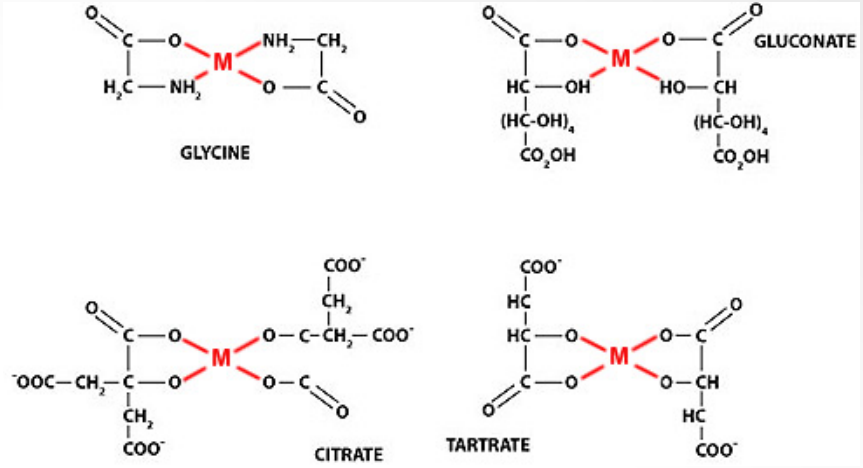
# Examples:

Graph of cities connected by highways



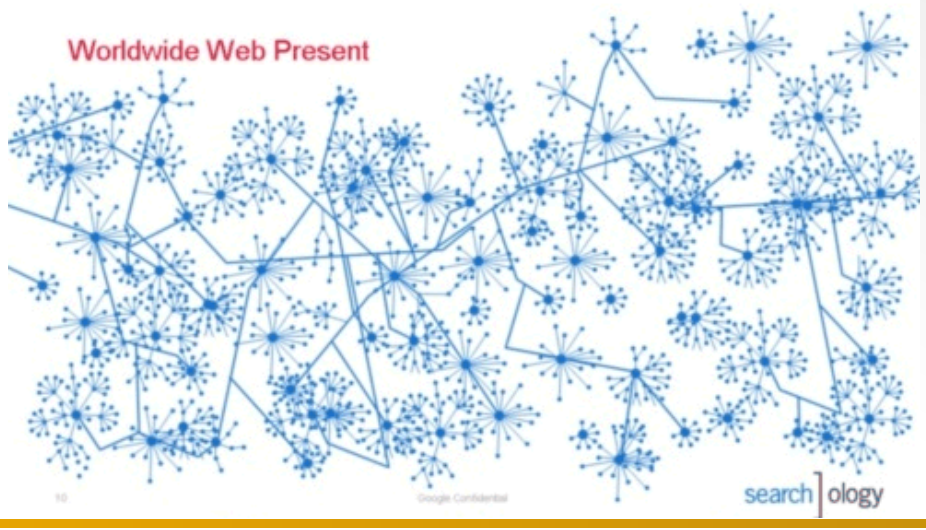

Graph of CS courses with prerequisites



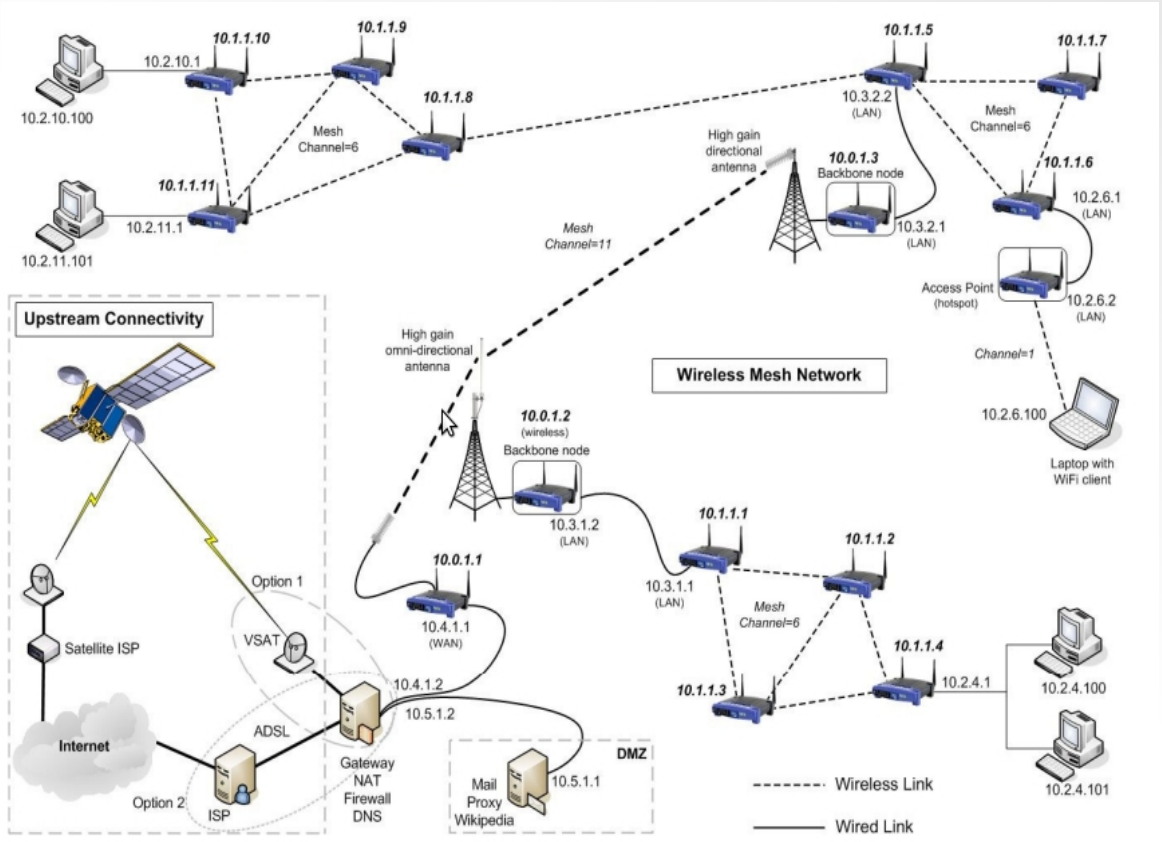



Graphs of molecular structures of organic compounds

Graph of web pages connected by hyperlinks



# Graph of a wireless/wired network

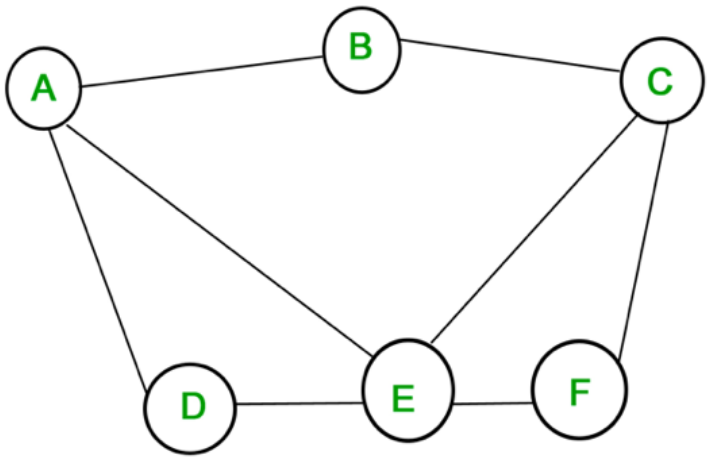



Source: http://www.nvtech.com.au/ProjCurr/LCMAN/LCMAN.html

# Graph Terminology

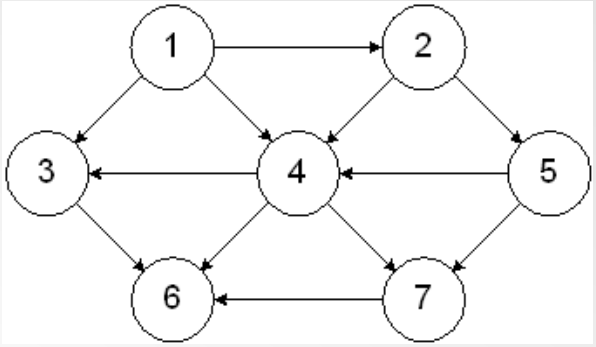
# Undirected Graph

- An undirected graph is one in which there is a symmetric relationship among the vertices.
- An undirected graph has edges without arrows.
- It means that if A and B are two vertices connected by an edge in an undirected graph, you can get to B from A and vice versa.



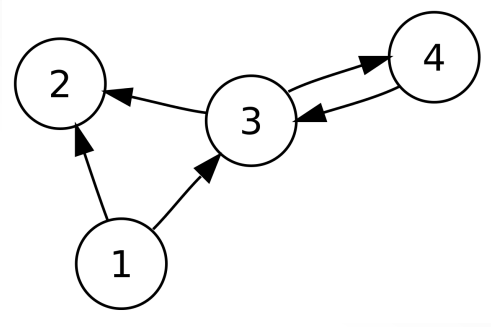
# Directed Graph

- A directed graph is one in which there is an asymmetric relationship among the vertices.



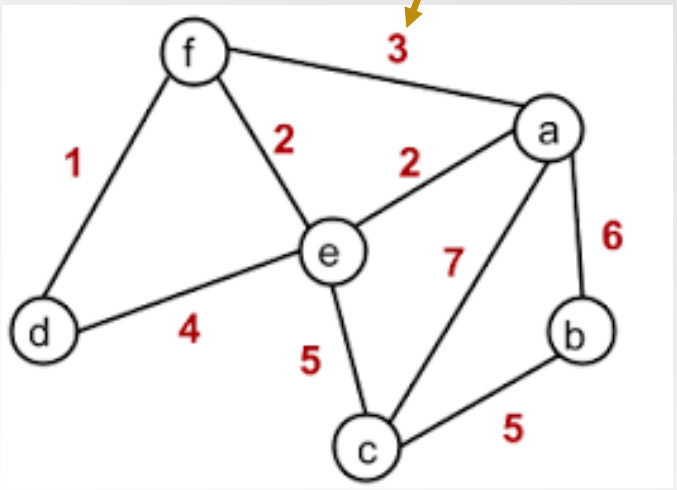

- In a directed graph, edges are represented by arrows starting at one vertex and ending at the neighbouring vertex.

- In the first figure, there is an edge from vertex 1 to 2. This means that 2 can be visited from 1 but 1 cannot be visited from 2 (directly).
- Edges can also be bidirectional as shown in the second figure (4 can be visited from 3 and vice versa).



# Weighted Graph

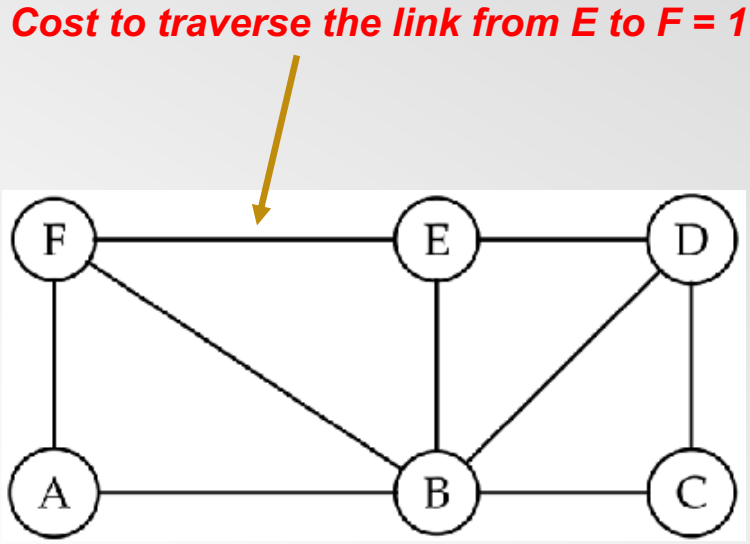
- Each edge has a number associated with it, called the cost or the weight. The number represents the "effort" it takes to traverse that edge.
- The weight could represent one parameter or an aggregate of multiple parameters.
- Weights play an important role in algorithms such as shortest path determination.




***Shortest path from a to d is via f***
***The total cost of that path is 3 + 1 = 4.***

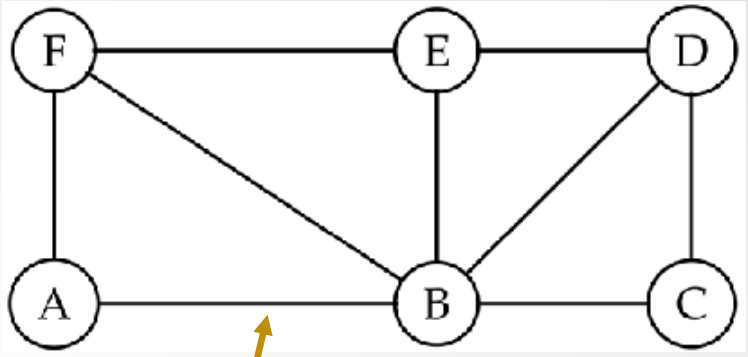
# Unweighted Graph

- Edges don't have any costs associated with them.
- In unweighted graphs, all edges are treated alike, that is, it is not beneficial to prefer one path over the another.
- An unweighted graph can be considered as a special case of a weighted graph in which the weight of every edge = 1.



# Representation of an Edge

- An edge between two vertices v1 and v2 is represented by the tuple (v1, v2).
- If the graph is undirected, if an edge (v1, v2) exists , then (v2, v1) also exists.
- If the graph is directed and there is an edge from v1 to v2 only, then only edge (v1, v2) exists.




***Edge (A, B)***

***The edge can also be (B, A).***

# Degree of a vertex (Undirected Graph)

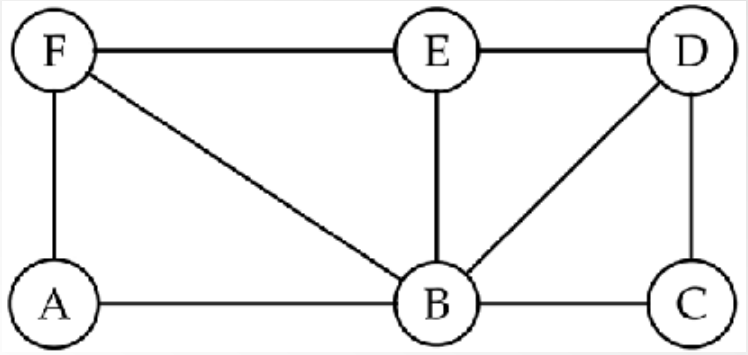
- Number of edges connected to the vertex

- In the figure:

*Degree of A = 2*

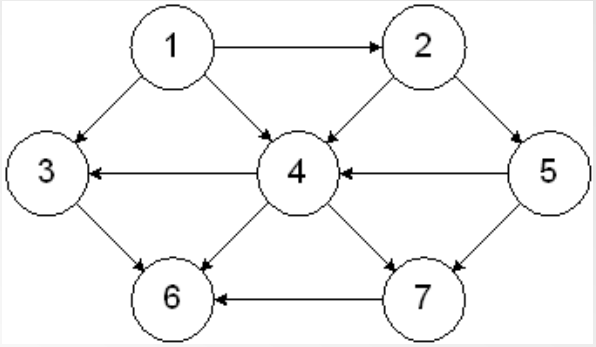
*Degree of B = 5*

*Degree of E = 3*



# Degree of a vertex (Directed Graph)

- **Indegree** = Number of edges terminating at the vertex
- **Outdegree** = Number of edges leaving the vertex




- In the figure:

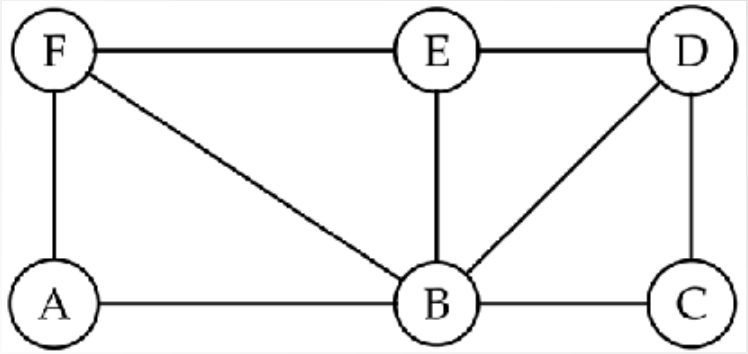
*Indegree of vertex 1 = 0*
*Outdegree of vertex 1 = 3*

*Indegree of vertex 4 = 3*

*Outdegree of vertex 4 = 3*

# Path in a graph

- A path from vertex v1 to vertex vk is any sequence of edges to get from v1 to vk.
- Sequence is represented by tuples (v1, v2), (v2, v3), etc.




- In the figure:

*Path from A to D: (A, B), (B, C), (C, D)*

*Another path from A to D: (A, F), (F, E), (E, B), (B, D)*

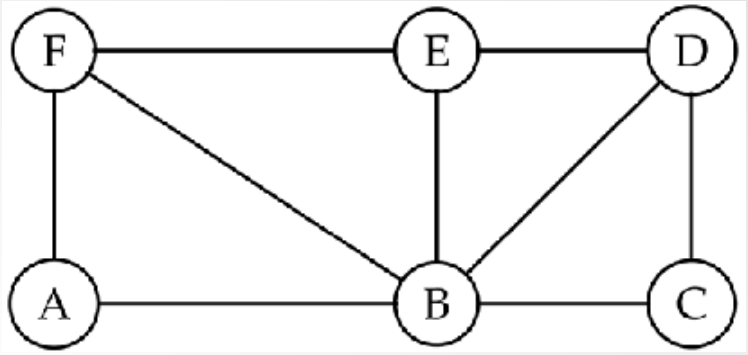
*Yet another path from A to D:*

*(A, F), (F, B), (B, E), (E, B), (B, D)*

# Simple Path

- A simple path is a path in which no vertex is visited twice.

- The only exception for repetition is the first and last vertex (they can be the same).




- *Some simple paths in the figure:*

*Simple path from A to D: (A,B) (B,C) (C,D)*

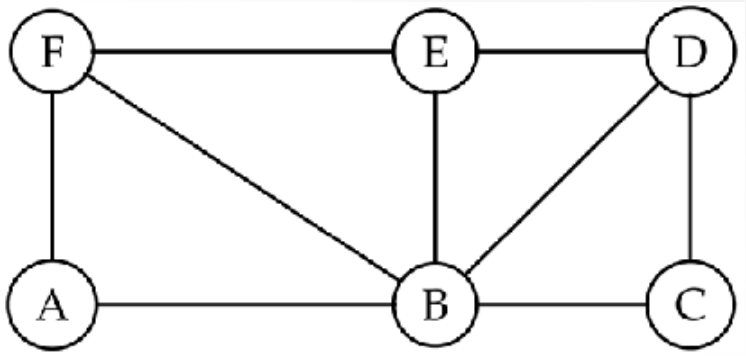
*Simple path from B to B: (B,E) (E,F) (F, A) (A,B)*

*This is not a simple path: (A, B) (B, F) (F, B) (B, C)*

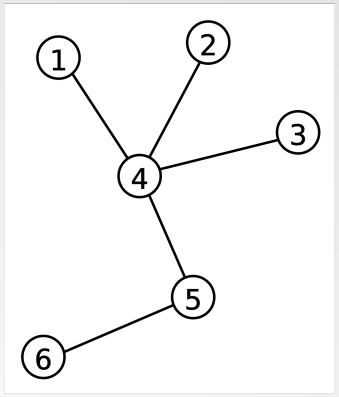
# Cycle and Cyclic and Acyclic Graphs

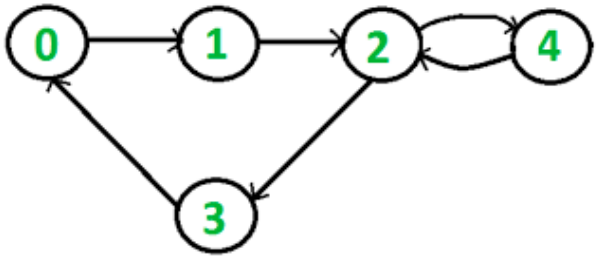
- A cycle is a simple path in which the first and last vertices are the same.
- A graph that has at least one cycle is a cyclic graph.
- A graph that has no cycles is an acyclic graph.

## Undirected Cyclic Graph

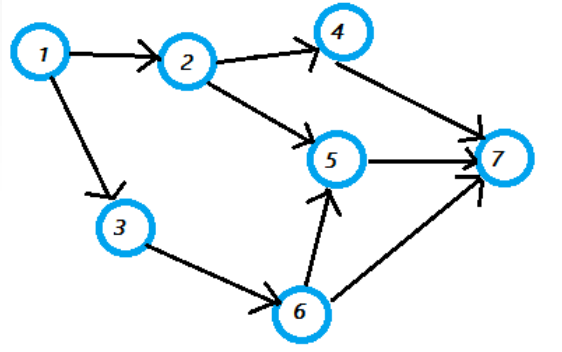



## Undirected Acyclic Graph




## Directed Cyclic Graph



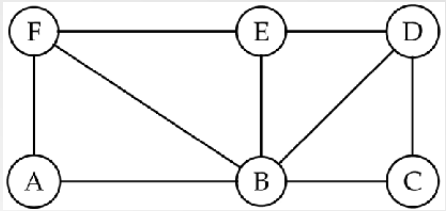

## Directed Acyclic Graph

***(No cycles!)***

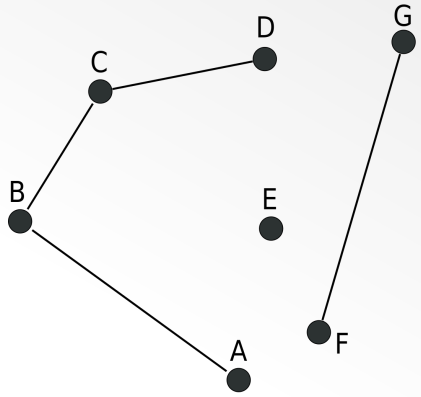
# Connected and Unconnected Graphs

- Connected Graph: There is a path between every pair of vertices.

- Unconnected Graph: A collection of graphs, each of which is connected.



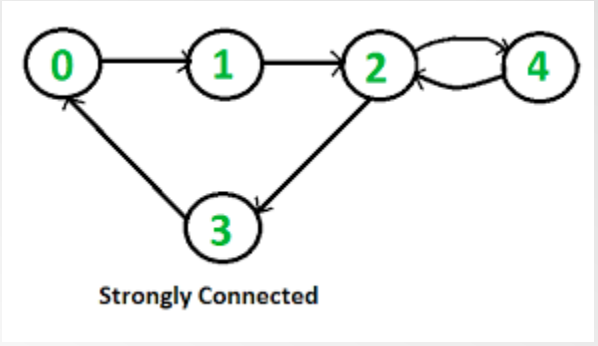

**Connected Graph**



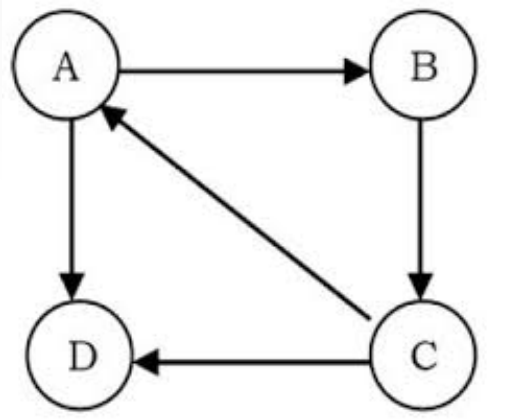

**Unconnected Graph**

# Directed Graphs: Strongly Connected and Weakly Connected

- A directed graph is strongly connected if there is a path from every vertex to every other vertex.




- A directed graph is weakly connected if, ignoring the directions of edges, the resulting undirected graph is connected.



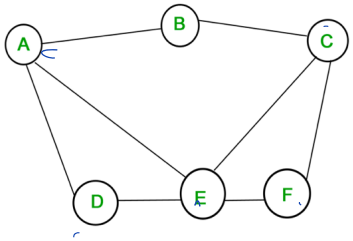

**Weakly Connected**
**(No path from D to A, B or C)**

# GRAPH REPRESENTATION

Graphs can be represented in one of two ways for implementation purposes:
ADJACENCY MATRIX REPRESENTATION or
ADJACENCY LIST REPRESENTATION.

**Adjacency Matrix Representation:** The graph is represented as a 2-d array of n rows and n columns, where each row or column represents a vertex. The element G[i,j] = 1 if there is an edge between vertex i and vertex j, and 0 otherwise.
(Note: If the graph is weighted, G[i,j] will be equal to the weight of the edge (i,j))

**Example 1 (Adjacency Matrix Representation of an undirected, unweighted graph):**




| | | A 0 | B 1 | C 2 | D 3 | E 4 | F 5 |
|---|---|---|---|---|---|---|---|
| A | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| B | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| C | 2 | 0 | 1 | 0 | 0 | 1 | 1 |
| D | 3 | 1 | 0 | 0 | 0 | 1 | 0 |
| E | 4 | 1 | 0 | 1 | 1 | 0 | 1 |
| F | 5 | 0 | 0 | 1 | 0 | 1 | 0 |

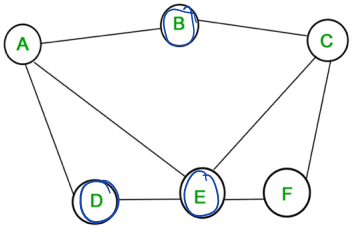
Advantage: Given an edge, it take O(1) time to find if it exists.

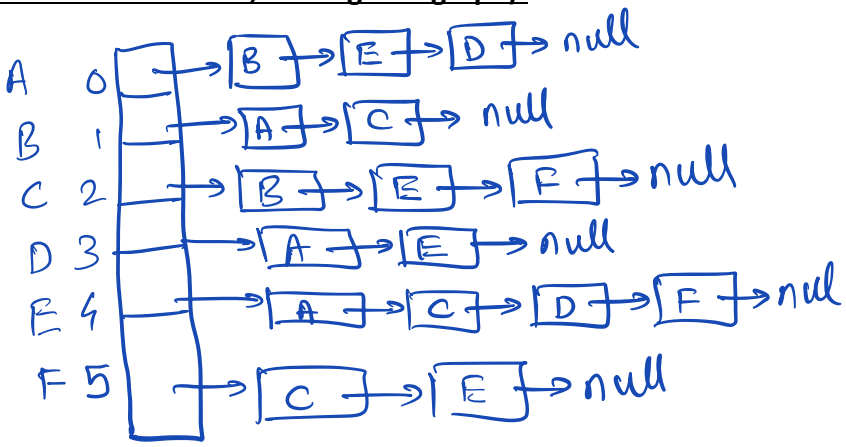
Drawbacks: 1. Waste of space for sparse matrices. (lots of vertices, few edges)

2. Search for all edges connected to a vertex: O(n)

**Adjacency List Representation:** The graph is represented by an array of linked lists. Each linked list represents a vertex, with the entries in the linked list representing the neighbors of that vertex. (Order of storing the elements in the linked list doesn't matter).

**Example 2 (Adjacency List Representation of an undirected, unweighted graph):**

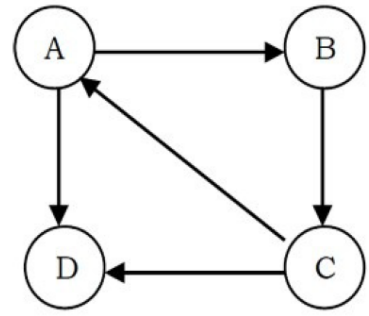


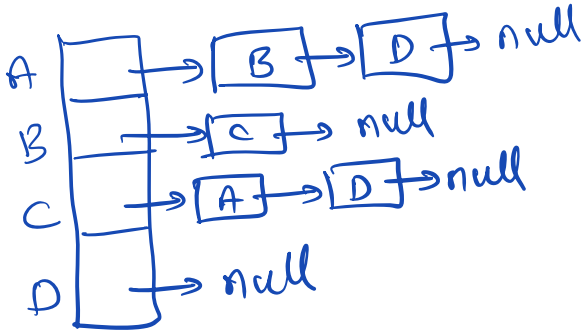

Advantages:
1. Space efficient
2. Search for all edges connected to a vertex → O(1) Best case, O(n) worst case

Drawbacks: Search for a given edge → O(1) Best case, O(n) worst case

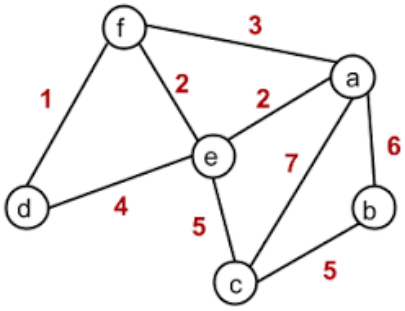
**Example 3 (Representation of an directed graph):**



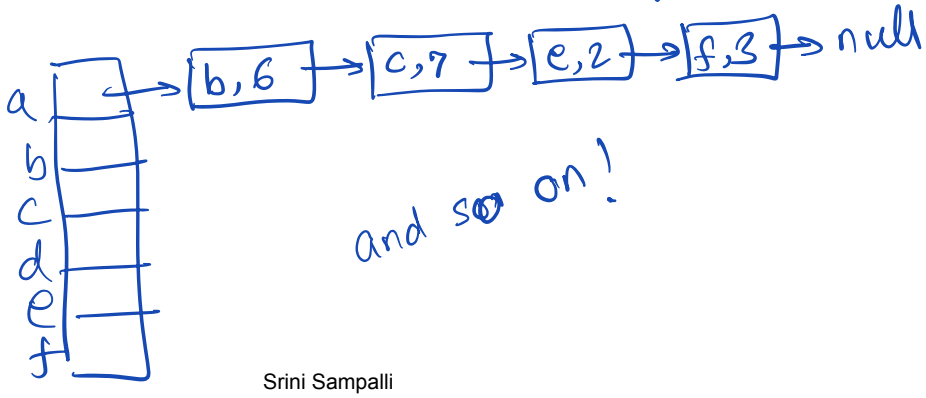

| | A | B | C | D |
|---|---|---|---|---|
| A | 0 | 1 | 0 | 1 |
| B | 0 | 0 | 1 | 0 |
| C | 1 | 0 | 0 | 1 |
| D | 0 | 0 | 0 | 0 |



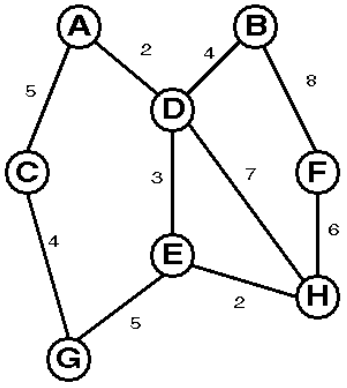

**Example 4 (Representation of a weighted graph):**




| | a | b | c | d | e | f |
|---|---|---|---|---|---|---|
| a | 0 | 6 | 7 | 0 | 2 | 3 |
| b | 6 | 0 | 5 | 0 | 0 | 0 |
| c | 7 | 5 | 0 | 0 | 5 | 0 |
| d | 0 | 0 | 0 | 0 | 4 | 1 |
| e | 2 | 0 | 5 | 4 | 0 | 2 |
| f | 3 | 0 | 0 | 1 | 2 | 0 |



**Exercise:** Study the graph shown in the diagram below and answer the following questions




a) Classify the graph as undirected/directed, unweighted/weighted, unconnected/connected, cyclic or acyclic.

Undirected, weighted, connected, Cyclic

b) List all simple paths from vertex A to vertex E.

ADE, ACGE, ADBFHE, ADHE

c) What is the largest degree in the graph? Which nodes have the largest degree?

4 D

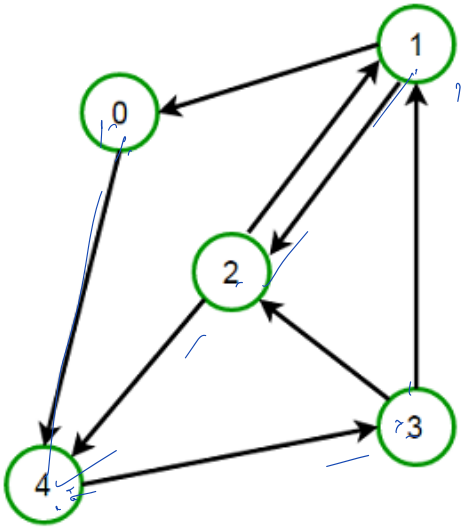
d) What is the smallest degree in the graph? Which nodes have the smallest degree?

2 A, C, G, B, F

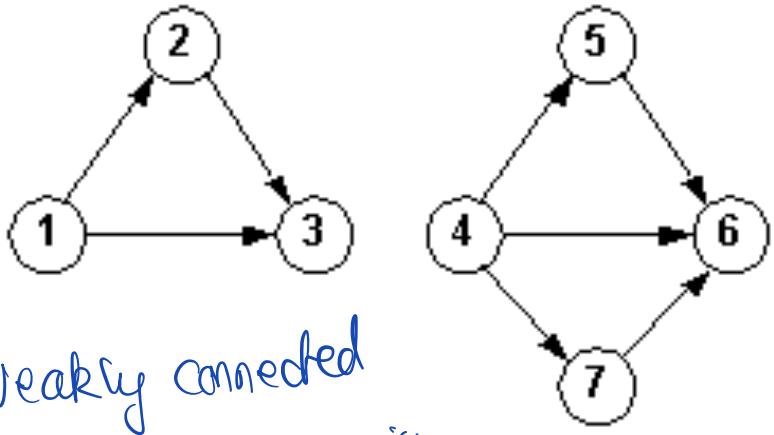
e) Write the adjacency matrix and the adjacency list representations of the graph.

Try it yourself!

**Exercise 2:** Determine if each of the following directed graphs are strongly connected or weakly connected. Give reasons.




Strongly connected!




Weakly connected

2 → 1 not possible

Weakly connected

4 cannot be visited from other nodes.