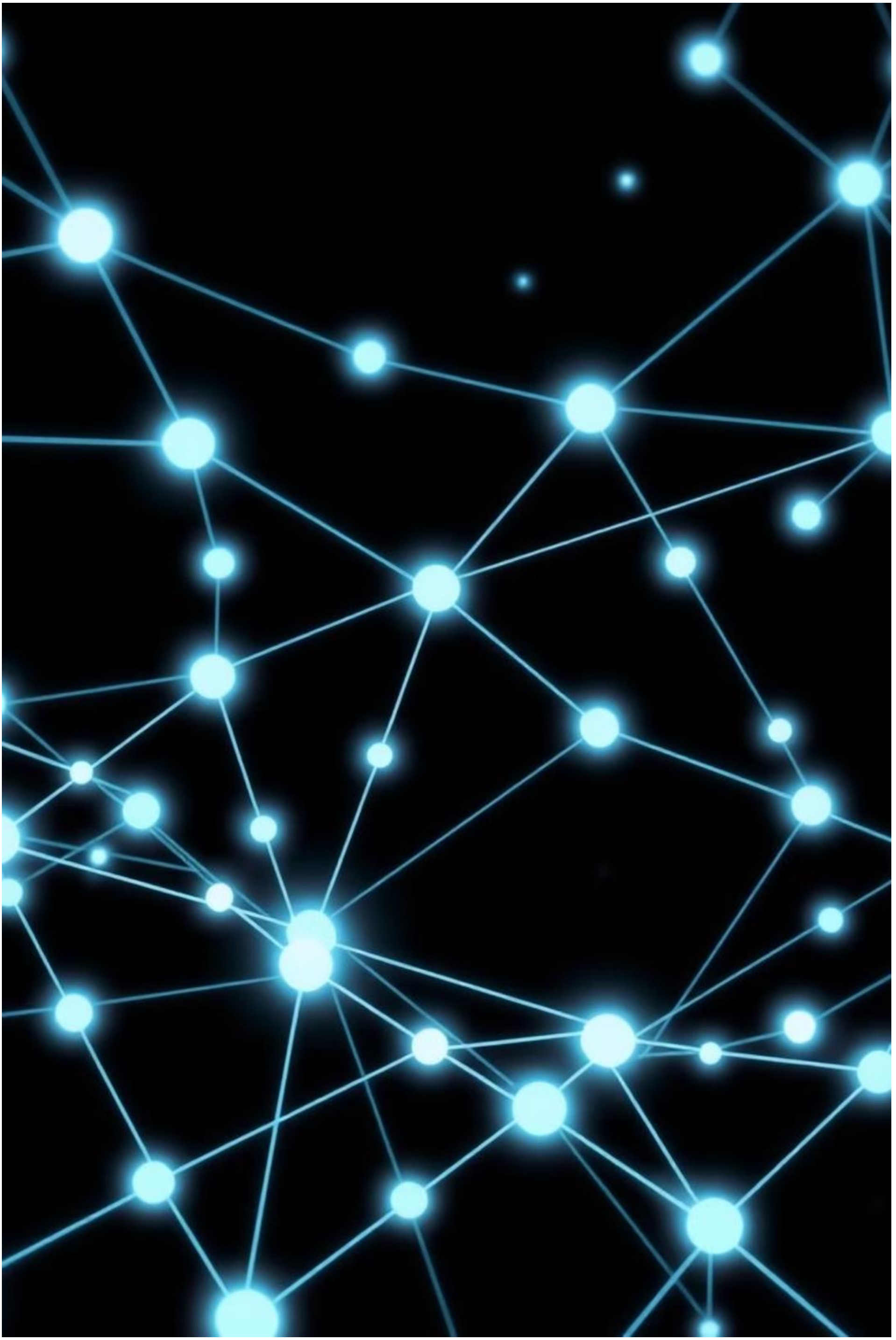


Data Structures

Lecture 7: Graph

Instructor:

Md Samsuddoha

Assistant Professor

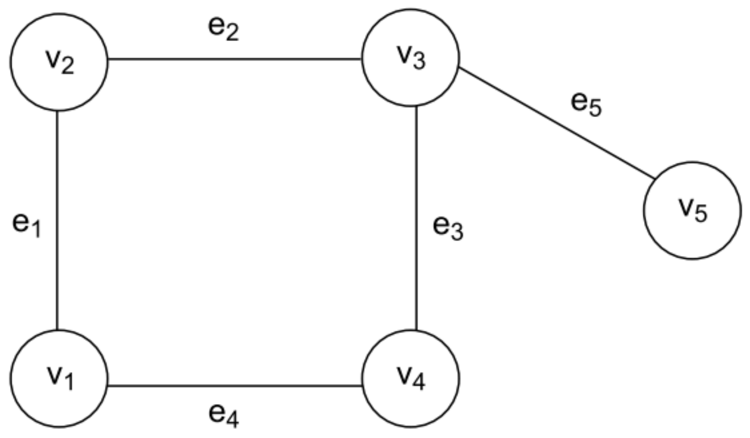
Dept of CSE, BU

Contents

Graph definition

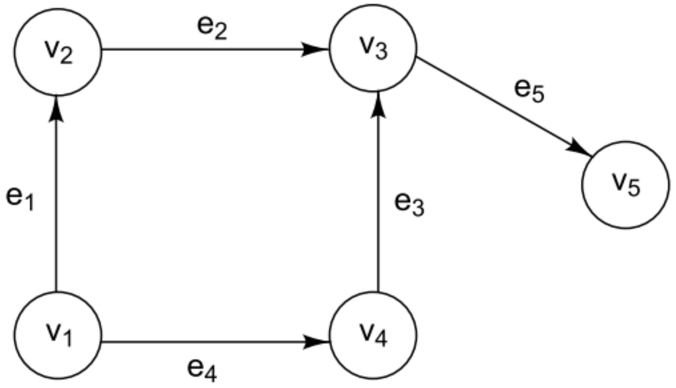
- A Graph is a non-linear data structure that consists of vertices (nodes) and edges.
- A graph ***G*** consists of the following elements:
  - A set ***V*** of vertices or nodes, where ***V={v1, v2, v3,……vn}***
  - A set ***E*** of edges also called arcs where, ***E={e1, e2, e3, …., en}***
  - Hence, ***G=(V,E)***

  - ***If e=(u,v) and e=(v, u) means same then the graph is undirected.***

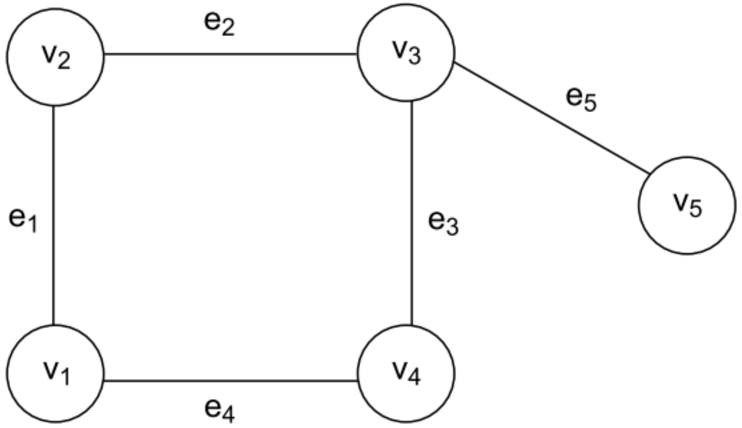


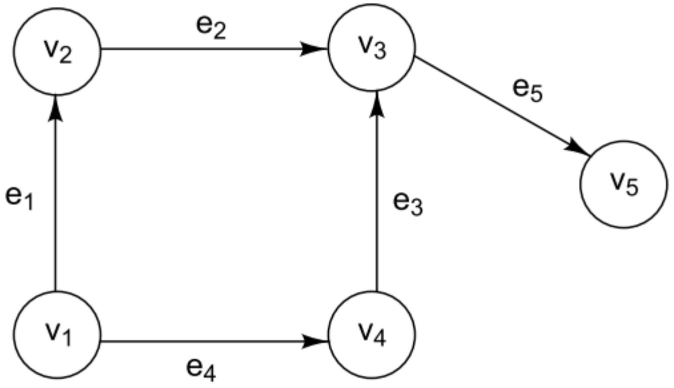
Directed Graph

- If we replace each edge of the Graph ***G*** with ***arrows***, then it will become a ***directed graph or diagraph.***
- In this graph, the set of vertices and edges are:
  - ***V(G)={v1, v2, v3, v4, v5}***
  - ***V(E)={(v1, v2), (v2, v3), (v1, v4), (v4, v3), (v3, v5)}***



Key Terms	Description
Adjacent node	If e (u, v) represents an edge between u and v vertices then both u and v are called adjacent to each other. That means, u is adjacent to v and v is adjacent to u.
Predecessor node	If e (u, v) represents a directed edge from u to v then u is a predecessor node of v.
Successor node	If e (u, v) represents a directed edge from u to v then v is a successor node of u.
Degree	Degree of a vertex is the number of edges connected to a vertex. For example, in the graph shown in Fig. 9.1, the degree of vertex v_3 is 3.
Indegree	In a directed graph, indegree of a vertex is the number of edges ending at the vertex.
Outdegree	In a directed graph, outdegree of a vertex is the number of edges beginning at the vertex.
Path	A path is a sequence of vertices each adjacent to the next. For example, in the graph shown in Fig. 9.2, the path between the vertices v_1 and v_5 is v_1–v_2–v_3–v_5.
Cycle	It is a path that starts and ends at the same vertex.
Loop	It is an edge whose endpoints are same that is, e = (u, u).
Weight	It is a non-negative number assigned to an edge. It is also called length.
Order	Order of a graph is the number of the vertices contained in the graph.
Labeled Graph	It is a graph that has labeled edges.
Weighted Graph	It is a graph that has weights assigned to each of its edges.
Connected Graph	It is an undirected graph in which there is a path between each pair of nodes.
Strongly Connected Graph	It is a directed graph in which there is a route between each pair of nodes.
Complete Graph	It is an undirected graph in which there is a direct edge between each pair of nodes.
Tree	It is a connected graph with no cycles.





Types of Graphs

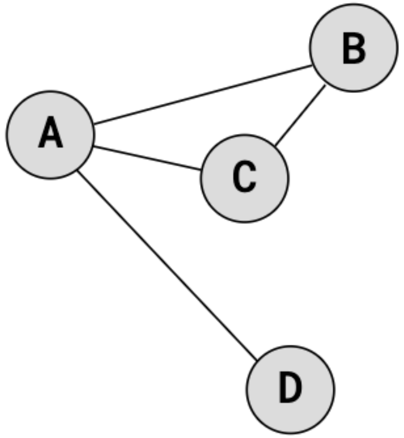
- Based on Edge Direction
  - Undirected Graph
  - Directed Graph
- Based on Weight of Edges
  - Weighted Graph
  - Unweighted Graph

Graph Representation

- Adjancency Matrix (2D Array)
- Adjacency List (Linked List)

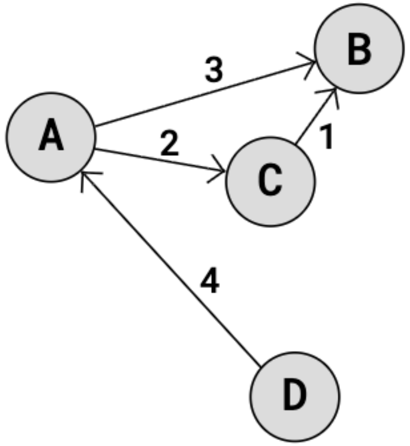
Adjacency Matrix

- The Adjacency Matrix is a ***2D array (matrix)*** where each cell on index ***(i,j)*** stores information about the edge from vertex ***i*** to vertex ***j***.
- 2D array → A[V][V]
- A[i][j] = 1, if edge exists between vertex ***i*** and ***j***, else ***0***.



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

An undirected Graph and the adjacency matrix

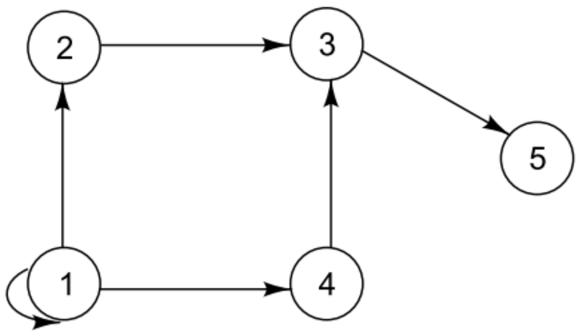


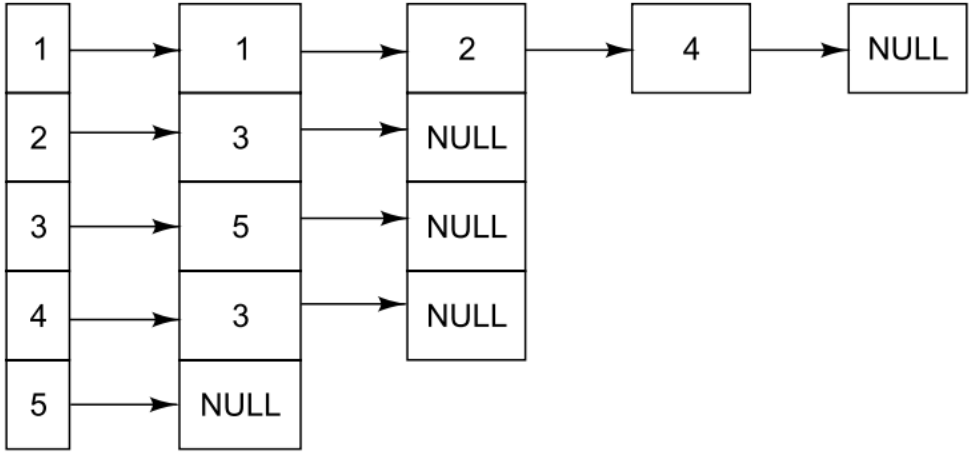
	A	B	C	D
A	0	3	2	0
B	0	0	0	0
C	0	1	0	0
D	4	0	0	0

A directed and weighted Graph, and its adjacency matrix.

Adjacency List

- Array of ***linked lists (or dynamic arrays)***
- Each vertex stores a list of its adjacent vertices.
- Suitable for ***sparse*** graphs.



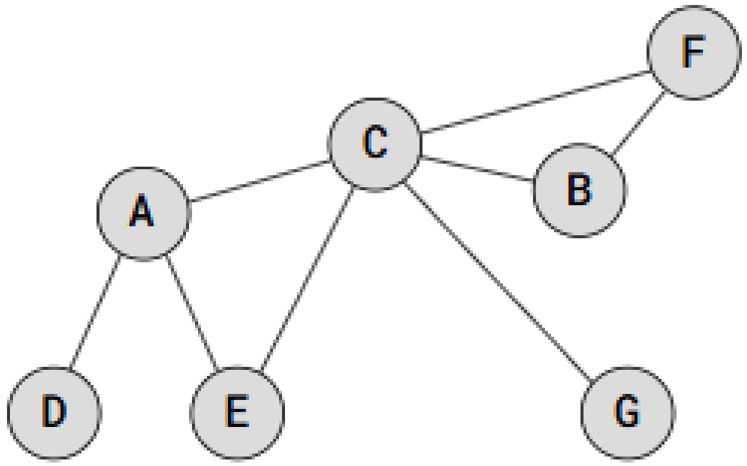


Graph Traversal Techniques

- **Depth First Search (DFS):**
  - Explores as far as possible along each branch before backtracking.
  - Uses a stack (or recursion).
- **Breadth First Search (BFS)**
  - Explores all neighbors of a vertex before moving to the next level.
  - Uses a queue.

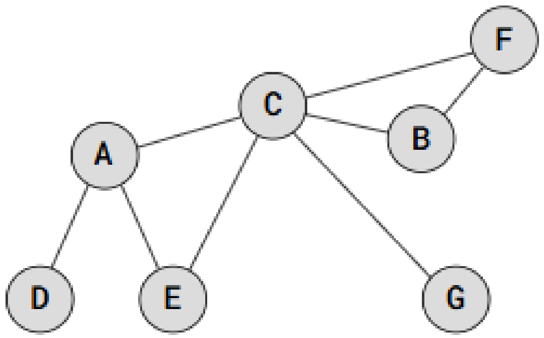
Depth First Search (DFS)

- Start DFS traversal on a vertex.
- Uses a stack (or recursion).
- Do a recursive DFS traversal on each of the adjacent vertices as long as they are not already visited.
- Path: ***D,A,C,B,F,E,G***



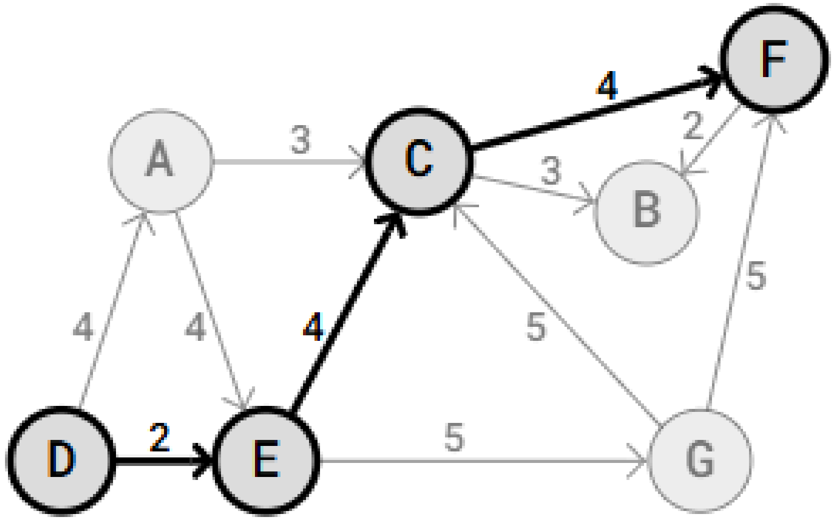
Breath First Search (BFS):

- Put the starting vertex into the **queue**.
- For each vertex taken from the queue, visit the vertex, then put all unvisited adjacent vertices into the queue.
- Continue as long as there are vertices in the queue.
- Path: ***D,A,C,E,B,F,G***



Shortest Path

- To solve the shortest path problem means to find the ***shortest possible route or path*** between two vertices (or nodes) in a Graph.
- In the shortest path problem, a Graph can represent anything from a ***road network*** to a communication network, where the vertices can be intersections, cities, or routers, and the edges can be ***roads, flight paths, or data links.***
- ***Solutions (SPP): Dijkstra's algorithm*** *and the* ***Bellman-Ford algorithm*** *find the shortest path from one start vertex, to all other vertices.*



Graph Algorithms

Algorithm	**Purpose**	**Concept**
Dijkstra’s Algorithm	Shortest path (single source)	Greedy algorithm
Floyd-Warshall	All-pairs shortest paths	Dynamic programming
Prim’s Algorithm	Minimum Spanning Tree	Greedy (build tree step-by-step)
Kruskal’s Algorithm	Minimum Spanning Tree	Greedy (sort edges, union-find)
Topological Sort	Ordering in a DAG	DFS-based sorting
Bellman-Ford	Shortest path (with negative weights)	Dynamic programming

Real life applications of Graphs

- ***Social Networks:*** Each person is a vertex, and relationships (like friendships) are the edges. Algorithms can suggest potential friends.
- ***Maps and Navigation***: Locations, like a town or bus stops, are stored as vertices, and roads are stored as edges. Algorithms can find the shortest route between two locations when stored as a Graph.
- ***Internet***: Can be represented as a Graph, with web pages as vertices and hyperlinks as edges.
- ***Biology***: Graphs can model systems like neural networks or the spread of diseases.

References

- **<u>Chapter 9:</u>**
  - **Data Structures using C** by E. Balagurusamy

Thank You