Graphs - Part 1

Introduction to Graphs

What is a Graph?

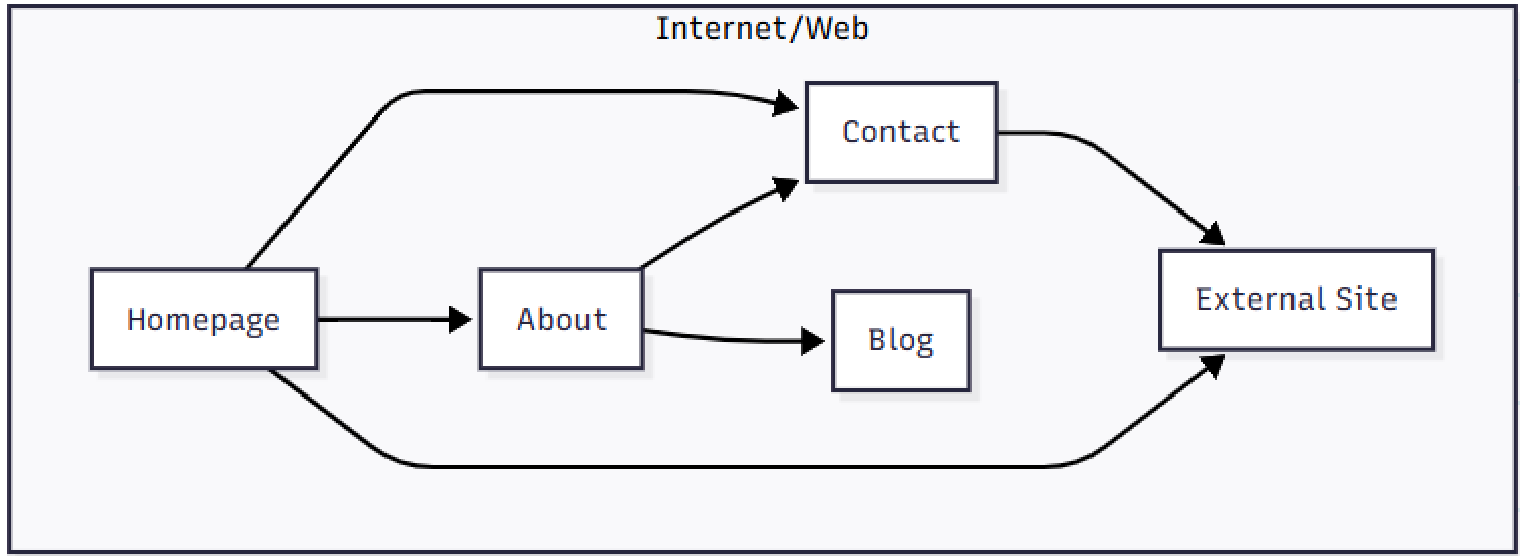
- A graph is a non-linear data structure consisting of **vertices** (nodes) and **edges** (connections between nodes).
- Unlike trees, graphs can have cycles and multiple paths between nodes.

Real-World Examples

- Social Networks: Users are vertices, friendships are edges
- Maps: Cities are vertices, roads are edges
- Internet: Webpages are vertices, links are edges
- Computer Networks: Devices are vertices, connections are edges

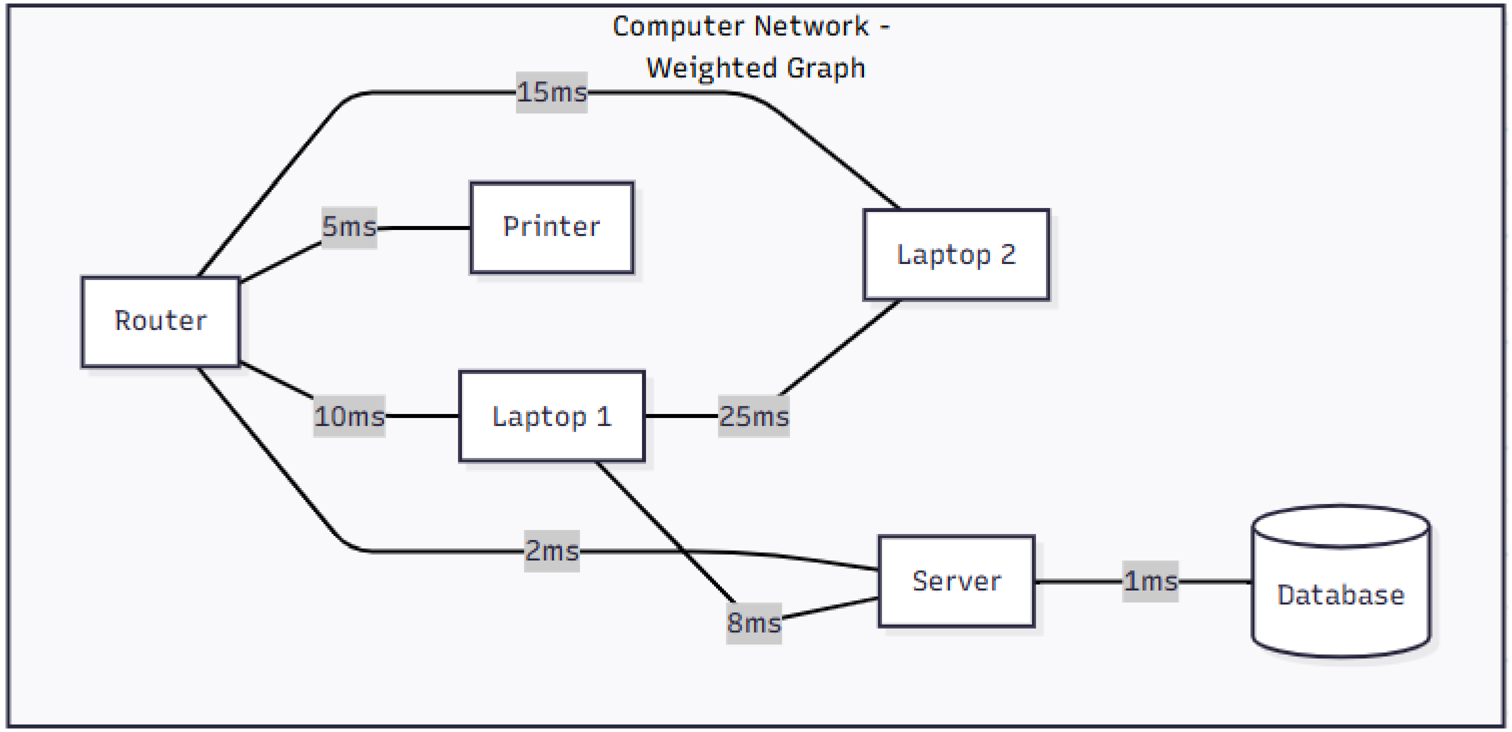
Internet as a Graph

- Webpages are vertices, links are edges

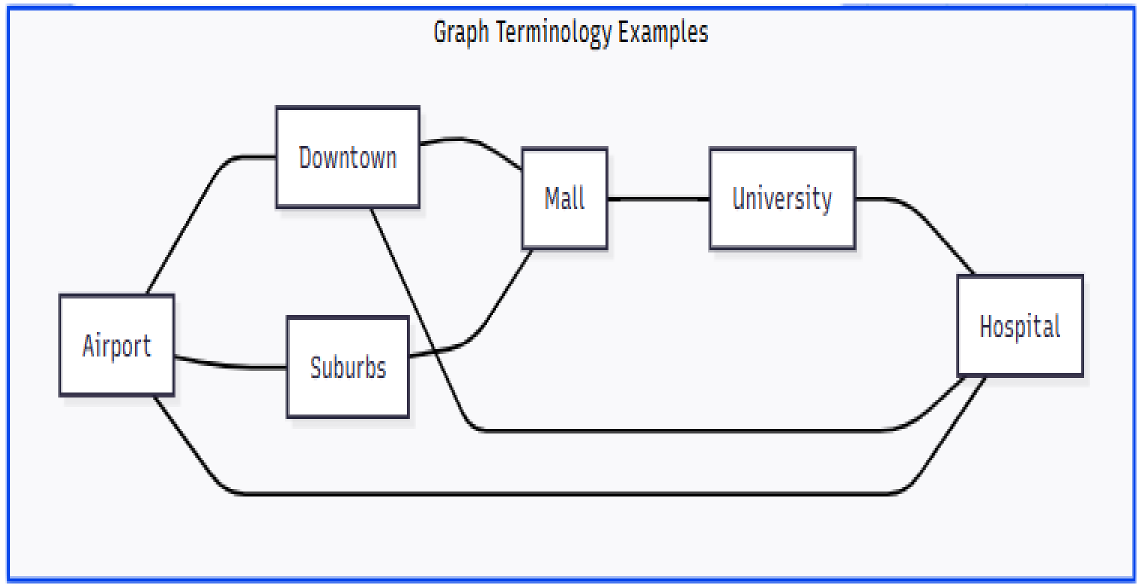


Computer Network as a Graph

- Devices are vertices, connections are edges



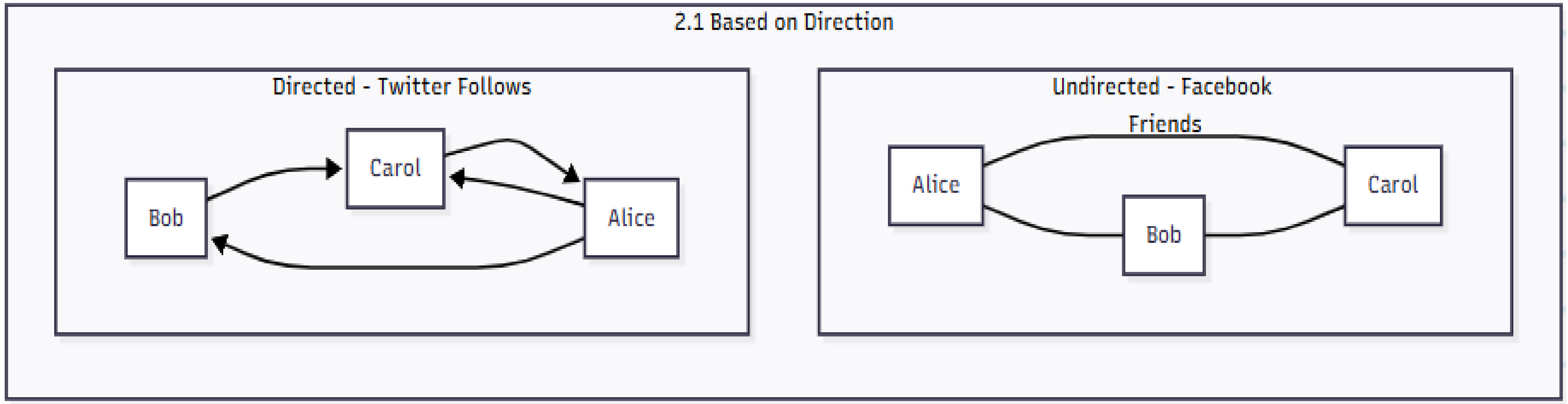
Basic Terminology



- **VERTEX/NODE**: Each location (Airport, Downtown, Mall, etc.)
- **EDGE**: Each road/route between locations“
- **ADJACENT**: Downtown & Mall are adjacent (connected by direct road)
- **DEGREE**: Downtown has degree 3 (connected to Airport, Mall, Hospital)
- **PATH**: Airport → Downtown → Mall → University“
- **CYCLE**: Airport → Downtown → Hospital → Airport"

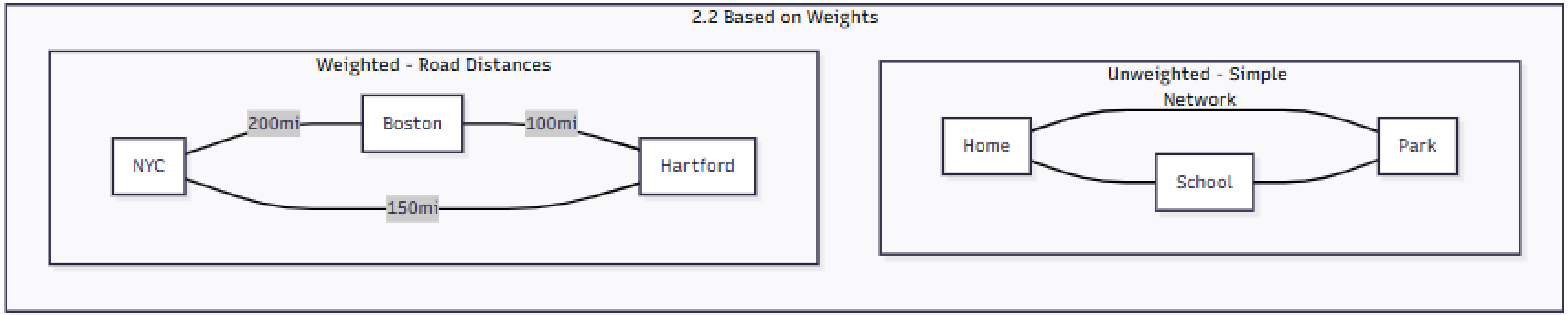
Types of Graphs : Based on direction

- **Undirected graph** : Edges have no direction (Facebook friendships)
    - If A connects to B, then B connects to A
- **Directed graph** : Edges have direction (Twitter followers)
    - If A → B, it doesn't mean B → A



Types of Graphs : Based on Weights

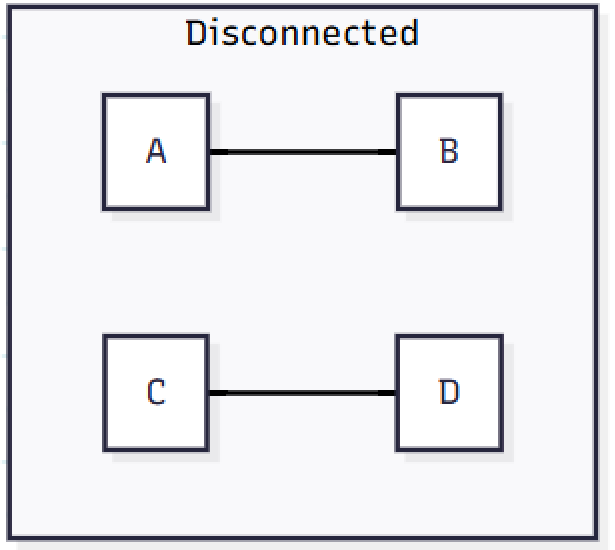
- **Unweighted Graph** : All edges have equal importance(Simple friendship network)
- **Weighted Graph** : Edges have associated weights/costs (Road network with distances)

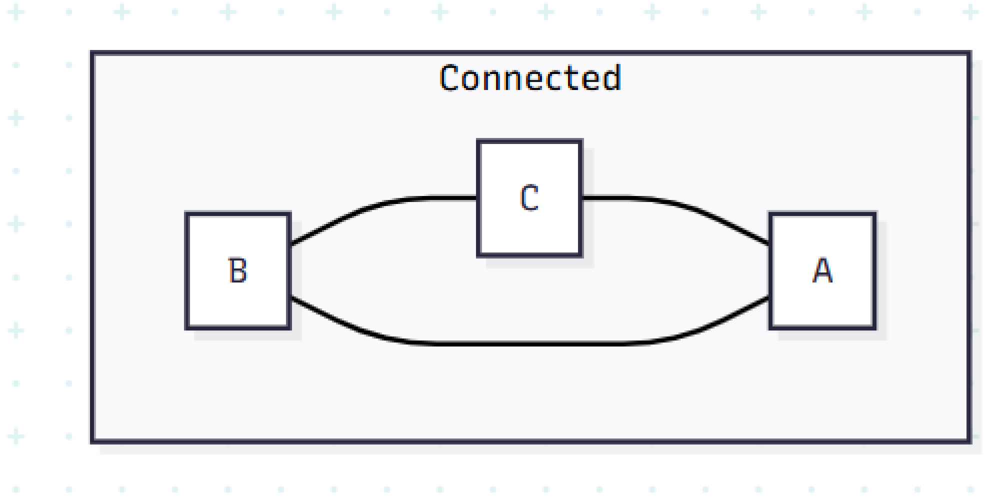


Types of Graphs : Special

- **Disconnected Graphs** : Some vertices are not reachable from the others

- **Connected Graphs** : Path exists between every pair of vertices

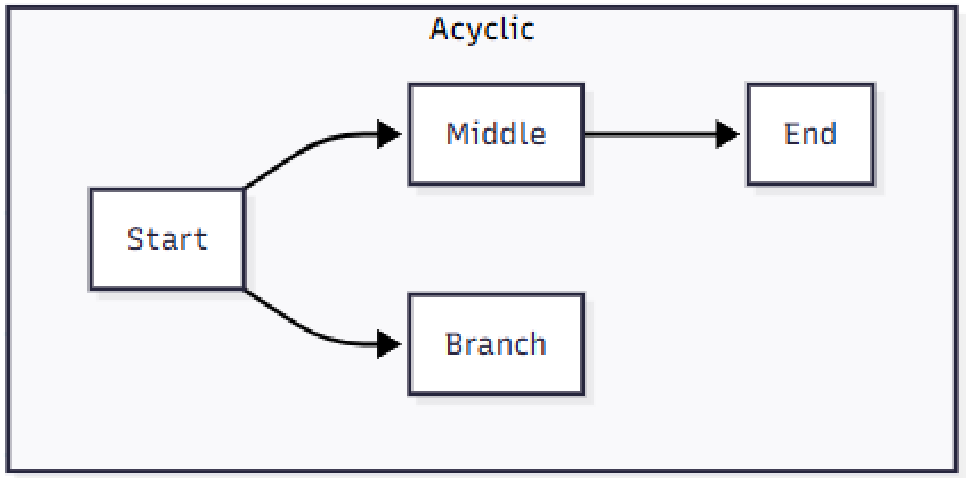


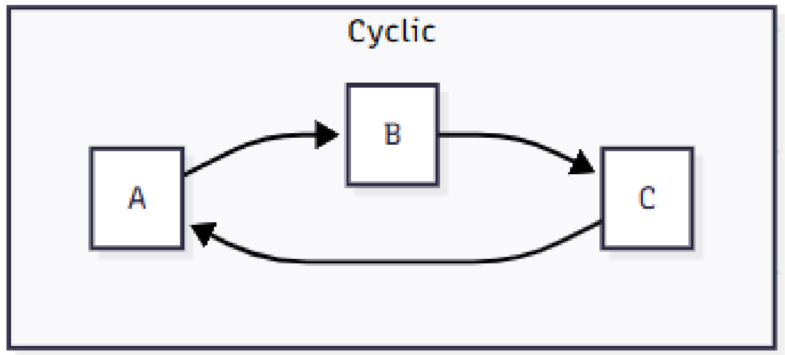


Types of Graphs : Special

- **Cyclic Graphs** : Contains at least one cycle

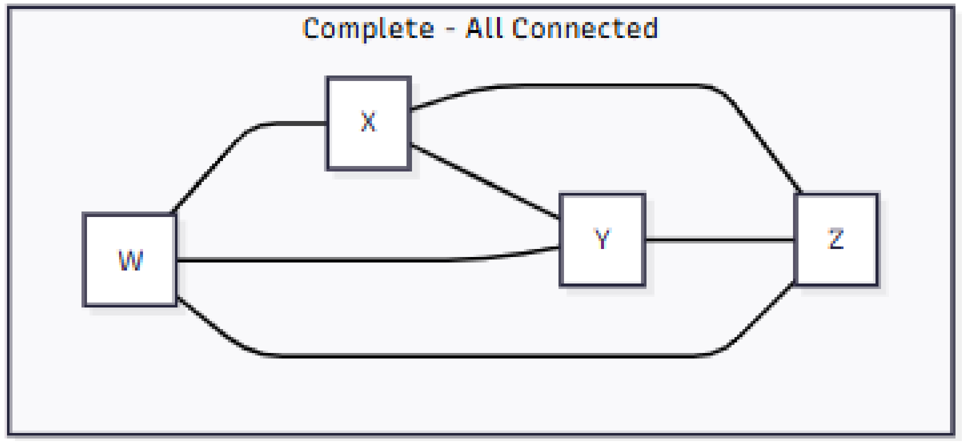
- **Acyclic Graphs** : Contains no cycles

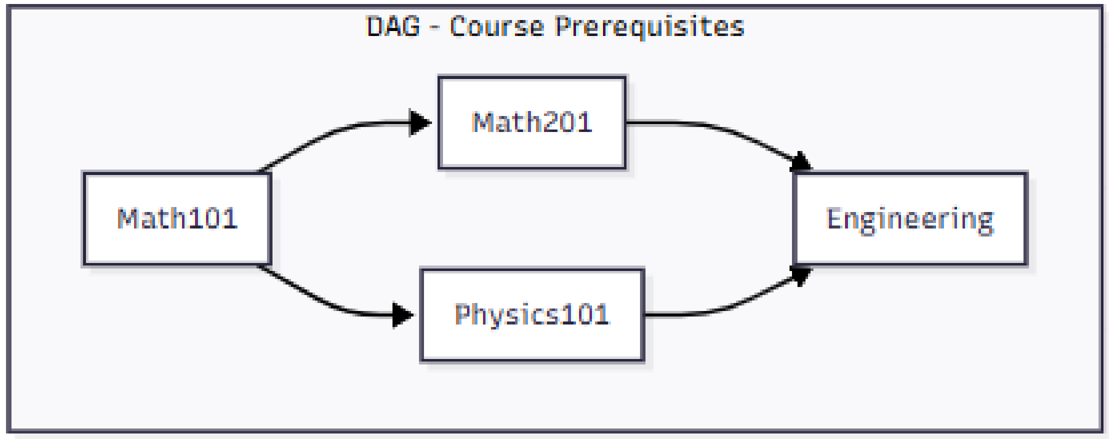




Types of Graphs : Special

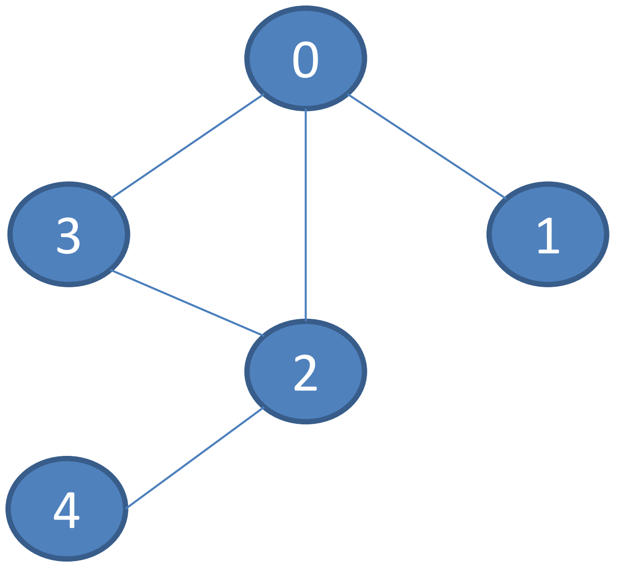
- **Directed Acyclic Graphs (DAG)**
- **Complete Graph** : Every vertex is connected to every other vertex





Graph Representations

- Adjacency List & Adjacency Matrix



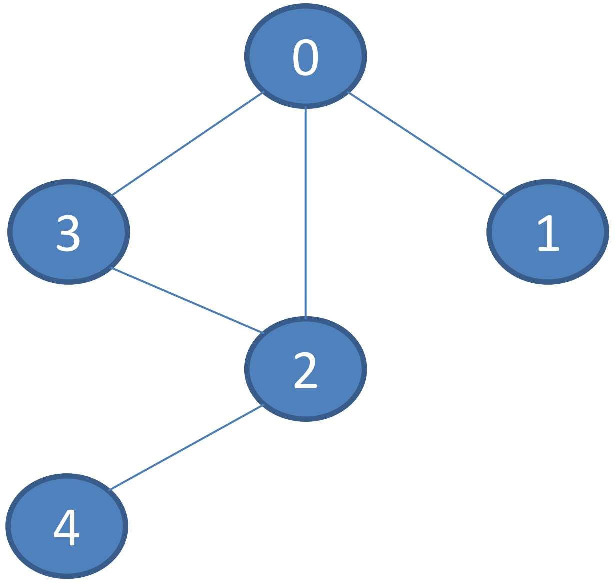
Vertex	Connected to
0	[1 2 3]
1	[0]
2	[0 3 4]
3	[0 2]
4	[2]

	0	1	2	3	4
0	0	1	1	1	0
1	1	0	0	0	0
2	1	0	0	1	1
3	1	0	1	0	0
4	0	0	1	0	0

Implementation : Adjacency List vs. Matrix

- Java implementation/data structure:

Vertex	Connected to
0	[1 2 3]
1	[0]
2	[0 3 4]
3	[0 2]
4	[2]



	0	1	2	3	4
0	0	1	1	1	0
1	1	0	0	0	0
2	1	0	0	1	1
3	1	0	1	0	0
4	0	0	1	0	0

A list of list : List<List<Integer>>

A 2-D array : matrix[i][j]