CSC 316 – Data Structures
Introduction to Graphs

George N. Rouskas

Department of Computer Science

North Carolina State University

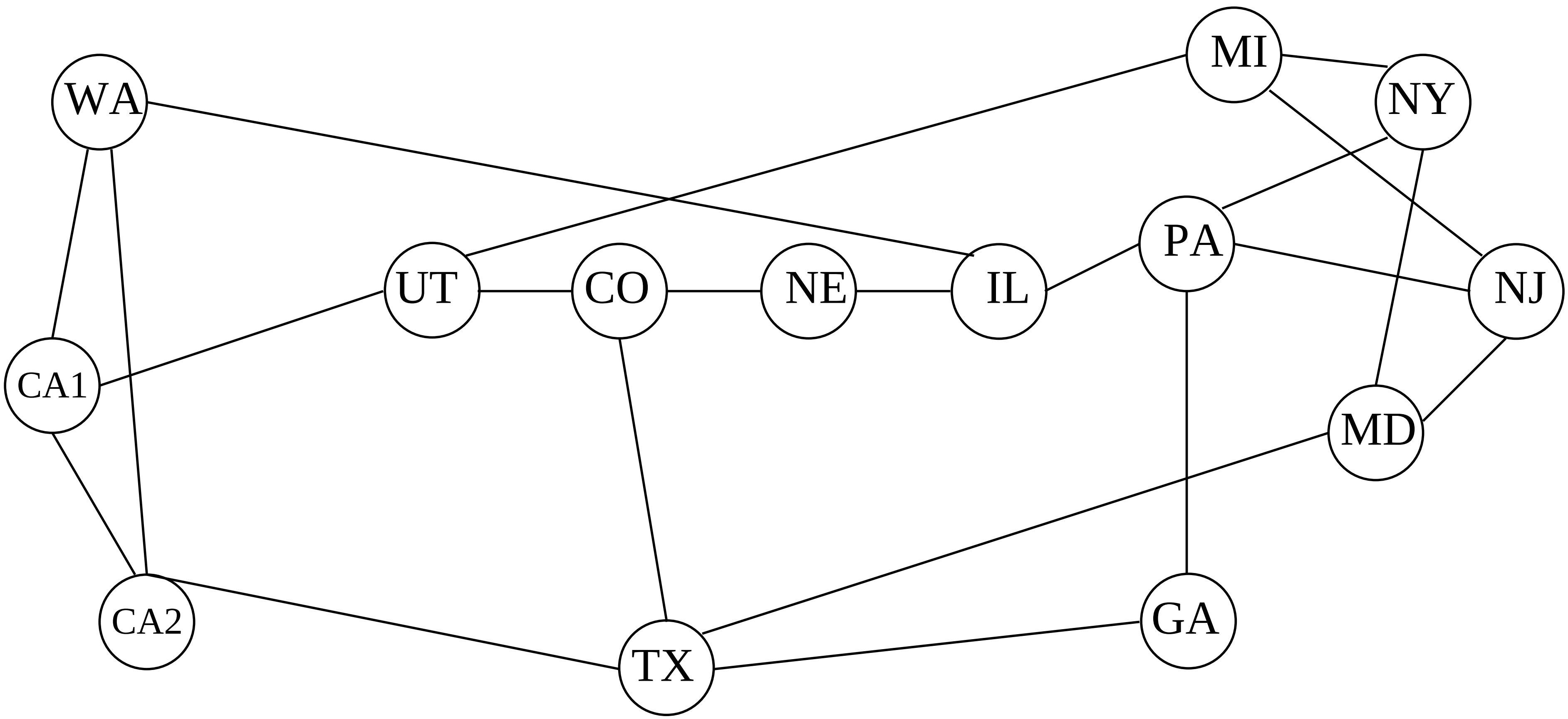
Outline

1. Definition
2. Terminology
3. Implementation
4. Trees as Graphs

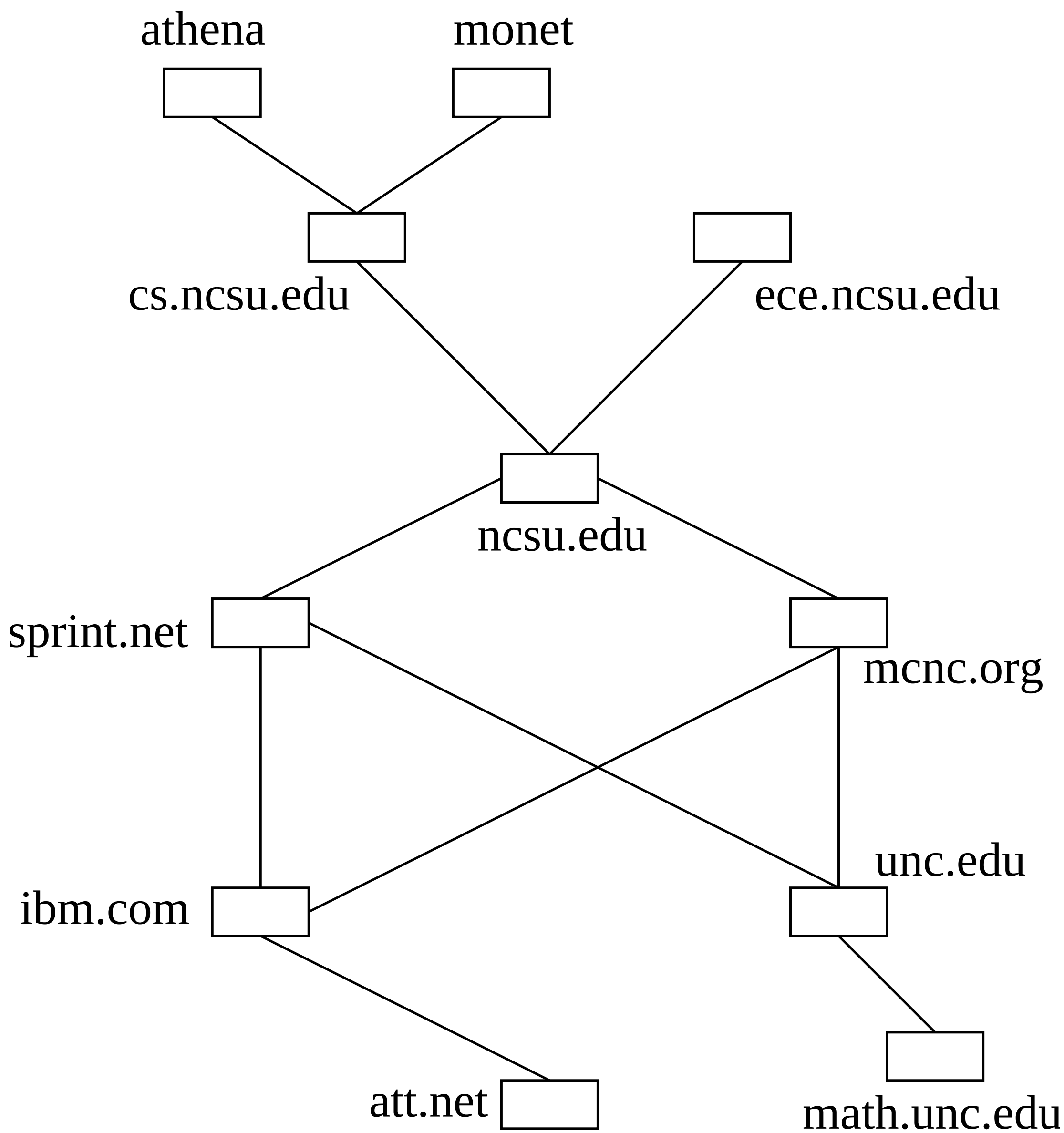
Undirected Graphs

- An undirected graph is a pair (V, E):
  - $V = \{v_1, v_2, \cdots, v_n\}$ is a set of nodes $\longrightarrow$ vertex set
  - $E = \{e_1, e_2, \cdots, e_m\}$ is a set of unordered pairs of vertices $\longrightarrow$ edge set
- Vertices:
  - represent arbitrary entities (e.g., cities)
  - store auxiliary information (e.g., city code)
- Edge $e = \{v, u\} \in E$
  - implies a relationship between v and u (e.g., highway exists between two cities)
  - relationship is symmetrical $\longrightarrow$ order of vertices unimportant
  - v and u are adjacent or neighbors $\longrightarrow$ endpoints of edge e

Undirected Graph Example



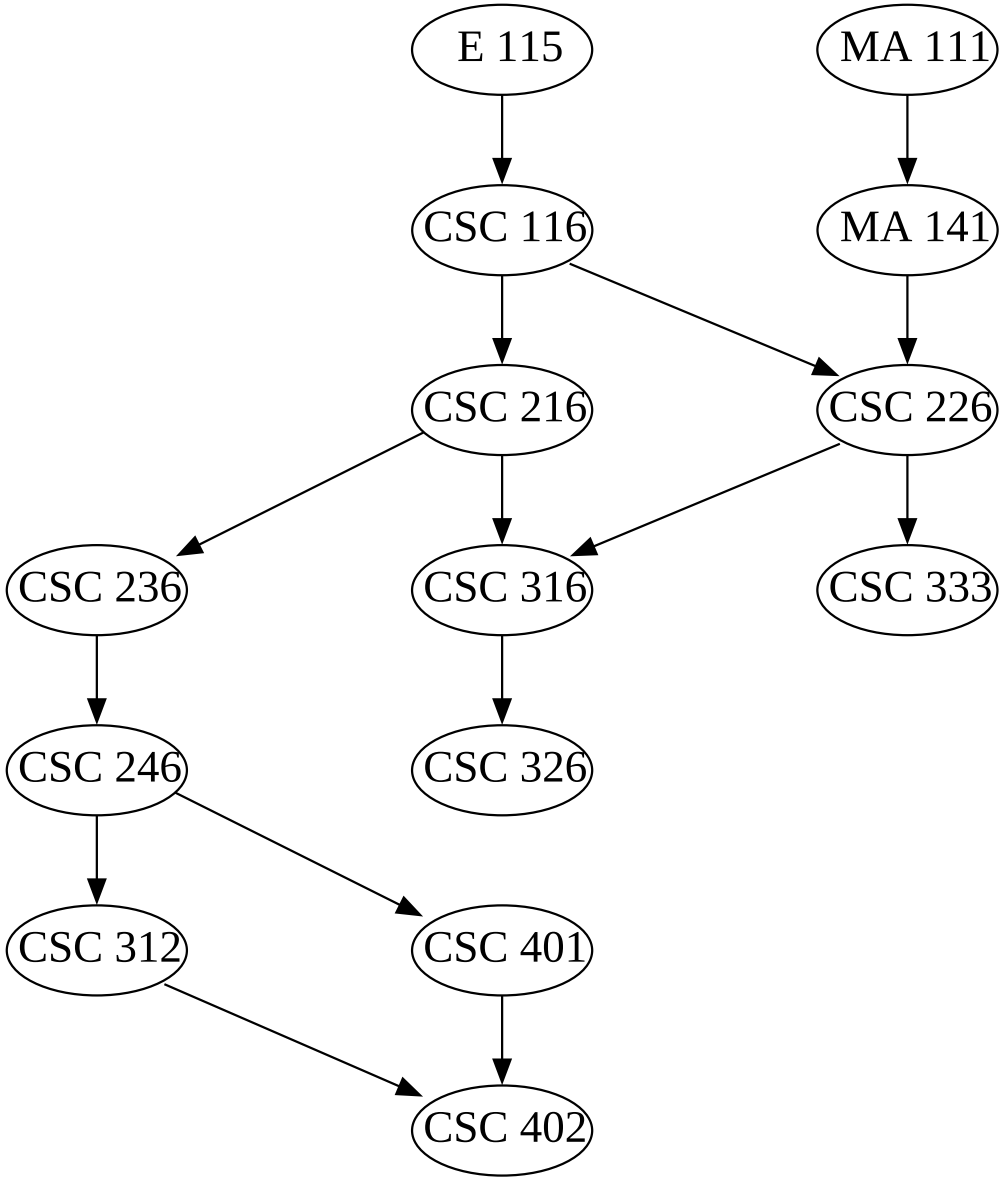
Undirected Graph Example (2)



Directed Graphs

- A directed graph is a pair (V, A)
  - V is the vertex set
  - $A = \{a_1, a_2, \cdots, a_m\}$ is a set of ordered pairs of vertices $\longrightarrow$ arc set
- Arc $a =< v, u >\in A$
  - departs from v and enters u
  - u is adjacent to (neighbor of) of v
  - the reverse is not true unless $< u, v >\in A$

Directed Graph Example

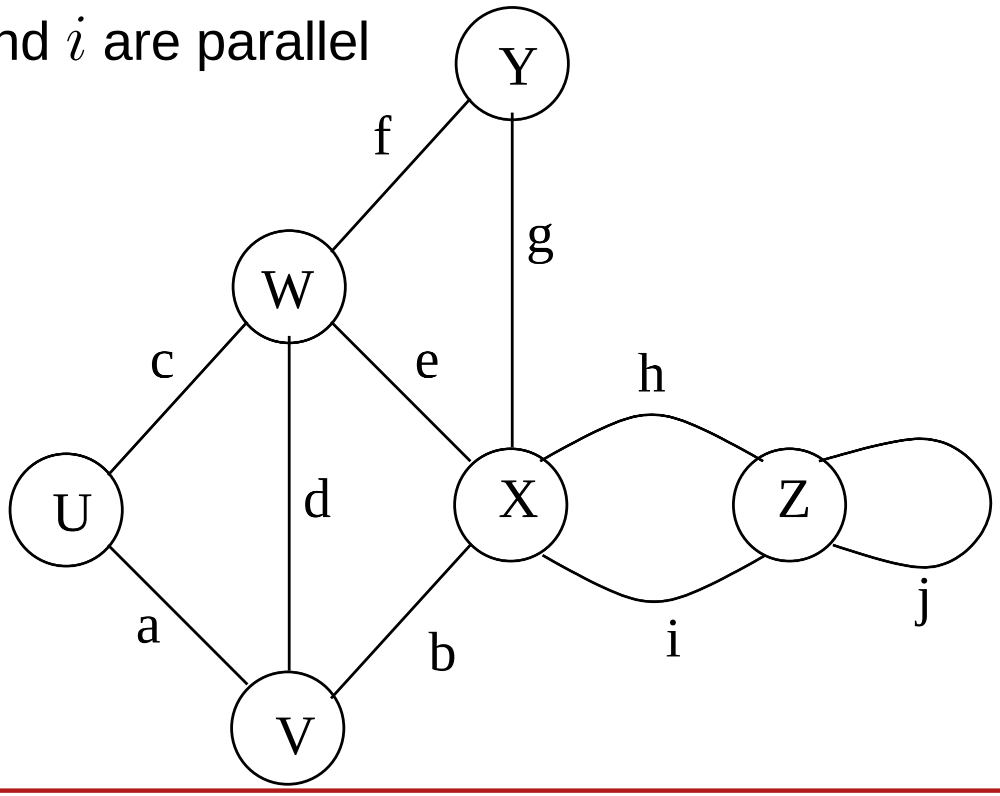


Applications

- Transportation networks:
  - highway network
  - flight network
- Computer networks:
  - campus network
  - internet
  - web
- Databases: entity-relationship diagrams
- Electronic circuits:
  - printed circuit board
  - integrated circuit
- Scheduling
- . . .

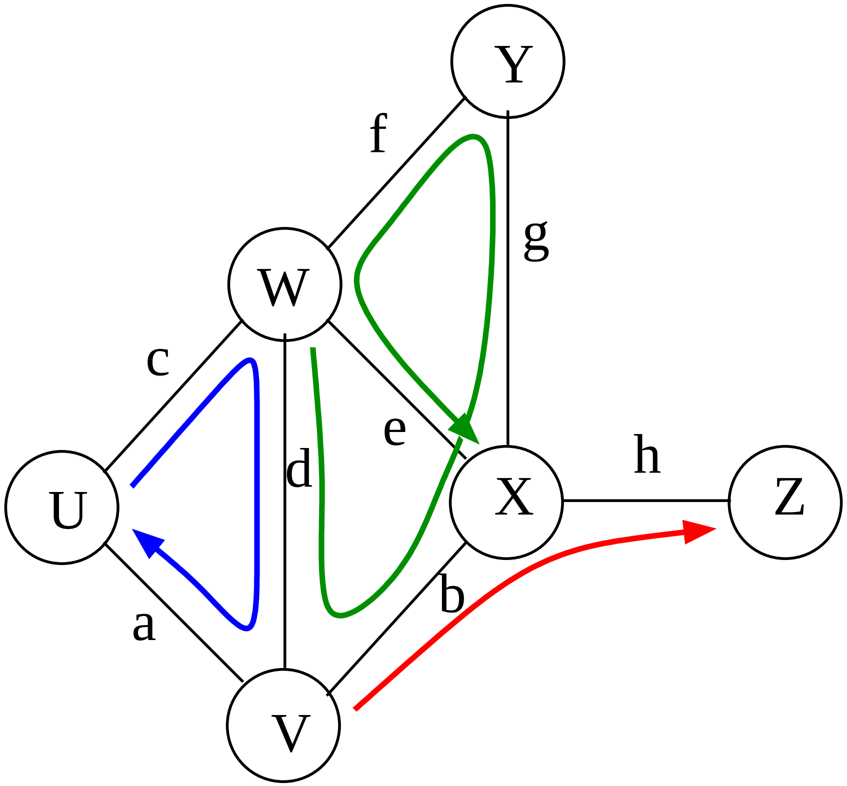
Terminology

- Endpoints of an edge $\longrightarrow$ U, V endpoints of a
- Edges incident on a vertex $\longrightarrow$ a, b, d incident on V
- Adjacent vertices $\longrightarrow$ U and V are adjacent
- Degree of a vertex $\longrightarrow$ X has degree 5
- Parallel edges $\longrightarrow$ h and i are parallel
- Self-loop $\longrightarrow$ edge j



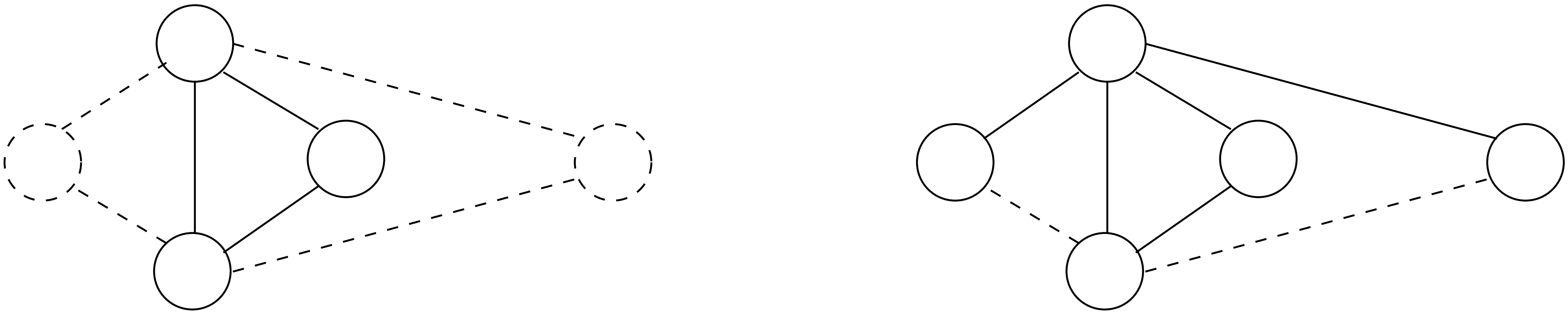
Terminology: Paths

- Path from vertex v_0 to n:
  - a sequence of vertices $v_0, v_1, \cdots, v_n$
  - $\{v_0, v_1\}, \{v_1, v_2\}, \cdots, \{v_{n-1}, v_n\}$ are edges of the graph
- Simple path: all vertices in the path are distinct
- Cycle: a path with $v_0 = v_n$, no two successive edges are the same



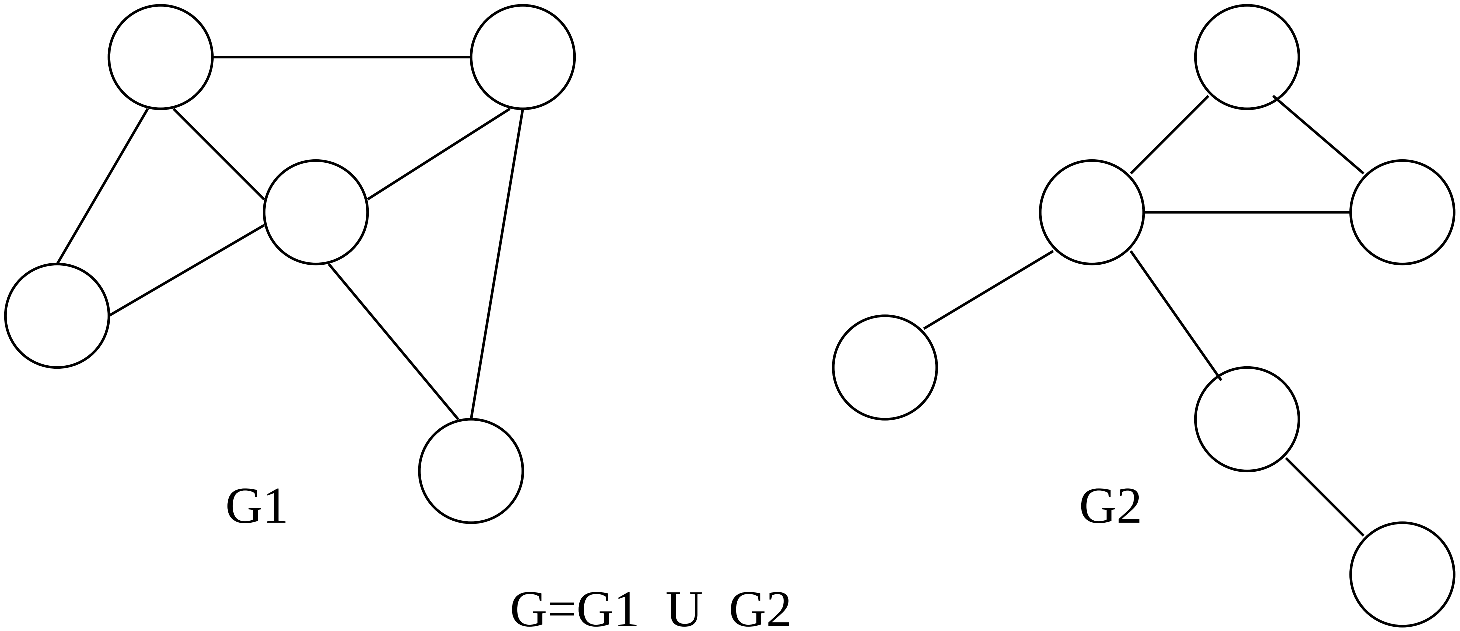
Terminology: Subgraphs

- Graph $G = (V, E)$
- Subgraph $G' = (V', E')$ of G
  - $V' \subseteq V$
  - $E' \subseteq E$
- Spanning subgraph: $V' = V$



Terminology: Connected Components

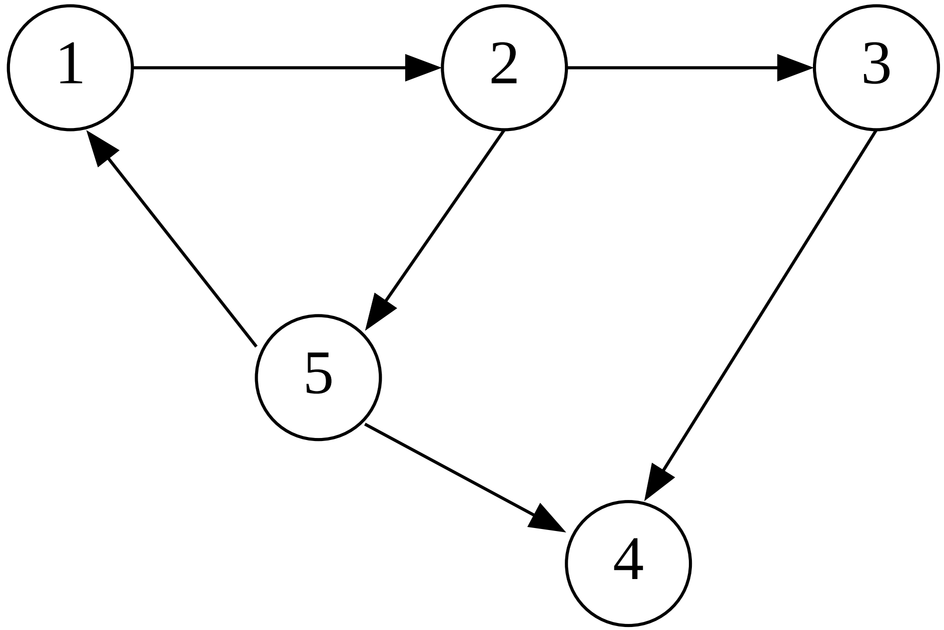
- Connected graph: $\forall u, v \in V$, there exists a path between u and v $\longrightarrow$ each vertex is reachable from any other vertex
- Connected component: a maximal connected subgraph of a graph G
- Two extreme cases:
  - connected graph $\longrightarrow$ single connected component
  - graph with no edges $\longrightarrow$ each vertex is a connected component



G=G1 U G2

Terminology: Connected Components (2)

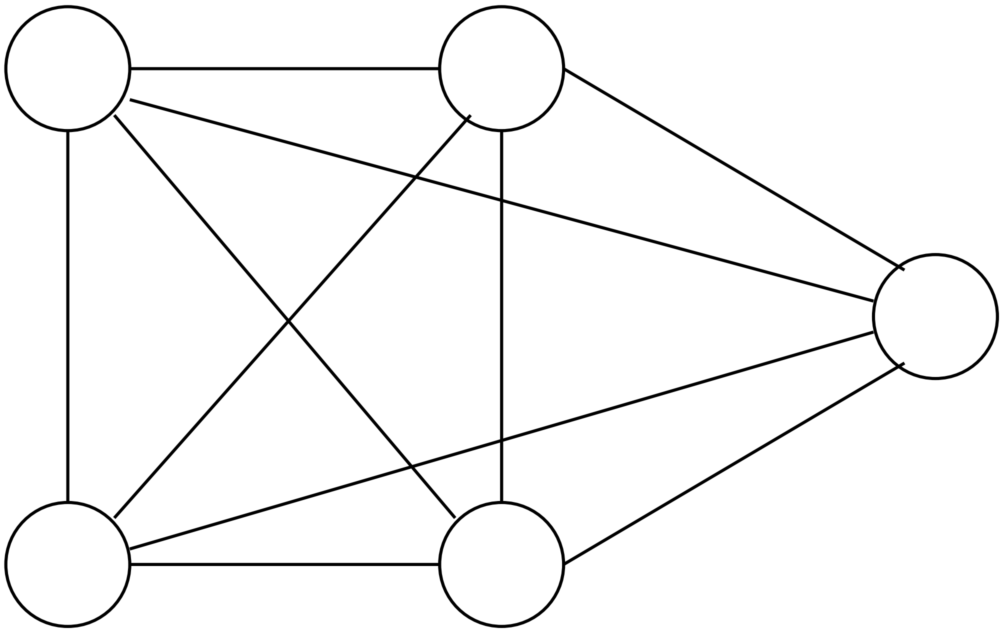
- Directed graph (digraph) $G = (V, A)$
- Strongly connected graph: connected in both directions
- Strongly connected component: a maximal strongly connected subgraph of G



Complete Graphs

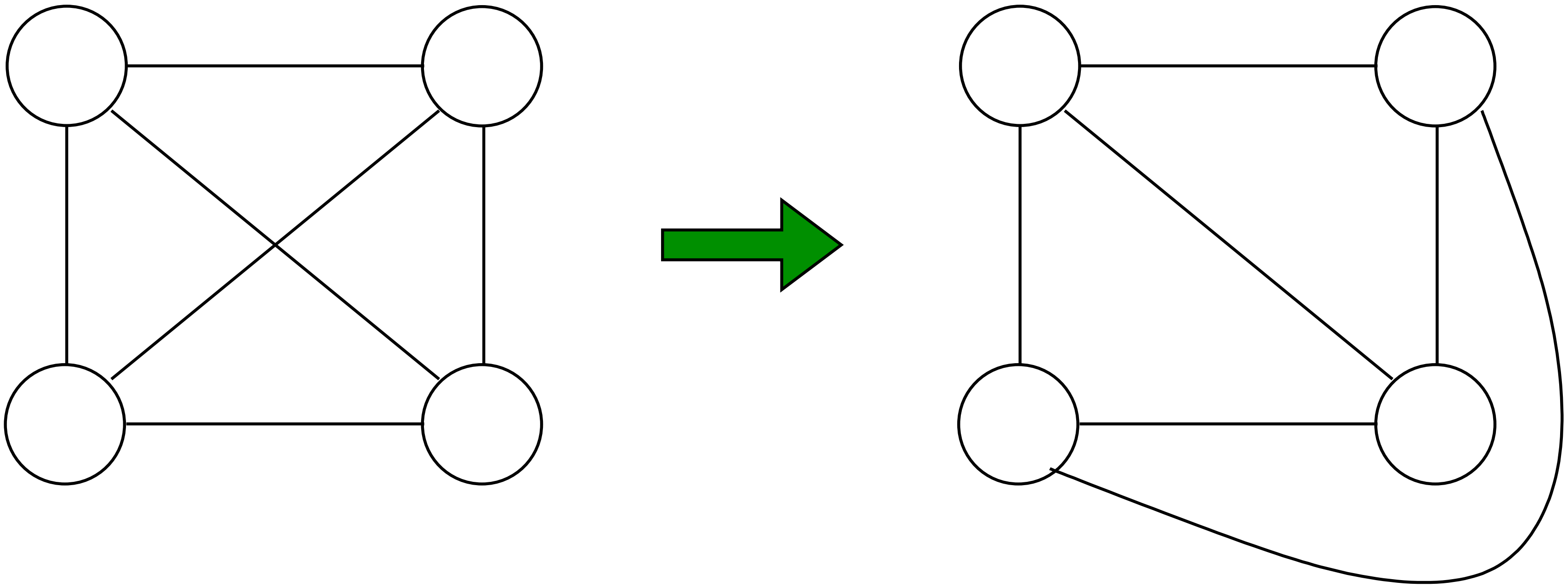
- Undirected graph with an edge between every distinct pair of vertices
- Property:

$$|E| = \frac{|V|\,(|V|-1)}{2}$$



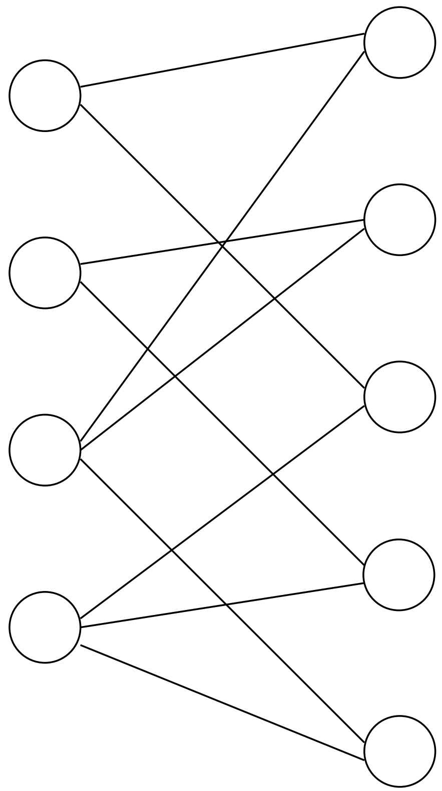
Planar Graphs

- Can be drawn so that their edges intersect only at the vertices



Bipartite Graphs

- The vertex set V can be partitioned into two sets, V_1 and V_2 $\rightarrow V_1 \cap V_2 = \phi$, $V_1 \cup V_2 = V$
- For all edges $\{u, v\} \in E \rightarrow u \in V_1$ and $v \in V_2$, or vice versa



Properties

- Simple undirected graph with:
  - n nodes
  - m edges
  - degree $deg(v)$ of vertex v
- Property 1:

$$\sum_{v \in V} deg(v) \quad = \quad 2m$$

- Property 2: if there are no parallel edges/self-loops, then:

$$m \quad \leq \frac{n(n-1)}{2}$$

Properties (2)

- Simple directed graph with:
  - n nodes
  - m arcs
  - in-degree $indeg(v)$ and out-degree $outdeg(v)$ of vertex v
- Property 1:

$$\sum_{v \in V} indeg(v) \;=\; \sum_{v \in V} outdeg(v) \;=\; m$$

- Property 2:

$$m \;\leq\; n(n-1)$$

Data Structures for Graphs

1. Edge List structure $\longrightarrow$ list of edge and vertex records
2. Adjacency Matrix
3. Adjacency List

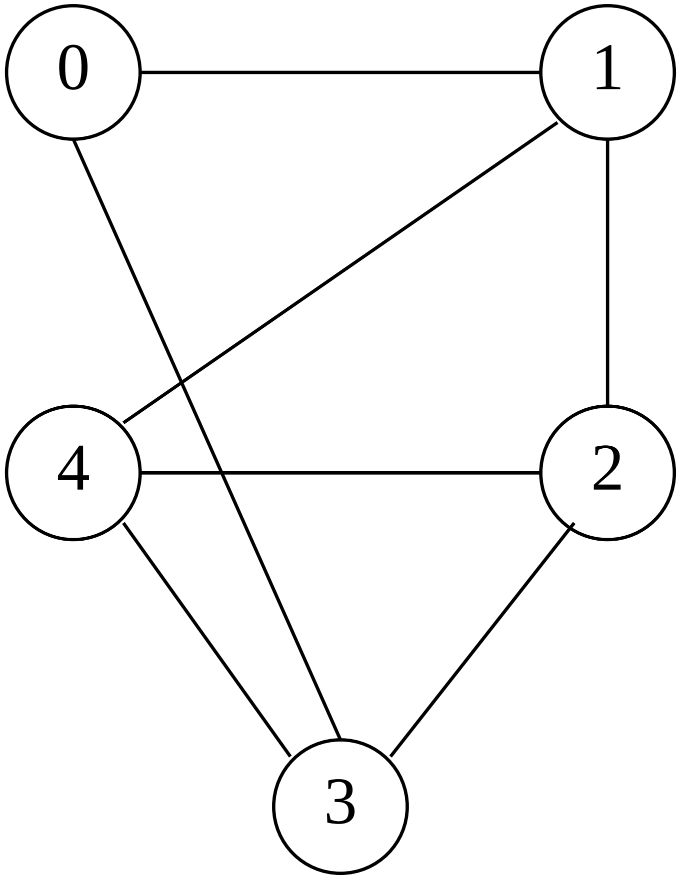
Adjacency Matrix

- Integer index associated w/ vertex $\longrightarrow$ vertices labeled $v_0, \cdots, v_{n-1}$
- A $n \times n$ matrix M defined as:

$$M[i,j] = \begin{cases} 1, & \text{if there is an edge (arc) from } v_i \text{ to } v_j \\ 0, & \text{otherwise} \end{cases}$$

- Space requirements: $O(n^2)$

Adjacency Matrix Example



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

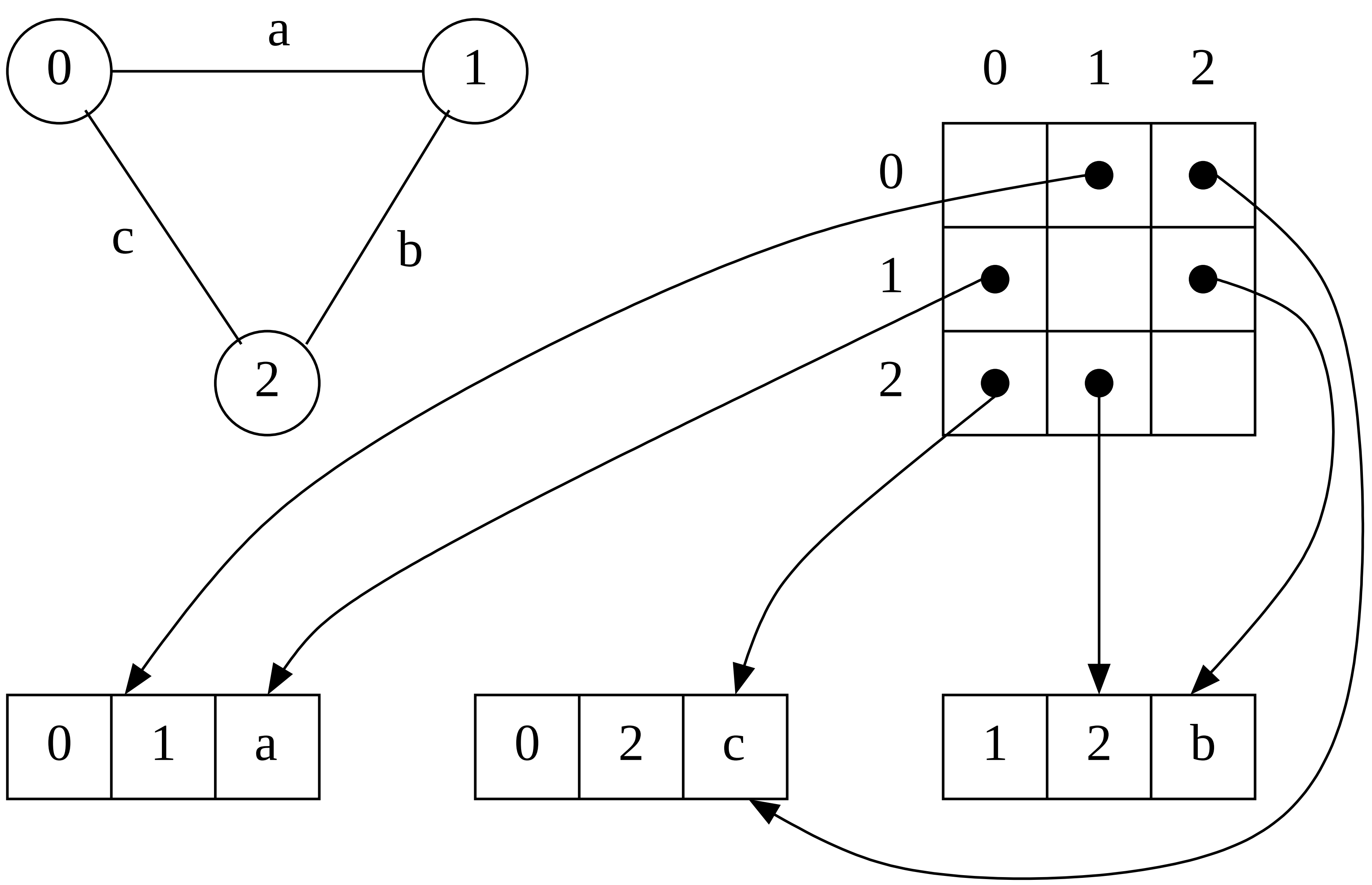
Adjacency Matrix, v.2

- Redefine matrix M as:

$$M[i,j] = \begin{cases} \text{pointer to edge object for } (v_i, v_j), & \text{if edge exists} \\ \texttt{null}, & \text{otherwise} \end{cases}$$

- Edge object holds references to:
  - the two vertices incident on this edge
  - the information (element) associated with the edge

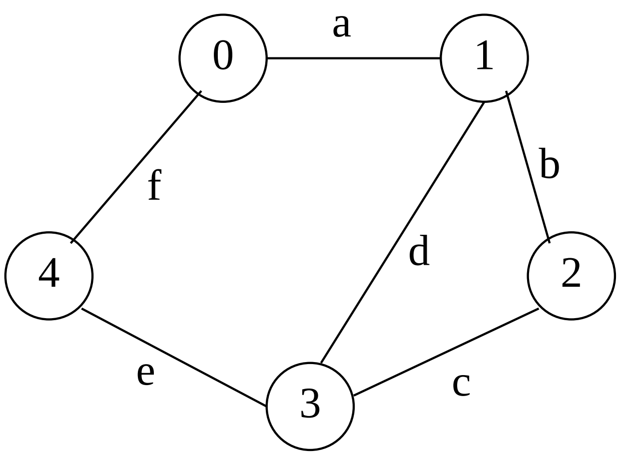
Adjacency Matrix, v.2 Example

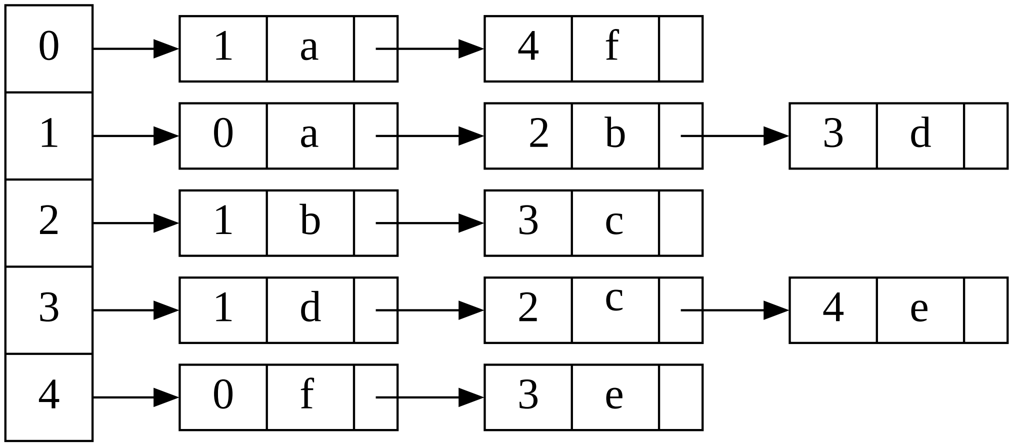


Adjacency List

- Adjacency list for vertex $v \longrightarrow$ list of vertex records $\forall$ neighbors of v
- Vertex record contains:
  - neighbor vertex u of v
  - the information (element) associated with edge (v, u)
  - pointer to next vertex record in the adjacency list
- Space requirements: $O(n + m)$
- More efficient for sparse graphs $\longrightarrow m \approx O(n)$

Adjacency List Example





Comparison: Assumptions

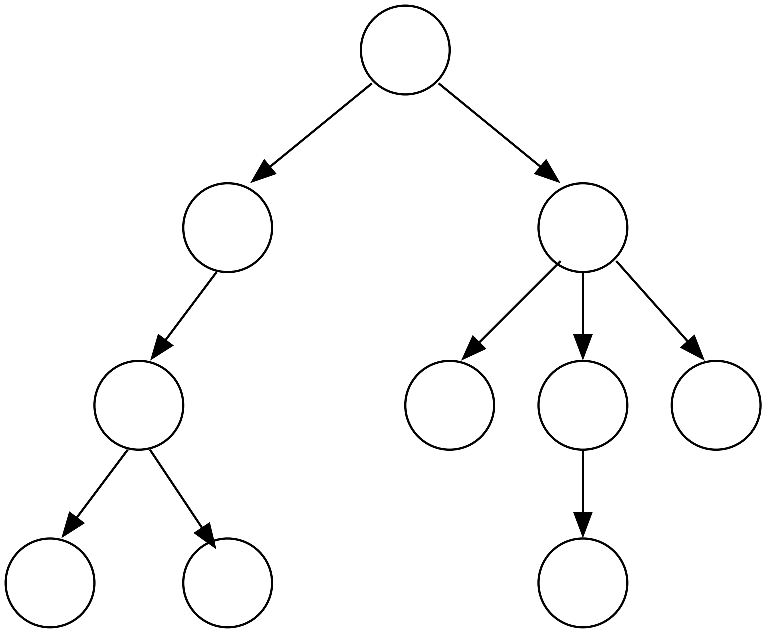
- n vertices
- m edges
- No parallel edges
- No self-loops

Asymptotic Performance

	Adjacency List	Adjacency Matrix
Space	$O(n+m)$	$O(n^2)$
incidentEdges(v)	$O(deg(v))$	$O(n)$
areAdjacent(v,w)	$O(\min\{deg(v), deg(w)\})$	$O(1)$
insertVertex(v)	$O(1)$	$O(n^2)$
insertEdge((v,w))	$O(1)$	$O(1)$
removeVertex(v)	$O(deg(v))$	$O(n^2)$
removeEdge((v,w))	$O(\min\{deg(v), deg(w)\})$	$O(1)$

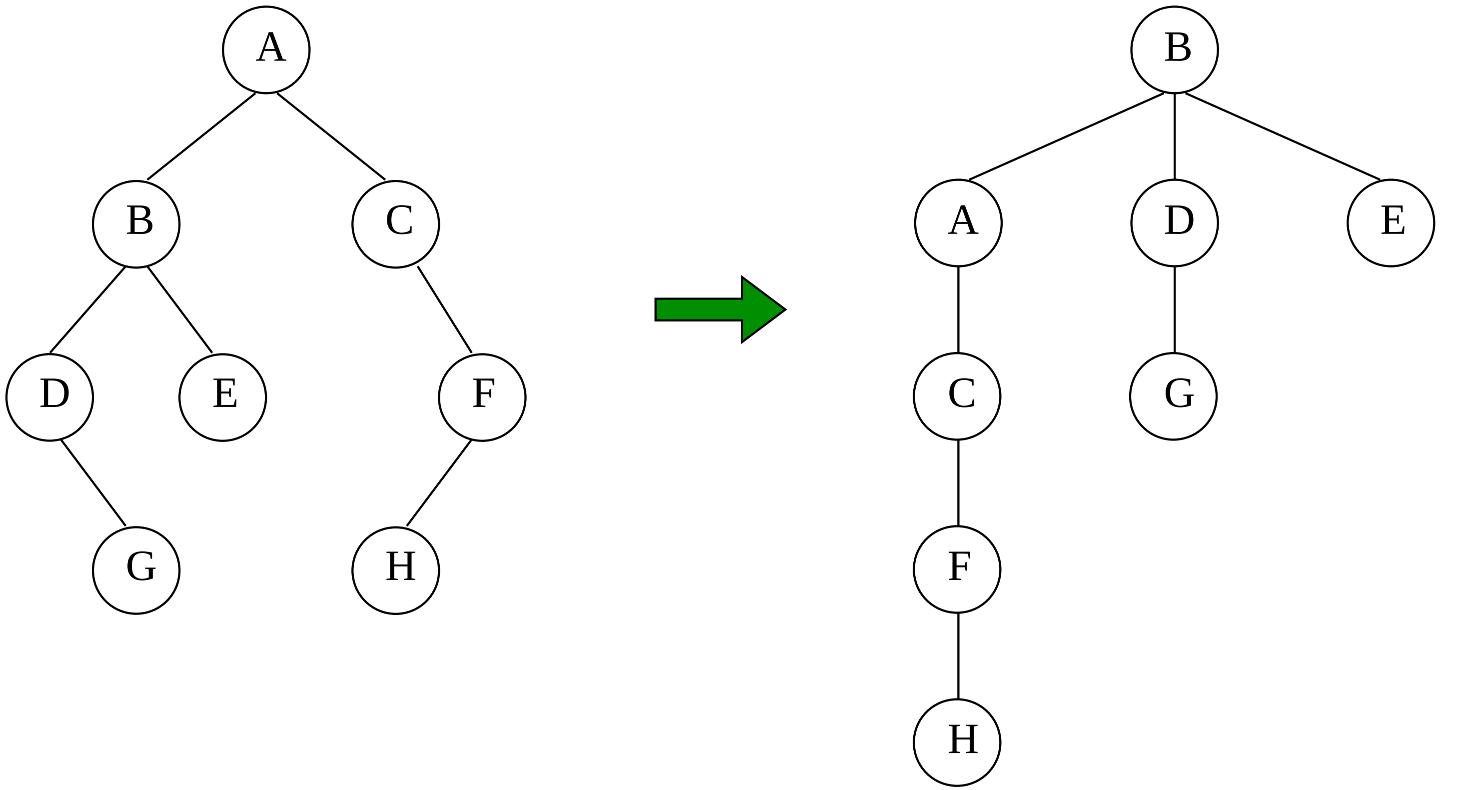
Trees as Graphs

- **Definition:** a directed tree is a directed acyclic graph (DAG) such that:
  - there is exactly one vertex, the root, which no arcs enter
  - every vertex except the root has exactly one entering arc
  - there is a unique path from the root to each vertex



Trees as Graphs (2)

- **Definition:** an undirected graph G is a tree if and only if:
  - $\forall u, v \in G$, there exists a unique simple path from u to v
- Undirected trees $\longrightarrow$ "root" not important $\longrightarrow$ any vertex can be root



Graph Properties

- Undirected graph $G = (V, E)$ with
  - $n = |V|$ vertices
  - $m = |E|$ edges
- Properties:
  1. G is connected $\Rightarrow m \geq n - 1$
  2. G is acyclic $\Rightarrow m \leq n - 1$
- Proof:
  1. Graph w/ n vertices, no edges $\longrightarrow$ add edges to make it connected
  2. By induction on number k of vertices of the graph

Checking for Tree Property

- A graph G with n vertices is a tree if any of these conditions are true:
  1. G is connected and acyclic
  2. G is connected and has exactly $n-1$ edges
  3. G is acyclic and has exactly $n-1$ edges
  4. G is connected, but deleting any edge disconnects the graph
  5. G is not complete, and adding any edge creates exactly one cycle that contains the new edge