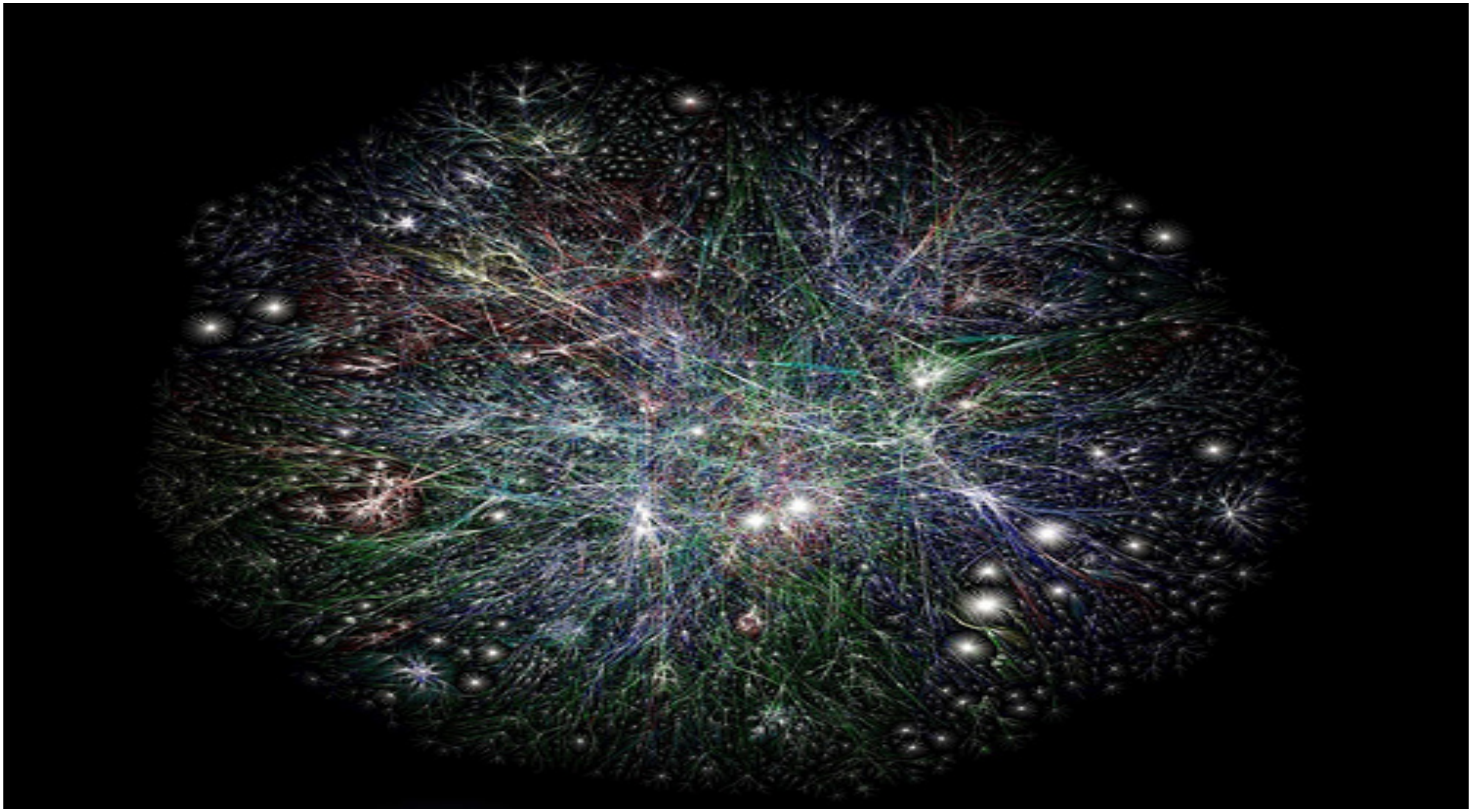
CS261 – Data Structures

Graphs

Goals

- Introduction and Motivation
- Representations

Why do we care about graphs?



Many Applications

- Social Networks – Facebook
- Video Games - Motion Graphs
- Machine Learning/AI
- Delivery Networks/Scheduling – UPS?
- Computer Vision – Image Segmentation
- …

Graphs

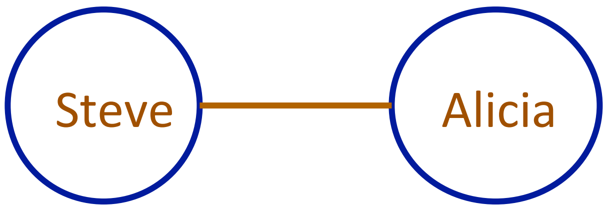
- Graphs represent relationships or connections
- Superset of trees (i.e., a tree is a restricted form of a graph):
  - A graph represents general relationships:
    - Each node may have many predecessors
    - There may be multiple paths (or no path) from one node to another
    - Can have cycles or loops
    - Examples: airline flight connections, friends, algorithmic flow, etc.
  - A tree has more restrictive relationships and topology:
    - Each node has a single predecessor—its parent
    - There is a single, unique path from the root to any node
    - No cycles
    - Example: less than or greater than in a binary search tree

Graphs: Vertices and Edges

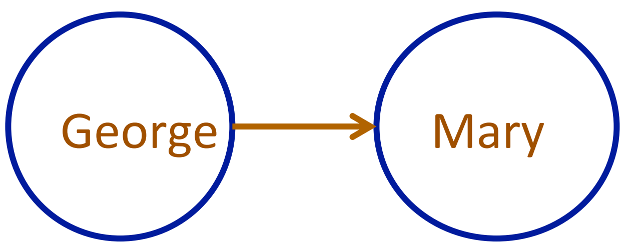
- A graph is composed of *vertices* and *edges*
- Vertices (also called *nodes*):
  - Represent objects, states (i.e., conditions or configurations), positions, or simply just place holders
  - Set $\{v_1, v_2, \ldots, v_n\}$: each vertex is unique → no two vertices represent the same object/state
- Edges (also called *arcs*):
  - An edge (v_i, v_j) between two vertices indicates that they are directly related, connected, etc.
  - Can be either *directed* or *undirected*
  - Can be either *weighted* (or labeled) or *unweighted*
  - If there is an edge from v_i to v_j, then v_j is a *neighbor* of v_i (if the edge is undirected then v_i and v_j are neighbors or each other)

Graphs: Directed and Undirected

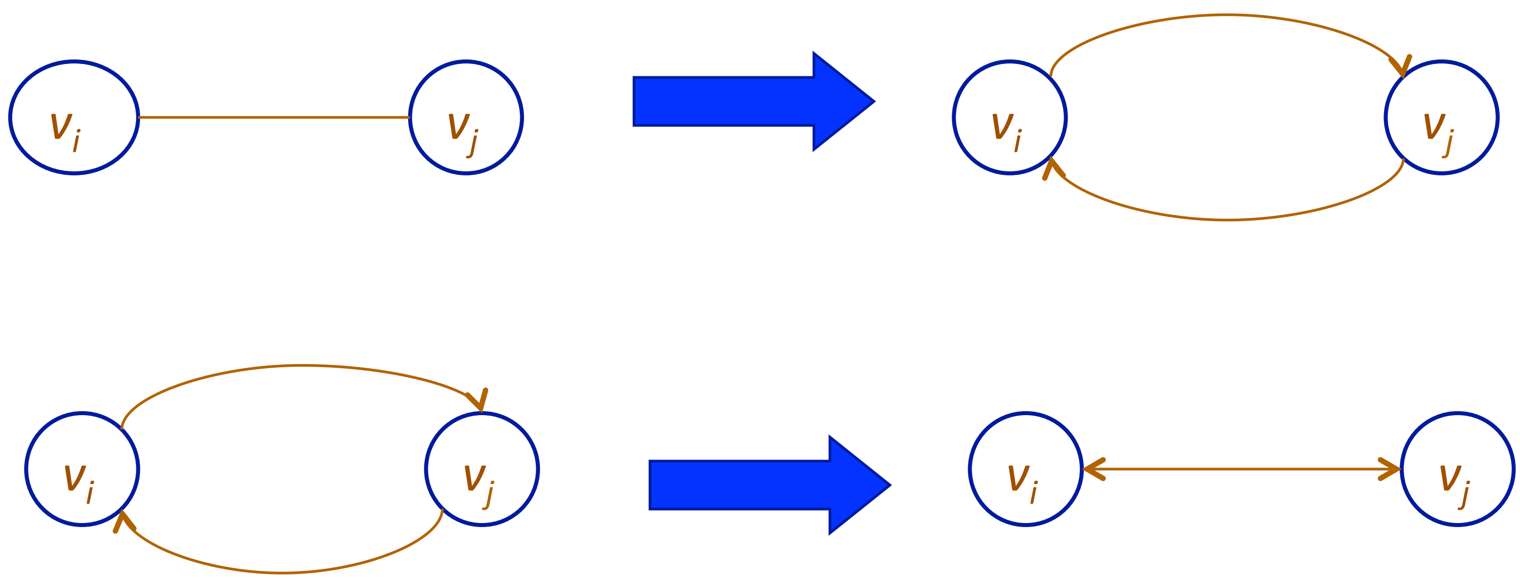
– Example: **friends** – Steve and Alicia are *friends*



– Example: **like** – George *admires* Mary

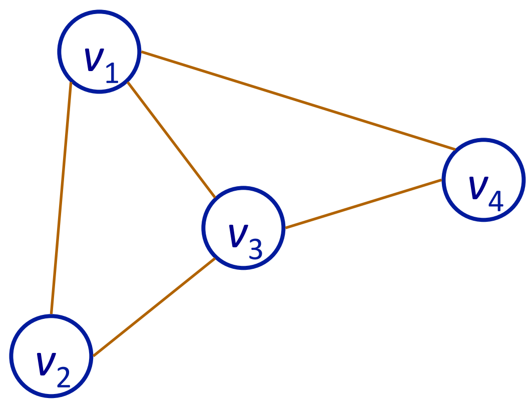
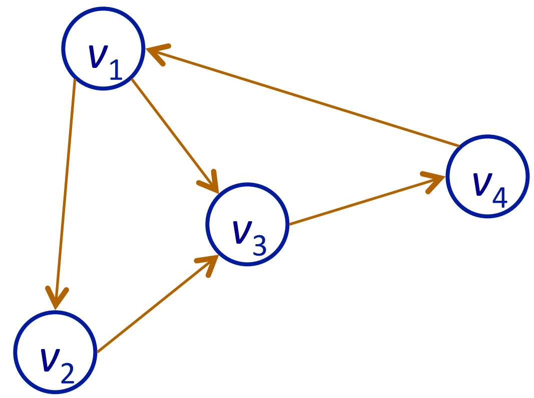
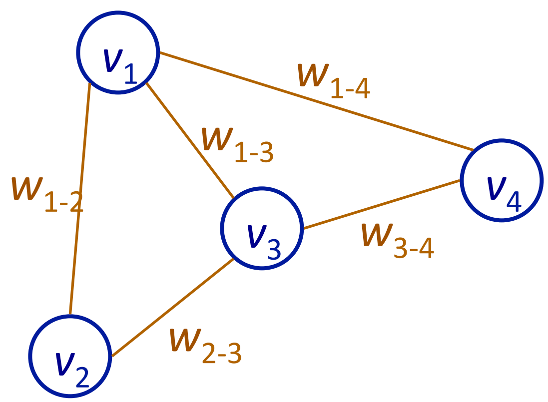
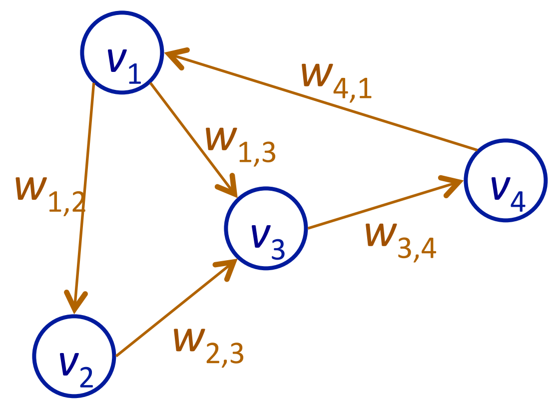


Graphs: Directed and Undirected (cont.)



e

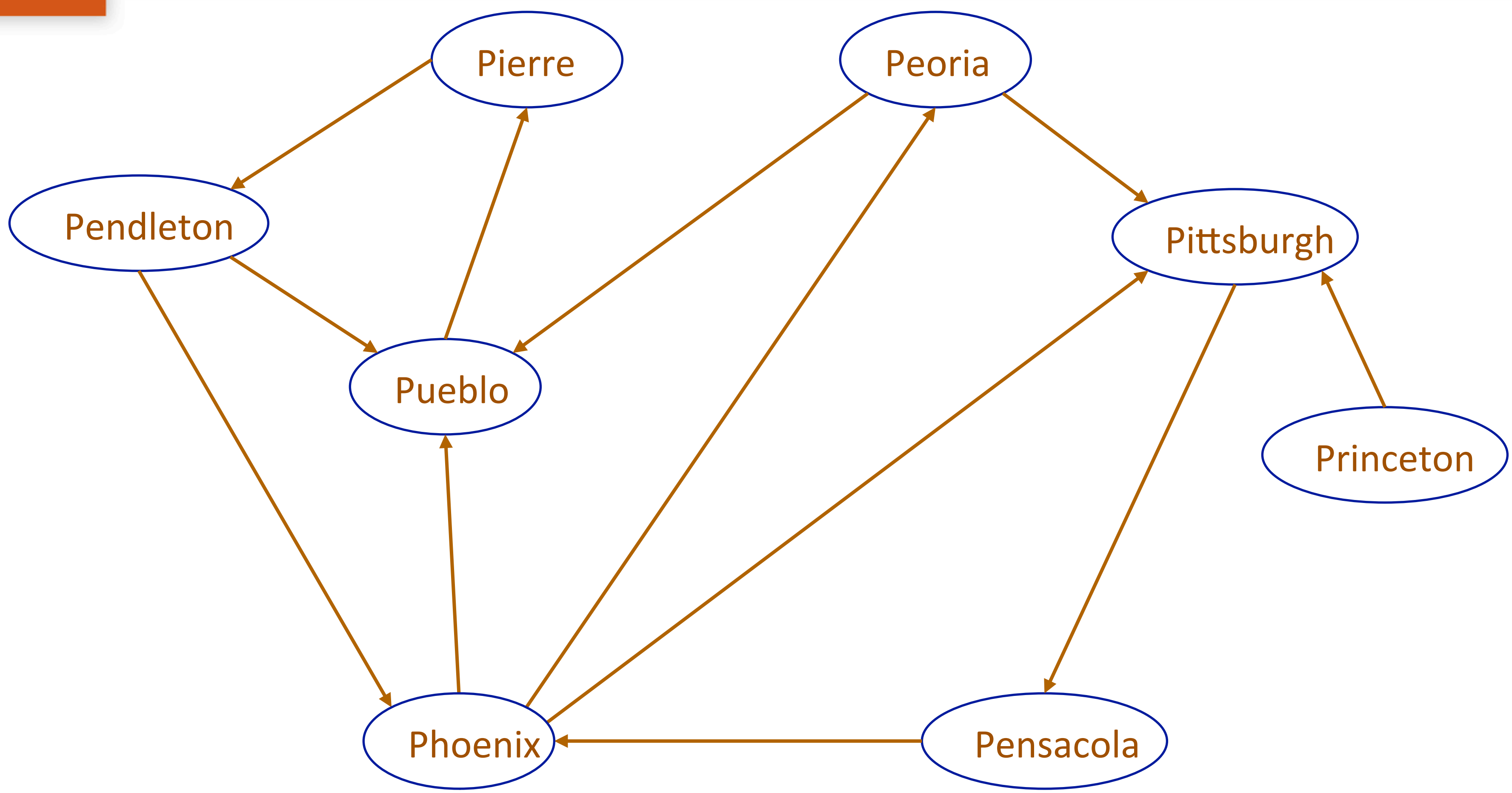
Graphs: Types of Edges

<table>
<tr><th></th><th>Undirected</th><th>Directed</th></tr>
<tr><td>Unweighted</td><td>
</td><td>
</td></tr>
<tr><td>Weighted</td><td>
</td><td>
</td></tr>
</table>

Graphs: What kinds of questions can we ask?

- Is A reachable from B?
- What nodes are reachable from A?
- What's the shortest path from A to B?
- Is A in the graph?
- etc…

Graphs: Example



Graphs: Representations (unweighted)

	Pendleton	Pensacold	Peoria	Phoenix	Pierre	Pittsburgh	Princeton	Pueblo
City	0	1	2	3	4	5	6	7
0: Pendleton	?	0	0	1	0	0	0	1
1: Pensacola	0	?	0	1	0	0	0	0
2: Peoria	0	0	?	0	0	1	0	1
3: Phoenix	0	0	1	?	0	1	0	1
4: Pierre	1	0	0	0	?	0	0	0
5: Pittsburgh	0	1	0	0	0	?	0	0
6: Princeton	0	0	0	0	0	1	?	0
7: Pueblo	0	0	0	0	1	0	0	?

Adjacency Matrix
$O(v^2)$ space

By convention, a vertex is usually connected to itself
(though, this is not always the case)

Stores only the edges → more space efficient for sparse graph: $O(V + E)$

where sparse means relatively few edges

Edge List
O(v+e) space

Pendleton:	{Pueblo, Phoenix}
Pensacola:	{Phoenix}
Peoria:	{Pueblo, Pittsburgh}
Phoenix:	{Pueblo, Peoria, Pittsburgh}
Pierre:	{Pendleton}
Pittsburgh:	{Pensacola}
Princeton:	{Pittsburgh}
Pueblo:	{Pierre}

Graphs: Representations (weighted)

City	0	1	2	3	4	5	6	7
0: Pendleton	?	0	0	13	0	0	0	22
1: Pensacola	0	?	0	1	0	0	0	0
2: Peoria	0	0	?	0	0	8	0	13
3: Phoenix	0	0	43	?	0	16	0	90
4: Pierre	7	0	0	0	?	0	0	0
5: Pittsburgh	0	10	0	0	0	?	0	0
6: Princeton	0	0	0	0	0	5	?	0
7: Pueblo	0	0	0	0	22	0	0	?

Adjacency Matrix
$O(v^2)$ space

Edge List
$O(v+e)$ space

Pendleton:	{(Pueblo,22), (Phoenix,13)}
Pensacola:	{(Phoenix,1)}
Peoria:	{(Pueblo,13), (Pittsburgh,8)}
Phoenix:	{(Pueblo,90), (Peoria,43), (Pittsburgh,16)}
Pierre:	{(Pendleton,7)}
Pittsburgh:	{(Pensacola,10)}
Princeton:	{(Pittsburgh,5)}
Pueblo:	{(Pierre,22)}

Your Turn

- Complete Worksheet #40 Graph Representations