Announcements

- Homework 2 is released, due next Tuesday at 3:30pm
- The grading for Programming Project 1 will be delayed
- The office hour schedule for the rest of the semester has been posted on Piazza and the course website

Topic 3: Graphs

By Professor Hudson Lynam

Graph Definition

- Definition: *Graph*

- *A graph G=(V,E) where V is a finite non-empty set of vertices (a.k.a. nodes) and E is a binary relation on V.*

- Note that definitions of "graph" vary: this is a common definition we'll be using for this class

Undirected versus Directed Graphs

- Undirected:
  - The pairs of vertices in E are **unordered**. In other words, each edge is a set, like {A,B}.
  - When we say "G is a graph," the default assumption is that G is undirected
- Directed
  - Each edge has an order, like a tuple (A,B).
 - Because directed graphs are common, we usually abbreviate to "digraph."
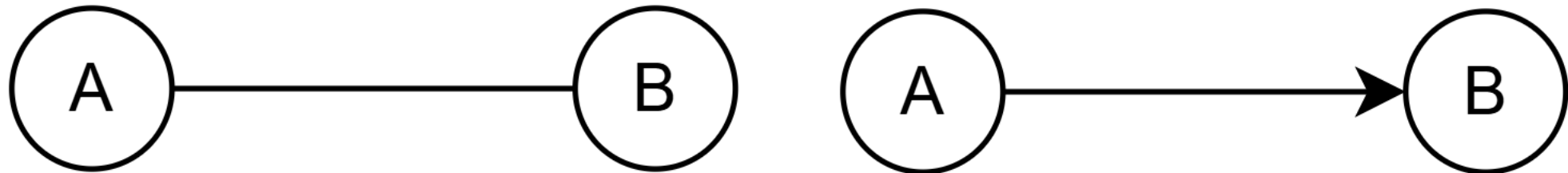
Graphs Example

- What would V be for this graph?
- V={A, B, C, D}
- E={{A,C}, {A,B}, {A,D}, {B, D}}
- What about V and E for this digraph?
- V={1, 3, 5, 7}
- E={(1,3), (3,1), (5,1), (7, 7), (7, 5)}
- Note: Don't use bidirectional arrows for undirected graphs, and always use arrows to show two edges between vertices in a digraph
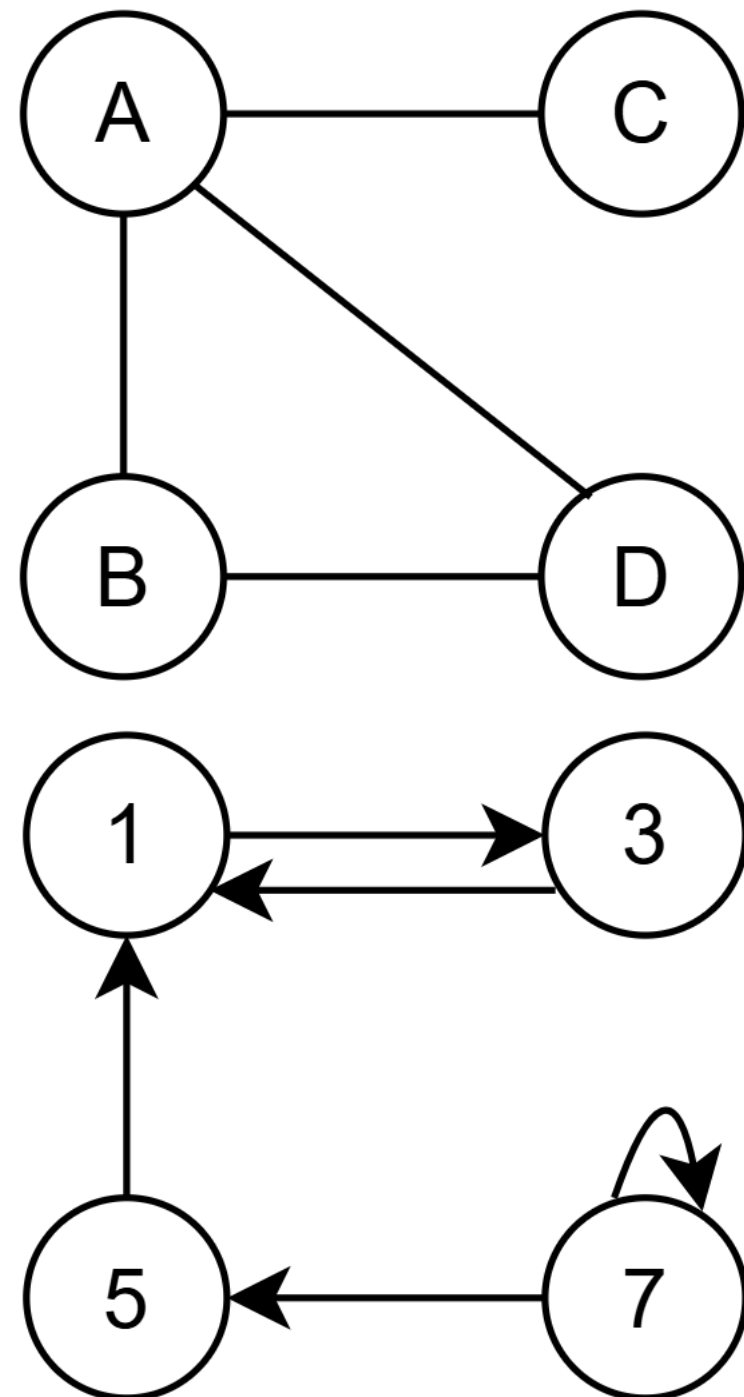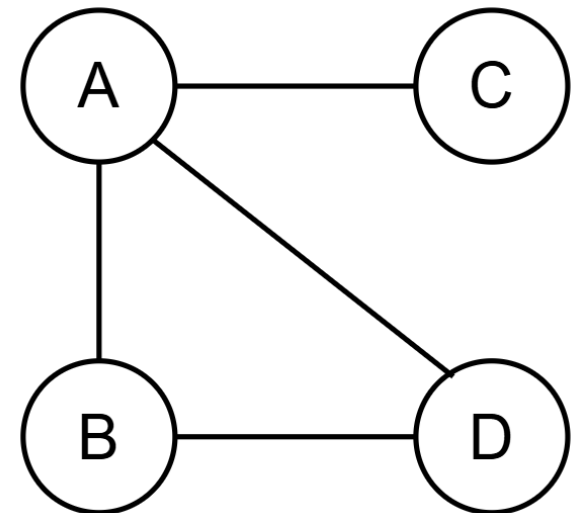
Graph Terminology

- Definition: *Adjacent Vertices*

- *2 vertices are adjacent when an edge exists between them*

- In our example graph, C and D are not adjacent, B and C are not adjacent, every other pair of vertices are adjacent

- Definition: *Path (a.k.a. Walk)*

- *A path is a sequence of adjacent vertices*

- Example: C-A-B-A-D-B.

- A **simple** path's vertices appear exactly once. The above example is *not* simple. What would be a simple path here?
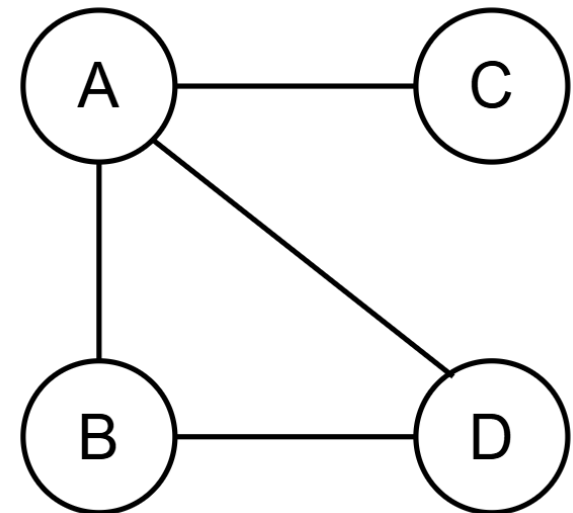
Graph Terminology

- Definition: *Cycle*
- *A cycle is a path of length ≥ 1 that begins and ends with the same vertex*

- In our example graph, A-C-A is a cycle. D-B-A-C-A-D is a cycle.

- A cycle is **<u>simple</u>** if all the vertices in the path only occur *once* (except for the first and last vertices)

- Notes: "the cycle ABD" means A-B-D-**A**

- Common assumption: simple cycles in undirected graphs must have length ≥ 3

Graph Terminology

- Definition: *Incident Edges*
- *In any graph, an edge is incident on both participating vertices*
- *In digraphs, an edge is **incident from** the source and **incident to** the destination vertex*

- Definition: *Degree*
- *The degree of a vertex is its # of incident edges*

- Note: a self-loop adds to a vertex's degree
- What is the degree of 1? In versus out?
- In-degree of 1: 2, out-degree of 1: 1
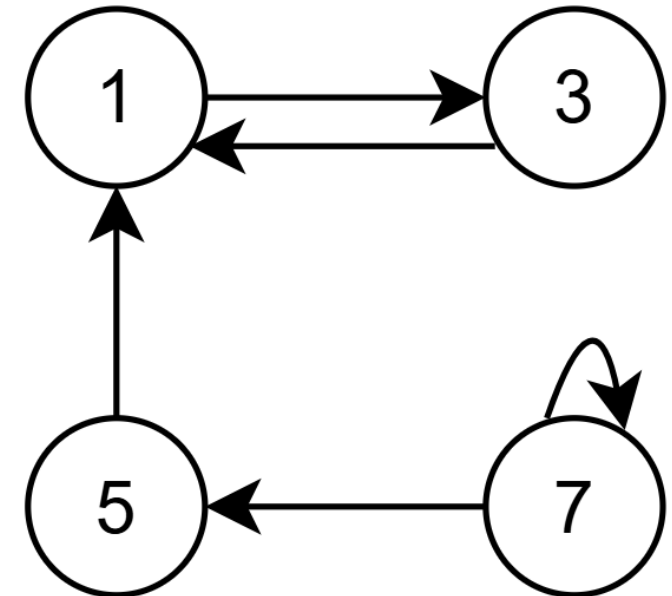
Graph Representations

- How can we store all of a graph's vertices and edges? Three common representations:

1. Edge Lists

2. Adjacency Matrices

3. Adjacency Lists

- Which should we choose? Depends on the operations we want to be efficient.

Edge Lists

- An edge list is a collection of edges within a graph. Example:

- I is an isolate in this graph. Obviously, not an ideal representation for graphs with isolates

- Note: In a digraph, just allow 1st vertex in each edge to the source or "from" vertex
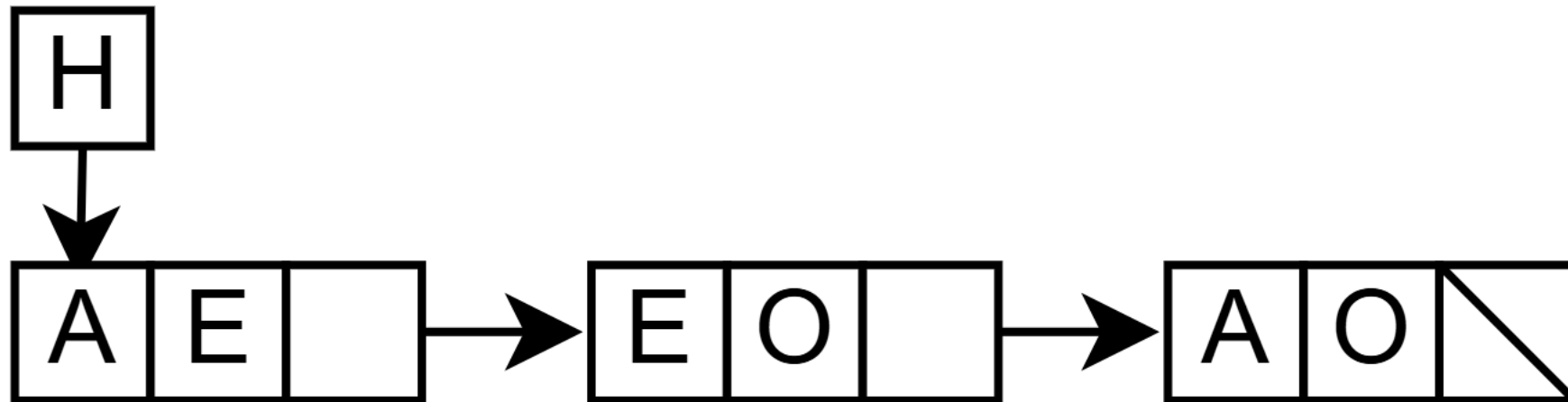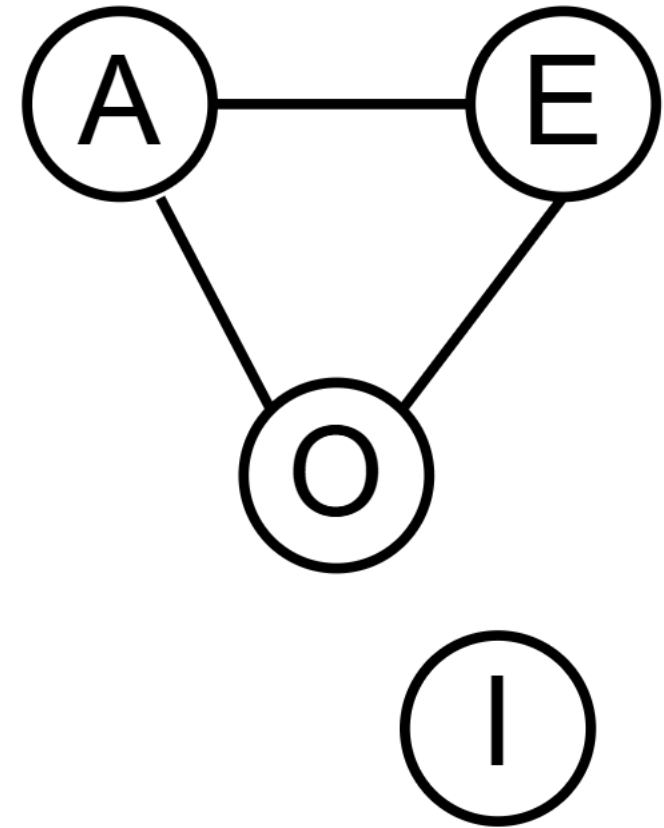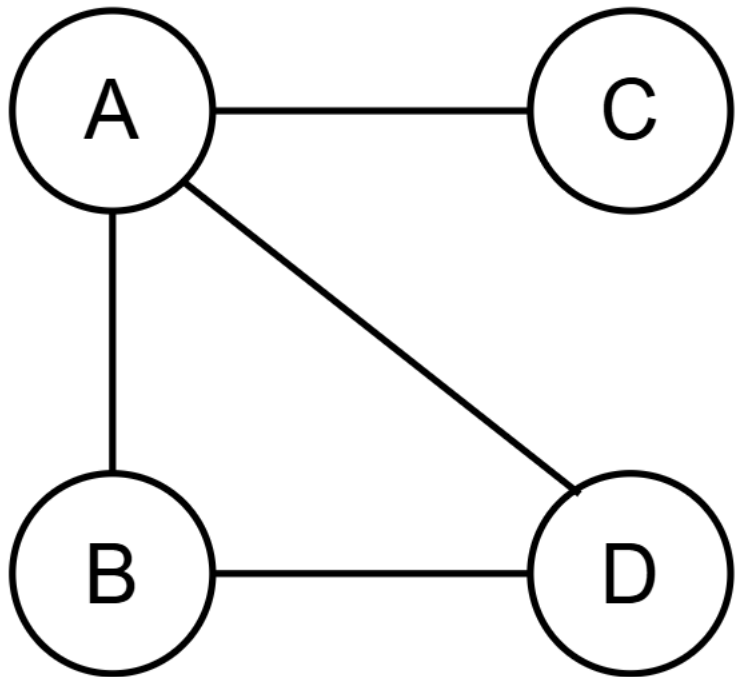
Adjacency Matrices

- Adjacency Matrices are a good choice when the quantity of vertices is bounded and the graph has a relatively large quantity of edges.

- The **adjacency matrix** of a graph G=(V, E) is a |V| x |V| matrix M in which element M[i][j] = 1 iff $\{v_i, v_j\}$ or (v_i, v_j) ∈ E. Otherwise, M[i][j] = 0.

Adjacency Matrices

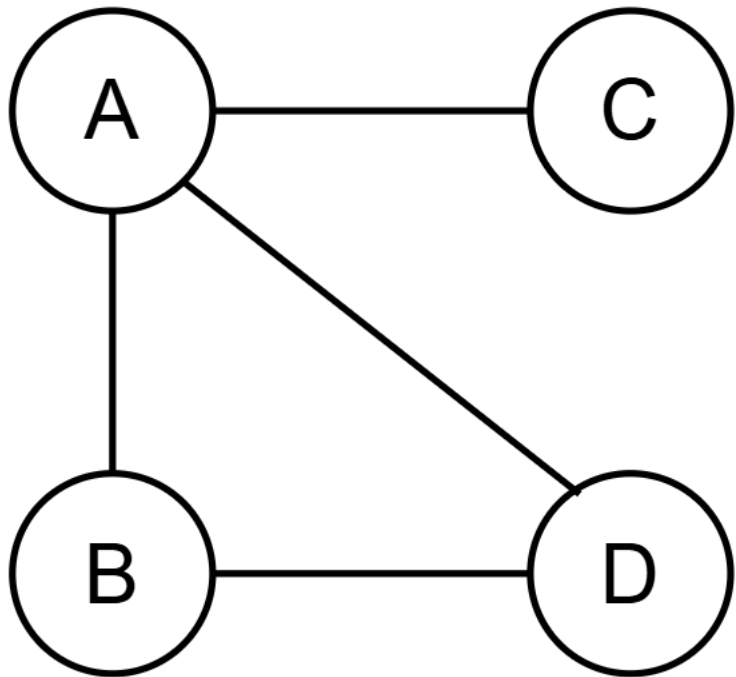- Let's take a look at our example graph:

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

Adjacency Matrices

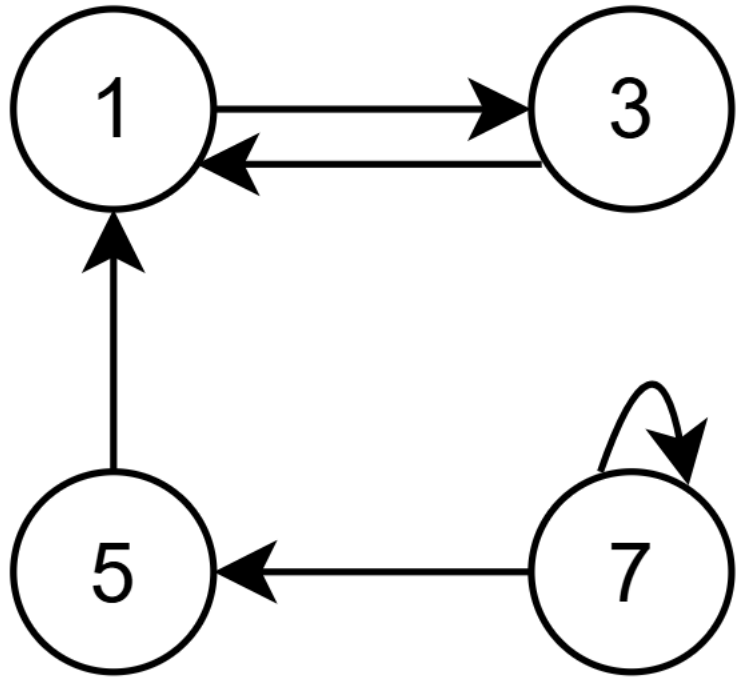- Did we even need the full matrix? Could we represent the edges more efficiently? How?

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

- Note: this is a lower left triangular matrix

Adjacency Matrices

- Try to make an adjacency matrix for this digraph. Cols are To, Rows are From

	1	3	5	7
1	0	1	0	0
3	1	0	0	0
5	1	0	0	0
7	0	0	1	1

Adjacency Matrices

- Adjacency matrices make many operations easy. For example…
- How would we find the degree of a vertex in an AM?
- Count the row or col for the vertex if it's a graph, count the row for out-degree in a digraph, or col for the in-degree
- Which edges are incident to a given vertex?
- Same solution as finding degree!
- But there's a downside:
- We need $O(|V|^2)$ space to store an AM. Which is why AM is a good choice when the number of vertices is low

Adjacency Lists

- The **<u>adjacency list</u>** of a graph G=(V, E) is an array of |V| lists of vertices, with each list holding the vertices to which the vertex is adjacent (for a graph) or the vertices incident from a vertex (for a digraph).

Adjacency Lists

- Let's take a look at our example graph: