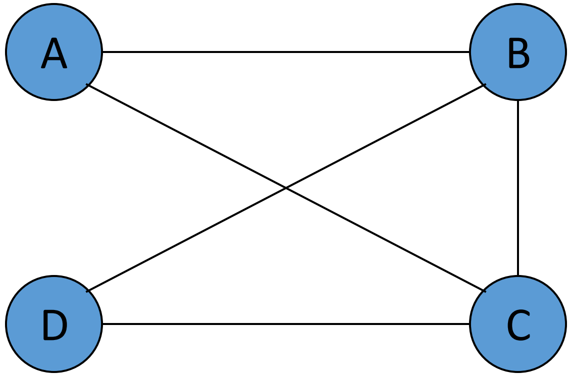
Introduction to Graphs

- Graph definition / applications
- Graph terminology
    - Undirected graphs
    - Directed graphs
    - Networks
- Implementing graphs
- Reading: L&C 15.1–15.3, 15.5
- Course Evaluation

Graph Definition/Applications

- A *graph* is a non-linear structure w/o any root or parent-child relationships between nodes



- Graphs can be used to represent:
  - Communication or transportation networks
  - State diagrams for finite automata
  - Prerequisites in a course curriculum

Problems Solved Using Graphs

- Giving travel directions (Turn right at the gas station, go though two traffic lights, turn left)
- Determining if the graph is connected or not (“Come to think of it now, there ain’t no way to get there from here” – punchline of a joke)
- Finding the shortest path across a network (Find me the lowest airfare from Boston to Hawaii)

Graph Terminology

- A *vertex* is a node in a graph
- Vertices are named or have a label, e.g. A
- An *edge* is a connection between vertices
- Edges are identified by the pair of vertices that they connect, e.g. (A, B)

Undirected Graphs

- In an *undirected graph*, the edges have no direction associated with them
- A connection between vertices A and B can be traversed in either direction
- Naming an edge (A, B) is the same as (B, A)
- Two vertices are *adjacent* if there is an edge connecting them
- Adjacent vertices are also called *neighbors*
- An edge connecting a vertex to itself is called a *self-loop* or *sling*

Undirected graphs

- An undirected graph is considered *complete* if it has the maximum number of edges for its number of vertices
- A complete graph is also sometimes called a *full mesh*
- For n vertices, the number of edges in a complete graph is equal to: n (n - 1) / 2

Undirected Graphs

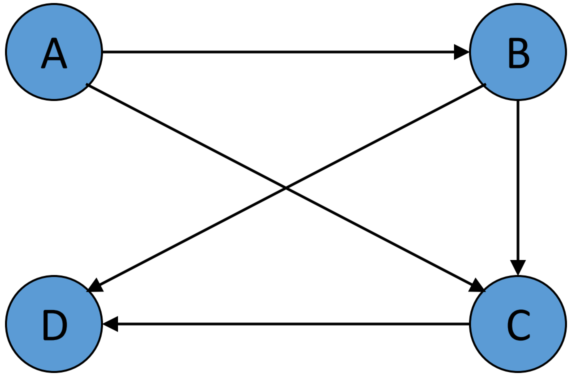
- A *path* is a sequence of edges between two vertices
- The *path length* is the number of edges
- An undirected graph is *connected* if for any two vertices in the graph, there is a path between them
- If an edge is removed from a graph so that there is no longer a path between any two vertices, we say the graph is *partitioned*

Graph Terminology

- A *cycle* is a path with the same first and last vertex and with no repeated edges
- A graph with no cycles is called *acyclic*
- A undirected tree is a sub-class of a graph
- A undirected tree is a connected, acyclic, undirected graph with one element designated as the root

Directed Graphs

- A *directed graph* also called a *digraph* is a graph where the edges are ordered rather than unordered pairs of vertices
- Edge (A, B) is no longer the same as (B, A)



Networks

- A *network* or *weighted graph* is a graph with a cost or weight associated with each edge
- The weight may reflect some physical cost of using each edge such as the cost for making a call or the fare for travel between two cities
- The *weight* of a path or the *path weight* is the sum of the weights of the edges in the path
- Networks may be undirected or directed
- The weight for any missing edge is infinity

Implementing Graphs

- Adjacency Lists
- For each node, we maintain a collection of references to the other nodes with which this node shares an edge
- For a directed graph, the collection contains references to the other nodes that can be reached from this node (reverse edge may not be stored in node at other end)
- For a network, the collection contains the other node and a weight value for the edge

Implementing Graphs

- Adjacency Matrices
- Use a two-dimensional array (with weights to represent a network if required)

Undirected Graph

	A	B	C	D
A	F	T	T	F
B	T	F	T	T
C	T	T	F	F
D	F	T	F	F

Undirected Network

	A	B	C	D
A	0	5	3	Infinity
B	5	0	2	6
C	3	2	0	Infinity
D	Infinity	6	Infinity	0

Final Exam

- Final Exam
  - See posted day, time, and location
  - It will be 1-1/2 hours – not 3 hours
- Watch the website for any late breaking announcements!
- See the posted practice final exam

Course Evaluations

- Please fill out the course evaluation forms
- Give to assigned student for turn-in
- Thank you