Activity #6: Graphs

Recorder's Report

Manager:	Reader:
Recorder:	Driver:
Date:	Score: Satisfactory / Not Satisfactory

Record your team's answers to the key questions (marked with 🔑) below.

(a) Model 1, Question #20

(b) Model 2, Question #21

(c) Model 2, Question #22

(d) Model 3, Question #23

(e) Model 3, Question #32

Activity #6: Graphs

Many items in the world are connected, such as computers on a network connected by wires, cities connected by roads, or people connected by friendship. A graph is a data structure for representing connections among items, and consists of vertices connected by edges.

Content Learning Objectives

After completing this activity, students should be able to:

- Explain the components of a graph (vetices and edges)
- Describe the difference between undirected and directed graphs
- Describe the cycle in a graph
- Explain how to represent a graph as a data structure

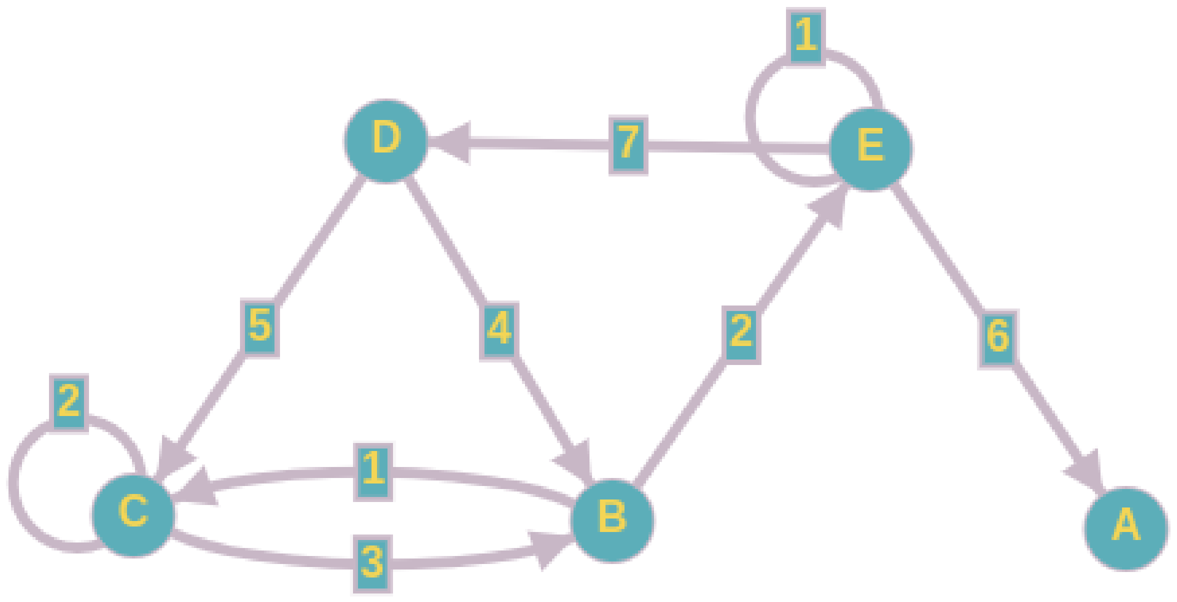
Process Skill Goals

During the activity, students should make progress toward:

- Find cycles in a graph
- Translate a graph into an adjacency list or adjacency matrix

Model 1 Model 1: Graphs

A graph in computer science consists of vertices (also called nodes) and edges (also called links). Nodes are generally depicted as circles and labeled with a unique identifier. Edges are generally depicted as lines for undirected graphs and arrows for directed graphs. See how much you can glean from the graph below.



1. What are the vertices?
2. Is this graph directed or undirected? directed undirected
3. Is there an edge from B to D? yes no
4. Is there an edge from D to C? yes no
5. What is the cost of the edge from E to A?

Graphs with edges that have costs are called **weighted** graphs.

6. What is the lowest cost path from D to A?

The **degree** of a vertex is the number of edges that are incident (touch) the vertex. For directed graphs, each node has an **"in degree"** and an **"out degree"**. The in degree is the number of edges that point to the vertex. The out degree is the number of edges that point away from the vertex. The in degree and out degree values do not need to match for a vertex. However, the total degree for a vertex of a directed graph is the in degree plus the out degree.

7. What is the out degree for E?

8. What is the in degree for E?

9. What is the out degree for B?

10. What is the in degree for B?

A **path** through a graph is a sequence of vertices where each successive pair of vertices has an edge in the graph.

11. Name two different paths, as a sequence of vertices, to get from B to C:

- path 1:

- path 2:

A **connected** graph has the property that every vertex is connected via a path to every other vertex in the graph.

12. Is the graph above connected? yes no

A **cyclic** graph has the property that it contains one or more cycles. A **cycle** is a path that starts and ends with the same vertex. An **acyclic** graph has no cycles. A directed graph that is acyclic is often called a **DAG** (directed acyclic graph).

13. Is the graph above cyclic? yes no

We can think of a graph as having $|V|$ vertices and $|E|$ edges where V is the set of vertices and E is the set of edges. The vertical bars mean "size of" in mathematical notation. So, $|V|$ is the size of the vertex set.

14. What is $|V|$ for the graph above?

15. What is $|E|$ for the graph above?

Two vertices are **adjacent** if they are connected by an edge. In an undirected graph, you can think of adjacent vertices as neighbors. In an undirected graph, if x is adjacent to y, then y is adjacent to x. In a directed graph, a vertex x is adjacent to vertex y if there is an edge from y to x.

16. In the graph above, is e adjacent to b? yes no

17. In the graph above, is d adjacent to b? yes no

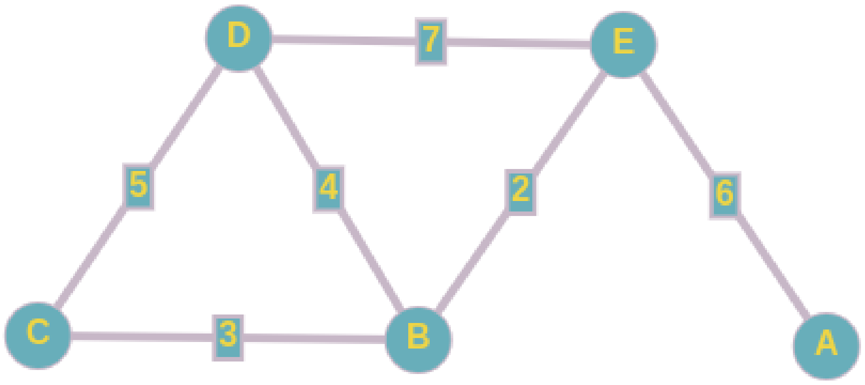
We have names for vertices in relationship to other vertices in a directed graph. A **successor** of a vertex v is any node n where there is a path from v to n. A **predecessor** of vertex v is any node n where there is a path from n to v.

18. Is a a predecessor of d? yes no

19. Is a a successor of d? yes no

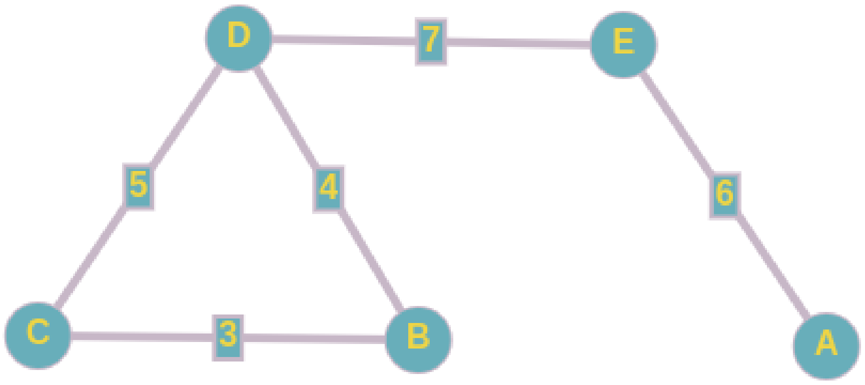
Undirected Graphs

Now, let's look at an undirected graph. An undirected graph may or may not be weighted (costs for edges). Recall that an undirected graph has links for edges instead of arrows.



20. What is different about this undirected graph versus a directed graph?

Note that the edges are links, so the cost between node b and c is the same as the cost between c and b. Here, the cost between b and c (and c and b) is 3. A path can be traversed in either direction of the link. So, in an undirected graph, if there is a path from vertex x to vertex y, there must also be a path from vertex y to vertex x (following the same set of vertices in reverse order). Because there are no more directed arrows, it no longer makes sense to think about cycles, successors, and predecessors in an undirected graph. An undirected graph could be connected, as follows:



Model 2 Modeling with Graphs

Now that we have looked at graphs and the terminology, let's now focus on how graphs are useful for computation. Graphs are the basis for modeling relationships among entities. Here are some examples:

- The global internetwork of computers
  - Vertices are routers and computers; Edges are links between routers/computers
- Social networks
  - Vertices are people; Edges represent the "friend" relationship
- State diagrams
  - Vertices represent current state of computer; Edges represent transition to new states
  - Often the model used by event-driven programming
- Rubik's cube
  - Vertices represent current state of cube; Edges represent one twist to new configuration of the Rubik's cube
- Transportation networks
  - Vertices are intersections; Edges are roads/highways
- Airline flights
  - Vertices represent cities/airports; Edges are flights between the cities

21. What real life scenario (not from above list) could you model with a directed graph?

22. What real life scenario (not from above list) could you model with an undirected graph?

At this point, hopefully, your group realizes the importance of graphs in computing. Note that a graph can be thought of as a tree that can have cycles. A tree is also always a graph. A graph is not always a tree.

Model 3 Representing Graphs

Now, to the implementation part of graphs. There are two standard ways to represent a graph in code. One representation uses an adjacency matrix. The other representation uses adjacency lists.

Adjacency Matrix

An adjacency matrix M is a $|V|x|V|$ (2D array) of integers. You can number the vertices 0 to $|V| - 1$ (or have a mapping of vertex names to numbers 0 to $|V| - 1$). If there is no edge from vertex x and vertex y, then $M[x][y] = 0$ (note that book uses infinity; either value works as long as the implementation knows how to treat the value representing "no edge"). If there is an edge from vertex x to y with cost C, then $M[x][y] = C$.

If the graph is directed, $M[x][y]$ does not need to equal $M[y][x]$. If the graph is undirected, $M[x][y] = M[y][x]$. If the graph edges have no costs, then M contains just 0's and 1's (and can store bits instead of ints).

Adjacency List

An adjacency list L is a list (linked list or array) of vertices. Each vertex stores a list of the vertices that are adjacent to it. If the edges have costs, then the adjacent vertices are stored as (V, C) pairs where V is the vertex and C is the cost.

Here is the adjacency matrix for the first graph in this handout:

	A	B	C	D	E
a	0	0	0	0	0
b	0	0	1	0	2
c	0	3	2	0	0
d	0	4	5	0	0
e	6	0	0	7	1

Here is the adjacency list for the first graph in this handout:

a $- > \{\}$

b $- > \{(c,1),(e,2)\}$

c $- > \{(b,3),(c,2)\}$

d $- > \{(b,4),(c,5)\}$

e $- > \{(a,6),(d,7),(e,1)\}$

23. Here is the adjacency matrix for an undirected graph. Draw the circles and edges (links) below.

$$\begin{array}{c|cccc} & a & b & c & d \\ a & 0 & 2 & 1 & 0 \\ b & 2 & 0 & 3 & 1 \\ c & 1 & 3 & 0 & 4 \\ d & 0 & 1 & 4 & 0 \end{array}$$

24. Here is the adjacency list for an undirected unweighted graph. Draw the circles and edges (links) below.

$a- > \{c, e\}$

$b- > \{c, d\}$

$c- > \{a, b, d, e\}$

$d- > \{b, c, e\}$

$e- > \{a, c, d\}$

25. What is the maximum number of edges an undirected graph with 5 vertices can have?

26. What is the maximum number of edges a directed graph with 5 vertices can have?

A *sparse* graph is a graph that has few edges compared to the number of vertices in the graph.

27. Draw a sparse graph here:

A *dense* graph is a graph that has lots of edges compared to the number of vertices in the graph.

28. Draw a dense graph here:

29. Suppose you know your graph is sparse. Would you use the adjacency matrix or adjacency list representation?

matrix   list

30. Suppose you know your graph is dense. Would you use the adjacency matrix or adjacency list representation?

matrix   list

31. Suppose you want to determine if node a is connected to node c in a graph. Node a is mapped to position 0 in the matrix and list representations. Node c is mapped to position 2 in the matrix and list representations.

How long (worst-case) does it take to determine if a is connected to c using the matrix representation? O(   )

How long (worst-case) does it take to determine if a is connected to c using the adjacency list representation? Assume the list of adjacent nodes is not in order. O(   )

32. What questions does your group have about graphs?