Graph Theory

GBW ACSL - Contest #3

Clayton O'Neill — clayton@oneill.net

2014-2015

What is Graph Theory?

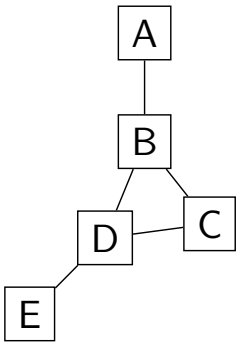
Graph theory deals with problems that are described in terms of points and the connections between the. Examples include friend relationships in social media, an airline route map or a flow chart. A graph is a mathematical way of modeling these sort of situations.

A graph has two key parts:

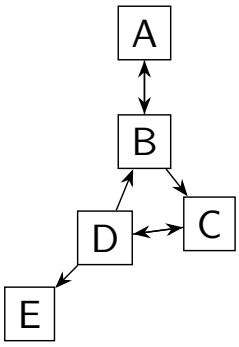
- Vertices (also known as nodes)
- Edges (also known as connections)

What Does a Graph Look Like?

The graphs below show examples of the two major types of graphs. In undirected graphs the edges simply show what vertices (or nodes) are connected. In directed graphs the edges also have a direction. If two nodes are connected in both directions then the arrow will be shown at both ends of the edge. In ACSL we will deal mostly with directed graphs. What are some examples of directed and undirected graphs you can think of?



Undirected Graph

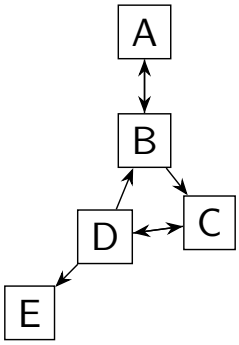


Directed Graph

Graph Notation

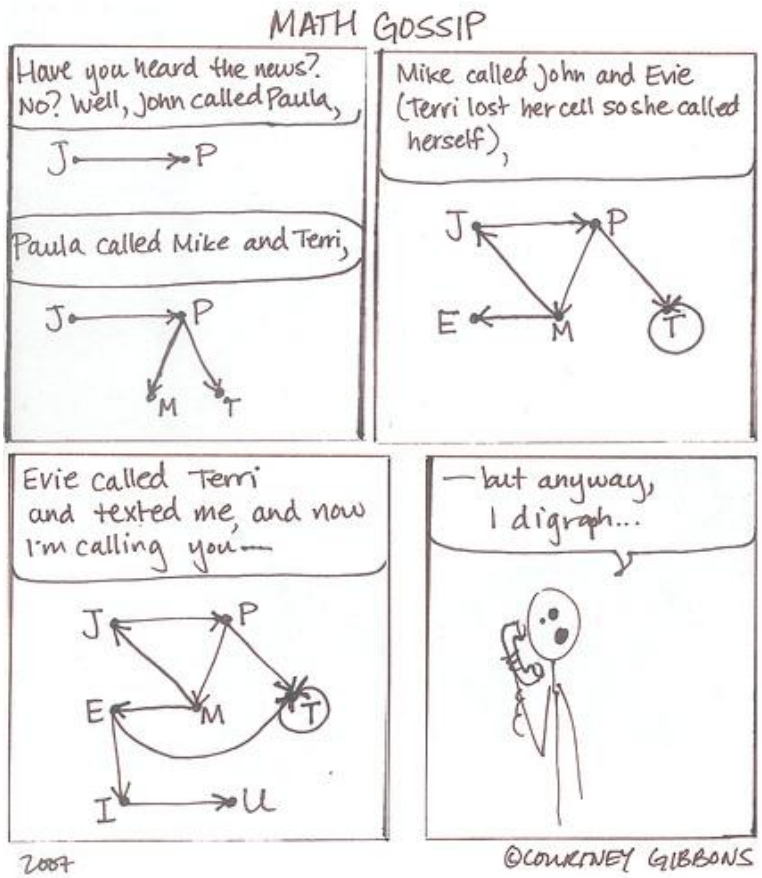
In the graph below, it has the follow nodes or vertices: A, B, C, D and E.

The edges (or connections) would be: AB, BA, BC, CD, DC, DB and DE. The edges are specified by listing first the vertex that the connection is from, and then the vertex that the connection is to. Note that listing AB and BA means that there is a bidirectional edge between those nodes. Listing CD and DC also gives us a bidirectional edge



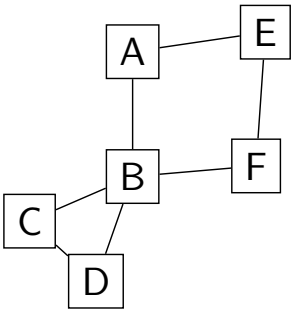
Graph Vocabulary I

- A *path* between two vertices is a list of vertices that would be traveled through. In the previous graph, *ABCDE* is a path from A to E.
- A *simple path* is a path that does not visit a vertex more than once. *ABCDE* is a simple path, but *ABCDCDE* is not.
- A *cycle* is a path that begins and ends at the same vertex. For example: *ABCDBA*. Since a cycle is a circle, you can rotate it and describe the same cycle. For example: *ABCDBA* is the same as *BCDBAB*.
- A *tree* is a graph that has no cycles.
- A *directed acyclic graph* or *DAG* is a directed graph that has no cycles.
- A directed graph is also known as a *digraph*.

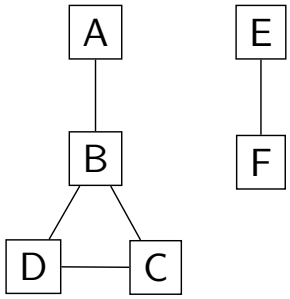


Graph Vocabulary II

- A *connected* graph is one where there is a path from any vertex to any other vertex.
- A graph that is not *connected* is made up of *connected components*. The graph on the right is made up of two connected components: $\{A, B, C, D\}$ and $\{E, F\}$.



Connected Graph

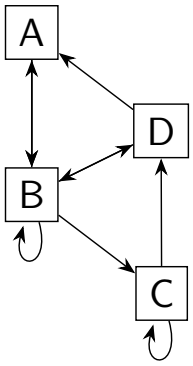


Unconnected Graph

Adjacency Matrix

There are two primary ways to describe a graph. You can list all of the vertices and edges, or you can use a *matrix* or array to describe them. When using a matrix, the table will be square, with a height and width equal to the number of vertices. The vertical axis shows where the edge starts and the horizontal axis shows where the edge ends.

		to			
		A	B	C	D
from	A	0	1	0	0
	B	1	1	1	1
	C	0	0	1	1
	D	1	1	0	1



Graph Vocabulary III

- A *complete* graph is one where the adjacency matrix is **all** ones.
- A *dense* graph is one where the adjacency matrix is **mostly** ones.
- A *sparse* graph is one where the adjacency matrix is **mostly** zeros.

		to			
		A	B	C	D
from	A	0	1	0	0
	B	1	1	1	1
	C	0	0	1	1
	D	1	1	0	1

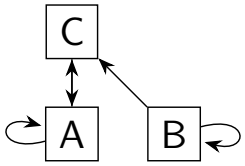
What Do You Need to Know?

1. How to draw a graph from a list of edges.
2. How to draw a graph from an adjacency matrix.
3. How to draw a an adjacency matrix from a graph drawing.
4. How to find paths of a given length from a vertex.

Items 1, 2 and 3 are fairly straightforward and just require some practice and understanding how adjacency matrices work. For item 4 there are two primary techniques.

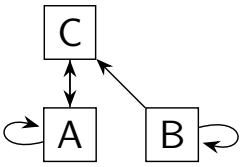
Finding the Number of Paths from a Vertex: Inspection

The first method for finding paths is "by inspection". If you were asked to find all paths of length two in the graph below, you would start at each node and traverse the graph mentally to find all of the options. You would find that the only length two paths were AAC, CAA, BBC and BCA. This approach is the simplest and works well on smaller sparser graphs. On larger denser graphs it can get confusing.



Finding the Number of Paths: Matrix Multiplication I

With *matrix multiplication* you can find all paths of a specific length in a single operation, but it is more complicated. The first step in this technique is to produce an adjacency matrix for the graph:



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

Note: In most ACSL materials you will not see the vertex names on the adjacency matrix. It is shown here only to clarify.

Finding the Number of Paths: Matrix Multiplication II

- The adjacency matrix shows how many paths there are between two vertices of length 1.
- If you multiply the matrix times itself, or square it, you will get the number of paths between two vertices of length 2.
- You can continue this pattern, raising the adjacency matrix to any point to find out how many paths of that length.
- For example: a matrix multiplied by itself 3 times (or cubed) will show all paths of length 3 in the graph. Multiplying it by itself 4 times will return all paths of length 4 in the graph, and so on.

Finding the Number of Paths: Matrix Multiplication III

How do we multiply a matrix by itself? Assume we have the 3x3 matrix shown below. To derive the answer for each cell in the resulting matrix, we have to multiply the rows and columns and then add the results:

$$\begin{vmatrix} A & B & C \\ D & E & F \\ G & H & I \end{vmatrix} x \begin{vmatrix} A & B & C \\ D & E & F \\ G & H & I \end{vmatrix} =$$

$$\begin{vmatrix} (AA + DB + CG) & (AB + BE + CH) & (AC + BF + CI) \\ (AD + DE + GF) & (BD + EE + FH) & (DC + EF + FI) \\ (AG + DH + GI) & (GB + HE + IH) & (GC + HF + II) \end{vmatrix}$$

As you can see, this requires a large amount of arithmetic. When doing the square of the matrix this isn't too bad, since all the numbers you're working with are zero or one.

Finding the Number of Paths: Matrix Multiplication IV

Below is shown the adjacency matrix for the graph shown earlier and the result of squaring it.

$$\begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{vmatrix}^2 =$$

	A	B	C
A	2	0	1
B	1	1	1
C	1	0	1

Because we raised the adjacency matrix to the second power, each entry in the matrix indicates how many length two paths there are from the x and y axes. For example, A/A indicates there are two length two paths from the vertex A back to itself. There are zero length two paths from A to B, and one length two path from vertex A to C.