ACSL Graph Theory

Billy Moses, Ryan Jian

February 15, 2013

1 Introduction

A graph is a collection of vertices and edges connecting different pairs of vertices. This general notion is extremely useful for modeling objects in the real world. Graph theory is a vast field, luckily however, ACSL only requires you to know the very basics. As such we will probably have a more advanced graph theory lecture sometime in the future.

2 Terminology

A **path** between two different vertices is a list of vertices where successive vertices are connected by an edge. A **simple path** is one where no vertex in the list is repeated.

A **cycle** is a simple path except that the starting and ending vertex are the same and.

Graphs can be **directed** meaning that an edge has a certain direction. An edge may connect vertex a to vertex b, but not the other way around. Typically this is denoted with an arrow indicating the direction of an edge.

Similarily, a graph can be **bidirectional** where all edges run in both directions. Typically this is indicated by edges without arrows, or arrows on both ends.

One way to represent a graph is using an **adjacency matrix**. A graph with N vertices is denoted by an N x N matrix where a 1 at the ith row and jth column indicates an edge between the ith and jth vertices.

3 Path Counting

Often, counting the number of paths can be very useful and is one of the major elements of graph theory that ACSL tests.

3.1 Naive method

We would highly recommend using this method to determine the number of simple paths as it is the computationally simplest, and it is harder to make careless errors.

One of the simplest ways to count the number of paths in a graph is just to manually find every single path. While this is computationally less efficient, and prone to error, this is one of the easiest ways for a human count the paths, and works well for graphs with a small number of vertices. In addition, this will always work for simple paths.

3.2 Matrix Multiplication

Do you LOVE multiplying matricies together again and again and again and again!? Well if so, do I have a method for you. Another method of counting paths uses the adjacency matrix. In order to determine the number of paths (simple **and** non-simple) of length k between two nodes, raise the adjancency matrix to the kth power.

3.2.1 Exponentiation by Squaring

To save time and reduce the possibility of error when multiplying matrices, we want to reduce the number of times we do it. Exponentiation by squaring relies on the fact that the exponent in raising a matrix to a power has a binary representation.

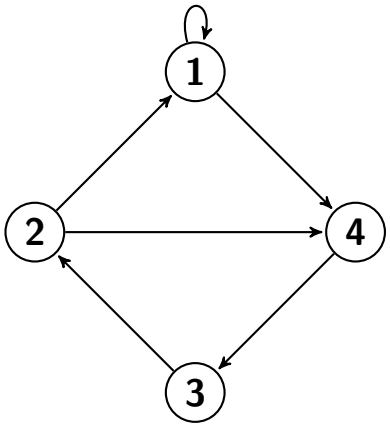
For example, if we wanted to compute some matrix to the 13th power, the naive method would be to repeatedly multiply the matrix by itself 12 times.

Instead, we can notice that 13 in binary is 1101, so we can repeatedly square a matrix, computing the 1st, 2nd, 4th, and 8th powers of the matrix. To get the 13th power, we just use the fact that $a^n * a^m = a^{n+m}$ and then multiply the 1st, 4th, and 8th powers. This uses a total of 5 multiplications, which is significantly less than what we had before. In general, for calculating A^N, this method uses $O(\log_2 N)$ multiplications.

4 Problems

1. Find the number of simple paths in the graph below from 3 to 4.

2. Find the total number of paths of length 3 in the graph below.



3. Draw a graph based on the adjacency matrix below

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$