CS472 Programming Assignment 1

Adam Lewis

January 5, 2017

Contents

1 Overview

Very few class libraries provide support for the graph data structure. In this assignment you will write a set of classes to implement a Graph ADT and use those classes to solve part of a classic problem from graph theory.

2 Background

2.1 Building a class library for working with graphs

You will need to implement a templated *Graph<type>* class that implements a graph whose nodes are of *type*. You will need to implement this in two ways: using an adjacency list and adjacency matrix as your private implementation. This means that you have to implement the *Graph<type>* as an pure virtual abstract class (if you're using C++) or as an interface (in either Java or C#).

The *Graph<type>* will need to implement provide the following interface:

- `bool adjacent(`*`type`*` x, `*`type`*` y)` : is there a node from x to y
- `Vector<type> neighbors(`*`type`*` x)` : Return a Vector containing the nodes have a edge from x to elements in the vector
- `void addEdge(`*`type`*` x, `*`type`*` y)` : add an edge from x to y if none exists
- `void deleteEdge(`*`type`*` x, `*`type`*` y)` : delete the edge from x to y if one exists
- *`Graph<type>`*` &dfs(`*`Graph<type>`*` &g, `*`type`*` &startNode)` : Do depth-first traversal of the graph *g*, starting from node *startNode*, and returning a reference to a new *Graph<type>* that contains the DFS traversal of *g*.
- *`Graph<type>`*` &bfs(`*`Graph<type>`*` &g, `*`type`*` &startNode)` : Do breath-first traversal of the graph *g*, starting from node *startNode*, and returning a reference to a *Graph<type>* that contains the traversal of *g*.

You should use the classes provided in the C Standard Library to implement this class (Vectors, Lists, and Maps are your friend).

2.1.1 Extra Credit

What is not specified in the functions *dfs()* and *bfs()* is what to do when you visit each node. Modify the two functions to accept an input parameter that accepts a function with the prototype:

- `void visit(`*`type`*` node)` What this function does is left up to the person using your classes. You will need to call the function at the correct point in your code.

1. Points You will get 10 extra credit points for attempting the problem and up to 25 points beyond those 10 points for a correct implementation.
2. A few hints The traditional way to do this in C++ is function pointers. The modern way to do this in C++11 is to use lambda objects. That's also how you would do this in Java. Note that this is painfully simple in dynamic typed languages such as Ruby.

3 Assignment

Implement the *Graph* classes described in the previous section.

Test your graph classes by writing a program that attempts to find a path from a starting node in your graph to some ending node.

A few assumptions:

- The graph is a directed graph.
- The nodes on the graph are numbered from 0 to $n - 1$, where n is the number of nodes in the graph.
- Assume that graph is stored in a text file where each line of the file defines the out-edges for that node. In other words, the first line of the file contains a list of space-separated integers that define the neighbors for node 0, second line defines the neighbors for node 1, and so on.
- We need only to find a single path, we aren't interested in things like the shortest path (yet...).
- You will have already implemented the hard part of this problem in your graph classes. Look at the functions for traversing the graph and think about what extra information you need to keep.

3.1 A few notes

- You may use your favorite programming language to solve the problems in this assignment. Note that one of the requirements for this assignment is that you code be reusable (users of scripting languages such as Ruby, Python, or Javascript take note).
- You cannot use the Boost libraries for this assignment if you're using C++ (this will change in future assignments).

4 Submission instructions

Combine your source code, test data, and examples of your program's execution into a single PDF document. Attach this document to your submission on Blackboard.