CS417 Programming Assignment 1

Adam Lewis

Contents

1 Overview

The C++ Standard Library does not provide support for tree and graph data structures. In this assignment you will write C++ classes that use the different components of the CSL to implement a Tree ADT and a Graph ADT.

2 Background

This is a two-part assignment

2.1 Graphs

In the second part of the assignment, you will need to implement a template `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. Each implementation needs to be subclasses of a the `Graph<type>` virtual abstract class.

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 that have a edge from the node x to elements in the vector
- `void addNode(type x)` : add a new node to the graph
- `void deleteNode(type x)`: Delete the node `x` from the graph, addressing the removal of any edges from `x` to any other node.
- `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.

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

Hints:

- It will simplify your implementation if the graph maintains a private vector that stores the contents of nodes. Do this and then you can refer to a node by using that node's index into this vector. Be careful... the contract for this ADT specifies that you use node values as parameters, not node indexes!

3 Using your graph class

Write a program in C++ that, given a directed graph, will check to see if a user-specified path from that graph contains a cycle. Your program will need to read the graph from a text file. Each line in the text file needs to be in the format:

```
node_id: node_list
```

where `node_id` is an integer used to identify a particular node and the `node_list` is a list of integers separated by commas that define the destination of each out-edge from the graph node. The path to be tested can be entered from the command line as a collection of node-ids separated by white space.

3.1 Example Data File

```
1: 1,2,3,4
2: 4,5
3: 1,5
4: 1
```

3.2 Notes

- A node that does not have any out-edges can be omitted from the data file.
- Note that loops are perfectly acceptable.
- In this case, assume that we are working with a directed graph.
  - We can represent undirected graphs in this scheme by including directed edges in both directions; i.e., from and to a node.

4 Assignment

Implement the `Tree` and `Graph` classes described in the previous section. Provide the application program that uses your graph class.

Provide the appropriate unit tests for your program, again either using the unit test framework in Visual Studio or a test program you write yourself.

4.1 HINT

If you are uncomfortable with the Tree and Graph ADTs, then I strongly suggest that you review Chapter 10 (Trees) and Chapter 15 (Graphs) of the textbook from CS372.

5 Submission instructions

You will be required to provide the following documentation for your submission:

- A summary of the class design for your classes (provide UML diagrams).
- Unit test scripts for your code.

- A listing of the source code for all programs
- A set of screenshots showing your program executing

This information needs to be combined together into a single document in PDF format (use MS-Word if you must, but export the file to PDF format). Please name the submitted file using the naming convention: "LastName.FirstInitial.assn01.pdf". Attach this file to your submission in Blackboard.

NOTE: Failure to follow submission instructions will result in point deductions from your grade. These point deductions will follow the Fibnancci sequence: first occurrence, 1 point; second occurrence: 2 points, third occurrence, 3 points, fourth occurrence: 5 points; and so on.