Csci 335 Assignment 5

Due May 18, 2017

Graph Representation (20 points)

You will read a directed graph from a text file (input graphs as text files are provided). Below is an example:

`Graph1.txt`

```
1 2 0.2 4 10.1 5 0.5
2 1 1.5
3 2 100.0 4 50.2
4
5 2 10.5 3 13.9
```

Each vertex is represented by an integer from 1 to N. Each line is of the form

<vertex> <connected vertex 1> <weight 1> <connected vertex 2> <weight 2> …

For each vertex you have a list of the adjacent vertices with positive edge weights. For instance, in the above example, vertex 1 is connected to vertex 2 (edge weight 0.2), to vertex 4 (edge weight 10.1) and to vertex 5 (edge weight 0.5). Vertex 2 is connected to vertex 1 (edge weight 1.5), vertex 4 has no outgoing edges, etc.

Represent a graph using an adjacency list. In order to test your implementation you will also read a second text file (let us call it AdjacencyQueries.txt) that will contain a set of pair of vertices. Your program (name it CreateGraphAndTest) will have to first create the graph by reading it from text file Graph1.txt. It will then open the file AdjacenyQueries.txt and for each pair of vertices in it you will cout whether the vertices are adjacent or not, and if they are you will cout the weight of the edge that connects them.

For example if the file AdjacencyQueries.txt contains
4 1
3 4
1 5
5 1
1 3
Then the output should be
4 1: Not connected
3 4: Connected, weight of edge is 50.2
1 5: Connected, weight of edge is 0.5
5 1: Not connected
1 3: Not connected

So, your program can be called for example as:

./**CreateGraphAndTest** Graph1.txt AdjacencyQueries.txt

Dijkstra's Algorithm Implementation 1 (50 points)

Implement Dijkstra's Algorithm, using a priority queue.

Write a program that runs as follows:

`./FindPaths <GRAPH_FILE> <STARTING_VERTEX>`

This program should use Dijkstra's Algorithm to find the shortest paths from a given starting vertex to all vertices in the graph file. The program should output all paths in the form:

Destination: Start, V1, V2, … , Destination, Total cost: X

You should print out the paths to every destination.

For example if you run the program having as input Graph2.txt (provided) starting from vertex 1, i.e.

./FindPaths Graph2.txt 1

Then the output should be

1: 1 (Cost: 0)

2: 1, 2 (Cost: 2.0)

3: 1, 4, 3 (Cost: 3.0)

4: 1 (Cost: 1)

5: 1, 4, 5 (Cost: 3)

6: 1, 4, 7, 6 (Cost: 6)

7: 1, 4, 7 (Cost: 5)

Topological Sorting (30 points)

Write a program that computes a sequence of vertices that satisfy the topological sorting sequence (i.e. implement the topological sorting algorithm).

./TopologicalSort <GRAPH>

The output should be the sequence of vertices. If a cycle is detected, just print the message “Cycle found” and terminate the program.

For instance:
./**TopologicalSort Graph2.txt**
should produce the message
“Cycle found”
Since there is not vertex with indegree zero in the graph.

./TopologicalSort Graph3.txt
should produce the sequence
1, 2, 5, 4, 3, 7, 6.

You are provided with three simple graphs to start and debug your program with.
For graphs 2 and 3 you are also provided with a picture corresponding to the input text file.
