California State Polytechnic University, Pomona

Project 3

Graphs

CS 241 – Data Structures and Algorithms II

Jose J. Sandoval

Jjsandoval1@cpp.edu

Dr. Hao Ji

March 10, 2017

**Section 1. Project description**

A program that implements a graph data structure for a practical application. The program reads city and road data from files and places them into a graph. The user can then query information from a city and get back the city’s number, 2 letter code, full name, population, and elevation. The user can also get the shortest distance between two cities and the path needed to be taken to get there. The user can also insert additional roads (edges) by providing the cities codes and a distance between them. Finally the user can remove a road by proving two city codes.

**Section 2. Project specification**

Files had to be read to get the information that make up the vertices and edges in the graph. I used ArrayLists to hold the vertices and another one that was an attribute of each vertex and had the edges going out of the vertex. I guess this method is resembles the LinkedList method.

**Section 3. Testing methodology**

To verify code correctness, the output was compared against the given output in the assignment. Each option was tested for errors by proving wrong input. However, I could not get Dijkstra’s algorithm working on my program so I could not get the path that makes up the shortest distance. Had I been able to get it programmatically, I would have done it manually to compare results.

**Section 4. Lesson learned**

I could not get the PriorityQueue to work. I tried following tutorials online to see how to use it and I even implemented the Comparator that it needs to compare the values in the queue. But I just could not get it to work and I ran out of time to start debugging it. I also found something weird with my code, I could not access the public modifiers of the Vertex class from the Digraph class. I could not figure out why that was happening.

**Section 5. Analysis of output (Project dependent)**

The time complexity appears to be O(E) for removing because I have to iterate through the edges to find the ending vertex and verify that that is the correct edge to be deleted. For adding an edge, the time complexity is O(1) because I just need to add the edge to the Arraylist of edges that is part of the startVertex.