Haussmann, Benjamin

Diaz – Winter 2018

CS 241-01

Section 1.

Implement a graph data structure for a practical application. The program takes in two hard coded input files for input, one representing city data and another representing connection data.

Section 2.

The program is designed to make a directed graph out of two hard coded data files. These files contain the cities (vertices) and the roads (edges). Together these build a map between various cities. The user is then allowed to select from 6 commands. The first allows them to query city information by entering a city code. The second will find the minimum distance between two cities. The third will insert a new road between user inputted cities with a given distance. The fourth will remove an existing road by entering the cities it connects. Fifth, will display a help message, and the last option is to exit.

Section 3.

I tested different aspects of this program in different ways. Testing the help and exit commands were easy and straight forward. Testing the other commands took a lot of effort. To test the ‘Q’ command, I first started by attempting to print out the Pomona information. When that failed, it was time to start digging. I was having issues within the linked list I used to store all of the cities information. I spent a while working on that and eventually fixed it. Then I tried the ‘D’ command. After a couple of minor fixes, I was able to get a path to print out, but the length was always 0. Eventually I made some changes to the algorithm, but then I was receiving very large distances and complex paths. Turns out, I had my compareTo() algorithm backwards and the method was finding the longest path. Once I reversed the comparison, I received the correct answers. The remove method took the longest and is still not finished. I believe all the code will work and the edge between two cities would be deleted, except for an error eclipse keeps throwing. When I try to pass the end vertex into the disconnect method, Eclipse says that the types do not match and suggests changing the type to the current type. The only way the error goes away is if I cast to the vertex to the vertex. This should make difference as the classes are literally the same, but when I do that, I get an exception during running that endVertex cannot be casted to type Vertex. It is a very frustrating error and one that I have spent many hours looking for answers to. Google had no answers and neither did anyone I talked to. I have finally given up and will continue to debug this problem in the next few days.

Section 4.

I spent a lot of time on this project. It took a very long time as it used almost every ADT we’ve been designing since the beginning of CS240. Not every student had the same teacher for CS 240, so everyone’s implementations of the various ADT’s this program relied on was a little bit different. I ended up rewriting all of my necessary ADT’s just for this project. It took a very long time and created a lot of confusion. (This may be the one upside of semester system.) Not everything is in the book and I had to figure out a lot of it on my own. In the end, I believe I did a good job and figured out most of it with robust solutions. I tried to make everything using generics so that it would all be reusable in the future. It definitely added to the amount of time it took.