Project #3

Dijkstra’s Algorithm

CS 241 Section 1

Matt Dohlen

[matt.dohlen@gmail.com](mailto:matt.dohlen@gmail.com)

4/22/2014

Section 1. Project description

In this project a graph structure was created using two data files. Then the user was able to perform a few operations on the graph to alter and display information about the graph. The operations were query a city, insert and delete a road, and find the shortest path from one city to another using Dijkstra’s algorithm.

Section 2. Project specification

The first step that my project took was to insert the data from data files into my graph object which consisted of an adjacency matrix to represent the roads between cities and an array of city object which each contained an abbreviated and full name, population, and elevation of the city. The graph object also contained many functions to insert, delete, and query elements of the graph. The main function in this object was the shortestPath function. This function used Dijkstra’s algorithm to find the shortest path between two elements. Notice that it only finds the shortest distance from a source element to a target and not a source element to all elements in the graph.

Section 3. Testing methodology

This project required a heavy amount of testing with the debugger. I had to use the debugger many times to ensure that the data from the files was being properly inputted into the graph. I also had to use it many times to map out what was happening in my implementation of Dijkstra’s algorithm. One of the major problems I was having was finding a way to represent distances from city to city as infinity. Initially I wasn’t testing to see if a road was set to infinity I was just choosing the shortest road, which when implementing infinity as -1 does not work. Although this did cause me a few problems, once I started up the debugger my errors were fairly obvious.

Section 4. Lessons learned

As stated before, this project provided me a lot of practice using an IDE’s debugger which is proving to be a very powerful and valuable tool as I get further in to Computer Science. This project also taught me to be a little more careful with my test cases because as I also stated before I had some simple errors that could have been had I been a little more attentive which would have saved me a lot of time.

Section 5. Analysis of output

The biggest loss in speed for my project was in the way I implemented Q in my Dijkstra’s algorithm. I used an array of visited vertices. This works fine until my I have to find which element to use as the next *u* (the next shortest road element). In order to find the next u I had to search the entire array of elements to find the next *u* which by itself has a complexity of V (number of elements in the graph). So my worst case complexity is of my Dijkstra’s algorithm is O(V\*(V-1) + V\*V) or O(V2+V2). This is because, if each (V)ertex has a road to every other vertex (V-1), it has to iterate V times through (V-1) neighbors to set their distances from the source and it also has to search through V elements to find the next *u* on each iteration. There is one aspect to my algorithm that could potentially cut the time considerable. This is because if my algorithm finds the target vertex it stops. This would cut the iterations of the algorithm considerably (although I do not know if it would be reduce to V/2 or log(V)). If I implemented my graph using a min heap the complexity of finding *u* would be reduced to *time to reheapify* \* 1 which is log(V) and the other portion would initially be V but after every iteration would drop by one because an element is being popped out of the heap.