Project 3

Shortest Path Problem

CS 241

Pablo Leon

E-mail: [pleon@csupomona.edu](mailto:pleon@csupomona.edu)

March 13, 2014

Section 1. Project Description

The purpose of this project was to better learn the inner workings of a directed graph, its implementations, and applications to everyday tasks. Such as a GPS system that can find you the shortest path from your current location to your desired destination. In order to achieve this task we also needed to learn the inner workings of Dijkstra’s algorithm to find said shortest path.

Section 2. Project Specification

My program allows the user to pick from a list of options that provide information about 20 different destinations, the shortest path to arrive at a destination, as well as add or remove new paths from destination to destination. When a path is remove or added to my graph it readjust the shortest distance to all destinations if affected by the new change. I did make an oversight and towards the end of working on this project noticed I had forgotten to keep track of the route taken in the shortest path. But despite that the calculated shortest overall path still result in correct values.

Section 3. Testing methodology

When testing my project I would often write a simpler version of the piece of code I was trying to implement in my main function, and run it. If the piece of code worked after several pieces of testing information it would then be moved to the function it was meant for. Plus once that code was situated properly in the function several tests would be carried out from main to make sure that function worked. By putting in many ouput statements between the code I was capable of discovering where my code was faulty much fast and was able to work on it.

In addition once all the program was done and working I tested my results to those given by the program on the assignment website, and when it came to shortest path I always got the correct values.

Section 4. Lessons Learned

Well my biggest lesson learned was that I need to read the prompt for the project several times over because I discovered some crucial oversights when I was too far and too late into the project to makes changes on time. Firstly misread my notes and somehow ended up using the Bellman-Ford algorithm instead of the Dijkstras’s algorithm that was asked for. Also I failed to notice I needed to show the path traversed in the shortest path, which mean by the time I noticed it was too late to get it done. I did attempt to work on displaying the shortest path traversal but with the time I had I ran into too many issue and decided to pass over it and work at making the rest of the program work properly. Lastly I programmed my graph as a linked list of nodes for the vertices that each held an array of size 20 to hold the distances to all its neighbors. These arrays would then be loaded into a 2-D array where the calculations for shortest path would take place. I choose this structure because my aim was to take up the least space possible when toring my information. I do know that I failed in my goal since I am technically holding two arrays of size 20 by 20 in memory instead of what should be one. This was due to my inexperience and poor planning that left me with little choices but to just do what I had to get the results I wanted.

Section 5. Analysis and Output

I was able to duplicate the total shortest paths calculated by the program on your websites, although I had many issues when structuring out these tasks, resulting in many errors. But as shown by my output the path to Pomona from Chino Hills is indeed 143 so for as many errors as I made with this project, I can say that I believe it achieves part of its main purpose, to find the shortest path. The time complexity of this program I believe is sadly O(n^3) if not worse, because not only must it read in and write all the nodes and edges from the text documents to start with, but then it also rewrites all the edge relations into another 2-D array. In this array I go over all the possible paths from vertex to vertex K number of times which is equal to the number of vertexes. So it ends up making V^3 comparisons when the Bellman-Ford algorithm is implemented. So in the end I feel I did quite poorly when planning and implementing this graph resulting in wasted storage space, and extra computing power then should be needed.