**Software Requirements Document for**

**Dijkstra’s Algorithm**

CS312 Analysis of Algorithms

Chris Sousa, Cameron Jewett, Marissa Kulz

12/6/2020

1 Introduction

1.1 Project Scope and Purpose

This project is a demonstration of Dijkstra’s Algorithm on a weighted directed graph built for the purpose of logical demonstration. It demonstrates the design team’s understanding of Dijkstra’s algorithm, how to follow from start to finish, record the results, and display them afterwards.

1.2 Intended Audience

The intended audience is the class professor and peers.

2 Description

2.1 Product Description

The project is small in scope, consisting of a single C++ class. This class implements the weighted digraph by loading from a file containing a matrix representation. It then executes Dijkstra’s algorithm, finding the shortest paths from a given starting node to all other nodes in the graph and recording both the distance and the path travelled to each. Finally, it prints out the calculated values.

2.2 Operating Environment

This program is designed to run on a 64-bit Windows system through Visual Studio.

2.3 User Classes

2.3.1 Weighted\_Digraph:

void clearPaths()

void printPath()

Weighted\_Digraph()

~Weighted\_Digraph()

std::vector<int> calculateDistances(int start)

void loadGraph(std::string path)

void printSolution()

3 Interface

3.1 User Interface

This is a console application. It will run in the command line once started through the Visual Studio debugger.

3.2 Hardware Interface

A computer running 64-bit Windows, a keyboard, a mouse, and a monitor.

3.3 Software Interface

Visual Studio. There is not a precompiled executable, so the program must be run in a debugger.

4 User Document – How To

The software is built and preloaded with a single digraph in the file “matrix.txt”. To use a different graph, you can either replace the contents of “matrix.txt”, or change the path in the function on line 11 of the main program.

The substitute graph should be formatted as a matrix, where if there exists an edge between nodes *i* and *j*, then position *(i, j)* in the matrix is the weight of that edge; if there is no edge, then the value of *(i, j)* should be -1.

5 Conclusions and Lessons Learned

Dijkstra’s Algorithm is used to find the shortest path between nodes within a graph. The algorithm keeps track of the shortest distance between the source node and other nodes, constantly updating them until it finds the shortest path to the destination. We learned how to implement this algorithm in a C++ environment to analyze a weighted graph.

A lesson that we learned was how to implement Dijksta's algorithm into a matrix. Most of our other coding assignments used other structures with pointers to other nodes. So having a matrix and understanding it was something new. This was a good learning experience though.