Brandon Nguyen
brandonn@sfu.ca
Student #: 301219561

CMPT 310 – Assignment 1 Written Document

I decided to interpret our n by n matrix as a weighted directed graph and ran a modified **Djikstra's algorithm** on it to find the shortest path from our start location to our goal location.

Typically, Djikstra's algorithm finds the shortest path from a node, to EVERY other node, but I stop the algorithm once we know that the cost of our goal location can no longer be updated with a smaller cost.

The **algorithm** is as follows:

```
FOR(node in S):
    IF(node is goal location):
        Traverse S to find the least cost path from start location to goal location
        Return least cost path

    Check all adjacent nodes

    IF(adjacent node is already in S):
        Do nothing
    ELSE IF(adjacent node hasn't been visited)
        Add adj_node, node(parent), and total cost of adj_node to visited list
    ELSE:
        IF(cost to get to adj_node < known cost to get to adj_node):
            Update cost of adj_node

    Remove the smallest cost node from visited and add it into S
```

The algorithm keeps track of two lists of nodes, visited and S.

Visited tracks nodes that have been visited by the algorithm but haven't yet been solidified into set S. In this list, you'll find 3-tuples consisting of the node itself, it's parent node, and the smallest known cost to this node.

S tracks the set of nodes from which we know that we cannot possibly update their cost with a smaller cost. We expand (exploring surrounding nodes) from nodes in this set as to ensure the smallest possible cost of visiting nodes not yet solidified in S.

Initially, all node costs, besides the start location, are unknown and thus set to infinity. From each node in S, starting from our start location, the algorithm takes a look at the surrounding nodes. If the surrounding node has not yet been visited, add that node and

Brandon Nguyen
brandonn@sfu.ca
Student #: 301219561

relevant information to our set visited. If the cost of getting to the surrounding node is smaller than our current known cost, then we update that surrounding node's cost to our new smaller cost. After exploring all current possibilities, we take the node in visited with the smallest known cost and add that into set S. This smallest cost node cannot be updated with a smaller cost in the future because all costs are non-negative and thus we cannot possibly get back to this node without incurring a great cost in future iterations.

This solution may not be optimal or efficient because it explores all possibilities instead of using an informed heuristic (which would give us an estimation). However, because it explores all possibilities, Djikstra's algorithm is certainly complete. There is no possibility that we will get a smaller cost path, assuming all costs are non-negative.

I used the 5x5 grid that was supplied in the assignment to test my code, along with varying the start and goal locations. I did not test the algorithm on a bigger grid, however I'm confident that my solution will still produce a correct answer, given that the algorithm is complete.