Elliot Lee
U0344180

1. Who is your programming partner? Which of you submitted the source code of your program?

My partner for this assignment is Will Frank. He submitted the source code for this assignment.

2. Evaluate your programming partner.

Will is a very capable programmer. He is good at catching my mistakes, and I find working with him very easy.

3. Does the straight-line distance (the absolute distance, ignoring any walls) from the start point to the goal point affect the running time of your algorithm?

No, the straight line difference does not affect the running time of our algorithm at all, since our path has to go around walls.

4. Explain the difference between the straight-line distance and the actual solution path length. Give an example of a situation in which they differ greatly. How do each of them affect the running time of your algorithm? Which one is a more accurate indicator of run-time?

The straight line distance is the direct path between the start and the goal nodes, ignoring any walls in the maze. The actual solution path length is the path our algorithm is trying to find, in other words, the shortest path from the start node to the goal node, going around all of the walls. The straight-line distance has very little impact on the run time of our algorithm, because our algorithm is not trying to calculate the straight-line distance. For example, below, the straight-line distance from the goal node to the start node is 2. However, this does not impact the run time of our algorithm, because our algorithm must calculate the actual solution path, which in this case is 10 nodes long. This means that the actual solution path length is going to be a much more accurate indicator of run-time than straight-line distance.

```
XXXXXX
XS    X
XXXX X
XG X X
X     X
XXXXXX
```

5. Assuming that the input maze is square (height and width are the same), consider the problem size, N to be the length of one side of the maze. What is the worst-case performance of your algorithm in Big-Oh notation? Your analysis should take in to account the density of the maze (how many wall segments there are in the field). For example, a completely open field with no walls other than the perimeter is not dense at all, as opposed to the example maze given "bigMaze.txt", which is very dense. There is no one correct answer to this since solutions may vary, but you must provide an analysis that shows you are thinking about the problem in a meaningful way related to your solution.

For this problem, a completely open maze represents a much more dense graph than a thickly walled maze. In an open maze, especially larger ones, most of the nodes have four neighbors, three of which get added to the queue during a DFS, and whose own three neighbors must be further explored later. This takes significantly more time than in a maze which has many walls, since the walls do not count as neighbors. This means that many nodes will only have two neighbors, and the DFS will have to pursue many, many fewer potential paths to the goal node than if all the nodes had four neighbors. Because of this, the worst case scenario for our algorithm is a completely open maze with the goal and start nodes in opposite corners. In this case, the algorithm would have to explore every node in the graph before reaching the goal node, so the big-oh would be O(V + E). To put it in terms of N, the outside perimeter of the maze are all walls, so we only have to look at a graph with (N – 2) * (N – 2) vertices. There are four vertices with two edges (the corners), 4 * (N – 4) vertices with three edges, and the remaining vertices all have four edges. So there are a total of 8 + 4 * (N – 4) + $(N – 4)^2$ edges. Since we only care about the asymptotic behavior of the algorithm, we can reduce this to $(N – 2)^2 + (N – 4)^2 = 2N^2 = N^2$. Therefore, the worst case behavior of our BFS algorithm is $O(N^2)$;

6. How many hours did you spend on this assignment?

We spent about 6 hours on this assignment.