Savannah Simmons u1086770

1. Who is your programming partner? Which of you submitted the source code of your program?
Jana Klopsch was my partner and she submitted the source code.

2. Evaluate your programming partner.
Jana was great and we worked really well together. I probably took the lead more often on the coding, while she had a knack for figuring out our bugs. I also got really sick the last couple days and she was very accommodating to the wrench that threw in our plans.

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?
Generally, yes. As explained below, there are cases where the straight line distance is not indicative of the path length. But as a rule, the longer the straight line distance, the more steps the path has to take at minimum to reach the goal, regardless of walls. For each extra step in our path, we're going to have to follow up to three different paths.

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?
One obvious one would be in a situation such as this:

```
XXXXXXXXXX
XS       X
XXXXXXXX X
XG       X
XXXXXXXXXX
```

Where the straight line distance is significantly shorter than the actual path length. The straight line distance tells us the minimum length of our path, and so affects the minimum or best case scenario of our run time. If there is a straight vertical or horizontal line between our start and goal, then that is the shortest possible path. If the straight line is a diagonal and the distance is c, and the shortest possible path length (i.e. if there were no walls) is going to be (2^(½))c by the Pythagorean Theorem.

The actual path length is going to be what affects our run time in practice though. For each step in our solution path, we have potentially explored up to two additional paths to the same depth.

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 into 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.
While this seems counterintuitive (at least to me), a more dense maze is going to have a faster run time than a less dense one with the same distance between S and G, because we have fewer possible paths to explore. For example, a maze like this:

```
XXXXXXXXXX
XS       X
XXXXXXXX X
XXXXXXXX X
XXXXXXXXGX
XXXXXXXXXX
```

Would be ideal because for each node we dequeue, we only have to look at and explore one of its neighbors.

Our worst case scenario would be a maze with no walls where S and G are at opposite corners, where for each node we dequeue we likely have to explore three paths. In a case such as this, we are going to end up hitting every node. If N is the length of one side of the maze, then N-2 is going to be the length of one side of the actual nodes. And since we'll hit each one once, complexity would be (N-2)^2, i.e. O(N^2).

6. How many hours did you spend on this assignment?
Around 10 hours.