Harrison Smith Assignment 10-Pacman Swaroop Fall 2019
Ben Malohi

1. 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? In other words, does how close the Start and Goal are affect how fast your algorithm finds a shortest "Manhattan" distance?

Yes, the closer the Start and Goal are together, the quicker the time it would take to compute. With bfs, we basically make Circles around Start's neighbors and then circles around those neighbors Making basically one large circle by going level by level of neighbors. With the straight-line distance, the time to compute, maybe worse than the actual implementation because there would be more neighbors to check.

2. 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?

Straight line distance there is no obstacles to work around, thus you just have to circle around each neighbor layer to get to the Goal. However, if you with the actual solution path, some neighbors are blocked off because of walls or "obstacles". If the straight line would be all the way from the upper left corner to the bottom left corner, in both instances, there would be more neighbors that bfs would have to check all the neighbors leading up to it. The actual solution path length is more accurate of the run time because we would have less neighbors to touch as our path would be limited.

3. 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). Create a test example to verify your thinking. If possible, augment your code to keep a counter of how often a node is looked at (i.e., every time your code checks to see if a node has already been visited, it is being "looked at").

Worst case, we would have to touch each neighbor node, resulting in O(N) run time.

4. Hypothesize what affect using depth first search would have on the path planning? Provide an example maze for the purpose of this discussion. Would depth first solve the above example in a faster or slower manner?
   Provide enough discussion to show us you have seriously considered this as well as all the other questions.

```
XXXXXXX
X S     X
X       X
X      GX
XXXXXXX
```

In this scenario, depth first search would perform worse than bfs, because bfs would have to touch each and every node, since it is the furthest right node, however, bfs would only have to touch 3 layers down because it would visit all of S's neighbors, then their neighbors, and their neighbors then would have it found.

Thus, we conclude that overall that bfs would perform most efficient for the Mazes because it would usually have touched less neighbors. There maybe a few cases where dfs would perform more efficiently such as if goal is furthest to the left, we would end up touching less neighbors. However, since we do not know where G is, bfs is the quickest algorithm to use for this assignment

5. One more thought problem (don't spend more that 15 minutes on this). Say you created an entire matrix of nodes representing the maze and did not store any edges. To know if an edge exists for any node at (R,C) you would have to check (R-1,C), (R+1,C), (R,C+1), and (R,C-1). Can you think of an other algorithm to find the shortest path from a given start (R1,C1) to a goal (R2,C2).

There are other ways that you can try to use to approach that quicker, however, we would need to be able to move diagonally because then we could the step for (R – 1 , C – 1) and (R + 1, C + 1), if we were able to add the diagonal movement, it would be able to get from (R1,C1) to (R2,C2) in the first iteration of checking the neighbors. However, if we assume that we cannot violate that condition. Then we conclude that the given is the best approach to finding the path.

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

We spent about 11 hours in all on this assignment