Nickolas Komarnitsky
Analysis
Assignment 9
03/30/2017

**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?**

No, the program is only affected by the obstacles in between, not the straight line distance.

**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 distance between the start and goal without any obstacles in the way. The actual solution path length is the distance that has to be traveled to get around any obstacles in the way of the start and goal. For example, if the start and goal are only one space away from each other, the straight-line distance is 1, but there is a very large wall in between them then the actual solution path length becomes a much bigger number. The straight-line distance does not affect the running time, but the actual solution path length can affect the run time based on the amount or size of the obstacles. The actual solution path length is the more accurate indicator of run-time.

**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").**

The worst case performance of our algorithm would be O(N), the bigger the maze is the longer it will take to run. Density would be a constant.

**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.**

Depth first search would affect path planning because it is checking every node at once. It goes through each and every node, not towards the goal. It won't always find the path.

Example maze: bigMaze.txt

It would solve the above example in a slower manner, as it checks every node in the maze, while breadth-first only checks the nodes on the way to the goal.

**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).**

Yes, you could start by checking the straight line distance to go from the start to the goal, then check all of the nodes around it, ignoring the walls, and get their straight-line distance to the goal and

move to the node with the lowest straight-line distance, keep going like that until you reach the goal node.

**How many hours did you spend on this assignment? 10 hours**