CS 4710 – Homework #2 Write-Up

Pak Hin Luu and Ellie O’Neil

1. We used an A\* path finding algorithm. Before we dive into the various steps of the algorithm, we need to first outline the infrastructure needed to carry out such processes. We needed an open list, which is a priority queue ranked by a particular f-score to look at nodes at the “frontier” but have not been examined and a closed list for nodes that we have examined. The f score is the sum of the g score and the h score. The g score measures the length of the line that connects the starting node and the node being passed in. The h score is the number of blocks you need to move horizontally and vertically to get to the ending node. Also, we set up a node structure which keeps track of any relevant parent nodes (to backtrack the path we eventually find) and also the associated point in the worldMap. We can now proceed to examining the algorithmic process.

We first added the node at the starting location to the open list. We then repeated the following until the open list was empty:

Pop the open list. If the popped node is the end point, trace the parents iteratively until you reach the starting node to return the path. Otherwise, add the popped node to the closed list. Now, check all the “neighbor” nodes (adjacent nodes that the robot may move to). If a neighbor node is neither in the open list nor in the closed list, add the neighbor node to the open list and set the neighbor node’s parent to the popped node.

After this loop, the search algorithm should find the optimal path, as measured by number of moves (as opposed to being measured by distance). Here are our results for the various tests we ran:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | # of moves (with uncertainty ON) | # of Pings (with uncertainty ON) | # of moves (with uncertainty OFF) | # of Pings (with uncertainty OFF) |
| **Test Case #1** | 3 | 13,005 | 3 | 18 |
| **Test Case #2** | 7 | 113,009 | 7 | 122 |
| **Test Case #3** | 19 | 517,021 | 19 | 538 |
| **Test Case #4** | 54 | 992,040 | 38 | 1,032 |

In terms of space complexity, in the worst case is (x\*y), with x being the length of the world map and y being the width of the world map. In the worst case, every node in the graph will be evaluated and added to both the open and closed list.

1. We adapted the algorithm to uncertainty by the following steps: We first looked at the pingMap code to see how the method dealt with uncertainty. We realized that the further the node that was passed into pingMap was, the greater the uncertainty was associated with the validity of the string pinged (either a wall or a point the robot may move to). Though it may be inefficient, our solution was to run pingMap on a node a sufficient number of times to be reasonably confident with the result pinged. The number we chose was 1000. Every single time ping was called with uncertainty, we implicitly ran ping actually 1000 more times. The final result returned to the initial ping call was the result that happened more frequently (an “X’ or an “O”). The reasonably sized test cases we ran never had the starting distance sufficiently far enough from the ending point, such that 1000 “experimental” pings were insufficient.
2. We initially had an algorithm that calculated the g score that only took into account the number of horizontal and vertical moves from the starting node. Therefore, this algorithm, on some test cases where diagonal moves were more efficient, was not optimal. For instance, the test case, “myInputFile4.txt” returned 54 total moves, even though the optimal path was 38 moves. So, we updated our algorithm to produce a more robust g score, taking into account diagonal moves. Diagonal moves are more efficient as shown by the triangle inequality theorem, where the hypotenuse of a triangle is always less than the sum of its two legs. We updated our g score calculations to a distance formula that represents the length of the line that connects the current node and the starting position. For the four test cases (with uncertainty OFF) we ran, these were the results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | # of moves (old g score) | # of Pings (old g score) | # of moves (updated g score) | # of Pings (updated g score) |
| **Test Case #1** | 3 | 18 | 3 | 18 |
| **Test Case #2** | 7 | 122 | 7 | 122 |
| **Test Case #3** | 19 | 538 | 19 | 538 |
| **Test Case #4** | 54 | 1545 | 38 | 1032 |

Though there were not differences in performance for the first 3 test cases, the

incremental benefit the updated g score provided is obvious in test case #4.