Jason Hunter
Gardner Fiveash
10/7/16

Homework 2 Writeup

Our basic path-finding algorithm works in three stages. Stage one applies the heuristic function to the list of open nodes. An open node is a node that is in a cardinal direction of an already visited or 'closed' node with a value of 'O'. The heuristic function selects the best open node available. That nodes around the selected node are examined and added to the beginning of the list of open nodes if they meet the criteria. Then the selected node is added to the closed list and removed from the open list. The heuristic function we used was the a* method described in class:

$$F(n) = g(N) + h(N)$$

This function evaluates nodes based on how far they are from the start and how far they are from the end (Manhattan distance). This process of pinging the best node on the open list, examining spaces around it, and moving it to closed will continue until the list of open nodes is exhausted or the end point is reached.

Stage two is handled by our method "certainTravel." This method analyzes the list of closed nodes, determines which are needed for the path from start to end, and creates an ArrayList of strings that is used in Stage three as a map. "CertainTravel" works backwards. It searches the closed list for nodes that are distance 1 away from the end, adds a string to the map ArrayList, updates the current node to be the node just found, and removes that node from the closed list. This process continues until it reaches the start. If two nodes satisfy the criterion of being 1 away from the current node, the method selects the node that is closer to the start position. If they are also both the same distance away from the start position, the node that was added to the closed list first is selected.

Stage three uses the map of sequential directions produced by the "certainTravel" to actually move the robot from start to finish.

Datasets that are more inefficient for our algorithm have many forks where both options are equally attractive to the heuristic function. These make the first stage of our algorithm do many more pings of the map, which in turn makes the second stage of the algorithm need to correctly decide between options, adding in more of a risk of not back-tracing the correct path.

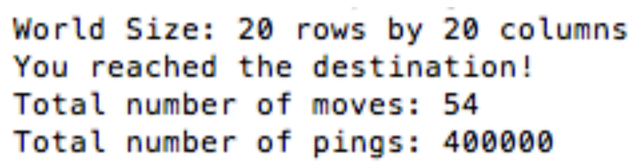
We found a solution for the dealing with uncertainty that required little change to our algorithm. We wrote two methods that called pingMap on every point in the map 1000 times, and found that even the farthest away points were pinged correctly more than 50% of the time. We used this observation to build an array that represented the original map, and then modified our certainMove and certainAddToOpen methods to take in an array instead of a world. This approach minimizes moves at expense of many pings. When testing with our 20x20 world, it reached the destination with 54 moves and 400,000 pings (400 points and 1000 runs to calculate the probability). We tried reducing the number from 1000 but this caused our program to break. Because we didn't need any additional moves, our run times for certainty and uncertainty were about the same.

If you change the heuristic to pure Manhattan distance, the first stage of the algorithm is much less efficient because it doesn't take distance from the start in to account. This keeps the while loop running longer as more and more open nodes have to be examined to determine the correct path.

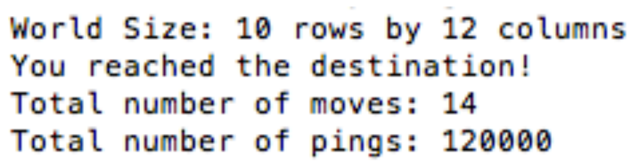
If the backtrace method changed selection criteria for how to handle two possible paths, it wouldn't work because it would get stuck at a dead end with no way to backtrack.

Below are screenshots of our number of moves and pings for teach map size, for both uncertainty and certainty. You can see that each uncertain test has the same number of moves as the certain test, but many more pings.

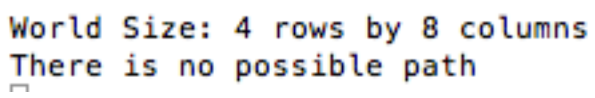
Uncertainty:



```
World Size: 20 rows by 20 columns
You reached the destination!
Total number of moves: 54
Total number of pings: 400000
```

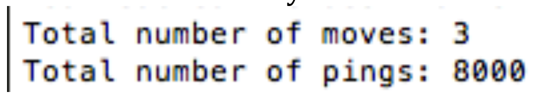


```
World Size: 10 rows by 12 columns
You reached the destination!
Total number of moves: 14
Total number of pings: 120000
```



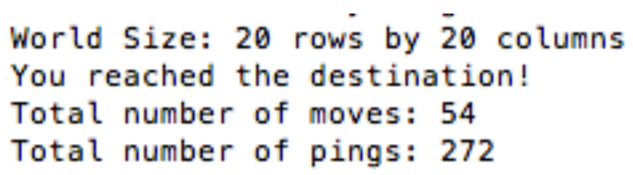
```
World Size: 4 rows by 8 columns
There is no possible path
```

World size: 2 rows by 4 columns



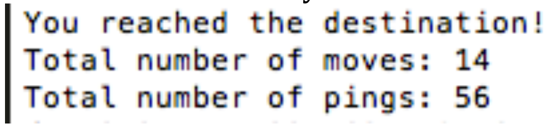
```
Total number of moves: 3
Total number of pings: 8000
```

Certainty:

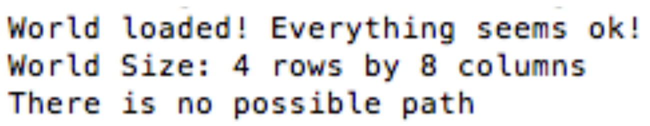


```
World Size: 20 rows by 20 columns
You reached the destination!
Total number of moves: 54
Total number of pings: 272
```

World size: 10 rows by 20 columns

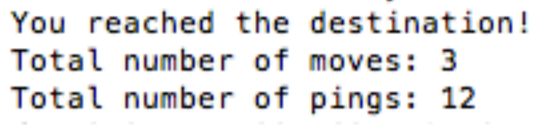


```
You reached the destination!
Total number of moves: 14
Total number of pings: 56
```



```
World loaded! Everything seems ok!
World Size: 4 rows by 8 columns
There is no possible path
```

World size: 2 rows by 4 columns



```
You reached the destination!
Total number of moves: 3
Total number of pings: 12
```