Tom and Jack: Irrational Agents Write-up

Description of Search Problem:

States: Each state of the problem contains two main pieces of information: The map and coordinate location(s) of the block. The map will never change, so in practice the state can just be represented by a Node object containing the block's two (x,y) values, a boolean describing whether it is standing or not, the parent of the node, and the path cost to get there from the starting condition.

Initial state: Contains the map and a starting node. The starting node has both coordinates as the location of 'S' on the board. This node has no parents or cost, and it is standing up.

Goal test: If the block is standing up on the map tile marked 'G,' the maze is solved.

Actions: At any time, the block can roll left, right, up, or down from a standing or lying down position. The block will not roll over spaces marked '*,' and it will not roll off of the map.

Path cost: Each move costs 1, so the total path cost is just the number of moves made up until that point which is incremented within the Actions() function.

Heuristics:

Manhattan: Our "manhattan" heuristic is a spin on the classic Manhattan distance. Because our block in this program can move 2 spaces at a time by tipping on its side from a standing position, the standard calculation of the Manhattan distance would be an overestimate of the number of moves in the case of a straight line to the goal. That would make this heuristic no longer admissible. To get around this issue, we mapped out multiple variations of the game on paper and found that the minimum possible moves that it would take to get to the goal would be ⅔ of the total Manhattan distance. To get around this in the code, we use the formula in Eq. 1.

$$f(n) = path\ cost + 2/3(Manhattan\ distance),\ where \qquad \text{[Eq. 1]}$$

$$Manhattan\ distance = |goal.x - current.x| + |goal.y - current.y|$$

We compared the f(n) values between nodes in the frontier to determine which move would be chosen next. Because of this fraction multiplication, the heuristic will never overestimate the number of moves to the goal state and is thus admissible.

Trivial: Because our Manhattan distance heuristic worked and was non-trivial, we wanted to make a trivial, yet admissible heuristic to compare to our manhattan heuristic. To make sure this heuristic was never an overestimate of the real move cost, we just decided to add 1 to the current cost and compare that to the next node's cost plus 1. Although this does not get us to the goal while visiting less nodes than a uniform-cost search, it is a good representation of how much a good heuristic can improve the number of nodes visited. While this heuristic will still find an optimal path, it will have to visit more nodes without the guidance of knowing how close it might be to the goal.

Maze Results:

Maze 1:

((0, 0), (0, 0)), ((0, 1), (0, 2)), ((0, 3), (0, 3)), ((0, 4), (0, 5)), ((0, 6), (0, 6)), ((1, 6), (2, 6)), ((3, 6), (3, 6))
((4, 6), (5, 6)), ((6, 6), (6, 6))
Number of moves using manhattan heuristic: 8
Number of nodes generated using manhattan heuristic: 45
Number of nodes visited using manhattan heuristic: 18

Number of moves using trivial heuristic: 8
Number of nodes generated using trivial heuristic: 324
Number of nodes visited using trivial heuristic: 120

Maze 2:

((1, 0), (1, 0)), ((2, 0), (3, 0)), ((2, 1), (3, 1)), ((4, 1), (4, 1)), ((4, 2), (4, 3)), ((4, 4), (4, 4)), ((3, 4), (2, 4))
((1, 4), (1, 4)), ((1, 3), (1, 2)), ((2, 3), (2, 2)), ((2, 4), (2, 4)), ((1, 4), (0, 4)), ((1, 3), (0, 3)), ((1, 2), (0, 2))
((2, 2), (2, 2)), ((2, 3), (2, 4)), ((3, 3), (3, 4)), ((4, 3), (4, 4)), ((4, 2), (4, 2))
Number of moves using manhattan heuristic: 18
Number of nodes generated using manhattan heuristic: 138
Number of nodes visited using manhattan heuristic: 60

Number of moves using trivial heuristic: 18
Number of nodes generated using trivial heuristic: 150
Number of nodes visited using trivial heuristic: 67

Maze 3:

((6, 3), (6, 3)), ((6, 4), (6, 5)), ((6, 6), (6, 6)), ((5, 6), (4, 6)), ((5, 7), (4, 7)), ((3, 7), (3, 7)), ((3, 6), (3, 5))
((2, 6), (2, 5)), ((1, 6), (1, 5)), ((1, 7), (1, 7))
Number of moves using manhattan heuristic: 9
Number of nodes generated using manhattan heuristic: 156
Number of nodes visited using manhattan heuristic: 52

Number of moves using trivial heuristic: 9
Number of nodes generated using trivial heuristic: 684
Number of nodes visited using trivial heuristic: 226

Maze 4:

((1, 0), (1, 0)), ((2, 0), (3, 0)), ((2, 1), (3, 1)), ((4, 1), (4, 1)), ((4, 2), (4, 3)), ((4, 4), (4, 4)), ((3, 4), (2, 4))
((1, 4), (1, 4))
Number of moves using manhattan heuristic: 7
Number of nodes generated using manhattan heuristic: 24
Number of nodes visited using manhattan heuristic: 10

Number of moves using trivial heuristic: 7

Number of nodes generated using trivial heuristic: 62
Number of nodes visited using trivial heuristic: 24

Maze 5:

((3, 2), (3, 2)), ((3, 1), (3, 0)), ((2, 1), (2, 0)), ((2, 2), (2, 2)), ((3, 2), (4, 2)), ((3, 1), (4, 1)), ((2, 1), (2, 1))
((2, 2), (2, 3)), ((2, 4), (2, 4)), ((1, 4), (0, 4)), ((1, 3), (0, 3)), ((2, 3), (2, 3)), ((2, 2), (2, 1)), ((3, 2), (3, 1))
((3, 0), (3, 0)), ((2, 0), (1, 0)), ((0, 0), (0, 0)), ((0, 1), (0, 2)), ((0, 3), (0, 3)), ((1, 3), (2, 3)), ((1, 4), (2, 4))
((3, 4), (3, 4)), ((3, 5), (3, 6)), ((3, 7), (3, 7)), ((2, 7), (1, 7)), ((2, 6), (1, 6)), ((3, 6), (3, 6)), ((3, 5), (3, 4))
((4, 5), (4, 4)), ((4, 6), (4, 6)), ((5, 6), (6, 6)), ((5, 7), (6, 7)), ((7, 7), (7, 7)), ((7, 6), (7, 5)), ((7, 4), (7, 4))
((7, 3), (7, 2)), ((6, 3), (6, 2)), ((6, 1), (6, 1)), ((5, 1), (4, 1)), ((3, 1), (3, 1))
Number of moves using manhattan heuristic: 39
Number of nodes generated using manhattan heuristic: 289
Number of nodes visited using manhattan heuristic: 136

Number of moves using trivial heuristic: 39
Number of nodes generated using trivial heuristic: 295
Number of nodes visited using trivial heuristic: 140

Discussion of Results:

Going into our tests of each maze, we expected to see our Manhattan heuristic outperform our trivial heuristic in every case. It was interesting to see the difference in the number of nodes visited, and thus the number of nodes generated by each heuristic. Since the Manhattan heuristic visits fewer nodes, there are fewer nodes generated by the Actions() function. One case in which the Manhattan heuristic did not perform much better than the trivial heuristic was in maze 5. In order to solve the maze, the block needs to traverse through the entire map before returning to the goal. The Manhattan distance heuristic has a tendency to explore nodes closer to the goal than the trivial heuristic, so the improvement was not as significant. In fact, the Manhattan heuristic only saved on 6 generated nodes.

In maze 3, however, the Manhattan heuristic outperformed our trivial heuristic by quite a bit, generating only 156 nodes as opposed to trivial's 684 generated nodes. Because maze 3 is fairly open with few obstacles in the way of the goal, this is a great case for our Manhattan heuristic, which will immediately head towards the goal state. These results were consistent with what we expected to see, as the Manhattan distance is a much more sophisticated heuristic than simply adding 1 to the path costs. It makes sense that the Manhattan distance would outperform the trivial heuristic in most cases, with the exceptions being the worst-case mazes for a Manhattan distance heuristic (such as in maze 5).