Introduction to AI Assignment 1 Report

Dev Patel, Efrain, Sahil

February 24, 2025

We are using Python to do this project.

Part 0

The "main.py" file is used to create the mazes. First we are creating a gird of zeros that represent the unvisited cells(they are all unvisited at the start). It then starts at a random cell and begins DFS travesal to visit its neighboring cells. We are using the random function to see if the cells are blocked or not, with about 30 percent probability of being blocked and 70 percent being unblocked, if the cell is unblocked we continue to explore the grid.

Part 1

a. The first move of the agent is to move to the east rather than the north because the program is used to find the shortest path from the starting position to the ending postion. In Figure 8, the starting cell is E3 and the ending cell is E5. At the start the program looks at all the neighboring cells of E3 and sees that it can't go south because there is no cell there, and West and North are both blocked. Hence, the only option left is to go east.

b. Since there is a finite amount of grids, the agent is bound to reach its destination one way or another. In the worst case, the agent would have to travel the whole grid to reach its destination. Since, A* uses a close list, it will not go back to cells that it has already visited. To address the possibility of the goal being blocked, the program will terminate once all cells have been expanded. If the goal is completely surrounded by obstacles, the program will simply stop executing.

Part 2

The smaller g-value strategy expands more nodes near the start, leading to higher runtime and inefficiency. In contrast, the larger g-value strategy prioritizes deeper exploration, reducing unnecessary expansions and speeding up the search. Our smaller g-value strategy is about 10-15 times slower than the larger g-value.

Part 3

After comparing both algorithms, we found that their runtime and number of expanded nodes were fairly similar. Although the path length remained the same, Repeated Backward A* performed better, expanding fewer nodes and running faster than Repeated Forward A*. Specifically, it had a 10 percent reduction in expanded cells and a quicker overall runtime

Part 4

a. Manhattan distance measures the shortest path between two points on a grid using only horizontal and vertical movements. In grid-based navigation, the agent can move up, down, left, or right, making Manhattan distance the most efficient way to estimate the shortest route to the goal. Since diagonal movement is not allowed, the heuristic remains consistent and always finds the shortest path. However, if diagonal movement were permitted, the heuristic would overestimate the actual distance.

b. A heuristic h(x) is consistent if for every node x and evrey successor x‘ of x by an action a, the cost c of getting to the goal from x is no more than the step cost of getting to x‘ added by the cost of getting to the goal from x‘. So, if the action cost increases then we get a new cost c‘. To put this in math term,

$$h(x) \leq c(x, a, x') + h(x')$$

So

$$hnew(x) \leq hnew(x') + c(x, a, x') \leq c'(x, a, x')$$

Which shows the heuristic is consistent.

Part 5

Both of the algorithms are pretty similar and have almost the same run time. We found that Adapative A* might be a little bit faster compare to Repeated Forward A*. The amount of nodes were pretty much the same as well. Adapative A* was around 5 percent better than the Repeated Forward A*.

Part 6

To determine whether the performance differences between two search algorithms are systematic or due to sampling noise, we can use a statistical hypothesis test, such as the paired t-test or the Wilcoxon signed-rank test. We are going to use paired t-test to prove the answer in part 3. The null hypothesis would be that there is no significant difference in the number of expanded nodes between Repeated Forward A* and Repeated Backward A*; any observed difference is due to random variation, and then create a Alternative Hypothesis that states that Repeated Backward A* expands fewer nodes than Repeated Forward A*. Then we would collect the data that supports or oppose the hypothesis. Then you would compute the test statistic(t value), use the t value to compute the p-value. if the p value is less than 0.05 it would mean that the hypothesis is correct and Repeated Backward A* systematically outperforms Repeated Forward A*. On the other hand, if the p-value is greater than or equal to 0.05 then the observed difference is not statistically significant, meaning it could be random variation.