April Nguyen

CMPT 310 Assignment 1: Q6 Analysis

We define and refer to the following:

g(n) = path cost to n (backward cost)

h(n) = goal proximity from n using Manhattan distance (forward cost)

Cost: Step	Explored Count	Path Count
a. Greedy_Corners	79	72
b. A*_Corners	104	72

Q. In the second test, b, if you change cost function from Step to StepTurn, what do you expect to get for the counts? What did you get?

Changing the cost function from Step to StepTurn for (b) A*_Corners, I expect the Path Count value to remain the same since A* returns the optimal solution. I expect Explored Count to decrease because the StepTurn cost gives less weight to the heuristic than Step cost does (Step costs 1 unit per move in any direction, whereas StepTurn only has costs at turns, but moving forward costs nothing), increasing its similarity to the Greedy search, which has a lower Explored Count.

A reminder that A* uses f(n) = g(n) + h(n). Changing Step to StepTurn affects g(n), but not h(n). Step is expected to return a larger value than StepTurn for their g(n)'s. Since StepTurn returns a smaller g(n), there is more weight on the heuristic h(n), which makes A* search using StepTurn more similar to the Greedy search, which only puts weight on the heuristic h(n). I obtained the following results using StepTurn:

	Explored Count	Path Count
b. A*_Corners	79	72

which are the same results as the Greedy search. Also notice how the paths taken in Greedy and A* are the same with StepTurn.

Q. Interpret and explain the difference between a, b, and the switch of cost function.

A. Explanation for the difference between a and b

Cost: Step	Explored Count	Path Count
a. Greedy Corners:	79	72
b. A* Corners:	104	72

In this comparison between (a) and (b), we see that Greedy and A* search returned the same Path Count, but their Explored Count is different, with A* having explored more rooms than Greedy. In this implementation, we notice that the actual paths taken by Greedy and A* are different, but they are both optimal (more than one optimal solution in this problem).

Note that Greedy search only uses h(n) to find the path, whereas A* uses f(n) = g(n) + h(n) to find the path. A* performs an exhaustive search and so must use past knowledge to ensure optimality, whereas Greedy does not store nodes that have already been expanded into memory. This difference in calculation is why A* has an increased Explored Count (104 vs 79), as A* expands toward the goal but still hedges its bets to ensure optimality, and thus exploring more area

A. Explanation for the difference with the switch of cost function for the second test, b

Cost: Step Turn	Explored Count	Path Count
a. Greedy Corners:	79	72
b. A* Corners:	79	72

In this comparison we have changed the path cost to StepTurn. We see that Greedy search does not change for either explored count or path count, and running the program, the paths taken are the same. Again, Greedy only uses the heuristic h(n) to calculate its path. The heuristic function, the Manhattan distance, does not rely on the cost, StepTurn, and so we expect Greedy search to return the same path as when the cost was Step, which it does.

Running the program for A* Corners with StepTurn, we see that while the paths are different, they still result in the same path count (72), which is optimal, but the explored count has decreased (79 vs 104 with Step). A reminder that A* search uses f(n) = g(n) + h(n) to find the (optimal) path. Since we changed the path cost from Step to StepTurn, we have changed how g(n) is calculated, which affects f(n). Less weight has been put on the cost (Step > StepTurn), so the function f(n) = g(n) + h(n) has moved closer to what f(n) = h(n) looks like for the Greedy search. This change is the reason why the results have moved toward the same results as Greedy search.