Dylan Quach
u1223503
Erickson Nguyen
u1117938

3.2.1

1. Yes, the exploration path was what we expected. The algorithm would take one long path before coming back and exploring a different one. Pacman himself would only go the best path, rather than all the explored states because he found the first path through exploration, and then acted after the search was done
2. This is not the least cost solution, Pacman expands many nodes down a tree that may be far away from the proper solution. Using a different algorithm can get us a solution faster whereas DFS can find a solution that isn't optimal

3.2.2

1. BFS does return the least cost solution. It treads through each node in the search layer by layer, so that when it does hit the solution, it'll be the earliest one possible. This comes at the cost of a quickly growing frontier.

3.3

1. Priority Queue

3.4

1. For UCS, exploration extends from the start state in a diamond pattern, ever-expanding until it hits a goal state. It creates a zone that largely grows until it reaches the goal. A* acts differently, however. A* has a similar idea, in terms of it creating a radiating zone from the start to the finish. But, instead, it doesn't explore nodes that get farther away from the goal. It creates black patches where the algorithm never searches because ti doesn't have a good heuristic score.

3.5

1. For the corners problem, we had our state contain the position of Pacman at all times. But, it also kept track of visited corners in an array. This way, we can tell if we reached a goal state if we reached all possible corners.

3.6

1. We would figure out the closest unvisited corner to Pacman, and then give the distance to that corner from a state as a heuristic cost. This way, Pacman would go to the closest unvisited corner, and then would follow the walls to the next corner ideally.

3.7

1. For eating all the dots, we would calculate the manhattan distance for all of the dots and choose the dot that was closest to Pacman. He would then chase the shortest distance and eats dots in that order.

3.8

1. It’s a greedy algorithm. It tracks down the nearest dot based on distance but has no account for walls or how to get there. It might head to the closest dot, but it might be faster to head to a different dot that’s farther in distance but requires no detours to reach.

DYLAN QUACH Self Analysis

1. I think the hardest part of the assignment was developing UCS and A* algorithms. We had to get some help because we ended up using a format from algorithms. This caused us data issues and we had to rewrite them in the end.
2. I think the easiest part was BFS. We just had to finish DFS and then we copy and pasted with a queue. So that wasn’t too challenging.
3. I think doing the corners problem really helped me understand how AI works. The way you set up variables and goal states opened up what the search agents were working with and how they really kept track of the situation. We had issues at first, but the more we put it together, the more the rest of the assignment became clear for us.
4. I think the majority of the assignment was pretty useful, I’d be happy to do it again and I feel like each part was helpful. Plus, it was fun to write code to make Pacman move around.
5. If I had only one critique, I think a lot of the TA’s use reliance on the pseudo-code that we had in class. When my partner and I tried a different implementation of Djikstra’s for UCS, the TAs asked us to start again based on the pseudo-code despite it working previously when we had an issue with data types. It was still helpful, but definitely caused some grief when we tried to explain the situation.

ERICKSON NGUYEN Self Analysis

1. THe hardest parts of the assignment for me was probably A* and heuristics. Dylan and I had to get some help with the A* since we implemented it a weird way. However, personally for me, the heuristics were much more difficult. I have trouble with formulating any efficient program without needing an ample amount of time alongside trial and error.
2. BFS was the easiest by far. After implementing DFS, it was simply swapping out the data structure used from a stack to a queue and it worked immediately.
3. I found implementing the UCS and A* the most fruitful in teaching me about the overall course material. While DFS and BFS were useful, it was more so a refresher to search algorithms. The heuristics problems helped quite a bit too, but I found myself struggling with that much more.
4. I don’t think so. While the problems were difficult, it was important to do them since it helped me learn the material much better. I wouldn’t regard any of the problems as tedious or useless.
5. I think if this assignment allowed more room for creative solutions I would’ve liked that. For example, our weird solution for A* when shown to a TA was immediately shot down and we were strongly encouraged to follow the pseudo-code. While I have no large issue

with that, I personally think being able to go about the problem any which way we would like and learn through that is what I would’ve preferred for this assignment.