Ryan Brooks
Prof. Kuntz
CS 4300
15, February 2021

Project 1

3.1:

1. **Is the exploration order what you would have expected? Does Pacman actually go to all the explored squares on his way to the goal?**
   Pacman does not visit all the explored squares on the way to the goal. Explored nodes simply signify the algorithm checking a specific direction for the goal. Consequently, the found path is often not going to use all of the explored nodes due to the non-trivial nature of the problems being solved. This is what I expected as we previewed visualizations of path-finding algorithms in class previously. Also, as expected DFS did not find an optimal path, instead taking a winding path through the mazes.

2. **Is this a least cost solution? If not, think about what depth-first search is doing wrong.**
   This is not a least cost solution under any normal definition of cost. DFS essentially follows a path as deep as possible before checking other earlier branches. Consequently, the path it finds will almost always take unnecessary steps to reach the goal. This is ultimately because making sequential arbitrary decisions does not guarantee that the goal will be reached by following a least cost path.

3.2:

2. **Does BFS find a least cost solution? If so explain why?**
   Yes, in situations where cost is purely defined by the length of the path BFS does find a least cost solution. This is because BFS explores in each direction equally, expanding until it reaches the goal. Consequently, at any stage being on the fringe means that the recorded path is the least cost path to that state. Thus, when the goal is found this will be the least cost path to the goal under this definition of cost.

3.3:

2. **Specify the data structure used from the util.py for the uniform cost search?**
   I used a priority queue as the fringe for uniform cost search. The priority given to each state pushed onto the queue was equal to the cost of the path to reach that state. This was given by the getCostOfActions() function.

3.4:

2. **What happens on openMaze for the various search strategies? Describe your answer in terms of the solution you get for A* and Uniform cost search.**
   A*, UCS, and BFS act the same for the openMaze as all spaces are weighted the same in this maze. First, they all follow a path down until they are even with the top of the rightmost inner wall. Then, they go left and turn around the wall and go straight down until they hit the bottom of the maze. Once Pacman is at the bottom of the maze it goes straight left until it reaches the goal.

DFS acts differently from the rest winding back and forth across a large portion of the maze, only going downward when it hits a wall. It does this until it reaches the goal. Consequently, this is far from a least cost solution.

3.5:

2. **Describe in few words / lines the state representation you chose or how you solved the problem of finding all corners?**
   To represent the state, I used a tuple of six items. Firstly, I kept the integers X and Y to refer to the position of Pacman in the given state. I added to this four Boolean values referring to whether or not each corner has been visited. To update these values, I analyzed whether or not the agent could enter a corner in the successor function. If a corner could be entered, this state would set that corners corresponding Boolean value to true in the successive state to represent this non-positional change.

3.6:

2. **Describe the heuristic you had used for the implementation.**
   The heuristic I used for the corners problem was extremely simple. I returned the Manhattan distance to the furthest unvisited corner or zero if there were no remaining corners. I came upon this idea by simplifying the problem, only requiring Pacman to reach the furthest corner and allowing him to go through any walls in the maze. As a simplification of the actual problem this gives credence to the notion that this heuristic is admissible and by extension likely consistent.

3.7:

2. **Describe the heuristic you used for the FoodSearchProblem.**
   The heuristic I used for the food search problem was quite similar to that of the corners problem. After getting a list of all of the positions of food remaining in the maze, I returned the Manhattan distance to the furthest piece of food. This was obviously inspired by my heuristic for the corners problem. It is also stems from a simplification of the base problem as it only requires Pacman to get the furthest piece of food and allows him to cut through any walls in his way.

3.8:

2. **Explain why the ClosestDotAgent won't always find the shortest possible path through the maze.**
   By always going for the closest dot in every situation, this agent avoids making intelligent choices that prevent backtracking. Consequently, this causes the agent to rarely find the shortest possible path. For example, if the agent was going down a straight hallway filled with dots and a single dot positioned to the right of the middle of the hallway, the following situation can occur. Pacman will first follow the hallway eating dots until he reaches the intersection. At the intersection the agent may choose to continue down the hallway instead of eating the lone dot to the right. Then at the end of the hallway, Pacman will have eaten the farthest dot from the start and realize that the new closest dot is back down the hallway. In summary, this oversight by the agent is caused by an inability to make choices based on "looking ahead". Instead, the agent only has the immediate information of the closest dot as a decision making tool.

4:

1. **What was the hardest part of the assignment for you?**
   The hardest part of the assignment was coming up with a good heuristic for question 7. I tried for a few hours to come up with an efficient, admissible, consistent heuristic but was not able to reach full credit in terms of minimizing expanded states.
2. **What was the easiest part of the assignment for you?**
   The easiest part of this assignment was implementing the rest of the search algorithms once I had completed DFS. For A*, BFS, and UCS it was as simple as changing up the data structures and method of
3. **What problem(s) helped you further your understanding of the course material?**
   The questions where I had to implement my own heuristics were extremely helpful. My original heuristic idea was not admissible or consistent. Re-evaluating my understanding of these terms and how they correlated to the assignment in coming up with a better heuristic clarified a lot of the confusion for me.
4. **Did you feel any problems were tedious and not helpful to your understanding of the material?**
   No, everything in this homework improved my understanding of the material. No parts were particularly tedious.
5. **What other feedback do you have about this homework?**
   I think that the explanation of how states were expressed in the code itself were not explained very well. It was discussed later in class, but I think clarifying the specifics of how states are implemented as well as how to initially interpret them could have saved me a lot of wasted time in the early stages of the assignment.