Aaron Templeton
U0734119
CS 4300 - Project 1

3.1 Depth First Search

1. Code Implementation
2. Yes, the exploration order is what I expected. No he does not go to all the explored squares on his way to the goal. He takes the “left-most” path in the depth first search.
3. No, it is not a least cost solution. Sometimes the path is longer than needed. It is going all the way down the tree first, which is not necessarily the best or least cost.

3.2 Breadth First Search

1. Code Implementation
2. Yes, in the sense of how many actions for pacman to reach the food. BFS does not traverse the entire tree. It reaches the solution without seeing all the nodes

3.3 Uniform Cost Search

1. Code Implementation
2. Priority Queue data structure was used from util.py

3.4 A* Search

1. Code Implementation
2. UCS and A* traverse the maze in the exact same path. Down first and then left. If you use the manhattan heuristic for A* it searches less of the maze than UCS, if you do not use the heuristic, it is the same as UCS.

3.5 Finding All Corners

1. Code Implementation
2. Get the position of pacman and the position x,y of the directions to the corners. For all the actions possible, make sure the next action is not a wall (illegal move) then and the position to the successors. Based off of the sample code provided in the comment

3.6 Corners Problem Heuristic

1. Code Implementation
2. I used the absolute difference to find the distance from pacman to the corners

3.7 Eating All Dots

1. Code Implementation
2. Iterate over all the food and get the position of the food to pacman. Take the sum of the max x - min x + max y - min y which gives the bounding rectangle of the food (convex hull) relative to pacman

3.8 Suboptimal Search

1. Code implementation
2. It doesn't look for the shortest path first. It looks for the closest food first and foregoes the shortest path or most optimal path to find the closest food instead. It may miss a close food for a closer food option and it won't go back to the other food for a long time and may have been more optimal to get both and then continue.

4. Self Analysis

1. The hardest part was finding the heuristic for the corners problem and eating all the dots
2. The easiest part was breadth first search and depth first search algorithms
3. Building a heuristic to understand what it really is and how it is used
4. I think all of the problems were helpful to understand further the material. Nothing was extremely tedious
5. No other feedback