**CS205 - PacMan Week 7 README**

Gayatri Bhosale : 862394365

Sanjana Senthilkumar: 862384460

Puneet Singhania : 862327375

**Question 7: Eating All The Dots:**

For this problem, we utilize the code for the A\* star search that we used to answer question 4 from the previous weeks. We use the foodHeuristic function. This gives an estimate of the cost it will take to start from the current position and reach the goal where all food pellets have been eaten. The goal is to estimate a path where less than 7000 nodes are expanded to reach the goal.

We approached this problem by testing the Manhattan distance function that was written previously but it did not work here. We believe the reason for this is because Manhattan distance assumes direct paths with no obstacles. Pacman has many walls that it probably doesn’t account for and as a result, expands more nodes than strictly necessary. We tried using Euclidean distance but this did not work as well. This takes into account straight line distances which can include diagonal movement that isn’t typically allowed in Pacman and give incorrect paths.

So, we found out that mazeDistance() is the most optimal as this takes into account the maze-like nature of the problem (i.e.) it considers the constraints of the maze itself, like walls and obstacles. This helps improve the performance of A\* as it expands more nodes that will be more likely to take us to the goal, and it won’t expand unnecessary nodes.

Also, our A\* with a null heuristic (equivalent to uniform-cost search) quickly finds an optimal solution to testSearch with no code change on our part, with a total cost of 7. Below is the screenshot:

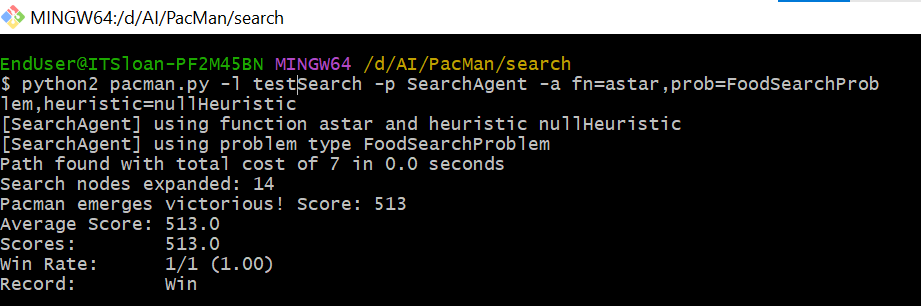
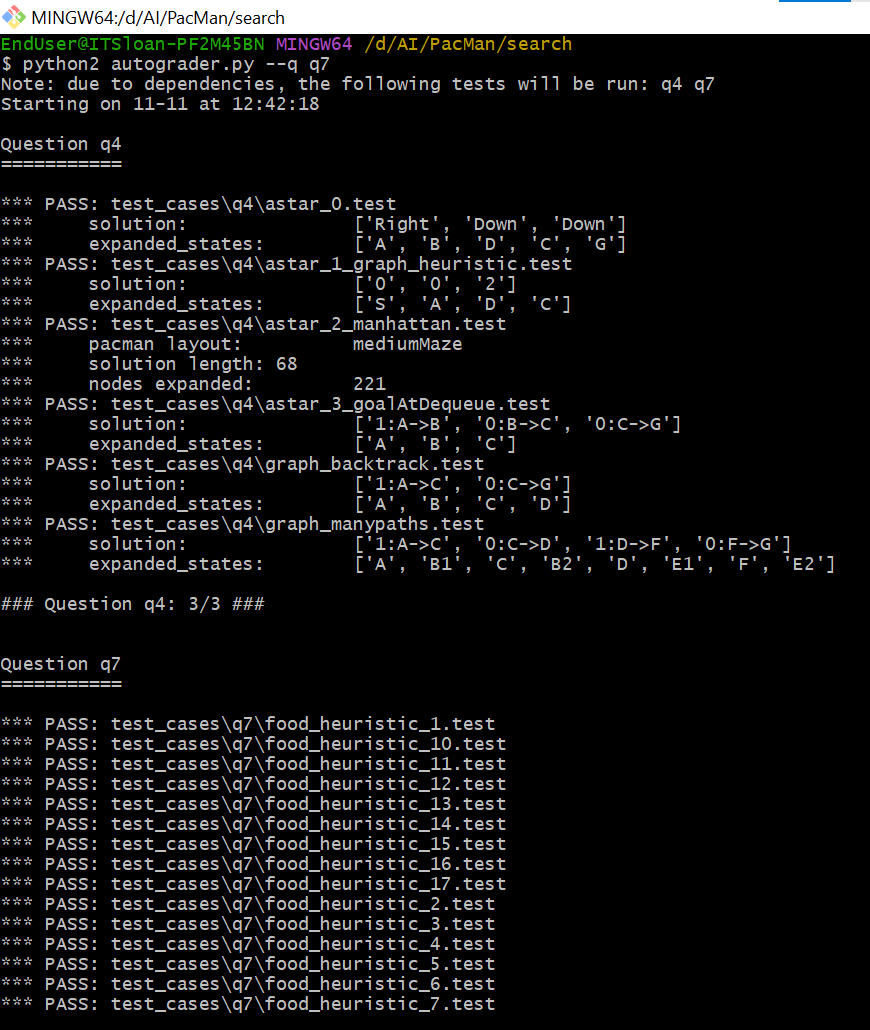


Figure 1: PacMan command for Question 7 with nullHeuristic

Our solution defined a path that collects all of the food in the Pacman world. We have not taken into account any ghosts or power pellets and we only depend on the placement of walls, regular food and Pacman.

**AUTOGRADER RESULTS:**



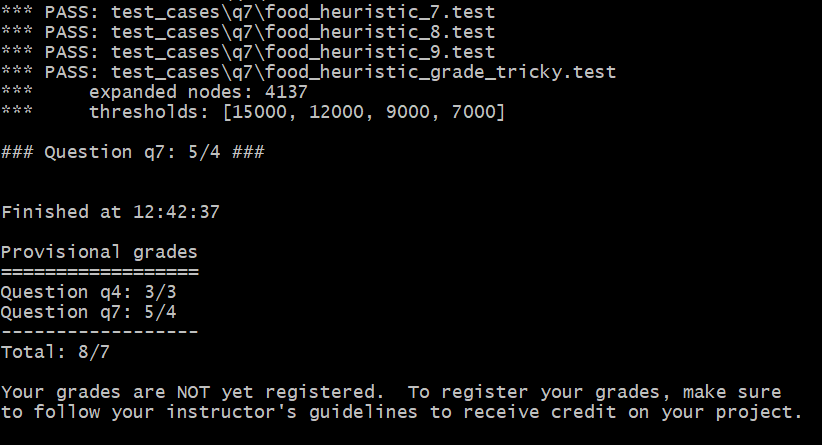


Figure 2: Autograder for Q7

**SOLUTION FOR THE PACMAN COMMANDS PROVIDED:**

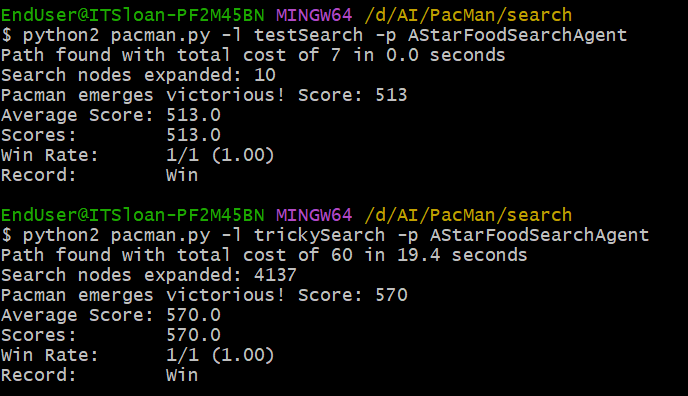


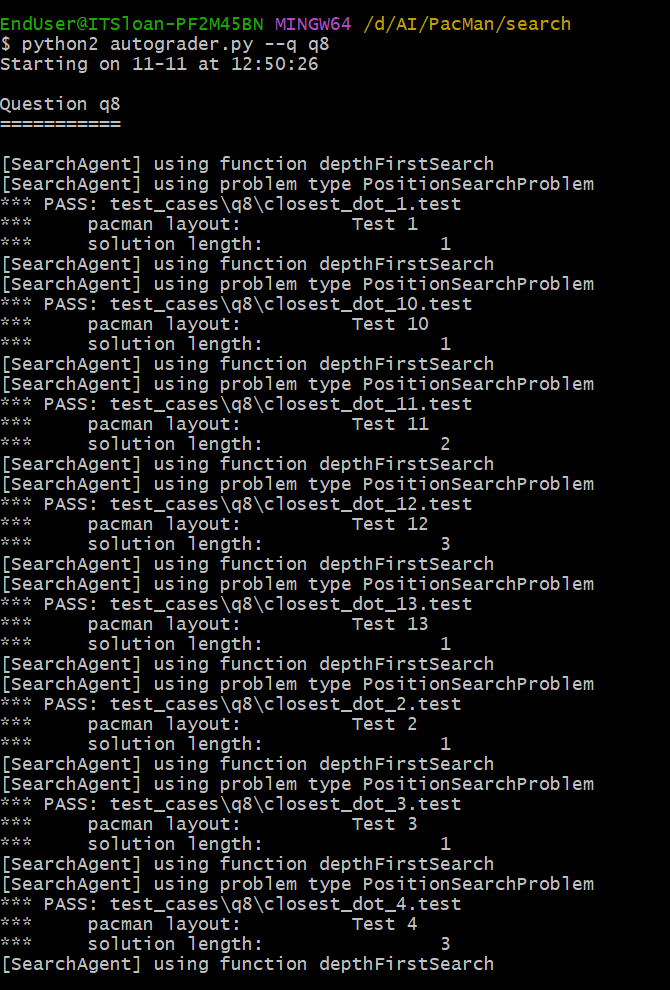
Figure 3: PacMan commands for Question 7

**Question 8: Suboptimal Search:**

We wrote an agent that always greedily eats the closest dot and this implementation is in ClosestDotSearchAgent in findPathToClosestDot(). The agent found the path in less than 1 second with a path cost of 350. The screenshot is attached below (after autograder results).

The reason the most optimal path isn’t always selected is because of the greedy nature of the solution. It will prioritize short-term gains in favor of optimality. We think Astar search expands lesser nodes than this greedy approach.

**AUTOGRADER RESULTS:**



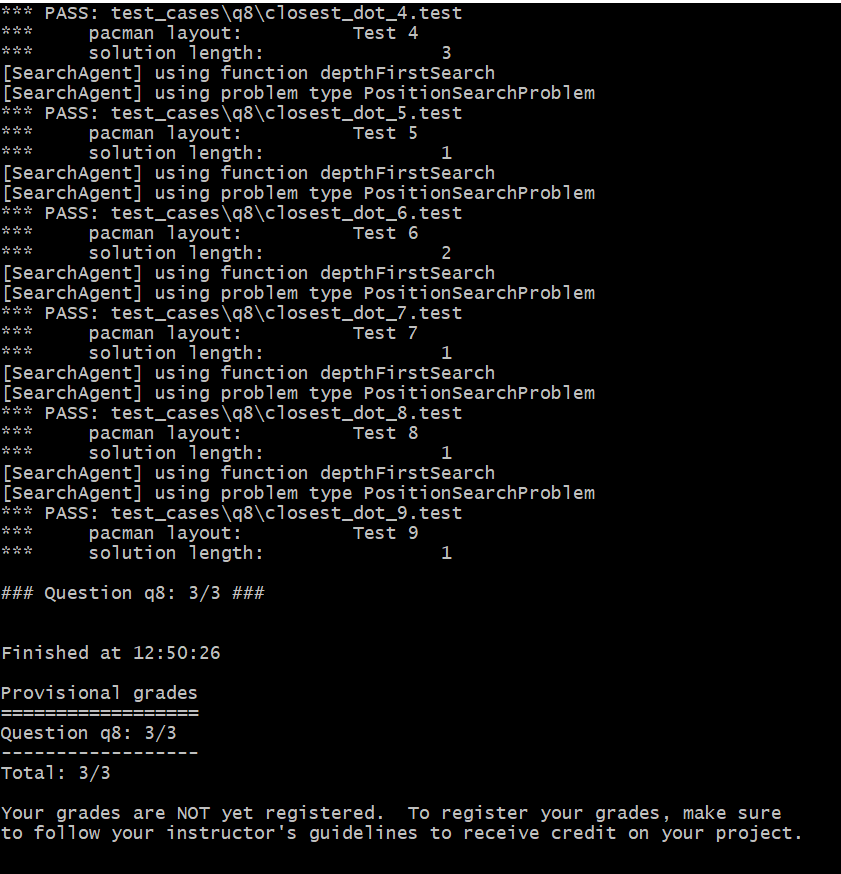


Figure 4: Autograder results for Q8

**SOLUTION FOR THE COMMANDS PROVIDED:**

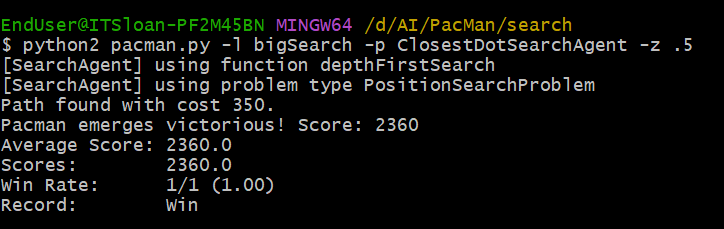
****

Figure 5: PacMan commands for Question 8

**Learnings:**

This week we learned that A\* works very well with a heuristic if it is consistent and designing such a heuristic can be tough. Also, ghosts increase the trouble and we also need to consider them. Greedy solutions are not always optimal! We think BFS is better than DFS because it is not as greedy as DFS and results in an optimal solution sometimes (at least for this Pacman).