**CS205 - PacMan Week 5 README**

[Gayatri Bhosale](mailto:bhosale.gayatri@gmail.com) : 862394365

[Sanjana Senthilkumar](mailto:sanjana.senthilkumar@email.ucr.edu): 862384460

Puneet Singhania : 862327375

**Question 1: Finding a Fixed Food Dot using Depth First Search:**

**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 doesn’t go to all the explored squares, which is what we expected based on the algorithms we utilized here. When using DFS, it’ll keep going along a path until it hits an end and then it will backtrack. However, there’s no guarantee that this will happen for all paths.

Once Pacman has reached its goal, it will not visit the rest of the already explored squares.

**2. Is this a least-cost solution? If not, think about what depth-first search is doing wrong.**

No, DFS does not give the least-cost solution. The reason for this is that, when DFS is finding the path it does take into consideration the cost and length of path. What DFS does wrong is that it only takes into account the successors in the path. So there could be a path that is less costly or shorter in length and DFS might not consider it.

For the depth-first search**,** we employ a stack that will take a starting node and perform addition of all of its adjacent nodes onto the stack**.** We utilize it to monitor the explored and yet to be explored nodes. As each node is popped from the stack, it will visit the node. You keep track of the visited nodes so you don’t visit them twice. We know that until the goal is accomplished, Pacman must travel in all directions.

**SOLUTION FOR THE PACMAN COMMANDS PROVIDED:**

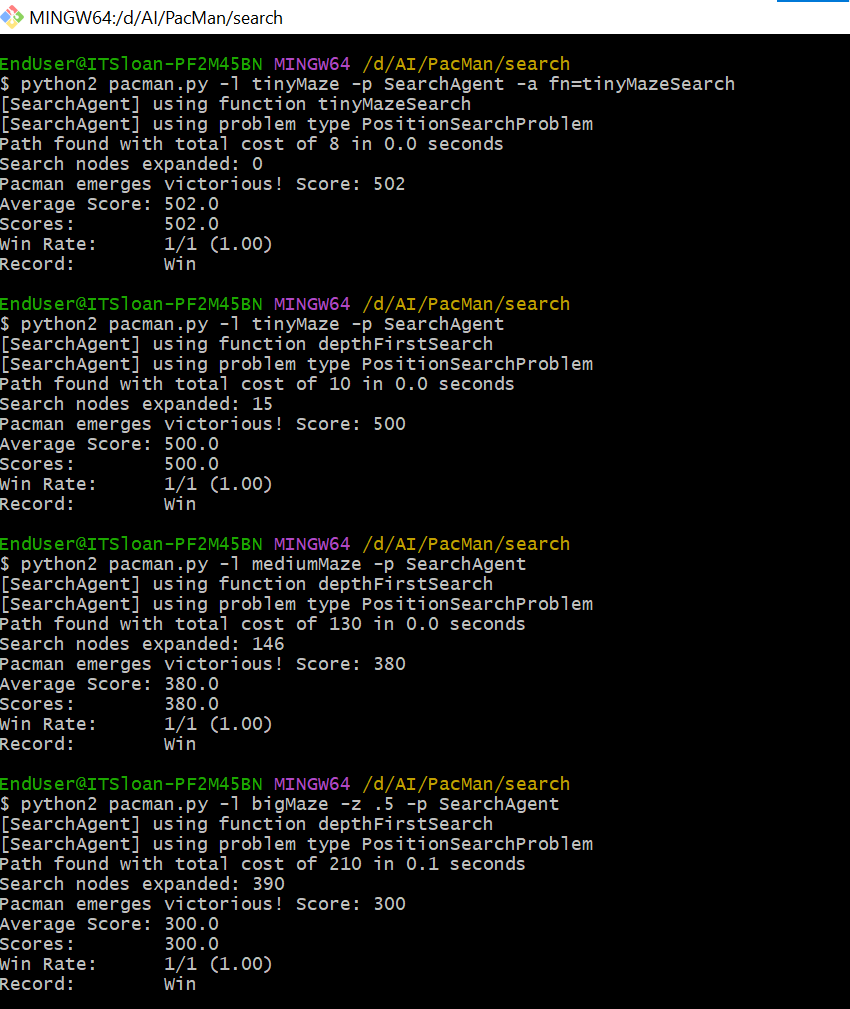


Figure 1: PacMan commands

**AUTOGRADER RESULTS:**

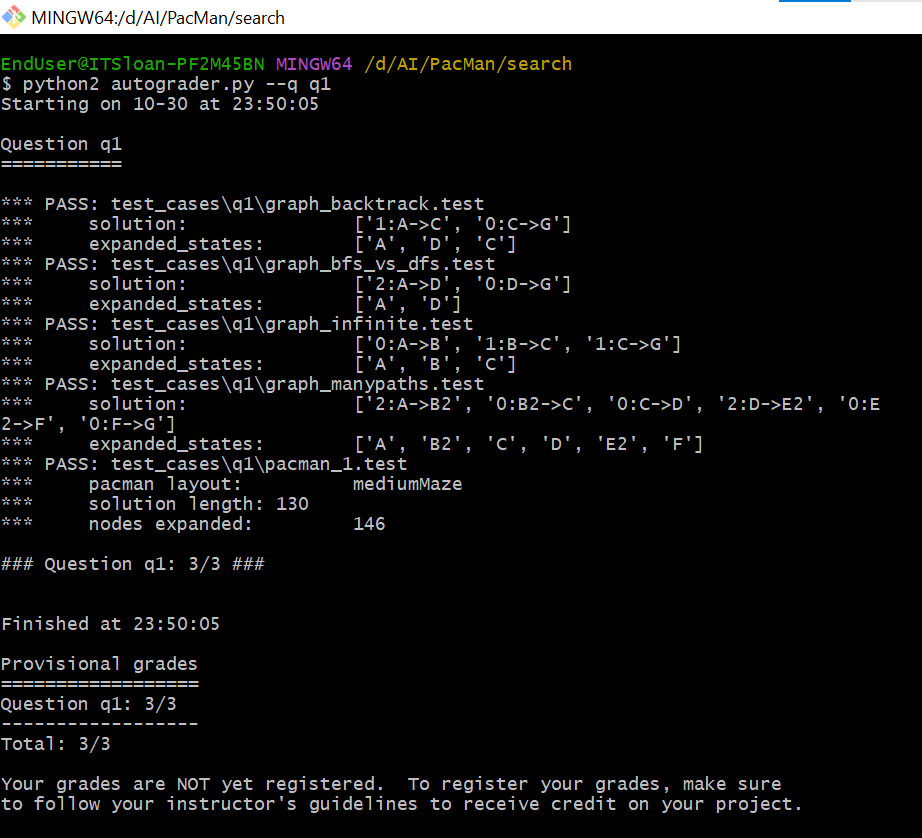


Figure 2: Autograder for Q1

**SOLUTION FOR MORE COMMANDS SPECIFIC TO DFS:**

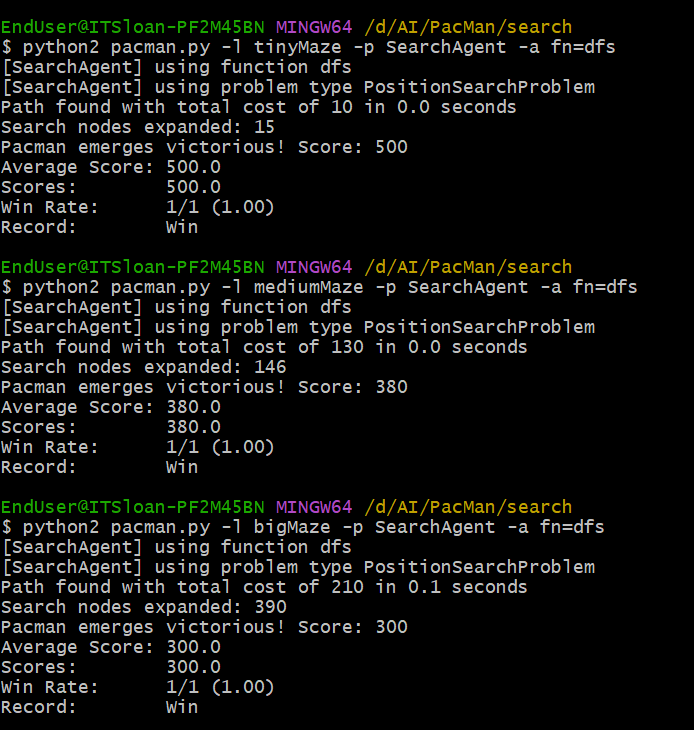


Figure 3: PacMan commands for DFS

**Question 2: Finding a Fixed Food Dot using Breadth First Search:**

**Does BFS find a least cost solution? If not, check your implementation.**

BFS utilizes a queue instead of a stack and finds the shortest path. BFS works by searching at the breadth layers. So, yes, BFS does give a least cost solution.

**AUTOGRADER RESULTS:**

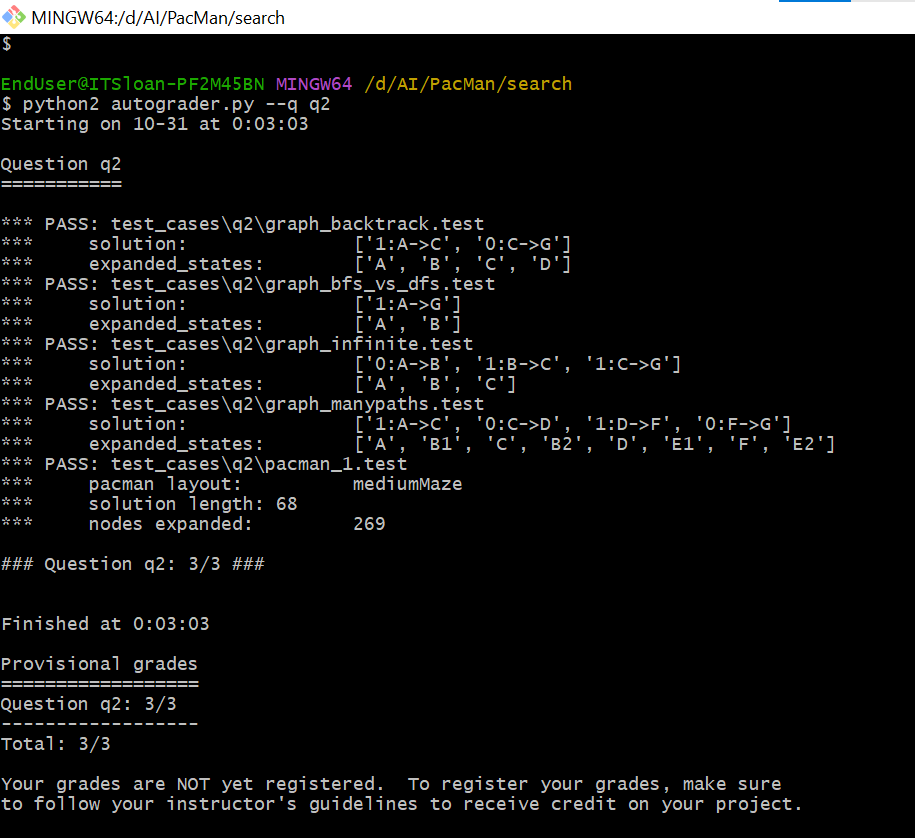
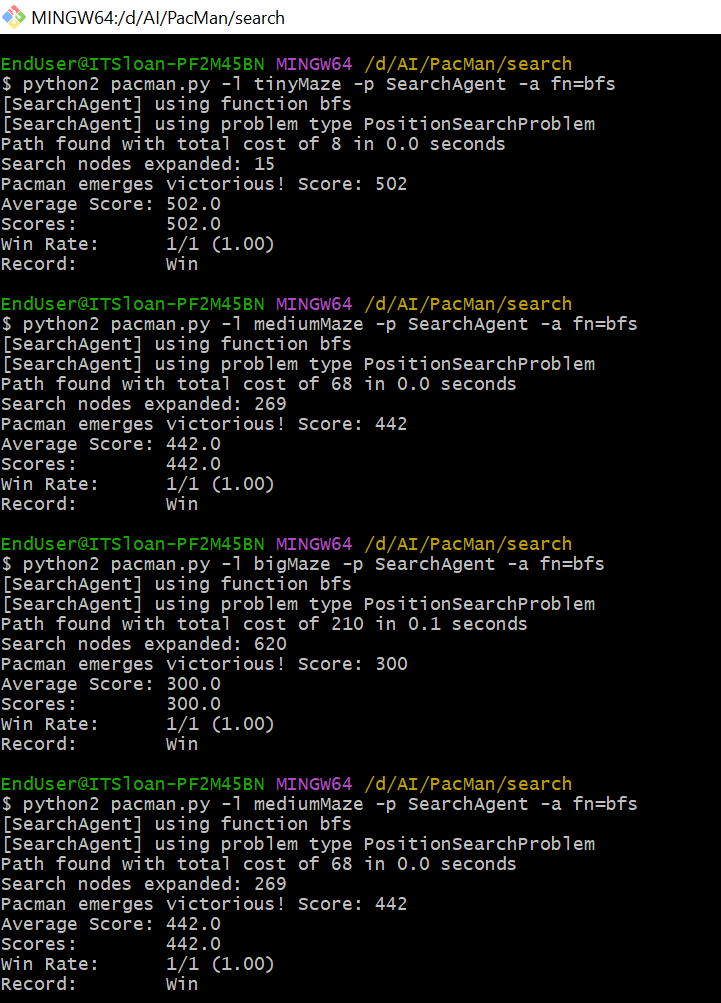
****

Figure 4: Autograder results for Q2

**SOLUTION FOR THE COMMANDS PROVIDED:**

****

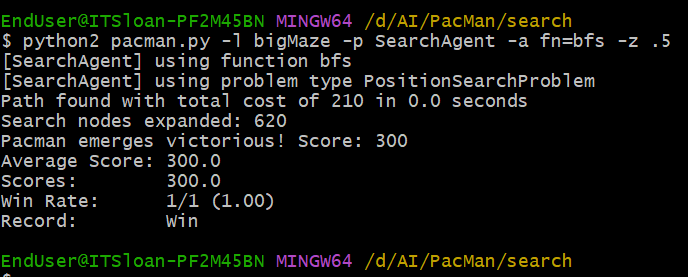
****

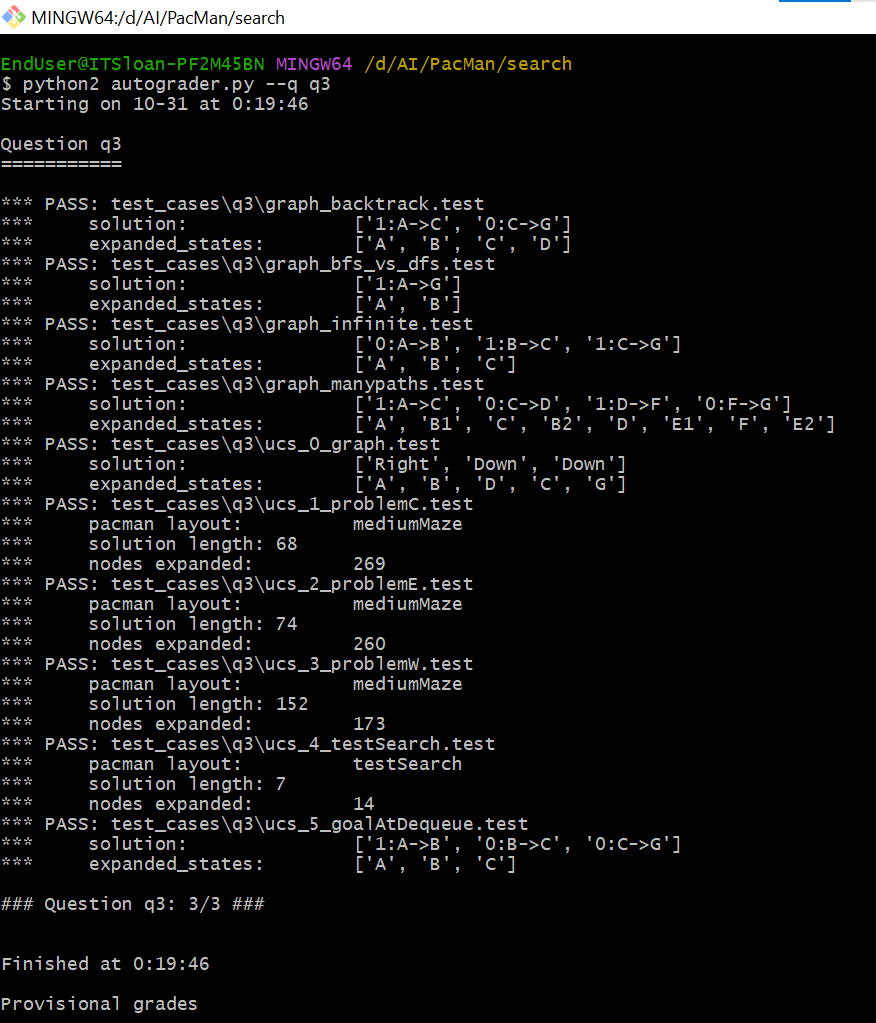
Figure 5: PacMan commands for BFS

**Question 3: Varying the Cost Function:**

**Uniform Cost Search:**

This search works utilizing the PriorityQueue data structure. It takes into account the cost and helps find the path that costs the least. It calculates the cumulative cost it would take from the chosen node to the goal and takes the path that will incur the least cost. The nodes along the path are chosen such that they have the least cost among the existing options. The lowest costing nodes are given priority.

**AUTOGRADER RESULTS:**

****

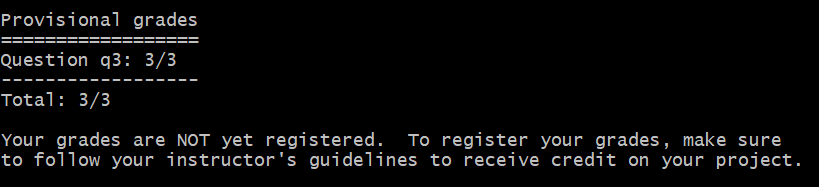
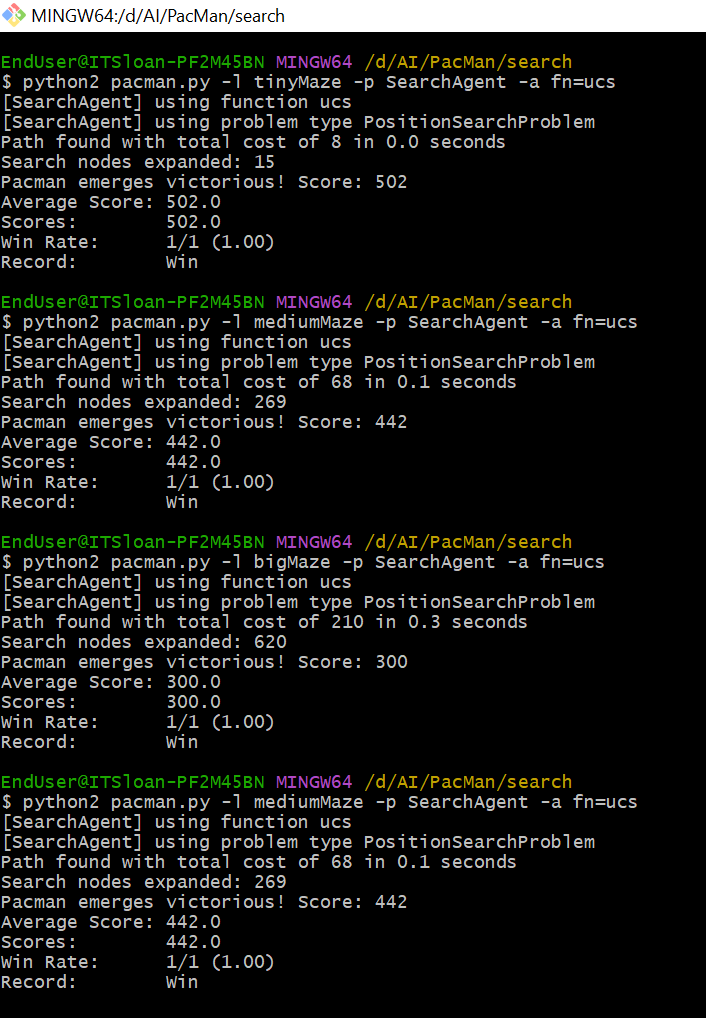
****

Figure 6: Autograder Results for Q3

**SOLUTION FOR THE COMMANDS PROVIDED:**

****

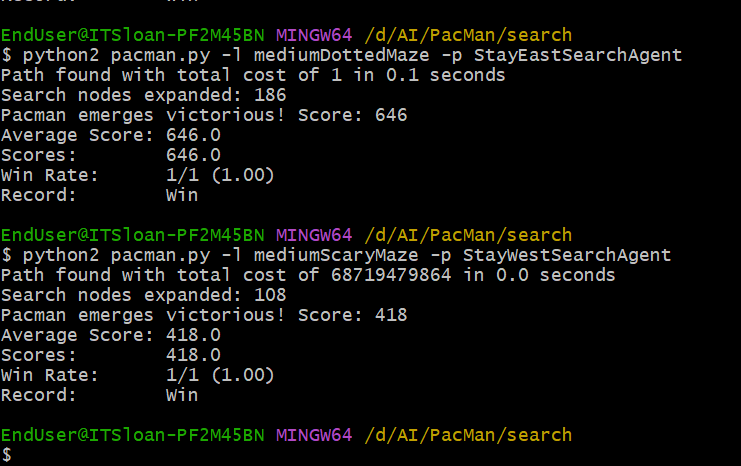
****

Figure 7: PacMan commands for UCS

We can observe in the above screenshots of the Pacman execution that we get the total path cost as 1(very low) for the StayEastSearchAgent and the total path cost as 68719479864 (very high) for the StayWestSearchAgent.

**Question 4: A\* SEARCH:**

This algorithm is dependent upon two factors: the cost it has incurred from moving from the original node to the current node and a heuristic or an estimated cost that it would take to move from the current node to the goal (which you won’t know until you actually reach the goal so you estimate). Each time it will check the sum of these both and move to the node that has the least value.

**1. What happens on openMaze for the various search strategies?**

We tried openMaze (attached the screenshots after autograder results) for all the search strategies and found out that many more nodes are expanded by DFS than we had anticipated in comparison to the other methods. The score for DFS was lesser than the other strategies. On the other hand, BFS, UCS and A\* perform similarly.

**AUTOGRADER RESULTS:**

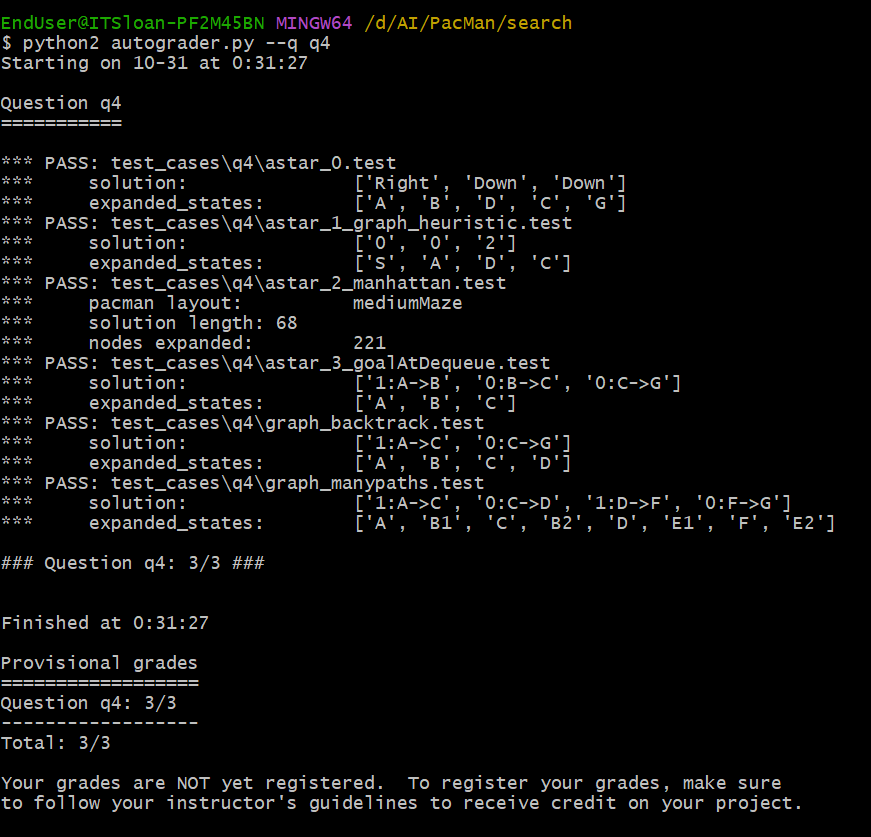
****

Figure 8: Autograder results for Q4

**OPENMAZE FOR VARIOUS SEARCH STRATEGIES:**

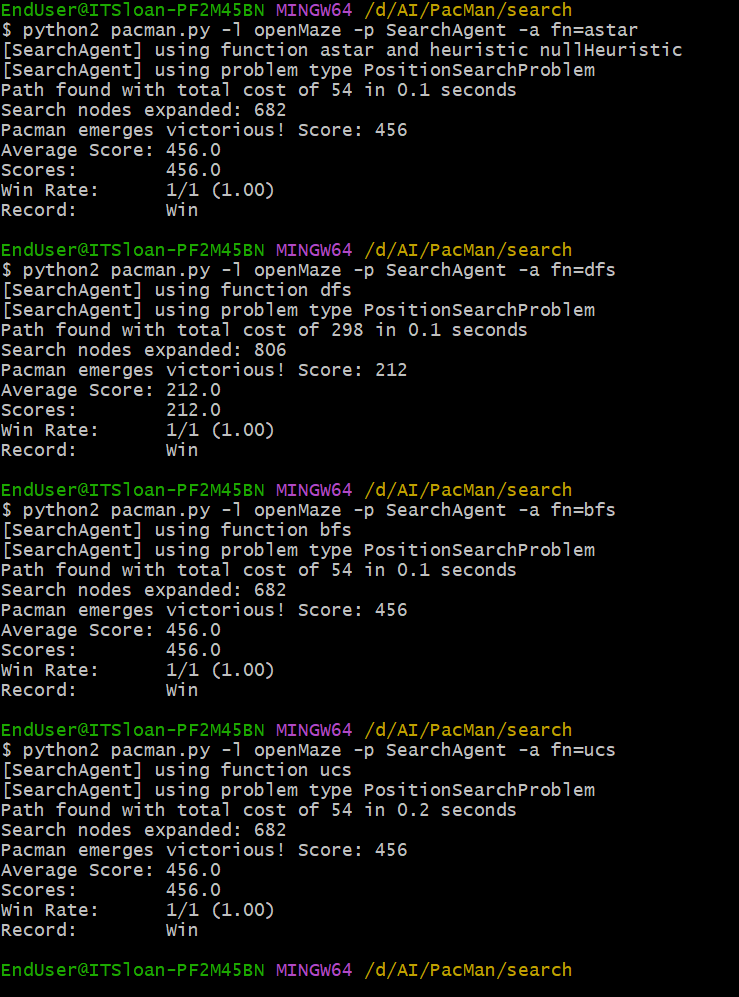
****

Figure 9: openMaze outputs

**SOLUTION FOR THE COMMANDS PROVIDED:**

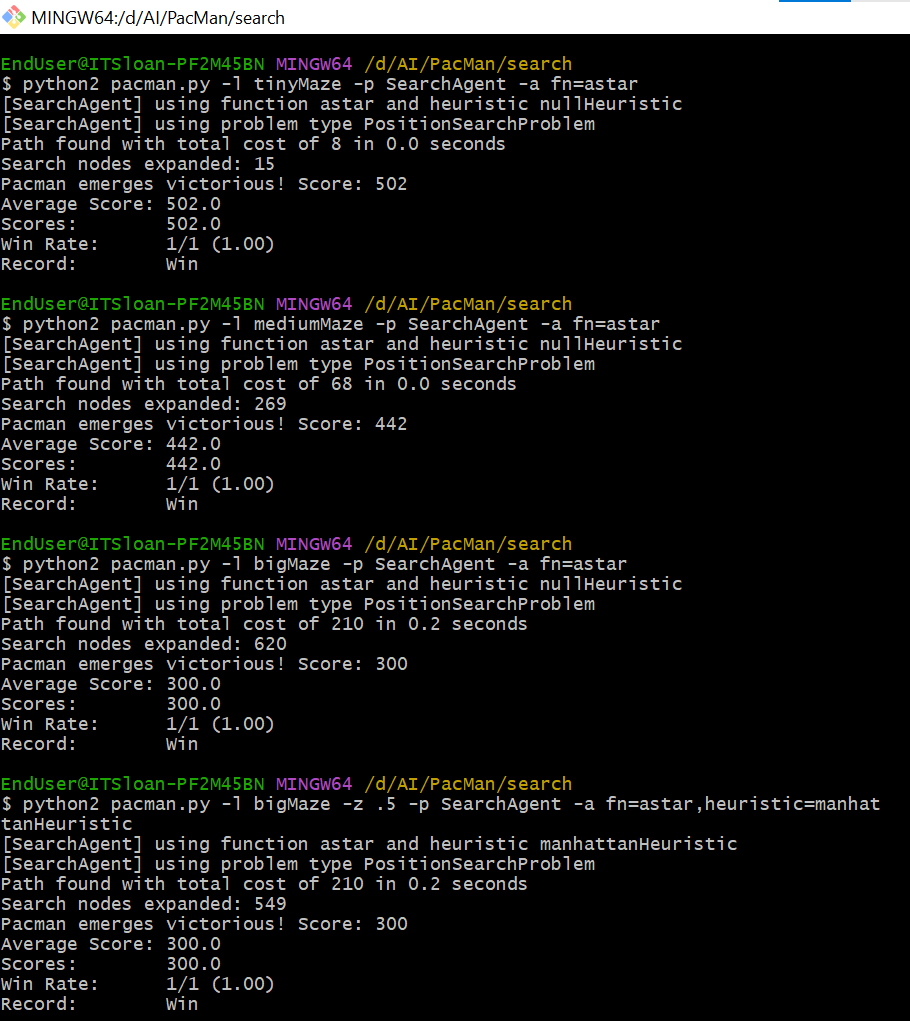
****

Figure 10: PacMan commands for A\*

Learnings:

We learned different algorithms - DFS, BFS, A\*, and UCS and their implementations in Week 5. The main difference between these algorithms is the way of traversing the nodes, using different data structures, and consideration of the cost factor.