Report of Project 1 of Artificial Intelligent

Cheng Zhong 16307110259

Environment: Python 3.7 in Windows 10

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

When using depth-first search to search the goals, we use stack as the data structure and store the information of current node and the motion from the root towards it into the stack. Then we turn to the successor of this node until we find the goal of the test. The result as follows:

	Tiny Maze	Medium Maze	Big Maze
Expand nodes	15	146	390

It's worth mentioning that when we test with the medium maze, every time the pacman encountering different intersections, it will follow the order of search direction set in advance. For example, in Medium maze, the pacman will move to the left in first intersection, so in the next intersection it will search for some useless paths (the path will return to the origin road).

Question 2: Breadth First Search

The structure of BFS is similar to DFS, expect that we replace the date structure to queue. So that we can search all successor instead of choosing one road. We also test the mazes in different sizes. The result as follows:

	Medium size	Big Maze
Expand Nodes	267	617

In comparison, BFS will expand more nodes than DFS, but it can also find the best path, it can also solver the eight-puzzle problem.

Question 3: Varying the Cost Function

In this question, we use priority queue as the data structure. By pushing the cost of reaching node into the queue, we can preferentially extend the node with the lowest cost. The result as follows:

	Medium Maze	Medium Dotted maze	Medium scary maze
Nodes	269	186	108
Costs	68	646	418

As can be seen from the path, Pacman will prefer a path with food and avoid the path with monsters.

Question 4: A* search

In A* search, we use the prior queue as the data structure and takes a heuristic function as an argument. When we find a path through the maze to a fixed position using the Manhattan distance heuristic. We can find the optimal solution slightly faster than uniform cost search. From its performance in openMaze, we can find that this algorithm will search for nodes closer to the target.

Question 5: Finding All the Corners

This problem is actually another form of search problem. The difference is that

the parameters of the pushing into data structure change from the position of the node to the position of the node and the corner position that has been traversed.

	Tiny corners	Medium Corners
nodes	410	2381

Question 6: Corners Problem: Heuristic

In this question, algorithms using Manhattan distance and the Euclidean distance need to expand more than 1200 nodes. So, I finally chose distance as a function:

1) From the position, go to the unexpanded corner we can reach the fastest.
2) From the corner we get, go to the next fastest reachable unexpanded corner.
3) Repeat 2), until we have reached all the corners. Calculate the total distance.

During the procession, we ignore the impact of the wall on our movement, so the function is admissible. After testing, we can only expand 902 nodes to get the optimal solution, which is better than the function we mention above.

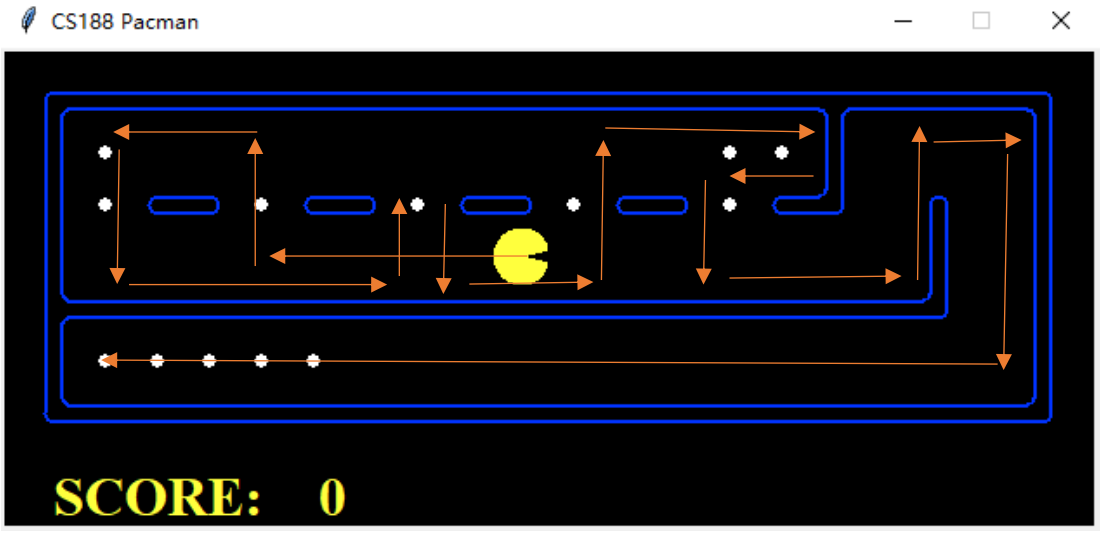
Question 7: Eating All the Dots

In this question, our UCS agent finds the optimal solution in about 13 seconds, exploring over 16,000 nodes. In my solution, I assume that there is only food that is farthest from the current location and ignore the other foods, and calculate the distance between them using the BFS method in question 2. Obviously, the function is admissible. Using this heuristic function, we can get the optimal solution in 14.0 seconds expanding 4137 nodes.

But it seems awkward when solving the medium search problem, because there is too much food in this problem. Every time you expand the node, you have to traverse all the food's location, this greatly increases the time complexity.

Question 8: Suboptimal Search

In this question, we only need to compare the distance of each food to the current position and compare it. From Pacman's trajectory, we found that it took lots of repetitive roads, in the tricky search problem, the optimal solution is in figure 3.



But when we using the greedy algorithm, the trajectory turns to suboptimal one: