COMP 7404 Computational Intelligence and Machine Learning

3035348102 ZHANG Yupeng

Question 1:

DFS: In this question, we know that the frontier should be last in first out, and I also read the *util.py* file, therefore, I decided to use *Stack* data structure to store the states. And this is a graph search algorithm, I also add an *explored set* to store the states which visited before the current state.

But we should know that DFS is not an optimal solution. This algorithm just goes through the nodes and layer by layer, if there is a solution, it will return the solution right now. Therefore, DFS just returns the first solution which it could find. Of course, it is not guaranteed to be an optimal one. That's why in the medium size maze, it is not a least cost solution.

Question 2:

BFS: This question is quite similar with the previous one, what I changed is only the data structure from *Stack* to *Queue*.

For BFS, it will return the least node solution. When you don't define the cost of every two nodes, of course, it will return an optimal solution.

Question 3:

UCS: This question is quite similar with the previous one. After I read the *util.py* file, I decided to use the *priority queue* data structure as the frontier. And I also define the cost.

Question 4:

A star: This question is quite similar with the previous one. The only different one is that I plus a *heuristic function*.

When A star and UCS deal with the open maze problem, they return 535 and 682 nodes expanded respectively. Although they both return the optimal solution, *heuristic function* introduced could speed up the process.

4.1:

Cost / Expanded	DFS	BFS	UCS	A* + ManH	A* + EucH
tinyMaze	10/15	8/15	8/15	8/14	8/13
mediumMaze	130/146	68/269	68/269	68/221	68/226

bigMaze	210/390	210/620	210/620	210/549	210/557

Question 5:

Representation for corners problem: I add a *FLAG* in this question, when the corners become expanded, which means the pacman eats the food dot, I change the value to *True*. And then, I return the successor which includes the coordinates, corner values, action and cost.

Question 6:

Heuristic for corners problem: I define a *Manhattan distance* list in this question, which stores the distance between corners and pacman, and return the greatest one in the list as the *heuristic function*.

This heuristic method will return 1136 nodes expanded, which is less than 1200, so I think it is not bad one.

6.1: Q1: ==; Q2: non-trivial; Q3: admissible; Q4: >; Q5: consistent

Question 7:

Food heuristic: First, I immediately want to use the same *Manhattan distance* to calculate the distance as the *heuristic function*, but it will return 9551 nodes expanded, which beyond 9000 nodes. So, I decided to use *MST* to solve this problem. This new *heuristic function* will return the distance between every food dots and the pacman as the heuristic. Finally, it will return 7547 nodes expanded, which are smaller than 9000.

Question 8:

8.1: I just search the square and judge whether current position is 1 to get the coordinates of queens, and then I compare the coordinates of 8 queens, if the previous one can attack the next one, number of attacks will add one,

8.2: First I calculate the cost of current board and find the position of smallest cost, and use the y coordinates to find the position of queen in current column. And then I change the value of previous position to 1 and the value of current queen to 0, which means I move the queen to the smallest position.

8.3: First find the search function in the “solveEightQueens.py” file and then change the iteration i, judge if the current is more than 100, break the while-loop, or it has already become the solved case, break the while-loop.