The Report For Project 1:Pacman-Search

Quanyin Tang 17210190007

1. The Design of Search Algorithms

In question 1:Finding a Fixed Food Dot using Depth First Search, I just follow the pseudo code which uses a special data structure, stack. So I first initialize the fringe as a stack, and then push the starting point into the stack, pop out the point and add the point to visited list. Also, the goal test is needed. In question 2: Breadth First Search, I just port the DFS's code to the program by modifying the stack to queue. Also, the UCS is the same, just using the PriorityQueue. In question 4, A star Search, the default heuristic function is a trivial example, null. The program is similar with the UCS, and the difference is that the A star program uses the sum of the actual cost and the heuristic function as the total cost in priority queue while the ucs program just use the actual cost.

I am pleased that the program that I implement works fine under the test cases, such as tinyMaze, mdiumMaze, bigMaze, even the openMaze.

2. The Heuristic Function

To find the real power of the A star, I want to solve the problem of searching all corners. Firstly, I formulate the problem. I define the set of the {(position, cornerVisited = Boolen(corner))} as the state space and choose the (startPosition, InitCorner) as the start state. To get the successor, I just define as the next position and update the cornerVisitied. Also, the goal test and wall test is needed. Under this problem, I can solve the bigCorner problem within 3 seconds with expanding 7949 nodes and total cost of 162.

In question 6, what I need do is to design a heuristic function for A star what I implement in the question 4 to find all the corners. Considering the goal is just a state with four corners visited, this Heuristic function simply calculate the total manhattan distance from the current position to the final state. First I calculate the distance between the current position and the closest corner. Then I calculate the distance between the first closest corner and the new closest corner until it reaches the

state with four corners all visited. Obviously, this heuristic function is admissible and optimal. As the result, the A star with the corner heuristic function solve the bigCorner problem just expanding 1743 nodes and the mediumCorner problem expanding 693 nodes while the solution in question 5 expanding 7949 nodes for bigCorner and 1966 for medium.

To solve the question 7:eating all the dots, the most important thing I need to do is designing an effective and admissible heuristic function. As the goal state is eating all out all the food, one suitable heuristic function is the number of the food left and the distance of the furthest food is another function. I combine the two into one, I choose the sum mazedistance of all the food left. It is so effective that it just only expand 343 nodes within one second!

To solve the question 8, what I need to do is that design a key function that finds a path to the closest dot which missing in 'searchAgents.py'. I implement this just by BFS search to find the closest food. Another thing to do is the goal test which trivially implemented by check whether the food list is empty or not.

3. The Effective Analysis of the Heuristic Function

The effectiveness of the heuristic functions I designed are already analyzed in the part two. I think all of them are effective and admissible. Firstly, all of them will return 0 at every goal state and never return a negative value. Then they are always in accord with h(A)-h(C) <= cost(A to C), which means the heuristic function is optimal.

4. Conclustion

I solve all the question except the mediumSearch in question 7, and all the program is complete and optimal, I think. Also, the comments to explain my code can be seen in the file, which are helpful to reading my program.