Report of Project 1: Pac-man

Deng Wen 17307130171

2019/10/6

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

In this part, we need to implement a DFS algorithm.
And by the knowledge in class,we should be aware that **DFS expand the deepest node in current frontier.**And by using **a LIFO stack**, I can complete this method easily.

1.1 Result:

By typing the given code in command line, I got the following result:

Maze size	Total cost	Node expand	Score
tinyMaze	10	15	500
mediumMaze	130	146	380
bigMaze	210	390	300

We can see that the agent does not go to every square it explored to its goal. And the DFS method does not guarantee the optimal solution, for it always expand the deepest node in the frontier and may miss the optimal solution if it exsits in another shallower node.

Question 2: Breadth First Search

Similarly in this part, we need to implement a BFS algorithm.
And by the knowledge in class,we should be aware that **BFS expand the shallowest node in current frontier**.And by using **a FIFO queue**, I can complete this method easily,too.

2.1 Result:

By typing the given code in command line, I got the following result:

Maze	Total cost	Node expand	Score
tinyMaze	8	15	502
mediumMaze	68	267	442
bigMaze	210	617	300

As we can see, BFS method will find the optimal solution.But it expands more nodes than DFS method.
For I have written my code generically, it also work well in the eight-puzzle search problem.

Question 3: Varying the Cost Function

In this part,we are to implement an UCS method.And test how agent works in under different cost function. For **UCS method always expand the node that has the lowest path-cost,by using a priorityqueue**, we can complete it easily.

3.1 Result:

Maze	Total cost	Node expand	Score
mediumMaze	68	269	442
mediumDottedMaze	1	186	646
mediumScaryMaze	68719479864	108	418

From the result,we can find that UCS method can also find the optimal solution. **And due to the cost function, mediumDottedMaze has a very low cost because of** 0.5^x **exponential cost while mediumScaryMaze has a very high cost because of** 2^x **exponential cost.**

Question 4: A* search

When expanding, **A* search using information that combine the path-cost to current node and an estimation of lower bound of the cost from the node to the goal.** So we use a priorityqueue with this information as priority.

4.1 Result:

Maze	Method	Total cost	Node expand	Score	Optimal
bigMaze	UCS	210	620	300	Yes
bigMaze	A*	210	549	300	Yes
openMaze	DFS	298	806	212	No
openMaze	BFS	54	679	456	Yes
openMaze	UCS	54	682	456	Yes
openMaze	A*	54	535	456	Yes

It can be seen from the result that in bigMaze, A* search expanded fewer nodes than UCS to find an optimal solution.
In openMaze, DFS expanded many nodes and did not find an optimal solution. BFS and UCS both found an optimal solution with very slightly differences in number of nodes expanded. And A* search found an optimal solution with much fewer nodes expanded for it had used more information when searching.

Question 5: Finding All the Corners

In this question, we need to implement a new search problem: Corners problem.
For we are aim to touch all corners in a maze, so we can easily come up with an state defination that contains the current position and the four corners' state(0 touched and 1 untouched). Then step by step we can build our corners problem.

5.1 Result:

Maze	Total cost	Node expand	Score
tinyCorners	28	243	512
mediumCorners	106	1921	434
bigCorners	162	7862	378

From the result we can see our defination of the CornersProblem works well.

Question 6: Corners Problem: Heuristic

In this question, we need to define a Heuristic function for Corners problem to see how well A* search performs in this problem.
We can use the sum of the max distance from current position to the corners which are still untouched in four diretions(east, west, north, south).

6.1 Proof of Admissibility and consistency

Let a be the longest distance horizontally from current location to the eastest corner while b to the westest corner.
Similarly,c denote the longest distance vertically from current location to the northest corner while d to the southest corner.
Note: $a,\ b,\ c,\ d \geq 0$
1. When there is only one corner untouched. It's actually Manhattan distance which is certainly satisfied the two properties.
2. When there are multiple corners untouched.The Heuristic is simplified by only moving to the the four farrest diretions only at cost $a+b+c+d$ without counting the cost that the repeated squares that the agent may have touched.So it is admissible.
3. When the agent take a step in one diretion, it can only move a step which means the Heuristic will only drop 1 if the opposite diretion has no untouched corner or drop 0 if the opposite diretion has at least one untouched corner.So it is consistent.
Finally, under this Heuristic, when there is multiple corners untouched, the agent tend to move to the corner that have less cost to goal first. And When there is one corner untouched , it is actualy a position search problem, the agent choose the less cost way to the corner.

6.2 Result:

Maze	Total cost	Node expand	Score
mediumCorners	106	931	434
bigCorners	162	2924	378

As we can see, in the CornersProblem, A* search not only found optimal solution but also had a big reduction in the number of nodes expanded than BFS method.

Question 7: Eating All The Dots

In this problem, agent need to eat all the dots in maze with fewwest path cost. We need to implement a foodHeuristic function to help the agent find the optimal solution faster.

1.If we want to reduce the nodes expanded, only by the simple farrest Manhattan distance is not so effective(More than 12000 nodes in mediumCorners).**2.** If using the hendreric Similar to Question 6, it would also be ineffective(About 8600 nodes in mediumCorners).
So we come up with a more accurate distance which considering the walls in the maze. We use the positionSearchProblem to find the dot with longest actual distance in maze.And using the problem.heuristicInfo to store the information to avoid repeated computation.
For the agent must eat all the dots, so the heuristic is certainly admissible.And for every step the agent move, it will cause at most 1 drop in heuristic. So it is also consistent.

7.1 Result:

Maze	Method	Total cost	Node expand	Score	Time
tinySearch	UCS	27	5057	573	0.6s
tinySearch	A*	27	2411	573	0.3s
trickySearch	UCS	60	16688	570	3.4s
trickySearch	A*	60	4239	570	1.1s

From the result, we can find that both the A* search and UCS method found a optimal solution. However, A* search expanded much fewer nodes method and worked faster than UCS.
For medium search, I have tried a few relatively more complex heuristics, but they all failed for the exponential increase in the situations.

Qusetion 8: Suboptimal Search

In this part, we need to implement a agent which always find the closet dot. The agent is very useful to find suboptimal method when finding an optimal solution is very hard.
By the hint given in the Question, we can implement the goal test of anyFoodSearchProblem. Then complete the agent at ease.

8.1 Result:

After we run the code in command line, the agent quickly found a solution with **350 cost and 2360 Score.**

6.2 Small example:

The following is a small example that shows this method will not be optimal.

In this example, if the pacman always find the closet dot, the solution will be 11 while the optimal solution is 9.

Summary:

After finishing this Project, I gained a deeper understanding of different search methods in AI, and a great improvement in my coding experience.