Project 1 - Search Based Planning

Jared Miller

September 2024

1 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?

   > Yes! You can see that Pacman explores "deep" or "further away" from himself before exploring the nodes close to himself.

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

   > No. Depth First Search doesn't ever guarantee the least cost solution. It is simply an algorithm that checks to see if a path exists. I think that in this case Depth First Search finds a solution and returns early before checking all possibilities. This is expected behavior.

2 Breadth First Search

1. Does BFS find a least cost solution? If so, explain why.

   > Yes because of the nature of BFS search nodes are expanded "one node at a time" starting with the starting node. Therefore when the path is found a shorter path could not have been found sooner.

3 Varying the Cost Function (UCS)

1. Specify the data structure used from util.py for uniform cost search

   > The data structure used is a Priority Queue. This allows us to always take the cheapest edge.

4 A* Search

1. What happens on openMaze for the various search strategies? Describe your answer in terms of the solution you get for A* and uniform cost search.

   For UCS we notice that just about ever single space is explored. This makes sense as we are trying to find the shortest path and similarly to Breadth First Search we start from the starting posistion and take the shortest edge possible. There isn't anything guiding the search besides edge weights. For A* we notice that less spaces are explored. This is because we are able to use the Manhattan distance as a heuristic to bias pacman to explore nodes that will get him closer to his goal.

5 Finding All the Corners

1. Describe in few words/ lines the state representation you chose or how you solved the problem of finding all corners.

   To solve this problem I thought through what information was needed to know if we have reached a solution. This would decide how a state was defined. I realized that you would need a way to track your current posistion (to see if a corner was reached) and you would need to keep track of what corners you have already reached (so that when you reach all corners you can specify that a goal has been reached). To do this I tracked Pacmans current position along with a frozenset that contained the corners that were visited. The frozenset was used because it is an immutable data structure and could be hashed to be added to the PQ.

6 Corners Problem: Heuristic

1. Describe the heuristic you used for the implementation.

   The heuristic I used was Pacmans distance to the closest corner + the distance from that closest corner to the next closest corner + the distance from that closest corner to the next closest corner + the distance from that closest corner to the final corner. This allows Pacman to see the entire problem which helps him make educated descisions on which paths to take. To improve this even more rather than using the Manhattan distance you can use the actual maze distance. I tried a heuristic where Pacman only worries about the closest food and it wasn't quick enough.

7 Eating All Dots

1. Describe the heuristic you used for the FoodSearchProblem

   For this problem I tried multiple heuristics. I first tried using the Manhattan Distance and the Euclidean Distance but didn't get a consistent heuristic. I ended up finding a Minimum Spanning Tree and adding the cost of all the edge weights of that tree. I then added the distance from Pacmans position to the closest food.

8 Sub-optimal Search

1. Explain why the ClosestDotSearchAgent won't always find the shortest possible path through the maze.

   There are multiple reasons but one big one is the possibility of back-tracking. Even if a food is the closest food to Pacman it doesn't necessarily mean it should be the first food eaten. This is especially true if eating the closest food causes Pacman to have to back-track to the next closest food. Every time Pacman has to backtrack more distance is added to the path he needs to take.

9 Self Analysis

1. What was the hardest part of the assignment for you?

   (a) The main thing that was difficult was working with the pre-existing code. Because this is my first class in Python it was quite difficult to wrap my head around.

2. What was the easiest part of the assignment for you?

   Probably solving number eight

3. What problem(s) helped further your understanding of the course material?

   I feel that each question accomplished this.

4. Did you feel any problems were tedious and not helpful to your understanding of the material?

   No

5. What other feedback do you have about this homework?

   I really enjoyed this homework!