Project 1: Search

Franziska Rothen - franziska.rothen@students.unibe.ch - November 15, 2021

Abstract

The idea of this exercise was to solve various search problems for pacman.

1 Question 1: Depth First Search

With the depth first search method, we choose "the left most" direction and follow it as far as possible. When we end up in a dead end (end node that is not the goal node), we jump back to the last node where another path could have been taken and continue from there.

In the case seen in figure 1, Pacman took a wrong turn at node A and explored the nodes east of it. Once it discovered that this is a dead-end, it jumped back to node A and explored the south (where the goal is found eventually). The path that includes the goal state is saved and given as input of the path for Pacman. Nevertheless, the path in the east was explored therefore marked red.

For this algorithm, a stack was used. It works with a last-in-first-out (LIFO) queuing policy. The successor states for each node are ordered as follows:

```
[((x, y), 'North', cost), ((x, y), 'South', cost), //
((x, y), 'East', cost), ((x, y), 'West', cost)]
```

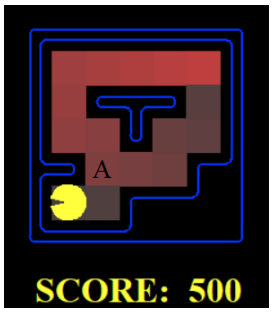

Figure 1: Example of end state of Pacman in a tiny maze after DFS.

For the choice of the next node, the stack is popped (starting with the last element), giving the direction the following descending priorities: 'West' - 'East' - 'South' - 'North'

This coincides with the decision taken in figure 1 ('West' over 'South', 'East' over 'South')

2 Question 2: Breadth First Search

The breadth first search explores all the successor states of the starting node and expands evenly in depth. The cost for each step is ignored, resulting in a solution that is possibly not the least cost path.

3 Question 3: Uniform Cost Search

We explore again each of the successor states of the starting node as in BFS. For further exploration however, the total cost of each following step is calculated. Therefore, the path of the lowest cost is followed and finally the winning path is the least cost solution.

4 Question 4: A* Search

On the open maze, the goal was found with the following results for the different search strategies:

Strategy	Cost	Speed	Nodes expanded
DFS	298	0.3 s	806
BFS	54	0.1 s	682
UCS	54	0.1 s	682
A* Search	54	0.1 s	535

DFS uses the most computational power since it expands the most nodes before finding the goal. This is due to the fact that in the worst case, the DFS will explore the entire search tree before finding the goal. The cost of the winning path is clearly not optimal.

When comparing BFS and UCS we see that they yield the same results. This can be explained by the fact that the cost of each node is equal, resulting in no improvement from BFS to UCS. With the A* Search on the other hand, the heuristic function is included in the strategy of choosing the next node. It adds the closest path to the goal from a given state (Manhattan heuristic). This leads to the path never taking a turn that will lead it further away from the goal in the case of an open path.

5 Question 5: Corner Problem BFS

When using an A* search, the algorithm makes sure the exploration is directed towards the missing corner. This means that if a corner has already been found, the heuristic will lead it away from the found corner respectively it will attract it to the corners yet to be explored. this way, the amount of nodes expanded can be minimized.

The most important part of the corners heuristic is to give the states that are closer to a nearby corner a lower heuristic value. This will direct Pacman towards the closest corner. Once a corner is detected, the distance won't have any influence on the heuristic value. To find the corner with the lowest distance to the current node (whose heuristic is to be determined) the Manhattan distance to each corner was calculated and the lowest value chosen. From there, depending on how many corners are left undiscovered, the total smallest distance to the remaining corners is added.

7 Question 8: Suboptimal Search

Your ClosestDotSearchAgent won't always find the shortest possible path through the maze. Make sure you understand why and try to come up with a small example where repeatedly going to the closest dot does not result in finding the shortest path for eating all the dots. Write down a descriptive for this example and why you chose this example