Explain.pdf: PacMan

Abgabe 21.06.2022, 9:55 Uhr

Marcel Bauer 5570300, Roman Svetlitckii 5124010

Question 1: Finding a Fixed Food Dot using DFS

Is the exploration order what you would have expected?

Yes, the search process is very similar to the exercises, especially when you tilt the mazes a bit and imagine it as a tree with the startknot on top.

Does Pacman actually go to all the explored squares on his way to the goal?

Not to all squares, because the algorithm terminates, when we find the goal.

IQ2: What difference can you observe between the used search strategies? Explain your observation.

1) The results for "python pacman.py -l mediumMaze -p SearchAgent -a fn=ucs" and
"python pacman.py -l mediumMaze -p SearchAgent -a fn=GreedyBestFirstSearch" are shown in the figure below.

Pacman chooses the same path in both cases. UCS and Greedy use the PriorityQueue. In UCS in each step a node with the best cost will be taken from the PriorityQueue and in the end the optimal path will be chosen. But in greedy search the nullHeuristic is used meaning that all nodes have the same value, so they will be pushed into the queue and taken out according to FIFO. Greedy search also chooses the optimal path because it will reach the "crossroads-nodes" first, thus not letting the more long paths to expand the nodes at these "crossroads-nodes".

```
[SearchAgent] using function GreedyBestFirstSearch and heuristic nullHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 269
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
[SearchAgent] using function ucs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 269
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win
```

2) The results for "python pacman.py -l mediumDottedMaze -p SearchAgent -a fn=GreedyBestFirstSearch" and
"python pacman.py -l mediumDottedMaze -p StayEastSearchAgent" are shown below.

Greedy also goes to the left and does not collect the group of dots. The reason why greedy search avoids collecting them is the same as in the previous subtask: it tries to reach the lower left food and the shortest path "captures" the crossroad. Pacman will also keep hitting the wall and the game will not be terminated. The StayEastSearchAgent as the name states chooses the east side and collects the dots. According to the searchAgents.py file it penalizes the west action and uses the UCF, so the west actions will have worse costs and will not be expanded, thus the difference in expanded nodes between both search strategies. As mentioned in the project StayEastSearchAgent uses an exponential formula: costFn = .5 ** pos, so the final cost will be very low, 1 in our case.

```
[SearchAgent] using function GreedyBestFirstSearch and heuristic nullHeuristic
[SearchAgent] using problem type PositionSearchProblem
Warning: this does not look like a regular search maze
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 208
```

```
Path found with total cost of 1 in 0.0 seconds
Search nodes expanded: 186
Pacman emerges victorious! Score: 646
Average Score: 646.0
Scores:        646.0
Win Rate:      1/1 (1.00)
Record:        Win
```

3) Finally "python pacman.py -l mediumScaryMaze -p SearchAgent -a fn=GreedyBestFirstSearch " and
"python pacman.py -l mediumScaryMaze -p StayWestSearchAgent".

Pacman acts exactly as before and goes through a very dangerous area gladly rushing into ghosts. The StayWestSearchAgent strategy forces Pacman to go to the west, so he avoids the ghosts and reaches the goal.
The exp function used is costFn = 2 ** pos, so the final cost is very high, 68719479864 in our case. StayWestSearchAgent also uses UCS, so it has fewer expanded nodes.

```
[SearchAgent] using function GreedyBestFirstSearch and heuristic nullHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 72 in 0.0 seconds
Search nodes expanded: 279
Pacman died! Score: -514
Average Score: -514.0
Scores:        -514.0
Win Rate:      0/1 (0.00)
Record:        Loss
```

```
Path found with total cost of 68719479864 in 0.1 seconds
Search nodes expanded: 108
Pacman emerges victorious! Score: 418
Average Score: 418.0
Scores:        418.0
Win Rate:      1/1 (1.00)
Record:        Win
```

IQ3: What happens on openMaze for the various search strategies? Replace manhattanHeuristic with euclideanHeuristic. What difference can you notice? Wich heuristic is better here and why?

1) DFS, BFS and UCS dont use the heuristic but Pacman chooses a very different and an extremely long path for DFS. It is not the longest possible way but Pacman truly goes into the depth. BFS and UCS just produce the same optimal way.

```
[SearchAgent] using function dfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 298 in 0.0 seconds
Search nodes expanded: 806
Pacman emerges victorious! Score: 212
Average Score: 212.0
Scores:        212.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.1 seconds
Search nodes expanded: 682
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
[SearchAgent] using function ucs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.4 seconds
Search nodes expanded: 682
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
```

2) Greedy search is more efficient with euclidean. Using manhattan it goes through an unnecessary area. Only heuristic values have affect on the priority, so the first chosen node(left or down by starting in the top right corner) will decide where the expansion will go. It goes downwards in our case and hits the wall, after that it tries to go to the left, but the wall there is relatively high, thus it expands the nodes there in form of a pyramid, because the bottom nodes from this "pyramid" have better cost values and will be expanded first. After Pacman finally overcomes the wall it starts moving to the south again but this time there are no obstacles, so after reaching the bottom of the map there is only one option: go to the left to the food. Euclidean is better because it gives a higher priority to the diagonal node and also prefers to reduce the longer axis(the formula produces lower results if we reduce the higher distance). The x axis in the openmaze is longer, so Pacman goes to the west at first, then it reaches a node, where the distance to the food is equal on both axises, so it starts to move diagonally. After some time Pacman hits the wall and must go downwards. When the end of the wall is reached the x axis is longer again, so Pacman goes strictly to the left until x axis is equal to the y axis, after that it goes diagonally to the goal.

```
[SearchAgent] using function GreedyBestFirstSearch and heuristic euclideanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.0 seconds
Search nodes expanded: 54
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
[SearchAgent] using function GreedyBestFirstSearch and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 68 in 0.0 seconds
Search nodes expanded: 89
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442.0
Win Rate:      1/1 (1.00)
Record:        Win
```

3) The A* search takes into account the already visited nodes, so it will expand way more nodes, because each new expanded node gets some extra cost depending on the chain of its predecessors. Since the euclidean distance is a straight line it will add some relatively small number(compared to manhattan) to some nodes in the bottom part of the map, thus giving these nodes a chance to be expanded after some nodes closer to the goal state have collected more nodes into their chains and gained more weight, which is practically equal to the manhattan distance. Thus manhattan distance is slightly better.

```
[SearchAgent] using function astar and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.1 seconds
Search nodes expanded: 535
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
[SearchAgent] using function astar and heuristic euclideanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 54 in 0.1 seconds
Search nodes expanded: 550
Pacman emerges victorious! Score: 456
Average Score: 456.0
Scores:        456.0
Win Rate:      1/1 (1.00)
Record:        Win
```

IQ4: IQ: Describe your heuristic used in the foodHeuristic method.

In the Q6 we have calculated the optimal cost using the recursion, but this time it will not be feasible because there are many food dots on the map. A better approach would be to get as close to the optimal solution as possible. First the true distances between all non-wall nodes are calculated with UCS, so there is a dictionary entry for each pair: (some non-wall node, some non-wall node). A food node is also a non-wall node, so all food is stored in the dictionary. In each call to the heuristic function a set of goal nodes and the actual position are given. One possible heuristic could be:

1) Using the precalculated information find out the maximum distance from the current node to some food from a given foodGrid, associated with the current node.

2) Return this max value.

This heuristic is admissible because either all food can be collected along the way to the farthest food, then the heuristic is equal to the optimal cost. Or Pacman must travel a longer way to collect the food, then the actual cost is higher than the estimated one.

It is also consistent, because:

1) Given two adjacent nodes A and B with the same set of foodGrid. The max distance to some food will be the same for both nodes, because they have the same set of not collected food. The cost of moving from between A to B is 1(as mentioned in the FoodSearchProblem) and the difference in heuristic is also 1.

2) If A and B have different foodGrid, because moving from A to B results in deletion of some food. Then they will have the same farthest food, because there is no way to point to the farthest food and collect it at the same time, except for the last food of course.