Assignment 1 – AI

Report

Lakshmi Narayanan Rajendran (SBU ID: 112046482)

Abhinav Srivastava (SBU ID: 112129749)

Finding a Fixed Food Dot using Search Algorithms

Question 1

1.1 `python pacman.py -l tinyMaze -p SearchAgent`

Result:

[SearchAgent] using function depthFirstSearch

[SearchAgent] using problem type PositionSearchProblem

Path found with total cost of 10 in 0.0 seconds

Search nodes expanded: 15

Pacman emerges victorious! Score: 500

Average Score: 500.0

Scores: 500.0

Win Rate: 1/1 (1.00)

Record: Win

1.2 `python pacman.py -l mediumMaze -p SearchAgent`

Result:

[SearchAgent] using function depthFirstSearch

[SearchAgent] using problem type PositionSearchProblem

Path found with total cost of 130 in 0.0 seconds

Search nodes expanded: 146

Pacman emerges victorious! Score: 380

Average Score: 380.0

Scores: 380.0

Win Rate: 1/1 (1.00)

Record: Win

1.3 `python pacman.py -l bigMaze -z .5 -p SearchAgent`

Result:

[SearchAgent] using function depthFirstSearch

[SearchAgent] using problem type PositionSearchProblem

Path found with total cost of 210 in 0.1 seconds

Search nodes expanded: 390

Pacman emerges victorious! Score: 300

Average Score: 300.0

Scores: 300.0

Win Rate: 1/1 (1.00)

Record: Win

Question 2

2.1 `python pacman.py -l mediumMaze -p SearchAgent -a fn=bfs`

Result:

[SearchAgent] using function bfs

[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

`2.2 python pacman.py -l bigMaze -p SearchAgent -a fn=bfs -z .5`

Result:

[SearchAgent] using function bfs

[SearchAgent] using problem type PositionSearchProblem

Path found with total cost of 210 in 0.0 seconds

Search nodes expanded: 620

Pacman emerges victorious! Score: 300

Average Score: 300.0

Scores: 300.0

Win Rate: 1/1 (1.00)

Record: Win

Eight-puzzle search problem (R&N 3ed Section 3.2, Figure 3.4) without any changes.

A random puzzle:

| 1 | 2 | 8 |

| 3 | 4 | |

| 6 | 5 | 7 |

BFS found a path of 9 moves: ['up', 'left', 'down', 'down', 'right', 'up', 'left', 'up', 'left']

After 1 move: up

| 1 | 2 | |

| 3 | 4 | 8 |

| 6 | 5 | 7 |

Press return for the next state...

After 2 moves: left

| 1 | | 2 |

| 3 | 4 | 8 |

| 6 | 5 | 7 |

Press return for the next state...

After 3 moves: down

| 1 | 4 | 2 |

| 3 | | 8 |

| 6 | 5 | 7 |

Press return for the next state...

After 4 moves: down

| 1 | 4 | 2 |

| 3 | 5 | 8 |

| 6 | | 7 |

Press return for the next state...

After 5 moves: right

| 1 | 4 | 2 |

| 3 | 5 | 8 |

| 6 | 7 | |

Press return for the next state...

After 6 moves: up

| 1 | 4 | 2 |

| 3 | 5 | |

| 6 | 7 | 8 |

Press return for the next state...

After 7 moves: left

| 1 | 4 | 2 |

| 3 | | 5 |

| 6 | 7 | 8 |

Press return for the next state...

After 8 moves: up

| 1 | | 2 |

| 3 | 4 | 5 |

| 6 | 7 | 8 |

Press return for the next state...

After 9 moves: left

| | 1 | 2 |

| 3 | 4 | 5 |

| 6 | 7 | 8 |

Varying the Cost Function

Question 3

`3.1 python pacman.py -l mediumMaze -p SearchAgent -a fn=ucs`

Result:
[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

`3.2 python pacman.py -l mediumDottedMaze -p StayEastSearchAgent`

Result:

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

```
3.3   python pacman.py -l mediumScaryMaze -p
StayWestSearchAgent
```

Result:

Path found with total cost of 68719479864 in 0.0 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

A* search

Question 4

```
4.1    python pacman.py -l bigMaze -z .5 -p SearchAgent
-a fn=astar,heuristic=manhattanHeuristic
```

Result:

[SearchAgent] using function astar and heuristic manhattanHeuristic

[SearchAgent] using problem type PositionSearchProblem

Path found with total cost of 210 in 0.1 seconds

Search nodes expanded: 549

Pacman emerges victorious! Score: 300

Average Score: 300.0

Scores: 300.0

Win Rate: 1/1 (1.00)

Record: Win

Finding All the Corners

Question 5

`5.1 python pacman.py -l tinyCorners -p SearchAgent -a fn=bfs,prob=CornersProblem`

Start State: (start_position, corners[])
Goal State Check: visited[corners] == true / Number of unvisited corners == 0

Result:

[SearchAgent] using function bfs

[SearchAgent] using problem type CornersProblem

Path found with total cost of 28 in 0.0 seconds

Search nodes expanded: 252

Pacman emerges victorious! Score: 512

Average Score: 512.0

Scores: 512.0

Win Rate: 1/1 (1.00)

Record: Win

`5.2 python pacman.py -l mediumCorners -p SearchAgent -a fn=bfs,prob=CornersProblem`

Result:

[SearchAgent] using function bfs

[SearchAgent] using problem type CornersProblem

Path found with total cost of 106 in 0.2 seconds

Search nodes expanded: 1966

Pacman emerges victorious! Score: 434

Average Score: 434.0

Scores: 434.0

Win Rate: 1/1 (1.00)

Record: Win

Question 6

`6.1 python pacman.py -l mediumCorners -p AStarCornersAgent -z 0.5`

Heuristic Chosen: Shortest path from current position to the last unvisited corner covering all the unvisited corners (using Manhattan Distances). This heuristic is consistent because it denotes the distance the Pacman has to travel to reach goal state.

Result:

Path found with total cost of 106 in 0.0 seconds

Search nodes expanded: 692

Pacman emerges victorious! Score: 434

Average Score: 434.0

Scores: 434.0

Win Rate: 1/1 (1.00)

Record: Win

Eating All the Dots

`python pacman.py -l testSearch -p AStarFoodSearchAgent`

Result:

Path found with total cost of 7 in 0.0 seconds

Search nodes expanded: 12

Pacman emerges victorious! Score: 513

Average Score: 513.0

Scores: 513.0

Win Rate: 1/1 (1.00)

Record: Win

Question 7

```
python pacman.py -l trickySearch -p
AStarFoodSearchAgent
```

Heuristic Chosen: Sum of Manhattan distances between the two farthest food points in the layout and Manhattan distance from the start position and the nearest of the two points. Let's say S(xs,ys) and L(xl,yl) are the two farthest food points and P(xp,yp) is the current Pacman position.

Heuristic, **h = manhattanDistance(P, nearest(S,L)) + manhattanDistance(S,L)**

The heuristic is consistent because all the food points lie inside this rectangle and it would give the minimum distance the Pacman would have to travel to reach the goal state.

Result:

Path found with total cost of 60 in 6.0 seconds

Search nodes expanded: 7481

Pacman emerges victorious!

Average Score: 570.0

Scores: 570.0

Win Rate: 1/1 (1.00)

Record: Win