Report on Project 1: The Searchin’ Pac-Man

Date: 09/17/2018

Project Members:

Sunny Shah (112044068)

Dhruva Gaidhani (111971181)

MS CS Fall ’18

Q1. Depth First Search

1. Maze type: Tiny  
   Code output:

[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. Maze type: Medium  
   Code Output:

[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. Maze type: Big  
   Code Output:

[SearchAgent] using function depthFirstSearch

[SearchAgent] using problem type PositionSearchProblem

Path found with total cost of 210 in 0.0 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

DFS Analysis:

* The exploration is not exactly as is expected, the pacman actually takes a *longer* route as it explores deeper into the graph thus missing the optimal solution.
* The Pacman does not go to all the explored states on its way to the goal. This is because the current path that the Pacman is exploring reaches the Goal and returns the path before further states can be found.
* The algorithm runs with O(V+E) where V is number of Vertices and E is the Edges. The solution is found quickly in terms of temporal metrics but it is far from optimal.
* The space complexity is of the order O(d) which is of linear type where d is the maximum depth.

Q2. Breadth First Search

1. Maze type: Medium  
   Code output:

[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

1. Maze type: Big  
   Code output:

[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

BFS Analysis:

* BFS finds a lesser cost solution which is twice as less expensive as DFS.
* The algorithm runs with O(V+E) where V is number of Vertices and E is the Edges as well but provides a more complete solution as opposed to DFS.
* The space complexity is O(v) since in the worst case we have to hold all the vertices in the queue.

Q3. Uniform Cost Search

1. Maze type: Medium  
   Code output:

[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

1. Maze type: Medium Dotted  
   Code output:

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

1. Maze type: Medium Scary  
   Code output:

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

UCS Analysis:

* The algorithm’s worst-case time and space complexity is O(b^(1+C/ε)), which can be much greater than bd

-https://stackoverflow.com/questions/19204682/time-complexity-of-uniform-cost-search

Q4. A-Star search

1. Maze type: Big  
   Code output:

[SearchAgent] using function astar and heuristic manhattanHeuristic

[SearchAgent] using problem type PositionSearchProblem

Path found with total cost of 210 in 0.0 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

A-Star Analysis:

* The time complexity of A\* depends on the heuristic.
* In the worst case of an unbounded search space, the number of nodes expanded is exponential in the depth of the solution (the shortest path) d: O(bd), where b is the branching factor (the average number of successors per state).

-https://cs.stackexchange.com/questions/56176/a-graph-search-time-complexity

Analysis for OpenMaze:

1. D:\Game Plan\Stony Brook University\Courses\Artificial Intelligence\Project\Solu

tion codes\Sunny>python2 pacman.py -l openMaze -p SearchAgent -a fn=astar,heuris

tic=manhattanHeuristic

[SearchAgent] using function astar and heuristic manhattanHeuristic

[SearchAgent] using problem type PositionSearchProblem

Path found with total cost of 54 in 0.0 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

2. D:\Game Plan\Stony Brook University\Courses\Artificial Intelligence\Project\Solu

tion codes\Sunny>python2 pacman.py -l openMaze -p SearchAgent -a fn=bfs

[SearchAgent] using function bfs

[SearchAgent] using problem type PositionSearchProblem

Path found with total cost of 54 in 0.0 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

3. D:\Game Plan\Stony Brook University\Courses\Artificial Intelligence\Project\Solu

tion codes\Sunny>python2 pacman.py -l openMaze -p SearchAgent -a fn=ucs

[SearchAgent] using function ucs

[SearchAgent] using problem type PositionSearchProblem

Path found with total cost of 54 in 0.0 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

4. D:\Game Plan\Stony Brook University\Courses\Artificial Intelligence\Project\Solu

tion codes\Sunny>python2 pacman.py -l openMaze -p SearchAgent -a fn=dfs

[SearchAgent] using function dfs

[SearchAgent] using problem type PositionSearchProblem

Path found with total cost of 298 in 0.0 seconds

Search nodes expanded: 576

Pacman emerges victorious! Score: 212

Average Score: 212.0

Scores: 212.0

Win Rate: 1/1 (1.00)

Record: Win

Q5. Finding all the corners

1. Maze type: Tiny  
   Code output:

[SearchAgent] using function bfs

[SearchAgent] using problem type CornersProblem

Path found with total cost of 28 in 0.0 seconds

Search nodes expanded: **356**

Pacman emerges victorious! Score: 512

Average Score: 512.0

Scores: 512.0

Win Rate: 1/1 (1.00)

Record: Win

1. Maze type: Medium  
   Code output:

[SearchAgent] using function bfs

[SearchAgent] using problem type CornersProblem

Path found with total cost of 106 in 0.0 seconds

Search nodes expanded: **2742**

Pacman emerges victorious! Score: 434

Average Score: 434.0

Scores: 434.0

Win Rate: 1/1 (1.00)

Record: Win

Q6. Corners Problem

1. Maze type: Medium  
   Code output:

Path found with total cost of 106 in 0.0 seconds

Search nodes expanded: **965**

Pacman emerges victorious! Score: 434

Average Score: 434.0

Scores: 434.0

Win Rate: 1/1 (1.00)

Record: Win

1. Maze type: Test  
   Code output:

Path found with total cost of 7 in 0.0 seconds

Search nodes expanded: **14**

Pacman emerges victorious! Score: 513

Average Score: 513.0

Scores: 513.0

Win Rate: 1/1 (1.00)

Record: Win

Q7. Corners Problem

1. Maze type: Medium  
   Code output:

Path found with total cost of 60 in 2.0 seconds

Search nodes expanded: **376**

Pacman emerges victorious! Score: 570

Average Score: 570.0

Scores: 570.0

Win Rate: 1/1 (1.00)

Record: Win