Avery Nelson --- 0 late days used

010920903

amn018@uark.edu

Artificial Intelligence Assignment 2 Report

Description of Implementation

The goal of this assignment is to implement an algorithm that teaches the Pacman agent to find the path on the maze to collect his food. Given the map, the start position, and the food position, we are required to implement a searching algorithm to find the way from the start position to the food position. The searching function will return a list of actions that leads the Pacman from the start position to his food. The cost of this game is the total cost step of each action performed during the path.We were given files are used to render the Pacman game: graphicsDisplay.py, graphicsUtils.py, textDisplay.py, ghostDisplay.py, keyboardAgents.py, and layout.py. game.py defines how the game is played, describing AgentState, Agent, Direction, and Grid. util.py consists of useful data structures. pacman.py is the main program that runs the game. searchAgents.py is used to call the algorithms we are charged with implementing, found in search.py. Completing the tasks required implementing the provided breadth-first, depth-first, uniform cost, and A* functions in search.py. This was straightforward and mostly just required implementing basic algorithms. The challenging part was dealing with the required number of wall hits.

For each of the searching algorithms, Pacman needs to hit the walls at least once time and no more than 2 times (at max 2 times). For example, if Pacman reaches the destination without hitting the wall, you lose the game. Or if Pacman reaches the destination by hitting the wall three times, you also lose the game. All of them also need to track visited states to prevent cycles and store the path of actions taken to reach each state.

The DFS implementation uses a Stack data structure and explores the deepest nodes first. This implementation uses a Stack that stores tuples of state, path, and wall hits. It uses a set to track visited states with their wall hit counts and builds the path incrementally by appending new actions. It tracks wall hits using problem.isWall(successor) and determines success by the goal state reach with 1-2 wall hits and determines failure by exceeding 2 wall hits or exhausting all paths.

The BFS implementation explores nodes level by level using a Queue. This implementation uses a Queue storing tuples of state, path, and wall hits. It explores all nodes at current depth before moving deeper and stores states similarly to DFS but ensures shortest path due to level-wise exploration rather than depth-wise exploration. It uses the same wall counting and success determination as DFS.

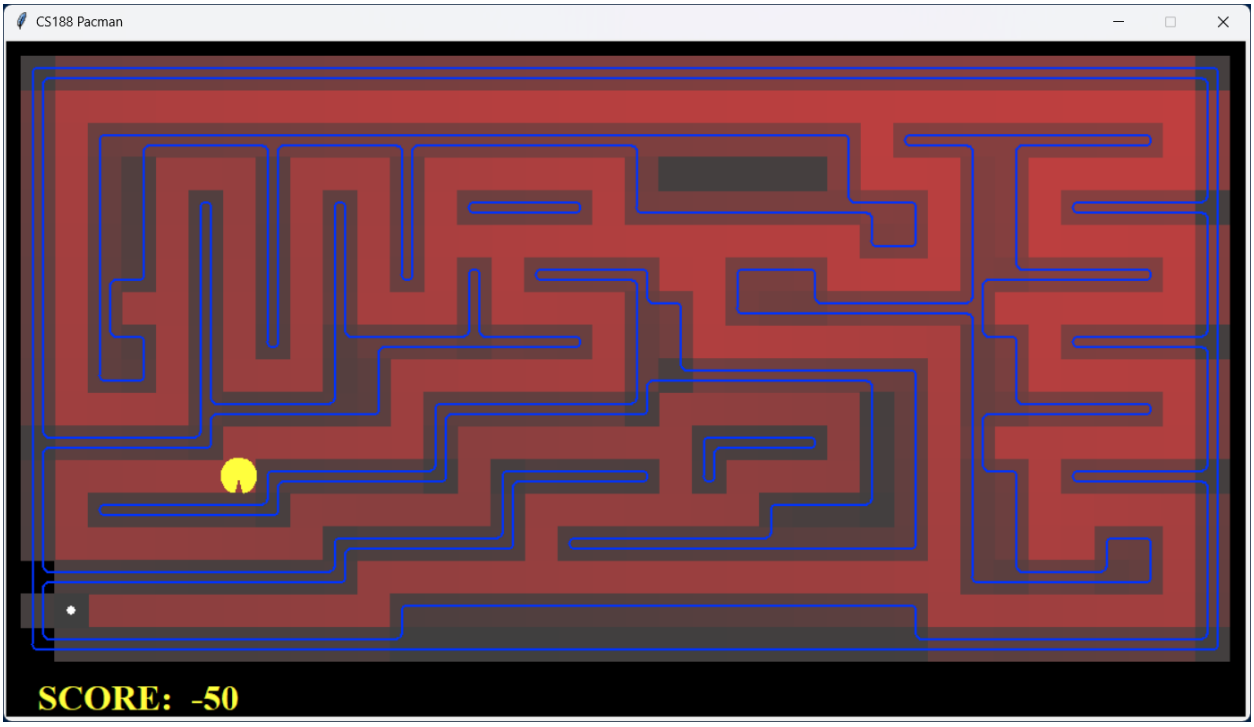
UCS explores nodes based on their path cost using a Priority Queue. This implementation uses a PriorityQueue with priority based on path cost. It gets the path cost with problem.getCostOfActions(). It stores tuples like BFS and DFS with the same elements. UCS

finds the lowest-cost path to goal due to cost-based exploration and handles walls and success/failure the same way DFS and BFS do.

A* enhances UCS by incorporating a heuristic function. It uses a PriorityQueue with priority f(n) = g(n) + h(n). The heuristic integration combines actual path cost with the heuristic estimate. g(n) is the actual cost from start to current node (getCostOfActions) and h(n) is the heuristic estimate from current node to goal.

Results

Overall, my implementations of each of the problems appears to be successful. All of them determine victory given the test codes provided. In each of them, PacMan reaches the goal node (food) in 1-2 moves. Additionally, I checked with our TA to confirm that my code was correct as the definition of hitting a wall confused me initially. Thus, the project successfully completes the required tasks given the test cases and parameters provided.



```
PS C:\Users\avery\Documents\GitHub\AI-sp25\HW2\search> python pacman.py --l tinyMaze --p SearchAgent --a fn=bfs
readCommand argv {argv}
[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 999999 in 0.0 seconds
Search nodes expanded: 90
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win
PS C:\Users\avery\Documents\GitHub\AI-sp25\HW2\search> python pacman.py --l tinyMaze --p SearchAgent --a fn=dfs
readCommand argv {argv}
[SearchAgent] using function dfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 999999 in 0.0 seconds
Search nodes expanded: 20
Pacman emerges victorious! Score: 490
Average Score: 490.0
Scores:        490.0
Win Rate:      1/1 (1.00)
Record:        Win
PS C:\Users\avery\Documents\GitHub\AI-sp25\HW2\search> python pacman.py --l tinyMaze --p SearchAgent --a fn=ucs
readCommand argv {argv}
[SearchAgent] using function ucs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 999999 in 0.0 seconds
Search nodes expanded: 75
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Win Rate:      1/1 (1.00)
Record:        Win
PS C:\Users\avery\Documents\GitHub\AI-sp25\HW2\search> python pacman.py -l tinyMaze -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic
readCommand argv {argv}
[SearchAgent] using function astar and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 999999 in 0.0 seconds
Search nodes expanded: 17
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
PS C:\Users\avery\Documents\GitHub\AI-sp25\HW2\search> python pacman.py --l mediumMaze --p SearchAgent --a fn=bfs
readCommand argv {argv}
[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 999999 in 0.0 seconds
Search nodes expanded: 1322
Pacman emerges victorious! Score: 462
Average Score: 462.0
Scores:        462.0
Win Rate:      1/1 (1.00)
Record:        Win
PS C:\Users\avery\Documents\GitHub\AI-sp25\HW2\search> python pacman.py --l mediumMaze --p SearchAgent --a fn=dfs
readCommand argv {argv}
[SearchAgent] using function dfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 999999 in 0.0 seconds
Search nodes expanded: 146
Pacman emerges victorious! Score: 366
Average Score: 366.0
Scores:        366.0
Win Rate:      1/1 (1.00)
Record:        Win
PS C:\Users\avery\Documents\GitHub\AI-sp25\HW2\search> python pacman.py --l mediumMaze --p SearchAgent --a fn=ucs
readCommand argv {argv}
[SearchAgent] using function ucs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 999999 in 0.5 seconds
Search nodes expanded: 1221
Pacman emerges victorious! Score: 450
Average Score: 450.0
Scores:        450.0
Win Rate:      1/1 (1.00)
Record:        Win
PS C:\Users\avery\Documents\GitHub\AI-sp25\HW2\search> python pacman.py --l mediumMaze --p SearchAgent --a fn=astar,heuristic=manhattanHeuristic
readCommand argv {argv}
[SearchAgent] using function astar and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 999999 in 0.4 seconds
Search nodes expanded: 275
Pacman emerges victorious! Score: 450
Average Score: 450.0
Scores:        450.0
Win Rate:      1/1 (1.00)
Record:        Win
```

```
PS C:\Users\avery\Documents\GitHub\AI-sp25\HW2\search> python pacman.py --l bigMaze --p SearchAgent --a fn=bfs
readCommand argv {argv}
[SearchAgent] using function bfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 999999 in 0.0 seconds
Search nodes expanded: 1355
Pacman emerges victorious! Score: 464
Average Score: 464.0
Scores:        464.0
Win Rate:      1/1 (1.00)
Record:        Win
PS C:\Users\avery\Documents\GitHub\AI-sp25\HW2\search> python pacman.py --l bigMaze --p SearchAgent --a fn=dfs
readCommand argv {argv}
[SearchAgent] using function dfs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 999999 in 0.0 seconds
Search nodes expanded: 382
Pacman emerges victorious! Score: 304
Average Score: 304.0
Scores:        304.0
Win Rate:      1/1 (1.00)
Record:        Win
PS C:\Users\avery\Documents\GitHub\AI-sp25\HW2\search> python pacman.py --l bigMaze --p SearchAgent --a fn=ucs
readCommand argv {argv}
[SearchAgent] using function ucs
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 999999 in 1.6 seconds
Search nodes expanded: 2631
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
PS C:\Users\avery\Documents\GitHub\AI-sp25\HW2\search> python pacman.py --l bigMaze --p SearchAgent --a fn=astar,heuristic=manhattanHeuristic
readCommand argv {argv}
[SearchAgent] using function astar and heuristic manhattanHeuristic
[SearchAgent] using problem type PositionSearchProblem
Path found with total cost of 999999 in 1.2 seconds
Search nodes expanded: 648
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300.0
Win Rate:      1/1 (1.00)
Record:        Win
```