Introduction to Artificial Intelligence HW#2

Name: Lee JaeJun

Student ID: 2021313030

**Abstract**  
 This project employs Q-learning algorithms to solve a maze displayed on PyMaze with 20x20 grids created by DFS backtracking. Initially, the agent is unaware of the maze layout, but it learns to find the optimal path without crossing walls using an ε-greedy strategy and a decaying exploration rate. Users can adjust the number of episodes, learning rate, discount factor, and exploration rate to influence the learning process. These adjustments facilitate easy observation of variable impacts. The final Q-table guides the agent in choosing the most efficient path through the maze.

**Introduction**

This report discusses the implementation of Q-learning algorithms and analyzes the variations resulting from changes in each variable, such as the learning rate and discount factor. I will examine the impact of these variables on the learning process and outcomes. Through this analysis, I aim to find the optimal parameters for navigating a 20x20 maze.

**Definition**

**Q-learning algorithm**: The Q-learning algorithm is a reinforcement learning technique that iteratively updates a Q-table based on actions taken and rewards received to discover the optimal path in each environment.

**Optimal path**: The optimal path is determined by selecting actions with the highest Q-values at each state until reaching the goal state.

**Cost**: The "cost" typically refers to the cumulative number of steps taken to reach the goal.

**Methodology**

**Maze Generation**

The maze utilized for testing was created through a depth-first search backtracking algorithm applied to a 20x20 grid. Each grid cell represents a possible step for the algorithm. In the `maze.py` file, `random.seed(integer)` is utilized to determine the placement of walls within the maze. Meanwhile, `random.seed(integer)` in `algorithm.py` is responsible for determining the entry and exit points. To guarantee consistency and replicability, a fixed seed value was employed during maze generation, ensuring the following:

|  |  |  |
| --- | --- | --- |
|  | src/maze.py | src/algorithm.py |
| Maze 1 | random.seed(0) | random.seed(0) |
| Maze 2 | random.seed(1) | random.seed(1) |
| Maze 3 | random.seed(2) | random.seed(2) |

**Cost Assignment**

The cost is standardized to 1 regardless of direction.

**Visualization**

The first animation depicts the process of moving through cells one by one, highlighting each cell as it is traversed. The cost increases with each movement. Then, the final path is highlighted with circles, showcasing the final route, which includes the final cost.

**Implementation Details**

**examples/solve\_Q\_learning.py**

**Modifications and Enhancements**

**Results and Discussion**

**Maze.py -> seed(미로 모양 바뀜)**

**Algorithm.py -> seed(출발, 도착 위치 바뀜)**

**Seed(0), seed(0)**

**Seed(1), seed(1)**

**Seed(2), seed(2)**