# Project 4: Maze Solver

Omar Atef 7619
Iyad Ashraf 7392

January 19, 2024

**Abstract**

This project introduces a maze-solving algorithm employing policy and value iteration techniques. The algorithm generates a maze of size NxN with randomly placed barriers, challenging an autonomous agent to learn the optimal path through the grid. The maze generation process, barrier placement, and the application of reinforcement learning methods are detailed. The report showcases the effectiveness of policy and value iteration in teaching the agent to navigate the maze, presenting experimental results, learning curves, and convergence analyses. Overall, this Maze Solver project contributes to the understanding of maze-solving strategies and reinforces the potential applications of reinforcement learning in autonomous navigation and path planning.

## Contents

## 1 Environment Representation

- The simplest way to represent the environment is using a 2D array. However, using 2D array uses ( $9 \times 9 \times$ `sizeofinteger` ). This representation would limits the agent by only reaching short nodes(states) and doesn't enable the agent to go deep in Finding the optimal path. Also it will slow down the algorithm and by increasing the size of the maze the algorithm time will increase Polynomially.

- Therefore we Represented the States with 1D array of int that makes the agent faster when solving the game.
- 3 variables used for environment visualization:
  - Maze size
  - Obstacle density
  - Toggle value/policy iteration

## 2 Algorithms

### 2.1 Value iteration Algorithm

Start with short-horizon values and build up to longer horizons.

- **Objective:**
  - Develop a value iteration algorithm for maze-solving.
- **Iteration Loop:**
  - Execute a value iteration loop until convergence.
  - Display the state of the algorithm every 5 iterations.
- **Value Update:**
  - For each non-obstacle state in the maze:
    * Compute the new value based on the maximum Q-value among possible actions.
    * Handle cases where no actions are available by assigning a predefined "stuck" value.
- **Solvability Check:**
  - Verify solvability for each state:
    * If not solvable, check neighboring states and update solvability status accordingly.
- **Convergence Criterion:**
  - Terminate the loop when the maximum change in values falls below a predefined threshold (1e-4).
- **Policy Improvement:**
  - Implement a policy improvement step after convergence.

- **Functions Used:**
  - get_actions
    - Retrieves valid actions for a given state in the maze.
    - Considers maze boundaries and obstacles to determine valid left, up, right, and down actions.
  - get_q_value
    - Calculates the Q-value for a given state-action pair in a maze-solving scenario.
    - Combines the penalty for the specified action with the value of the next state to represent the cumulative reward and expected future value.
  - get_next_state
    - Determines the next state in a maze based on the current state and a specified action.
    - Adjusts the next state considering maze boundaries and the chosen action (left, up, right, down).

## 2.2 Policy iteration Generation

Start with an arbitrary policy and iteratively improve it.

- **Objective:**
  - Implement the Policy Iteration algorithm for solving a maze.
- **Iteration Loop:**
  - Execute the Policy Iteration loop for a specified number of iterations.
  - Display the state of the algorithm every 5 iterations.
- **Policy Evaluation:**
  - Evaluate the current policy through policy evaluation function.
- **Policy Improvement:**
  - Implement policy improvement steps based on the evaluation.
  - Break the loop if no further improvement is achieved.
- **Functions Used:**
  - policy_evaluation
    - Iterate through non-obstacle states to calculate their new values based on the average Q-value from all possible actions.
    - Update the value function and handle state solvability during the evaluation.
  - policy_improvement

        * Iterate through non-obstacle states to compare old and new policies.
        * Update the policy and check for stability, indicating convergence when policies for all solvable states remain unchanged.
    - get_best_policy
        * Calculate Q-values for all possible actions in the current state.
        * Assign a binary policy where the action with the maximum Q-value receives a score of 1, and others receive 0.

## 3 Data Structure Used

The project utilizes specific data structures and algorithms for maze generation and solving through reinforcement learning. Below is a concise overview of the key data structures employed and the rationale behind their selection:

### 3.1 Grid Representation

Grid representations are fundamental to the project, serving as the basis for maze generation, exploration, and reinforcement learning. A 1D grid structure is employed to model the maze environment. Each cell in the grid represents a state, and the grid as a whole captures the spatial relationships between states.

### 3.2 Value and Policy Lists

Lists are extensively used to manage values and policies in the reinforcement learning process. They are employed to store values, representing the expected cumulative rewards for taking specific actions in each state. Additionally, policy lists indicate the chosen actions based on the learned values. Lists were selected for their flexibility, allowing efficient access and modification of values during the learning process.

### 3.3 Domain Lists

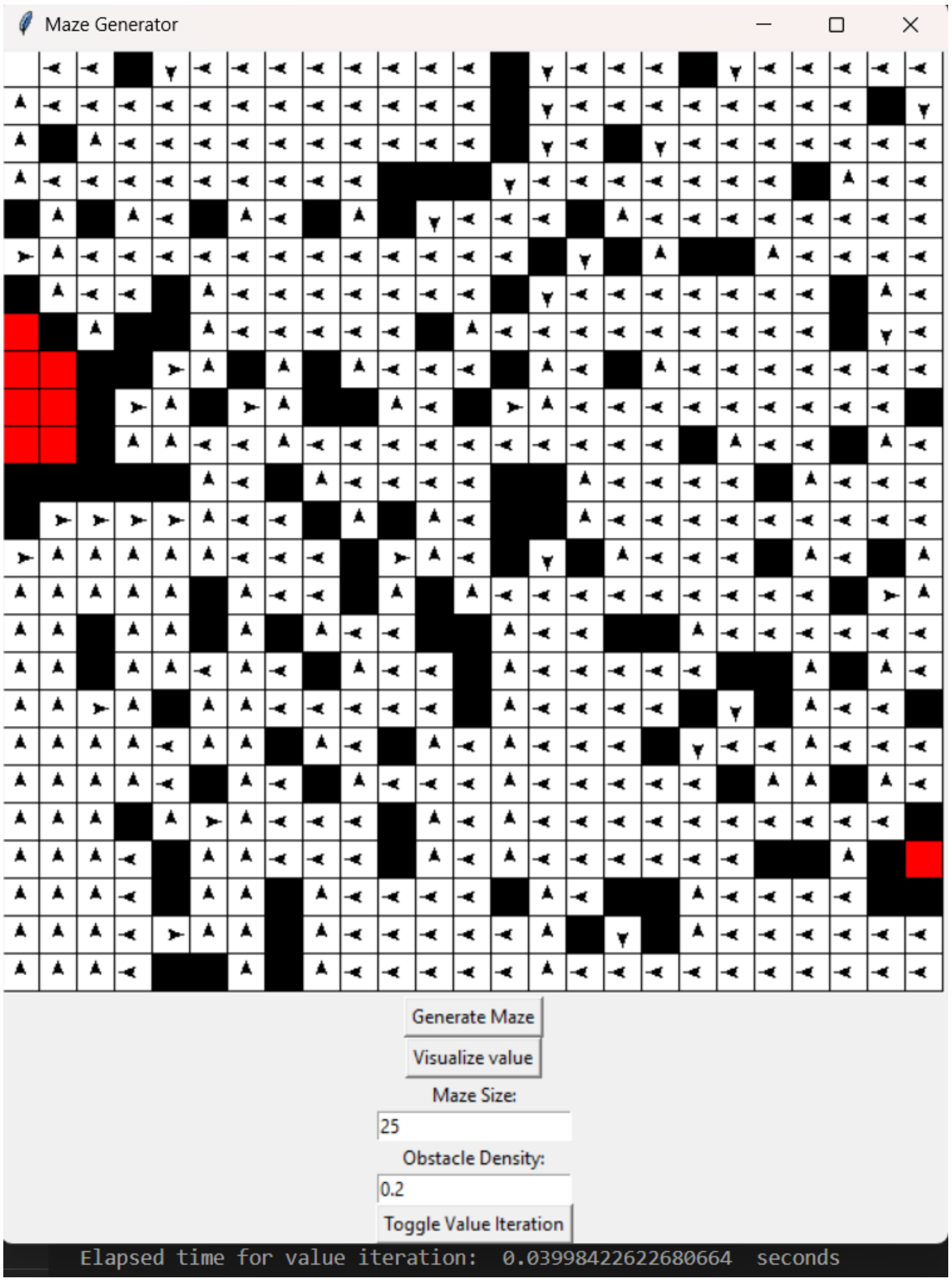
In the context of reinforcement learning, domain lists are employed to represent the possible actions or decisions that an agent can take in a given state. These lists define the action space and play a crucial role in shaping the agent's learning behavior. Lists offer a convenient structure for managing and updating the set of available actions.
These chosen data structures align with the requirements of reinforcement learning, providing an efficient representation of the maze environment, learned values, and available actions.

## 4 GUI

This project is centered around the Python implementation of a maze solver, featuring a user-friendly graphical user interface (GUI). The GUI seamlessly integrates data structure and employs sophisticated algorithms to facilitate the generation and solution of mazes.

In addition to maze solving functionality, the implementation includes a thorough analysis that calculates the runtime for both value and policy iteration methods. Furthermore, the project incorporates policy and value visualization mechanisms, offering insightful graphical representations to enhance understanding and comparison between the two approaches in the context of maze solving. These visualizations serve as valuable tools for analyzing decision-making policies and evolving values throughout the iterative processes, contributing to a deeper exploration of the efficiency and performance of each method in the maze-solving domain.
Here are some GUI samples:

- Value iteration with size 25 and 0.2 obstacles Density:

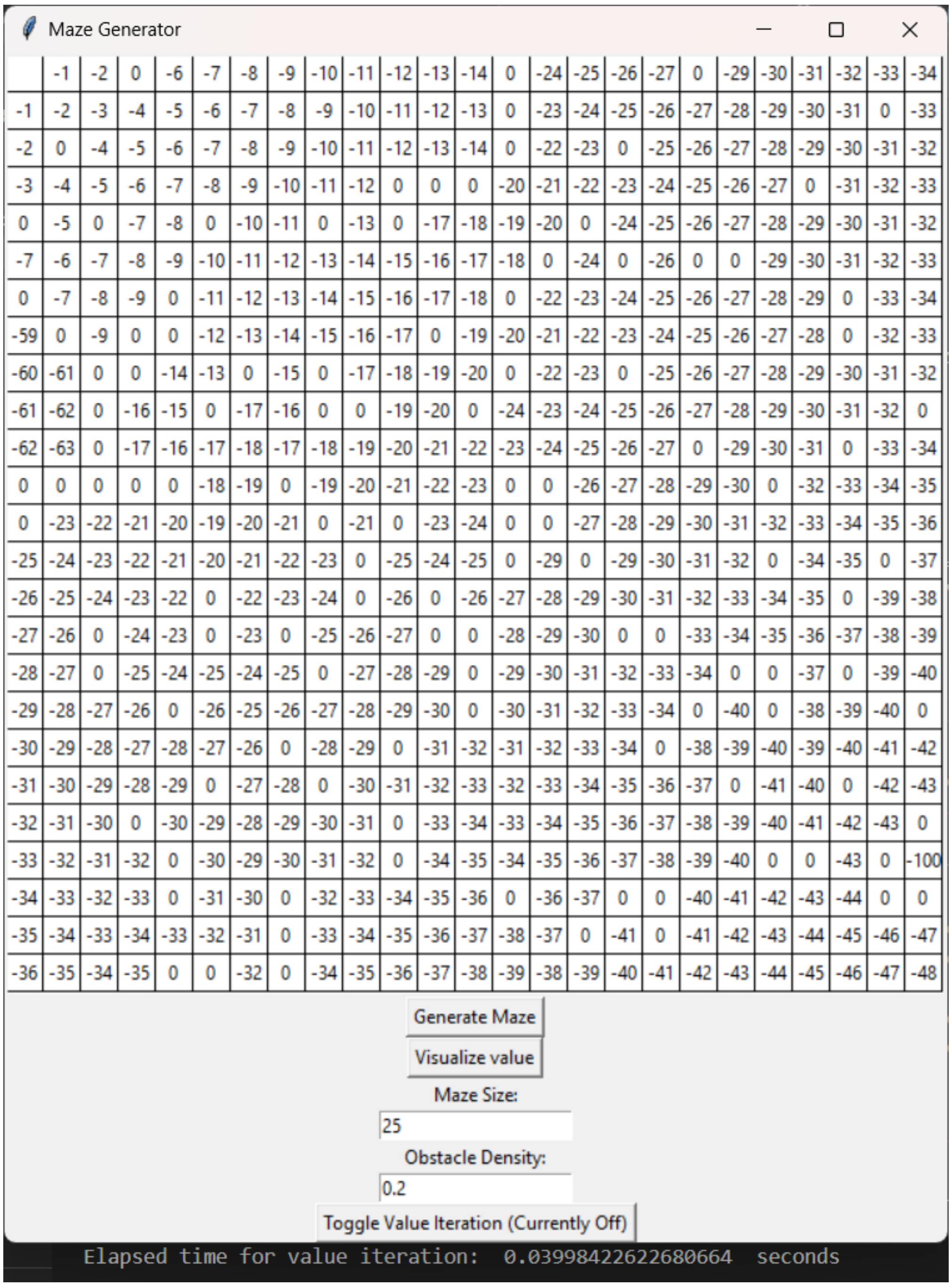


Maze Size:
25
Obstacle Density:
0.2

```
Elapsed time for value iteration:  0.03998422622680664  seconds
```

Figure 1: Policy Visualization.

- Value iteration with size 25 and 0.2 obstacles Density:



Maze Generator

| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | -1 | -2 | 0 | -6 | -7 | -8 | -9 | -10 | -11 | -12 | -13 | -14 | 0 | -24 | -25 | -26 | -27 | 0 | -29 | -30 | -31 | -32 | -33 | -34 |
| -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 | -9 | -10 | -11 | -12 | -13 | 0 | -23 | -24 | -25 | -26 | -27 | -28 | -29 | -30 | -31 | 0 | -33 |
| -2 | 0 | -4 | -5 | -6 | -7 | -8 | -9 | -10 | -11 | -12 | -13 | -14 | 0 | -22 | -23 | 0 | -25 | -26 | -27 | -28 | -29 | -30 | -31 | -32 |
| -3 | -4 | -5 | -6 | -7 | -8 | -9 | -10 | -11 | -12 | 0 | 0 | 0 | -20 | -21 | -22 | -23 | -24 | -25 | -26 | -27 | 0 | -31 | -32 | -33 |
| 0 | -5 | 0 | -7 | -8 | 0 | -10 | -11 | 0 | -13 | 0 | -17 | -18 | -19 | -20 | 0 | -24 | -25 | -26 | -27 | -28 | -29 | -30 | -31 | -32 |
| -7 | -6 | -7 | -8 | -9 | -10 | -11 | -12 | -13 | -14 | -15 | -16 | -17 | -18 | 0 | -24 | 0 | -26 | 0 | 0 | -29 | -30 | -31 | -32 | -33 |
| 0 | -7 | -8 | -9 | 0 | -11 | -12 | -13 | -14 | -15 | -16 | -17 | -18 | 0 | -22 | -23 | -24 | -25 | -26 | -27 | -28 | -29 | 0 | -33 | -34 |
| -59 | 0 | -9 | 0 | 0 | -12 | -13 | -14 | -15 | -16 | -17 | 0 | -19 | -20 | -21 | -22 | -23 | -24 | -25 | -26 | -27 | -28 | 0 | -32 | -33 |
| -60 | -61 | 0 | 0 | -14 | -13 | 0 | -15 | 0 | -17 | -18 | -19 | -20 | 0 | -22 | -23 | 0 | -25 | -26 | -27 | -28 | -29 | -30 | -31 | -32 |
| -61 | -62 | 0 | -16 | -15 | 0 | -17 | -16 | 0 | 0 | -19 | -20 | 0 | -24 | -23 | -24 | -25 | -26 | -27 | -28 | -29 | -30 | -31 | -32 | 0 |
| -62 | -63 | 0 | -17 | -16 | -17 | -18 | -17 | -18 | -19 | -20 | -21 | -22 | -23 | -24 | -25 | -26 | -27 | 0 | -29 | -30 | -31 | 0 | -33 | -34 |
| 0 | 0 | 0 | 0 | 0 | -18 | -19 | 0 | -19 | -20 | -21 | -22 | -23 | 0 | 0 | -26 | -27 | -28 | -29 | -30 | 0 | -32 | -33 | -34 | -35 |
| 0 | -23 | -22 | -21 | -20 | -19 | -20 | -21 | 0 | -21 | 0 | -23 | -24 | 0 | 0 | -27 | -28 | -29 | -30 | -31 | -32 | -33 | -34 | -35 | -36 |
| -25 | -24 | -23 | -22 | -21 | -20 | -21 | -22 | -23 | 0 | -25 | -24 | -25 | 0 | -29 | 0 | -29 | -30 | -31 | -32 | 0 | -34 | -35 | 0 | -37 |
| -26 | -25 | -24 | -23 | -22 | 0 | -22 | -23 | -24 | 0 | -26 | 0 | -26 | -27 | -28 | -29 | -30 | -31 | -32 | -33 | -34 | -35 | 0 | -39 | -38 |
| -27 | -26 | 0 | -24 | -23 | 0 | -23 | 0 | -25 | -26 | -27 | 0 | 0 | -28 | -29 | -30 | 0 | 0 | -33 | -34 | -35 | -36 | -37 | -38 | -39 |
| -28 | -27 | 0 | -25 | -24 | -25 | -24 | -25 | 0 | -27 | -28 | -29 | 0 | -29 | -30 | -31 | -32 | -33 | -34 | 0 | 0 | -37 | 0 | -39 | -40 |
| -29 | -28 | -27 | -26 | 0 | -26 | -25 | -26 | -27 | -28 | -29 | -30 | 0 | -30 | -31 | -32 | -33 | -34 | 0 | -40 | 0 | -38 | -39 | -40 | 0 |
| -30 | -29 | -28 | -27 | -28 | -27 | -26 | 0 | -28 | -29 | 0 | -31 | -32 | -31 | -32 | -33 | -34 | 0 | -38 | -39 | -40 | -39 | -40 | -41 | -42 |
| -31 | -30 | -29 | -28 | -29 | 0 | -27 | -28 | 0 | -30 | -31 | -32 | -33 | -32 | -33 | -34 | -35 | -36 | -37 | 0 | -41 | -40 | 0 | -42 | -43 |
| -32 | -31 | -30 | 0 | -30 | -29 | -28 | -29 | -30 | -31 | 0 | -33 | -34 | -33 | -34 | -35 | -36 | -37 | -38 | -39 | -40 | -41 | -42 | -43 | 0 |
| -33 | -32 | -31 | -32 | 0 | -30 | -29 | -30 | -31 | -32 | 0 | -34 | -35 | -34 | -35 | -36 | -37 | -38 | -39 | -40 | 0 | 0 | -43 | 0 | -100 |
| -34 | -33 | -32 | -33 | 0 | -31 | -30 | 0 | -32 | -33 | -34 | -35 | -36 | 0 | -36 | -37 | 0 | 0 | -40 | -41 | -42 | -43 | -44 | 0 | 0 |
| -35 | -34 | -33 | -34 | -33 | -32 | -31 | 0 | -33 | -34 | -35 | -36 | -37 | -38 | -37 | 0 | -41 | 0 | -41 | -42 | -43 | -44 | -45 | -46 | -47 |
| -36 | -35 | -34 | -35 | 0 | 0 | -32 | 0 | -34 | -35 | -36 | -37 | -38 | -39 | -38 | -39 | -40 | -41 | -42 | -43 | -44 | -45 | -46 | -47 | -48 |

Generate Maze

Visualize value

Maze Size: 25

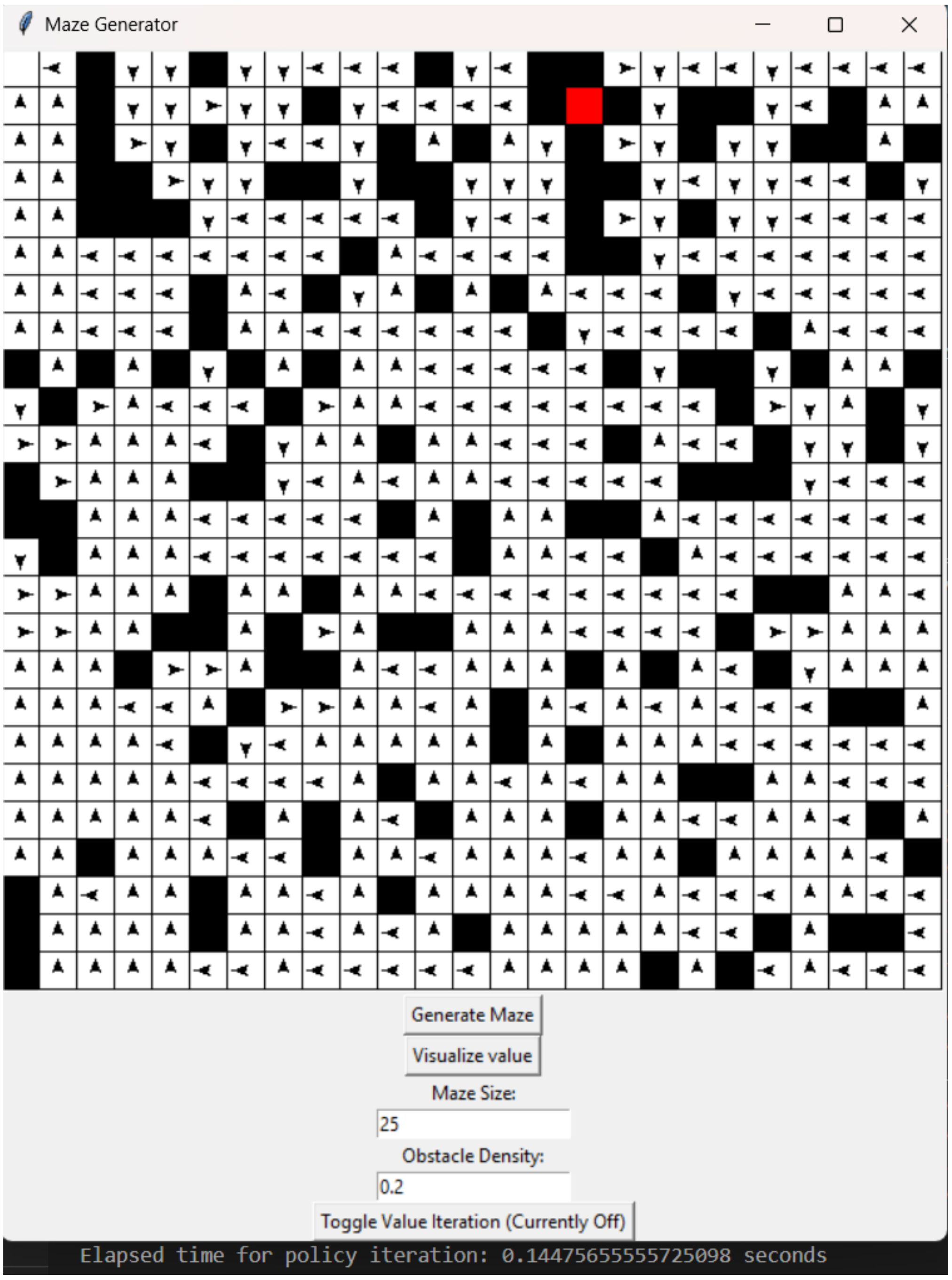
Obstacle Density: 0.2

Toggle Value Iteration (Currently Off)

```
Elapsed time for value iteration:  0.03998422622680664  seconds
```

Figure 2: Value Visualization.

- Policy iteration with size 25 and 0.2 obstacles Density:



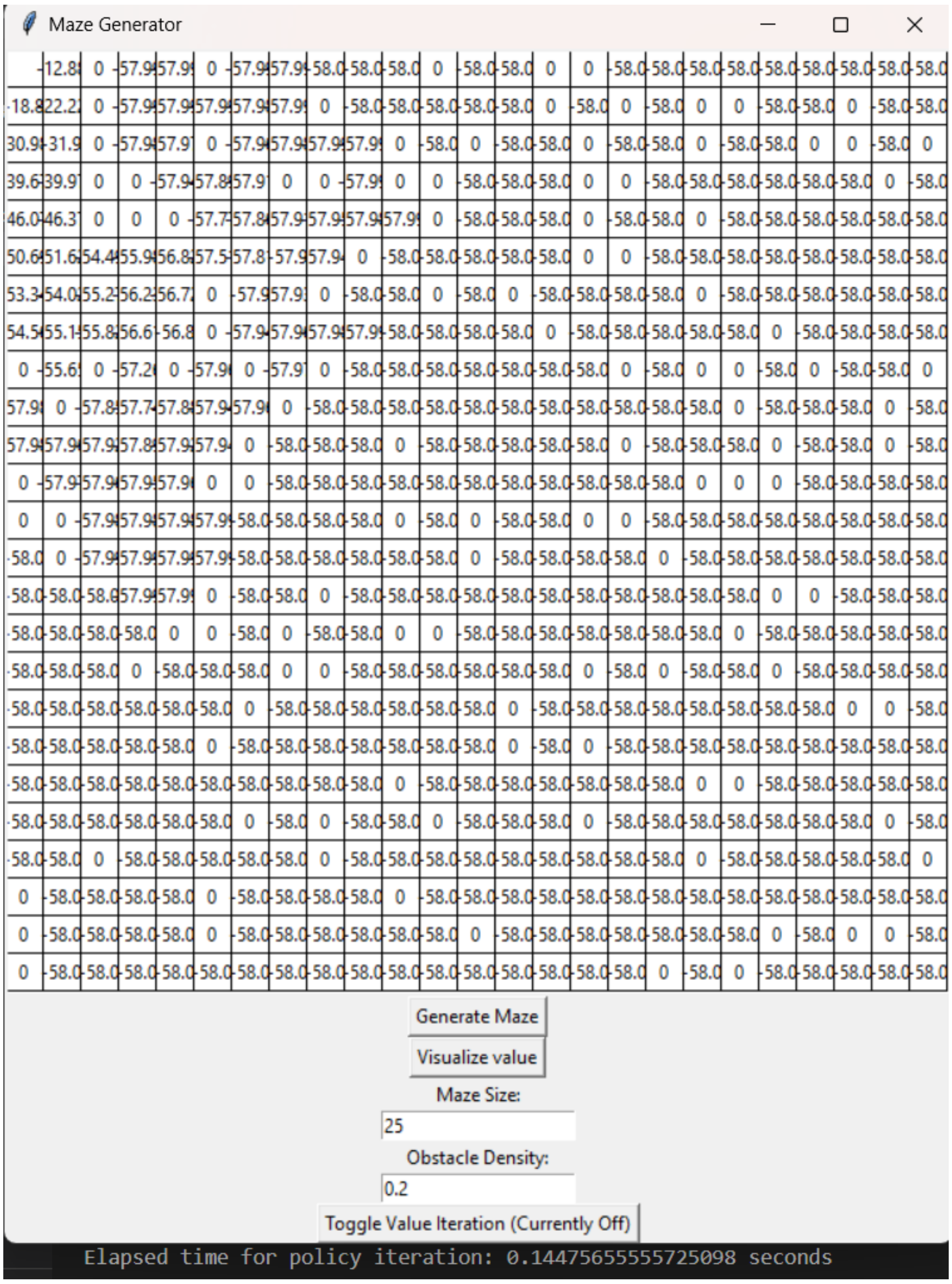
Maze Size: 25

Obstacle Density: 0.2

```
Elapsed time for policy iteration: 0.14475655555725098 seconds
```

Figure 3: Policy Visualization.

- Policy iteration with size 25 and 0.2 obstacles Density:



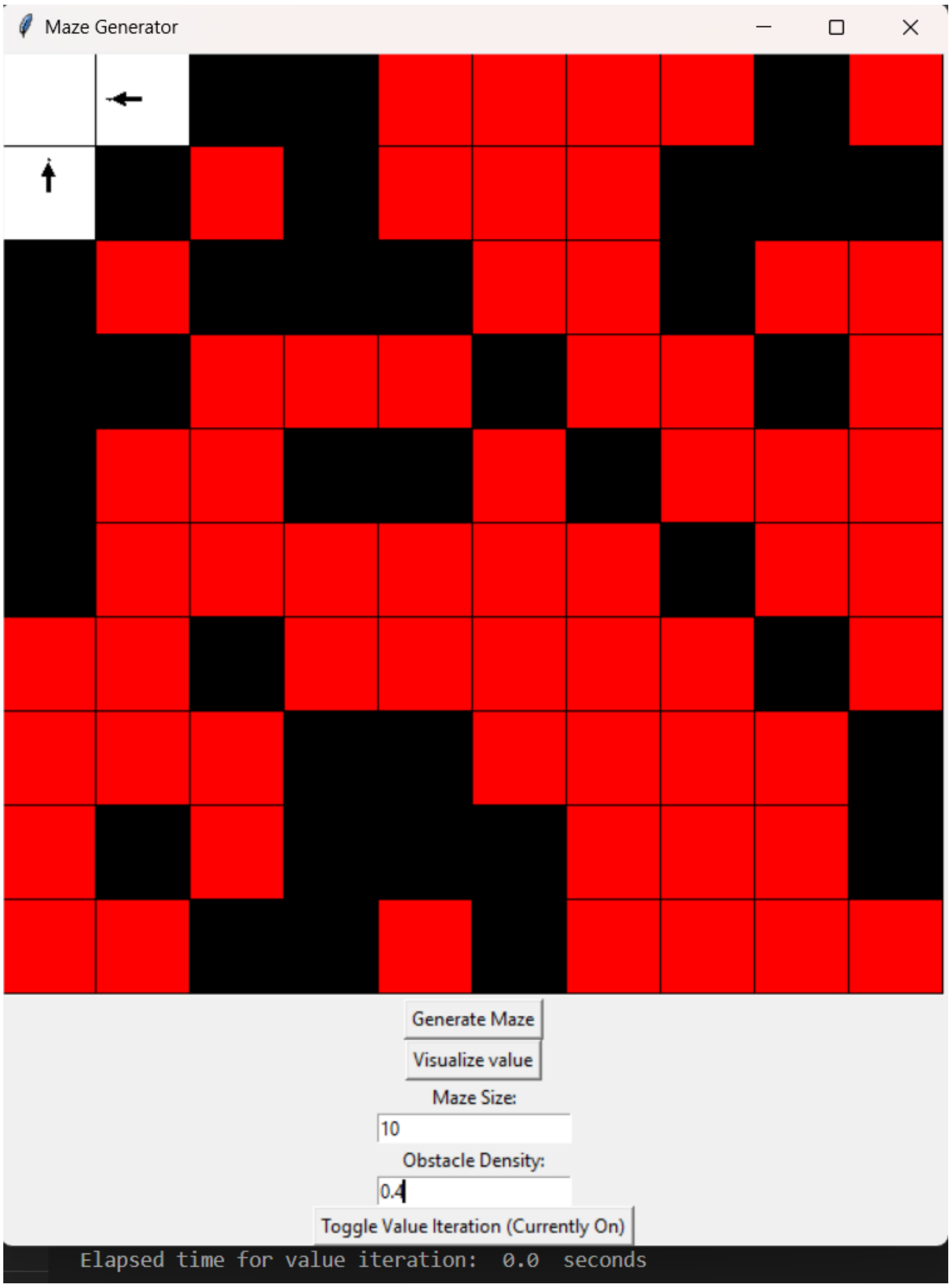
| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | -12.8 | 0 | -57.9 | -57.9 | 0 | -57.9 | -57.9 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | 0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 |
| -18.8 | -22.2 | 0 | -57.9 | -57.9 | -57.9 | -57.9 | -57.9 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | 0 | -58.0 | 0 | 0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 |
| -30.9 | -31.9 | 0 | -57.9 | -57.9 | 0 | -57.9 | -57.9 | -57.9 | -57.9 | 0 | -58.0 | 0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | 0 | 0 | -58.0 | 0 |
| -39.6 | -39.9 | 0 | 0 | -57.9 | -57.8 | -57.9 | 0 | 0 | -57.9 | 0 | 0 | -58.0 | -58.0 | -58.0 | 0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 |
| -46.0 | -46.3 | 0 | 0 | 0 | -57.7 | -57.8 | -57.9 | -57.9 | -57.9 | -57.9 | 0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 |
| -50.6 | -51.6 | -54.4 | -55.9 | -56.8 | -57.5 | -57.8 | -57.9 | -57.9 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 |
| -53.3 | -54.0 | -55.2 | -56.2 | -56.7 | 0 | -57.9 | -57.9 | 0 | -58.0 | -58.0 | 0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 |
| -54.5 | -55.1 | -55.8 | -56.6 | -56.8 | 0 | -57.9 | -57.9 | -57.9 | -57.9 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 |
| 0 | -55.6 | 0 | -57.2 | 0 | -57.9 | 0 | -57.9 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | 0 | 0 | -58.0 | 0 | -58.0 | -58.0 | 0 |
| -57.9 | 0 | -57.8 | -57.7 | -57.8 | -57.9 | -57.9 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 |
| -57.9 | -57.9 | -57.9 | -57.8 | -57.9 | -57.9 | 0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | 0 | -58.0 |
| 0 | -57.9 | -57.9 | -57.9 | -57.9 | 0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | 0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 |
| 0 | 0 | -57.9 | -57.9 | -57.9 | -57.9 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | 0 | -58.0 | -58.0 | 0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 |
| -58.0 | 0 | -57.9 | -57.9 | -57.9 | -57.9 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 |
| -58.0 | -58.0 | -58.0 | -57.9 | -57.9 | 0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | 0 | -58.0 | -58.0 | -58.0 |
| -58.0 | -58.0 | -58.0 | -58.0 | 0 | 0 | -58.0 | 0 | -58.0 | -58.0 | 0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 |
| -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | 0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | 0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 |
| -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | 0 | -58.0 |
| -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 |
| -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 |
| -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | 0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 |
| -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 |
| 0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 |
| 0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | 0 | 0 | -58.0 |
| 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 | 0 | -58.0 | 0 | -58.0 | -58.0 | -58.0 | -58.0 | -58.0 |

Maze Size:
25
Obstacle Density:
0.2

```
Elapsed time for policy iteration: 0.14475655555725098 seconds
```

Figure 4: Value Visualization.

- Value iteration with size 10 and 0.4 obstacles Density:



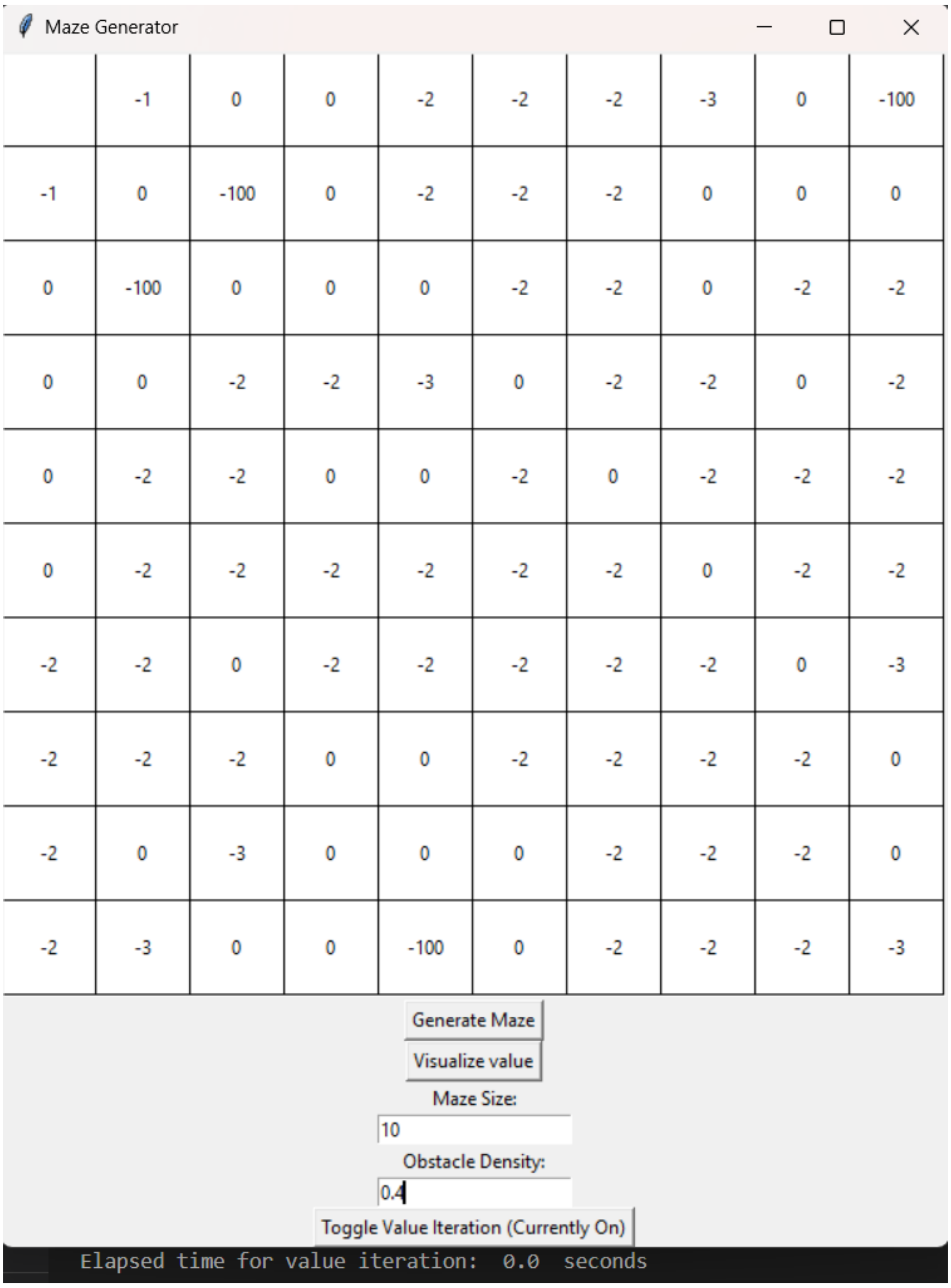
Maze Size: 10

Obstacle Density: 0.4

```
Elapsed time for value iteration:  0.0  seconds
```

Figure 5: Policy Visualization.

- Value iteration with size 10 and 0.4 obstacles Density:



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | -1 | 0 | 0 | -2 | -2 | -2 | -3 | 0 | -100 |
| -1 | 0 | -100 | 0 | -2 | -2 | -2 | 0 | 0 | 0 |
| 0 | -100 | 0 | 0 | 0 | -2 | -2 | 0 | -2 | -2 |
| 0 | 0 | -2 | -2 | -3 | 0 | -2 | -2 | 0 | -2 |
| 0 | -2 | -2 | 0 | 0 | -2 | 0 | -2 | -2 | -2 |
| 0 | -2 | -2 | -2 | -2 | -2 | -2 | 0 | -2 | -2 |
| -2 | -2 | 0 | -2 | -2 | -2 | -2 | -2 | 0 | -3 |
| -2 | -2 | -2 | 0 | 0 | -2 | -2 | -2 | -2 | 0 |
| -2 | 0 | -3 | 0 | 0 | 0 | -2 | -2 | -2 | 0 |
| -2 | -3 | 0 | 0 | -100 | 0 | -2 | -2 | -2 | -3 |

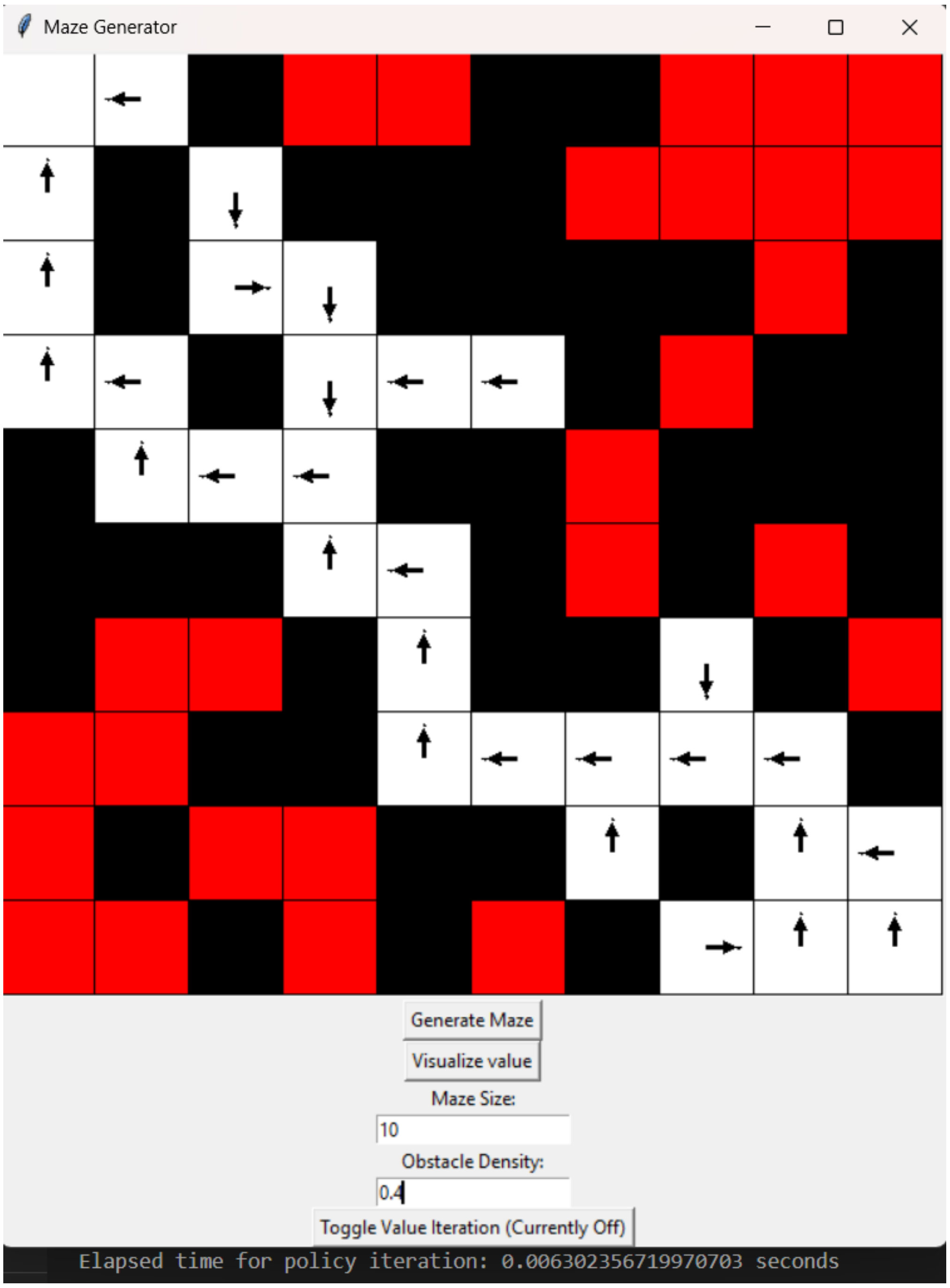
Maze Size: 10

Obstacle Density: 0.4

```
Elapsed time for value iteration:  0.0  seconds
```

Figure 6: Value Visualization.

- Policy iteration with size 10 and 0.4 obstacles Density:



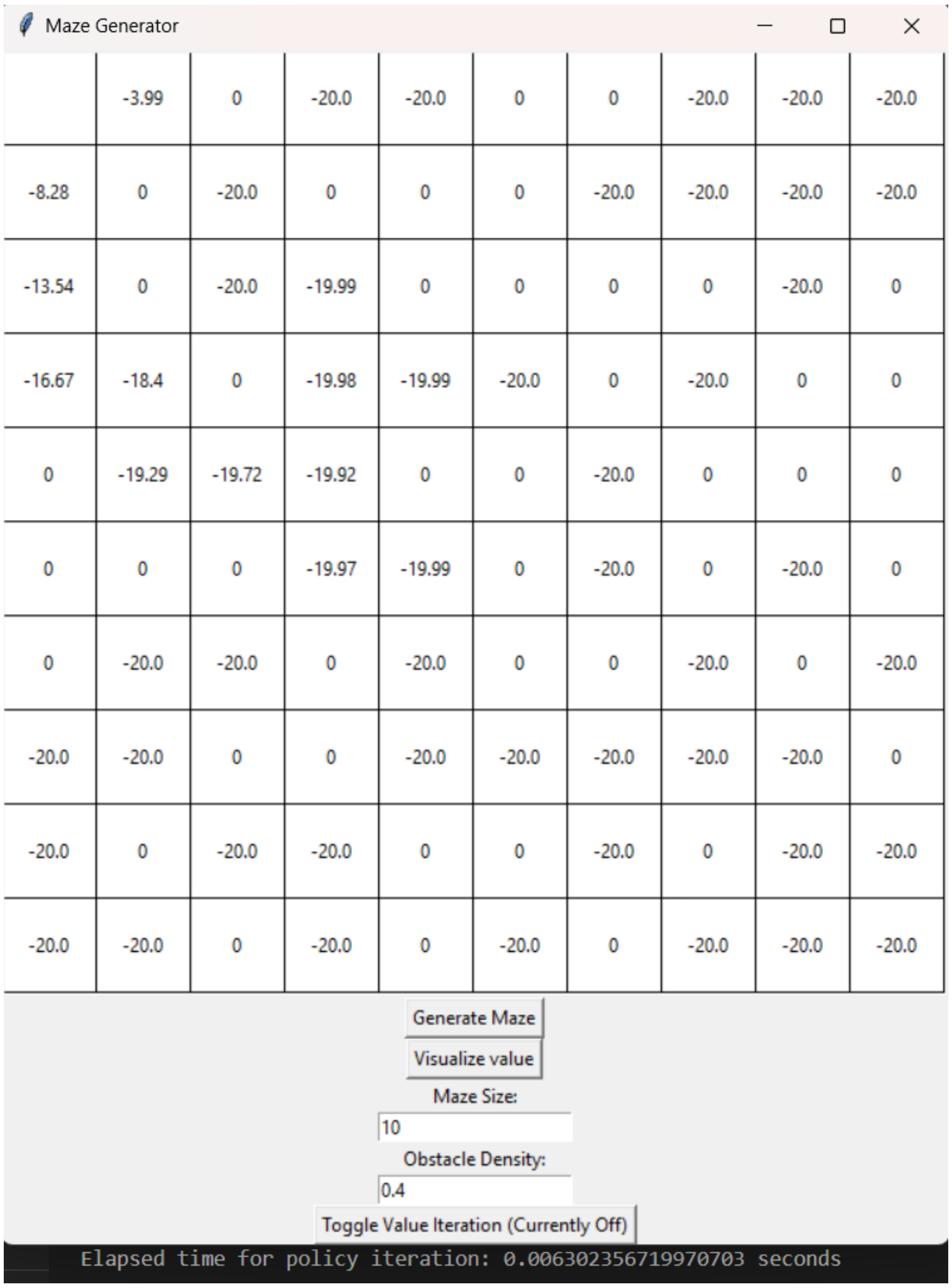
Maze Size: 10

Obstacle Density: 0.4

```
Elapsed time for policy iteration: 0.006302356719970703 seconds
```

Figure 7: Policy Visualization.

- Policy iteration with size 10 and 0.4 obstacles Density:



| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | -3.99 | 0 | -20.0 | -20.0 | 0 | 0 | -20.0 | -20.0 | -20.0 |
| -8.28 | 0 | -20.0 | 0 | 0 | 0 | -20.0 | -20.0 | -20.0 | -20.0 |
| -13.54 | 0 | -20.0 | -19.99 | 0 | 0 | 0 | 0 | -20.0 | 0 |
| -16.67 | -18.4 | 0 | -19.98 | -19.99 | -20.0 | 0 | -20.0 | 0 | 0 |
| 0 | -19.29 | -19.72 | -19.92 | 0 | 0 | -20.0 | 0 | 0 | 0 |
| 0 | 0 | 0 | -19.97 | -19.99 | 0 | -20.0 | 0 | -20.0 | 0 |
| 0 | -20.0 | -20.0 | 0 | -20.0 | 0 | 0 | -20.0 | 0 | -20.0 |
| -20.0 | -20.0 | 0 | 0 | -20.0 | -20.0 | -20.0 | -20.0 | -20.0 | 0 |
| -20.0 | 0 | -20.0 | -20.0 | 0 | 0 | -20.0 | 0 | -20.0 | -20.0 |
| -20.0 | -20.0 | 0 | -20.0 | 0 | -20.0 | 0 | -20.0 | -20.0 | -20.0 |

Maze Size: 10

Obstacle Density: 0.4

```
Elapsed time for policy iteration: 0.006302356719970703 seconds
```

Figure 8: Value Visualization.

- As seen in the previous figures the value iteration was always faster then the policy iteration as it reach convergence faster than policy does.