Dynamic Programming on Custom FrozenLake Environment

Divyansh Soni RollNo: 24124018

June 24, 2025

1 Introduction

This project implements and compares Dynamic Programming (DP) algorithms — Value Iteration and Policy Iteration — on a custom version of the FrozenLake environment using the Gymnasium library. The goal is to derive an optimal policy for navigating from the start state to the goal state while avoiding holes.

2 Environment Design

A custom 8x8 FrozenLake-like environment was implemented by subclassing `gym.Env`. The grid is represented as a list of strings with characters:

- `S` - Start
- `F` - Frozen tile
- `H` - Hole
- `G` - Goal

The transition dynamics are deterministic: the agent moves in the selected direction unless it hits the wall. Falling in a hole or reaching the goal ends the episode.

3 Algorithms Implemented

3.1 Value Iteration

Value Iteration uses the Bellman optimality equation to update the value function. The update continues until the maximum change in value is below a threshold θ. Once the value function converges, the optimal policy is extracted.

3.2 Policy Iteration

Policy Iteration consists of two steps: Policy Evaluation (compute V^π) and Policy Improvement (greedily update policy). These steps are repeated until the policy becomes stable.

4 Evaluation

The evaluation was performed by running 100 episodes and calculating:

- **Success Rate**: Number of successful episodes reaching the goal
- **Average Reward**: Mean reward over episodes
- **Convergence Iterations**: Number of iterations until convergence (for policy iteration)

5 Results

Method	Success Rate	Avg Reward	Iterations
Value Iteration	1.0	1.0	—
Policy Iteration	1.0	1.0	15

5.1 Policy Visualization

The learned policies were visualized using arrows (←, →, ↑, ↓) for each state. Below is the resulting arrow map for each algorithm:

5.1.1

Value Iteration Policy

```
↓   ↓   ↓   ↓   ↓   ↓   ↓   ↓
↓   ↓   ↓   →   ↓   ↓   ↓   ↓
↓   ↓   ↓   H   →   →   ↓   ↓
↓   →   ↓   ↓   ←   H   ↓   ↓
↓   H   ↓   ↓   H   ↓   →   ↓
↓   H   ↓   ↓   ↓   ↓   H   ↓
→   →   →   →   ↓   ↓   H   ↓
→   →   ↑   H   →   →   →   G
```

5.1.2

Policy Iteration Policy

```
↓   ↓   ↓   ↓   ↓   ↓   ↓   ↓
↓   ↓   ↓   →   ↓   ↓   ↓   ↓
↓   ↓   ↓   H   →   →   ↓   ↓
↓   →   ↓   ↓   ←   H   ↓   ↓
↓   H   ↓   ↓   H   ↓   →   ↓
↓   H   ↓   ↓   ↓   ↓   H   ↓
→   →   →   →   ↓   ↓   H   ↓
→   →   ↑   H   →   →   →   G
```

6 Conclusion

Both DP methods performed optimally on the custom environment. Policy Iteration converged in 15 iterations, while Value Iteration's iteration count was not tracked but can be added for analysis. The environment and code are modular and can be extended to larger maps or stochastic settings.