Assignment 1: Implementing and Analyzing a Basic RL Algorithm

Due: Sunday May 26, before midnight EST

Objective:

To gain hands-on experience in implementing a basic Markov Decision Process (MDP) and applying a Dynamic Programming algorithm, such as Policy Iteration or Value Iteration.

Tasks:

I. Implement a Gridworld Environment: **(20 points)**
   a. Create a gridworld environment, a simple and commonly used model in RL.
   b. Define states, actions, transition probabilities, and rewards.
   c. Include varied rewards, with some states having negative rewards (obstacles) and others positive rewards (goals).

II. A linear solver to solve the system Ax = b for the deterministic case. **(10 points)**

III. Apply a Dynamic Programming Algorithm: **(30 points)**
   a. Implement Policy Iteration or Value Iteration to find an optimal policy for navigating the gridworld.
   b. Ensure your implementation accurately reflects the theoretical principles of the chosen algorithm.

IV. Analyze Algorithm Convergence: **(10 points)**
   a. Examine the number of iterations your algorithm takes to converge.
   b. Discuss factors affecting convergence speed and policy quality.

Submission:

Python Code: Fully documented code implementing the gridworld and the Dynamic Programming algorithm. **(The points divided in the tasks total of 70 points)**

Report: A comprehensive report detailing the implementation, the algorithm's results, and an analysis of its convergence and effectiveness. **(30 points)**

The Problem Description:

Consider the following Gridworld example. There are 4 different actions (north, south, east, west). for all the states (cells in the grid), each one of the actions (north, south, east, and west) is chosen with probability ¼. The agent then moves with probability 1 to the chosen direction. While moving, if the agent hits a wall to go off-grid or the black cells (blocked), it cannot move and it receives a reward of 0; if moving to a cell in the grid, the reward is 0.; if it reaches red cells (fire), the game will terminate and receives a penalty -5; and, if it reaches the goal cell (G) it receives a reward +5 and the game will terminate.

State Space: Each cell in the grid represents a state. The agent can be in any cell except for those marked as blocked.

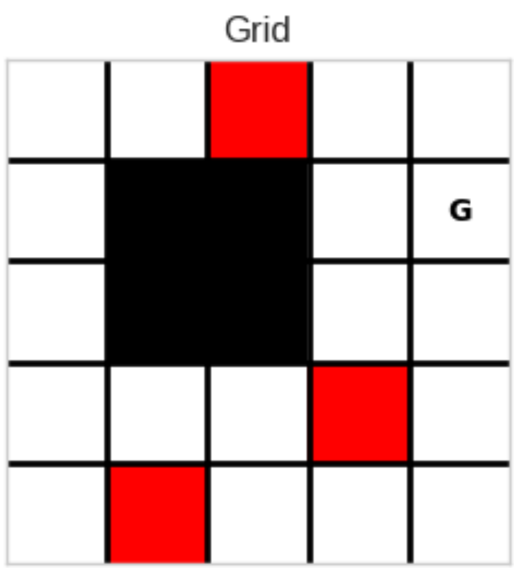
Actions: The agent can move up, right, down, or left. Moving into a blocked cell is not allowed, and attempting to move outside the grid bounds results in staying in the same cell.

Transition Model: The agent moves in the chosen direction with a probability of "1 - noise". With probability noise, it may end up moving in a different direction.

Rewards: The agent receives a specific reward (positive or negative) when it enters a cell. Moving into a danger cell yields a negative reward, reaching the goal cell yields a positive reward, and other moves typically have no reward (zero).

Termination: The episode ends if the agent reaches the goal cell or a danger cell.

Note: You are given a code snippet. Feel free to use it or use your own implementation.



You are required to write 2 programs (in Python 3) to find the state-value for each one of the states for discount rates of 0.95 and 0.75. You are also required to find the state-value for each state when the grid is noisy (probability of resulting state not being what was expected).

The 2 programs you will write are:

1. A linear solver to solve the system Ax = b for only deterministic case.
2. A dynamic programming approach (policy iteration or value iteration).