E1 277 Reinforcement Learning
Programming Assignment 1

January 2023

1 Problem Statement

A maze runner agent is trapped in a rectangular maze built up of walls. It needs to navigate the maze in order to reach goal state(s). The agent can take any one of the four actions among $left$, $right$,up, $down$ at every time step. Depending on the action taken, agent will move to the corresponding cell or will stay in the same cell if it collides with the walls. There are several doors in the maze that can open only with a lockpick. The agent also has a lockpick of infinite uses but is not an expert in lock-picking. If it tries to lockpick a door, there is only a certain probability that it will succeed and will move to the other side of the door, otherwise, it will stay in the same cell. The doors are 2-way. i.e., they can open from both sides. The agent incurs a penalty of 1 unit for each time step that it spends in the maze except when in a terminal state(s) where it gets a finite positive reward. Once the agent reaches the goal state, it stays in the same state and keeps on getting the positive reward. The agent has only a fixed amount of time $horizonLength$ units to traverse the maze.
Your task is to calculate the optimal reward that the agent can get in N steps from given initial cell of the maze.

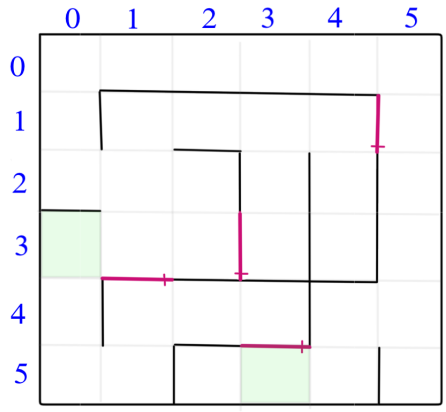

Figure 1: Sample Maze. Pink lines indicate doors, green cells indicate goal states.

- For the above maze, indexing from 0 (starting from left), At state (1,5) if the agent decides to move up, he goes to (0,5) and will get a reward of -1.
- At state (1,5) if the agent decides to move left, with probability p it goes to (1,4) and with probability $1-p$ will stay at (1,5) getting a reward of -1 in both the cases.
- An agent at (3,1), on taking the action $left$ goes to goal state (3,0) and receives a positive reward $terminalReward$.

- An agent at (3,0) (goal state), will stay at (3,0) irrespective of what action it takes and gets a positive reward $terminalReward$.

2 Implementation Details

Your task is to implement the function *optimalReward*() that takes current state s and time-step k and returns maximum reward that the agent can receive in $horizonLength$ time-steps starting from k^{th} time-step in state s using dynamic-programming. $0 \leq k < horizonLength$.
An agent can take one of the actions from $actionSpace = [left, right, up, down]$.
$takeAction()$ function from $Maze$ class takes as input current state : tuple (i, j) and action a; i,j being the current cell indices and $a \in actionSpace$ and returns list of lists of the form [next state, probability,reward]. For example, in the given maze, $takeAction((1, 5), left)$ will return $[[(1, 4), p, -1], [(1, 5), 1 - p, -1]]$.

3 Instructions for Submission

- Write your code only in the space indicated in the template file $RL\_Assignment1.py$.
- Try not to hard-code any variable values. The hidden test cases can be of different environment and your code may fail if any hard-coding is present.
- Submit your solution code as $RL\_Assignment1\_$ <Last 5 digits of your SR No.> $.py$
- We have very strong plagiarism checks in place. Any malpractice would lead to severe consequences as per the institute policy.