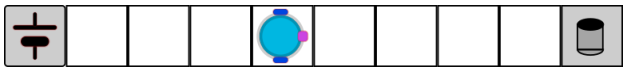
Homework 3: Dynamic Programming
Due: Monday, October 4th 11:59 pm

The purpose of this project is to study different properties of dynamic programming methods.

Part I

Consider a cleaning robot that must collect an empty can and also has to recharge its batteries.



This problem has a discrete state space $S = \{0, \dots, 9\}$, where state s describes the position of the robot in the corridor. The robot has only two actions $A = \{-1, 1\}$ for going one step to the left or right. States 0 and 9 are terminal, meaning that once the robot reaches either of them it can no longer leave, regardless of the action, and the episode ends. We assume this is a deterministic environment with $\gamma = 0.9$.

1. Write a function that describes the transition function $\acute{s} = T(s, a)$ and test it for a few state-action pairs including terminal states. Note that the transition function describes robot's environment model and should not allow the robot to move outside the environment.
2. Write a function that gives a reward of +5 for being at $s = 8$ and taking $a = 1$; a reward +1 for at $s = 1$ and taking $a = -1$; and reward 0 otherwise.
3. Implement the value iteration algorithm to find $V^*(s)$ and $\pi^*(s)$. Note that instead of using the dynamics of the environment $p(\acute{s}, r|s, a)$ you should use the deterministic transition function $T(s, a)$ that allows you to remove looping over possible rewards and the probability of transition (because it is a deterministic function). Print out the final $V^*(s)$ and $\pi^*(s)$.

Part II (*)

A gambler can make bets on the outcomes of a sequence of coin flips. If the coin comes up **heads**, he wins as many dollars as he has staked on that flip; if it is **tails**, he loses his stake. The game ends when the gambler wins by reaching his goal of $100 or loses by running out of money.

On each flip, the gambler must decide what portion of his capital to stake, in integer numbers of dollars. This problem can be formulated as an **undiscounted** ($\gamma = 1$), **episodic**, finite MDP. The state is the gambler's capital $s \in \{1, 2, \dots, 99\}$ and the actions are stakes $a \in \{0, 1, \dots, \min(s, 100 - s)\}$. The reward is zero on all transitions except those on which the gambler reaches his goal, then it is +1.

The state-value function for this problem gives the probability of winning from each state. A policy is mapping from levels of capital to stakes (state to action). And the optimal policy maximizes the probability of reaching the goal.

Let's p_h denote the probability of the coin coming up heads. If p_h is known the problem can be solved using value iteration.

1. Implement the Gambler's problem and then implement **value iteration** to solve the MDP for three scenarios where $p_h = \{0.4, 0.25, 0.55\}$ and find the optimal value function and optimal policy. **Tip:** When implementing, you might find it convenient to introduce two

dummy states corresponding to termination with capital of 0 and 100, giving them values of 0 and 1 respectively.

2. For all three scenarios:
   a. Plot the change in the value function over successive sweeps of value iteration w.r.t capital (state).
   b. Plot the final policy w.r.t capital (state).

3. Answer the following questions:
   a. What action does your optimal policy suggest for capital of 50? What about for capital of 51?
   b. Why do you think your optimal policy is a good policy? Explain.

Part III (extra credits)

Test the algorithm in Part II by decreasing θ the threshold for accuracy of value function estimation. What happens when $\theta \to 0$.

Evaluation: we will grade your submission according to the following table:

Item	COMP4600	COMP5300
Implement transition and reward functions in Part I	40	20
Implement Value Iteration with correct results in Part I	60	20
Implementation of the problem in Part II	-	20
Implementation of value iteration in Part II	-	20
Plotting value and optimal policy	-	20
Answering questions in Part III (extra credit)	10	10

Note 1: The parts marked with * are optional for COMP4600 (undergraduates) but mandatory for COMP5500 (graduates).

Note 2: We do not accept code in any programming language other than Python 3 (do not use Python 2).

Note 3: For the implementation of the testbed and the algorithms, you are not allowed to rely on any python library other than `numpy` and `matplotlib` (for plotting).

Note 4: All the code, plots, explanations should be included in a single Jupyter Notebook (`.ipynb`) file. Include your name as part of the filename and submit through Blackboard.

Submission: By 11:59pm on Monday, October 4th 2021, submit your `student_name.ipynb` files on Blackboard. Make sure everything is entirely contained within this file and it runs without any error.

Late Policy: Up to two late days are allowed, but a grade penalty of 50% and 75% will be applied at the first and second day, respectively.