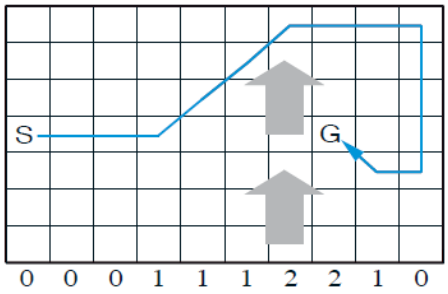
Homework 5: Temporal Difference
Due: Wednesday, October 14th 11:59 pm

The purpose of this project is to study different properties of Temporal Difference methods.

Part I

The following Windy Gridworld includes a crosswind running upward through the middle of the grid. The strength of wind is shown under each column that you should use for shifting. The goal is to reach the goal state G from the start state S. This is an undiscounted episodic task, with constant rewards of -1 until the goal state is reached.



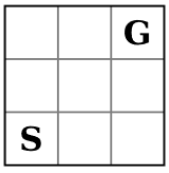
Implement SARSA (on-policy TD control) using ε–greedy policy with parameters $\varepsilon = 0.1, \alpha = 0.5, Q_0(s,a) = 0$ for all s, a. Apply your implementation to the windy gridworld in the following scenarios:

a. In this scenario, the agent's action set includes four moves (i.e. Up, Down, Right, Left). Plot the number of episodes vs. time steps (similar to graph in page 130). Then calculate the average episode length.
b. In this scenario, the agent has access to King's moves (eight actions). You need to define a new action set and re-run the experiment in previous scenario. Plot the number of episodes vs. time steps. Then calculate the average episode length.
c. In this scenario, you need to add a ninth action that causes no movement at all other than that caused by the wind. Plot the number of episodes vs. time steps. Then calculate the average episode length.
d. In this scenario, the agent has access to King's moves (eight actions). Assume that the effect of the wind, if there is any, is stochastic, sometimes varying by 1 from the mean values given for each column. That is, a third of the time you move exactly according to these values, as in the previous part, but also a third of the time you move one cell above that, and another third of the time you move one cell below that. For example, if you are one cell above the goal, one-third of the time you move two cells above the goal, and one-third of the time you move to the goal. Plot the number of episodes vs. time steps. Then calculate the average episode length.

Note: all four plots should be shown in one figure for comparison.

Part II

Consider the following gridworld with four actions (up, down, right, and left). If the action takes the agent off the gird, the agent stays in the same state. In each non-terminating step, the agent receives a random reward of -12 or +10 with equal probability. The reward for reaching the goal state (G) is +5 and the episode ends when the agent reaches the goal.



Use ε–greedy policy with $\varepsilon(s) = 1/\sqrt{n(s)}$ where $n(s)$ is the number of times state s has been visited, assuring infinite exploration in the limit which is a theoretical requirement for the convergence of both Q-learning and Double Q-learning.

a. Implement Q-learning and Double Q-learning and apply them to this problem for 10,000 experiments using the learning rate $\alpha = 1/n(s,a)$.
b. Plot the average reward per step vs. number of time steps averaged over 10,000 experiments. The length of an episode following the optimal policy is five actions, so the optimal average reward per step is +0.2. Plot this true value in your figure and see how close your algorithm gets to the true value.
c. Plot the maximal action value in the starting state S (i.e. $\max_a Q(S,a)$) averaged over 10,000 experiments. The optimal value of maximally valued action in the starting state is $5\gamma^4 - \sum_0^3 \gamma^k \approx 0.36$. Plot this true value in your figure and see how close your algorithm gets to the true value.
d. Repeat the experiments with $\alpha = 1/n(s,a)^{0.8}$ and redo steps b and c.

Note: Double Q-learning has two learning rates that should be updated separately. In other words, if you update Q_1 then the learning rate for this update formula should be updated.

Answer the following questions:
a. Which algorithm finds a better policy? Why?
b. Which learning rate performs better?

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

Item	COMP4600	COMP5300
Part I	40	40
Part II	60	60

Note 1: The parts marked with * are optional for COMP4600 (undergraduates) but mandatory for COMP5300 (graduates). There are no optional parts in this homework.

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 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 Wednesday, October 14th 2020, submit both your `student_name.ipynb` files on Blackboard. Make sure everything is entirely contained within this file and it runs without any error.