The objective of this homework assignment is to get a taste of implementing several classic RL algorithms. Please do not use any code that you find online. You may collaborate with others, but your code and your write-up must be your own. Include the names of your collaborators at the top of your assignment. Include your code at the end of the assignment. You should submit your assignment on Canvas.

Setting. Download the skeleton files “gridworld.py” and “homework1.py” and familiarize yourself with them. Next, implement the following basic RL algorithms: SARSA, Q-Learning, RMAX, and Thompson Sampling (also known as PSRL/Posterior Sampling Reinforcement Learning). You will use $\gamma = .95$ throughout the assignment.

Problem 1: Gridworld. For the gridworld domain, try each of the following parameter settings for the different algorithms:

1. SARSA: $\alpha = 1/\text{current-iteration}$; $\alpha = \min(.5, 10/\text{current-iteration})$; alpha = .1 let $\epsilon$ in the $\epsilon$-greedy policy be .1 and the initial Q-values be 0.

2. Q-Learning: $\alpha = 1/\text{current-iteration}$; $\alpha = \min(.5, 10/\text{current-iteration})$; alpha = .1 let $\epsilon$ in the $\epsilon$-greedy policy be .1 and the initial Q-values be 0.

3. RMAX: min visit count = 5, 10, 50. Recompute the policy every 50 iterations.

4. Thompson Sampling: Dirichlet params = 1, Reward params: start all $R(s,a,s) = 50$; Dirichlet params = 1, Reward params: start all $R(s,a,s) = 10$; Dirichlet params = 10, Reward params: start all $R(s,a,s) = 50$. Recompute the policy every 50 iterations. Assume that the rewards $R(s,a,s)$ are deterministic, so once you observe a reward you can update to the true value.

Run 50 trials for each pair of learner and parameter setting. Let each trial run for at least 5000 iterations of experience and at least 100 episodes (that is, the number of iterations will be max( 5000 , iterations it takes to run 100 episodes )). Make the following plots:

Cumulative Rewards on Gridworld. Plot the cumulative reward with respect to the number of iterations for each case. Use a different color for each algorithm, and a different line-type for the different parameter settings.

Episode Rewards in Gridworld. Plot the total reward per episode. Use a different color for each algorithm, and a different line-type for different parameter settings.

Discuss whether certain algorithms seem to perform better or worse in general. How do parameter choices affect the rate of learning and quality of optima reached? Hypothesize reasons for what you observe.

Aside: Error bars may make your graphs too cluttered; if so, you may omit them in the report, but you should investigate these on your own; increase the number of trials if you believe that the error bars are too large to discuss the main trends you observe in the plots.
Problem 2: Cliffworld  

Apply your algorithms with a reasonable set of parameter settings to the cliffworld domain. Make the following plots:

*Cumulative rewards vs. action-error probability.* This is the sum of all the rewards in a trial (each trial produces a single number). This metric captures how quickly the agent learns; an agent that learns quickly—and thus spends more time exploiting—will gather more cumulative rewards. Your plot should have the action-error probability as the x-axis and the cumulative reward as the y-axis. Each algorithm will be a different-colored curve. Use different line-weights to indicated the different settings of epsilon.

*End rewards vs. action-error probability.* This is the sum of all the rewards in the last 100 iterations. This metric captures how well the agent is doing once it’s done learning (that is, is it getting close to an optimal reward?) Your plot should have the action-error probability as the x-axis and the cumulative reward as the y-axis. Each algorithm will be a different-colored curve. Use different line-weights to indicated the different settings of epsilon.

Discuss your results above. What happens to the performance as the action-error probability increases? How is SARSA affected as the epsilon is changed? How is Q-Learning? Hypothesize what may be causing the difference.