Jessie Drake(50358141)
Holliday Sims(50346405)

Assignment 2

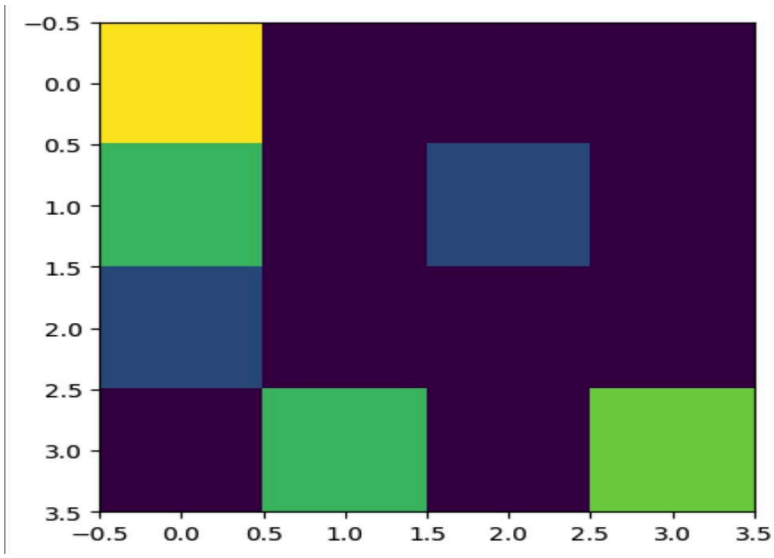
Part I

1. Describe the environment that you defined. Provide a set of states, actions, rewards, main objective, etc.

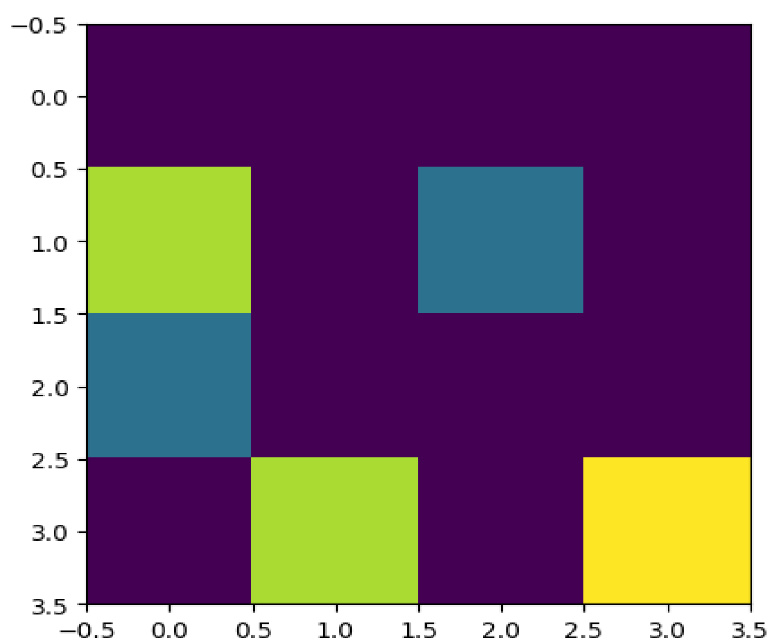
   States: {S1 = (0,0), S2 = (0,1), S3 = (0,2), S4 = (0,3),
   S5 = (1,0), S6 = (1,1), S7 = (1,2), S8 = (1,3),
   S9 = (2,0), S10 = (2,1), s11 = (2,2), s12 = (2,3),
   S13 = (3,0), s14 = (3,1), s15 =(3,2), s16 = (3,3)
   Actions: {Up, Down, Right, Left}
   Rewards: { -3,-4,+2,+5}
   Objective: Reach the goal state with maximum reward

2. Provide visualization of your environment.

   This is how the environment looks prior to running the algorithms



This is how the environment looks after running the algorithms



3. Safety in AI: Write a brief review (~ 5 sentences) explaining how you ensure the safety of your environment. E.g. how do you ensure that the agent chooses only actions that are allowed, that agent is navigating within defined state-space, etc

We are working with a 4x4 grid. To ensure the safety of our environment we first initialized the bounds it can not exceed which would be 4 on both axes. We ensure the agent chooses only actions that are allowed by observing the neighboring positions of the agent and determining the up down right and left position from the agent's current position. The observation space is limited to the bounds of our environment. We used np.clip() in our GridEnv() class to ensure the agent doesn't go outside the boundaries of the environment.

Part II & III

1. Briefly explain the tabular methods that were used to solve the problems. Provide their update functions and key features. What are the advantages/disadvantages?

SARSA

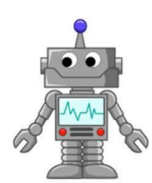
The SARSA algorithm is an on-policy algorithm. The main objective of the SARSA algorithm is to use the current estimate of the optimal policy to help dictate what actions will help them get to the goal state efficiently. As long as all the state action pairs are being visited repeatedly and the policy converges to the greedy policy, the SARSA algorithm will always converge to the optimal policy.

SARSA Example:

s_1, Right, 0, s_3, Up, +2, s_2, Up, -1, s_1, Left, +2, s_3, Down, -2, s_4, Left, 0, s_2, Right, -1, s_4, Up, +5, s_5

Current Q(s, a) Estimation

	Right	Left	Up	Down
s_1	1.5	2.1	-0.6	0.2
s_2	-0.1	1.9	0.5	1.7
s_3	2.4	1.9	1.1	-0.3
s_4	2.5	-1.4	2.1	0.4
s_5	0	0	0	0



Math : 1.5 + 0.4[2 + 0.8(0.4) - 1.5] **The solution to this implementation of SARSA:** 1.828

Jessie Drake(50358141)
Holliday Sims(50346405)

Psuedo Example of SARSA

start of program
~initialize any data structures, counters, starting nodes, etc.~
For x in total episodes
    Define the state
    Define the action
    For y in max steps
        If terminated:
            Break
        While not terminated:
            Take a step using the action
            Calculate the next action
            Step using the new action
            Update the q table with the old and new state, the old and new actons, and the reward
            Update to the new state
            Update to the new action
end of program

Advantages

SARSA's on policy method penalizes the agent when encountering negative rewards so that it exponentially learns how to not only avoid these spots but also reach the goal state. Also SARSA converges easier than the other algorithm

Disadvantages

The only disadvantage to the SARSA algorithm is that it doesn't prioritize collecting the maximum amount of positive rewards like the q learning methods.

Q LEARNING

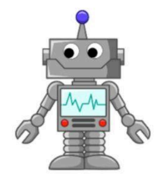
The Q learning algorithm is an off-policy algorithm. The main objective of the Q-Learning algorithm is to directly approximate the optimal action-value function, independent of the policy being followed. In our implementation this method can help the agent maximize the amount of rewards they collect while learning to avoid spots that do not have a positive rewarding system.

Q LEARNING Example:

s_1, Right, 0, s_3, Up, +2, s_2, Up, -1, s_1, Left, +2, s_3, Down, -2, s_4, Left, 0, s_2, Right, -1, s_4, Up, +5, s_5

Current Q(s, a) Estimation

	Right	Left	Up	Down
s_1	1.5	2.1	-0.6	0.2
s_2	-0.1	1.9	0.5	1.7
s_3	2.4	1.9	1.1	-0.3
s_4	2.5	-1.4	2.1	0.4
s_5	0	0	0	0



Math : 1.5 + 0.4[2 + 0.8(2.5) - 1.5] **The solution to this implementation of SARSA:** 2.5

<u>Psuedo Example of Q LEARNING</u>

The implementation of Q learning is very similar to SARSA, the changes will be bolded and colored blue or crossed out.

start of program
~initialize any data structures, counters, starting nodes, etc.~
Find the maximum Q-value for the next state
For x in total episodes
    Define the state
    Define the action
    For y in max steps
        If terminated:
          Break
        While not terminated:
          Take a step using the action
          Calculate the next action
          Step using the new action
          Update the q table with the old and new state, the old and new actons, and the reward
end of program

<u>Advantages</u>

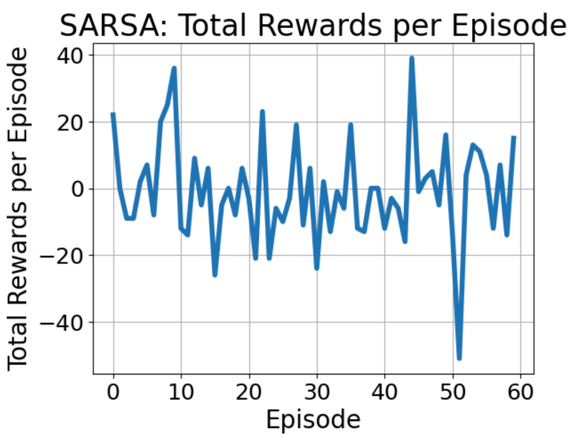
Jessie Drake(50358141)
Holliday Sims(50346405)

The Q learning method's off policy methods allows the agent to not only locate and reach the goal state but also maximizes the amount of rewards they accumulate along the way.

Disadvantages

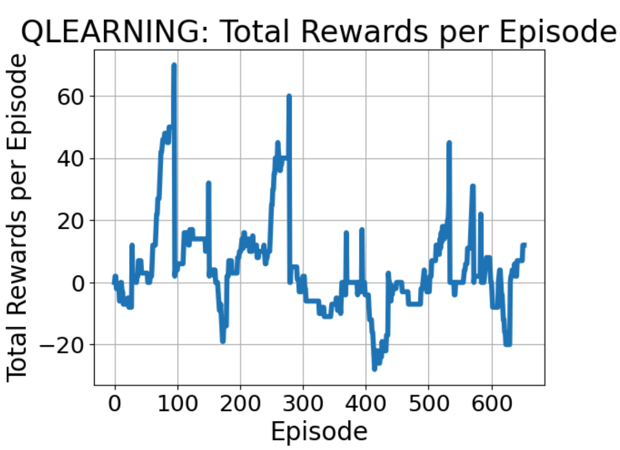
The only disadvantage to the Q Learning algorithm is that it doesn't have the same consistency like SARSA or ability to converge like SARSA.

2. Show and discuss the results after:
    - Applying SARSA to solve the environment defined in Part 1. Include Qtable. Plots should include epsilon decay and total reward per episode.




```
Action: 2 , Reward: 20 , Done: True
SARSA TERMINATED
For episode: 60, the Q table is:
 [[-2.03450691 -1.12515333 -1.94195632 -1.61670055]
 [-3.57438823 -1.90166032 -3.00624585 -1.82938948]
 [-2.50319261 -1.83211189 -2.40224136 -3.44632625]
 [ 2.08523115  0.32836227 -0.50707119 -1.21430556]
 [ 3.23977212  0.689984    0.92719976  0.35827661]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [-2.03619309 -1.01112253 -1.48096045 -1.4287098 ]]
```

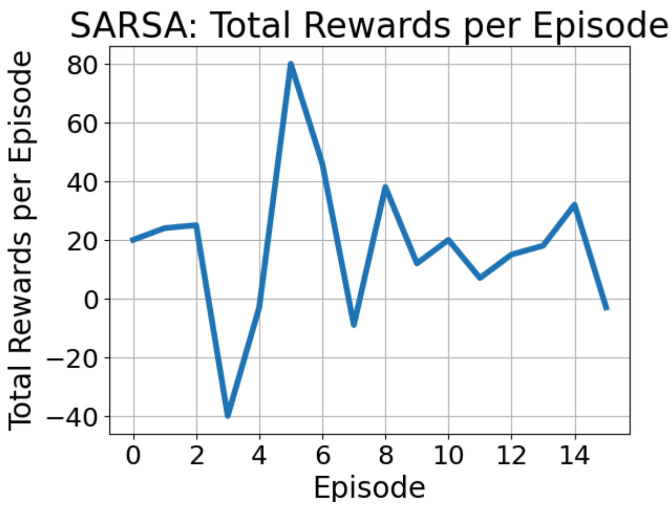
- Applying Q-learning to solve the environment defined in Part 1. Include Qtable. Plots should include epsilon decay and total reward per episode. Include the details of the setup that returns the best results.

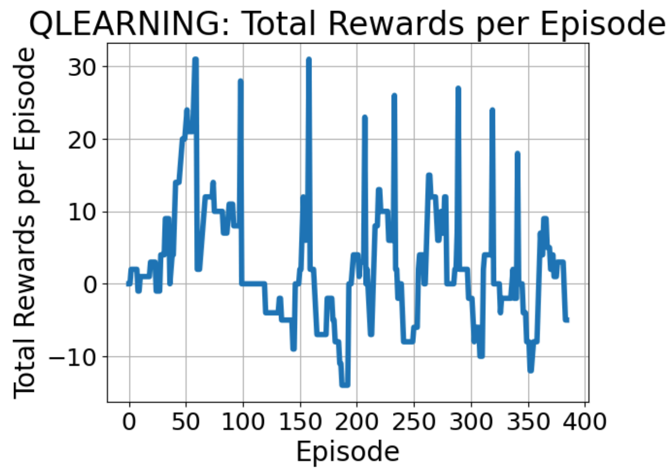



```
Action: 0 , Reward: 0 , Done: True
Q LEARNING TERMINATED
For episode: 59, the Q table is:
 [[22.57576856 22.82116201 22.35657421 21.7155041 ]
 [21.97147156 21.89559639 23.27087726 21.49270561]
 [21.43861258 21.83573311 23.9299481  21.36139576]
 [25.62026315 23.8266866  27.31102206 24.2596802 ]
 [27.16674585 20.62512177 27.08013569 19.07318771]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [22.47953123 21.54654343 21.8888864  20.97534732]]
```

Jessie Drake(50358141)
Holliday Sims(50346405)

- Provide the evaluation results for both SARSA and Q-learning. Run your environment for at least 10 episodes, where the agent chooses only greedy actions from the learned policy. Plot should include the total reward per episode.

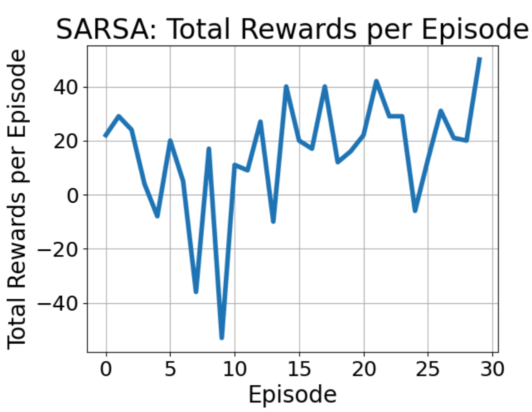


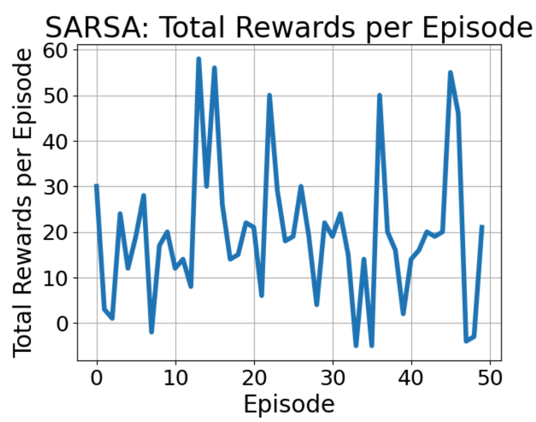


3. Provide the analysis after tuning at least two hyperparameters from the list above. Provide the reward graphs and your explanation for each of the results. In total, you should have at least 6 graphs for each implemented algorithm and your explanations. Make your suggestion on the most efficient hyperparameters values for your problem setup.

<u>SARSA</u>

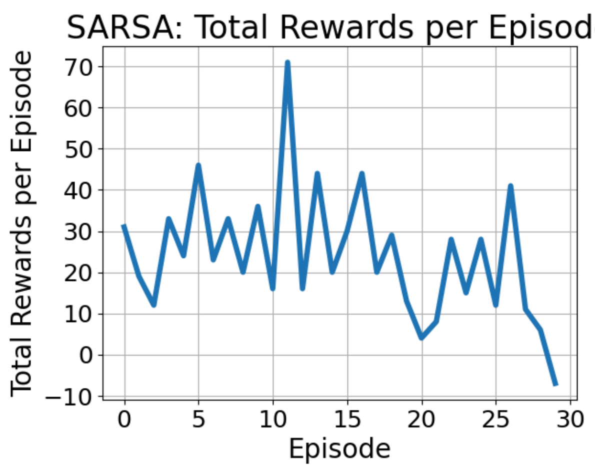
a. We tuned *total_episode*s from originally 30 episodes to 50.

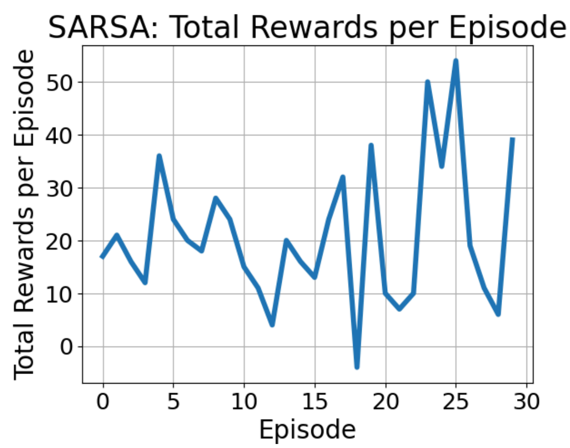




b. The graph on the left shows SARSA running for 30 episodes, and the graph on the right runs for 50 episodes. Running SARSA for 30 episodes results in a peak of collecting 40 rewards. As we can see in the graphs above, when we run SARSA for 50 episodes, it is much more efficient at collecting rewards when compared to running SARSA for 30 episodes. We can conclude that the longer that SARSA runs, the more positive rewards it collects.

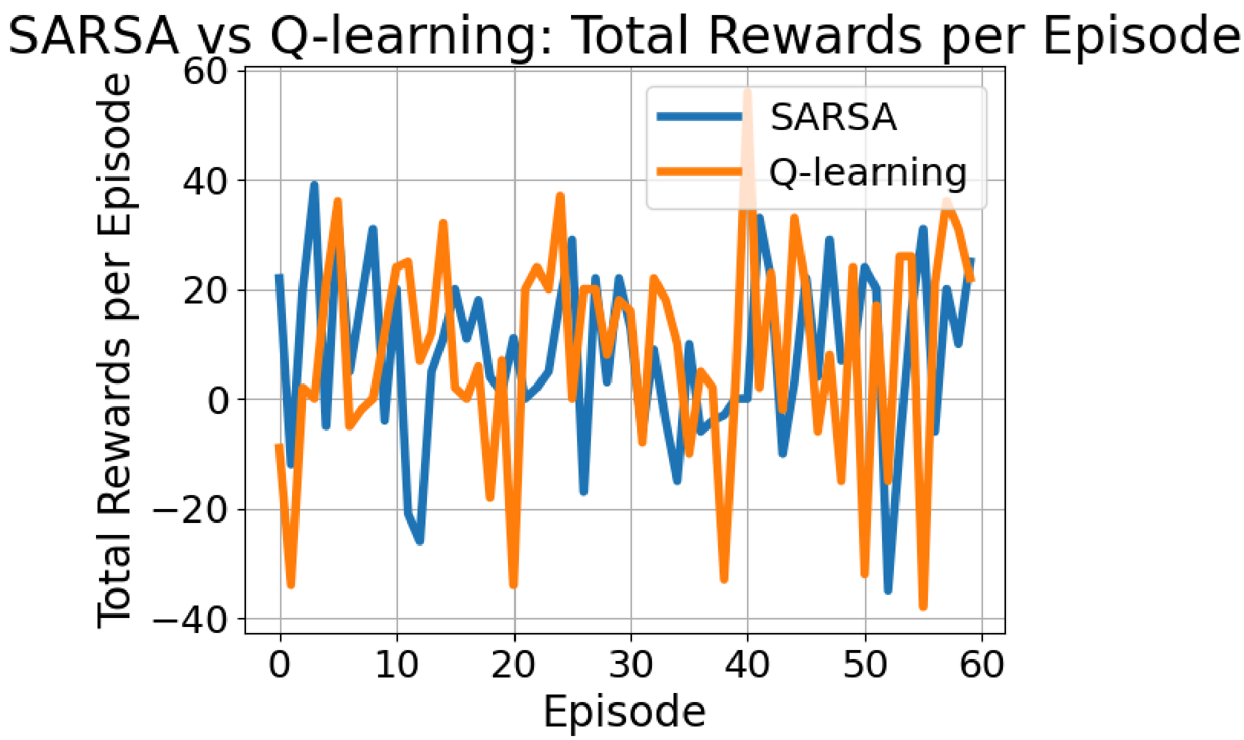
c. We tuned *alpha* from 0.10 to 0.15





d. We ran SARSA for 30 episodes, but the alpha values are different. The graph on the left has the *alpha* value = 0.1. When SARSA has an *alpha* value of 0.1, it collects the highest rewards per episode around episode 10, but never reaches around that peak for the rest of the SARSA implementation. We can also see that when it reaches the final episode, it has a negative reward around -10. The graph on the right has the alpha value = 0.15. When SARSA has an *alpha* value of 0.15, it collects just above 50 rewards at its peak around episode 25. When SARSA is done running, the agent leaves with a positive reward around 40.

4. Compare the performance of both algorithms on the same environment (e.g. show one graph with two reward dynamics) and give your interpretation of the results.



SARSA and Q Learning are run on the same environment for the same amount of episodes and hyperparameters. When Q learning reaches its peak, it collects around 60 episodes. However, when SARSA reaches its peak, it collects only 40 episodes. It is also important to note that both SARSA and Q Learning have negative rewards right before they reach the goal state/final episode. SARSA has a slower progression throughout running it, and doesn't maximize the amount of rewards like Q Learning does.

References:

https://www.geeksforgeeks.org/sarsa-reinforcement-learning/
https://builtin.com/machine-learning/sarsa
https://www.youtube.com/watch?v=FhSaHuC0u2M&t=1s
https://gymnasium.farama.org/api/env/#gymnasium.Env.render_mode
https://www.kaggle.com/code/nagakiranreddy/reinforcement-learning-sarsa-on-grid-world-env
https://vinitsarode.weebly.com/blogs/sarsa-vs-q-learning

Team Member	**Assignment Part**	**Contribution(%)**
Jessie Drake	1,2,3	50%
Holliday Sims	1,2,3	50%