**CSE 368, Fall 2022**

**A2: Defining and Solving Reinforcement Learning Task**

**Group # 12**

**Completed by: Yanbin Li, Baosheng Zheng**

**Part I**

1. Theme:Monopoly grid game with getting money as positive reward and paying money as negative reward. The reward will not disappear after visiting.

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)}

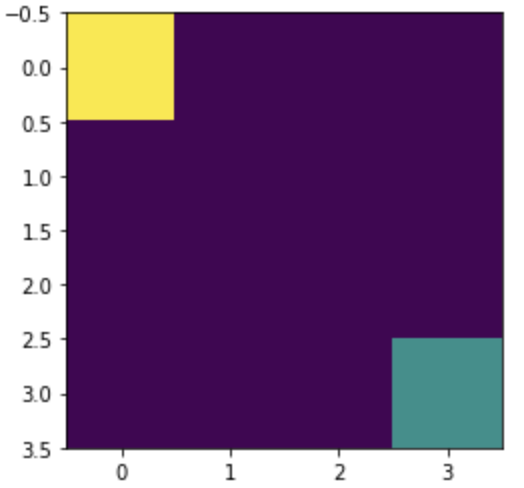
| (0,0)  0  START | (0,1)  0 | (0,2)  -7 | (0,3)  0 |
| --- | --- | --- | --- |
| (1,0)  0 | (1,1)  0 | (1,2)  0 | (1,3)  0 |
| (2,0)  0 | (2,1)  -8 | (2,2)  0 | (2,3)  +9 |
| (3,0)  +10 | (3,1)  0 | (3,2)  0 | (3,3)  +20 GOAL |

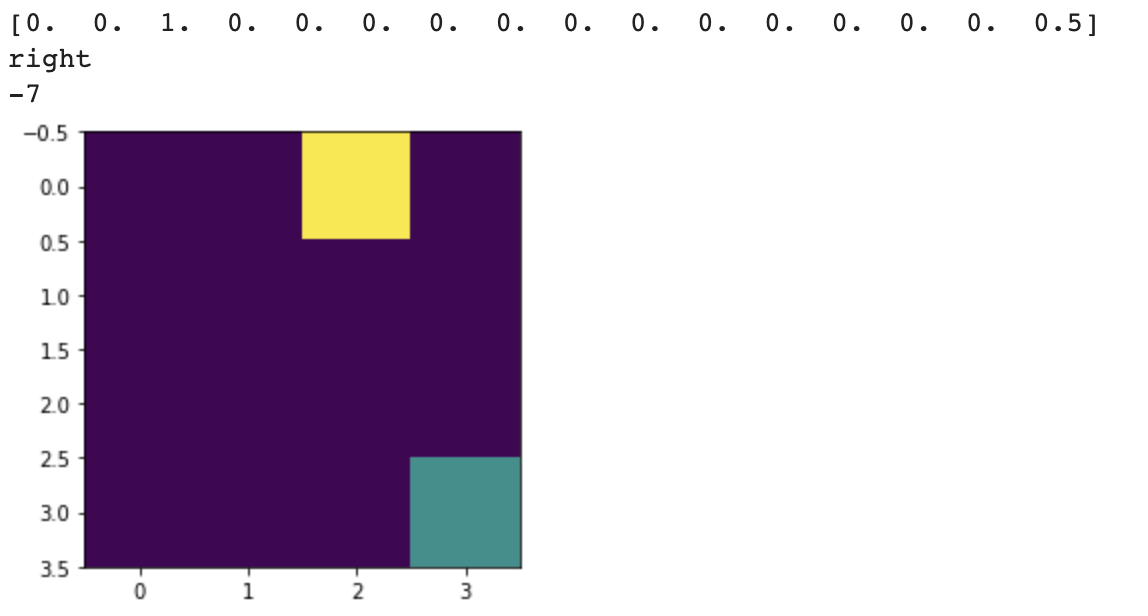
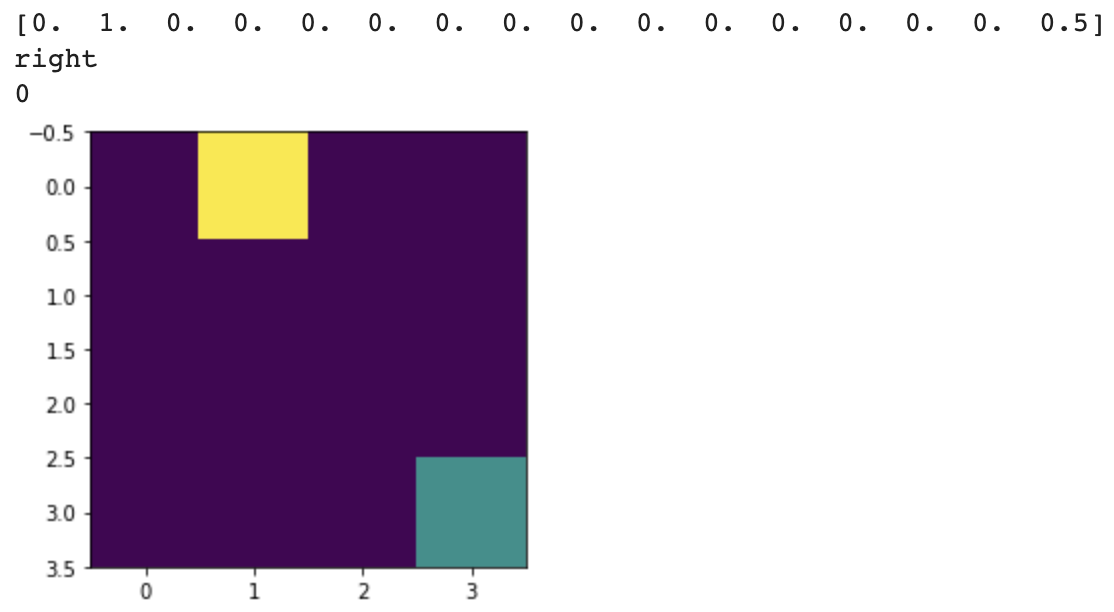
Actions:{Up(1), Down(0), Right(2), Left(3)}

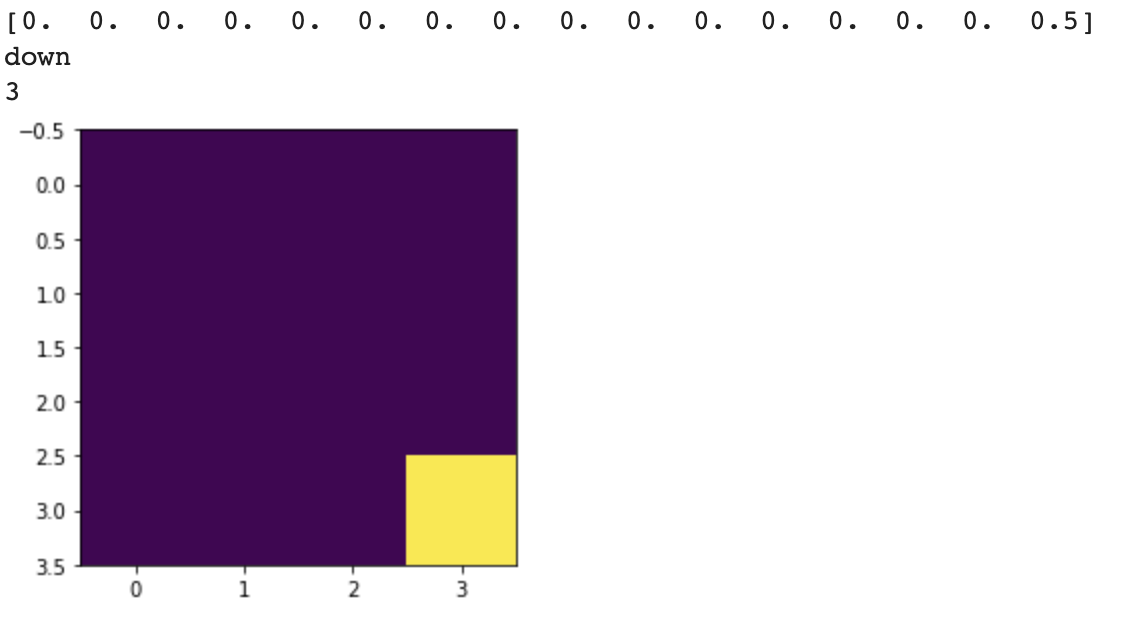
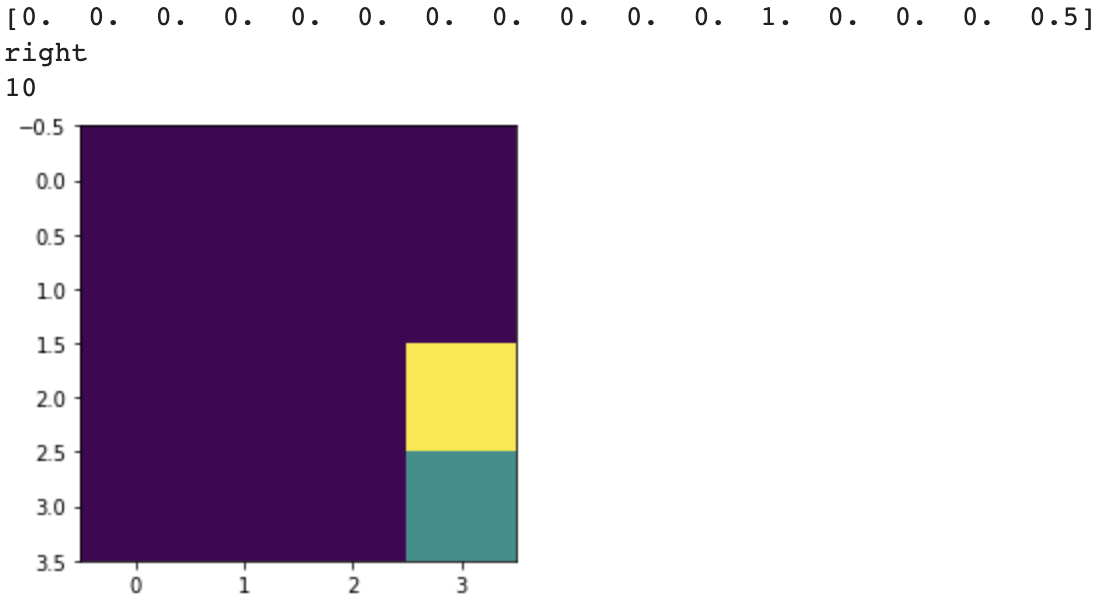
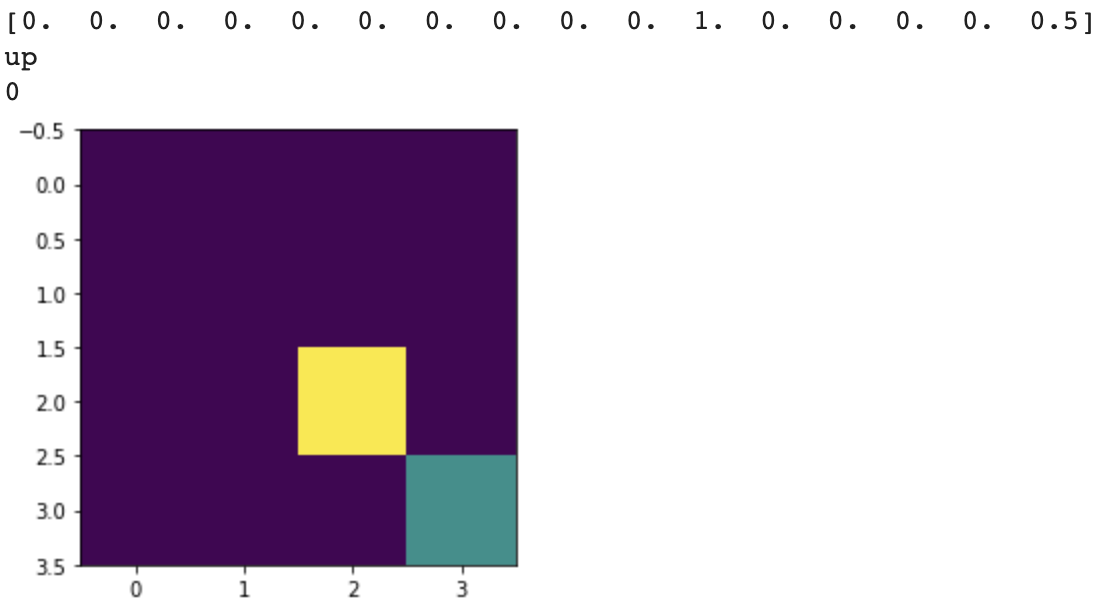
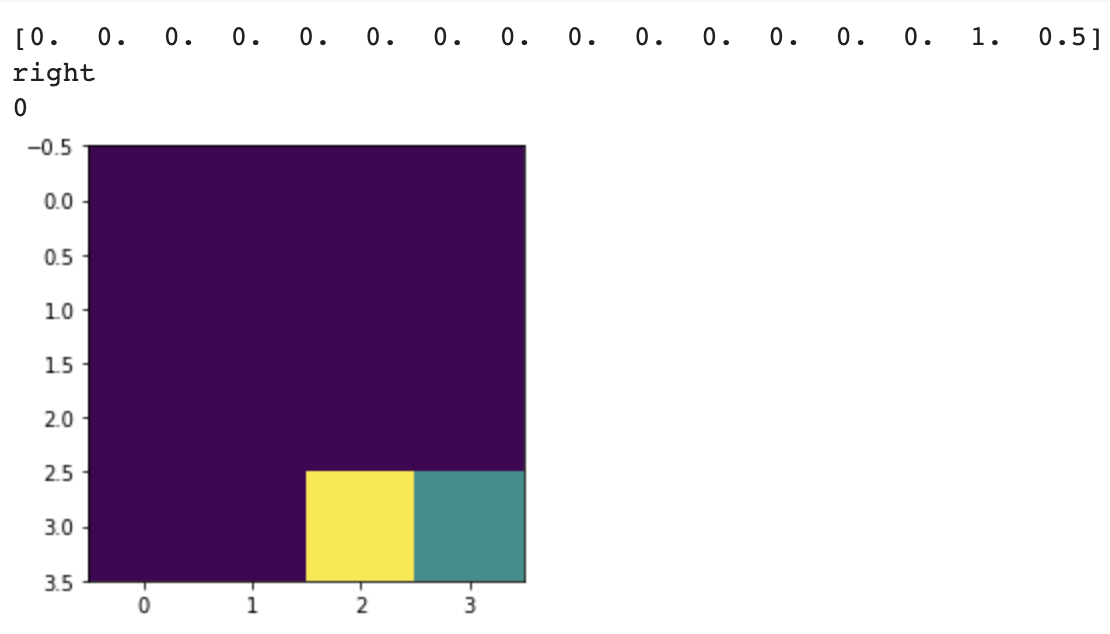
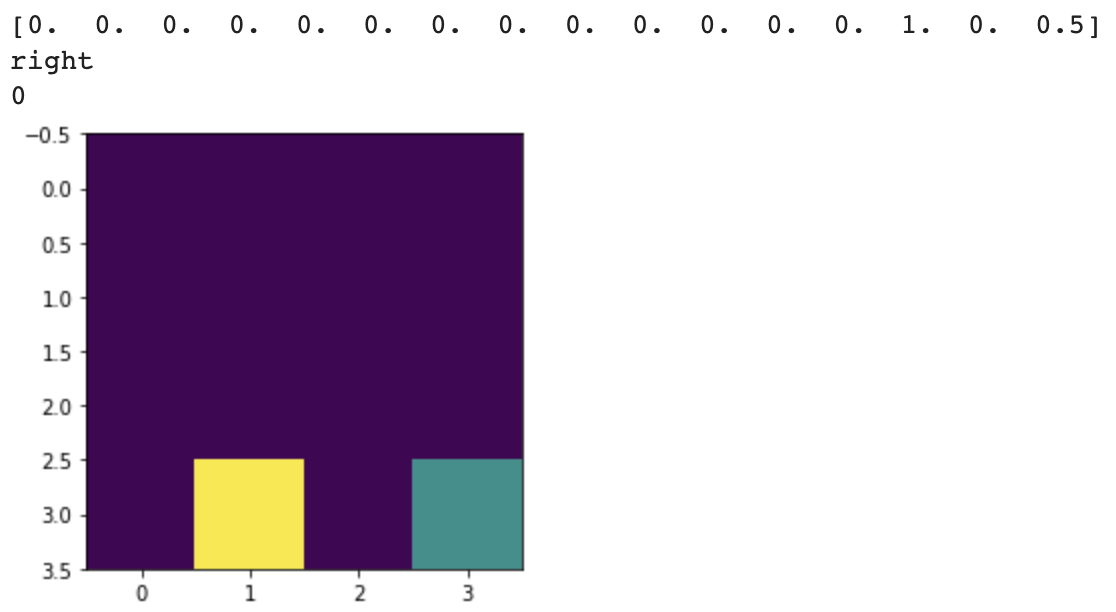
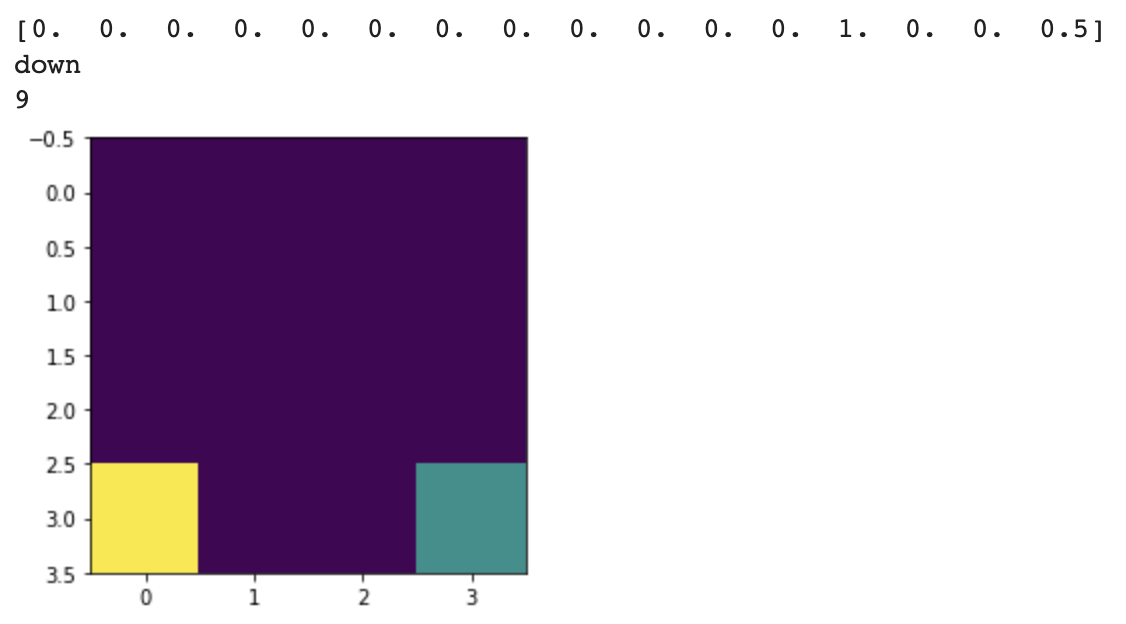
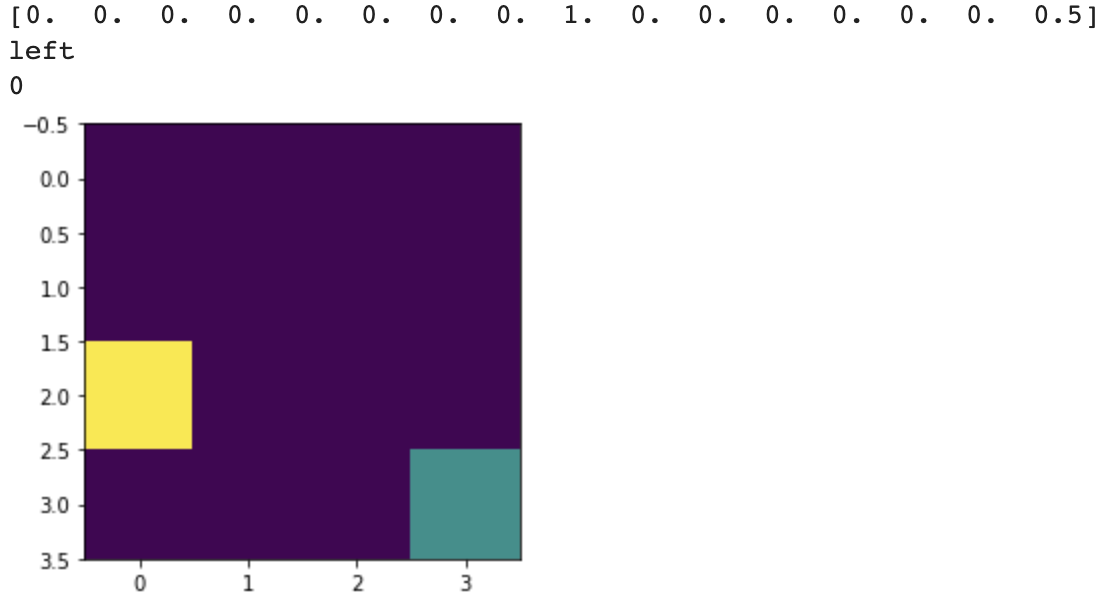
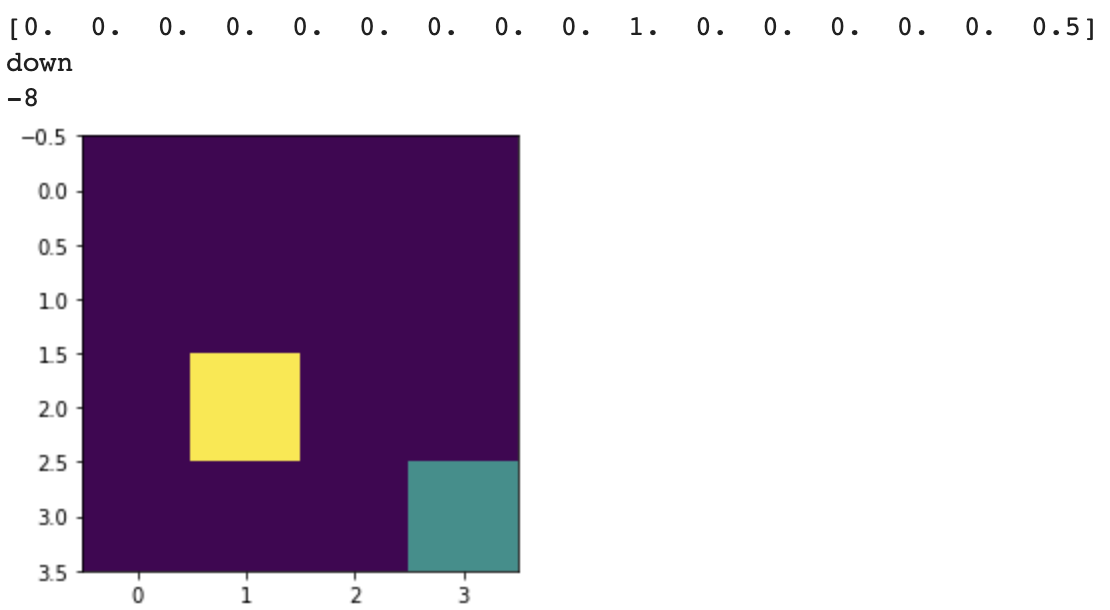
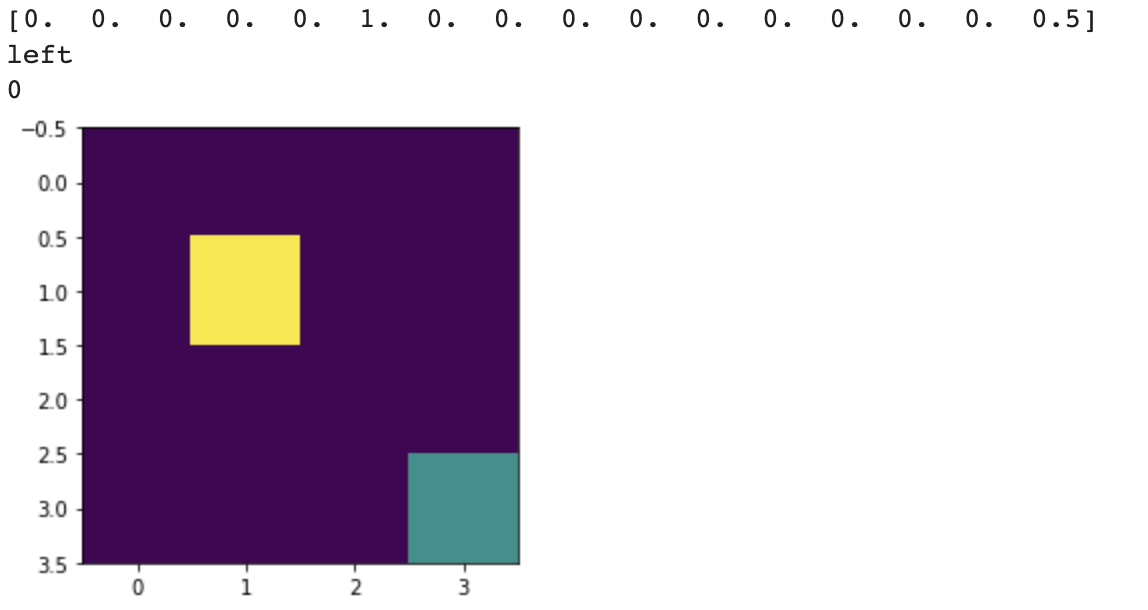
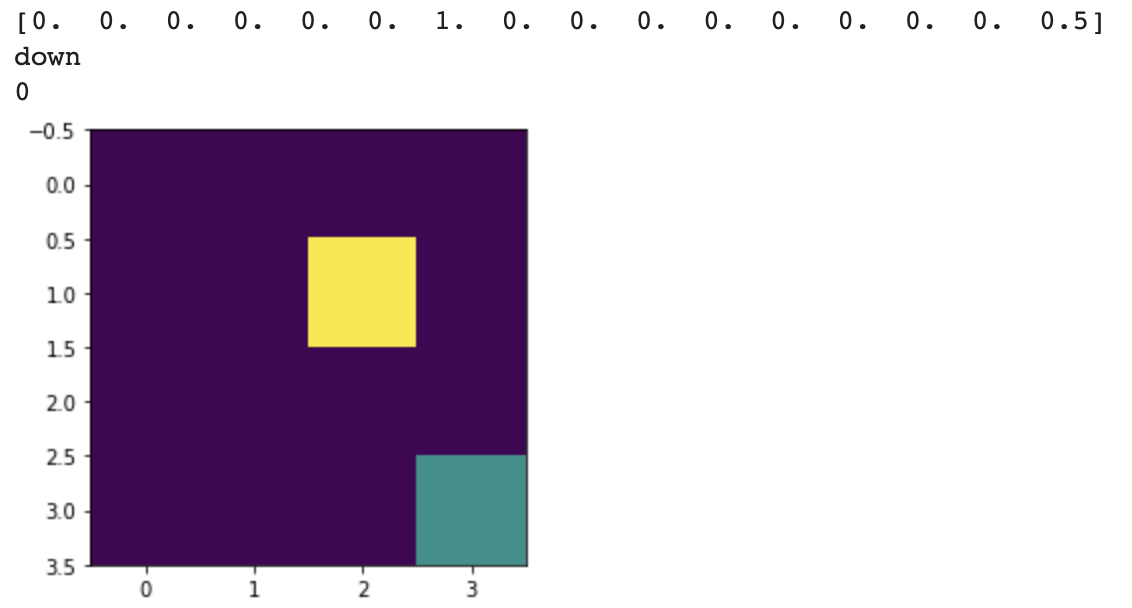
Rewards: {-8, +10, +9, -7}

Objective: Maximum total reward under the maximum timesteps.

start:







1. Safety in AI description

Within our Step() function, we set up the limitation of moving using the “self.agent\_pos = np.clip(self.agent\_pos, 0, 3) # set max move space”.

This makes sure that the AI will never move outside the grid. In this case, it ensures the safety of the environment.

**Part II**

1. Explain Sarsa

It uses the current states and actions to estimate the optimal policy in order to reach the maximum reward.

1. Explain Q learning

Similar to the SARSA, Q learning will choose the action with the maximum value of the next generation.

1. Results:

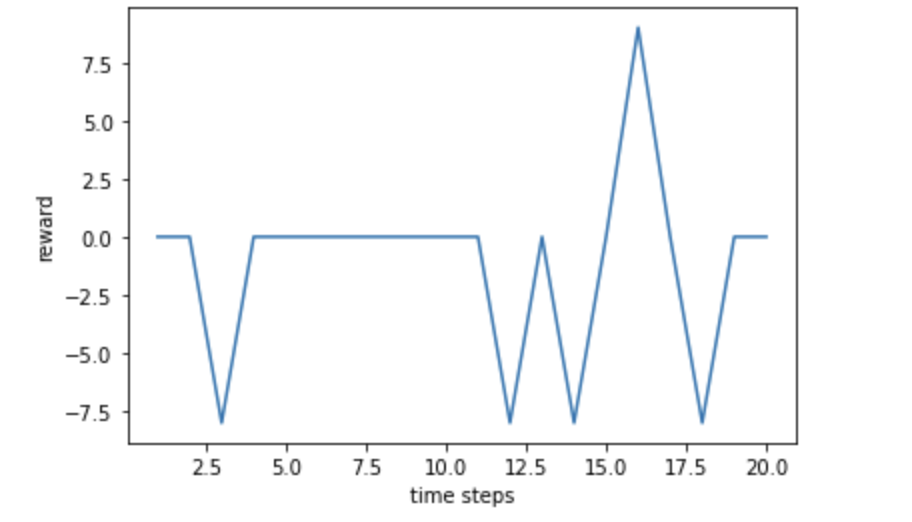
SARSA:

[insert epsilon greedy graph]

x: time step

y: average reward

epsilon: 0.1, 0.2, 0.5

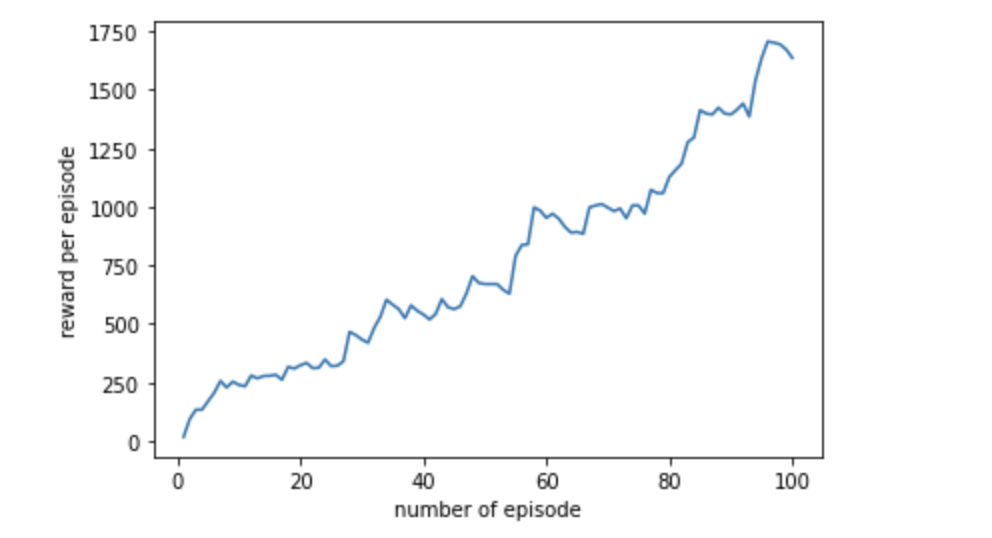


[insert total rewards per episode graph]

x: number of episode

y: average reward per episode

x: 20 50 100



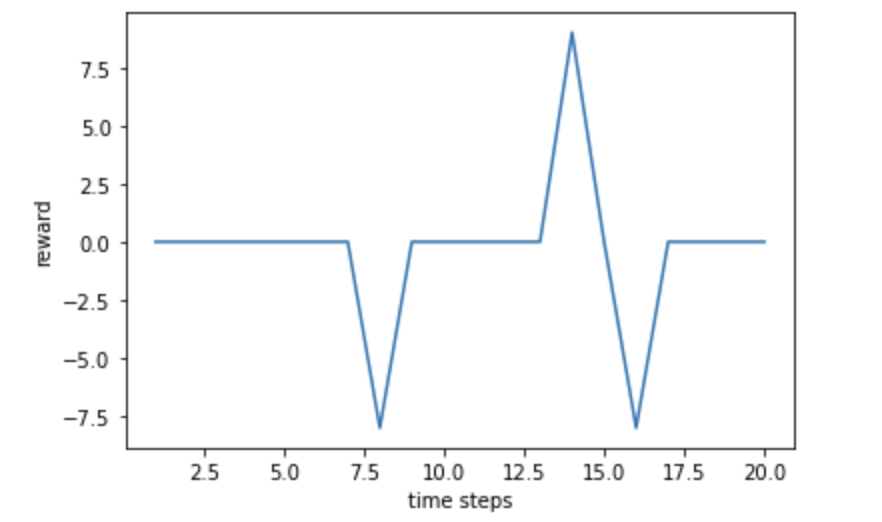
Q Learning:

[insert epsilon greedy graph]

x: time step

y: average reward

epsilon: 0.1, 0.2, 0.5

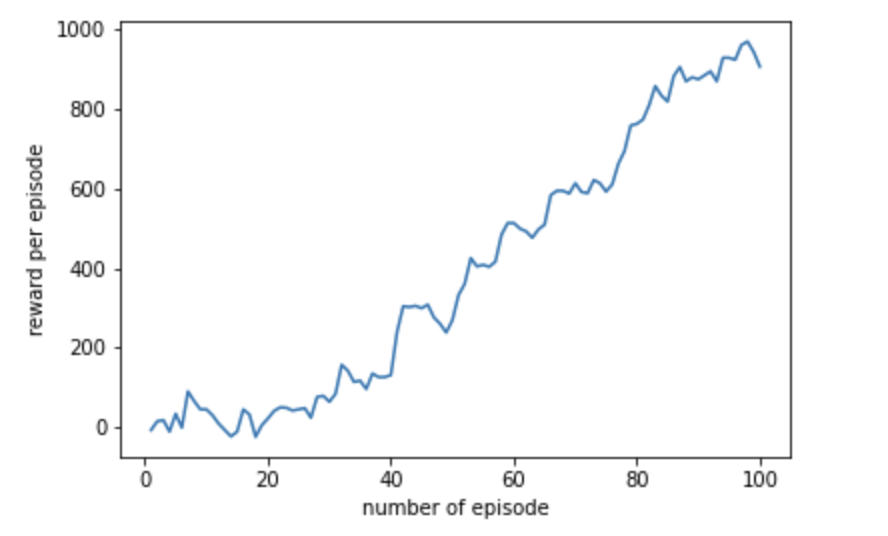


[insert total rewards per episode graph]

x: number of episode

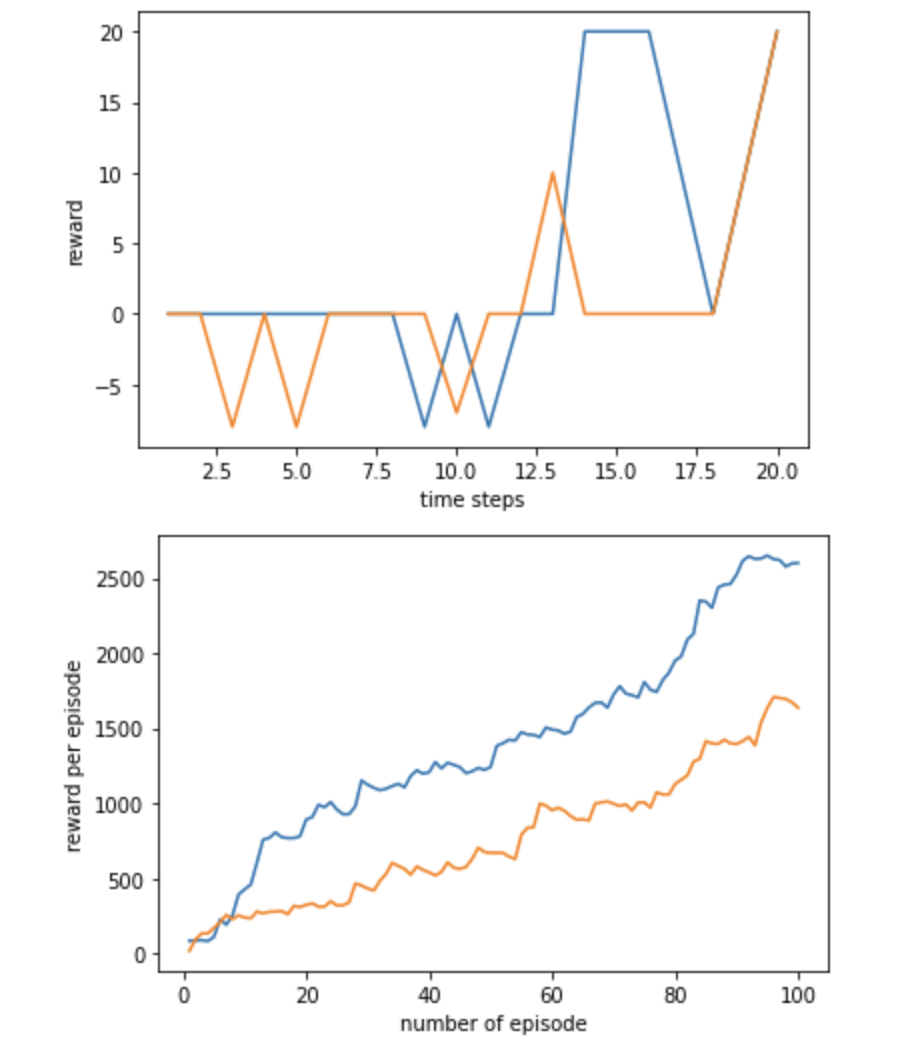
y: average reward per episode

x: 20 50 100



1. Plot the total rewards per episode graph of both Q-Learning and Sarsa on a single plot and give your interpretation of the results.

Combine into one plot:



Q-learning is always going for the best reward, so the total reward will increase while the number of episodes go up.

On the other hand, SARSA will choose its next actions strictly based on the epsilon greedy policy, which means it will pick the most secure paths but may not be the best reward path.

In conclusion, in order to get the best rewards, using Q-learning would be a better idea.

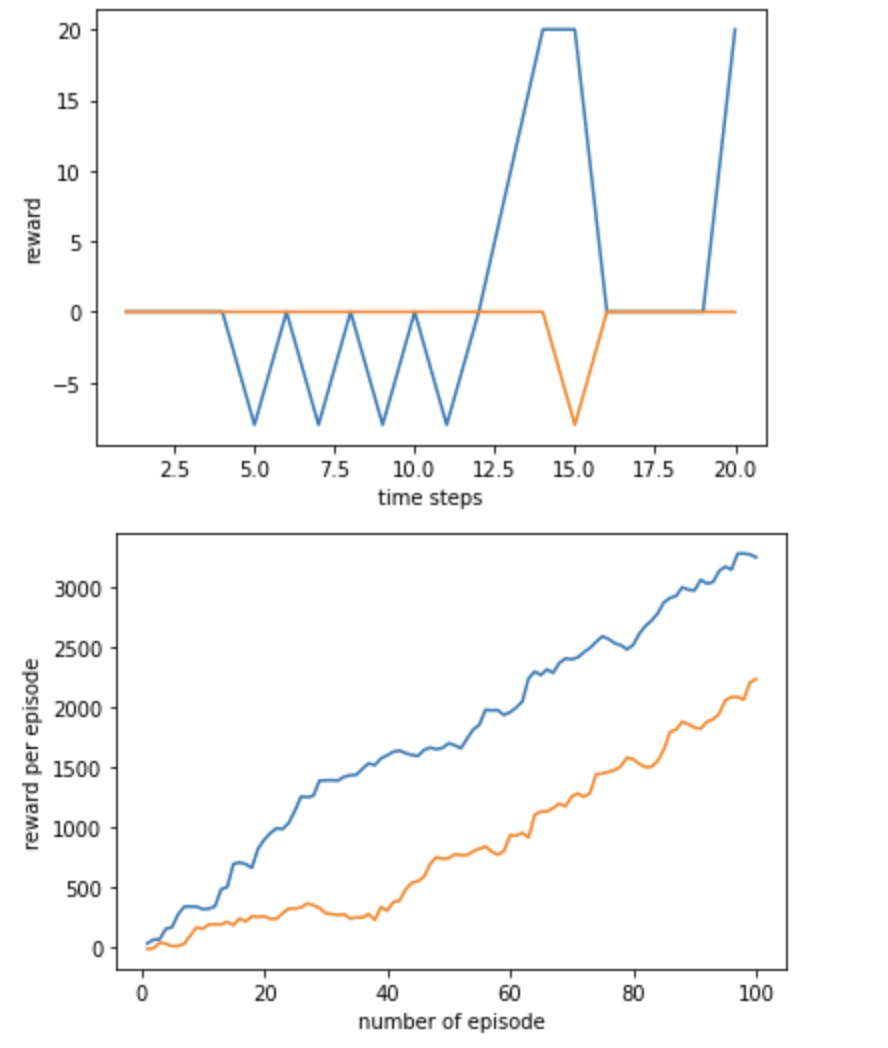
1, Change first hyperparameter:

SARSA’s step size is set to 0.1 from 1 .

Q-learning’s step size is set to 0.1 from 1 .

SARSA(0.1, 1)

Q\_learning(0.1, 1)



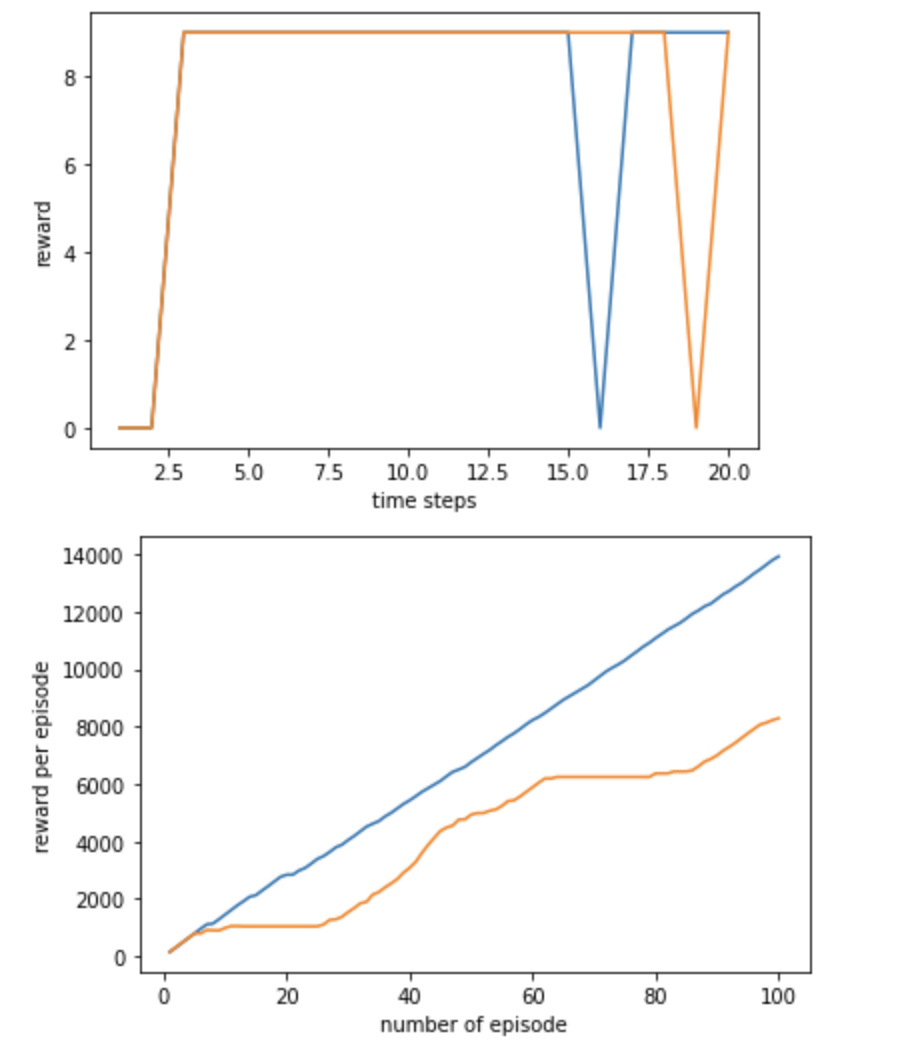
2, Change second hyperparameter:

SARSA’s epsilon\_greedy is also set to 0.1 from 1.

Q-learning’s epsilon\_greedy is also set to 0.1 from 1.

SARSA(1, 0.1)

Q\_learning(1, 0.1)



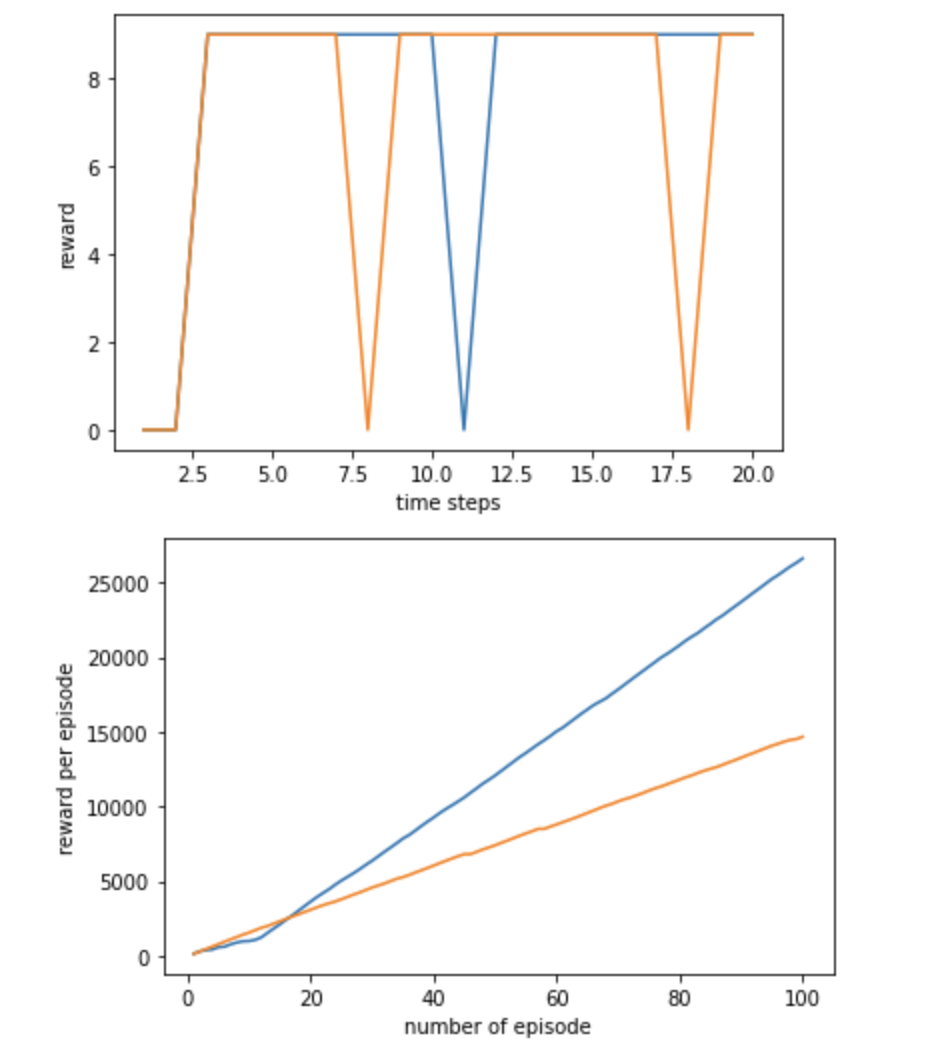
3, Change both hyperparameters:

SARSA’s step size is set to 0.1 from 1 and its epsilon\_greedy is also set to 0.1 from 1.

Q-learning’s step size is set to 0.1 from 1 and its epsilon\_greedy is also set to 0.1 from 1.

SARSA(0.1, 0.1)

Q\_learning(0.1, 0.1)



Based on the given graphs, we notice that when we the ‘epsilon-greedy’ will have more impact among two hyperparameters. As we can see within the graphs the Cumulative reward per episode will increase linearly over time. In conclusion, the smaller ‘epsilon-greedy’ is the better our result would be.

**Bonus**

1. Double Q Learning:

[insert epsilon greedy graph]

[insert total rewards per episode graph]

1. Explain your interpretation of the results in comparison to Q Learning and SARSA

If you are working in a team of two people, we expect equal contribution for the assignment. Provide a contribution summary by each team member in a form of a table:

| Team Member | Assignment Part | Contribution (%) |
| --- | --- | --- |
| Yanbin Li | 1 and 2 | 50% |
| Baosheng Zheng | 1 and 2 | 50% |
|  |  |  |

**References**

[fall22\_cse368\_lec\_12\_Value\_Functions.pdf](https://piazza.com/class_profile/get_resource/l79ev0g88vc7a7/l9epd216xcs244)

[fall22\_cse368\_lec\_13\_Dynamic\_Programming.pdf](https://piazza.com/class_profile/get_resource/l79ev0g88vc7a7/l9hxathaglc2bo)

[fall22\_cse368\_lec\_14\_DP\_TD.pdf](https://piazza.com/class_profile/get_resource/l79ev0g88vc7a7/l9oq2bg52km6wo)

[fall22\_cse368\_lec\_15\_TD\_Double\_Q\_learning.pdf](https://piazza.com/class_profile/get_resource/l79ev0g88vc7a7/l9rk6yp1fic2hd)