1. Implementation

(1) Briefly describe your implementation

- Q-learning:

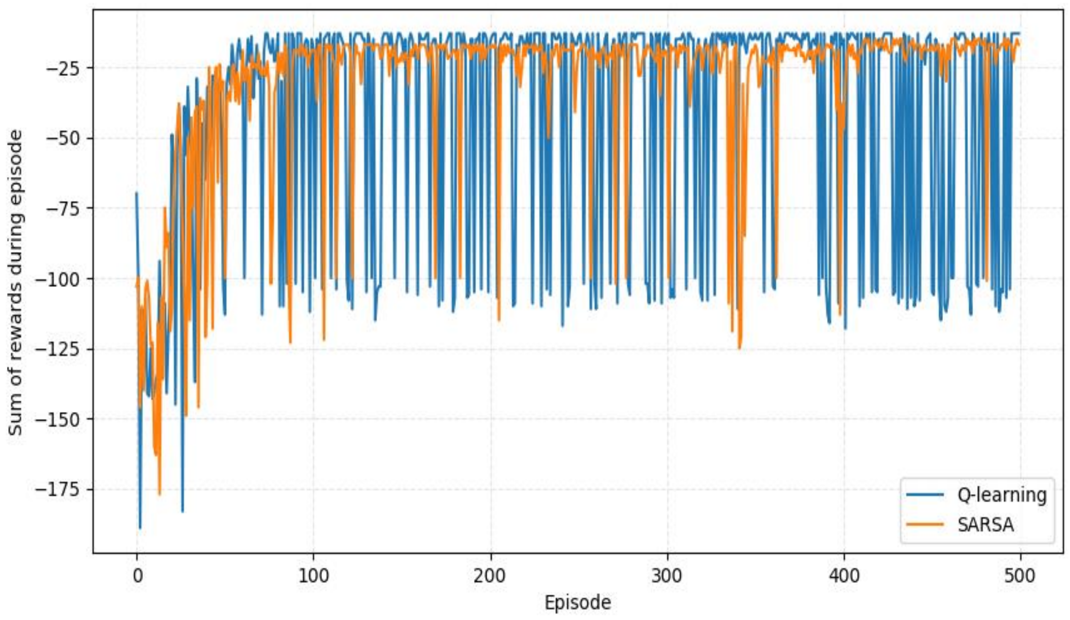
```
# start training
for i_episode in range(num_episodes):
    # Reset the environment and pick the first action
    state = env.reset()

    for t in itertools.count():
        #Raise NotImplementedError('Q-learning NOT IMPLEMENTED')
        #Choose action from state using policy derived from Q (epsilon greedy)
        action_probs = policy(state)
        action = np.random.choice(np.arange(len(action_probs)), p=action_probs)
        #Derived next state and reward from action we chose
        next_state=env.P[state][action][0][1]
        reward=env.P[state][action][0][2]
        #Calculate the sum of episode rewards and episode lengths
        episode_rewards[i_episode] += reward
        episode_lengths[i_episode] += 1
        #Derived "is done" from action we chose
        is_done = env.P[state][action][0][3]
        #Accoring to "a" to choose the max Q(next state,a)
        a=np.max(Q[next_state])
        #Update Q(state,action)
        Q[state][action] += alpha*(reward+ discount_factor*a-Q[state][action])
        #Update state
        state=next_state
        #If the episode reach goal we stop loop and break
        if(is_done == True):
            break
#Return Q, episode_rewards, episode_lengths
return Q, episode_rewards, episode_lengths
```

- Sarsa:

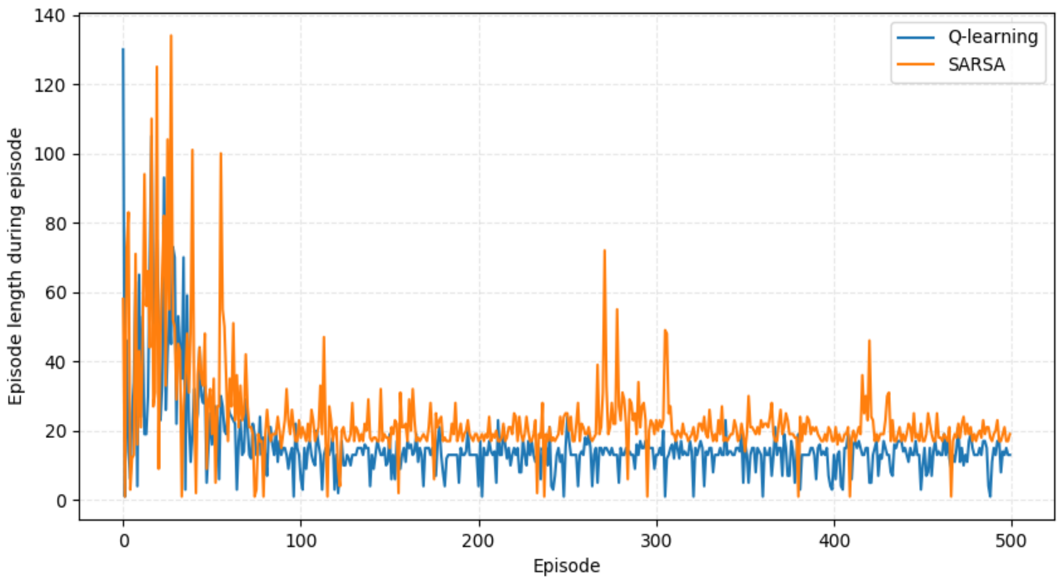
```
for i_episode in range(num_episodes):
    # Reset the environment and pick the first action
    state = env.reset()
    #Choose action from state using policy derived from Q (epsilon greedy)
    action_probs = policy(state)
    action = np.random.choice(np.arange(len(action_probs)), p=action_probs)

    for t in itertools.count():
        #raise NotImplementedError('SARSA NOT IMPLEMENTED')
        #Derived next state and reward from action we chose
        next_state=env.P[state][action][0][1]
        reward=env.P[state][action][0][2]
        #Calculate the sum of episode rewards and episode lengths
        episode_rewards[i_episode] += reward
        episode_lengths[i_episode] += 1
        #Derived "is done" from action we chose
        is_done = env.P[state][action][0][3]
        #Choose next action from next state using policy derived from Q (epsilon greedy)
        next_action_probs = policy(next_state)
        next_action = np.random.choice(np.arange(len(next_action_probs)), p=next_action_probs)
        #Update Q(state,action)
        b=Q[next_state][next_action]
        Q[state][action] += alpha*(reward+ discount_factor*b-Q[state][action])
        #Update state
        state=next_state
        #Update action
        action=next_action
        #If the episode reach goal we stop loop and break
        if(is_done == True):
            break
#Return Q, episode_rewards, episode_lengths
return Q, episode_rewards, episode_lengths
```

2. Experiments and Analysis

(1) Plot curves of different methods into a figure. (As example above)



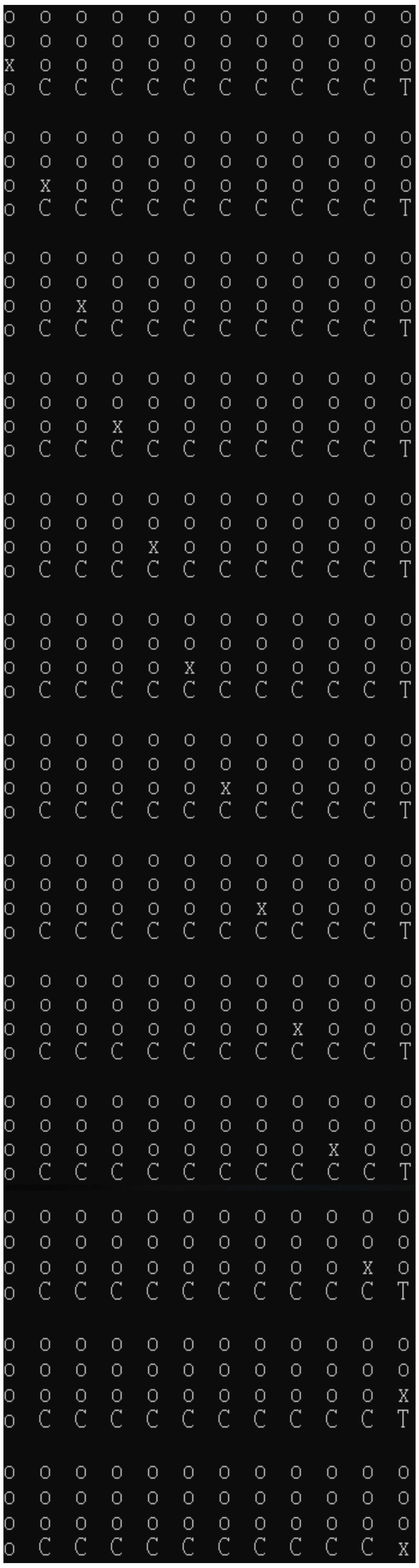
(2) Plot the episode length (time steps taken per episode) v.s. episode. What do you observe?



From the diagram above, I observed that in the beginning , the length of Q learning's and Sarsa's episodes are more, it's because both of them are exploring. And I also found out that Q learning's convergence speed is better than Sarsa's, this is because in the last Q learning will find the optimal policy. But, I also found out that there are many Q learning's episodes that are less than 20, so it means that it will be easy to falling of the cliff.

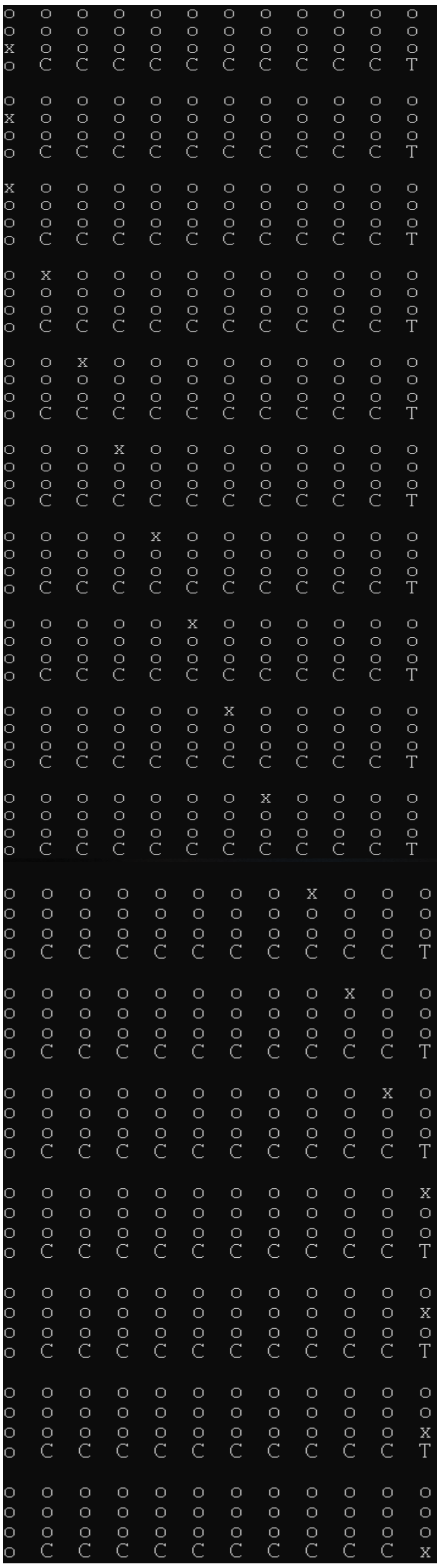
(3) Render and show the trajectory of each method. What do you observe?

Q-learning



```
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
x  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  x  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  x  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  x  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  x  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  x  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  x  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  x  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  x  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  x  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  x  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  x
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  x
```

Sarsa



```
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
x  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
x  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

x  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  x  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  x  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  x  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  x  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  x  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  x  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  x  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  x  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  x  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  x  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  x
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  x
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  x
o  C  C  C  C  C  C  C  C  C  C  T

o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  C  C  C  C  C  C  C  C  C  C  x
```

I observed that, Q learning will costs less steps than Sarsa to reach the goal, but it also means that it will be easy for Q learning to falling of the cliff. Conversely, Sarsa will not be easy to falling of the cliff. I run the plot for many times , I found that Sarsa choose the most peripheral path for many times, but Sarsa is not always do so, in fact, the path chose by Sarsa according to epsilon (because actions is picked by epsilon-greedy).

(4) Observe the reward curve of each algorithm. We can observe that the reward curve of SARSA is more stable than Q-learning (less severe drop to -100). Please explain.

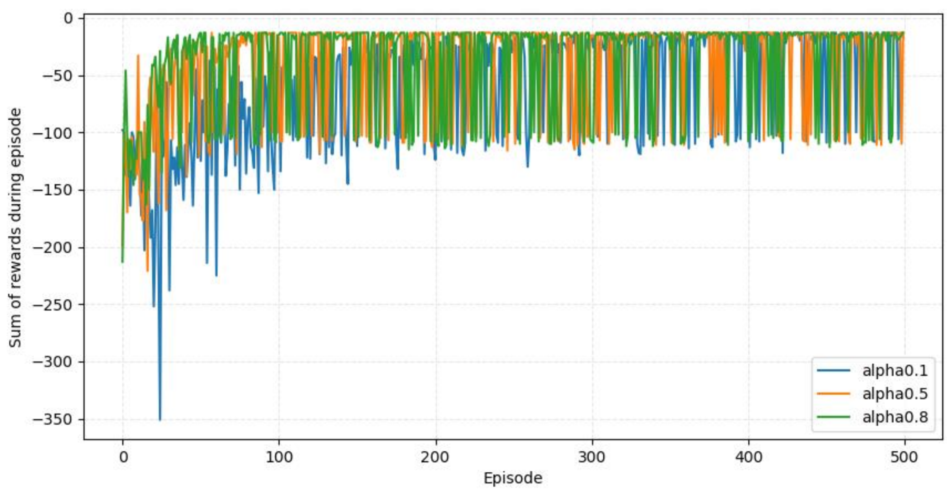
It’s because of that, Q-learning use greedy action to update Q table, but use epsilon-greedy method to choose the action, so it still have probabilities to falling of the cliff, so the reward curve of SARSA is more stable than Q-learning. Conversely, Sarsa use epsilon-greedy action to update Q table,and use epsilon-greedy action to choose the action, so the path is safer.

(5) Why is Q-learning considered an off-policy control method? How about SARSA?

1. It’s because of that, Q-learning use greedy action (a1) to update Q table, but use epsilon-greedy method to choose the action (a2), a1 is different with a2, and a1 is used to evaluate or improve policy, a2 is used to generate data, so Q-learning considered an off-policy control method.
2. Sarsa use epsilon-greedy action (a3) to update Q table,and use epsilon-greedy method to choose the action (a4), a3 is same with a4, and a3 is used to evaluate or improve policy, a4 is used to make decision, so Sarsa considered an on-policy control method.

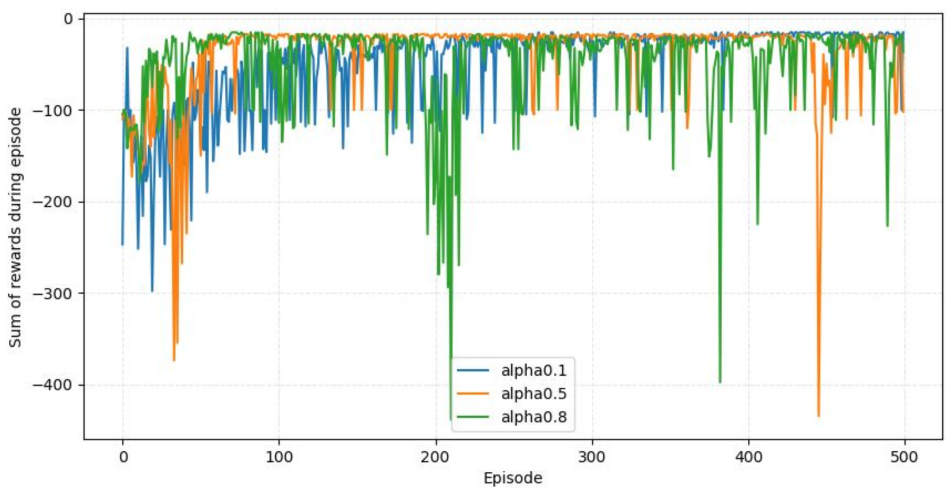
(6) Vary the TD learning rate alpha, what happens?

- Q-learning



Obviously, when alpha=0.1, sum of rewards are lower, it's because the speed using to update Q is slow, and I can still observe that when alpha=0.5 and alpha=0.8, the performance of sum of rewards are similar.

- Sarsa



I observed that, when alpha=0.8, sum of rewards is unstable, it's because of that sarsa use epsilon-greedy to choose next action, so the chosen action may be worse. Conversely, when alpha=0.1, sum of rewards is stable.