Q-Learning

In this homework, you should complete the Q-learning algorithm for the environment gym taxi

In [1]:
```
import gym
import numpy as np
import random

# the None is the position you should modeify to complete the algorithm
```

Step 1 Creat the environment

Using the API imported from gym

In [2]:
```
env = gym.make('Taxi-v3')
env.render()
```

```
+---------+
|R: | : :G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
```

Step 2 Create the Q-table and initialize it

You can use the gym api to fetch the dimension of action space and state space

In [3]:
```
action_space = env.action_space.n
state_space = env.observation_space.n

#Please complete this initialization in this line
Q_table = np.zeros((state_space, action_space))
```

Step 3 Configure the hyperparameters

In [4]:
```
total_episodes = 50000
total_test_episodes = 100
learning_rate= 0.7

# discount rate
gamma= 0.8

# Create the hyperparameters
sample_rewards = []
```

Step 4 Q Learning algorithm

Note: The formula of Q table update(Bellman equation)
Bellman equation

In [6]:
```
for episode in range(total_episodes):
    state= env.reset()
    step=0
    done=False
    sample_reward = 0
    while True:
        # Please complete this action selection in this line via the maximum value
        action = np.argmax(Q_table[state, :])

        # fetech the new state and reward by gym API
        new_state, reward, done, info = env.step(action)
        # Calculate the reward of this episode
        sample_reward += reward

        # Update the Q table
        Q_table[state, action] += learning_rate*(reward+gamma*np.max(Q_table[new_state,:])

        # Update the state
        state = new_state

        #store the episode reward
        if done == True:
            sample_rewards.append(sample_reward)
            break

    # print the average reward over 1000 episodes
    if episode%1000 == 0:
        mean_reward = np.mean(sample_rewards)
        sample_rewards = []
        print("average reward:" +str(episode)+ ": "+str(mean_reward))
```

```
average reward:0: -200.05446293494705
average reward:1000: -9.331
average reward:2000: 7.817
average reward:3000: 7.906
average reward:4000: 7.909
average reward:5000: 7.867
average reward:6000: 7.864
average reward:7000: 7.776
average reward:8000: 7.722
average reward:9000: 7.939
average reward:10000: 7.991
average reward:11000: 8.069
average reward:12000: 7.799
average reward:13000: 7.856
average reward:14000: 7.864
average reward:15000: 7.838
average reward:16000: 7.886
average reward:17000: 7.979
average reward:18000: 7.861
average reward:19000: 7.733
average reward:20000: 7.972
average reward:21000: 7.848
average reward:22000: 7.906
average reward:23000: 8.124
average reward:24000: 7.791
```

```
average reward:25000: 7.795
average reward:26000: 7.912
average reward:27000: 7.909
average reward:28000: 7.917
average reward:29000: 7.95
average reward:30000: 7.948
average reward:31000: 7.878
average reward:32000: 7.861
average reward:33000: 7.887
average reward:34000: 7.977
average reward:35000: 7.823
average reward:36000: 7.978
average reward:37000: 7.891
average reward:38000: 7.951
average reward:39000: 7.868
average reward:40000: 7.926
average reward:41000: 7.879
average reward:42000: 7.916
average reward:43000: 7.884
average reward:44000: 7.927
average reward:45000: 7.899
average reward:46000: 7.92
average reward:47000: 7.924
average reward:48000: 7.775
average reward:49000: 7.827
```

Step 5 Test your Q table

In [9]:

```
env.reset()
rewards=[]

max_steps = 1000

for episode in range(total_test_episodes):
    state=env.reset()
    step = 0
    done =False
    total_rewards = 0

    for step in range(max_steps):
        #  action selection
        action = np.argmax(Q_table[state,:])
        # fetech the new state and reward by gym API
        new_state, reward, done, info = env.step(action)

        total_rewards += reward
        if done:
            rewards.append(total_rewards)
            break
        state = new_state

env.close()
print("test:")
print("average reward over 100 episode:"+ str(np.mean(rewards)))
```

```
test:
average reward over 100 episode:7.65
```