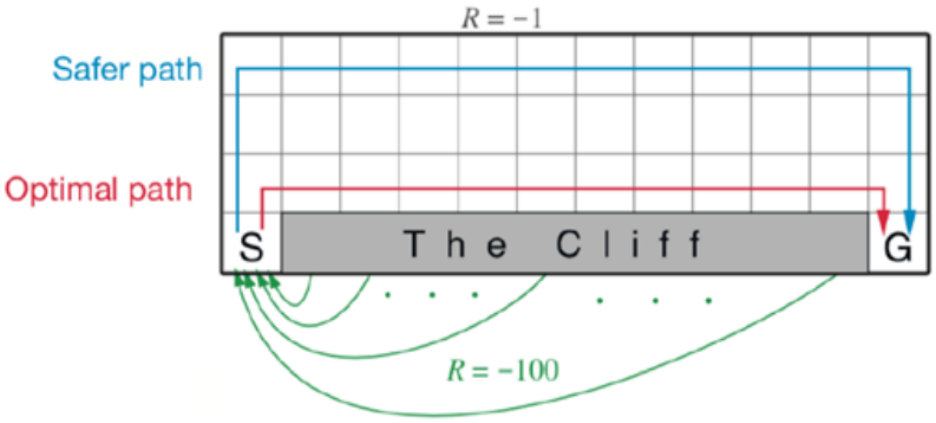
实验报告2

学号	姓名
20337025	崔璨明

1、实验内容

基于Cliff Walk例子实现SARSA、Q-learning算法，要求如下：

- This is a standard undiscounted, episodic task, with start and goal states, and the usual actions causing movement up, down,right, and left.
- Reward is -1 on all transitions except those into the region marked as Cliff. Stepping into this region incurs a reward -100 and sends the agent instantly back to the start.



2、核心思路

在该问题中，对每个状态进行编号，并建立相应的Q-table，在每次的迭代中，初始化状态为角色位于起点，每个状态有四个动作（上下左右），每一步转移的reward都设为-1，若采取某一动作后若到达了边界则待在原地，若到达了悬崖则返回起点，reward设为-100，并设置一个符号变量flag来判断某次迭代是否终止，若到达了终点则该次迭代终止，保留Q表，然后重新开始。

2-1 Q-learning

根据以上条件，采用Q-learning算法，该算法的伪代码如下：

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$;
  until S is terminal

参数解释：

$\epsilon - greedy$是用在决策上的一种策略，比如 epsilon = 0.9 时, 就说明有90% 的情况会按照Q-table的max-value选择行为, 10% 的情况使用随机选择的行为。

α：学习率, 来决定这次的误差有多少是要被学习的

γ：对未来 reward 的衰减值.

2-2 SARAS

saras算法假设前一时刻的状态价值的值$q_{t-1}(s_{t-1}$是最优的，利用当前的行动状态值$q_{t-1}(s_t, a_t)$和奖励值r_t来更新$q_t(s_t, a_t)$，公式如下：

$$q_t(s_t, a_t) = q_{t-1}(s_{t-1}, a_{t-1}) + \frac{1}{N}(r_t + \gamma * q_{t-1}(s_t, a_t) - q_{t-1}(s_{t-1}, a_{t-1}))$$

算法伪代码如下：

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

参数解释：

$\epsilon - greedy$是用在决策上的一种策略，比如 epsilon = 0.9 时, 就说明有90% 的情况会按照Q-table的max-value选择行为, 10% 的情况使用随机选择的行为。

α：学习率, 来决定这次的误差有多少是要被学习的

γ：对未来 reward 的衰减值.

3、关键代码展示

根据伪代码，python实现如下：

3-1 Q-learning

```
# Q-learning
def q_learning():
    Q_table=np.zeros((rows*cols,4)) #Q表
    clif_map=np.zeros((rows,cols))  #地图，用于每次存储路径
    rw_list=[]  #记录Reward的变化
    for step in range(episodes):
        #初始化S
        sum=0
        pos=(3,0)# 起点
        end_flag=False# 结束标志
        clif_map=np.zeros((rows,cols))# 记录路径
        clif_map[3][0]=1
        while(not end_flag):#判断是否到达目的地
            loc=int(12 * pos[0] + pos[1]) #计算此时的状态
            mov=epsilon_greedy(loc,Q_table) #根据Q表用epsilon_greedy选择动作
            pos=move(pos,mov) # 移动
            clif_map[pos[0]][pos[1]]=1
            new_loc=int(12 * pos[0] + pos[1]) #计算新状态
            new_loc,rwd, end_flag= reward(new_loc) #计算reward并判断是否结束或掉下悬崖
            sum+=rwd
            max_q_s_a=find_max(Q_table,pos) #选取当前状态最大Q表项
            # 根据公式更新Q表
            Q_table[loc][mov]=Q_table[loc][mov]+alpha*(rwd+(gamma*max_q_s_a)-
Q_table[loc][mov])
            pos=(int(new_loc/12),int(new_loc%12)) #更新位置
        rw_list.append(sum)
    return clif_map,Q_table,rw_list
```

3-2 Saras

```
# sarsa
def sarsa():
    Q_table=np.zeros((rows*cols,4))#Q表
    clif_map=np.zeros((rows,cols))#地图，用于每次存储路径
    rw_list=[]
    for step in range(episodes):
        sum=0
        pos=(3,0)# 起点
```

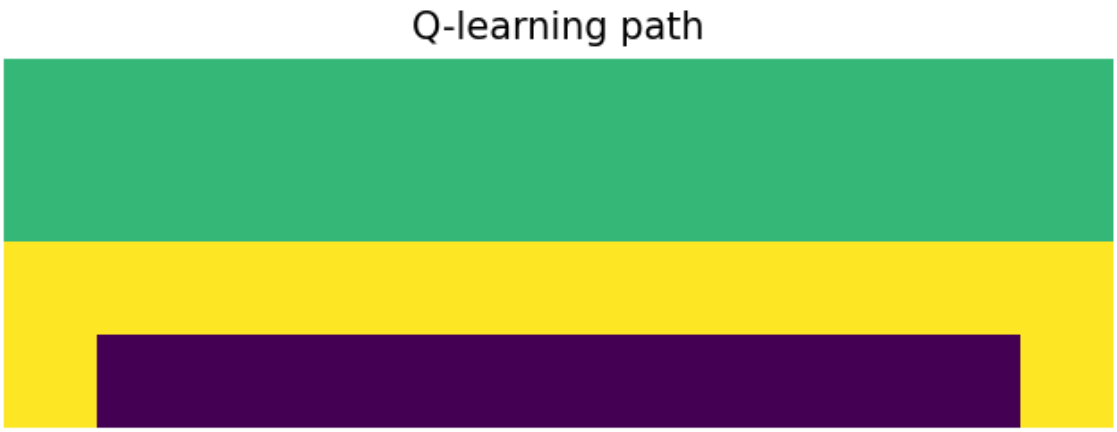
```
        end_flag=False# 结束标志
        clif_map=np.zeros((rows,cols))
        clif_map[3][0]=1
        loc=int(12 * pos[0] + pos[1])
        mov=epsilon_greedy(loc,Q_table)
        while(not end_flag):
            loc=int(12 * pos[0] + pos[1])#当前状态
            pos=move(pos,mov)#根据动作进行移动
            clif_map[pos[0]][pos[1]]=1
            new_loc=int(12 * pos[0] + pos[1])#新的状态
            new_loc,rwd, end_flag= reward(new_loc)#计算reward
            sum+=rwd
            #选择下一个状态的新动作
            next_mov=epsilon_greedy(new_loc,Q_table)
            q_s_a=Q_table[new_loc][next_mov]
            # 根据公式更新Q表
            Q_table[loc][mov]=Q_table[loc][mov]+alpha*(rwd+(gamma*q_s_a)-
Q_table[loc][mov])
            #更新位置和动作
            mov=next_mov
            pos=(int(new_loc/12),int(new_loc%12))
        rw_list.append(sum)
    return clif_map,Q_table,rw_list
```

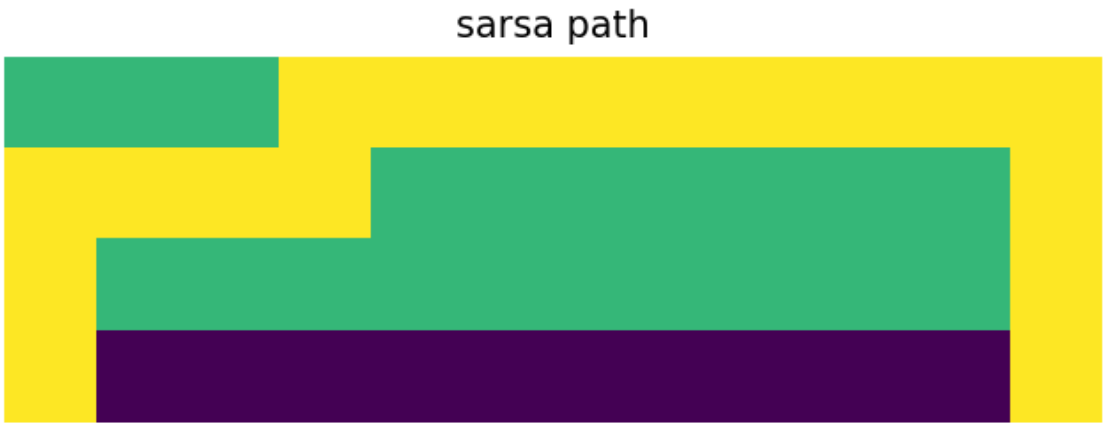
4、实验结果

初始条件设置为：

```
episodes = 500  #迭代次数
epsilon = 0.05  #选择随机方向的概率
alpha = 0.5     #学习率
gamma = 0.99    #衰减值
```

运行程序，得到分别使用Q-learning和sarsa算法的实验结果如下：
路径图：（绿色为可走区域，黄色为路径，紫色为悬崖）：

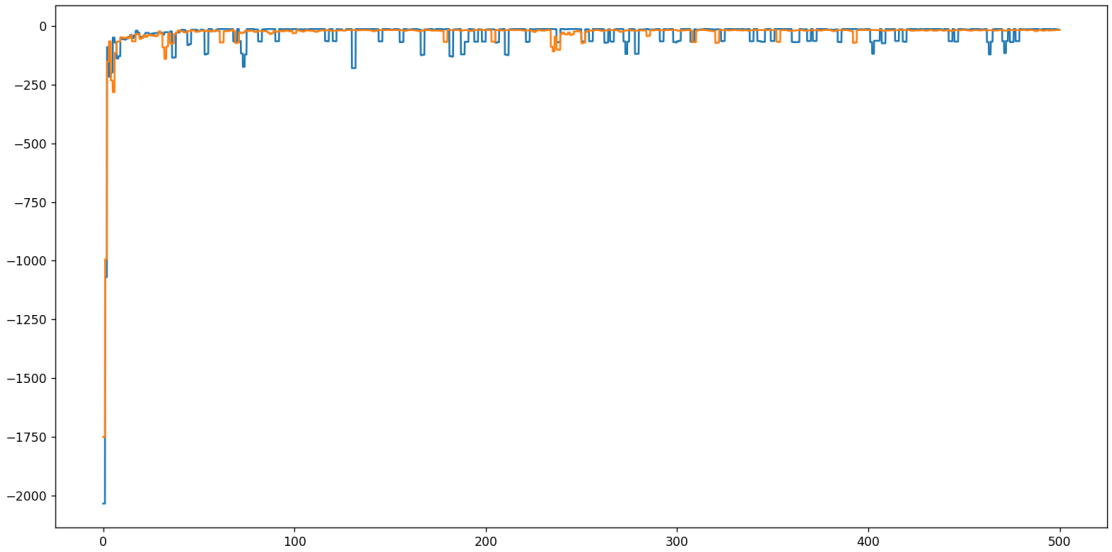




最后得到的Q-table如下：

```
Q_learnig table:

-11.113 -10.441 -9.939 -9.320 -8.628 -8.168 -7.236 -6.309 -5.305 -4.889 -3.931 -3.268 -11.335 -10.848 -10.150 -9.740 -8.617 -7.902 -6.885 -6.049 -5.039 -4.307 -2.986 -2.229 -11.
987 -11.074 -10.759 -10.062 -9.391 -8.341 -7.605 -6.779 -5.848 -4.872 -3.489 -2.970 -12.248 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000

-10.779 -10.571 -10.370 -9.768 -8.821 -8.168 -7.115 -6.410 -5.396 -4.650 -3.767 -2.938 -11.512 -10.903 -10.242 -9.483 -8.612 -7.708 -6.785 -5.849 -4.900 -3.940 -2.970 -1.990 -13
.116 -99.951 -98.438 -99.988 -99.805 -99.805 -98.438 -99.219 -96.875 -99.609 -99.219 -1.000 -13.097 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000

-10.889 -10.401 -10.109 -10.002 -9.301 -8.060 -7.682 -6.315 -5.905 -4.728 -3.657 -3.081 -11.271 -10.956 -10.175 -9.830 -9.716 -8.792 -7.597 -6.379 -5.802 -4.821 -4.628 -2.398 -1
2.238 -12.202 -11.348 -10.464 -9.469 -8.484 -7.619 -6.756 -5.797 -4.890 -3.800 -2.777 -13.124 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000

-10.876 -10.546 -9.956 -9.278 -8.510 -7.732 -6.938 -6.101 -5.295 -4.463 -3.732 -2.963 -11.257 -10.593 -10.140 -9.415 -8.593 -7.692 -6.781 -5.850 -4.900 -3.940 -2.970 -2.475 -11.
362 -10.466 -9.562 -8.648 -7.726 -6.793 -5.852 -4.901 -3.940 -2.970 -1.990 -1.986 -98.438 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000

----------------------------------------------------------------------------------------------------------------------------------------------------------------
----------------------------------------
sarsa table:

-16.0 -15.0 -13.2 -13.5 -11.6 -12.1 -10.8 -9.4 -9.1 -6.6 -5.3 -4.8 -16.2 -14.2 -13.1 -13.4 -11.3 -12.3 -11.8 -8.0 -7.7 -6.9 -4.4 -5.1 -15.6 -15.4 -13.4 -12.7 -13.0 -12.3 -11.9 -
11.1 -23.1 -7.2 -3.1 -4.5 -16.4 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

-16.8 -14.9 -14.4 -13.7 -12.7 -12.2 -10.4 -9.3 -9.0 -9.2 -5.1 -3.0 -17.6 -16.3 -47.7 -19.3 -12.8 -12.4 -11.7 -11.4 -23.4 -11.9 -12.7 -2.0 -18.6 -83.2 -99.6 -78.1 -51.0 -83.4 -51
.1 -51.1 -91.2 -51.5 -94.4 -1.0 -18.8 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

-16.2 -16.3 -15.6 -14.1 -13.7 -12.1 -11.6 -10.6 -9.9 -9.4 -6.8 -6.3 -16.3 -16.3 -14.3 -13.2 -13.1 -12.1 -11.8 -11.8 -10.7 -10.0 -9.1 -4.4 -17.1 -16.7 -14.0 -12.4 -12.7 -12.2 -30
.8 -11.6 -7.8 -15.8 -4.1 -2.3 -26.1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0

-14.8 -13.4 -12.3 -10.9 -9.8 -8.8 -7.8 -7.0 -6.1 -5.1 -4.0 -5.1 -14.9 -14.9 -15.9 -13.0 -12.9 -11.6 -9.1 -11.0 -12.2 -22.7 -3.5 -4.3 -18.7 -21.5 -12.5 -12.8 -12.5 -12.3 -12.1 -1
2.1 -11.1 -15.0 -2.1 -2.0 -115.9 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

在每次迭代的过程中，reward的总和的变化图如下，其中橙色的曲线为sarsa算法的reward总和变化，蓝色的曲线为q-learning算法的reward总和变化：



我们可以看到在on-line训练中，Q-learning的表现要比Sarsa差，但q-learning最后得到的结果比sarsa要好。并且通过分析可以看出两个算法的区别，Sarsa是一种on-policy算法，Q-learning是一种off-policy算法。Sarsa选取的是一种保守的策略，在更新Q值的时候已经为未来规划好了动作，对错误和死亡比较敏感。而Q-learning每次在更新的时候选取的是最大化Q的方向，而当下一个状态时，再重新选择动作，Q-learning是一种鲁莽、大胆、贪婪的算法，对于死亡和错误并不在乎。

5、实验心得

通过这次实验，我基于Cliff Walk例子实现了Q-learning算法和Sarsa算法，及时复习了课上所学的内容，并对两种算法有了更加深刻的理解，也对这两种算法的区别和各自的优点有了更深的体会。除此之外，通过将所学知识应用于实际问题中，我的代码编写能力和实践能力都有所提示，这让我受益匪浅。