Assignment 4

61075029H Jun Yu SHEN 沈峻宇

1. Temporal Differences TD(0):

Temporal Differences TD(0): $V(s) = V(s) + \alpha[r(s') + \gamma V(s') - V(s)]$

Input: the policy π that we want to evaluate
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize arbitrarily $\mathrm{V}(s)$ for all states $s \in S$, except that $\mathrm{V}(terminal) = 0$
Loop for each episode:
    Initialize s
    Loop for each step of episode:
        a = action given by policy π at s
        Take action a, and observe r, s_{t+1}
        $V(s) = V(s) + \alpha[r(s_{t+1}) + \gamma V(s_{t+1}) - V(s)]$
        $s = s_{t+1}$
    until s is terminal

Result:

```
---------------------------------------------------------
| T      | left   | right  | left   | left   | left   |
---------------------------------------------------------
| right  | right  | down   | up     | W      | up     |
---------------------------------------------------------
| down   | W      | up     | W      | down   | left   |
---------------------------------------------------------
| right  | right  | up     | left   | right  | left   |
---------------------------------------------------------

Optimal policy, T = Target, W = Wall
---------------------------------------------------------
| T      | left   | left   | left   | left   | left   |
---------------------------------------------------------
| up     | left   | left   | left   | W      | up     |
---------------------------------------------------------
| up     | W      | up     | W      | down   | up     |
---------------------------------------------------------
| up     | left   | up     | left   | left   | left   |
---------------------------------------------------------
```

2. Code:

```python
def iter(self): #Main loop 主要程式碼
    for iterration in range(self.Max_iteration):
        # 隨機決定位置
        self.current_state_coordinates = self.generate_initial_state()
        # 隨機決定動作
        action = self.generate_random_action()
        done,G = False,0
        while True:
            next_state_coordinates = self.env.transfer_state(self.current_state_coordinates, action)
            reward = -1
            if self.env.grid_world[next_state_coordinates[0]][next_state_coordinates[1]] == "T":
                reward = 100
                done = True
            returns = self.returns_dict[(tuple(self.current_state_coordinates), action)]
            if done:
                G = returns[0] + self.alpha * (reward - returns[0])
            else:
                next_action = self.policy(next_state_coordinates)
                next_returns = self.returns_dict[(tuple(next_state_coordinates), next_action)]
                G = returns[0] + self.alpha * (reward + self.gamma * next_returns[0] - returns[0])
            visited_count = returns[1]
            visited_count += 1
            self.returns_dict[(tuple(self.current_state_coordinates ), action)] = [G, visited_count]
            self.Q_values[(tuple(self.current_state_coordinates), action)] = self.returns_dict[(tuple(self.current_sta
            if done:
                break
            self.current_state_coordinates = next_state_coordinates
            action = next_action
```

I have modified this section.

Temporal Differences TD(0): $V(s) = V(s) + \alpha[r(s') + \gamma V(s') - V(s)]$

Input: the policy π that we want to evaluate
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize arbitrarily $\mathrm{V}(s)$ for all states $s \in S$, except that $\mathrm{V}(terminal) = 0$
Loop for each episode:
  Initialize s
  Loop for each step of episode:
    a = action given by policy π at s
    Take action a, and observe r, s_{t+1}
    $V(s) = V(s) + \alpha[r(s_{t+1}) + \gamma V(s_{t+1}) - V(s)]$
    $s = s_{t+1}$
  until s is terminal

Almost all of the calculations in the program are done here. Only this part was modified and it was done