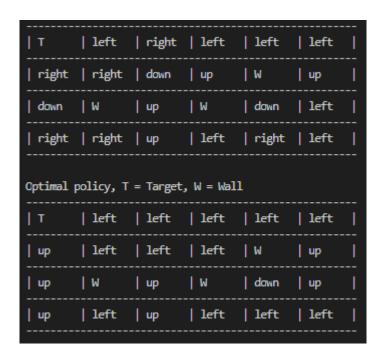
Assignment 4

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 \pi that we want to evaluate Algorithm parameter: step size \alpha \in (0,1] Initialize arbitrarily V(s) for all states s \in S, except that V(terminal) = 0 Loop for each episode: Initialize s Loop for each step of episode: a = action given by policy \pi 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:



2. Code:

```
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)
       if self.env.grid_world[next_state_coordinates[0]][next_state_coordinates[1]] == "T":
           reward = 100
       returns = self.returns_dict[(tuple(self.current_state_coordinates), action)]
        if done:
           G = returns[0] + self.alpha * (reward - returns[0])
           next_action = self.policy(next_state_coordinates)
           next_returns = self.returns_dict[(tuple(next_state_coordinates), next_action)]
           6 = 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)] = [6, visited_count]
        self.Q_values[(tuple(self.current_state_coordinates), action)] = self.returns_dict[(tuple(self.current_state_
        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 \pi that we want to evaluate
Algorithm parameter: step size \alpha \in (0, 1]
Initialize arbitrarily V(s) for all states s \in S, except that V(terminal) = 0
Loop for each episode:
Initialize s
Loop for each step of episode:
a = \text{action given by policy } \pi \text{ 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