Assignment2
Reinforcement Learning
Ferdowsi University of Mashhad Spring2023

Arya Ebrahimi 9822762175

Question1:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

But, this formula can be written recursively as follows:

$$G_t = R_{t+1} + \gamma \times G_{t+1}$$

A) $r_m = -4$:

1. Getting r_g

$$G_5 = 4$$
$$G_4 = R_5 + \gamma G_5 = -4 + 0.9 \times 4 = -0.4$$
$$G_3 = R_4 + \gamma G_4 = -4 + 0.9 \times -0.4 = -4.36$$
$$G_2 = R_3 + \gamma G_3 = -4 + 0.9 \times (-4.36) = -7.924$$
$$G_1 = R_2 + \gamma G_2 = -4 + 0.9 \times (-7.924) = -11.1316$$
$$G_0 = R_1 + \gamma G_1 = -4 + 0.9 \times (-11.1316) = -14.01844$$

2. Getting r_a

$$G_3 = -4$$
$$G_2 = R_3 + \gamma G_3 = -4 + 0.9 \times -4 = -7.6$$
$$G_1 = R_2 + \gamma G_2 = -4 + 0.9 \times (-7.6) = -10.84$$
$$G_0 = R_1 + \gamma G_1 = -4 + 0.9 \times (-10.84) = -13.756$$

B) $r_m = -1$:

1. Getting r_g

$$G_5 = 4$$
$$G_4 = R_5 + \gamma G_5 = -1 + 0.9 \times 4 = 2.6$$
$$G_3 = R_4 + \gamma G_4 = -1 + 0.9 \times 2.6 = 1.34$$
$$G_2 = R_3 + \gamma G_3 = -1 + 0.9 \times (1.34) = 0.206$$
$$G_1 = R_2 + \gamma G_2 = -1 + 0.9 \times (0.206) = -0.8146$$
$$G_0 = R_1 + \gamma G_1 = -1 + 0.9 \times (-0.8146) = -1.73314$$

2. Getting r_a

$$G_3 = -4$$
$$G_2 = R_3 + \gamma G_3 = -1 + 0.9 \times -4 = -4.6$$
$$G_1 = R_2 + \gamma G_2 = -1 + 0.9 \times (-4.6) = -5.14$$
$$G_0 = R_1 + \gamma G_1 = -1 + 0.9 \times (-5.14) = -5.626$$

C) $r_m = 0$:

1. Getting r_g

$$G_5 = 4$$
$$G_4 = R_5 + \gamma G_5 = 0.9 \times 4 = 3.6$$
$$G_3 = R_4 + \gamma G_4 = 0.9 \times 3.6 = 3.24$$
$$G_2 = R_3 + \gamma G_3 = 0.9 \times (3.24) = 2.916$$
$$G_1 = R_2 + \gamma G_2 = 0.9 \times (2.916) = 2.6244$$
$$G_0 = R_1 + \gamma G_1 = 0.9 \times (2.6244) = 2.36196$$

2. Getting r_a

$$G_3 = -4$$
$$G_2 = R_3 + \gamma G_3 = 0.9 \times (-4) = -3.6$$
$$G_1 = R_2 + \gamma G_2 = 0.9 \times (-3.6) = -3.24$$
$$G_0 = R_1 + \gamma G_1 = 0.9 \times (-3.24) = -2.916$$

D) $r_m = 2$:

1. Getting r_g

$$G_5 = 4$$
$$G_4 = R_5 + \gamma G_5 = +2 + 0.9 \times 4 = 5.6$$
$$G_3 = R_4 + \gamma G_4 = +2 + 0.9 \times 5.6 = 7.04$$
$$G_2 = R_3 + \gamma G_3 = +2 + 0.9 \times (7.04) = 8.336$$
$$G_1 = R_2 + \gamma G_2 = +2 + 0.9 \times (8.336) = 9.5024$$
$$G_0 = R_1 + \gamma G_1 = +2 + 0.9 \times (9.5024) = 10.55216$$

2. Getting r_a

$$G_3 = -4$$
$$G_2 = R_3 + \gamma G_3 = +2 + 0.9 \times -4 = -1.6$$
$$G_1 = R_2 + \gamma G_2 = +2 + 0.9 \times (-1.6) = 0.56$$
$$G_0 = R_1 + \gamma G_1 = +2 + 0.9 \times (0.56) = 2.504$$

Question2:

The Bellman equation can be used to find the state-value function.

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t | S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a)[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r | s, a)[r + \gamma v_\pi(s')], \quad \text{for all s} \in \text{S} \end{aligned}$$

Using the Bellman equation, the state values form a linear equation system which can be solved to find the values. Since our policy is random in this question, then $\pi(a|s)$ would be 0.5 for each action. However, solving the Bellman equation using the linear equation system can be computationally expensive or impossible in more oversized cases. So, an iterative approach is implemented to find the values (value evaluation).

CODE FOR CALCULATING STATE VALUES FOR RANDOM POLICY

```
WORLD_SIZE = 6
GOAL = [2, 5]
DISCOUNT = 0.9
GOAL_REWARD = +4.0
RED_REWARD = -4.0
DEFAULT_REWARD = 2.0

# up-right and down-right
ACTIONS = [np.array([-1, 1]),
        np.array([1, 1])]
# up action
UP = np.array([-1, 0])
RANDOM_TRANSITION = np.ones((WORLD_SIZE, WORLD_SIZE, 2))/2
red = []
for i in range(WORLD_SIZE):
   red.append([0, i])
   red.append((WORLD_SIZE-1, i))
red = red + [[1, 2], [1, 3], [4, 1], [4, 4]]
```

```
def transition(state, action):

  if state == GOAL:
     return state, GOAL_REWARD, True
  if state in red:
     return state, RED_REWARD, True

  next_state = (np.array(state) + action).tolist()
  x, y = next_state
  reward = DEFAULT_REWARD
  if y >= WORLD_SIZE:
     next_state = state + UP
  if x < 0 or x >= WORLD_SIZE or y < 0:
     next_state = state

  return next_state, reward, False
```

First, the transition function is implemented, which takes a state and an action and returns the next state and the reward. Next, the main loop is implemented, which iterates over all states and updates their state-values using the Bellman equation.

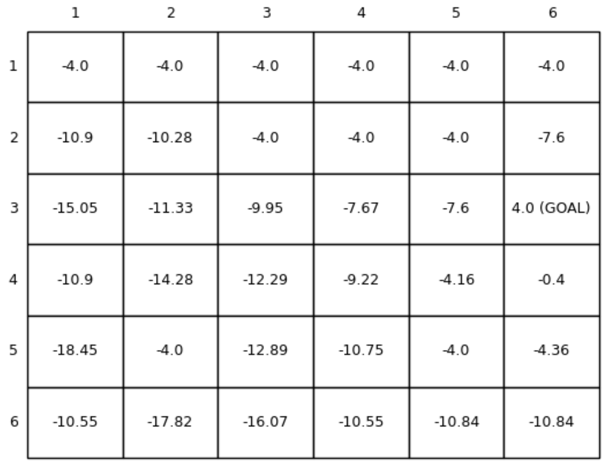
```
def value_evaluation(p=RANDOM_TRANSITION):
   value = np.zeros((WORLD_SIZE, WORLD_SIZE))
   for _ in range(100):
      new_value = np.zeros_like(value)
      for i in range(WORLD_SIZE):
         for j in range(WORLD_SIZE):
            values = []
            for index, action in enumerate(ACTIONS):
               (next_i, next_j), reward, terminal = transition([i, j], action)
               if terminal:
                  values.append(p[i, j, index]*(reward))
               else:
                  values.append(p[i, j, index]*(reward + DISCOUNT * value[next_i, next_j]))

            new_value[i, j] = np.sum(values)
      value = new_value
   return value
```

Note that if state is a terminal state, then its new value is only the reward its gets.
Using these values shown in Figure1, Figure2, Figure3 and Figure4, the action-values can also be calculated.

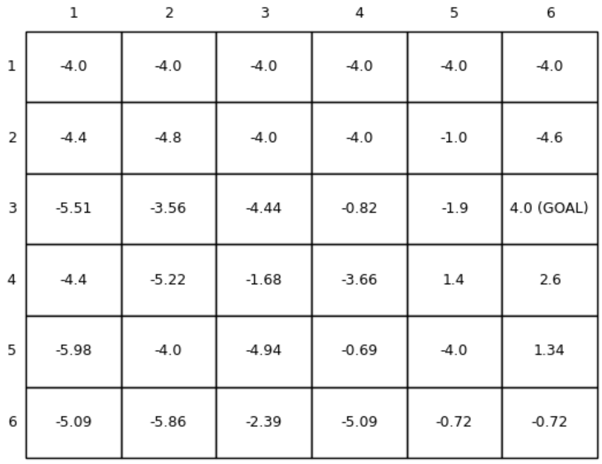
We know that $\upsilon_\pi(s) = \sum_a \pi(a|s) q_\pi(s,a)$, so using the Bellman equation we can define $q_\pi(s,a)$ as follows:

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \upsilon_\pi(s')]$$



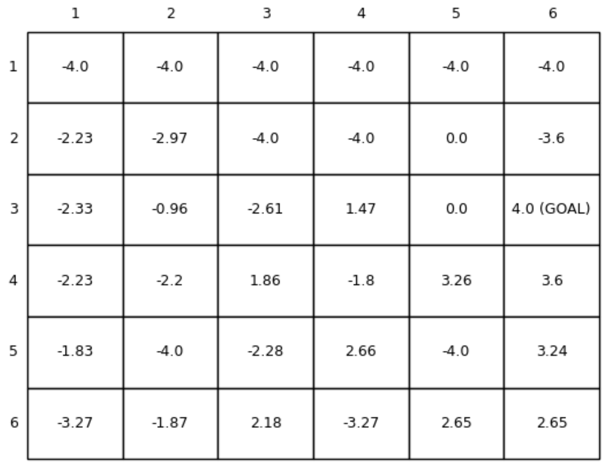
	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	-10.9	-10.28	-4.0	-4.0	-4.0	-7.6
3	-15.05	-11.33	-9.95	-7.67	-7.6	4.0 (GOAL)
4	-10.9	-14.28	-12.29	-9.22	-4.16	-0.4
5	-18.45	-4.0	-12.89	-10.75	-4.0	-4.36
6	-10.55	-17.82	-16.07	-10.55	-10.84	-10.84

Figure 1: State values for $r_m = -4$ following a random policy.



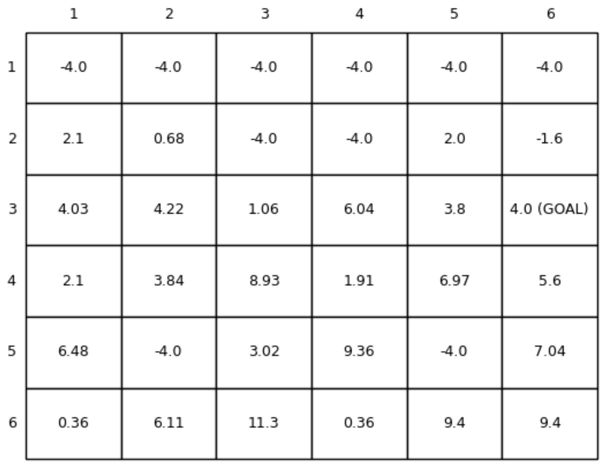
	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	-4.4	-4.8	-4.0	-4.0	-1.0	-4.6
3	-5.51	-3.56	-4.44	-0.82	-1.9	4.0 (GOAL)
4	-4.4	-5.22	-1.68	-3.66	1.4	2.6
5	-5.98	-4.0	-4.94	-0.69	-4.0	1.34
6	-5.09	-5.86	-2.39	-5.09	-0.72	-0.72

Figure 2: State values for $r_m = -1$ following a random policy.



	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	-2.23	-2.97	-4.0	-4.0	0.0	-3.6
3	-2.33	-0.96	-2.61	1.47	0.0	4.0 (GOAL)
4	-2.23	-2.2	1.86	-1.8	3.26	3.6
5	-1.83	-4.0	-2.28	2.66	-4.0	3.24
6	-3.27	-1.87	2.18	-3.27	2.65	2.65

Figure 3: State values for $r_m = 0$ following a random policy.



	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	2.1	0.68	-4.0	-4.0	2.0	-1.6
3	4.03	4.22	1.06	6.04	3.8	4.0 (GOAL)
4	2.1	3.84	8.93	1.91	6.97	5.6
5	6.48	-4.0	3.02	9.36	-4.0	7.04
6	0.36	6.11	11.3	0.36	9.4	9.4

Figure 4: State values for $r_m = 2$ following a random policy.

Thus, we can calculate the quality of state 15 and 23 with UP-RIGHT action as below:

$$\mathrm{q}_\pi(15, \text{UP-RIGHT}) == \begin{cases} r_m = -4 & -4 + 0.9 \times v_\pi(22) = -4 + 0.9 \times (-7.67) = -10.903 \\ r_m = -1 & -1 + 0.9 \times v_\pi(22) = -1 + 0.9 \times (-0.82) = -1.738 \\ r_m = 0 & 0.9 \times v_\pi(22) = 0.9 \times 1.47 = 1.323 \\ r_m = +2 & +2 + 0.9 \times v_\pi(22) = +2 + 0.9 \times 6.04 = 7.436 \end{cases}$$

$$\mathrm{q}_\pi(23, \text{UP-RIGHT}) == \begin{cases} r_m = -4 & -4 + 0.9 \times v_\pi(30) = -4 + 0.9 \times (-7.6) = -10.84 \\ r_m = -1 & -1 + 0.9 \times v_\pi(30) = -1 + 0.9 \times (-4.6) = -5.14 \\ r_m = 0 & 0.9 \times v_\pi(30) = 0.9 \times (-3.6) = -3.24 \\ r_m = +2 & +2 + 0.9 \times v_\pi(30) = +2 + 0.9 \times (-34) = 0.56 \end{cases}$$

Question3:

a) If $r_m = -4$, then the expected return following a policy that reaches the goal would be -11.1316 (using $\gamma = 0.9$), which is much smaller than the expected return for going to state 32. this happens because, at each timestep, the agent receives a large negative reward that cancels out the positive reward of reaching to the goal. Thus, the agent decides to reach a terminal state as soon as possible without considering its reward, so it takes the "up-right" action. Thus, in this case, the optimal policy is to take the "up-right" action and terminate the episode. Since no terminal state except state 32 can be reached in one step from state 25, this policy is unique. (for $\gamma = 0.9$)

b, c) If $r_m = -1$ or 0, the expected return for reaching the goal would be higher than bumping into red states, so the agent will follow a good policy that brings it to the goal state. However, the optimal policy is not unique in this case. 25 → 20 → 15 → 10 → 17 → 24, 25 → 20 → 15 → 22 → 17 → 24 and 25 → 20 → 15 → 22 → 29 → 24 are optimal policies.

d) If $r_m = 2$, the expected value would be higher if the agent lengthens its path because it receives a $+2$ reward in each timestep. So, the optimal policy would be 25 → 20 → 15 → 10 → 17 → 12 → 18 → 24 or 25 → 20 → 15 → 22 → 17 → 12 → 18 → 24.

In all these cases, if $\gamma = 0$, then the agent will choose randomly in blue states because it just chooses based on the reward it gets, and the reward is the same for all actions in blue states. For example, if the agent is in state 23, it chooses randomly between state 18 and state 30, which has a huge difference. In general, When the γ is set to 1, the agent places equal weight on immediate and future rewards. In this case, the optimal policy will prioritize actions that maximize the total expected reward over the entire future horizon. On the other hand, when the discount factor gamma is set to a value less than 1, the agent places more weight on immediate rewards than future rewards. In this case, the optimal policy will prioritize actions that maximize the immediate rewards over long-term rewards.

Question4:

As discussed in the previous question, the $r_m = -1$ and $r_m = 0$ cases are causing the agent to find a policy that follows the shortest path to the goal state.

$$\begin{aligned} v_*(s) &= \max_{a \in \mathbb{A}(s)} q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}[G_t | S_t = s, A_t = a] \\ &= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1} | S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')] \end{aligned}$$

A) $r_m = -1$

1. starting state = 25

   optimal policy = 25 → 20 → 15 → 10 → 17 → 24 → terminate

$$G_5 = 4$$
$$G_4 = R_5 + \gamma G_5 = -1 + 0.9 \times 4 = 2.6$$
$$G_3 = R_4 + \gamma G_4 = -1 + 0.9 \times 2.6 = 1.34$$
$$G_2 = R_3 + \gamma G_3 = -1 + 0.9 \times (1.34) = 0.206$$
$$G_1 = R_2 + \gamma G_2 = -1 + 0.9 \times (0.206) = -0.8146$$
$$G_0 = R_1 + \gamma G_1 = -1 + 0.9 \times (-0.8146) = -1.73314$$

$$v_*(25) = -1.73314$$

2. starting state = 33

   optimal policy = 33 → 28 → terminate

$$G_0 = -4$$

$$v_*(33) = -4$$

3. starting state = 22

optimal policy = 22 → 17 → 24 → terminate

$$G_2 = 4$$
$$G_1 = R_2 + \gamma G_2 = -1 + 0.9 \times 4 = 2.6$$
$$G_0 = R_1 + \gamma G_1 = -1 + 0.9 \times 2.6 = 1.34$$

$$v_*(22) = 1.34$$

4. starting state = 30

optimal policy = 30 → 36 → terminate

$$G_1 = -4$$
$$G_0 = R_1 + \gamma G_1 = -1 + 0.9 \times (-4) = -4.6$$

$$v_*(30) = -4.6$$

B) $r_m = 0$

- starting state = 25

optimal policy = 25 → 20 → 15 → 10 → 17 → 24 → terminate

$$G_5 = 4$$
$$G_4 = R_5 + \gamma G_5 = 0.9 \times 4 = 3.6$$
$$G_3 = R_4 + \gamma G_4 = 0.9 \times 3.6 = 3.24$$
$$G_2 = R_3 + \gamma G_3 = 0.9 \times (3.24) = 2.916$$
$$G_1 = R_2 + \gamma G_2 = 0.9 \times (2.916) = 2.6244$$
$$G_0 = R_1 + \gamma G_1 = 0.9 \times (2.6244) = 2.36196$$

$$v_*(25) = 2.36196$$

- starting state = 33

optimal policy = 33 → 28 → terminate

$$G_0 = -4$$

$$v_*(33) = -4$$

- starting state = 22

optimal policy = 22 → 17 → 24 → terminate

$$G_2 = 4$$
$$G_1 = R_2 + \gamma G_2 = 0.9 \times 4 = 3.6$$
$$G_0 = R_1 + \gamma G_1 = 0.9 \times 3.6 = 3.24$$

$$v_*(22) = 3.24$$

- starting state = 30

optimal policy = 30 → 36 → terminate

$$G_1 = -4$$
$$G_0 = R_1 + \gamma G_1 = 0.9 \times (-4) = -3.6$$

$$v_*(30) = -3.6$$

	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	-1.73	-4.6	-4.0	-4.0	2.6	-4.6
3	-5.14	-0.81	-0.81	1.34	1.34	4.0 (GOAL)
4	-1.73	-1.73	0.21	0.21	2.6	2.6
5	-2.56	-4.0	-0.81	1.34	-4.0	1.34
6	-4.6	-1.73	0.21	-4.6	0.21	0.21

Figure 5: State values for $r_m = -1$ following a optimal policy.

	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	2.36	2.36	-4.0	-4.0	3.6	-3.6
3	2.13	2.62	2.62	3.24	3.24	4.0 (GOAL)
4	2.36	2.36	2.92	2.92	3.6	3.6
5	0.0	-4.0	2.62	3.24	-4.0	3.24
6	0.0	0.0	2.92	0.0	2.92	2.92

Figure 6: State values for $r_m = 0$ following a optimal policy.

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')]$$

$$\text{q}_*(15, \text{UP-RIGHT}) == \begin{cases} r_m = -1 & -1 + 0.9 \times v_*(22) = -1 + 0.9 \times 1.34 = 0.206 \\ r_m = 0 & 0.9 \times v_*(22) = 0.9 \times 3.24 = 2.916 \end{cases}$$

$$\text{q}_*(23, \text{UP-RIGHT}) == \begin{cases} r_m = -1 & -1 + 0.9 \times v_*(30) = -1 + 0.9 \times (-4.6) = -5.14 \\ r_m = 0 & 0.9 \times v_*(30) = 0.9 \times (-3.6) = -3.24 \end{cases}$$

Question5:

A) $\gamma = 0$:

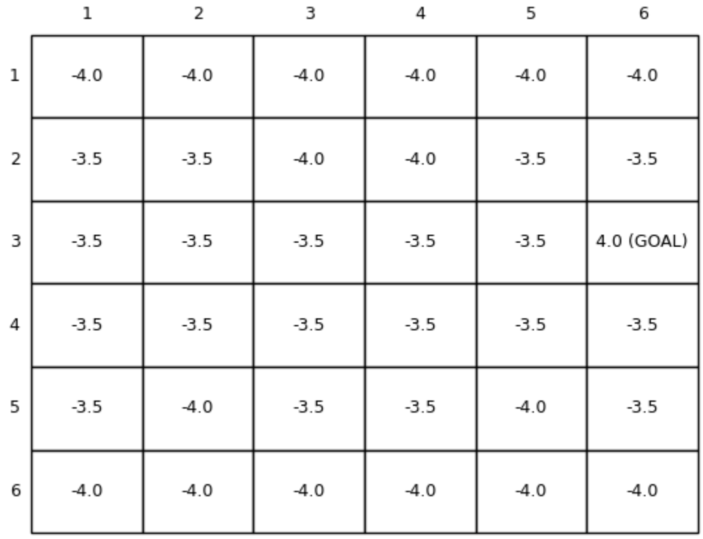
In this case, the values would only depend on the short-term reward. If the first policy is random, only one step of value evaluation will provide a better policy than a random policy because the value of red states would be -4 and blue states would be 1. The policy stops the car from bumping into a red state.

	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	1.0	1.0	-4.0	-4.0	1.0	1.0
3	1.0	1.0	1.0	1.0	1.0	4.0 (GOAL)
4	1.0	1.0	1.0	1.0	1.0	1.0
5	1.0	-4.0	1.0	1.0	-4.0	1.0
6	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0

Figure 7: State values when $\gamma = 0$ and $r_m = 1$with initial values of -5.

B) $0 < \gamma < 1$:

Like the previous case, the policy will change from a random policy to a better one that prevents bumping into a red state.

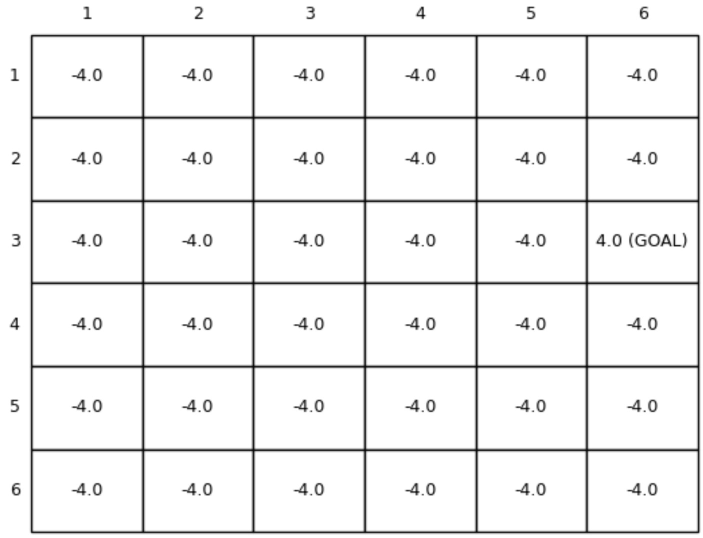


	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	-3.5	-3.5	-4.0	-4.0	-3.5	-3.5
3	-3.5	-3.5	-3.5	-3.5	-3.5	4.0 (GOAL)
4	-3.5	-3.5	-3.5	-3.5	-3.5	-3.5
5	-3.5	-4.0	-3.5	-3.5	-4.0	-3.5
6	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0

Figure 8: State values when $\gamma = 0.9$ and $r_m = 1$with initial values of -5.

C) $\gamma = 1$:

However, in this case, the total value of the following states would be considered to calculate the current state value. Since the initial value is -5 and the reward is +1, the sum of them would be -4, which is the same as the red state reward, so the policy stays random.



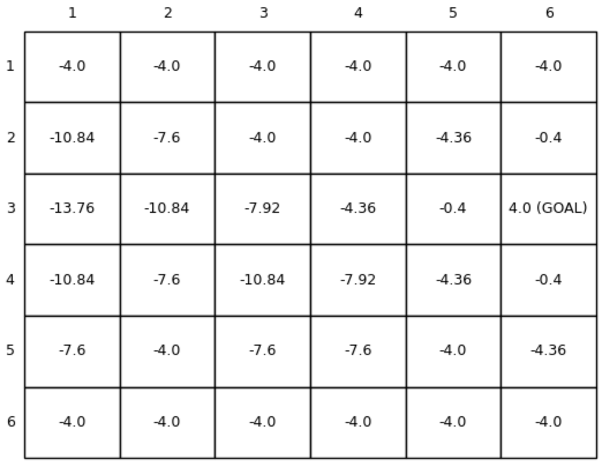
	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
3	-4.0	-4.0	-4.0	-4.0	-4.0	4.0 (GOAL)
4	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
5	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
6	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0

Figure 9: State values when $\gamma = 1$ and $r_m = 1$with initial values of -5.

Question6:

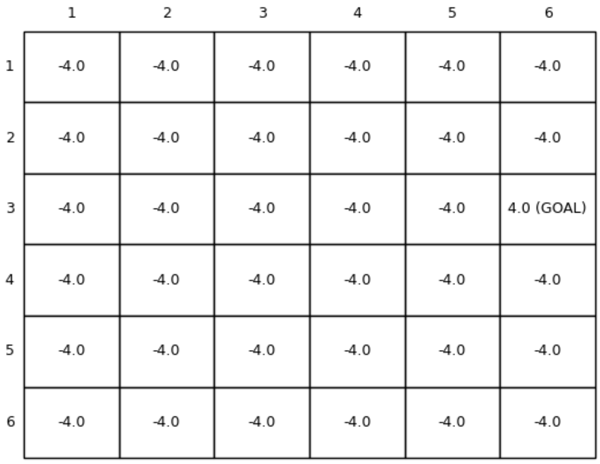
A) $r_e = -4$:

For the case that $\gamma > 0$, as shown in Figure 10 since at each timestep reward is -4, it cancels out the reward of the goal, and the value of state 25 would be small, so it chooses a path to a red state to terminate as fast as possible as optimal policy. However, for the case of $\gamma = 0$, the value of all states would be equal to -4 because it only depends on the reward, so the policy would be random except for the states 23 and 30. Another part of the code is implemented to choose the greedy action and improve the policy based on the evaluated values.



	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	-10.84	-7.6	-4.0	-4.0	-4.36	-0.4
3	-13.76	-10.84	-7.92	-4.36	-0.4	4.0 (GOAL)
4	-10.84	-7.6	-10.84	-7.92	-4.36	-0.4
5	-7.6	-4.0	-7.6	-7.6	-4.0	-4.36
6	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0

Figure 11: State values when $r_e = -4$ and $\gamma = 0.9$ following an optimal policy.

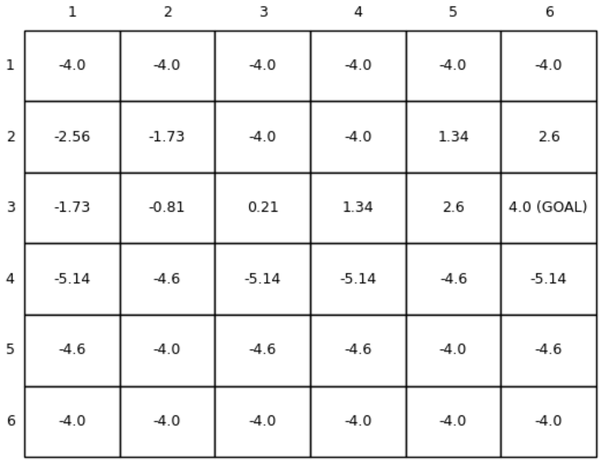


	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
3	-4.0	-4.0	-4.0	-4.0	-4.0	4.0 (GOAL)
4	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
5	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
6	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0

Figure 10: State values when $r_e = -4$ and $\gamma = 0$ following an optimal policy.

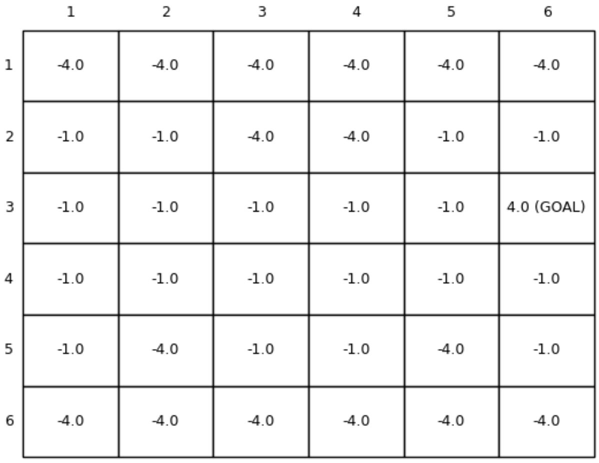
B) $r_e \in \{-1, 0, 2\}$:

In this case, if $\gamma > 0$, then the optimal policy would be the one which returns the shortest path to the goal from state 25, but if $\gamma = 0$, then in blue states, the policy would be random since it only depends on the rewards that it receives.



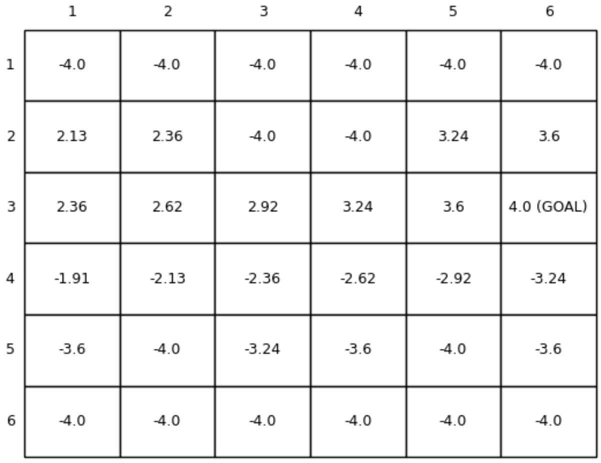
	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	-2.56	-1.73	-4.0	-4.0	1.34	2.6
3	-1.73	-0.81	0.21	1.34	2.6	4.0 (GOAL)
4	-5.14	-4.6	-5.14	-5.14	-4.6	-5.14
5	-4.6	-4.0	-4.6	-4.6	-4.0	-4.6
6	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0

Figure 12: State values when $r_e = -1$ and $\gamma = 0.9$ following an optimal policy.



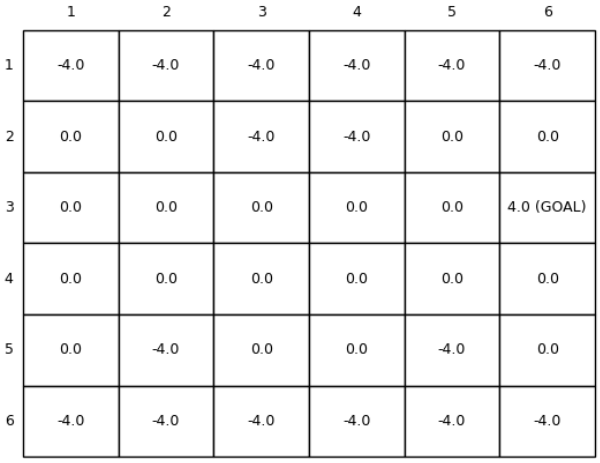
	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	-1.0	-1.0	-4.0	-4.0	-1.0	-1.0
3	-1.0	-1.0	-1.0	-1.0	-1.0	4.0 (GOAL)
4	-1.0	-1.0	-1.0	-1.0	-1.0	-1.0
5	-1.0	-4.0	-1.0	-1.0	-4.0	-1.0
6	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0

Figure 13: State values when $r_e = -1$ and $\gamma = 0$ following an optimal policy.



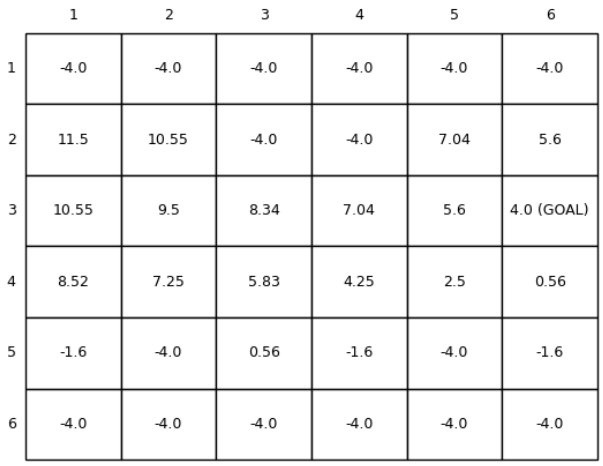
	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	2.13	2.36	-4.0	-4.0	3.24	3.6
3	2.36	2.62	2.92	3.24	3.6	4.0 (GOAL)
4	-1.91	-2.13	-2.36	-2.62	-2.92	-3.24
5	-3.6	-4.0	-3.24	-3.6	-4.0	-3.6
6	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0

Figure 14: State values when $r_e = 0$ and $\gamma = 0.9$ following an optimal policy.



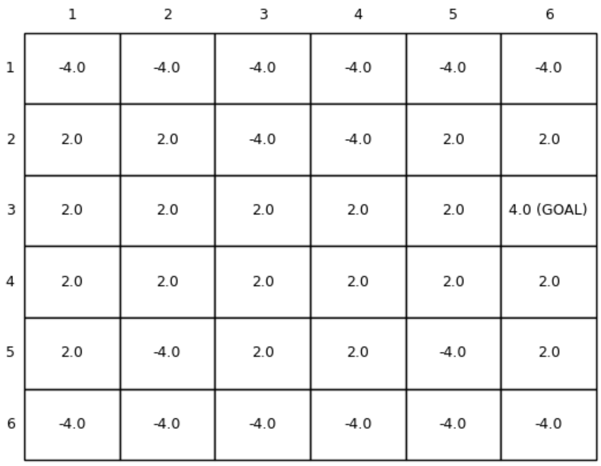
	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	0.0	0.0	-4.0	-4.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	4.0 (GOAL)
4	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	-4.0	0.0	0.0	-4.0	0.0
6	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0

Figure 15: State values when $r_e = 0$ and $\gamma = 0$ following an optimal policy.



	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	11.5	10.55	-4.0	-4.0	7.04	5.6
3	10.55	9.5	8.34	7.04	5.6	4.0 (GOAL)
4	8.52	7.25	5.83	4.25	2.5	0.56
5	-1.6	-4.0	0.56	-1.6	-4.0	-1.6
6	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0

Figure 16: State values when $r_e = +2$ and $\gamma = 0.9$ following an optimal policy.

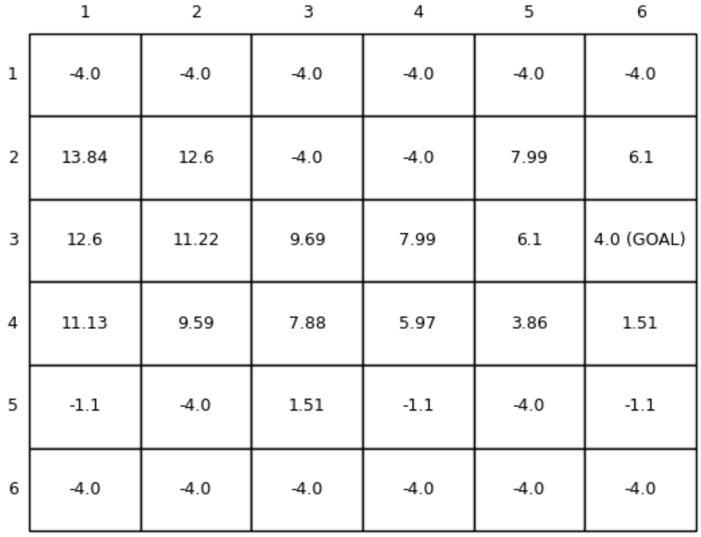


	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	2.0	2.0	-4.0	-4.0	2.0	2.0
3	2.0	2.0	2.0	2.0	2.0	4.0 (GOAL)
4	2.0	2.0	2.0	2.0	2.0	2.0
5	2.0	-4.0	2.0	2.0	-4.0	2.0
6	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0

Figure 17: State values when $r_e = +2$ and $\gamma = 0$ following an optimal policy.

Question7:

If $r_m = 0$, the only reward it gets using the regular actions is +4 when reaching the goal state. Thus, for $r_e > 0$, the reward would be strictly higher using efficient actions. However, if r_e becomes much larger, the optimal policy would change so that the agent will lengthen its path and not reach the goal state. When $\gamma = 0.9$, the range of r_e could be $(0, 2.5]$.



	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	13.84	12.6	-4.0	-4.0	7.99	6.1
3	12.6	11.22	9.69	7.99	6.1	4.0 (GOAL)
4	11.13	9.59	7.88	5.97	3.86	1.51
5	-1.1	-4.0	1.51	-1.1	-4.0	-1.1
6	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0

Figure 18: State values following an optimal policy when $r_e = 2.5$. If r_e becomes larger, the value of state 17 will become bigger than state 24, causing the optimal policy to change from reaching the goal.

Question8:

Since there are no up-action efficient actions, states below state 24 cannot reach state 24 using efficient actions, so states {7, 9, 10, 12, 13, 14, 15, 16, 17, 18} can only use regular actions to reach state 24.

States {19, 20, 21, 22, 23, 25, 26, 29} can use both regular and efficient actions, and state 30 is the only state that can reach the goal only using efficient actions.

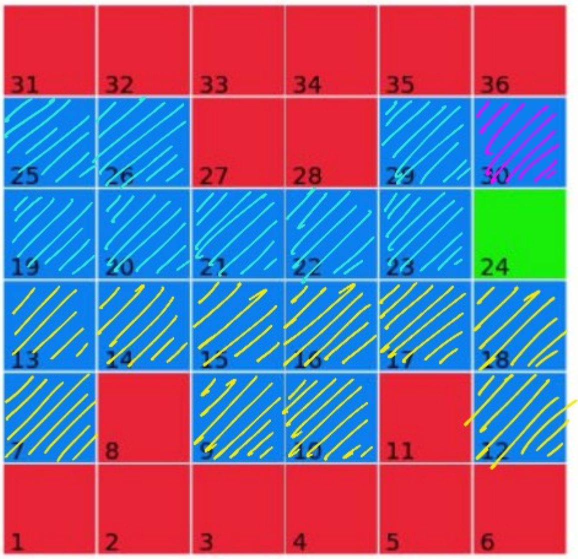


Figure 19:

Cyan → states that can use both actions.

Yellow → states witch can reach the goal only using the regular actions.

Purple → state which can reach the goal only using the efficient actions.

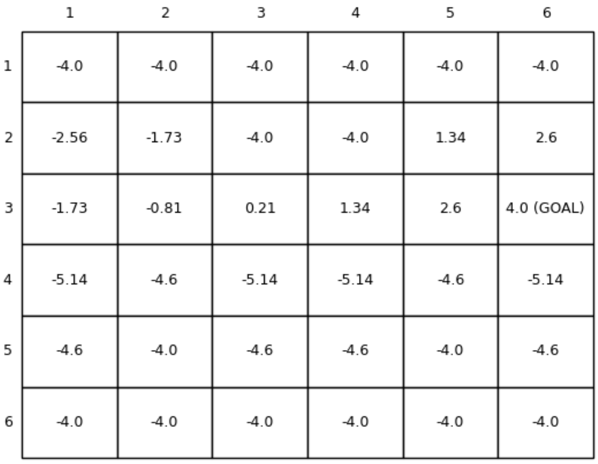
Question9:

Yes, it can change the optimal policy. For example, adding a constant number so that the agent receives a positive reward at each timestep can lengthen the path to the same state.

Agent	S2, -1	S4, -1	S6, +5	Agent	S2, +4	S4, +4	S6, +10
S1, -1	S3, -1	S5, -1	S7, -1	S1, +4	S3, +4	S5, +4	S7, +4

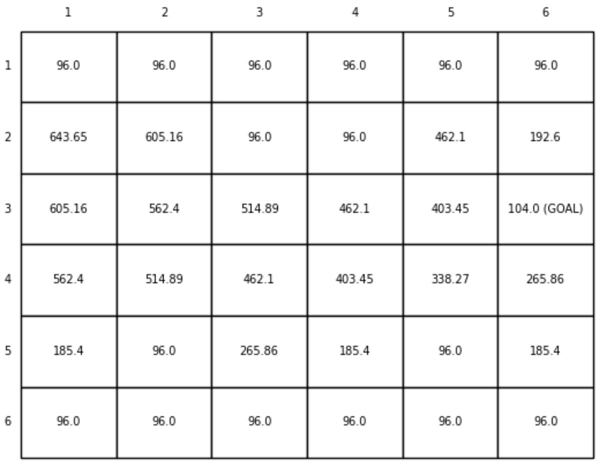
The optimal policy for the left environment is S2 → S4 → S6, but for the right environment, the optimal policy is to visit every state to gain more rewards. so it would be S1 → S3 → S2 → S4 → S5 → S7 → S6.

$$R_{new} = \sum_{i=0}^{K}(r_i + c)\gamma^i = \sum_{i=0}^{K} r_i\gamma^i + \sum_{i=0}^{K} c\gamma^i$$



	1	2	3	4	5	6
1	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0
2	-2.56	-1.73	-4.0	-4.0	1.34	2.6
3	-1.73	-0.81	0.21	1.34	2.6	4.0 (GOAL)
4	-5.14	-4.6	-5.14	-5.14	-4.6	-5.14
5	-4.6	-4.0	-4.6	-4.6	-4.0	-4.6
6	-4.0	-4.0	-4.0	-4.0	-4.0	-4.0

Figure 20: State values following an optimal policy $r_g = +4, r_a = -4, r_e = -1.$



	1	2	3	4	5	6
1	96.0	96.0	96.0	96.0	96.0	96.0
2	643.65	605.16	96.0	96.0	462.1	192.6
3	605.16	562.4	514.89	462.1	403.45	104.0 (GOAL)
4	562.4	514.89	462.1	403.45	338.27	265.86
5	185.4	96.0	265.86	185.4	96.0	185.4
6	96.0	96.0	96.0	96.0	96.0	96.0

Figure 21: State values following an optimal policy $r_g = +104, r_a = 96, r_e = 99.$

As shown in Figure 20 and Figure 21, the optimal policy is to reach the goal as soon as possible, but when c=100 is added, if the agent is at state 23, it will choose state 17 instead of 24.

Question10:

Some people may want to reach their destination faster, and some want to be efficient. Thus, one configuration for all is not a good decision. A better approach might be changing the rewards of these two actions based on the user's preferences. By asking some questions from the user, the application can find what that user prefers and change the reward function. For example, if a user desires to reach the destination faster, then $r_m > r_e$ so that regular actions will be taken.