Министерство науки и высшего образования РоссийскойФедерации

Федеральное государственное бюджетное образовательноеучреждение

высшего образования

«Московский государственный технический университетимени Н.Э. Баумана

(национальный исследовательский университет)»(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика, искусственный интеллект и системы управления

КАФЕДРА Системы обработки информации и управления

Лабораторная работа №5

«Обучение на основе временны’х различий»

Студент группы ИУ5-23М

Кучин Е.А.

Москва, 2023 г.

Цель работы

Ознакомление с базовыми методами обучения с подкреплением на основе временных различий.

Задание

На основе рассмотренного на лекции примера реализуйте следующие алгоритмы:

- SARSA
- Q-обучение
- Двойное Q-обучение

для любой среды обучения с подкреплением (кроме рассмотренной на лекции среды Toy Text / Frozen Lake) из библиотеки Gym (или аналогичной библиотеки).

Выполнение

Реализуем алгоритм Policy Iteration для среды Toy Text / CliffWalking-v0.

Код:

```
import numpy as np
import matplotlib.pyplot as plt
import gym
from tqdm import tqdm


# ************************************ БАЗОВЫЙ АГЕНТ
****************************************


class BasicAgent:
    '''
    Базовый агент, от которого наследуются стратегии
обучения
    '''
```

```
    # Наименование алгоритма
    ALGO_NAME = '---'

    def __init__(self, env, eps=0.1):
        # Среда
        self.env = env
        # Размерности Q-матрицы
        self.nA = env.action_space.n
        self.nS = env.observation_space.n
        # и сама матрица
        self.Q = np.zeros((self.nS, self.nA))
        # Значения коэффициентов
        # Порог выбора случайного действия
        self.eps = eps
        # Награды по эпизодам
        self.episodes_reward = []

    def print_q(self):
        print('Вывод Q-матрицы для алгоритма ',
self.ALGO_NAME)
        print(self.Q)

    def get_state(self, state):
        '''
        Возвращает правильное начальное состояние
        '''
        if type(state) is tuple:
            # Если состояние вернулось с виде кортежа,
то вернуть только номер состояния
            return state[0]
        else:
            return state

    def greedy(self, state):
        '''
```

```
        <<Жадное>> текущее действие
        Возвращает действие, соответствующее
максимальному Q-значению
        для состояния state
        '''
        return np.argmax(self.Q[state])

    def make_action(self, state):
        '''
        Выбор действия агентом
        '''
        if np.random.uniform(0, 1) < self.eps:

            # Если вероятность меньше eps
            # то выбирается случайное действие
            return self.env.action_space.sample()
        else:
            # иначе действие, соответствующее
максимальному Q-значению
            return self.greedy(state)

    def draw_episodes_reward(self):
        # Построение графика наград по эпизодам
        fig, ax = plt.subplots(figsize=(15, 10))
        y = self.episodes_reward
        x = list(range(1, len(y) + 1))
        plt.plot(x, y, '-', linewidth=1, color='green')
        plt.title('Награды по эпизодам')
        plt.xlabel('Номер эпизода')
        plt.ylabel('Награда')
        plt.show()

    def learn(self):
        '''
        Реализация алгоритма обучения
```

```
        '''
        pass


# ************************************ SARSA
****************************************

class SARSA_Agent(BasicAgent):
    '''
    Реализация алгоритма SARSA
    '''
    # Наименование алгоритма
    ALGO_NAME = 'SARSA'

    def __init__(self, env, eps=0.4, lr=0.1, gamma=0.98,
num_episodes=20000):
        # Вызов конструктора верхнего уровня
        super().__init__(env, eps)
        # Learning rate
        self.lr = lr
        # Коэффициент дисконтирования
        self.gamma = gamma
        # Количество эпизодов
        self.num_episodes = num_episodes
        # Постепенное уменьшение eps
        self.eps_decay = 0.00005
        self.eps_threshold = 0.01

    def learn(self):
        '''
        Обучение на основе алгоритма SARSA
        '''
        self.episodes_reward = []
        # Цикл по эпизодам
        for ep in tqdm(list(range(self.num_episodes))):
            # Начальное состояние среды
```

```
            state = self.get_state(self.env.reset())
            # Флаг штатного завершения эпизода
            done = False
            # Флаг нештатного завершения эпизода
            truncated = False
            # Суммарная награда по эпизоду
            tot_rew = 0

            # По мере заполнения Q-матрицы уменьшаем
вероятность случайного выбора действия
            if self.eps > self.eps_threshold:
                self.eps -= self.eps_decay

            # Выбор действия
            action = self.make_action(state)

            # Проигрывание одного эпизода до финального
состояния
            while not (done or truncated):

                # Выполняем шаг в среде
                next_state, rew, done, truncated, _ =
self.env.step(action)

                # Выполняем следующее действие
                next_action =
self.make_action(next_state)

                # Правило обновления Q для SARSA
                self.Q[state][action] =
self.Q[state][action] + self.lr * \
                                        (rew +
self.gamma * self.Q[next_state][next_action] -
self.Q[state][action])
```

```
                # Следующее состояние считаем текущим
                state = next_state
                action = next_action
                # Суммарная награда за эпизод
                tot_rew += rew
                if (done or truncated):
                    self.episodes_reward.append(tot_rew)


# ********************************** Q-обучение
*****************************************

class QLearning_Agent(BasicAgent):
    '''
    Реализация алгоритма Q-Learning
    '''
    # Наименование алгоритма
    ALGO_NAME = 'Q-обучение'

    def __init__(self, env, eps=0.4, lr=0.1, gamma=0.98,
num_episodes=20000):
        # Вызов конструктора верхнего уровня
        super().__init__(env, eps)
        # Learning rate
        self.lr = lr
        # Коэффициент дисконтирования
        self.gamma = gamma
        # Количество эпизодов
        self.num_episodes = num_episodes
        # Постепенное уменьшение eps
        self.eps_decay = 0.00005
        self.eps_threshold = 0.01

    def learn(self):
        '''
        Обучение на основе алгоритма Q-Learning
```

```
        '''
        self.episodes_reward = []
        # Цикл по эпизодам
        for ep in tqdm(list(range(self.num_episodes))):
            # Начальное состояние среды
            state = self.get_state(self.env.reset())
            # Флаг штатного завершения эпизода
            done = False
            # Флаг нештатного завершения эпизода
            truncated = False
            # Суммарная награда по эпизоду
            tot_rew = 0

            # По мере заполнения Q-матрицы уменьшаем
вероятность случайного выбора действия
            if self.eps > self.eps_threshold:
                self.eps -= self.eps_decay

            # Проигрывание одного эпизода до финального
состояния
            while not (done or truncated):

                # Выбор действия
                # В SARSA следующее действие выбиралось
после шага в среде
                action = self.make_action(state)

                # Выполняем шаг в среде
                next_state, rew, done, truncated, _ =
self.env.step(action)

                # Правило обновления Q для SARSA (для
сравнения)
                # self.Q[state][action] =
self.Q[state][action] + self.lr * \
```

```
                #     (rew + self.gamma *
self.Q[next_state][next_action] - self.Q[state][action])

                # Правило обновления для Q-обучения
                self.Q[state][action] =
self.Q[state][action] + self.lr * \
                                        (rew +
self.gamma * np.max(self.Q[next_state]) -
self.Q[state][action])

                # Следующее состояние считаем текущим
                state = next_state
                # Суммарная награда за эпизод
                tot_rew += rew
                if (done or truncated):
                    self.episodes_reward.append(tot_rew)


# ********************************** Двойное Q-
обучение ****************************************

class DoubleQLearning_Agent(BasicAgent):
    '''
    Реализация алгоритма Double Q-Learning
    '''
    # Наименование алгоритма
    ALGO_NAME = 'Двойное Q-обучение'

    def __init__(self, env, eps=0.4, lr=0.1, gamma=0.98,
num_episodes=20000):
        # Вызов конструктора верхнего уровня
        super().__init__(env, eps)
        # Вторая матрица
        self.Q2 = np.zeros((self.nS, self.nA))
        # Learning rate
        self.lr = lr
```

```
        # Коэффициент дисконтирования
        self.gamma = gamma
        # Количество эпизодов
        self.num_episodes = num_episodes
        # Постепенное уменьшение eps
        self.eps_decay = 0.00005
        self.eps_threshold = 0.01

    def greedy(self, state):
        '''
        <<Жадное>> текущее действие
        Возвращает действие, соответствующее
максимальному Q-значению
        для состояния state
        '''
        temp_q = self.Q[state] + self.Q2[state]
        return np.argmax(temp_q)

    def print_q(self):
        print('Вывод Q-матриц для алгоритма ',
self.ALGO_NAME)
        print('Q1')
        print(self.Q)
        print('Q2')
        print(self.Q2)

    def learn(self):
        '''
        Обучение на основе алгоритма Double Q-Learning
        '''
        self.episodes_reward = []
        # Цикл по эпизодам
        for ep in tqdm(list(range(self.num_episodes))):
            # Начальное состояние среды
            state = self.get_state(self.env.reset())
```

```
            # Флаг штатного завершения эпизода
            done = False
            # Флаг нештатного завершения эпизода
            truncated = False
            # Суммарная награда по эпизоду
            tot_rew = 0

            # По мере заполнения Q-матрицы уменьшаем
вероятность случайного выбора действия
            if self.eps > self.eps_threshold:
                self.eps -= self.eps_decay

            # Проигрывание одного эпизода до финального
состояния
            while not (done or truncated):

                # Выбор действия
                # В SARSA следующее действие выбиралось
после шага в среде
                action = self.make_action(state)

                # Выполняем шаг в среде
                next_state, rew, done, truncated, _ =
self.env.step(action)

                if np.random.rand() < 0.5:
                    # Обновление первой таблицы
                    self.Q[state][action] =
self.Q[state][action] + self.lr * \
                                            (rew +
self.gamma *
self.Q2[next_state][np.argmax(self.Q[next_state])] -
                                             self.Q[stat
e][action])
                else:
```

```
                    # Обновление второй таблицы
                    self.Q2[state][action] =
self.Q2[state][action] + self.lr * \
                                                (rew +
self.gamma *
self.Q[next_state][np.argmax(self.Q2[next_state])] -
                                                 self.Q2[st
ate][action])

                # Следующее состояние считаем текущим
                state = next_state
                # Суммарная награда за эпизод
                tot_rew += rew
                if (done or truncated):
                    self.episodes_reward.append(tot_rew)


def play_agent(agent):
    '''
    Проигрывание сессии для обученного агента
    '''
    env2 = gym.make('CliffWalking-v0',
render_mode='human')
    state = env2.reset()[0]
    done = False
    while not done:
        action = agent.greedy(state)
        next_state, reward, terminated, truncated, _ =
env2.step(action)
        env2.render()
        state = next_state
        if terminated or truncated:
            done = True


def run_sarsa():
```

```
    env = gym.make('CliffWalking-v0')
    agent = SARSA_Agent(env)
    agent.learn()
    agent.print_q()
    agent.draw_episodes_reward()
    play_agent(agent)


def run_q_learning():
    env = gym.make('CliffWalking-v0')
    agent = QLearning_Agent(env)
    agent.learn()
    agent.print_q()
    agent.draw_episodes_reward()
    play_agent(agent)


def run_double_q_learning():
    env = gym.make('CliffWalking-v0')
    agent = DoubleQLearning_Agent(env)
    agent.learn()
    agent.print_q()
    agent.draw_episodes_reward()
    play_agent(agent)


def main():
    run_sarsa()
    #run_q_learning()
    #run_double_q_learning()


if __name__ == '__main__':
    main()
```

Результат работы программы для алгоритма SARSA:

```
Вывод Q-матрицы для алгоритма  SARSA
[[ -13.21015393  -12.42996941  -14.29613805  -13.24574563]
 [ -12.48749754  -11.66639835  -13.42073519  -13.40697399]
 [ -11.69587112  -10.89505749  -12.57737613  -12.61879238]
 [ -10.88311414  -10.11227071  -11.72924207  -11.8352084 ]
 [ -10.0419899    -9.30820907  -11.0054496   -11.11029379]
 [  -9.25989339   -8.54590809  -10.17318446  -10.28687395]
 [  -8.40886875   -7.57956895   -9.31565224   -9.49054147]
 [  -7.56078203   -6.68413961   -8.40163671   -8.59973287]
 [  -6.66559792   -5.76683922   -6.28501812   -7.69775871]
 [  -5.75696866   -4.83729482   -5.2238815    -6.86249333]
 [  -4.85677362   -3.89454678   -4.11151426   -5.99704317]
 [  -3.91317819   -3.91242427   -2.94337292   -5.06793162]
 [ -13.36491905  -13.82214835  -14.96107432  -13.93559129]
 [ -12.42894123  -15.62397886  -20.5239411   -16.12683897]
 [ -11.6908463   -16.05227282  -30.04550455  -17.98859378]
 [ -10.88239699  -15.78240856  -20.51226137  -16.58667141]
 [ -10.05864754  -14.54346638  -18.22055197  -16.42490816]
 [  -9.25048011  -13.00032823  -24.10346982  -14.78789313]
 [  -8.40076148  -11.12026322  -16.85540569  -17.24159146]
 [  -7.53874372   -9.2911096   -14.91678344  -13.68765708]
 [  -8.48475534   -5.26445821  -13.43412488  -11.18970381]
 [  -7.20246703   -4.95648496  -13.71546024  -10.16971641]
 [  -5.64987908   -2.950992    -17.24856808   -6.82281677]
 [  -4.10683931   -2.96312386   -1.98030212   -3.93327622]
 [ -13.98743904  -14.59224088  -15.60650741  -14.68899878]
 [ -13.37198818  -28.17511889 -124.06346486  -21.24484419]
 [ -17.41992889  -20.58117552 -110.65649897  -34.77547164]
 [ -15.69849349  -23.00234341 -114.51921178  -27.82403547]
 [ -15.13898009  -17.01952102 -108.10380725  -17.30157339]
 [ -15.45391474  -18.50195183  -92.56229261  -23.96747221]
 [ -13.60649575  -22.5595971  -113.13314213  -18.99061304]
 [ -12.67095676  -20.12032339 -105.99206364  -17.04193753]
 [ -10.37764034  -11.65327985 -108.26544233  -17.6272184 ]
 [  -6.80516723  -19.88266747 -112.95616525  -19.98106297]
 [  -6.67918154   -2.08833782 -124.13140443  -13.83511482]
 [  -3.16494695   -1.98225499   -1.           -3.1106409 ]
 [ -14.68094891 -114.78397415  -15.4109003   -15.46183668]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]]
```

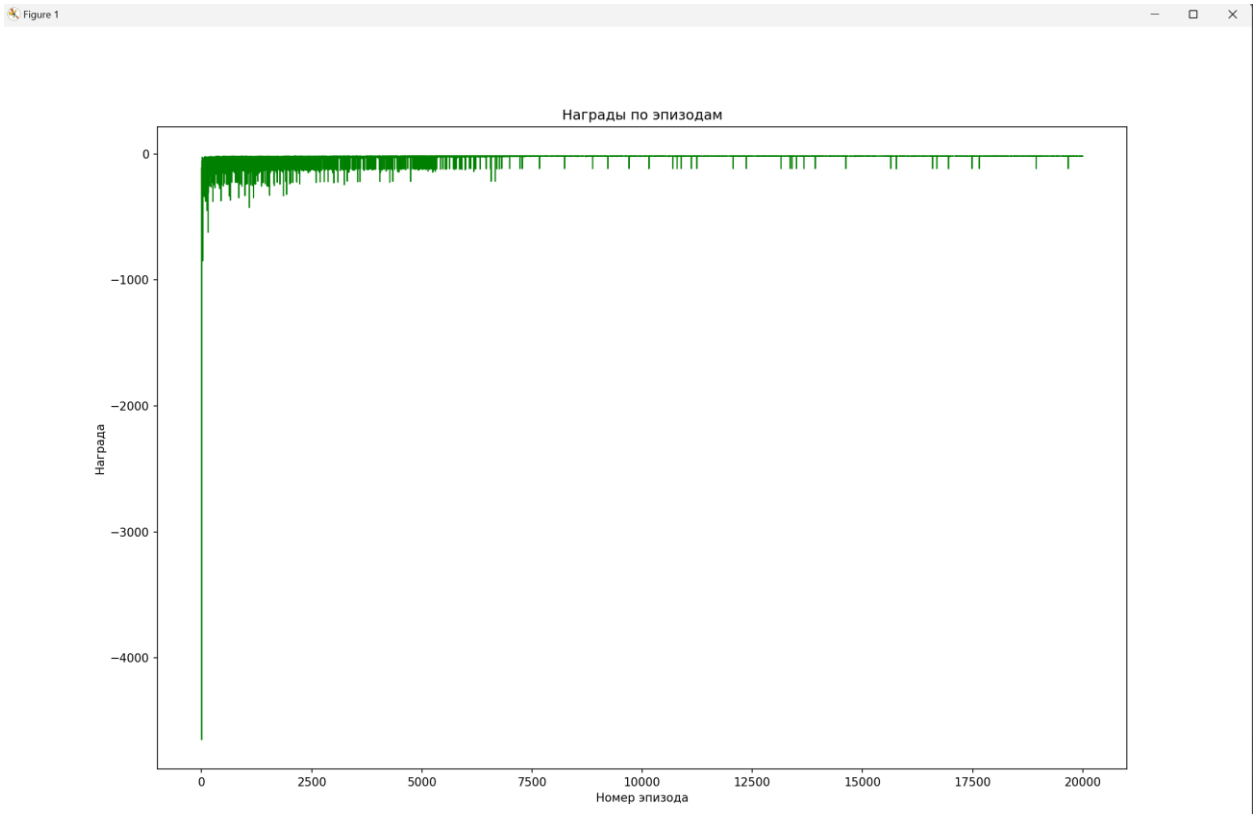
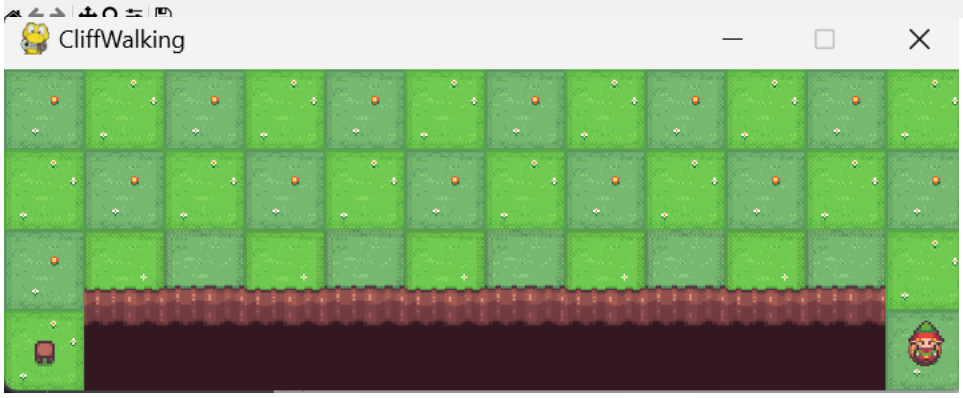


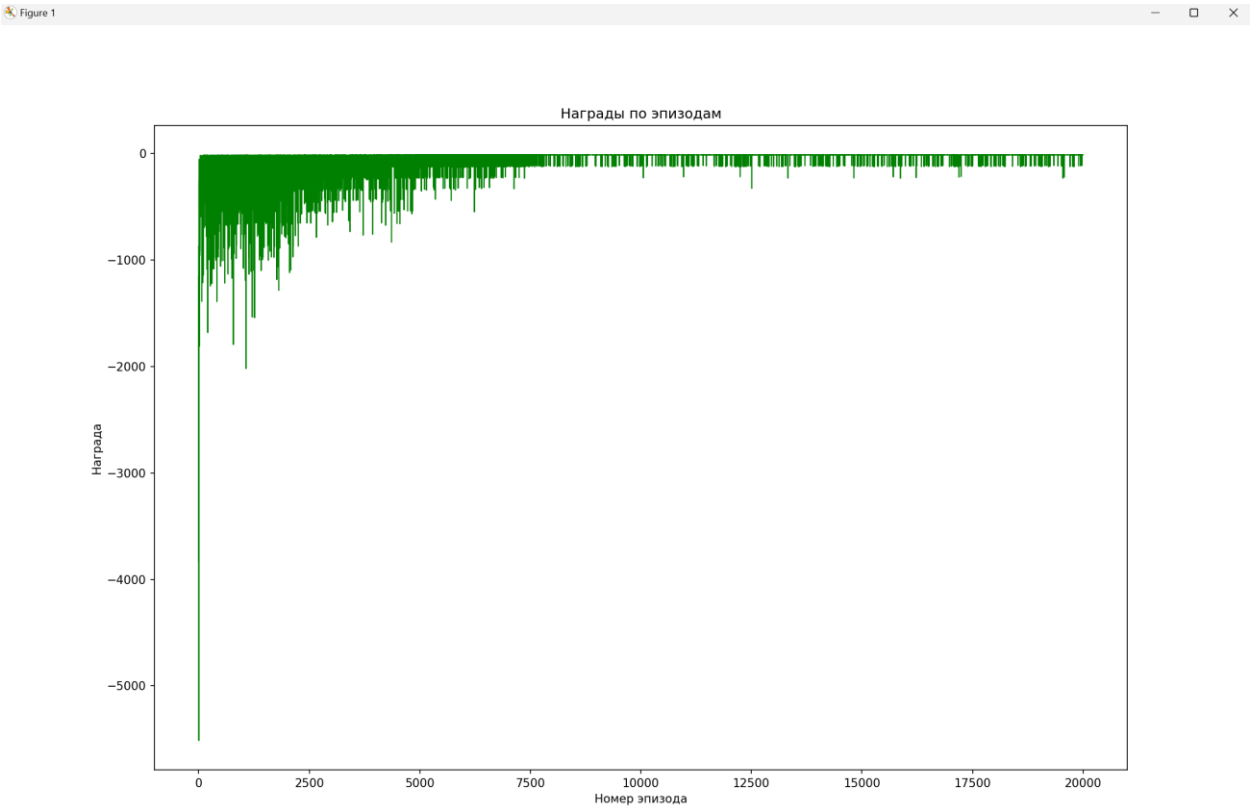
Figure 1
Награды по эпизодам
0
−1000
−2000
−3000
−4000
Награда
0
2500
5000
7500
10000
12500
15000
17500
20000
Номер эпизода

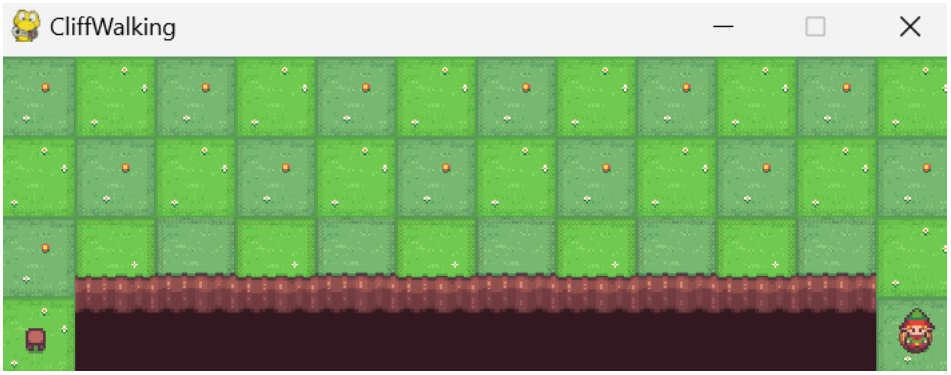

CliffWalking

Результат работы программы для алгоритма Q-обучение:

```
[[ -12.39040286  -12.29707966  -12.29595193  -12.41197684]
 [ -11.8773398   -11.54815013  -11.54809451  -12.32948971]
 [ -11.47097796  -10.76412481  -10.76412849  -12.16752484]
 [ -10.69827652   -9.96342835   -9.96342804  -11.30099155]
 [  -9.92683095   -9.14635924   -9.14635924  -10.69710935]
 [  -9.11229633   -8.31261181   -8.31261181   -9.94743834]
 [  -8.30698013   -7.46184886   -7.46184886   -9.13200698]
 [  -7.44902392   -6.59372333   -6.59372333   -8.27439168]
 [  -6.59130676   -5.70788096   -5.70788096   -7.44057164]
 [  -5.69320462   -4.80396016   -4.80396016   -6.58044772]
 [  -4.74769989   -3.881592     -3.881592     -5.68125047]
 [  -3.87614256   -3.8642663    -2.9404       -4.76388909]
 [ -13.02802     -11.54888054  -11.54888054  -12.31765028]
 [ -12.31522008  -10.76416381  -10.76416381  -12.31789818]
 [ -11.54875765   -9.96343246   -9.96343246  -11.54888046]
 [ -10.76414179   -9.14635966   -9.14635966  -10.76416377]
 [  -9.96343087   -8.31261189   -8.31261189   -9.96343245]
 [  -9.14635892   -7.46184887   -7.46184887   -9.14635965]
 [  -8.3126118    -6.59372334   -6.59372334   -8.31261189]
 [  -7.4618488    -5.70788096   -5.70788096   -7.46184886]
 [  -6.59372333   -4.80396016   -4.80396016   -6.59372331]
 [  -5.70788092   -3.881592     -3.881592     -5.70788093]
 [  -4.80396015   -2.9404       -2.9404       -4.80396015]
 [  -3.881592     -2.9404       -1.98         -3.881592  ]
 [ -12.31790293  -10.76416381  -12.31790293  -11.54888054]
 [ -11.54888054   -9.96343246 -111.31790293  -11.54888054]
 [ -10.76416381   -9.14635966 -111.31790293  -10.76416381]
 [  -9.96343246   -8.31261189 -111.31790293   -9.96343246]
 [  -9.14635966   -7.46184887 -111.31790293   -9.14635966]
 [  -8.31261189   -6.59372334 -111.31790293   -8.31261189]
 [  -7.46184887   -5.70788096 -111.31790293   -7.46184887]
 [  -6.59372334   -4.80396016 -111.31790293   -6.59372334]
 [  -5.70788096   -3.881592   -111.31790293   -5.70788096]
 [  -4.80396016   -2.9404     -111.31790293   -4.80396016]
 [  -3.881592     -1.98       -111.31790293   -3.881592  ]
```

```
[  -2.9404       -1.98         -1.           -2.9404    ]
[ -11.54888054 -111.31790293  -12.31790293  -12.31790293]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]]
```




CliffWalking

Результат работы программы для алгоритма двойное Q-обучение:

```
Вывод Q-матриц для алгоритма  Двойное Q-обучение
Q1
[[ -31.30239245  -32.78573759  -12.35500081  -32.29949003]
 [ -30.44992982  -23.4106236   -11.5556959   -25.00168031]
 [ -22.79817579  -28.29203216  -11.00737396  -27.64824564]
 [ -24.25554908  -10.10026098  -15.82347474  -20.41712112]
 [ -18.50653389   -9.15347292  -15.01359927  -20.73031304]
 [ -12.49020543   -8.31269022  -11.50301622  -16.660006  ]
 [ -10.85105875  -13.48027072   -7.46184911  -13.61604636]
 [ -18.52512188  -22.19278191   -6.70817639  -16.78416999]
 [ -18.789363     -6.24827683  -14.75298559  -17.79091852]
 [ -10.20231857   -4.97712694  -15.07025526  -15.35438463]
 [  -7.209865     -4.62463389   -3.8815927    -8.69575185]
 [  -4.88373469   -5.56844509   -2.94182622   -7.04418965]
 [ -13.97497275  -11.54888054  -11.55888215  -12.62502239]
 [ -12.33777678  -10.77935272  -10.76416381  -12.32620431]
 [ -11.88903093   -9.96343246  -10.21971625  -11.59614557]
 [ -11.41208797   -9.15937848   -9.14635966  -10.77604421]
 [ -11.15673676   -8.31261189   -8.52837283  -10.89006617]
 [  -9.28834393   -7.46184887   -7.66507269   -9.24189047]
 [  -8.32171158   -6.59557023   -6.59372334   -8.3328309 ]
 [  -9.40682605   -6.96996154   -5.70788096   -7.68570575]
 [ -13.9177877    -5.37173809   -5.86801328   -8.93239961]
 [  -5.94856238   -3.881592     -3.86199857   -5.95351265]
 [  -5.14991675   -2.9404       -2.95484618   -6.54286376]
 [  -3.88712193   -2.94078113   -1.98         -3.8811746 ]
 [ -12.31790293  -10.76416381  -12.31790293  -11.54888054]
 [ -11.54888054   -9.96343246 -111.31790293  -11.54888054]
 [ -10.76416381   -9.14635966 -111.31790293  -10.76416381]
 [  -9.96343246   -8.31261189 -111.31790293   -9.96343246]
 [  -9.14635966   -7.46184887 -111.31790293   -9.14635966]
 [  -8.31261189   -6.59372334 -111.31790293   -8.31261189]
 [  -7.46184887   -5.70788096 -111.31790293   -7.46184887]
 [  -6.59372334   -4.80396016 -111.31790293   -6.59372334]
 [  -5.70788111   -3.881592   -111.31790293   -5.70788096]
```

```
 [  -5.36942648   -2.9404     -111.31790293   -4.80396016]
 [  -3.881592     -1.98       -111.31790292   -3.881592  ]
 [  -2.9404       -1.98         -1.           -2.9404    ]
 [ -11.54888054 -111.31790293  -12.31790293  -12.31790293]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]
 [   0.            0.            0.            0.        ]]
Q2
[[ -27.95999372  -27.65522049  -12.43118643  -34.12091714]
 [ -22.70521736  -30.51486467  -11.55067282  -31.69851536]
 [ -25.02207967  -25.63618176  -10.79829408  -20.3871938 ]
 [ -18.92953685  -10.25796951  -20.67466875  -27.72676873]
 [ -20.48767408   -9.15631827  -14.96758357  -19.00979969]
 [ -13.54411419   -8.31266702  -11.9656263   -13.2958664 ]
 [ -11.17451455  -11.61018662   -7.46184893  -12.33040904]
 [ -15.15761976  -19.21849256   -6.86591433  -19.19134475]
 [ -17.32274286   -8.38988048  -20.36604338  -14.93061311]
 [ -13.26012052   -4.81213728   -5.17641022  -16.51875629]
 [  -6.31997876   -4.91501992   -3.90889175  -12.87767264]
 [  -5.33209214   -5.83233982   -2.94092812   -5.92178926]
 [ -13.40752705  -11.54888054  -11.61738802  -12.38571099]
 [ -12.37367744  -10.76938292  -10.76416381  -12.31859505]
 [ -13.33359774   -9.96343246  -10.18938012  -11.68346535]
 [ -12.02677672   -9.19382348   -9.14635966  -10.79240262]
 [ -11.77905376   -8.31261189   -9.16826357  -10.97717942]
 [  -9.22294796   -7.46184887   -7.49308202   -9.17915539]
 [  -8.32386027   -6.59623927   -6.59372334   -8.32800159]
 [  -8.84438772   -9.27048179   -5.70788096   -7.84560688]
```

```
[ -11.49465463   -4.80396016   -5.42172141   -8.85363475]
[  -8.12892592   -3.89576725   -4.45130001   -8.22197768]
[  -4.8227216    -2.9404       -2.93945816   -4.80328943]
[  -3.89927613   -2.94146745   -1.98         -3.96889878]
[ -12.31790293  -10.76416381  -12.31790293  -11.54888054]
[ -11.54888054   -9.96343246 -111.31790293  -11.54888054]
[ -10.76416381   -9.14635966 -111.31790293  -10.76416381]
[  -9.96343246   -8.31261189 -111.31790293   -9.96343246]
[  -9.14635966   -7.46184887 -111.31790293   -9.14635966]
[  -8.31261189   -6.59372334 -111.31790293   -8.31261189]
[  -7.46184887   -5.70788096 -111.31790293   -7.46184887]
[  -6.59372334   -4.80396016 -111.31790293   -6.59372334]
[  -6.27862955   -3.881592   -111.31790293   -5.70788096]
[  -4.80396016   -2.9404     -111.31790293   -4.80396016]
[  -3.89632284   -1.98       -111.31790293   -3.881592  ]
[  -2.9404       -1.98         -1.           -2.9404    ]
[ -11.54888054 -111.31790293  -12.31790293  -12.31790293]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]]
```

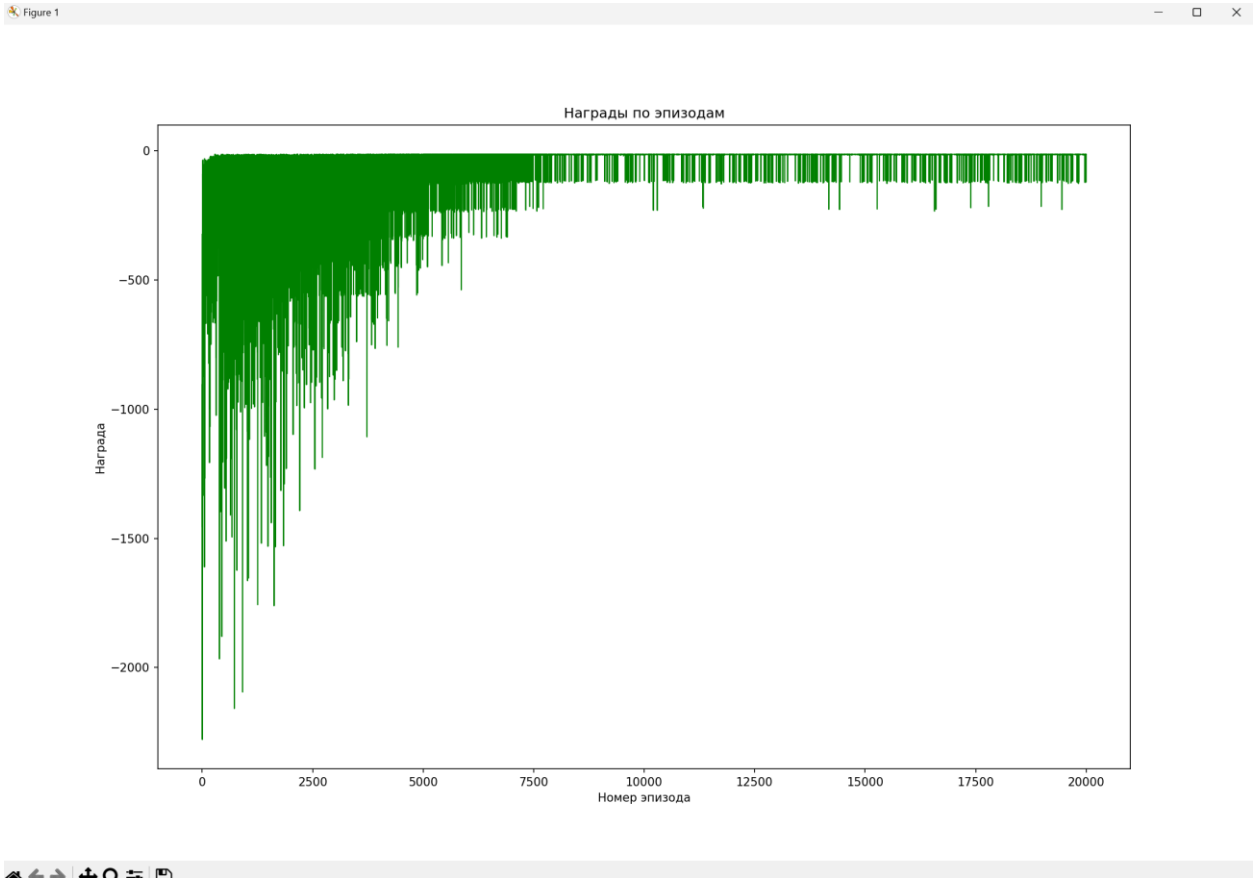
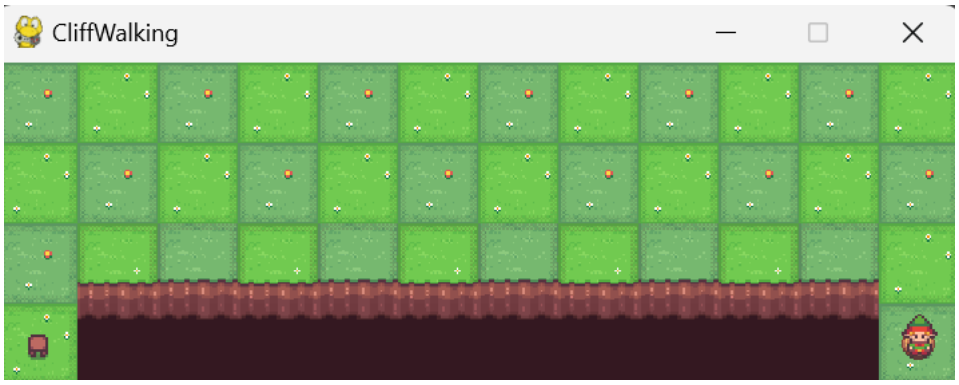


Figure 1
Награды по эпизодам
Награда
0
−500
−1000
−1500
−2000
0
2500
5000
7500
10000
12500
15000
17500
20000
Номер эпизода


CliffWalking