Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления и искусственный интеллект

КАФЕДРА Системы обработки информации и управления

Лабораторная работа № 5

По курсу
«Методы машинного обучения»
На тему:
«Обучение на основе временны’х различий»

Подготовил:

Студент группы

ИУ5-25М Клюкин Н. А.

27.03.2024

Проверил:

Гапанюк Ю.Е.

2024 г.

- **Цель лабораторной 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

Подключение библиотек

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import gym
from tqdm import tqdm
```

In [4]:
```
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 сост
            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():
        '''
        Реализация алгоритма обучения
        '''
        pass
```

Ход работы

SARSA

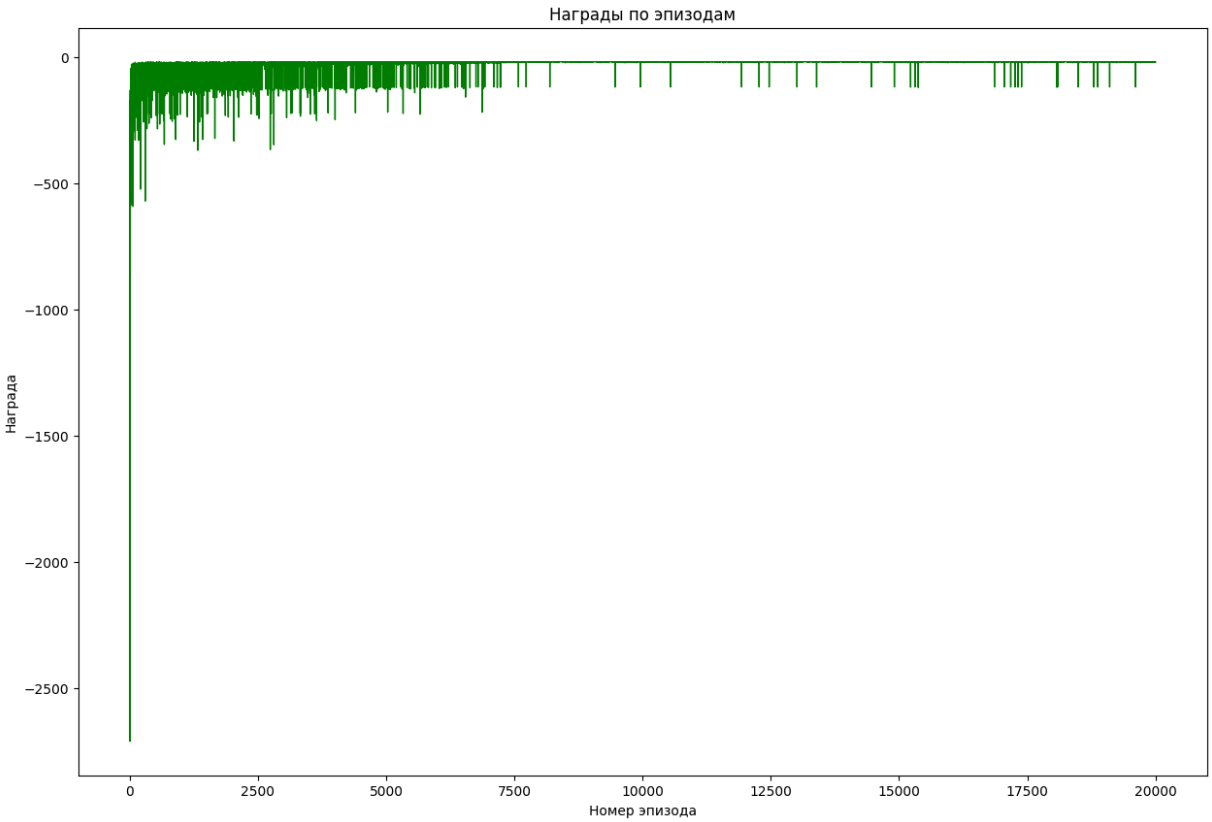
In [6]:
```
# ********************************** SARSA ***************************************

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
            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[st

                # 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)




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 main():
    run_sarsa()


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

```
100%|██████████| 20000/20000 [00:05<00:00, 3352.77it/s]
```

```
Вывод Q-матрицы для алгоритма  SARSA
[[ -13.26470653  -12.50088708  -14.12708179  -13.2602266 ]
 [ -12.44810083  -11.66480746  -13.22626654  -13.37544225]
 [ -11.6741787   -10.83985372  -12.58114044  -12.6421409 ]
 [ -10.88449284  -10.00515452  -11.74521468  -11.82916784]
 [ -10.07614536   -9.16500291  -11.07197463  -11.06165504]
 [  -9.29169      -8.31889331  -10.29875341  -10.28629389]
 [  -8.42245999   -7.46371483   -9.54298258   -9.41219891]
 [  -7.53198068   -6.59559956   -8.49446658   -8.57624353]
 [  -6.64322634   -5.7141876    -7.57035543   -7.73828363]
 [  -5.79709371   -4.82459413   -5.49548661   -6.84024735]
 [  -4.85539281   -3.92891243   -4.11349098   -5.96644982]
 [  -3.93925862   -3.93892335   -3.00725422   -5.05824473]
 [ -13.29836889  -13.35975581  -14.88402218  -13.98048052]
 [ -12.44108167  -16.57064714  -30.53772836  -17.14037655]
 [ -11.65896373  -16.88879954  -28.44540438  -18.92766783]
 [ -10.87488589  -15.42479893  -22.25254099  -17.57347329]
 [ -10.07842007  -14.58037328  -21.21584087  -16.53332701]
 [  -9.36335467  -13.53868895  -17.82947405  -16.5246988 ]
 [  -8.42222896  -11.49177578  -18.29985297  -14.77134129]
 [  -7.55118889  -10.87637871  -15.4390487   -13.35611667]
 [  -6.63602141   -8.99357359  -14.84165671  -12.12995581]
 [  -7.81828798   -3.88286671  -10.87108182  -10.56736033]
 [  -5.65519628   -2.94072075   -4.10997813   -5.49335182]
 [  -4.07645936   -2.99643981   -2.02385475   -4.01082675]
 [ -14.03577859  -14.45222474  -15.96825662  -14.71374521]
 [ -13.24817728  -24.02802056 -126.15707484  -21.23481088]
 [ -17.90342998  -23.38484894 -114.95476683  -21.12832926]
 [ -16.2500107   -20.17532624 -113.52527234  -24.61424562]
 [ -16.03235068  -22.27359304 -101.46759995  -20.97163167]
 [ -13.66790198  -16.758422   -109.89934135  -15.61642488]
 [ -14.03299862  -17.80043385 -104.55476806  -15.96723849]
 [ -12.15135778  -21.72149916 -102.43878562  -14.64334574]
 [ -10.6230545   -15.52372518 -101.55628835  -18.19900775]
 [  -7.39964116  -13.66643919  -93.62787353  -22.16091851]
 [  -6.14044219   -2.00982344 -125.50678742  -10.68887623]
 [  -3.17103369   -1.986425     -1.           -3.0454338 ]
 [ -14.74106658 -114.77992745  -15.54372228  -15.86679908]
 [   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.        ]]
```



Q-обучение

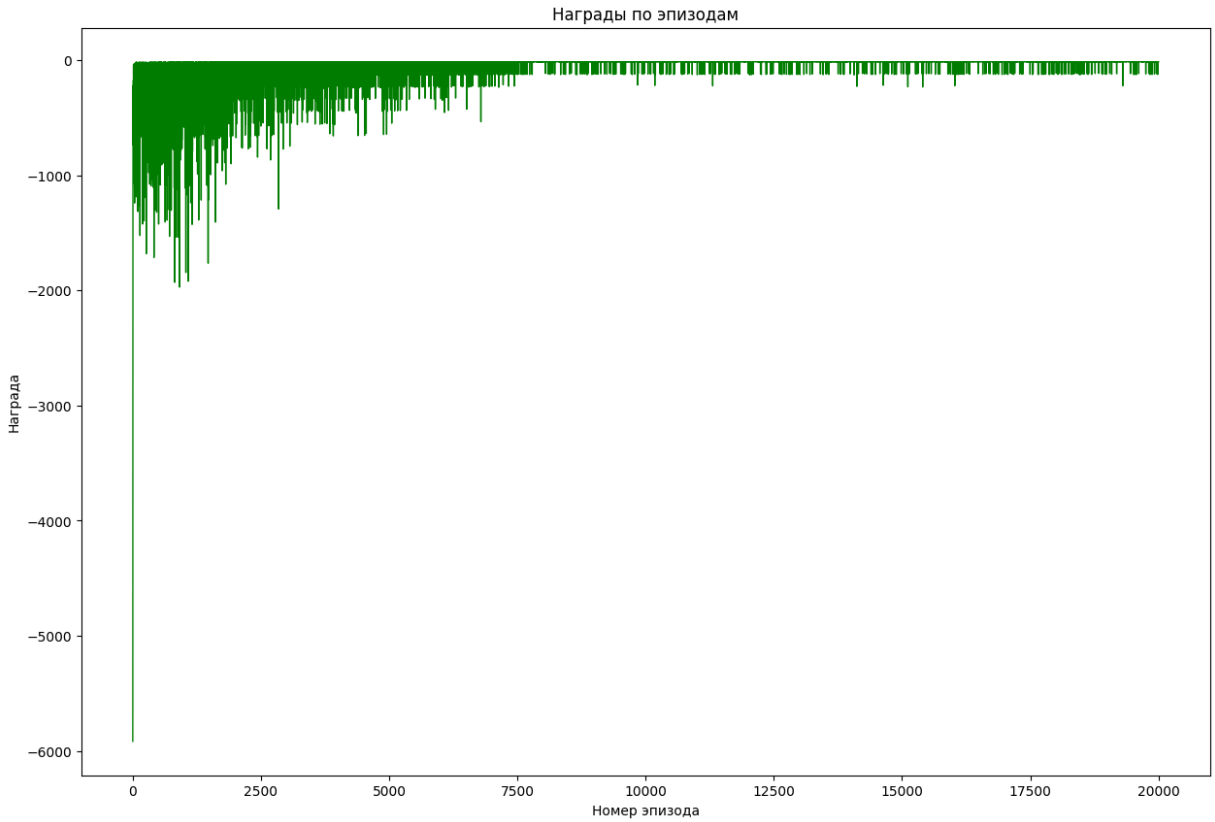
In [5]:
```
# ********************************** Q-обучение **********************************
class QLearning_Agent(BasicAgent):
    '''
    Реализация алгоритма Q-Learning
    '''
    # Наименование алгоритма
    ALGO_NAME = 'Q-обучение'


    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
            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[

                # 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][

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



def main():
    run_q_learning()


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

```
100%|██████████| 20000/20000 [00:06<00:00, 2974.67it/s]
```

```
Вывод Q-матрицы для алгоритма  Q-обучение
[[ -12.6380218   -12.3013737   -12.30091891  -12.46266369]
 [ -12.15963951  -11.54824073  -11.54830237  -12.508789  ]
 [ -11.23082209  -10.76413455  -10.76413705  -11.95132332]
 [ -10.72813239   -9.96342902   -9.96342911  -11.39771031]
 [  -9.90958869   -9.14635926   -9.14635924  -10.67547904]
 [  -9.0718052    -8.31261181   -8.3126118    -9.91521179]
 [  -8.29003744   -7.46184884   -7.46184884   -9.0861385 ]
 [  -7.45246347   -6.59372333   -6.59372333   -8.30107967]
 [  -6.58077485   -5.70788096   -5.70788096   -7.45038973]
 [  -5.69866972   -4.80396016   -4.80396016   -6.56206694]
 [  -4.78195251   -3.881592     -3.881592     -5.6975148 ]
 [  -3.86749983   -3.86932428   -2.9404       -4.78368401]
 [ -13.03262668  -11.54888054  -11.54888054  -12.31780693]
 [ -12.31644188  -10.76416381  -10.76416381  -12.3178932 ]
 [ -11.54879626   -9.96343246   -9.96343246  -11.54887927]
 [ -10.76415213   -9.14635966   -9.14635966  -10.7641638 ]
 [  -9.96343102   -8.31261189   -8.31261189   -9.96343245]
 [  -9.14635899   -7.46184887   -7.46184887   -9.14635966]
 [  -8.3126114    -6.59372334   -6.59372334   -8.31261189]
 [  -7.46184879   -5.70788096   -5.70788096   -7.46184887]
 [  -6.59372332   -4.80396016   -4.80396016   -6.59372333]
 [  -5.70788093   -3.881592     -3.881592     -5.70788095]
 [  -4.80396012   -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.        ]]
```



Двойное Q-обучение

In [5]:
```
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
```

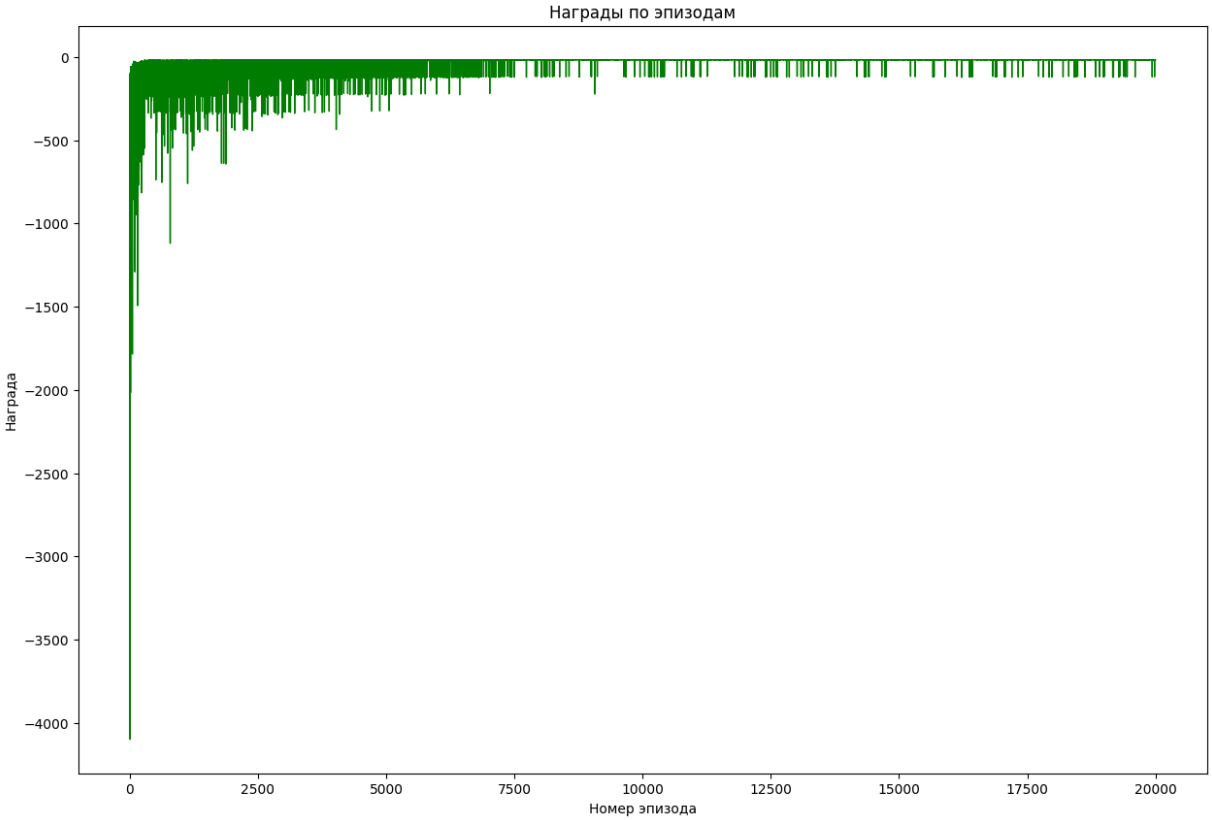
```
        Возвращает действие, соответствующее максимальному Q-значению
        для состояния state
        '''
        temp_q = self.Q[state] + self.Q2[state]
        return np.argmax(temp_q)


    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
            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[ne
                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[ne

                # 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_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_double_q_learning()


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

```
  0%|          | 0/20000 [00:00<?, ?it/s]c:\Users\NKliukin\Code\bmstu\mmo_2_2024\ven
v\Lib\site-packages\gym\utils\passive_env_checker.py:233: DeprecationWarning: `np.bo
ol8` is a deprecated alias for `np.bool_`.  (Deprecated NumPy 1.24)
  if not isinstance(terminated, (bool, np.bool8)):
100%|██████████| 20000/20000 [00:07<00:00, 2670.36it/s]
```

```
Вывод Q-матриц для алгоритма  Двойное Q-обучение
Q1
[[ -15.36493702  -12.46666513  -14.85526681  -15.25505368]
 [ -13.82732067  -11.55036323  -13.97898739  -14.62249367]
 [ -11.72163427  -10.76416381  -12.53295404  -12.66857913]
 [ -10.76416381   -9.96343246  -11.54888054  -11.54888054]
 [  -9.96343246   -9.14635966  -10.76416381  -10.76416381]
 [  -9.14635966   -8.31261189   -9.96343246   -9.96343246]
 [  -8.31261189   -7.46184887   -9.47385463   -9.14635966]
 [  -7.46184887   -6.59372334   -6.59844374   -8.31261189]
 [  -6.59372334   -5.70788096   -5.70788102   -7.46184887]
 [  -5.70788096   -4.80396016   -4.80781695   -6.59372334]
 [  -4.80396016   -3.881592     -3.881592     -5.70788096]
 [  -3.881592     -3.881592     -2.9404       -4.80396016]
 [ -13.76098274  -13.07154487  -14.85429231  -13.96948718]
 [ -12.44530401  -12.31790293  -13.89840133  -13.87063031]
 [ -11.55895978  -11.54888054  -13.07674458  -13.07361402]
 [ -10.76416381  -10.76416381  -12.31790293  -12.31790293]
 [  -9.96343246   -9.99386416  -11.58521359  -11.57577745]
 [  -9.14635966   -9.51060824  -11.51539891  -10.86332842]
 [  -8.31261189   -8.29442408  -11.20568102  -10.27254373]
 [  -7.46184978   -5.70790115   -8.04419515   -9.40737507]
 [  -6.88795262   -4.80757994   -6.74806205   -7.822562  ]
 [  -5.74745053   -3.881592     -3.88853009   -5.81678344]
 [  -4.80600758   -2.9404       -2.94416419   -4.85153968]
 [  -3.881592     -2.9404       -1.98         -3.881592  ]
 [ -13.81011397  -13.81011397  -15.24323346  -14.53391169]
 [ -13.07154487  -13.07154487 -114.24323346  -14.53391169]
 [ -12.31790293  -12.31790293 -114.24323346  -13.81011397]
 [ -11.54888054  -11.54888093 -114.24323346  -13.0715449 ]
 [ -10.76416381  -11.81491629 -114.1129963   -12.84580525]
 [ -10.06373445  -13.39143335 -113.43607419  -12.77208877]
 [  -9.51415045  -11.1945929  -110.59653266  -12.64178946]
 [  -9.85348397   -6.85777193 -102.39097712  -11.48549893]
 [  -7.41377153   -3.88855595  -75.64586083   -7.62377718]
 [  -5.70875655   -2.94056795  -92.69088891   -5.61653464]
 [  -3.83448666   -1.98       -113.07328178   -3.84540636]
 [  -2.9404       -1.98         -1.           -2.9404    ]
 [ -14.53391169 -114.24323346  -15.24323346  -15.24323346]
 [   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
[[ -14.95408106  -12.52006851  -15.50393527  -15.54412301]
 [ -13.43673236  -11.5492644   -14.67834838  -14.45424143]
 [ -11.82099416  -10.76416381  -12.37479083  -12.55636365]
 [ -10.76416381   -9.96343246  -11.54888054  -11.54888054]
 [  -9.96343246   -9.14635966  -10.76416381  -10.76416381]
```

```
[  -9.14635966   -8.31261189   -9.96343246   -9.96343246]
[  -8.31261189   -7.46184887   -9.14635966   -9.14635966]
[  -7.46184887   -6.59372334   -6.59388232   -8.31261189]
[  -6.59372334   -5.70788096   -5.71152204   -7.46184887]
[  -5.70788096   -4.80396016   -4.80396016   -6.59372334]
[  -4.80396016   -3.881592     -3.8853476    -5.70788096]
[  -3.881592     -3.881592     -2.9404       -4.80396016]
[ -13.85563909  -13.07154487  -14.72459132  -14.06007819]
[ -12.4128323   -12.31790293  -13.85199746  -13.99282515]
[ -11.55283127  -11.54888054  -13.07896046  -13.10014362]
[ -10.76416381  -10.76416381  -12.31790293  -12.31790293]
[  -9.96343246  -10.03936573  -11.58568402  -11.57054672]
[  -9.14635966   -9.37512778  -11.66223149  -10.86088986]
[  -8.31261189   -8.64707792  -10.93320793  -10.19636172]
[  -7.46185068   -5.71162921   -9.44086479   -9.6457225 ]
[  -6.81481816   -4.80396016   -5.08135111   -7.74556159]
[  -5.71490001   -3.88528105   -3.97898051   -6.10655313]
[  -4.8030787    -2.9404       -2.93887822   -4.81775412]
[  -3.881592     -2.9404       -1.98         -3.88535241]
[ -13.81011397  -13.81011397  -15.24323346  -14.53391169]
[ -13.07154487  -13.07154487 -114.24323346  -14.53391169]
[ -12.31790293  -12.31790293 -114.24323346  -13.81011397]
[ -11.54888054  -11.54888101 -114.24323346  -13.07154487]
[ -10.76416387  -11.96279875 -113.05688561  -12.7807903 ]
[ -10.06511034  -12.64796227 -113.08087991  -13.30689489]
[  -9.6246915   -13.33167689 -106.70980197  -11.62818858]
[ -10.72280119   -5.37241561 -107.51870328  -11.07186718]
[ -10.27542993   -4.06681929  -80.64443715  -11.62771124]
[  -5.09050062   -2.94027     -78.44916248   -6.18240356]
[  -4.15266128   -1.98       -111.28887907   -4.35411716]
[  -2.9404       -1.98         -1.           -2.9404    ]
[ -14.53391169 -114.24323346  -15.24323346  -15.24323346]
[   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.        ]]
```


Награды по эпизодам
Награда
0
−500
−1000
−1500
−2000
−2500
−3000
−3500
−4000
0
2500
5000
7500
10000
12500
15000
17500
20000
Номер эпизода