Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика, искусственный интеллект и системы управления»

Кафедра «Системы обработки информации и управления»

Отчет по лабораторной работе №5

по дисциплине «Методы машинного обучения»

Обучение на основе временных рядов

(тема работы)

ИСПОЛНИТЕЛЬ:

Пасатюк А.Д.

группа ИУ5-23М

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.А.

Москва, 2023

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 программы:

```
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:
```

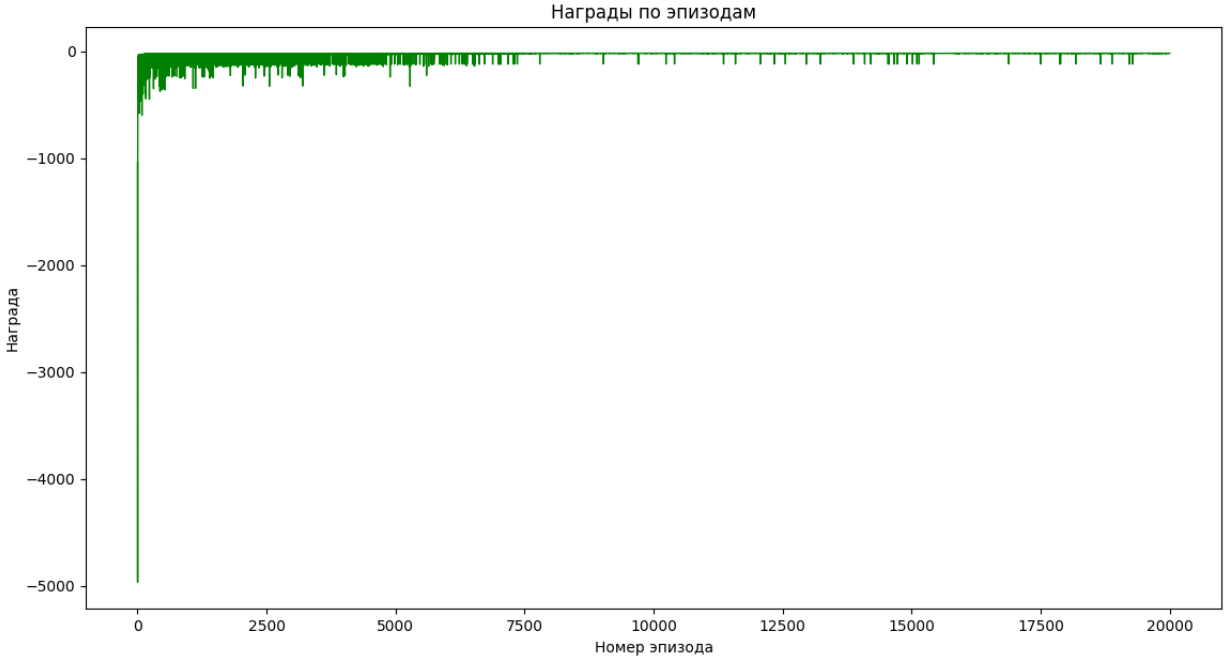
```
                self.eps -= self.eps_decay

            # 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[state][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[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)


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
[[ -13.24280268  -12.33827493  -14.14500067  -13.22801022]
 [ -12.53573138  -11.56746437  -13.53437896  -13.41043807]
 [ -11.71722625  -10.80098742  -12.57949968  -12.68506624]
 [ -10.87506982  -10.0226327   -11.83674168  -11.84720483]
 [ -10.06324214   -9.15263012  -11.14644176  -11.24622758]
 [  -9.2638957    -8.3214858   -10.13247619  -10.31609026]
 [  -8.41184622   -7.47493946   -9.38277949   -9.43477108]
 [  -7.52700726   -6.61432264   -8.47462063   -8.56898341]
 [  -6.68749719   -5.75190573   -5.97075948   -7.74311114]
 [  -5.76838121   -4.8801391    -5.38542232   -6.89952518]
 [  -4.859053     -3.88981631   -4.11147429   -6.00114071]
 [  -3.97741023   -3.92299908   -2.94361401   -5.05141178]
 [ -13.10213026  -13.59260511  -14.93346175  -13.98085216]
 [ -12.44448976  -16.06102918  -21.05956151  -16.21328582]
 [ -11.71258746  -16.44282548  -22.60262199  -17.97379366]
 [ -10.92857112  -16.54726923  -27.94128565  -18.01292031]
 [ -10.06642079  -14.41541409  -31.77301912  -15.50879347]
 [  -9.2449463   -13.03968398  -17.54022433  -15.81909273]
 [  -8.43800259  -12.22189727  -17.43712313  -14.48709245]
 [  -7.66773696  -10.64936963  -20.17556753  -13.31724833]
 [  -8.88253201   -5.11340844  -18.58396853  -12.0547976 ]
 [  -7.58229299   -4.06628147   -9.43231586   -8.8110638 ]
 [  -5.74958669   -3.2180679    -6.41517785   -5.74353066]
 [  -4.09069836   -2.97814649   -1.98061493   -4.02644261]
 [ -13.86059037  -14.49072593  -15.71501172  -14.76059843]
 [ -13.32765534  -35.70351714 -130.23437341  -21.09892779]
 [ -16.61974791  -22.28787276 -124.39785827  -22.99976486]
 [ -17.12162905  -23.45401337 -103.69041681  -20.61361946]
 [ -15.71849053  -24.68171503 -112.82483424  -24.0780417 ]
 [ -14.05281921  -23.18034032 -110.24408909  -30.20507334]
 [ -13.75845418  -20.73110067 -105.00397236  -18.044477  ]
 [ -11.75974424  -25.10064777 -117.57980328  -17.02139791]
 [ -10.68171105  -19.80971604 -103.11682722  -14.68457968]
 [  -6.34285232  -20.54998233 -119.39813729  -14.62394801]
 [  -6.56139827   -1.98349247 -130.27737235  -27.56451153]
```

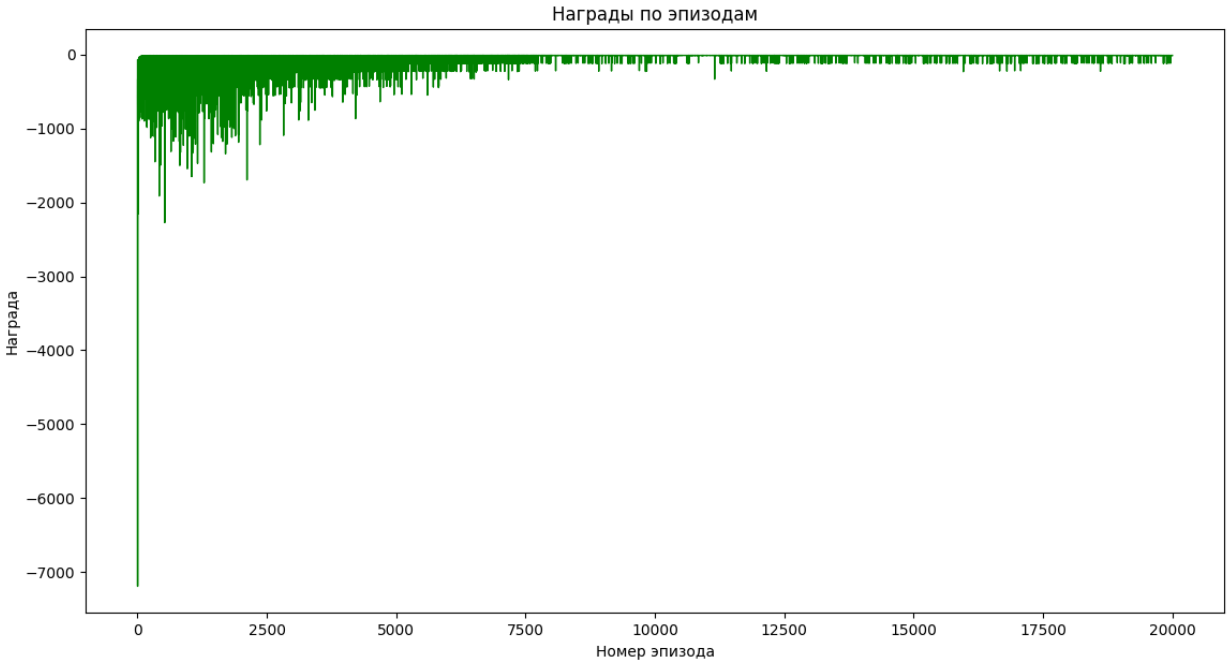
```
[  -3.21983721   -1.98556121   -1.           -3.55788611]
[ -14.61573483 -114.44566431  -15.55490564  -15.55862833]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            0.            0.            0.        ]
[   0.            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.        ]]
```



Результат работы программы для алгоритма Q-обучение:

```
Вывод Q-матрицы для алгоритма  Q-обучение
[[ -12.45270397  -12.3044813   -12.30313661  -12.37444341]
 [ -11.94528463  -11.54846627  -11.54850084  -12.65160216]
 [ -11.36513741  -10.76413164  -10.76413048  -11.96558577]
 [ -10.6956063    -9.96342817   -9.96342842  -11.40114599]
 [  -9.94465234   -9.14635911   -9.14635912  -10.73384939]
 [  -9.10583371   -8.31261184   -8.31261184   -9.92426865]
 [  -8.3014061    -7.46184886   -7.46184886   -9.10313758]
 [  -7.45439753   -6.59372334   -6.59372334   -8.29873603]
 [  -6.57977369   -5.70788096   -5.70788096   -7.40531547]
 [  -5.69685138   -4.80396016   -4.80396016   -6.57994263]
 [  -4.78366593   -3.881592     -3.881592     -5.69826102]
 [  -3.87322555   -3.8563188    -2.9404       -4.78174766]
 [ -13.03838437  -11.54888054  -11.54888054  -12.31767499]
 [ -12.31681597  -10.76416381  -10.76416381  -12.31789481]
 [ -11.5487584    -9.96343246   -9.96343246  -11.54888046]
 [ -10.76414587   -9.14635966   -9.14635966  -10.76416368]
 [  -9.96342819   -8.31261189   -8.31261189   -9.9634323 ]
 [  -9.14635925   -7.46184887   -7.46184887   -9.14635965]
 [  -8.31261184   -6.59372334   -6.59372334   -8.31261189]
 [  -7.46184885   -5.70788096   -5.70788096   -7.46184887]
 [  -6.59372333   -4.80396016   -4.80396016   -6.59372333]
 [  -5.70788095   -3.881592     -3.881592     -5.70788096]
```

```
[  -4.80396015   -2.9404       -2.9404       -4.80396016]
[  -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.        ]]
```



Результат работы программы для алгоритма двойное Q-обучение:

```
Вывод Q-матриц для алгоритма  Двойное Q-обучение
Q1
[[ -20.24893287  -12.42113121  -20.4598979   -22.09643158]
 [ -18.37088166  -14.88602271  -11.54889228  -20.13714044]
 [ -14.60492691  -20.4646009   -10.76464702  -16.79268653]
 [ -18.0067942   -10.13625225   -9.12440379  -13.78612599]
 [ -10.86500247   -9.16407886  -12.95083805  -18.39174401]
 [  -9.9686973    -8.31262931   -8.3949891   -10.69501427]
 [  -8.3551094    -7.59083722   -7.46184895   -9.67052011]
 [  -7.43643878   -6.59375135   -7.187719     -8.30314786]
 [  -6.63314037   -6.95706163   -5.70788096   -7.31245025]
 [  -5.68262692   -5.71506012   -5.85322386   -6.11543893]
```

```
 [  -5.08493959   -3.88111234   -3.62852598   -5.62934531]
 [  -3.80633858   -3.66819364   -2.94042119   -5.19008259]
 [ -13.74740423  -11.54888054  -11.72003728  -12.46406178]
 [ -12.32124895  -10.76416381  -10.76453926  -12.32176541]
 [ -11.54925187   -9.96353081   -9.96343246  -11.54895048]
 [ -19.76179207   -9.75866866   -9.14635966  -11.26247268]
 [ -10.41258869   -8.31261189   -8.31030792  -10.16074892]
 [  -9.21377726   -7.46184887   -7.50017907   -9.63678875]
 [  -8.31276334   -6.59372334   -6.59407776   -8.31275968]
 [  -7.46238245   -5.70788096   -5.70809777   -7.46185484]
 [  -6.59373967   -5.1584056    -4.80396016   -6.5937367 ]
 [  -6.89238334   -3.87469223   -3.881592     -5.67149436]
 [  -5.62632295   -2.9404       -2.98272293   -5.45125187]
 [  -3.83971848   -2.9382792    -1.98         -3.84496327]
 [ -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]
 [  -9.30267827   -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]
 [  -5.05548322   -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.        ]]
Q2
[[ -23.7794868   -12.50184646  -21.23338956  -19.91619608]
 [ -16.07657826  -16.79118781  -11.54895166  -21.75233315]
 [ -18.16627066  -13.60260676  -10.76418727  -14.71341602]
 [ -13.82659066  -10.15568303  -18.99368237  -14.3879406 ]
 [ -12.20373337   -9.1484039   -10.13462005  -13.05977619]
 [  -9.46884918   -8.31267166   -9.29186642  -11.62374269]
 [  -8.52677264   -7.67627448   -7.46184888   -9.19779521]
 [  -7.60739356   -6.59374755   -6.68174306   -8.41597217]
 [  -6.52571059   -6.64111144   -5.70788098   -7.43310165]
 [  -6.21587359   -4.801829     -4.83733421   -5.8595572 ]
 [  -4.60298275   -3.88200137   -4.77479504   -6.21674975]
 [  -3.55357628   -3.56214998   -2.94039854   -4.19613069]
 [ -13.78295523  -11.54888054  -11.8244807   -12.37383918]
 [ -12.33892703  -10.76416381  -10.76466349  -12.32039733]
 [ -11.55323295   -9.96376201   -9.96343246  -11.54982109]
 [ -11.64083689   -9.22941258   -9.14635966  -10.78132408]
 [ -10.41648719   -8.31261189   -8.47180742  -10.20669349]
 [  -9.18916849   -7.46184887   -7.46274967   -9.14967247]
 [  -8.31299411   -6.59372334   -6.59421545   -8.31669598]
 [  -7.46228557   -5.70788096   -5.70808101   -7.46201497]
 [  -6.59371338   -4.803963     -4.80396016   -6.59372318]
 [  -6.63021649   -4.13422389   -3.881592     -5.65092916]
 [  -4.59198203   -2.9404       -2.93707598   -4.2432909 ]
 [  -3.87025928   -2.9305449    -1.98         -3.96291   ]
 [ -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]
[  -3.92323849   -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.        ]]
```