Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Кафедра автоматизації проектування енергетичних процесів і систем

Лабораторна робота №4

з дисципліни «МЕТОДОЛОГІЯ РОЗРОБКИ ІНТЕЛЕКТУАЛЬНИХ КОМП'ЮТЕРНИХ ПРОГРАМ»

««Інтелектуальні агенти. Алгоритм Q-навчання»

Виконала: студентка 3 курсу

ТЕФ групи ТІ-01

Круть Катерина

Перевірив: д.т.н. Мусієнко А. П.

КИЇВ 2023

ПОСТАНОВКА МЕТИ ДОСЛІДЖЕННЯ

Мета дослідження: Ознайомитися з поняттям інтелектуального агента та одним з методів його навчання – Q-learning

Завдання 1

Побудувати математичну модель інтелектуального агента та його зовнішнього середовища:

– записати множину станів інтелектуального агента та цільовий стан;

– зобразити зовнішнє середовище агента у вигляді графа, в якому стани – вузли, а дії – ребра;

– записати матрицю суміжності побудованого графа.

Проілюструвати роботу алгоритму Q-навчання розрахунками без програмної реалізації. Для цього показати 2 спроби агента досягти цільової мети. Перший раз, обираючи маршрут цілком довільним чином (так як пам’ять агента порожня), другий раз – враховуючи здобуту пам’ять агента. Розрахунки повинні містити обчислення винагород за дії агента, значення матриці Q та зображення графа зовнішнього середовища з позначенням шляху агента і змінених значень ваг ребер.

Відповідно до побудованої в 1 завданні математичної моделі реалізувати програмне забезпечення для ілюстрації роботи алгоритму Q-навчання. Примітка: на початку програми ініціалізувати матрицю Q нулями

Набір данних

Варіант 2

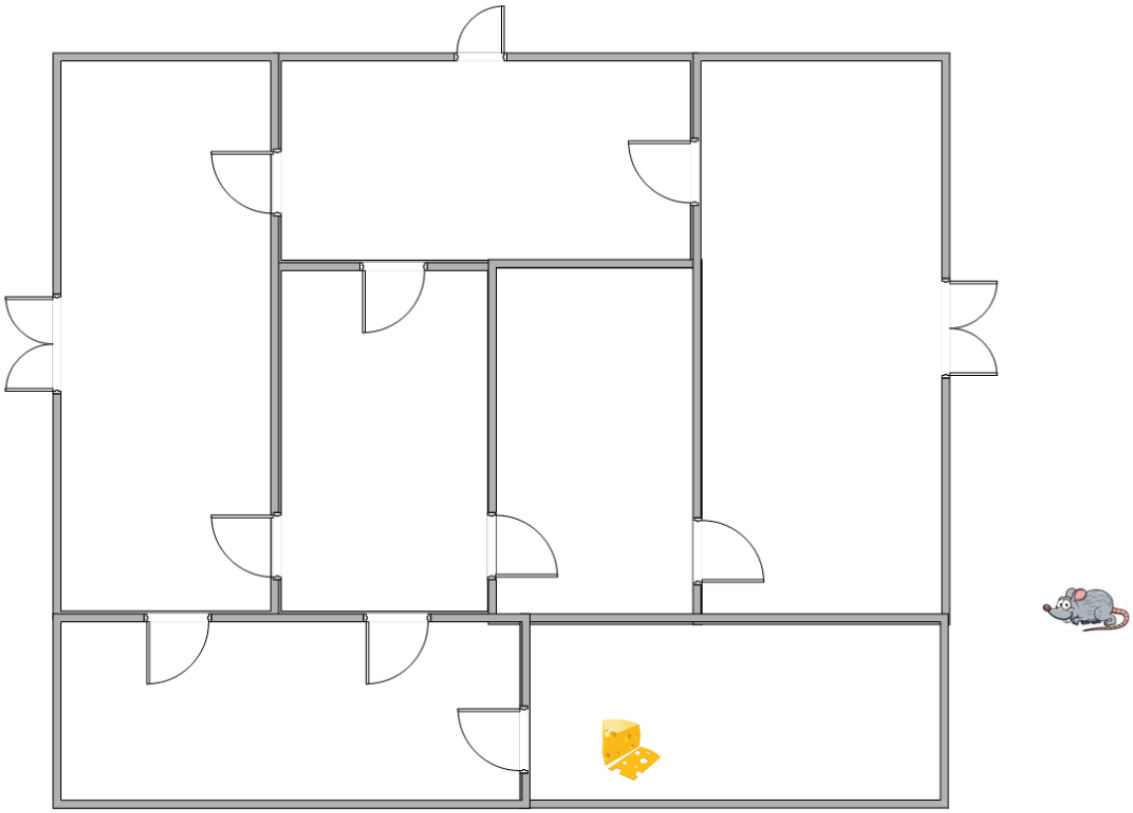


Рисунок А2

Ціль інтелектуального агента - вийти на вулицю.

ХІД РОБОТИ

Варіант 2

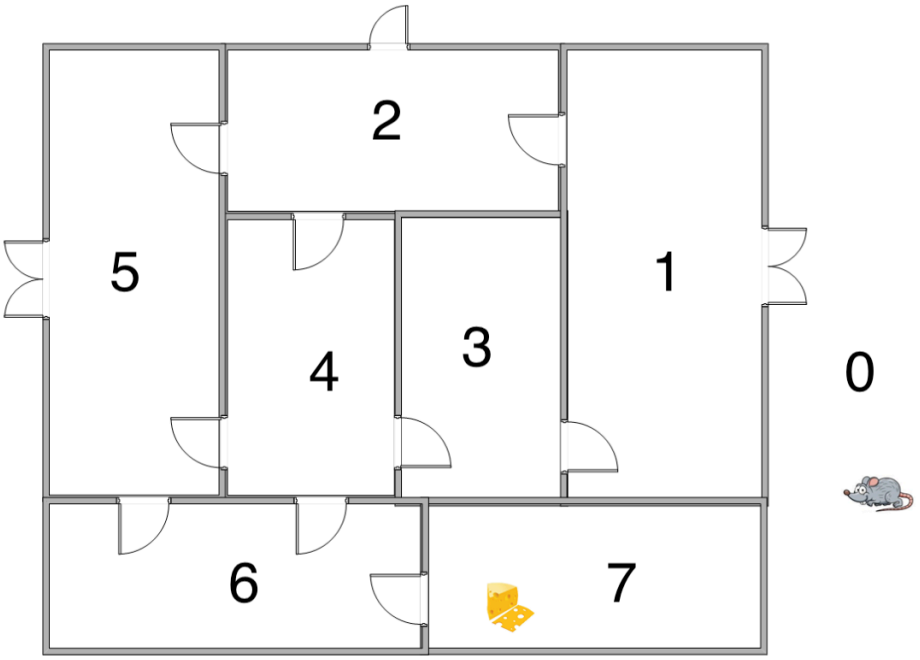
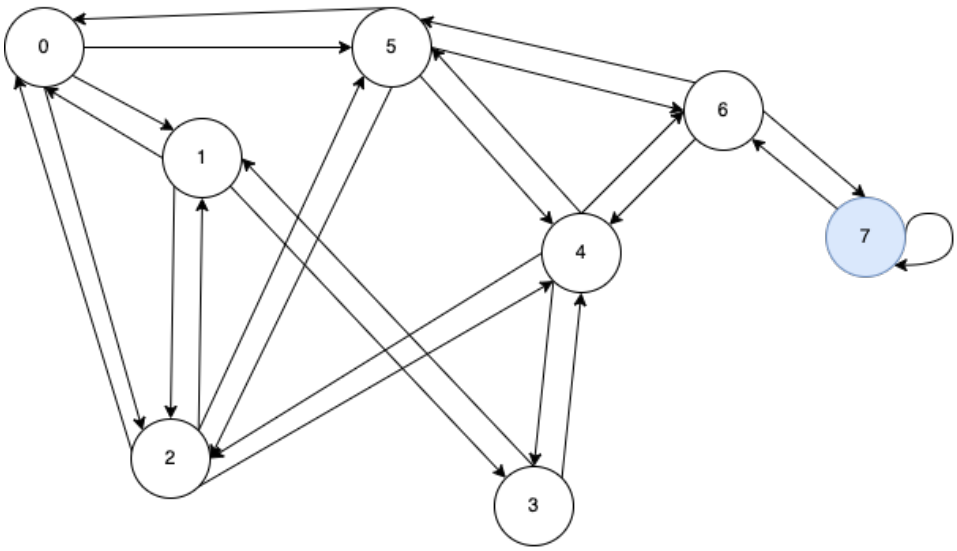


Рисунок А2

8 станів від 0 до 7, 7 – цільовий



Граф станів

Ініціалізація матриці R:

	0	1	2	3	4	5	6	7
0	-1	0	0	-1	-1	0	-1	-1
1	0	-1	0	0	-1	-1	-1	-1
2	0	0	-1	-1	0	0	-1	-1
3	-1	0	-1	-1	0	-1	-1	-1
4	-1	-1	0	0	-1	0	0	-1
5	0	-1	0	-1	0	-1	0	-1
6	-1	-1	-1	-1	0	0	-1	100
7	-1	-1	-1	-1	-1	-1	0	100

Ініціалізація матриці Q:

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

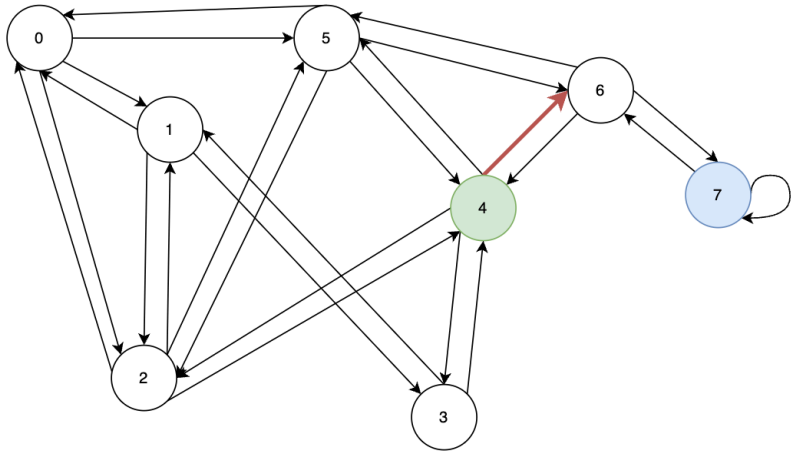
Ініціалізація параметра швидкості навчання

Gamma = 0,8

1 спроба.

1 крок. Агент у кімнаті №4.

З матриці R агент дізнається про свої можливі дії. Зі стану 4 агент може перейти в стани: 6, 5, 2 або 3, тобто здійснити дії (4, 6), (4, 5), (4, 2) або (4, 3). Оскільки агент ще не знає, який стан наблизить його до цільового стану, він обирає дію випадково. Припустимо, що агент випадковим чином обрав дію (4, 6).

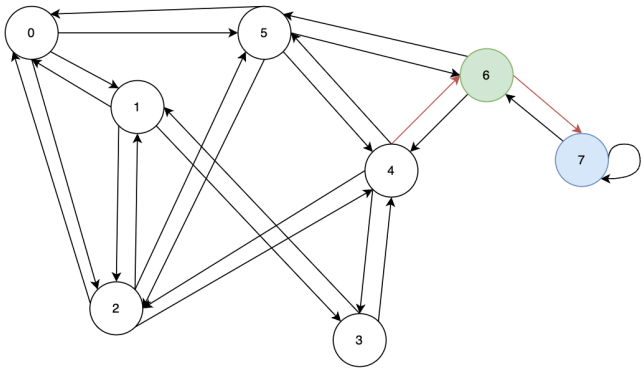


Матриця Q

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0

2 крок. Агент у кімнаті №6.

З матриці R агент дізнається про свої можливі дії. Зі стану 6 агент може перейти в стани: 5, 4 або 7, тобто здійснити дії (6, 4), (6, 5) або (6, 7). Оскільки агент ще не знає, який стан наблизить його до цільового стану, він обирає дію випадково. Припустимо, що агент випадковим чином обрав дію (7,9).



Агент отримує винагороду за дію:

$$Q(7, 9) = R(7, 9) + 0{,}8 * max(Q(7, 6), Q(7, 7)) = 100 + 0 = 100$$

Матриця Q

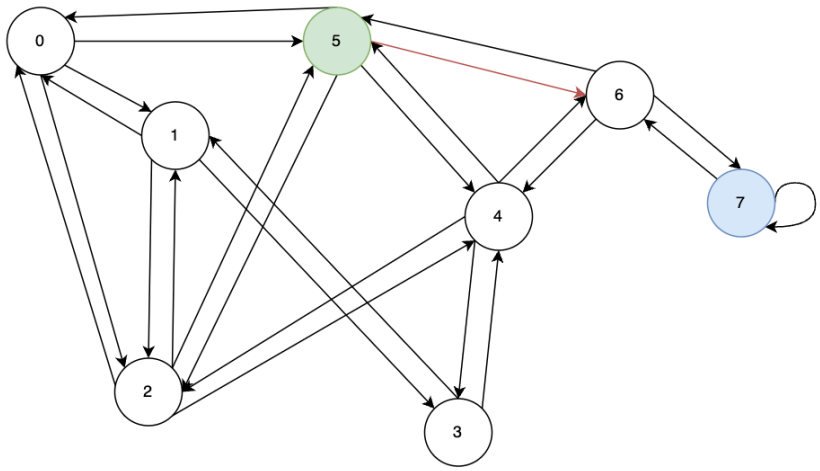
	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	100

Агент знаходиться у цільовому стані

2 спроба.

1 крок. Агент у кімнаті №5.

З матриці R агент дізнається про свої можливі дії. Зі стану 5 агент може перейти в стани: 0, 2, 4 або 6, тобто здійснити дії (5, 0), (5, 2), (5, 4) або (5, 6). Оскільки агент ще не знає, який стан наблизить його до цільового стану, він обирає дію випадково. Припустимо, що агент випадковим чином обрав дію (5, 6).



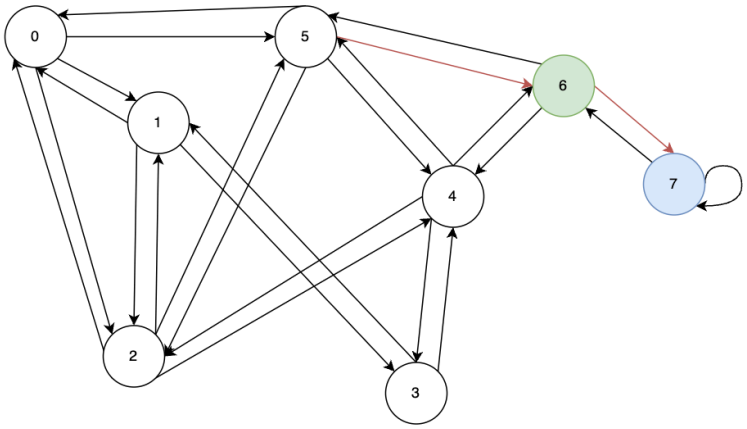
Агент отримує винагороду за дію:

$$Q(5, 6) = R(5, 6) + 0{,}8*max(Q(6, 4), Q(6, 5), Q(6, 7)) = 100 + 0{,}8*100 = 180$$

	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	80
7	0	0	0	0	0	0	0	0

2 крок. Агент у кімнаті №6.

З матриці R агент дізнається про свої можливі дії. Зі стану 6 агент може перейти в стани: 5, 4 або 7, тобто здійснити дії (6, 5), (6, 4) або (6, 7). Оскільки стан 7 відмічений вагою 100, то агент розуміє, що даний стан є цільовим. Отже агент вибирає перехід у цільовий стан, тобто дію (6, 7).



Агент отримує винагороду за дію:

$$Q(6, 7) = R(6, 7) + 0{,}8 * max(Q(7, 6), Q(7, 7)) = 0{,}8 * 100 + 100 = 180$$

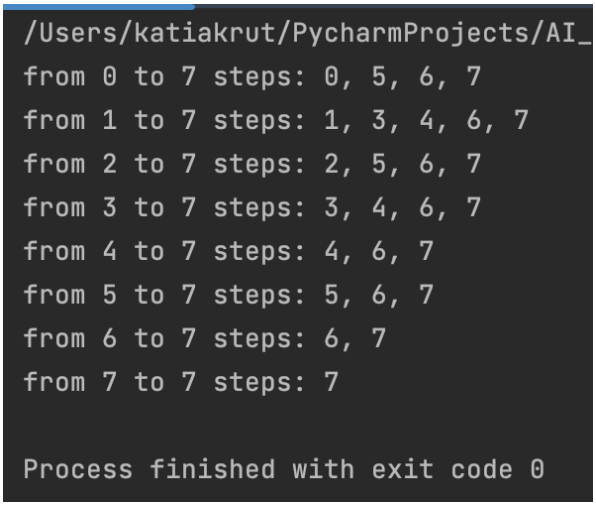
Агент знаходиться у цільовому стані

Код програми:

```
import numpy as np

R_MATRIX = np.array([
    [-1, 0, 0, -1, -1, 0, -1, -1],
    [0, -1, 0, 0, -1, -1, -1, -1],
    [0, 0, -1, -1, 0, 0, -1, -1],
    [-1, 0, -1, -1, 0, -1, -1, -1],
    [-1, -1, 0, 0, -1, 0, 0, -1],
    [0, -1, 0, -1, 0, -1, 0, -1],
    [-1, -1, -1, -1, 0, 0, -1, 100],
    [-1, -1, -1, -1, -1, -1, 0, 100],
])
END_STATE = 7

class Agent:
    def __init__(self, r_matrix, end_state, gamma=0.8, epsilon=0.1,
attempts=500):
        self.R = r_matrix.copy()
        self.end_state = end_state
        self.gamma = gamma
        self.epsilon = epsilon
        self.attempts = attempts
        self.Q = np.zeros(self.R.shape)
        self.states_l = len(self.R)

    def get_all_moves(self, state):
        return np.where(self.R[state] > -1)[0]

    def random_move(self, state):
        return np.random.choice(self.get_all_moves(state))

    def move(self, state):
        possible_moves = self.get_all_moves(state)
        max_move = np.max(self.Q[state, possible_moves])
        best_moves = [i for i in possible_moves if self.Q[state, i] ==
max_move]
        return np.random.choice(best_moves)

    def give_reward(self, prev_state, state):
        possible_move = self.get_all_moves(state)
        self.Q[prev_state, state] = self.R[prev_state, state] +
self.gamma * np.max(self.Q[state, possible_move])

    def get_next_state(self, state, epsilon):
        return self.random_move(state) if np.random.random() < epsilon
else self.move(state)

    def make_attempt(self, state, epsilon=0.0):
        steps = [state]
        while state != self.end_state:
            next_state = self.get_next_state(state, epsilon)
```

```
            self.give_reward(state, next_state)
            state = next_state
            steps.append(state)
        return steps

    def train(self, attempts):
        attempts = np.random.randint(0, self.states_l, attempts)
        for j in attempts:
            self.make_attempt(j, self.epsilon)


agent = Agent(R_MATRIX, END_STATE)
agent.train(agent.attempts)
for i in range(agent.states_l):
    print(f"from {i} to {END_STATE} steps: {', '.join(map(str,
agent.make_attempt(i)))}")
```

Приклад виконання програми



Висновки:

Під час виконання лабораторної роботи було набуто знань про можливості використання інтелектуального агента для пошуку шляху, та алгоритм його навчання Q-learning. Отримані знання було використано для створення програми, для знаходження найкоротшого шляху.