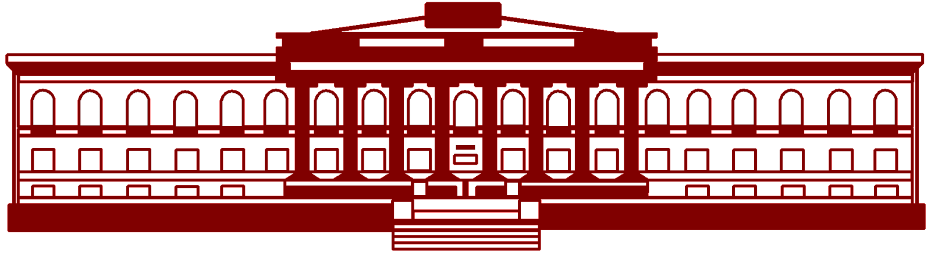
**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**імені ТАРАСА ШЕВЧЕНКА**

****

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра прикладних інформаційних систем**

**Звіт до лабораторної роботи №5**

**з курсу**

**«Системний аналіз та теорія прийняття рішень»**

*Студента 3 курсу*

*групи ПП-32*

*спеціальності 122 «Комп'ютерні науки»*

*ОП «Прикладне програмування»*

Чалого Єгора Олеговича

*Викладач:*

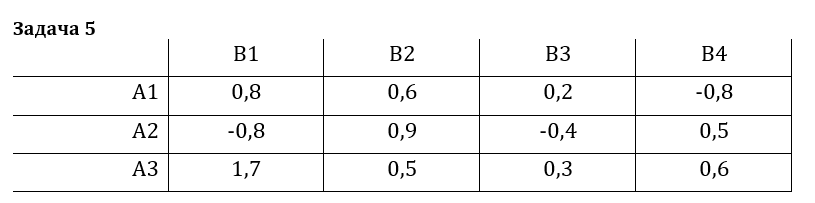
*Білий Р.О.*

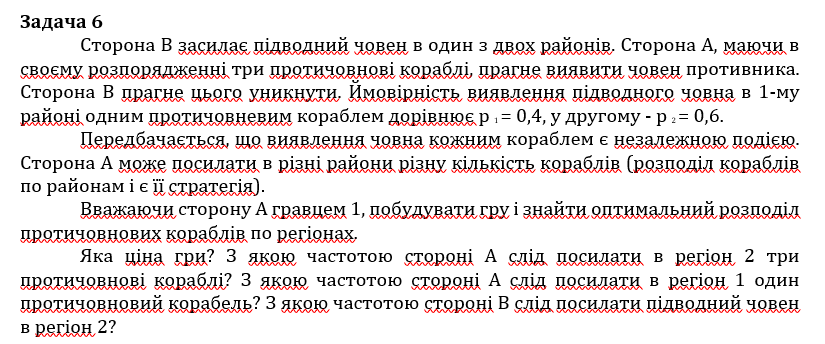
**Київ – 2023**

**Тема:** Теорія ігор та прийняття рішень в умовах конфлікту.

**Мета роботи:** Набути навички пошуку раціональних рішень в умовах конфліктів.

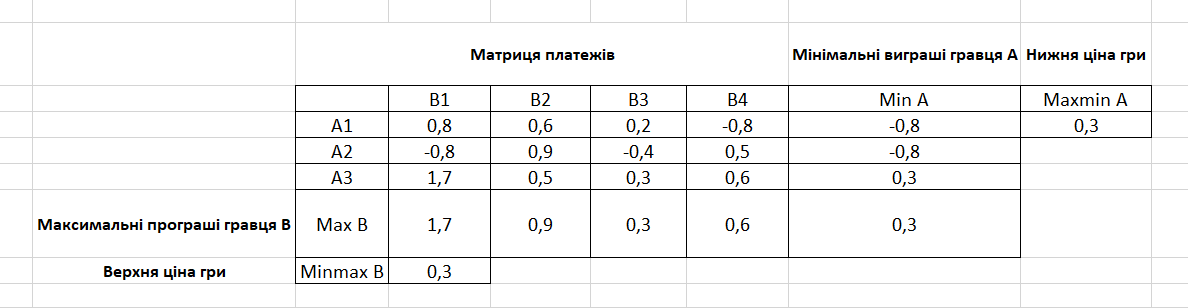
**Завдання:**





**Хід роботи:**

Розв’язав першу задачу засобами Excel:



Робимо висновок, що оскільки , то гра має сідлову точку та ціна гри буде y = = 0,3. Так як існує сідлова точка, тоді гра має рішення в чистих стратегіях.

Запрограмував рішення 1 задачі:

import numpy as np

import scipy.optimize as optimize

def solve\_game(gains\_matrix):

    min\_payoffs = gains\_matrix.min(axis=1)

    max\_payoffs = gains\_matrix.max(axis=0)

    maxmin = max(min\_payoffs)

    minmax = min(max\_payoffs)

    maxmin\_indexes = np.where(min\_payoffs == maxmin)

    minmax\_indexes = np.where(max\_payoffs == minmax)

    if minmax == maxmin:

        return minmax, maxmin\_indexes[0][0], minmax\_indexes[0][0]

    else:

        return minmax, maxmin\_indexes[0][0], maxmin, minmax\_indexes[0][0]

def main():

    task1\_matrix = np.array([

        [0.8, 0.6, 0.2, -0.8],

        [-0.8, 0.9, -0.4, 0.5],

        [1.7, 0.5, 0.3, 0.6]

    ])

    result = solve\_game(task1\_matrix)

    if len(result) == 3:

        print("\nThe game has a saddle point in pure strategies.")

        print(f"Game price: {result[0]}")

        print("Saddle point:", tuple(result[1:]))

    else:

        print("\nThe game doesn't have a saddle point in pure strategies.")

        print(f"Lower game price: {result[0]}")

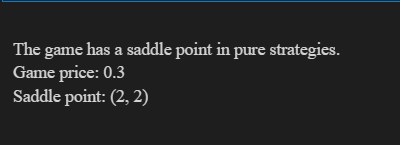
        print("Maximin strategy for Player 1 (rows):", result[1])

        print(f"Upper game price: {result[2]}")

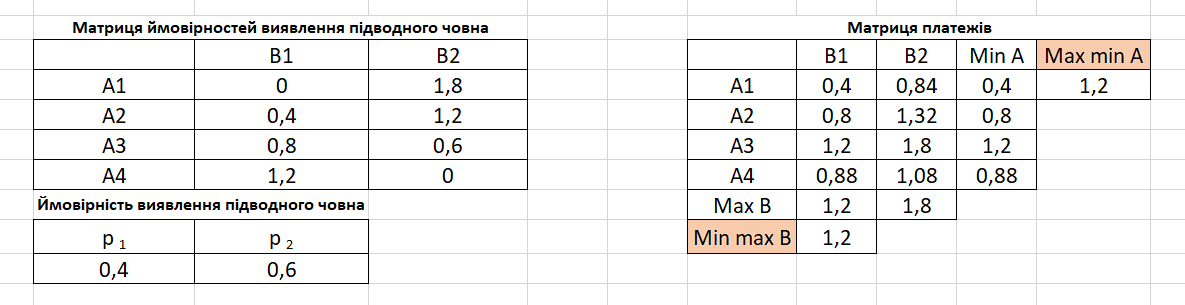
        print("Minimax strategy for Player 2 (columns):", result[3])

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Результат виконання:  


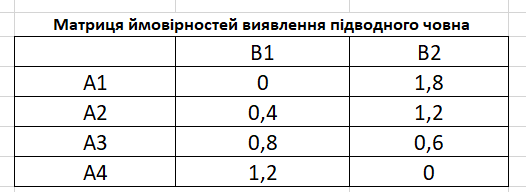
Розв’язав другу задачу засобами Excel:



Стратегії A:

Можна відправити 0 кораблей в 1 регіон та 3 кор. в 2 регіон, 1 кор. в 1 регіон та 2 кор. в 2 регіон, 2 кор. в 1 регіон та 1 кор. в 2 регіон і 3 кор. в 1 регіон та 0 кор. в 2 регіон.

Далі створюємо матрицю ймовірностей виявлення підводного човна, з урахуванням самих ймовірностей: в 1-му районі p 1 = 0.4, у другому - p 2 = 0.6.



Створюємо матрицю платежів враховуючи ймовірність взаємного розташування:



Робимо висновок, що оскільки , то гра має сідлову точку та ціна гри буде y = = 1,2. Так як існує сідлова точка, то гра має рішення в чистих стратегіях.

Оптимальна стратегія для сторони А полягає в тому, щоб максимізувати ймовірність виявлення підводного човна. Оскільки ймовірність виявлення у районі 2 більша, сторона А має надсилати всі свої кораблі у район 2.

Оптимальний розподіл протичовнових кораблів стороною А такий:

1. надсилати три протичовнових кораблі у район 2.
2. ніколи не надсилати кораблі у район 1.

Сторона В має надсилати підводний човен у район, який менш вигідний для сторони А. Оскільки сторона А не буде надсилати кораблі у район 1, сторона В також не буде цього робити, але буде надсилати їх у район 2.

Оптимальні частоти надсилання стороною А протичовнових кораблів - 100% у район 2 і стороною В - 100% у район 2.

Запрограмував рішення 2 задачі:

import numpy as np

payoff\_matrix = np.array([

    [0.4, 0.84],

    [0.8, 1.32],

    [1.2, 1.8],

    [0.88, 1.08]

])

min\_payoffs\_A = np.min(payoff\_matrix, axis=1)

lower\_price\_A = np.max(min\_payoffs\_A)

print("Нижня ціна гри для гравця A:", lower\_price\_A)

max\_payoffs\_B = np.max(payoff\_matrix, axis=0)

upper\_price\_B = np.min(max\_payoffs\_B)

print("Верхня ціна гри для гравця B:", upper\_price\_B)

saddle\_points = []

for i, row in enumerate(payoff\_matrix):

    for j, payoff in enumerate(row):

        if payoff == lower\_price\_A and payoff == upper\_price\_B:

            saddle\_points.append((i, j))

if saddle\_points:

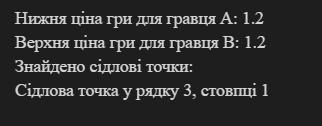
    print("Знайдено сідлові точки:")

    for point in saddle\_points:

        print(f"Сідлова точка у рядку {point[0]+1}, стовпці {point[1]+1}")

else:

    print("Сідлові точки не знайдено.")

Результат виконання:  


**Висновок:**

В ході виконання лабораторної роботи набув навички пошуку раціональних рішень в умовах конфліктів. Також навчився розв’язувати гру двох осіб в чистих і змішаних стратегіях за допомогою MS Excel і мови програмування Python.