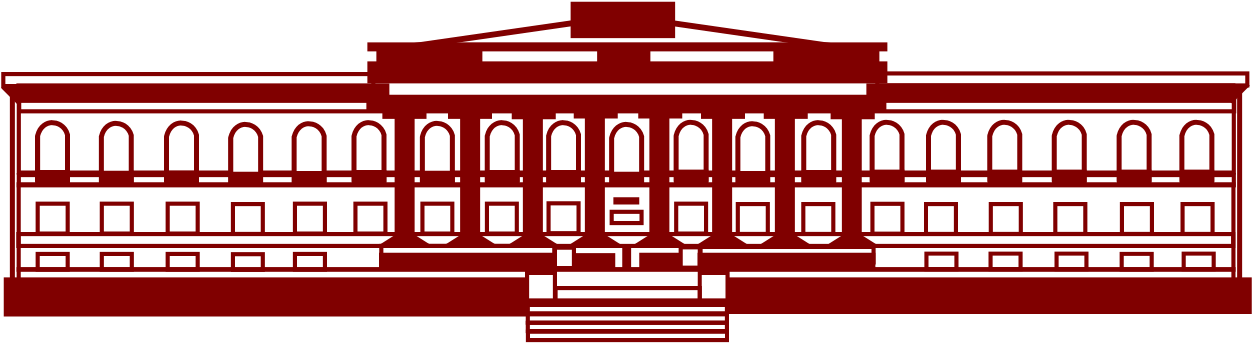
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ТАРАСА ШЕВЧЕНКА



**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра прикладних інформаційних систем**

Звіт до практичної роботи №8

**з курсу**

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

*студента 3 курсу групи ПП-32/3*

*спеціальності 122 «Комп'ютерні науки»*

*ОП «Прикладне програмування»*

Карпенка Глєба Олеговича

*Викладачі:*

д.е.н., професор Плескач В.Л.

ас. Білий Р.О.

**Київ – 2023**

**Тема.** Транспортна задача

**Мета.** Дослідити методи розв’язку транспорної задачі лінійного програмування.

**Хід роботи.**

**Завдння**

Автоматизувати процес розв’язку задачі методом північно-західного кута та методом потенціалів. (згідно варіанту).

ЗАДАЧА 7.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 40 | 19 | 25 | 25 | 35 | 230 |
| 49 | 26 | 27 | 18 | 38 | 250 |
| 46 | 27 | 36 | 40 | 45 | 170 |
| 140 | 90 | 160 | 110 | 150 |  |

ЗАДАЧА 3.

| 3 | 12 | 9 | 1 | 7 | 350 |
| --- | --- | --- | --- | --- | --- |
| 2 | 4 | 11 | 2 | 10 | 330 |
| 7 | 14 | 12 | 5 | 8 | 270 |
| 210 | 170 | 220 | 150 | 200 |  |

Автоматизований розв’язок з можливістю вибору різних методів.

import pandas as pd

import numpy as np

from scipy.optimize import linprog

class TransportTask:

    def read\_task(self, filename):

        df = pd.read\_csv(filename, header=None)

        self.supplies = df.iloc[-1, :-1].to\_numpy()

        self.demands = df.iloc[:-1, -1].to\_numpy()

        self.costs = df.iloc[:-1, :-1].to\_numpy()

    def print\_matrix(self):

        print('Supplies: ', self.supplies)

        print('Demands: ', self.demands)

        print('Costs:\n', self.costs)

    def solve(self):

        # Flatten the cost matrix to 1D array

        c = self.costs.flatten()

        # Coefficients for the inequality constraints Ax <= b

        A\_eq = np.zeros((len(self.supplies) + len(self.demands), len(c)))

        b\_eq = np.zeros(len(self.supplies) + len(self.demands))

        # Constraints for supplies (rows in A\_eq)

        for i in range(len(self.supplies)):

            A\_eq[i, i \* len(self.demands):(i + 1) \* len(self.demands)] = 1

            b\_eq[i] = self.supplies[i]

        # Constraints for demands (columns in A\_eq)

        for j in range(len(self.demands)):

            A\_eq[len(self.supplies) + j, j:len(c):len(self.demands)] = 1

            b\_eq[len(self.supplies) + j] = self.demands[j]

        # Bounds for decision variables (each variable is non-negative)

        bounds = [(0, None) for \_ in range(len(c))]

        # Solve the linear programming problem

        result = linprog(c, A\_eq=A\_eq, b\_eq=b\_eq, bounds=bounds, method='highs')

        # Reshape the result to a matrix

        allocation = result.x.reshape(self.costs.shape)

        return allocation

    def north\_west(self):

        n\_suppliers = len(self.supplies)

        n\_customers = len(self.demands)

        allocation = np.zeros((n\_suppliers, n\_customers))

        i, j = 0, 0

        while i < n\_suppliers and j < n\_customers:

            quantity = np.min([self.supplies[i], self.demands[j]])

            allocation[i, j] = quantity

            self.supplies[i] -= quantity

            self.demands[j] -= quantity

            if self.supplies[i] == 0:

                i += 1

            if self.demands[j] == 0:

                j += 1

        return allocation

**Висновок**

Під час виконання лабораторної роботи я аналізував процес розв’язку задачі методами північно-західного кута та потенціалів, а також розробив прогррамне рішення автоматизації розв’язку даної транспортної задачі.