МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ”ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра «Систем Автоматизованого Проектування»

Звіт
Лабораторна робота № 2
з курсу “ Дискретні моделі”
на тему: « АЛГОРИТМ РІШЕННЯ ЗАДАЧІ ЛИСТОНОШІ»

Виконав:
Ст.гр. КН-409
Єлечко Олег
Прийняв:
Кривий Р.З.

Львів 2023

Мета роботи: Метою даної лабораторної роботи є вивчення алгоритмів рішення задачі листоноші.

Теоретичні відомості:

Будь-який листоноша перед тим, як відправитись в дорогу повинен підібрати на пошті листи, що відносяться до його дільниці, потім він повинен рознести їх адресатам, що розмістились вздовж маршрута його проходження, і повернутись на пошту. Кожен листоноша, бажаючи втратити якомога менше сил, хотів би подолати свій маршрут найкоротшим шляхом. Загалом, задача листоноші полягає в тому, щоб пройти всі вулиці маршрута і повернутися в його початкову точку, мінімізуючи при цьому довжину пройденого шляху.

Перша публікація, присвячена рішенню подібної задачі, появилась в одному з китайських журналів, де вона й була названа задачею листоноші. Очевидно, що така задача стоїть не тільки перед листоношею. Наприклад, міліціонер хотів би знати найбільш ефективний шлях патрулювання вулиць свого району, ремонтна бригада зацікавлена у виборі найкоротшого шляху переміщення по всіх дорогах.

Задача листоноші може бути сформульована в термінах теорії графів. Для цього побудуємо граф G = (X , E), в якому кожна дуга відповідає вулиці в маршруті руху листоноші, а кожна вершина - стик двох вулиць. Ця задача являє собою задачу пошуку найкоротшого маршруту, який включає кожне ребро хоча б один раз і закінчується у початковій вершині руху.

ЕЙЛЕРОВИЙ ЦИКЛ

Ейлеревим циклом в графі називається шлях, який починається і закінчується в тій самій вершині, при чому всі ребра графа проходяться тільки один раз.

Ейлеревим шляхом називається шлях, який починається в вершині А, а закінчується в вершині Б, і всі ребра проходяться лише по одному разу. Граф, який включає в себе ейлерів цикл називається ейлеревим.

КРИТЕРІЙ ІСНУВАННЯ ЕЙЛЕРОВОГО ШЛЯХУ

Якщо G (ейлерів граф, то будь-який його ейлерів цикл неєдиний і відрізняється від інших ейлерових циклів графа G принаймні або зміною початкової вершини і/або зміною порядку проходження.

Для знаходження деякого ейлерового циклу в ейлеровому графі G можна застосувати так званий алгоритм Фльорі. Фіксуємо довільну початкову вершину циклу. На кожному кроці процедури до шуканого циклу обираємо (доки це можливо) те ребро, після вилучення якого граф не розіб’ється на дві нетривіальні зв’язні компоненти. Кожне обране ребро вилучаємо з G.

Процедура завершується, коли всі ребра буде вичерпано. Неважко обгрунтувати, що сформульований алгоритм будує ейлерів цикл графа G.

ЛАБОРАТОРНЕ ЗАВДАННЯ

1. Отримати у викладача індивідуальне завдання.
2. Підготувати програму для вирішення виданого завдання.
3. Запустити на виконання програму, що розв’язує задачу листоноші.

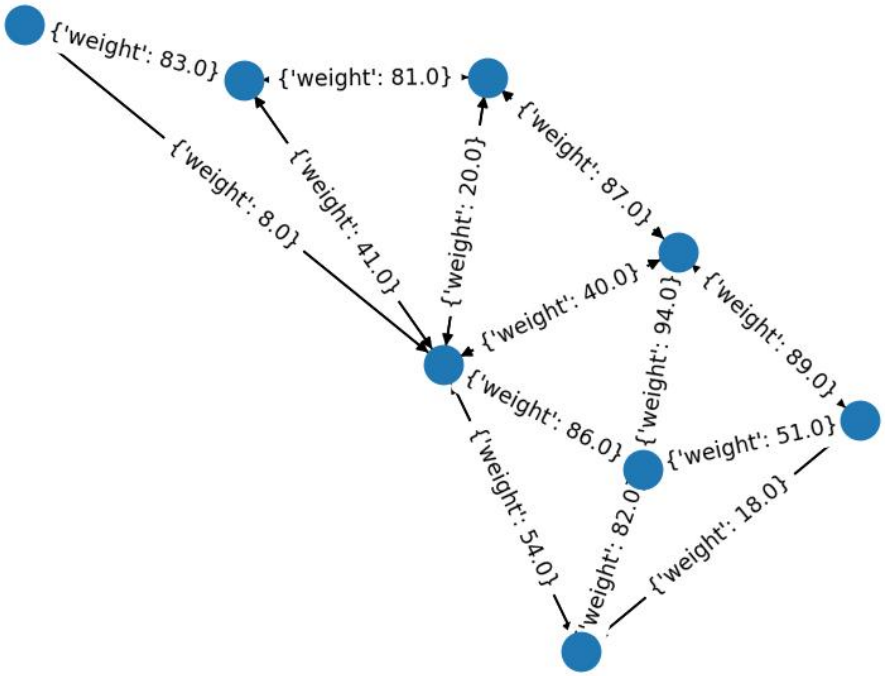
4. Проглянути результат роботи програми. Результат може бути позитивний (шлях знайдено) або негативний (шлях відсутній).
5. Здійснити перевірки роботи програм з результатами розрахунків.
6. Зафіксувати результати роботи.
7. Оформити і захистити звіт
   Github-link: https://github.com/OlehYelechko/Labs_DM/tree/main/Lab2

Код алгортиму

```
# Імпортуємо необхідні бібліотеки
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np
import itertools
from prettytable import PrettyTable


path1 = "l1_1.txt"
path2 = "l1_2.txt"
path3 = "l1_3.txt"


def tsp(weights_matrix):
    # ініціалізуємо змінні
    n = len(weights_matrix)
    nodes = range(n)
    best_path = None
    best_distance = float('inf')

    # перебираємо всі можливі перестановки вузлів
    for path in itertools.permutations(nodes):
        distance = 0

        # розраховуємо загальну відстань по маршруту
        for i in range(n - 1):
            distance += weights_matrix[path[i]][path[i + 1]]
        distance += weights_matrix[path[-1]][path[0]]  # додаємо відстань від останнього вузла до першого

        # перевіряємо, чи є даний маршрут найкращим
        if distance < best_distance:
            best_distance = distance
            best_path = path

    # повертаємо найдешевший маршрут та його відстань
    return best_path, best_distance
# Граф на основі матриці суміжності


with open(path1) as f:
    lines = (line for line in f if not line.startswith('#'))
    graph = np.loadtxt(lines, skiprows=1)
```

```
G = nx.from_numpy_matrix(np.matrix(graph), create_using=nx.DiGraph)
layout = nx.spring_layout(G)
nx.draw(G, layout)
nx.draw_networkx_edge_labels(G, pos=layout)
plt.show()


print(graph, end="\n\n")
# Визначаємо кількість вершин у графі
V = len(graph)

# викликаємо функцію tsp()
best_path, best_distance = tsp(graph)
result = [i for i in best_path]
result.append("Довжина:" + str(best_distance))
table = PrettyTable(result)
print(table)
```

Результат виконання:




```
[[ 0.  0.  0.  0.
 [ 0.  0. 81.  0.
 [ 0. 81.  0. 83.
 [ 0.  0. 83.  0.
 [86. 20. 41.  8.
 [94. 87.  0.  0.
 [51.  0.  0.  0.
 [82.  0.  0.  0.

+---+---+---+---+
| 0 | 1 | 6 | 4 |
+---+---+---+---+---+---+---+---+
+---+---+---+---+---+---+---+---+
```

Висновок: на цій лабораторній роботі було реалізовано найкоротшого шляху для листоноші. Ідея реалізації алгоритму полягала в алгоритмі глибокого пошуку.