МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ”ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра «Систем Автоматизованого Проектування»

Звіт
Лабораторна робота № 3
з курсу *“* Дискретні моделі*”*
на тему: « АЛГОРИТМ РІШЕННЯ ЗАДАЧІ КОМІВОЯЖЕРА»

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

Львів 2023

Мета роботи: Метою даної лабораторної роботи є вивчення і дослідження алгоритмів рішення задачі комівояжера.

Теоретичні відомості:

Невідомо, коли проблему комівояжера було досліджено вперше. Однак, відома видана в 1832 році книжка з назвою «Комівояжер — як він має поводитись і що має робити для того, аби доставляти товар та мати успіх в своїх справах — поради старого Кур'єра», в якій описано проблему, але математичний апарат для її розв'язання не застосовується. Натомість, в ній запропоновано приклади маршрутів для деяких регіонів Німеччини та Швейцарії.

Проблему комівояжера можна представити у вигляді моделі на графі, тобто, використовуючи вершини та ребра між ними. Таким чином, вершини графу відповідають містам, а ребра між вершинами і та j сполучення між цими містами. У відповідність кожному ребру можна зіставити вагу , яку можна розуміти як, наприклад, відстань між містами, час або вартість подорожі. Маршрутом (також гамільтоновим маршрутом) називається маршрут на цьому графі до якого входить по одному разу кожна вершина графа. Задача полягає у відшуканні найкоротшого маршруту.

З метою спрощення задачі та гарантії існування маршруту, зазвичай вважається, що модельний граф задачі є повністю зв'язним, тобто, що між довільною парою вершин існує ребро. Це можна досягти тим, що в тих Національний університет «Львівська політехніка». Кафедра САП Інструкція до лабораторної роботи №3 з курсу “Дискретні моделі в САПР” випадках, коли між окремими містами не існує сполучення, вводити ребра з максимальною вагою (довжиною, вартістю тощо). Через велику довжину таке ребро ніколи не потрапить до оптимального маршруту, якщо він існує.

Загальною задачею комівояжера називають задачу пошуку маршруту найменшої довжини. Задачею комівояжера називають задачу пошуку гамільторового контура найменшої довжини. Контур комівояжера, який має найменшу довжину, називають оптимальним гамільтоновим контуром він є оптимальним рішенням задачі комівояжера. Оптимальний маршрут комівояжера не обов`язково є гамільтоновим контуром.

ЗАСТОСУВАННЯ МЕТОДУ ГІЛОК І ГРАНІЦЬ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ КОМІВОЯЖЕРА

Нехай розглядається граф G= (X , A). Для отримання нижньої границі L1 довжини гамільтонового контура найменшої довжини на графі G може бути використаний алгоритм побудови потоку мінімальної вартості.

Якщо отриманий за допомогою цього алгоритма оптимальний потік відповідає деякому контуру на графі G, то цей контур є оптимальним гамільтоновим контуром, і на цьому рішення задачі закінчується. Але існує достатньо велика ймовірність того, що оптимальний потік, отриманий для будьякого графа, буде відповідати декільком незв’язним контурам. В цьому випадку довільно вибираємо один контур і позначаємо його через Gi, тоді Xc = { x1, x2, …, xk } множина вершин, що входить в нього.

В оптимальному рішенні комівояжер, виходячи з вершини x1 Xc переміщується або у вершину, яка належить множині Xc, або у вершину, яка не належить множині Xc. Якщо він

прийшов у вершину x Xc, то з неї він знову переміщується або у вершину множини Xc, або у вершину, що не належить Xc і т. д

ЛАБОРАТОРНЕ ЗАВДАННЯ

1. Отримати у викладача індивідуальне завдання.
2. Підготувати програму для вирішення виданого завдання.
3. Запустити на виконання програму відповідного методу.
4. Проглянути результат роботи програм. Результат роботи може бути позитивним (шлях знайдено) або негативним (шлях відсутній).
5. Порівняти результати, отримані за допомогою різних алгоритмів і зробити висновок.
6. Зафіксувати результати роботи у викладача.
7. Оформити і захистити звіт
   Github-link: https://github.com/OlehYelechko/Labs_DM/tree/main/Lab3

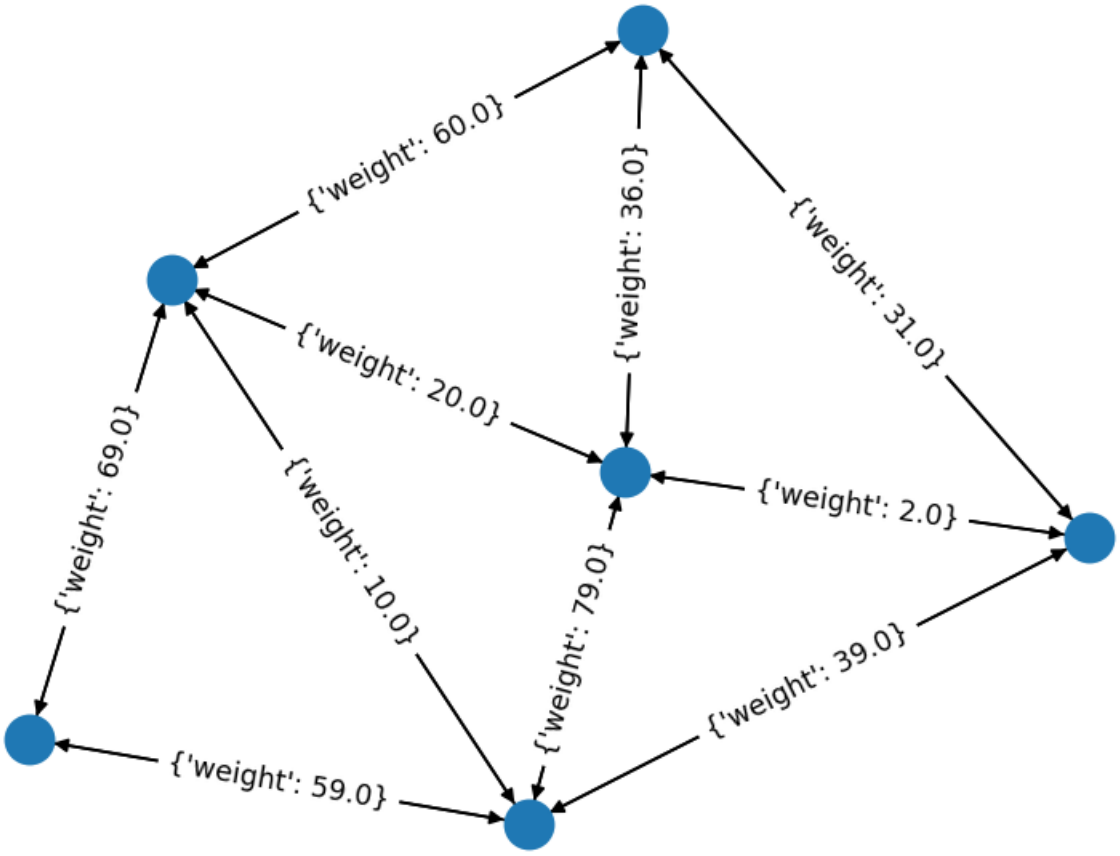
Код алгортиму МЕТОДУ ГІЛОК І ГРАНІЦЬ.

```
# Імпортуємо необхідні бібліотеки
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np

path1 = "l3_1.txt"
path2 = "l3_2.txt"

def branch_and_bound(adj_matrix):
    # Initialize variables
    n = len(adj_matrix)
    best_path = []
    best_cost = float('inf')
    stack = [(0, [0], 0)]

    # Iterate over stack until empty
    while stack:
        # Pop top element from stack
        node, path, cost = stack.pop()

        # If path is complete, check if it's better than the current best
        if len(path) == n:
            if cost + adj_matrix[node][0] < best_cost:
                best_path = path + [0]
                best_cost = cost + adj_matrix[node][0]

        # Otherwise, expand the node
        else:
            # Get possible next nodes and their costs
            next_nodes = [i for i in range(n) if i not in path]
            next_costs = [adj_matrix[node][i] for i in next_nodes]
```

```
        # Sort nodes and costs by increasing cost
        sorted_indices = np.argsort(next_costs)

        # Add nodes to stack in increasing order of cost
        for i in sorted_indices:
            next_node = next_nodes[i]
            next_cost = next_costs[i]
            if cost + next_cost + adj_matrix[next_node][0] < best_cost:
                stack.append((next_node, path + [next_node], cost + next_cost))

    return best_path, best_cost

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)

best_path, best_distance = branch_and_bound(np.array(graph))
result = [str(i) for i in best_path]
result.append("Довжина:" + str(best_distance))
print(result)
```

Результат виконання:

```
['0', '5', '1', '2', '3', '4', '0']
```



Висновок: на цій лабораторній роботі було реалізовано метод гілок та границь для вирішення задачі Комівояжера на мові Python.