НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

«ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Лабораторна робота №2

з курсу «Дискретні моделі в САПР»:

АЛГОРИТМ РІШЕННЯ ЗАДАЧІ ЛИСТОНОШІ

Виконав:
Ст.гр.КН-409
Погуляєв В.В.

.

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.

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. У випадку, коли шлях знайдено (не знайдено), необхідно модифікувати граф, коректуючи два або три зв’язки таким чином, щоб знайти граф, на якому задача листоноші не розв’язується (розв’язується).

6. Здійснити перевірки роботи програм з результатами розрахунків, проведених вручну.

7. Зафіксувати результати роботи.

8. Оформити і захистити звіт.

Код програми

Пошук Ейлерового шляху:

```
public List<Integer> findEulerPath() {
    List<Boolean> visited = new ArrayList<>(edges.size());
    for (int i = 0; i < edges.size(); i++) {
        visited.add(false);
    }

    Stack<Edge> stack = new Stack<>();
```

```
    List<Integer> path = new ArrayList<>();

    stack.push(edges.get(0));

    while (!stack.isEmpty()) {
        Edge cur = stack.pop();

        if (visited.get(edges.indexOf(cur))) {
            continue;
        }

        visited.set(edges.indexOf(cur), true);

        if (path.isEmpty()) {
            path.add(cur.from);
            path.add(cur.to);
        } else if (path.get(path.size() - 2) == cur.from) {
            Collections.swap(path, path.size() - 1, path.size() - 2);
            path.add(cur.to);
        } else if (path.get(path.size() - 1) == cur.from) {
            path.add(cur.to);
        } else {
            path.add(cur.from);
        }
        System.out.println(path);

        for (Edge edge : edges) {
            if (visited.get(edges.indexOf(edge))) {
                continue;
            } else if (edge.from == cur.from || edge.to == cur.to || edge.to ==
cur.from || edge.from == cur.to) {
                stack.push(edge);
            }
        }
    }

    return path;
}
```

Перевірка на існування Ейлерового циклу:

```
public static boolean isEulerianGraph(int[][] graph) {
    int begin = 0;
    int end = graph.length;
    for (int i = 0; i < graph.length; ++i) {
        int counter = 0;
        for (int j = begin; j < end; ++j) {
            if (graph[i][j - begin] != 0) {
                ++counter;
            }
        }
        begin = end;
        end  +=  graph.length;
        if (counter % 2 != 0) {
            return false;
        }
    }
    return true;
}
```

Обрахунок ціни Ейлерового шляху:

```
public int calculateEulerPath(List<Integer> path, Graph graph){
    int cost = 0;

    for (int i = 0; i < path.size() - 1; i++){
        int from = path.get(i);
        int to = path.get(i+1);
        for (Edge edge : graph.edges){
            if(edge.from == from && edge.to == to || edge.from == to && edge.to
== from){
                cost += edge.weight;
                break;
            }
        }
    }
    return cost;
}
```

Посилання на GitHub –https://github.com/flipppflopp/DM_Pohuliaiev

Аналіз результатів

Аналітичний розв’язок:

Матриця суміжності:

0 3 6 0 0

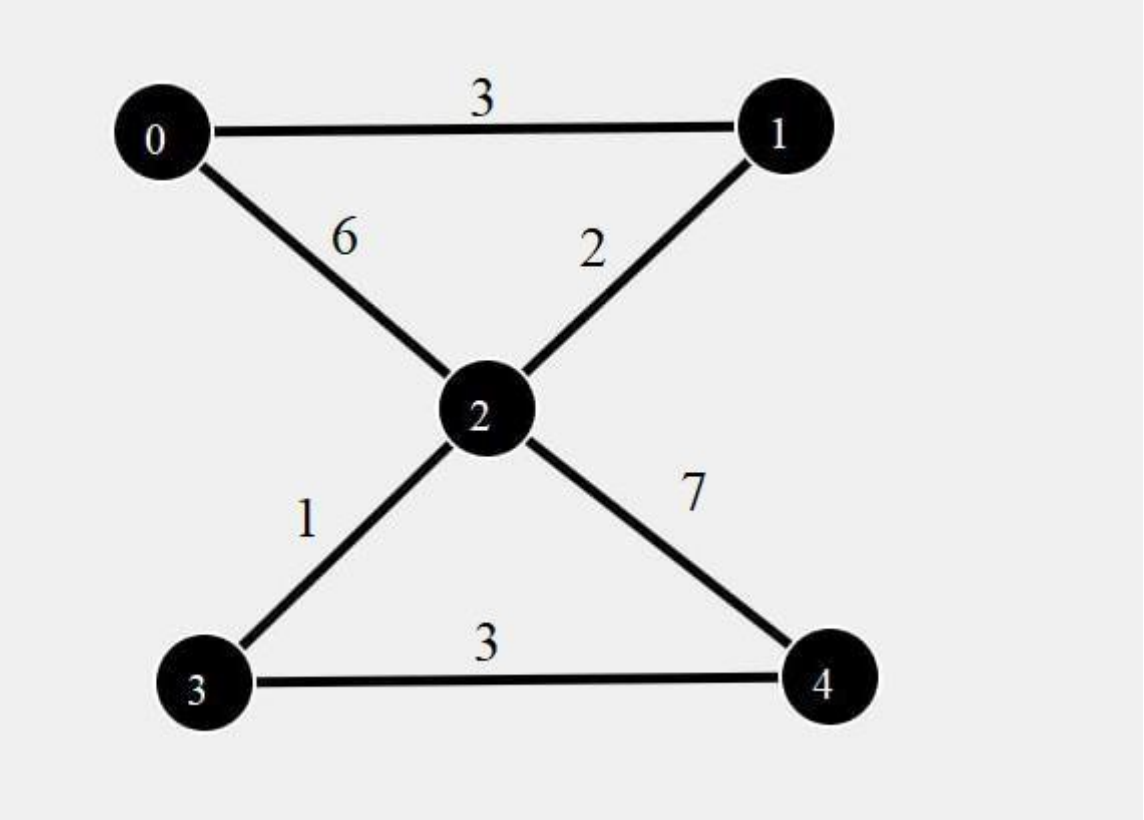
3 0 2 0 0

6 2 0 1 7

0 0 1 0 3

0 0 7 3 0

Граф:



Аналітичний обрахунок:

Насамперед перевіримо чи даний граф містить Ейлеровий цикл. Для цього необхідно перевірити чи є кожна вершина, даного графа, парною, тобто містить парну кількість ребер. Отже, даний граф містить Ейлеровий цикл, тому що кожна вершина є парною.

Тепер необхідно знайти Ейлеровий шлях і обрахувати його ціну. Починаємо з вершини “0”: 0->1->2->4->3->2->0.

Ціна даного шляху – 22.

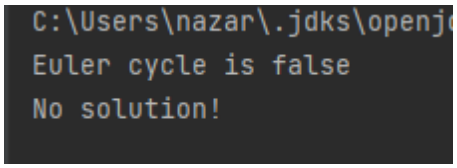
Результат виконання програми для аналітичного обрахунку:

```
C:\Users\nazar\.jdks\openjdk-18.0.1\bin
Euler cycle is true
Eulerian Path: [0, 1, 2, 4, 3, 2, 0]
Cost Eulerian Path: 22
```

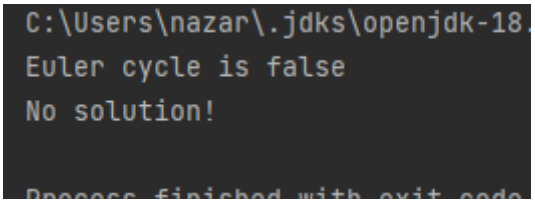
Результат виконання програми використовуючи тестовий файл I2_1.txt:

```
C:\Users\nazar\.jdks\openj
Euler cycle is false
No solution!
```

Результат виконання програми використовуючи тестовий файл I2_2.txt:



Результат виконання програми використовуючи тестовий файл I2_3.txt:



Висновок

В ході виконання лабораторної роботи, вивчив алгоритми рішення задачі листоноші. Розробив програму на основі якої перевірив виконання тестових завдань, якщо в графі існує Ейлеровий цикл, то шукається шлях і його ціна, якщо ні, то програму завершується. Створив матрицю суміжностей для аналітичних розрахунків, провів розрахунки і порівняв із програмним результатом.