Прізвище:
Дацишин
Ім'я: Роман
Група: КН-405
Варіант: 4

Кафедра.: Кафедра Систем
Автоматизованого Проектування
Дисципліна: Дискретні моделі в САПР
Перевірив: Кривий Р.З.

Звіт
До лабораторної роботи №2
На тему "Алгоритм рішення задачі листоноші"

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

Короткі теоретичні відомості:

Задача листоноші для неорієнтованого графа G(X,E) - це задача для графа, в якому ребра можна проходити в будь-якому з двох напрямків.

Необхідно розгянути окремо наступні два випадки :

Випадок 1 : Граф G парний.

Випадок 2 : Граф G непарний.

Випадок 1 : Якщо граф парний, то оптимальним рішенням задачі є ейлеровий маршрут. В цьому випадку листоноша не повинен обходити більше одного разу будь-яку вулицю, в даному випадку ребро графа.

Як найти на графі G ейлеровий маршрут, в якому "S" – початкова вершина? Для цього необхідно пройти будь-яке ребро (S,x) інцидентне вершині "S", а потім ще невикористане ребро, інцидентне вершині "x". Кожен раз, коли листоноша приходить в деяку вершину, є невикористане ребро по якому листоноша покидає цю вершину. Дуги, по яким здійснений обхід, створюють цикл C1 . Якщо в цикл C1 ввійшли всі ребра графа G, то C1 є ейлеровим маршрутом (оптимальним для задачі).

В іншому випадку треба створити цикл C2, який складається з невикористаних ребер і який починається з невикористаного ребра. Створення циклів C3, C4,..., продовжується доти, доки не будуть використані всі ребра графа. Далі треба об'єднати цикли C1,C2,C3,... в один цикл C, який містить всі ребра графа G. В цикл C кожне ребро графа входить лише один раз, і тому він є оптимальним рішенням задачі листоноші.Два цикла C1 і C2 можуть бути з'єднані тільки тоді, коли вони мають спільну вершину "x".

Для з'єднання двох таких циклів необхідно вибрати в якості початкового довільне ребро циклу C1 і рухатися вздовж його ребер до вершини "x". Далі потрібно пройти всі ребра циклу C2 і повернутися у вершину "x". На кінець, потрібно продовжити прохід ребер цикла C1 до повернення назад до початкового ребра. Пройдений маршрут є циклом, отриманим в результаті з'єднання циклів C1 і C2. Ця процедура може бути легко розширена на випадок з'єднання довільної кількості циклів і може виконуватись до тих пір, поки не утвориться дві їх підмножини, які не мають загальних вершин.

Індивідуальне 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.

Вхідні дані (файл 1):

8
0 0 0 0 86 94 51 82
0 0 81 0 20 87 0 0
0 81 0 83 41 0 0 0
0 0 83 0 8 0 0 0
86 20 41 8 0 40 0 54
94 87 0 0 40 0 89 0
51 0 0 0 0 89 0 18
82 0 0 0 54 0 18 0

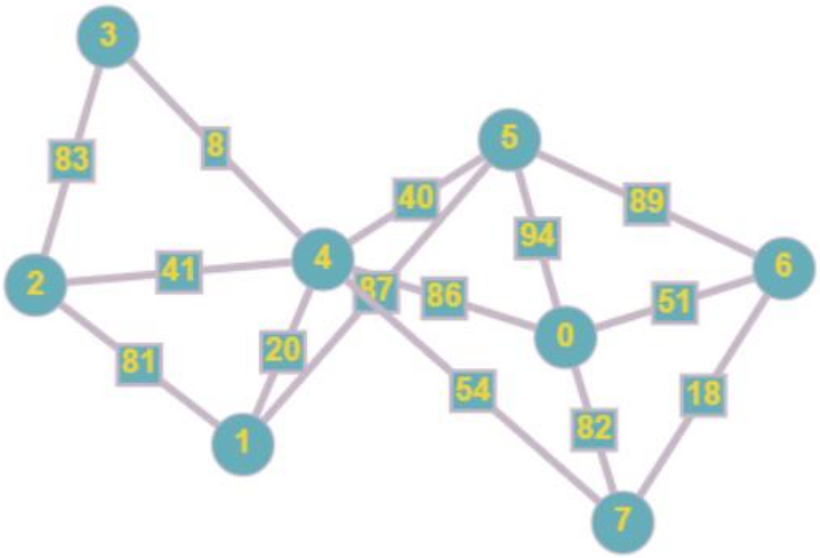

Рис. 1 Візуальне представлення графа

Було написано програму для знаходження Ейлерового циклу. Код програми , а також результат її виконання наведено нижче

Код програми

```
let initialString = "";
let matrixLength;
let html = "";
let m1 = [];

let input = document.querySelector("input");

input.addEventListener("change", () => {
  let files = input.files;

  if (files.length == 0) return;

  const file = files[0];
```

```
  let reader = new FileReader();

  reader.onload = (e) => {
    let file = e.target.result;
    file = file.replace(/(\r\n|\n|\r)/gm, " ");

    initialString = file;
    parseMatrix(initialString);
  };

  reader.onerror = (e) => alert(e.target.error.name);

  reader.readAsText(file);
});

const parseMatrix = (matrix) => {
  matrixLength = matrix[0];
  let str = matrix.substring(2);

  let arr = [];
  let num = "";

  for (let i = 0; i < str.length; i++) {
    if (str[i] !== " ") {
      num += str[i];
    } else {
      arr.push(parseInt(num));
      num = "";
    }
  }

  let m = [];
  for (let i = 0; i < matrixLength; i++) {
    m[i] = new Array(matrixLength);
  }

  let j = 0;
  let k = 0;

  for (let i = 0; i < arr.length; i++) {
    if (j < Math.sqrt(arr.length)) {
      m[k][j] = arr[i];
      j++;
    } else {
      j = 0;
      k++;
      m[k][j] = arr[i];
      j++;
    }
  }

  html += `<br/><h1>Open a terminal</h1><br/>`;
  document.getElementById("container").innerHTML = html;
  postman(m);
};
```

```
const eulerianPath = (a, b = []) => {
  let sum = 0;

  for (let i = 0; i < a.length; i++) {
    for (let j = 0; j < a[i].length; j++) sum += a[i][j];
  }

  sum /= 2;

  for (let i = 0; i < b.length; i++) sum += a[b[i][0]][b[i][1]];

  let stack1 = [0];
  let stack2 = [0];

  let edgeCount = 2;
  let limit = 0;

  while (edgeCount > 1 && limit < 25) {
    limit++;

    let next = "O";

    for (let i = 0; i < a[stack1[stack1.length - 1]].length; i++) {
      if (a[stack1[stack1.length - 1]][i] != 0 && next == "O") next = i;
    }

    console.log("STACK1 = " + stack1);
    console.log("STACK2 = " + stack2);

    if (next == stack2[stack2.length - 1]) {
      stack2.push(stack1[stack1.length - 1]);

      let newB = [];
      let del = 0;

      for (let i = 0; i < b.length; i++) {
        if (
          !(
            b[i][0] == stack1[stack1.length - 2] &&
            b[i][1] == stack1[stack1.length - 1]
          ) &&
          !(
            b[i][0] == stack1[stack1.length - 1] &&
            b[i][1] == stack1[stack1.length - 2]
          )
        )
          newB.push(b[i]);
        else {
          del++;
        }
      }

      b = [...newB];

      if (del == 0) {
```

```
      a[stack1[stack1.length - 1]][next] = 0;
      a[next][stack1[stack1.length - 1]] = 0;
    }
  } else {
    stack1.push(next);

    console.log(
      stack1[stack1.length - 1] + " : " + stack1[stack1.length - 2]
    );

    let newB = [];
    let del = 0;

    for (let i = 0; i < b.length; i++) {
      if (
        !(
          b[i][0] == stack1[stack1.length - 2] &&
          b[i][1] == stack1[stack1.length - 1]
        ) &&
        !(
          b[i][0] == stack1[stack1.length - 1] &&
          b[i][1] == stack1[stack1.length - 2]
        )
      )
        newB.push(b[i]);
      else {
        del++;
      }
    }

    b = [...newB];

    if (del == 0) {
      a[stack1[stack1.length - 1]][stack1[stack1.length - 2]] = 0;
      a[stack1[stack1.length - 2]][stack1[stack1.length - 1]] = 0;
    }
  }

  edgeCount = 0;

  for (let i = 0; i < a.length; i++) {
    for (let j = 0; j < a[i].length; j++) if (a[i][j] != 0) edgeCount++;
  }

  edgeCount /= 2;
  edgeCount += b.length;
  console.log(edgeCount);
}

console.log(stack1);
console.log(stack2);

let result = [...stack2];

for (let i = stack1.length - 1; i >= 0; i--) result.push(stack1[i]);
```

```
  console.log(result);
  console.log(sum);
};
```

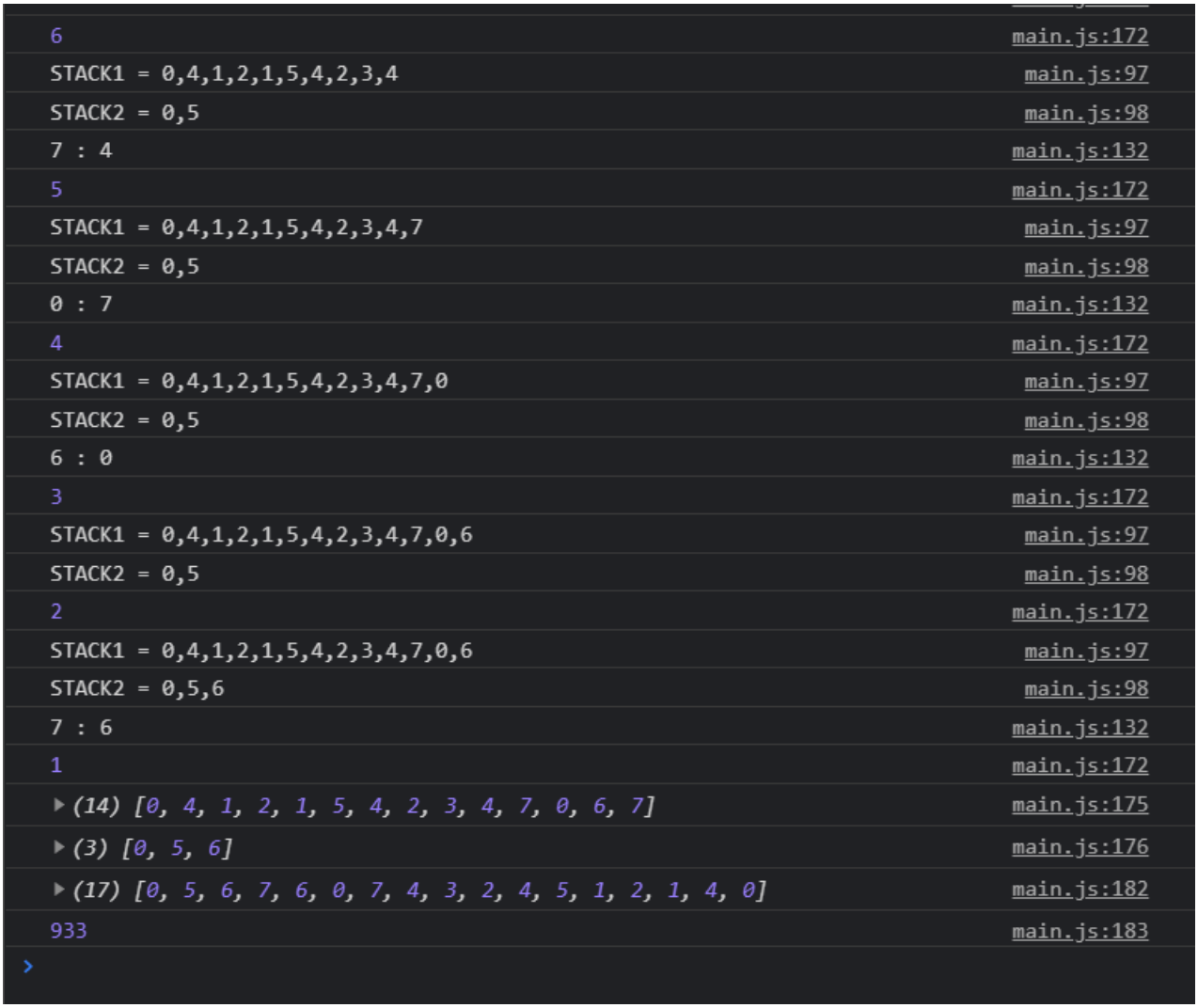

Рис. 2. Результат виконання програми

Отже, результат виконання програми збігається з вище отриманими графічними розрахунками.

У файлі №3, щоб утворити Ейлерів цикл необхідно добавляти ребра, а не тільки дублювати тому моя програма не може його виконати (оскільки такий функціонал не прописаний).

Повна версія коду доступна на GitHub:
https://github.com/RomanDatsyshyn/DiscreteModelsLabs

Висновок: під час виконання цієї лабораторної роботи я ознайомився з алгоритмами рішення задачі листоноші та застосував ці знання на практиці, а саме написав програму для знаходження Ейлерового циклу в графі з дублюванням граней за необхідності.