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 №4
На тему "Потокові алгоритми"

Мета роботи: Метою даної лабораторної роботи є вивчення потокових алгоритмів.

Короткі теоретичні відомості:

Алгоритм пошуку максимального потоку.

Основна задача алгоритму пошуку максимального потоку полягає в пошуку способів пересилання максимальної кількості одиниць потоку з витоку в стік при умові відсутності перевищення пропускних здатностей дуг вихідного графа. В основі алгоритму пошуку максимального потоку лежить наступна ідея: вибираємо початковий потік з витоку "s" в стік "t", потім використовуємо алгоритм пошуку збільшуючого ланцюга. Цей алгоритм дозволяє знайти єдиний збільшуючий ланцюг з "s" в "t", якщо той існує. Послідовність, в якій повинні розфарбовуватись вершини і дуги, конкретно не визначається. Тому можливі два варіанти розфарбування вершин і дуг: 1) вибирається яка-небудь вершина, а потім проводиться розфарбування максимальної кількості вершин; 2) проводиться розфарбування, виходячи з останньої розфарбованої вершини. Який саме спосіб розфарбування буде вибраний, буде залежати від характеру більш загальної задачі, тобто від алгоритму пошуку максимального потоку, який в якості підалгоритму використовує алгоритм пошуку збільшуючого ланцюга. Якщо пошук збільшуючого ланцюга вдалий, тобто знайдено збільшуючий ланцюг з "s" в "t", то за допомогою алгоритму пошуку максимального потоку здійснюється максимально можливе збільшення потоку вздовж знайденого ланцюга. Потім повторюється пошук нового збільшуючого ланцюга і т.д. Виконання алгоритму завершується за скінчене число кроків, коли ланцюг, що збільшує потік, знайти не вдається: це означає, що біжучий потік з "s" в "t" є максимальним.

Індивідуальне завдання:

1. Отримати у викладача індивідуальне завдання.
2. Підготувати програму для вирішення виданого завдання.
3. Запустити на виконання програму, що вирішує задачу пошуку збільшуючого ланцюга.
4. Проглянути результат роботи програми. Результат роботи програми, що шукає збільшуючий ланцюг, може бути позитивний (стік є розфарбований) або негативний (стік не вдається розфарбувати).
5. У випадку, коли результат позитивний (або негативний) необхідно модифікувати граф (коректуючи два або три зв'язки), що дозволить знайти такий граф, на якому стік, відповідно, не вдається розфарбувати (розфарбовується).
6. Запустити на виконання програму, що вирішує задачу пошуку максимального потоку.
7. Проглянути результат роботи програми. Вибрати маршрут, який має максимальне збільшення потоку, за зразком, описаним в інструкції для роботи з учбовою програмою.
8. Зафіксувати результати роботи.
9. Оформити і захистити звіт.

Вхідні дані (файл 1):
8
0 20 20 20 0 0 0 0
0 0 0 0 30 0 0 0
0 10 0 0 0 10 20 0
0 0 0 0 0 15 0 0
0 0 10 0 0 10 0 20
0 0 0 0 0 0 10 20
0 0 0 10 0 0 0 20
0 0 0 0 0 0 0 0

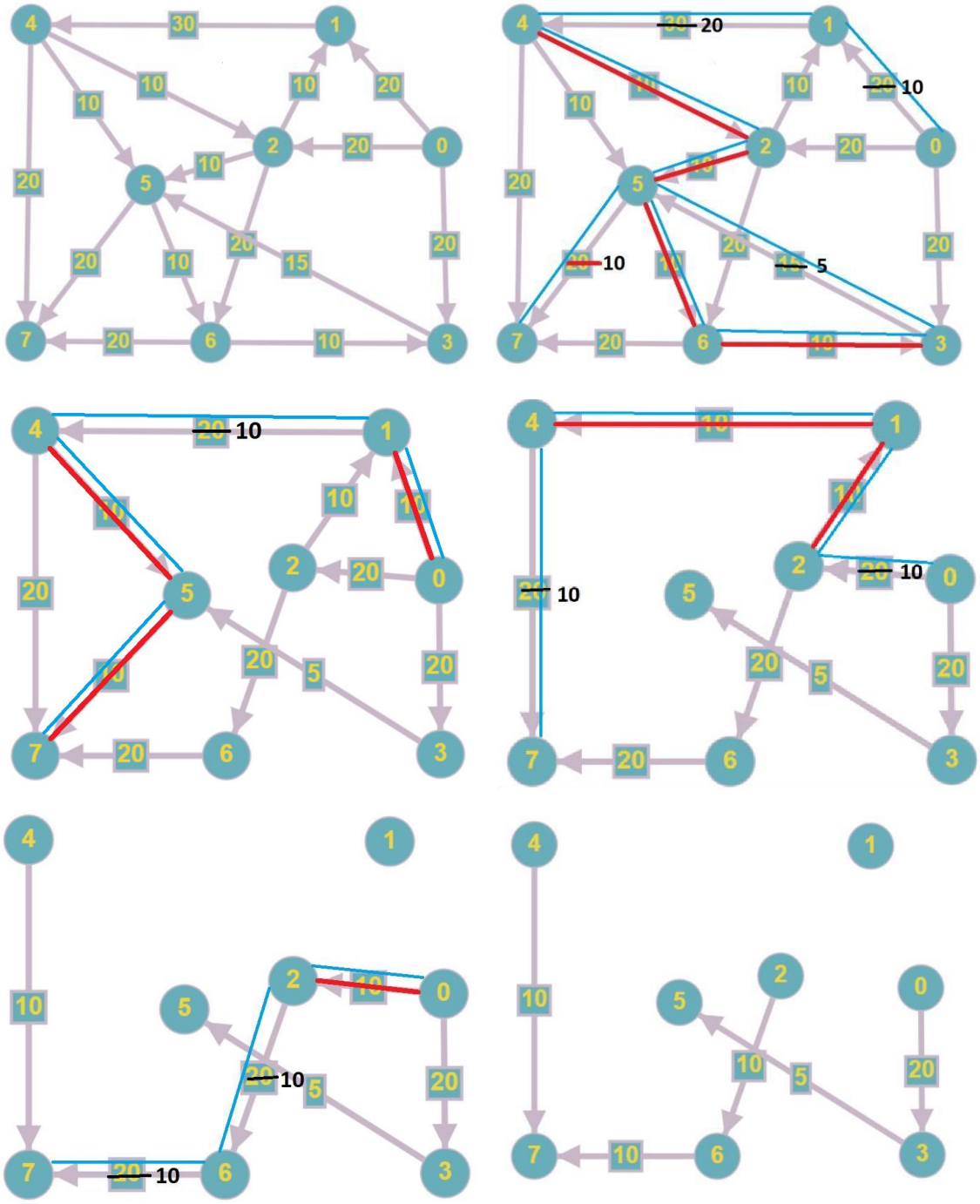

Рис. 1. Графічний розв’язок

Було написано програму для вирішення задачі пошуку максимального потоку методом Форда — Фалкерсона. 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;
  fordFulkerson(m);
};

const fordFulkerson = (iArr) => {
  let arr = iArr.map(function (arr1) {
    return arr1.slice();
  });

  let maxFlow = [];
  let paths = [];
  let limit = 0;
  while (pathFinder(arr) && limit < 25) {
    limit++;

    let path = pathFinder(arr);
    console.log(path);

    let minEdge = {
      coordinates: [],
      value: Infinity,
      i: 0,
    };

    for (let i = 0; i < path.length; i++) {
      if (arr[path[i][0]][path[i][1]] < minEdge.value) {
        minEdge.coordinates = [path[i][0], path[i][1]];
        minEdge.value = arr[path[i][0]][path[i][1]];
        minEdge.i = i;
      }
    }

    console.log(minEdge);

    for (let i = 0; i < path.length; i++) {
      if (i == minEdge.i) arr[path[i][0]][path[i][1]] = 0;
      else arr[path[i][0]][path[i][1]] -= minEdge.value;
    }

    maxFlow.push(minEdge.value);
    paths.push(path);
  }

  console.log(...maxFlow);
  console.log(paths);
};
```

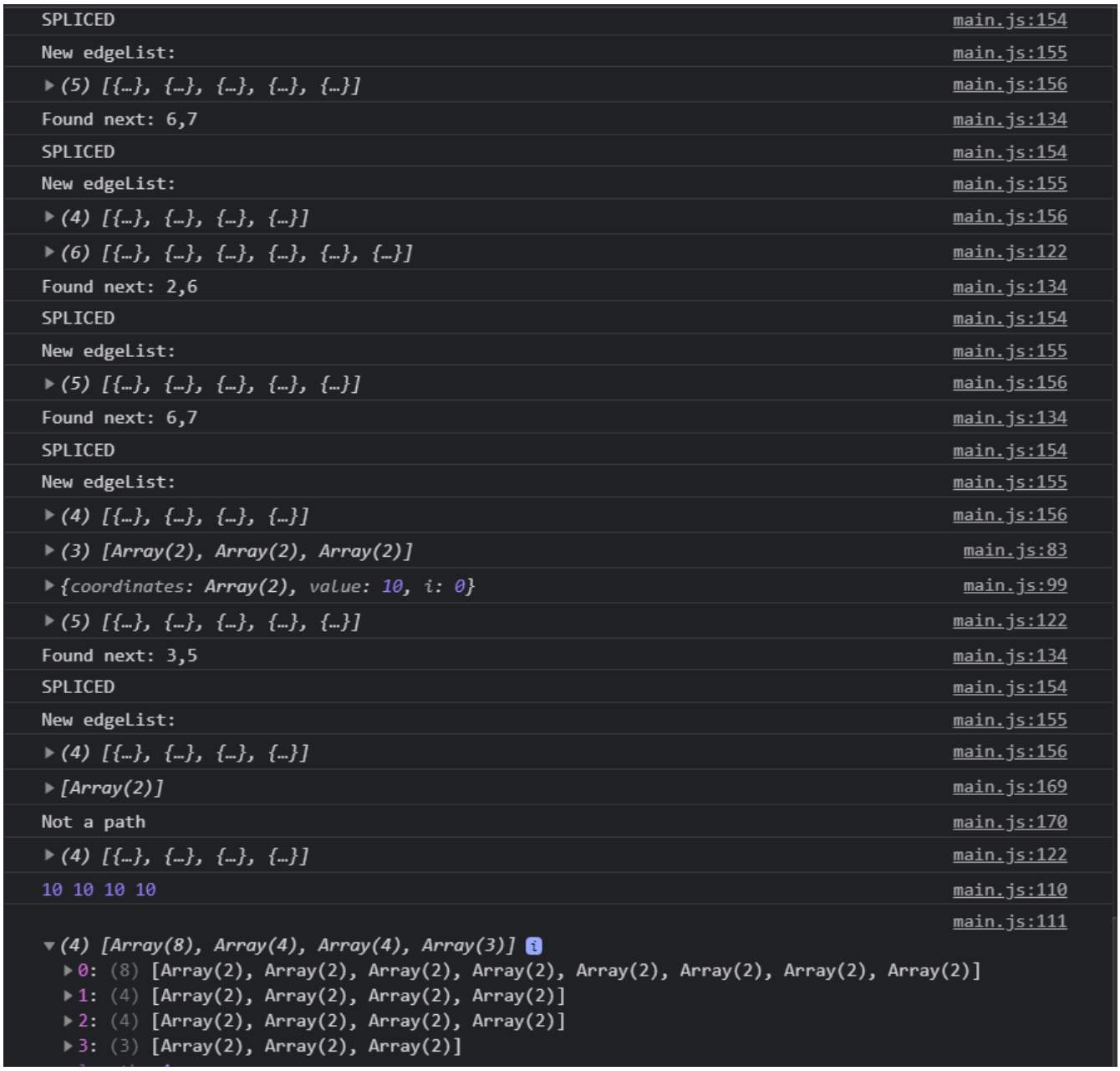

Рис. 2. Результат виконання програми

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

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

Висновок: під час виконання цієї лабораторної роботи я ознайомився з потоковими алгоритмами та застосував отримані знання на практиці, а саме програмно реалізував алгоритм пошуку максимального потоку методом Форда — Фалкерсона та порівняв результат програми з графічними обрахунками.