Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №3

з дисципліни «СРМ-1. Дискретна математика»

на тему

«Операції над матрицями графів»

Виконав:
студент гр. ІС-12
Фундерат Денис

Викладач:
доц. Рибачук Л.В.

Київ – 2021

Зміст

1 Постановка задачі

Реалізувати програмне застосування (програму), яке виконує наступні функції. Причому на вхід програми подається вхідний файл з описом графу, зі структурою, яка вказана у практичному завданні No1 «Представлення графів».

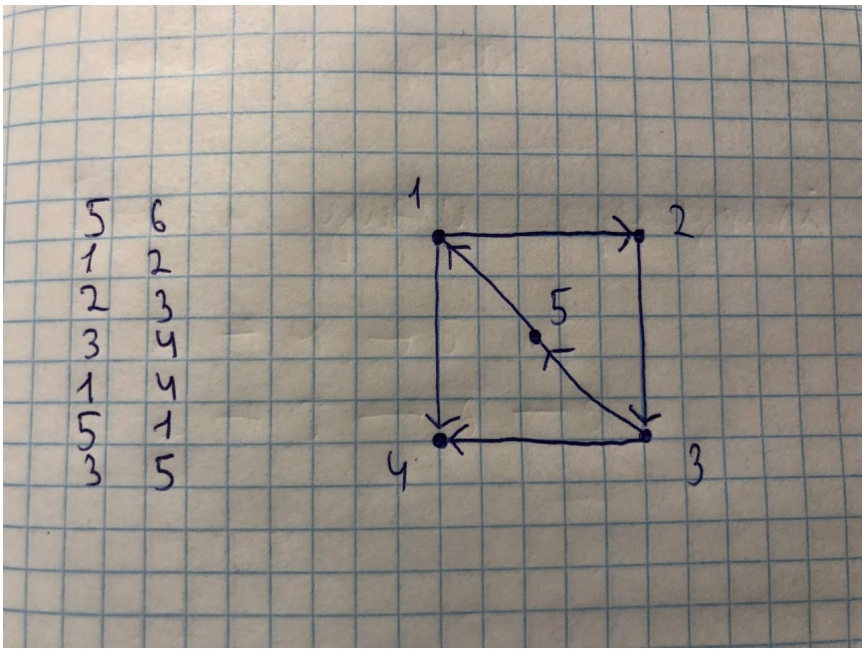
1. Визначити матриці відстаней та досяжності графу. Програма за запитом користувача виводить на екран та/або у файл матрицю відстаней *D* та матрицю досяжності *R* графу.

2. Визначити наявність простих циклів у графі. Програма визначає чи наявні у графі прості цикли та, в разі позитивної відповіді, виводить деякі цикли на екран.

3. Визначити тип зв’язності графу. Програма виводить на екран тип зв’язності графу.

2. Результати виконання програми

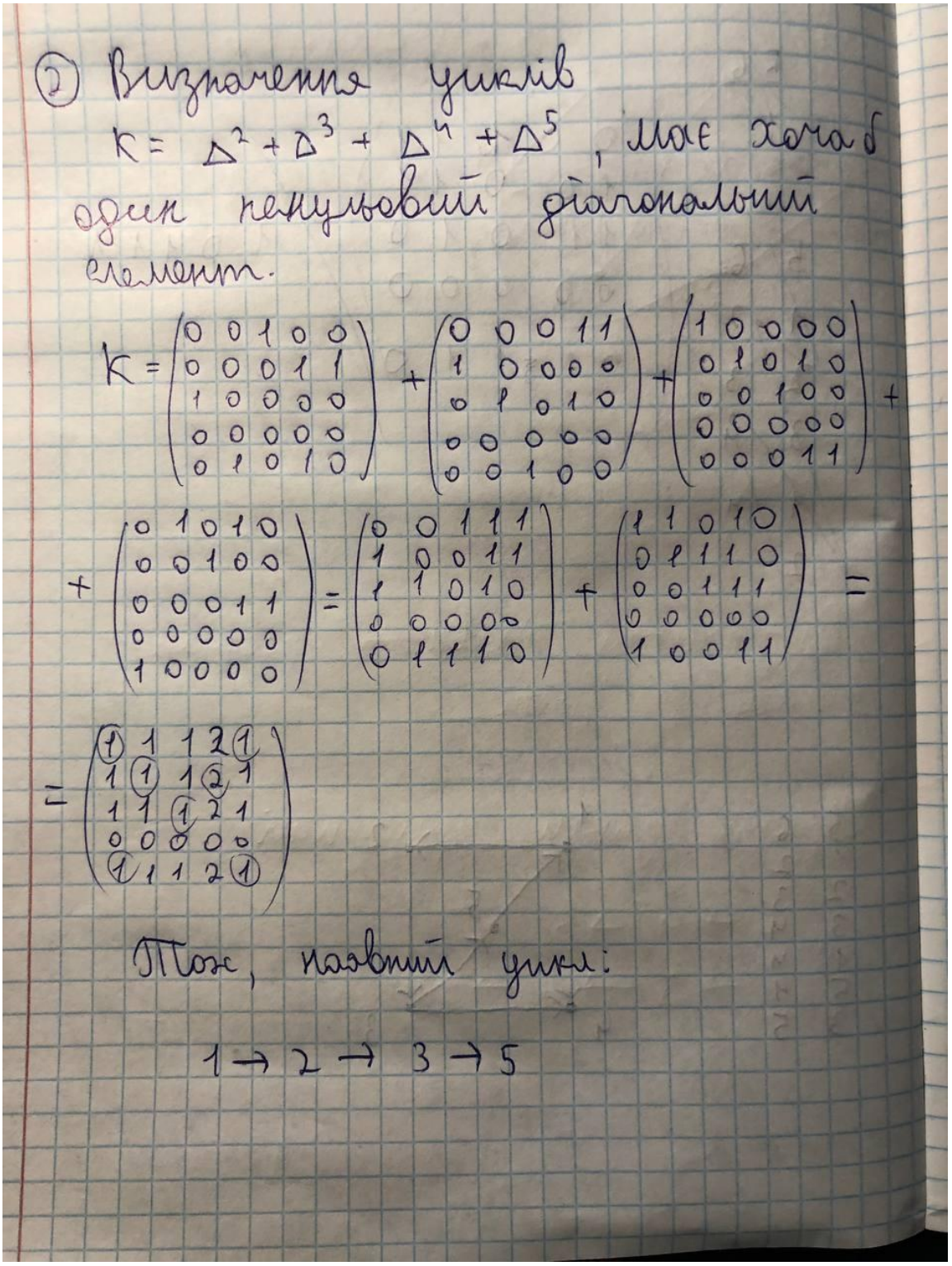
Заданий граф:



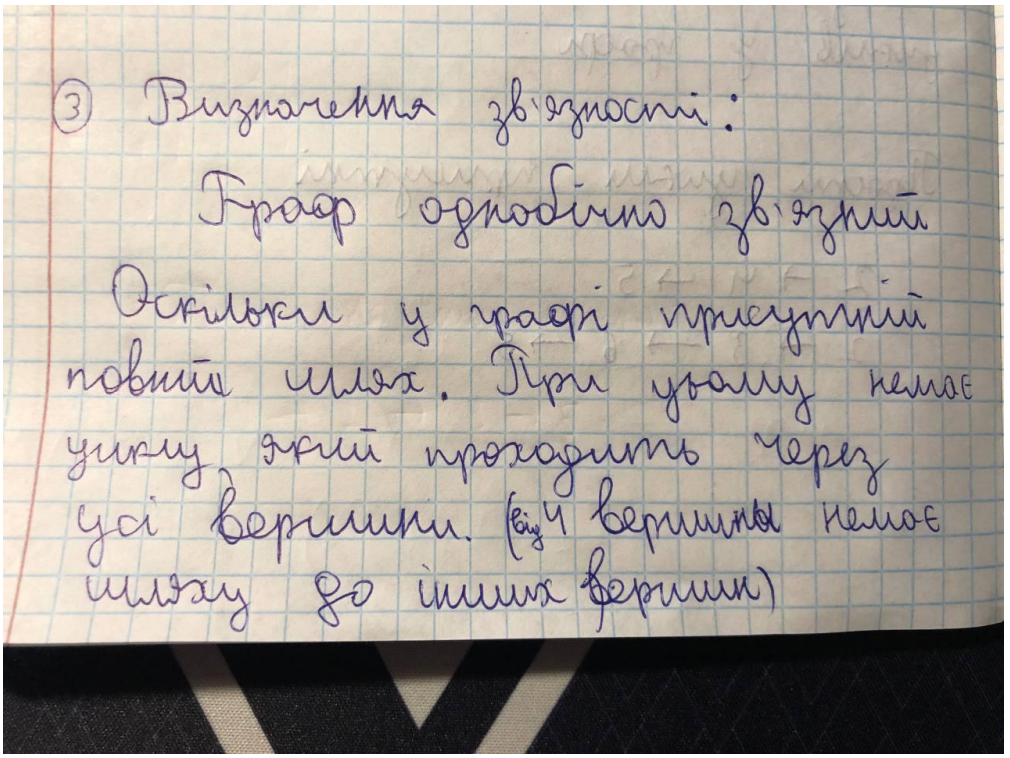
Матриця відстаней та досяжності:

```
матриця суміжності:                                          script.js:30
                                                             script.js:31
▼(5) [Array(5), Array(5), Array(5), Array(5), Array(5)] i
  ▶0: (5) [0, 1, 0, 1, 0]
  ▶1: (5) [0, 0, 1, 0, 0]
  ▶2: (5) [0, 0, 0, 1, 1]
  ▶3: (5) [0, 0, 0, 0, 0]
  ▶4: (5) [1, 0, 0, 0, 0]
    length: 5
  ▶[[Prototype]]: Array(0)
матриця відстані:                                            script.js:89
 (-1 - немає ніякого зв'язку між цими двома вершинами)
                                                             script.js:90
▼(5) [Array(5), Array(5), Array(5), Array(5), Array(5)] i
  ▶0: (5) [0, 1, 2, 1, 3]
  ▶1: (5) [3, 0, 1, 2, 2]
  ▶2: (5) [2, 3, 0, 1, 1]
  ▶3: (5) [-1, -1, -1, 0, -1]
  ▶4: (5) [1, 2, 3, 2, 0]
    length: 5
  ▶[[Prototype]]: Array(0)
матриця досяжності:                                          script.js:153
                                                             script.js:154
▼(5) [Array(5), Array(5), Array(5), Array(5), Array(5)] i
  ▶0: (5) [0, 1, 1, 1, 0]
  ▶1: (5) [0, 0, 1, 1, 1]
  ▶2: (5) [1, 0, 0, 1, 1]
  ▶3: (5) [0, 0, 0, 0, 0]
  ▶4: (5) [1, 1, 0, 1, 0]
    length: 5
  ▶[[Prototype]]: Array(0)
>
```

Цикли:


② Визначення циклів
K = Δ² + Δ³ + Δ⁴ + Δ⁵, має хоча б один ненульовий діагональний елемент.
Тож, наявні цикли:
1 → 2 → 3 → 5

Тип зв’язності:



3 Лістинг програми

```
const n = graph[0];
const m = graph[1];
const n_column = [];
const m_column = [];

for (let i = 2; i < graph.length; i++) {
   if (i % 2 == 0) n_column.push(graph[i]);
   else m_column.push(graph[i]);
}




// ============ матриця суміжності ============
const adjMatrix = []; // adjacency matrix
let adjRow = [];

for (let i = 1; i <= n; i++) { // ініціалізація матриці (only with 0)
   adjRow = [];
   for (let j = 1; j <= n; j++) {
```

```
            adjRow.push(0);
        }
        adjMatrix.push(adjRow);
    }


    for (let i = 0; i < n_column.length; i++) { // підстановка 1
        adjMatrix[n_column[i]-1][m_column[i]-1] = 1;
    }

    console.log(`матриця суміжності:`);
    console.log(adjMatrix);
    // ========================




    // ============ матриця відстаней ============
    const distMatrix = [];
    let distMatrix_row = [];

    for (let i = 0; i < n; i++) {
        distMatrix_row = [];
        for (let j = 0; j < n; j++) {
            if (i == j) distMatrix_row.push(0);
            else {
                if (adjMatrix[i][j] == 1) distMatrix_row.push(adjMatrix[i][j]);
                if (adjMatrix[i][j] == 0) distMatrix_row.push(null);
            }
        }
        distMatrix.push(distMatrix_row);
    }


    let expMatrix_forDist = [];
    let iter = 1;

    function distMatrixRender() {
        let ckeckOut_main = false;
        let ckeckOut_matrixExpon = false;
```

```
        iter++;

        for (let i = 0; i < n; i++) {
            for (let j = 0; j < n; j++) {
                if (distMatrix[i][j] === null) {
                    if (ckeckOut_matrixExpon == false) {
                        matrixExpon(expMatrix_forDist);
                        ckeckOut_matrixExpon = true;
                    }

                    if (expMatrix_forDist[i][j] != 0) distMatrix[i][j] = iter;
                    else if (expMatrix_forDist[i][j] == 0) {
                        if (iter == 40) distMatrix[i][j] = 0;
                        ckeckOut_main = true;
                    }
                }
            }
        }

        if (ckeckOut_main == true) distMatrixRender();
        else {
            console.log(`матриця відстані:`);
            console.log(distMatrix);
        }
    }
    distMatrixRender();



    let expMatrix_forReach = [];

    function matrixExpon(matrix) {
        let expMatrix_row = [];
        let expMatrix_num = 0;
        let ckeckOut = String(matrix);

        for (let i = 0; i < n; i++) {
            expMatrix_row = [];
            for (let k = 0; k < n; k++) {
                expMatrix_num = 0;
```

```
                for (let j = 0; j < n; j++) {
                            if (ckeckOut != '') expMatrix_num += matrix[i][j] *
adjMatrix[j][k];
                    else expMatrix_num += adjMatrix[i][j] * adjMatrix[j][k];
                }
                expMatrix_row.push(expMatrix_num);
            }
            matrix[i] = expMatrix_row;
        }

        // for (let elem of matrix) {
        //     console.log(elem);
        // }
        // console.log(`---`);
    }
    // ========================



    // ============ матриця досяжності ============
    const reachMatrix = [];
    let reachMatrix_row = [];

    for (let i = 0; i < n; i++) {
        reachMatrix_row = [];
        for (let j = 0; j < n; j++) {
            reachMatrix_row.push(adjMatrix[i][j]);
        }
        reachMatrix.push(reachMatrix_row);
    }



    let ckeckOut_matrixExpon = false;

    for (let i = 0; i < n; i++) {
        for (let j = 0; j < n; j++) {
            if (ckeckOut_matrixExpon == false) {
                matrixExpon(expMatrix_forReach);
                ckeckOut_matrixExpon = true;
```

```
            }
            if (expMatrix_forReach[i][j] == 1 && reachMatrix[i][j] == 0) {
                reachMatrix[i][j] = 1;
            }
        }
    }
    console.log(`матриця досяжності:`);
    console.log(reachMatrix);
    // ========================
```