ЗВІТ

про виконання лабораторної роботи № 6

Представлення графів

з дисципліни «Програмування дискретних структур»

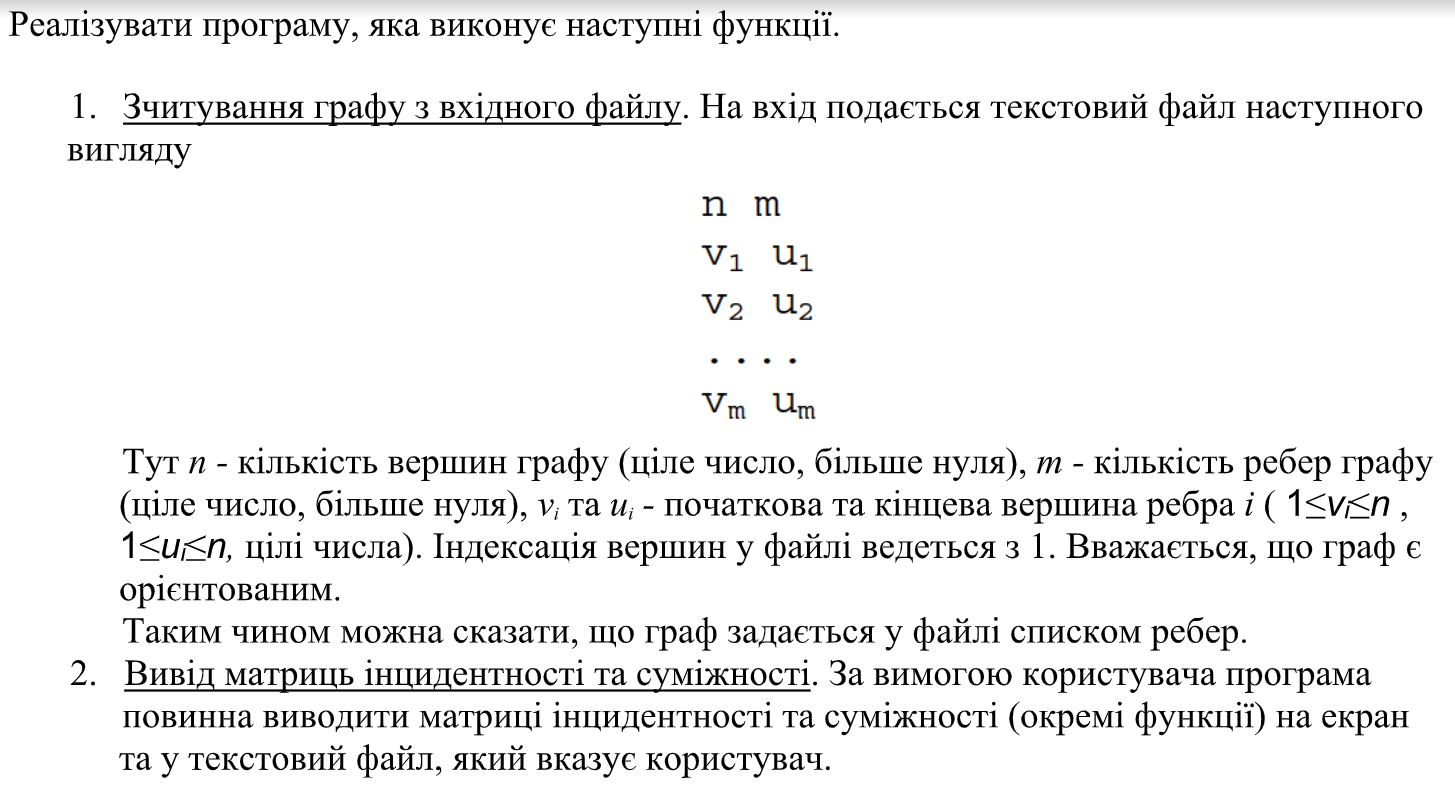
студента групи КН-2226Б

Медвідь Дмитро Степанович

Варіант 11

Завдання №1

Умова завдання;



Опис алгоритму (блок-схема або структурна схема);

Текст програми;

const fs = require('fs');

function readGraphFromFile(filename) {

    try {

        const data = fs.readFileSync(filename, 'utf8');

        const lines = data.split('\n');

        const [n, m] = lines[0].split(' ').map(Number);

        const edges = {};

        for (let i = 1; i <= m; i++) {

            const [vi, ui] = lines[i].split(' ').map(Number);

            if (!edges[vi]) {

                edges[vi] = [];

            }

            if (!edges[ui]) {

                edges[ui] = [];

            }

            edges[vi].push(ui);

        }

        return { n, m, edges };

    } catch (error) {

        console.error('Помилка при читанні файлу:', error.message);

        return null;

    }

}

function IncidenceMatrix(graph) {

    const incidenceMatrix = [];

    for (let i = 1; i <= graph.n; i++) {

        const row = [];

        for (let j = 1; j <= graph.m; j++) {

            const edge = graph.edges[i].includes(j) ? 1 : 0;

            row.push(edge);

        }

        incidenceMatrix.push(row);

    }

    return incidenceMatrix;

}

function AdjacencyMatrix(graph) {

    const adjacencyMatrix = [];

    for (let i = 1; i <= graph.n; i++) {

        const row = [];

        for (let j = 1; j <= graph.n; j++) {

            const connected = graph.edges[i].includes(j) ? 1 : 0;

            row.push(connected);

        }

        adjacencyMatrix.push(row);

    }

    return adjacencyMatrix;

}

function writeMatrixToFile(matrix, filename) {

    const data = matrix.map(row => row.join(' ')).join('\n');

    fs.writeFileSync(filename, data, 'utf8');

    console.log(`Матрицю записано у файл: ${filename}`);

}

const graph = readGraphFromFile("./lab6/lb6.txt");

console.log(graph);

for (let i = 1; i <= graph.n; i++) {

    console.log(`Edges for vertex ${i}: ${graph.edges[i].join(', ')}`);

}

if (graph) {

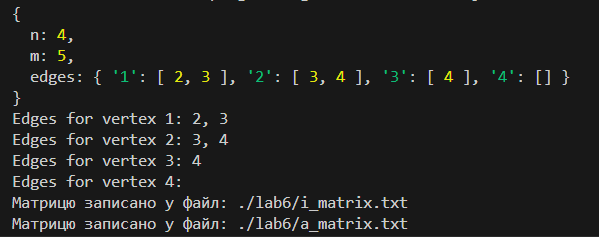
    writeMatrixToFile(IncidenceMatrix(graph), "./lab6/i\_matrix.txt");

    writeMatrixToFile(AdjacencyMatrix(graph), "./lab6/a\_matrix.txt");

}

module.exports = { AdjacencyMatrix };

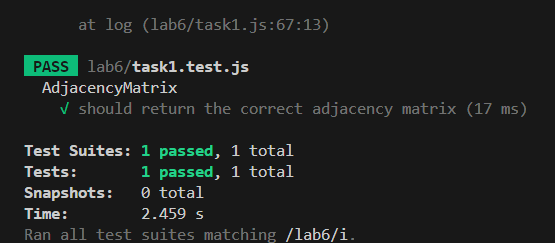
Значення вхідних даних та результати виконання програми;



Посилання на публічний репозиторій GitHub з кодом програми;

<https://github.com/DartVeyder/Discrete-programming/blob/main/lab6/task1.js>

Результати unit-тесту



Висновок

Я розглянув та вивчив різні типи представлення графів в памяті компютерів.