ЗВІТ

про виконання лабораторної роботи № 6

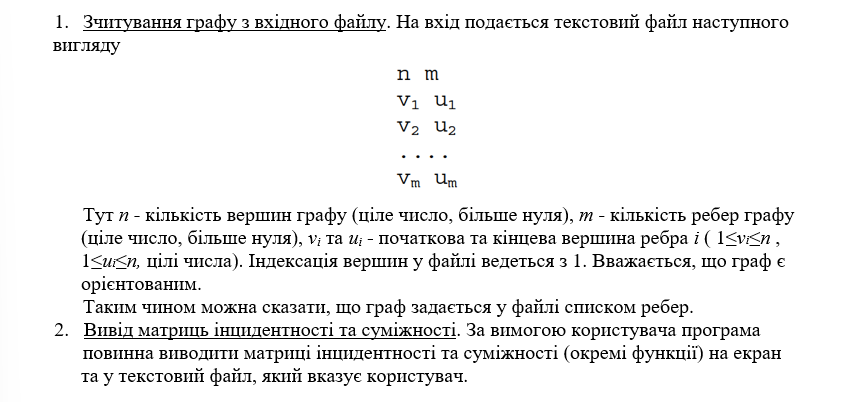
*Представлення графів*

з дисципліни «Програмування дискретних структур»

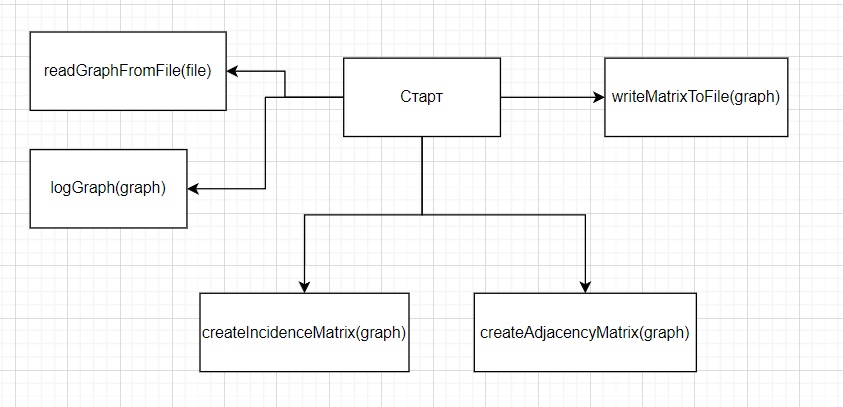
студента групи КН-2226

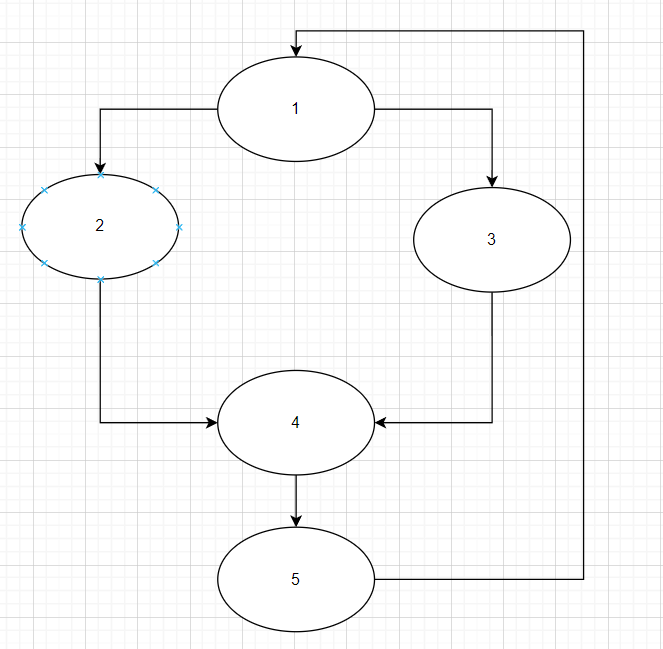
*Веждел Андрій Іванович*

**Умова завдання:**



**Блок схема:**

****

**Граф:**

**Текст коду:**

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 list = {};

    for (let i = 1; i < lines.length; i++) {

      let [v, u] = lines[i].split(" ").map(Number);

      if (!u || u === undefined) {

        u = [];

      }

      if (!list[v]) {

        list[v] = [];

      }

      list[v].push(u);

    }

    return { n, m, list };

  } catch (error) {

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

    return null;

  }

}

function logGraph(graph) {

  const list = graph.list;

  for (const key in list) {

    const neighbors = list[key];

    neighbors.forEach((neighbor) => {

      console.log(` ${key} --> ${neighbor}`);

    });

  }

}

function createIncidenceMatrix(graph) {

    const { n,m, list } = graph;

    const incidenceMatrix = Array.from({ length: n }, () => Array(m).fill(0));

    let edgeIndex = 0;

    for (const key in list) {

      const rebra = list[key];

      for (let j = 0; j < rebra.length; j++) {

        const connectedkey = rebra[j];

        incidenceMatrix[key - 1][edgeIndex] = -1;

        incidenceMatrix[connectedkey - 1][edgeIndex] = 1;

        edgeIndex++;

      }

    }

    return incidenceMatrix;

  }

function createAdjacencyMatrix(graph) {

  const { n, list } = graph;

  const adjacencyMatrix = [];

  for (let i = 1; i <= n; i++) {

    const row = [];

    for (let j = 1; j <= n; j++) {

      if (list[i] && list[i].includes(j)) {

        row.push(1);

      } else {

        row.push(0);

      }

    }

    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("./lb6/lb6.txt");

logGraph(graph);

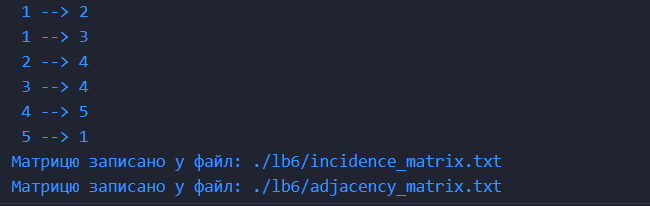
const incidenceMatrix = createIncidenceMatrix(graph);

const adjacencyMatrix = createAdjacencyMatrix(graph);

writeMatrixToFile(incidenceMatrix, "./lb6/incidence\_matrix.txt");

writeMatrixToFile(adjacencyMatrix, "./lb6/adjacency\_matrix.txt");

**Результат виконання:**

****

**Посилання на гіт-репозиторій:**

[disc-systems/lb6.js at main · 1duxa/disc-systems (github.com)](https://github.com/1duxa/disc-systems/blob/main/lb2.js)

**Результат юніт-тесту:**

const lb6 = require("./lb6");

test("Матриця інцидентності", () => {

    const incidenceMatrix = [[-1,-1,0,0,0,1],[1,0,-1,0,0,0],[0,1,0,-1,0,0],[0,0,1,1,-1,0],[0,0,0,0,1,-1]];

    const graph = {

        n: 5,

        m: 6,

        list: {

            "1": ["2", "3"],

            "2": ["4"],

            "3": ["4"],

            "4": ["5"],

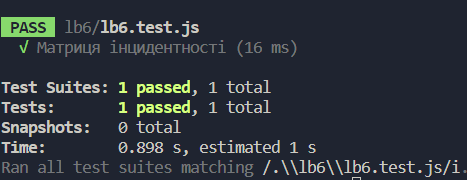
            "5": ["1"],

        },

    };

    expect(lb6.createIncidenceMatrix(graph)).toEqual(incidenceMatrix);

});

****

**Висновок:**

Розглянув та вивчити різні типи представлення графів в пам'яті комп'ютерів.