Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти і газу

Кафедра ІПЗ

**Лабораторна робота №6**

**Тема: “Графи”**

Виконав

ст.гр. ІП-22-1

Токарєв С.О.

Перевірив:

Григорчук Л.І.

Івано-Франківськ

2023

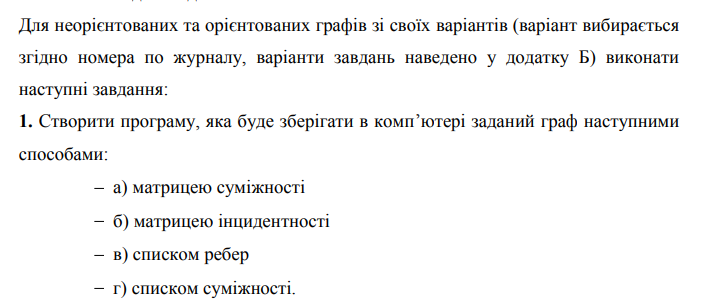
**Теоретичні відомості:**

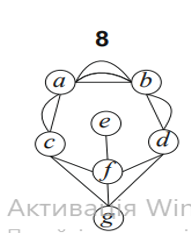
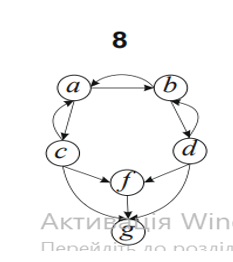
1. **Матриця суміжності:**
   * *Визначення:* Матриця суміжності є квадратною матрицею, в якій визначається наявність або відсутність ребра між вершинами графа.
   * *Застосування:* Цей метод дозволяє легко визначити, чи існує ребро між конкретними вершинами та отримувати загальний погляд на структуру графа.
2. **Матриця інцидентності:**
   * *Визначення:* Матриця інцидентності визначається так, що рядок представляє вершину, а стовпець ребро. Значення 1 вказує, що вершина інцидентна ребру, а -1, що вершина є кінцем ребра.
   * *Застосування:* Цей метод дозволяє легко визначати структуру ребер графа та їхній зв'язок з вершинами.
3. **Список ребер:**
   * *Визначення:* Список ребер є простим переліком усіх ребер у графі, де кожне ребро представлено парою вершин, які його з'єднують.
   * *Застосування:* Цей метод легко вказує, які ребра існують у графі та які вершини вони з'єднують.
4. **Список суміжності:**
   * *Визначення:* Список суміжності - це структура даних, в якій для кожної вершини графа вказано список вершин, які є її сусідами (з якими є ребра).
   * *Застосування:* Цей метод надає важливу інформацію про сусідство вершин та полегшує виконання операцій, пов'язаних із знаходженням суміжних вершин.

Мета:

Набуття практичних вмінь і навичок при представленні заданих графів різними способами та можливістю їх комп'ютерної реалізації.

Завдання



Graph.java:  
package lab6;

import java.util.Scanner;

import java.util.Stack;

public class Graph {

private int MAX\_VERTS;//макс к-сть вершин

private int MAX\_E;//макс к-сть вершин

private int nEdges = 0;//спочатку 0 ребер

private Vertex vertexArray[]; //масив вершин

private int adjMat[][]; // матриця суміжності

private int incidenceMat[][]; // матриця інцидентності

private int nVerts; // поточна к сть вершин

private Stack<Integer> stack;

public Graph(int qtV, int qtE) { // ініціалізація внутрішніх полів

MAX\_VERTS = qtV;

MAX\_E = qtE;

vertexArray = new Vertex[qtV];

adjMat = new int[qtV][qtV];

incidenceMat = new int[qtV][qtE];

nVerts = 0;

for (int j = 0; j < qtV; j++) {

for (int k = 0; k < qtV; k++) {

adjMat[j][k] = 0;

}

}

for (int j = 0; j < qtV; j++) {

for (int k = 0; k < qtE; k++) {

incidenceMat[j][k] = 0;

}

}

stack = new Stack<>();

}

public void addVertex(char lab) {

vertexArray[nVerts++] = new Vertex(lab);

}

public void addEdge(int start, int end) {

adjMat[start][end] = 1;

adjMat[end][start] = 1;

incidenceMat[end][nEdges] = 1;

incidenceMat[start][nEdges] = 1;

nEdges++;

}

public void displayVertex(int v) {

System.out.println(vertexArray[v].getLabel());

}

public void printAdjMatrix() {

for (int j = 0; j < MAX\_VERTS; j++) {

for (int k = 0; k < MAX\_VERTS; k++) {

System.out.print(adjMat[j][k] + " ");

}

System.out.println();

}

}

public void printIncMatrix() {

for (int j = 0; j < MAX\_VERTS; j++) {

for (int k = 0; k < MAX\_E; k++) {

System.out.print(incidenceMat[j][k] + " ");

}

System.out.println();

}

}

public void printEdgesList() {

for (int j = 0; j < nEdges; j++) {

for (int i = 0; i < nVerts; i++) {

if (incidenceMat[i][j] == 1) {

System.out.print(vertexArray[i].getLabel() + " ");

}

}

System.out.println("");

}

}

public void printAdjList() {

for (int j = 0; j < MAX\_VERTS; j++) {

for (int k = 0; k < MAX\_VERTS; k++) {

if (adjMat[j][k] == 1) {

System.out.print("(" + j + "; " + k + "), ");

}

}

}

}

public void createIncUsingAdj() {

Scanner s = new Scanner(System.in);

System.out.println("Enter the adjacency matrix: ");

for (int i = 0; i < MAX\_VERTS; i++) {

for (int j = 0; j < MAX\_VERTS; j++) {

adjMat[i][j] = s.nextInt();

if (j <= i) {

if (adjMat[i][j] == 1) {

incidenceMat[i][nEdges] = 1;

incidenceMat[j][nEdges] = 1;

nEdges++;

}

}

}

}

}

public void createAdjListUsingAdjM() {

Scanner s = new Scanner(System.in);

System.out.println("enter the incidence matrix: ");

for (int i = 0; i < MAX\_VERTS; i++) {

for (int j = 0; j < MAX\_VERTS; j++) {

adjMat[i][j] = s.nextInt();

}

}

}

public void printEdgesListWithoutL() {

for (int j = 0; j < nEdges; j++) {

for (int i = 0; i < MAX\_VERTS; i++) {

if (incidenceMat[i][j] == 1) {

System.out.print( i + " ");

}

}

System.out.println("");

}

}

}

Solution.java

package lab6;

import java.util.Scanner;

public class Solution {

public static void main(String[] args) {

System.***out***.println("Unoriented graph\n");

Graph graph = new Graph(7, 12);

graph.addVertex('a'); //0

graph.addVertex('b'); //1

graph.addVertex('c'); //2

graph.addVertex('d'); //3

graph.addVertex('e'); //4

graph.addVertex('f'); //5

graph.addVertex('g'); //6

graph.addEdge(0, 1);

graph.addEdge(0, 1);

graph.addEdge(0, 1);

graph.addEdge(0, 2);

graph.addEdge(0, 2);

graph.addEdge(1, 3);

graph.addEdge(1, 3);

graph.addEdge(2, 5);

graph.addEdge(2, 6);

graph.addEdge(3, 5);

graph.addEdge(3, 6);

graph.addEdge(4 , 5);

System.***out***.println("Adjacency matrix");

graph.printAdjMatrix();

System.***out***.println("Incedence matrix");

graph.printIncMatrix();

System.***out***.println("Edge list");

graph.printEdgesList();

System.***out***.println("Adjacency list");

graph.printAdjList();

System.***out***.println("\n\nOriented graph\n");

OrientedGraph orGraph = new OrientedGraph(6, 11);

orGraph.addVertex('a'); //0

orGraph.addVertex('b'); //1

orGraph.addVertex('c'); //2

orGraph.addVertex('d'); //3

orGraph.addVertex('f'); //5

orGraph.addVertex('g'); //6

orGraph.addEdge(0, 1);

orGraph.addEdge(0, 2);

orGraph.addEdge(1, 0);

orGraph.addEdge(2, 0);

orGraph.addEdge(1, 3);

orGraph.addEdge(3, 1);

orGraph.addEdge(3, 4);

orGraph.addEdge(3, 5);

orGraph.addEdge(2, 4);

orGraph.addEdge(2, 5);

orGraph.addEdge(4, 5);

System.***out***.println("Adjacency matrix");

orGraph.printAdjMatrix();

System.***out***.println("Incedence matrix");

orGraph.printIncMatrix();

System.***out***.println("Edge list");

orGraph.printEdgesList();

System.***out***.println("Adjacency list");

orGraph.printAdjList();

///за матрицею суміжності побудувати матрицю інцидентності

System.***out***.println("\n\nEnter the vertecies amount: ");

Scanner s = new Scanner(System.***in***);

int v = s.nextInt();

System.***out***.println("\n\nEnter the amount of edges: ");

int e = s.nextInt();

Graph graph2 = new Graph(v,e);

graph2.createIncUsingAdj();

graph2.printAdjMatrix();

System.***out***.println("Incidence matrix:");

graph2.printIncMatrix();

///за матрицею суміжності побудувати список суміжності

System.***out***.println("\n\nEnter the vertecies amount: ");

v = s.nextInt();

Graph graph3 = new Graph(v,e);

graph3.createAdjListUsingAdjM();

System.***out***.println("Adjacency list");

graph3.printAdjList();

//за аматрицую суміжності побудувати список ребер

System.***out***.println("\n\nEnter the vertecies amount: ");

int v2 = s.nextInt();

System.***out***.println("\n\nEnter the amount of edges: ");

int e2 = s.nextInt();

Graph graph4 = new Graph(v2,e2);

graph4.createIncUsingAdj();

System.***out***.println("List of edges");

graph4.printEdgesListWithoutL();

s.close();

}

}

Oriented graph

package lab6;

public class OrientedGraph {

private int MAX\_VERTS;//макс к-сть вершин

private int MAX\_Edges;//макс к-сть ребер

private int nEdges = 0;//спочатку 0 ребер

private Vertex vertexArray[]; //масив вершин

private int adjMat[][]; // матриця суміжності

private int incidenceMat[][]; // матриця інцидентності

private int nVerts; // поточна к сть вершин

public OrientedGraph(int qtV, int qtE) { // ініціалізація внутрішніх полів

MAX\_VERTS = qtV;

MAX\_Edges = qtE;

vertexArray = new Vertex[qtV];

adjMat = new int[qtV][qtV];

incidenceMat = new int[qtV][qtE];

nVerts = 0;

for (int j = 0; j < qtV; j++) {

for (int k = 0; k < qtV; k++) {

adjMat[j][k] = 0;

}

}

for (int j = 0; j < qtV; j++) {

for (int k = 0; k < qtE; k++) {

incidenceMat[j][k] = 0;

}

}

}

public void addVertex(char lab) {

vertexArray[nVerts++] = new Vertex(lab);

}

public void addEdge(int start, int end) {

adjMat[start][end] = 1;

incidenceMat[end][nEdges] = 1;

incidenceMat[start][nEdges] = -1;

nEdges++;

}

public void displayVertex(int v) {

System.***out***.println(vertexArray[v].getLabel());

}

public void printAdjMatrix() {

for (int j = 0; j < MAX\_VERTS; j++) {

for (int k = 0; k < MAX\_VERTS; k++) {

System.***out***.print(adjMat[j][k] + " ");

}

System.***out***.println();

}

}

public void printIncMatrix() {

for (int j = 0; j < MAX\_VERTS; j++) {

for (int k = 0; k < MAX\_Edges; k++) {

System.***out***.print(incidenceMat[j][k] + " ");

}

System.***out***.println();

}

}

public void printEdgesList() {

for (int j = 0; j < nEdges; j++) {

for (int i = 0; i < nVerts; i++) {

if (incidenceMat[i][j] == -1) {

System.***out***.print("Start: " + vertexArray[i].getLabel() + " ");

}

else if (incidenceMat[i][j] == 1) {

System.***out***.print("End: " + vertexArray[i].getLabel() + " ");

}

}

System.***out***.println("");

}

}

public void printAdjList() {

for (int j = 0; j < MAX\_VERTS; j++) {

for (int k = 0; k < MAX\_VERTS; k++) {

if (adjMat[j][k] == 1) {

System.***out***.print("(" + j + "; " + k + "), ");

}

}

}

}

}

Vertex

package lab6;

public class Vertex {

private char label; //назва вершини буквою

public Vertex(final char label) {

this.label = label;

}

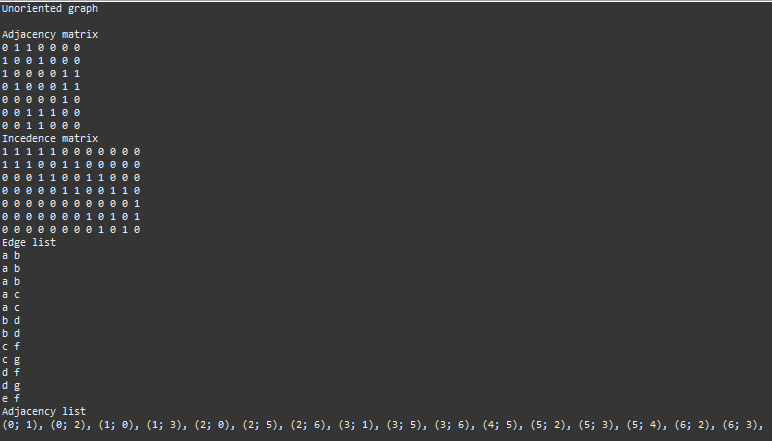
public char getLabel() {

return this.label;

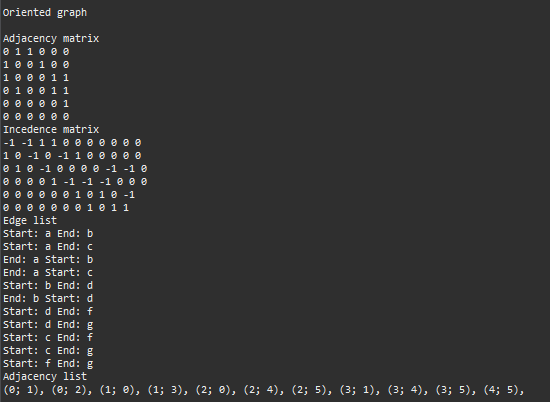
}

}

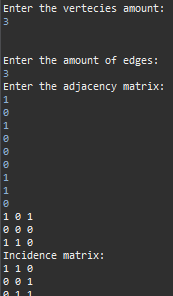
Результати виконання



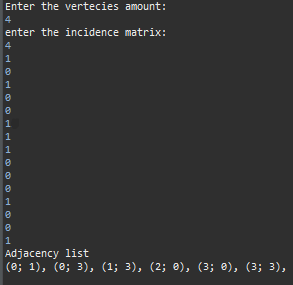
1)Виведення неорієнтованого графа у вигляді мтариці і списку суміжності, матриці інцидентності і списку ребер



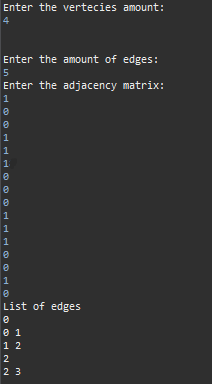
2)Виведення орієнтованого у вигляді мтариці і списку суміжності, матриці інцидентності і списку ребер



за матрицею суміжності побудувати матрицю інцидентності



за матрицею суміжності побудувати список суміжності



за аматрицую суміжності побудувати список ребер

Висновок:

Я навчився працювати з графами, використовуючи засоби мови програмування Java.