

MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII

Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică și Microelectronică

Tehnologia Informației

Raport

Disciplina: Matematica discretă

Lucrarea de laborator nr. 1

Tema: ‘Păstrarea grafului în memoria calculatorului’

Student: _____ **Raevschi Grigore TI-231**

Coordonator: _____ **Ciobanu Ecaterina, asist. univ.**

Chișinău 2024

Cuprinsul

Condiția problemei – Introducere	2
Listing-ul programului.....	3
Rezultatul în consolă	6
Concluzie	8

Condiția problemei – Introducere

Lucrarea de laborator №1 Păstrarea grafului în memoria calculatorului.

Sarcina de bază:

1. De elaborat procedura introducerii unui graf în memoria calculatorului în formă de matrice de incidență, matrice de adiacență și listă de adiacență cu posibilități de analiză a corectitudinii.
2. Elaborați procedura de transformare dintr-o formă de reprezentare în alta.
3. Folosind procedurile menționate, elaborați programul care va permite:
 - a) introducerea grafului reprezentat sub oricare din cele trei forme cu posibilități de corecție a datelor;
 - b) păstrarea grafului în memoria externă în forma de listă de adiacență;
 - c) extragerea informației în una din cele trei forme la imprimantă și display.

La prima lucrare de laborator am elaborat o aplicație în consolă ce permite păstrarea grafului în memoria calculatorului conform teoriei grafurilor:

- Matricea de incidență
- Matricea de adiacență
- Lista de adiacență

Pentru dezvoltarea aplicației am ales limbajul de programare Java, și librăria JgraphT ce facilitează lucrul cu grafurile prin multitudinea de funcții oferite. Programul elaborat permite introducerea nodurilor, afișarea grafului introdus în consolă, afișare lor conform teoriei grafurilor și ștergerea arcelor. Programul este intuitiv pentru utilizator prin faptul că conține un meniu de alegere.

Listing-ul programului

```
import org.jgrapht.Graph;
import org.jgrapht.graph.DefaultEdge;
import org.jgrapht.graph.SimpleGraph;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Graf_reprezentare {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Declararea unui graf de numere intregi
        Graph<Integer, DefaultEdge> graph = new SimpleGraph<>(DefaultEdge.class);

        while (true) {
            printMenu(); // afisare meniu
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    addEdge(graph);
                    break;
                case 2:
                    displayGraph(graph);
                    break;
                case 3:
                    displayIncidenceMatrix(graph);
                    break;
                case 4:
                    displayAdjacencyMatrix(graph);
                    break;
                case 5:
                    displayAdjacencyList(graph);
                    break;
                case 6:
                    deleteEdge(graph);
                    break;
                case 7:
                    deleteVertex(graph);
                    break;
                case 0:
                    System.out.println("STOP program (tasta 0 )!");
                    System.exit(0);
                    break;
                default:
                    System.out.println("Optiune incorecta");
            }
        }

        // Afisare meniu
        private static void printMenu() {
            System.out.println("----- Menu -----");
            System.out.println("1. Adaugare arcuri noi");
            System.out.println("2. Afisare graf");
            System.out.println("3. Afisare matrice de incidenta");
            System.out.println("4. Afisare matrice de adiacenta");
            System.out.println("5. Afisare lista de adiacenta");
            System.out.println("6. Sterge arcuri");
            System.out.println("7. Sterge nod");
            System.out.println("0. Exit");
        }
    }
}
```

```

        System.out.print("Enter optiunea --> ");
    }

    private static void addEdge(Graph<Integer, DefaultEdge> graph) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Dati numarul de noduri ce vrei sa citesti: ");
        int numVertices = scanner.nextInt();

        for (int i = 0; i < numVertices; i++) {
            System.out.print("enter nodul: ");
            int vertex = scanner.nextInt();
            graph.addVertex(vertex);
            System.out.println("nodul adaugat: " + vertex);
        }

        System.out.print("Dati numarul de arcuri ce vrei sa citesti: ");
        int numEdges = scanner.nextInt();

        for (int i = 0; i < numEdges; i++) {
            System.out.print("arcul de inceput: ");
            int source = scanner.nextInt();
            System.out.print("nodul final: ");
            int target = scanner.nextInt();
            graph.addEdge(source, target);
            System.out.println("arcul adaugat cu succes: (" + source + ", " + target
+ ")");
        }
    }

    // Metoda pentru a sterge o legatura dintre noduri
    private static void deleteEdge(Graph<Integer, DefaultEdge> graph) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Introduce arcul initial: ");
        int source = scanner.nextInt(); // declarare nodul sursa
        System.out.print("Introduce arcul final: ");
        int target = scanner.nextInt(); // declarare nodul destinatie
        // verificam daca graful contine arce
        if (graph.containsEdge(source, target)) {
            graph.removeEdge(source, target); // stergem
            System.out.println("Arcul sters: (" + source + ", " + target + ")");
        } else {
            System.out.println("Arcul nu a fost gasit: (" + source + ", " + target +
")");
        }
    }

    // Metoda pentru a sterge o legatura dintre noduri
    private static void deleteVertex(Graph<Integer, DefaultEdge> graph) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Introduce nodul initial: ");
        int source = scanner.nextInt(); // citire nod necesar de sters
        // verificam daca graful contine nodul introdus
        if (graph.containsVertex(source)) {
            graph.removeVertex(source); // stergem nodul
            System.out.println("Nodul sters: (" + source + ")");
        } else {
            System.out.println("Nodul nu a fost gasit: (" + source + ")");
        }
    }

    // Afisare graf
    private static void displayGraph(Graph<Integer, DefaultEdge> graph) {
        System.out.println("Graful: " + graph);
    }

    // Metoda de afisare a matricei de incidenta

```

```

// 1, dacă nodul i este extremitatea finală a arcului j;
// -1, dacă nodul i este extremitatea inițială a arcului j;
// 0, dacă nodul i nu este extremitate a arcului j.
private static void displayIncidenceMatrix(Graph<Integer, DefaultEdge> graph) {
    System.out.println("Matricea de incidenta:");
    // Declaram o lista
    List<DefaultEdge> edges = new ArrayList<>(graph.edgeSet());
    // Parcurgem nodurile grafului
    for (Integer vertex : graph.vertexSet()) {
        // Parcurgem arcurile
        for (DefaultEdge edge : edges) {
            // Declaram nodul sursa si destinatie
            Integer source = graph.getEdgeSource(edge);
            Integer target = graph.getEdgeTarget(edge);
            // Verificam conditia
            if (vertex.equals(source)) {
                System.out.print("-1 ");
            } else if (vertex.equals(target)) {
                System.out.print("1 ");
            } else {
                System.out.print("0 ");
            }
        }
        System.out.println();
    }
}

// Metoda de afisare a matricei de adiacenta
private static void displayAdjacencyMatrix(Graph<Integer, DefaultEdge> graph) {
    System.out.println("Matricea de adiacenta:");
    // Parcurgem fiecare varf din arc
    for (Integer source : graph.vertexSet()) {
        for (Integer target : graph.vertexSet()) {
            // Verificam daca exista un arc intre varful sursa si destinatie
            System.out.print(graph.containsEdge(source, target) ? "1 " : "0 ");
        }
        System.out.println();
    }
}

// Metoda de afisare a listei de adiacenta
private static void displayAdjacencyList(Graph<Integer, DefaultEdge> graph) {
    System.out.println("Lista de adiacenta:");
    // Iterarea dupa noduri
    for (Integer vertex : graph.vertexSet()) {
        System.out.print(vertex + ": ");
        // Declaram o lista pentru stocarea listei
        List<Integer> neighbors = new ArrayList<>();
        // Iterarea dupa arce
        for (DefaultEdge edge : graph.edgesOf(vertex)) {
            // Verificam daca nodul curent este sursa sau destinatia arcului
            Integer neighbor = graph.getEdgeSource(edge).equals(vertex) ?
graph.getEdgeTarget(edge)
                : graph.getEdgeSource(edge);
            // Adaugam elementele in lista
            neighbors.add(neighbor);
        }
        // Afisarea listei in ordine crescatoare
        neighbors.sort(Integer::compareTo);
        for (Integer neighbor : neighbors) {
            System.out.print(neighbor + " ");
        }
        // Afisam 0 la sfarsit
        System.out.print("0");
        System.out.println();
    }
}
}

```

Rezultatul în consolă

1. Introducerea nodurilor , nodurile se introduc în perechi câte 2 , nodul sursă și destinație

```
----- Menu -----
1. Adaugare arcuri noi
2. Afisare graf
3. Afisare matrice de incidenta
4. Afisare matrice de adiacenta
5. Afisare lista de adiacenta
6. Sterge arcuri
7. Sterge nod
0. Exit
Enter optiunea --> 1
Dati numarul de noduri ce vrei sa citesti: 4
enter nodul: 1
nodul adaugat: 1
enter nodul: 2
nodul adaugat: 2
enter nodul: 3
nodul adaugat: 3
enter nodul: 4
nodul adaugat: 4
Dati numarul de arcuri ce vrei sa citesti: 5
arcul de inceput: 1
nodul final: 2
arcul adaugat cu succes: (1, 2)
arcul de inceput: 2
nodul final: 3
arcul adaugat cu succes: (2, 3)
arcul de inceput: 1
nodul final: 4
arcul adaugat cu succes: (1, 4)
arcul de inceput: 1
nodul final: 3
arcul adaugat cu succes: (1, 3)
arcul de inceput: 2
nodul final: 4
arcul adaugat cu succes: (2, 4)
```

Figura 1

2. Afișare grafului

```
----- Menu -----
1. Adaugare arcuri noi
2. Afisare graf
3. Afisare matrice de incidenta
4. Afisare matrice de adiacenta
5. Afisare lista de adiacenta
6. Sterge arcuri
7. Sterge nod
0. Exit
Enter optiunea --> 2
Graful: ([1, 2, 3, 4], [{1,2}, {2,3}, {1,4}, {1,3}, {2,4}])
```

Figura 1-1

3. Afișarea matricei de incidenta

```
----- Menu -----
1. Adaugare arcuri noi
2. Afișare graf
3. Afișare matrice de incidenta
4. Afișare matrice de adiacenta
5. Afișare lista de adiacenta
6. Sterge arcuri
7. Sterge nod
0. Exit
Enter optiunea --> 3
Matricea de incidenta:
-1 0 -1 -1 0
1 -1 0 0 -1
0 1 0 1 0
0 0 1 0 1
```

Figura 1-2

4. Afișarea matricei de adiacenta

```
----- Menu -----
1. Adaugare arcuri noi
2. Afișare graf
3. Afișare matrice de incidenta
4. Afișare matrice de adiacenta
5. Afișare lista de adiacenta
6. Sterge arcuri
7. Sterge nod
0. Exit
Enter optiunea --> 4
Matricea de adiacenta:
0 1 1 1
1 0 1 1
1 1 0 0
1 1 0 0
```

Figura 1-3

5. Afișarea listei de adiacență

```
----- Menu -----
1. Adaugare arcuri noi
2. Afișare graf
3. Afișare matrice de incidenta
4. Afișare matrice de adiacenta
5. Afișare lista de adiacenta
6. Sterge arcuri
7. Sterge nod
0. Exit
Enter optiunea --> 5
Lista de adiacenta:
1: 2 3 4 0
2: 1 3 4 0
3: 1 2 0
4: 1 2 0
```

Figura 1-4

6. Ștergere arcuri

```
----- Menu -----
1. Adaugare arcuri noi
2. Afisare graf
3. Afisare matrice de incidenta
4. Afisare matrice de adiacenta
5. Afisare lista de adiacenta
6. Sterge arcuri
7. Sterge nod
0. Exit
Enter optiunea --> 6
Introduce arcul initial: 1
Introduce arcul final: 3
Arcul sters: (1, 3)
```

Figura 1-5

```
----- Menu -----
1. Adaugare arcuri noi
2. Afisare graf
3. Afisare matrice de incidenta
4. Afisare matrice de adiacenta
5. Afisare lista de adiacenta
6. Sterge arcuri
7. Sterge nod
0. Exit
Enter optiunea --> 2
Graful: ([1, 2, 3, 4], [{1,2}, {2,3}, {1,4}, {2,4}])
```

Figura 1-6

Concluzie

În urma elaborării acestui program, am avut ocazia să mă familiarizez cu elemente fundamentale din teoria grafurilor. Acesta a fost un proces educativ și provocator, care mi-a permis să îmi dezvolt înțelegerea și abilitățile în lucrul cu grafuri.

Pentru prima dată, am avut ocazia să elaborez o aplicație în Java, utilizând librăria JgraphT. Acesta a fost un pas important în dezvoltarea mea ca programator, permițându-mi să mă familiarizez cu o nouă librărie și să îmi îmbunătățesc abilitățile de programare în Java. Am învățat cum să utilizez diferitele funcții și metode oferite de JgraphT, și cum să le aplic în rezolvarea problemelor practice.

Unul dintre cele mai mari obstacole pe care le-am întâmpinat a fost elaborarea listei de adiacență. Acesta a fost un proces complex, care a necesitat o înțelegere profundă a structurilor de date și a modului în care acestea pot fi utilizate pentru a reprezenta grafuri. Cu toate acestea, prin perseverență și muncă asiduă, am reușit să depășesc aceste dificultăți și să finalizez cu succes lista de adiacență.

Acest laborator m-a ajutat să înțeleg mai bine modul în care un graf este stocat în memoria calculatorului. Am învățat despre diferitele metode de reprezentare a grafurilor în memorie, și despre avantajele și dezavantajele fiecărei metode. Aceasta a fost o lecție valoroasă, care mi-a oferit o perspectivă mai profundă asupra modului în care calculatoarele procesează și stochează informații.