**Nome: Kamilly Nobre Santos CURSO: Análise e desenvolvimento de sistemas**

**Nome: Ingrid Sampaio de Albuquerque CURSO: Sistema de informação**

**OAT - SETE PONTES DE KÖNIGSBERG**

import java.util.\*;

public class App {

static class Grafo {

private int numeroDeVertices;

private LinkedList<Integer>[] listaDeAdjacencia;

private int[][] matrizDeAdjacencia;

Grafo(int numeroDeVertices) {

this.numeroDeVertices = numeroDeVertices;

listaDeAdjacencia = new LinkedList[numeroDeVertices];

matrizDeAdjacencia = new int[numeroDeVertices][numeroDeVertices];

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

listaDeAdjacencia[i] = new LinkedList<>();

Arrays.fill(matrizDeAdjacencia[i], 0);

}

}

void adicionarAresta(int vertice1, int vertice2, int peso) {

listaDeAdjacencia[vertice1].add(vertice2);

listaDeAdjacencia[vertice2].add(vertice1);

matrizDeAdjacencia[vertice1][vertice2] = peso;

matrizDeAdjacencia[vertice2][vertice1] = peso;

}

boolean eEuleriano() {

if (!estaConectado())

return false;

int numeroDeVerticesImpares = 0;

int inicio = 0;

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

if (listaDeAdjacencia[i].size() % 2 != 0) {

numeroDeVerticesImpares++;

inicio = i;

}

}

if (numeroDeVerticesImpares != 0 && numeroDeVerticesImpares != 2)

return false;

encontrarCaminhoEuleriano(inicio);

return true;

}

private boolean estaConectado() {

boolean visitado[] = new boolean[numeroDeVertices];

int i;

for (i = 0; i < numeroDeVertices; i++)

visitado[i] = false;

for (i = 0; i < numeroDeVertices; i++)

if (listaDeAdjacencia[i].size() != 0)

break;

if (i == numeroDeVertices)

return true;

realizarDFS(i, visitado);

for (i = 0; i < numeroDeVertices; i++)

if (visitado[i] == false && listaDeAdjacencia[i].size() > 0)

return false;

return true;

}

private void realizarDFS(int vertice, boolean visitado[]) {

visitado[vertice] = true;

Iterator<Integer> iterator = listaDeAdjacencia[vertice].iterator();

while (iterator.hasNext()) {

int verticeAdjacente = iterator.next();

if (!visitado[verticeAdjacente])

realizarDFS(verticeAdjacente, visitado);

}

}

private void encontrarCaminhoEuleriano(int inicio) {

Stack<Integer> caminho = new Stack<>();

List<Integer> caminhoEuleriano = new ArrayList<>();

caminho.push(inicio);

while (!caminho.isEmpty()) {

int verticeAtual = caminho.peek();

if (listaDeAdjacencia[verticeAtual].isEmpty()) {

caminhoEuleriano.add(verticeAtual);

caminho.pop();

} else {

int verticeAdjacente = listaDeAdjacencia[verticeAtual].poll();

listaDeAdjacencia[verticeAdjacente].remove((Integer) verticeAtual);

caminho.push(verticeAdjacente);

}

}

System.out.print("Sequência de bairros (vértices) visitados: ");

for (int i = 0; i < caminhoEuleriano.size(); i++) {

System.out.print((char) ('A' + caminhoEuleriano.get(i)));

if (i < caminhoEuleriano.size() - 1) {

System.out.print(" -> ");

}

}

System.out.println();

}

void imprimirMatrizDeAdjacencia() {

System.out.println("Matriz de Adjacência:");

System.out.println(" ");

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

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

System.out.print(matrizDeAdjacencia[i][j] + " ");

}

System.out.println();

}

}

public static void main(String args[]) {

Grafo grafo = new Grafo(4);

grafo.adicionarAresta(0, 1, 1);

grafo.adicionarAresta(0, 1, 2);

grafo.adicionarAresta(0, 2, 3);

grafo.adicionarAresta(0, 2, 4);

grafo.adicionarAresta(0, 3, 5);

grafo.adicionarAresta(1, 3, 6);

grafo.adicionarAresta(2, 3, 7);

grafo.imprimirMatrizDeAdjacencia();

System.out.println(" ");

if (grafo.eEuleriano())

System.out.println("Foi possível passear por todos os bairros sem repetir o mesmo caminho.");

else

System.out.println("Não foi possível passear por todos os bairros sem repetir o mesmo caminho.");

}

}

}