**Nome:** Arthur Silva Berdusco de Souza

**Curso:** Tecnologia em Análise e Desenvolvimento de Sistemas

**Turma:** 3ºB

**EXERCÍCIO: PROVA 02 – B**

São Paulo, 23 de Novembro de 2023

Classe: **Vertice**

package br.com.senac.exemplografo;

import java.util.ArrayList;

public class Vertice<TIPO> {

private TIPO dado;

private ArrayList<Aresta<TIPO>> arestasEntrada;

private ArrayList<Aresta<TIPO>> arestasSaida;

public Vertice(TIPO dado) {

this.dado = dado;

this.arestasEntrada = new ArrayList<Aresta<TIPO>>();

this.arestasSaida = new ArrayList<Aresta<TIPO>>();

}

public TIPO getDado() {

return dado;

}

public void setDado(TIPO dado) {

this.dado = dado;

}

public ArrayList<Aresta<TIPO>> getArestasEntrada() {

return arestasEntrada;

}

public void setArestasEntrada(ArrayList<Aresta<TIPO>> arestasEntrada) {

this.arestasEntrada = arestasEntrada;

}

public ArrayList<Aresta<TIPO>> getArestasSaida() {

return arestasSaida;

}

public void setArestasSaida(ArrayList<Aresta<TIPO>> arestasSaida) {

this.arestasSaida = arestasSaida;

}

public void adicionarArestaEntrada(Aresta<TIPO> aresta){

this.arestasEntrada.add(aresta);

}

public void adicionarArestaSaida(Aresta<TIPO> aresta){

this.arestasSaida.add(aresta);

}

}

Classe: **Aresta**

package br.com.senac.exemplografo;

public class Aresta<TIPO> {

private int peso;

private Vertice<TIPO> inicio;

private Vertice<TIPO> fim;

public Aresta(int peso, Vertice<TIPO> inicio, Vertice<TIPO> fim) {

this.peso = peso;

this.inicio = inicio;

this.fim = fim;

}

public int getPeso() {

return peso;

}

public void setPeso(int peso) {

this.peso = peso;

}

public Vertice<TIPO> getInicio() {

return inicio;

}

public void setInicio(Vertice<TIPO> inicio) {

this.inicio = inicio;

}

public Vertice<TIPO> getFim() {

return fim;

}

public void setFim(Vertice<TIPO> fim) {

this.fim = fim;

}

}

Classe: **Grafo**

package br.com.senac.exemplografo;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.LinkedList;

import java.util.List;

import java.util.Queue;

import java.util.Set;

public class Grafo<TIPO> {

private ArrayList<Vertice<TIPO>> vertices;

private ArrayList<Aresta<TIPO>> arestas;

public Grafo(ArrayList<Vertice<TIPO>> vertices, ArrayList<Aresta<TIPO>> arestas) {

this.vertices = vertices;

this.arestas = arestas;

}

public Grafo() {

vertices = new ArrayList<Vertice<TIPO>>();

arestas = new ArrayList<Aresta<TIPO>>();

}

public void adicionarVertice(TIPO dado) {

Vertice<TIPO> novoVertice = new Vertice<TIPO>(dado);

this.vertices.add(novoVertice);

}

public void adicionarAresta(int peso, TIPO dadoInicio, TIPO dadoFim) {

Vertice<TIPO> inicio = this.getVertice(dadoInicio);

Vertice<TIPO> fim = this.getVertice(dadoFim);

Aresta<TIPO> arestaDirecional = new Aresta<TIPO>(peso, inicio, fim);

inicio.adicionarArestaSaida(arestaDirecional);

fim.adicionarArestaEntrada(arestaDirecional);

Aresta<TIPO> arestaOposta = new Aresta<TIPO>(peso, fim, inicio);

fim.adicionarArestaSaida(arestaOposta);

inicio.adicionarArestaEntrada(arestaOposta);

this.arestas.add(arestaDirecional);

this.arestas.add(arestaOposta);

}

public Vertice<TIPO> getVertice(TIPO dado) {

Vertice<TIPO> vertice = null;

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

if (this.vertices.get(i).getDado().equals(dado)) {

vertice = this.vertices.get(i);

break;

}

}

return vertice;

}

//Metodos de busca

public void buscaEmLargura(TIPO dado) {

ArrayList<Vertice<TIPO>> marcados = new ArrayList<Vertice<TIPO>>();

ArrayList<Vertice<TIPO>> fila = new ArrayList<Vertice<TIPO>>();

Vertice<TIPO> atual = this.getVertice(dado);

marcados.add(atual);

System.out.println(atual.getDado());

fila.add(atual);

while (fila.size() > 0) {

Vertice<TIPO> visitado = fila.get(0);

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

Vertice<TIPO> proximo = visitado.getArestasSaida().get(i).getFim();

if (!marcados.contains(proximo)) {

marcados.add(proximo);

System.out.println(proximo.getDado());

fila.add(proximo);

}

}

fila.remove(0);

}

}

public void encontrarMelhoresCaminhos(TIPO inicio, TIPO fim) {

Vertice<TIPO> verticeInicio = this.getVertice(inicio);

Vertice<TIPO> verticeFim = this.getVertice(fim);

List<List<Aresta<TIPO>>> todosCaminhos = new ArrayList<>();

Queue<List<Aresta<TIPO>>> fila = new LinkedList<>();

Set<Vertice<TIPO>> visitados = new HashSet<>();

List<Aresta<TIPO>> caminhoInicial = new ArrayList<>();

fila.add(caminhoInicial);

while (!fila.isEmpty()) {

List<Aresta<TIPO>> caminhoAtual = fila.poll();

Aresta<TIPO> arestaAtual = caminhoAtual.isEmpty() ? null : caminhoAtual.get(caminhoAtual.size() - 1);

Vertice<TIPO> verticeAtual = (arestaAtual == null) ? verticeInicio : arestaAtual.getFim();

if (verticeAtual.equals(verticeFim)) {

todosCaminhos.add(caminhoAtual);

continue;

}

if (!visitados.contains(verticeAtual)) {

visitados.add(verticeAtual);

for (Aresta<TIPO> aresta : verticeAtual.getArestasSaida()) {

Vertice<TIPO> vizinho = aresta.getFim();

if (!visitados.contains(vizinho)) {

List<Aresta<TIPO>> novoCaminho = new ArrayList<>(caminhoAtual);

novoCaminho.add(aresta);

fila.add(novoCaminho);

}

}

}

}

// Imprimir todos os caminhos encontrados

int caminhos = 1;

for (List<Aresta<TIPO>> caminho : todosCaminhos) {

System.out.print("[Caminho=" + caminhos + " ---> ");

imprimirCaminho(caminho);

System.out.print("]\n");

caminhos++;

}

}

private void imprimirCaminho(List<Aresta<TIPO>> caminho) {

StringBuilder resultado = new StringBuilder();

int distanciaTotal = 0;

for (Aresta<TIPO> aresta : caminho) {

resultado.append(aresta.getInicio().getDado())

.append(" - ")

.append(aresta.getFim().getDado())

.append(" ")

.append(aresta.getPeso())

.append(" metros, ");

distanciaTotal += aresta.getPeso();

}

if (resultado.length() > 0) {

resultado.delete(resultado.length() - 2, resultado.length());

}

System.out.print(resultado.toString() + " - Distância Total: " + distanciaTotal + " metros");

}

}

Classe: **ExemploGrafo**

package br.com.senac.exemplografo;

import java.util.Scanner;

public class ExemploGrafo {

public static void main(String[] args) {

Grafo<String> grafo = new Grafo<String>();

grafo.adicionarVertice("A");

grafo.adicionarVertice("B");

grafo.adicionarVertice("C");

grafo.adicionarVertice("D");

grafo.adicionarVertice("E");

grafo.adicionarVertice("F");

grafo.adicionarVertice("G");

grafo.adicionarVertice("H");

grafo.adicionarVertice("I");

grafo.adicionarVertice("J");

grafo.adicionarVertice("K");

grafo.adicionarVertice("L");

grafo.adicionarVertice("M");

grafo.adicionarVertice("N");

grafo.adicionarVertice("O");

grafo.adicionarVertice("P");

grafo.adicionarVertice("Q");

grafo.adicionarVertice("R");

grafo.adicionarVertice("S");

grafo.adicionarVertice("T");

grafo.adicionarVertice("U");

grafo.adicionarVertice("V");

grafo.adicionarVertice("X");

grafo.adicionarAresta(300, "A", "B");

grafo.adicionarAresta(47, "B", "C");

grafo.adicionarAresta(62, "C", "D");

grafo.adicionarAresta(8, "D", "E");

grafo.adicionarAresta(13, "E", "F");

grafo.adicionarAresta(230, "E", "G");

grafo.adicionarAresta(141, "C", "H");

grafo.adicionarAresta(138, "H", "I");

grafo.adicionarAresta(153, "I", "J");

grafo.adicionarAresta(512, "J", "K");

grafo.adicionarAresta(135, "K", "L");

grafo.adicionarAresta(50, "L", "M");

grafo.adicionarAresta(187, "L", "N");

grafo.adicionarAresta(108, "N", "O");

grafo.adicionarAresta(82, "O", "P");

grafo.adicionarAresta(215, "P", "Q");

grafo.adicionarAresta(97, "Q", "R");

grafo.adicionarAresta(243, "R", "T");

grafo.adicionarAresta(207, "T", "S");

grafo.adicionarAresta(33, "R", "S");

grafo.adicionarAresta(38, "S", "V");

grafo.adicionarAresta(22, "T", "U");

grafo.adicionarAresta(210, "U", "V");

grafo.adicionarAresta(370, "V", "A");

grafo.adicionarAresta(317, "X", "A");

grafo.adicionarAresta(107, "U", "X");

Scanner input = new Scanner(System.in);

System.out.println("Escolha o ponto de partida: ");

String inicio = input.next().toUpperCase();

System.out.println("Escolha o ponto de destino: ");

String fim = input.next().toUpperCase();

grafo.encontrarMelhoresCaminhos(inicio, fim);

}

}