Tecnólogo em Análise e Desenvolvimento de Sistemas
Estrutura de Dados – Prova 2B
Professor Claudio Benossi

Nome: NATHAN HENRIQUE VIEIRA FERREIRA

Turma: TURMA A

Data: 28/05/2024

Aresta.java:

```java
public class Aresta {
  No destino;
  int peso;

  public Aresta(No destino, int peso) {
    this.destino = destino;
    this.peso = peso;
  }
}
```

No.java:

```java
import java.util.ArrayList;
import java.util.List;

public class No {
  String nome;
  List<Aresta> arestas;
  int distancia;
  No anterior;

  public No(String nome) {
    this.nome = nome;
    this.arestas = new ArrayList<>();
    this.distancia = Integer.MAX_VALUE;
    this.anterior = null;
  }
```

```
  public void adicionarAresta(Aresta aresta) {
    this.arestas.add(aresta);
  }
}
```

Grafo.java:

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Grafo {
  Map<String, No> nos;

  public Grafo() {
    nos = new HashMap<>();
  }

  public void adicionarNo(String nome) {
    nos.put(nome, new No(nome));
  }

  public void adicionarAresta(String de, String para, int peso) {
    No noDe = nos.get(de);
    No noPara = nos.get(para);
    noDe.adicionarAresta(new Aresta(noPara, peso));
    noPara.adicionarAresta(new Aresta(noDe, peso)); // Grafo não-
direcionado
  }

  public void encontrarTodosOsCaminhos(String inicio, String fim,
List<String> caminhoAtual,
      List<List<String>> todosOsCaminhos) {
    caminhoAtual.add(inicio);

    if (inicio.equals(fim)) {
      todosOsCaminhos.add(new ArrayList<>(caminhoAtual));
    } else {
      No noAtual = nos.get(inicio);
      for (Aresta aresta : noAtual.arestas) {
        if (!caminhoAtual.contains(aresta.destino.nome)) {
```

```
          encontrarTodosOsCaminhos(aresta.destino.nome, fim, caminhoAtual,
todosOsCaminhos);
        }
      }
    }

    caminhoAtual.remove(caminhoAtual.size() - 1);
  }

  public int calcularDistancia(List<String> caminho) {
    int distancia = 0;
    for (int i = 0; i < caminho.size() - 1; i++) {
      No noAtual = nos.get(caminho.get(i));
      for (Aresta aresta : noAtual.arestas) {
        if (aresta.destino.nome.equals(caminho.get(i + 1))) {
          distancia += aresta.peso;
          break;
        }
      }
    }
    return distancia;
  }

  public void encontrarEImprimirCaminhos(String nomeInicio, String
nomeFim) {
    List<List<String>> todosOsCaminhos = new ArrayList<>();
    encontrarTodosOsCaminhos(nomeInicio, nomeFim, new ArrayList<>(),
todosOsCaminhos);

    if (todosOsCaminhos.isEmpty()) {
      System.out.println("Nenhum caminho encontrado.");
      return;
    }

    List<String> caminhoMaisCurto = null;
    List<String> caminhoMaisLongo = null;
    int menorDistancia = Integer.MAX_VALUE;
    int maiorDistancia = Integer.MIN_VALUE;

    for (List<String> caminho : todosOsCaminhos) {
      int distancia = calcularDistancia(caminho);
      if (distancia < menorDistancia) {
        menorDistancia = distancia;
        caminhoMaisCurto = caminho;
      }
      if (distancia > maiorDistancia) {
        maiorDistancia = distancia;
```

```
        caminhoMaisLongo = caminho;
      }
    }

    System.out.println("Caminho mais curto:");
    imprimirCaminho(caminhoMaisCurto);
    System.out.println("Distância total: " + menorDistancia + " metros");

    System.out.println("Caminho mais longo:");
    imprimirCaminho(caminhoMaisLongo);
    System.out.println("Distância total: " + maiorDistancia + " metros");
  }

  private void imprimirCaminho(List<String> caminho) {
    if (caminho == null) {
      System.out.println("Nenhum caminho encontrado.");
      return;
    }
    for (int i = 0; i < caminho.size(); i++) {
      System.out.print(caminho.get(i));
      if (i < caminho.size() - 1) {
        System.out.print(" -> ");
      }
    }
    System.out.println();
  }
}
```

Main.java:

```
import java.util.Scanner;

public class Main {
  public static void main(String[] args) {
    Grafo grafo = new Grafo();

    String[] pontos = { "A", "B", "C", "D", "E", "F", "G", "H", "I", "J",
"K", "L", "M", "N", "O", "P", "Q", "R", "S",
        "T", "U", "V", "X" };
    for (String ponto : pontos) {
      grafo.adicionarNo(ponto);
    }

    grafo.adicionarAresta("A", "B", 300);
    grafo.adicionarAresta("B", "C", 47);
```

```
        grafo.adicionarAresta("C", "D", 62);
        grafo.adicionarAresta("C", "H", 141);
        grafo.adicionarAresta("D", "E", 8);
        grafo.adicionarAresta("E", "F", 13);
        grafo.adicionarAresta("E", "G", 230);
        grafo.adicionarAresta("H", "I", 138);
        grafo.adicionarAresta("I", "J", 153);
        grafo.adicionarAresta("J", "K", 512);
        grafo.adicionarAresta("K", "L", 135);
        grafo.adicionarAresta("L", "M", 20); // NÃO TINHA NA IMAGEM, ENTÃO
COLOQUEI UM VALOR QUE ACHEI QUE ERA DE ACORDO
        grafo.adicionarAresta("L", "N", 187);
        grafo.adicionarAresta("N", "O", 108);
        grafo.adicionarAresta("O", "P", 82);
        grafo.adicionarAresta("P", "Q", 215);
        grafo.adicionarAresta("Q", "R", 97);
        grafo.adicionarAresta("R", "S", 33);
        grafo.adicionarAresta("R", "T", 243);
        grafo.adicionarAresta("S", "T", 207);
        grafo.adicionarAresta("S", "V", 38);
        grafo.adicionarAresta("T", "U", 22);
        grafo.adicionarAresta("V", "U", 210);
        grafo.adicionarAresta("V", "A", 370);
        grafo.adicionarAresta("U", "X", 107);
        grafo.adicionarAresta("X", "A", 317);

        Scanner sc = new Scanner(System.in);
        System.out.print("Digite o ponto de partida: ");
        String inicio = sc.nextLine().toUpperCase();
        System.out.print("Digite o ponto de chegada: ");
        String fim = sc.nextLine().toUpperCase();

        if (!grafo.nos.containsKey(inicio) || !grafo.nos.containsKey(fim)) {
          System.out.println("Ponto de partida ou chegada inválido.");
          sc.close();
          return;
        }

        grafo.encontrarEImprimirCaminhos(inicio, fim);

        sc.close();
    }
}
```