**ATIVIDADE 2**

**ATENÇÃO:**

1. **Esta Atividade deverá ser feita em GRUPO DE PELO MENOS 04 ALUNOS E DE NO MÁXIMO 08 ALUNOS embora a entrega deverá ser feita INDIVIDUALMENTE.**
2. **Atividades feitas individualmente ou entregues com atraso NÃO SERÃO CONSIDERADAS.**

**Grupo**

Rafael Rossetto Guitarrari RA : 823158602

Andrey de Freitas Souza RA : 823217536

Gabriel Farah De lima RA: 822231424

Fabrício de Barros Narbon RA:822227166

Bianca Alves Ribeiro RA: 8222240261

Luiz Gustavo França de Abreu RA: 823210075

Gabrielle Garcia Paz  RA: 823126085

Webster Diógenes Rodrigues RA:8222242764

Implementar **em Java e em Python** os exercícios abaixo.

1. Faça um programa que permita o usuário fornecer os vértices de um grafo e suas arestas. Exibir as matrizes de incidência e de adjacência do grafo fornecido. Exibir, em seguida, o grau de cada vértice.
2. Faça um programa que receba permita o usuário fornecer um grafo (como no exercício anterior). Em seguida o programa deverá testar se há caminho de um vértice do gráfico para todos os demais vértices.
3. Faça um programa que receba permita o usuário fornecer um grafo cujas arestas tenham peso. Imprimir a matriz de adjacência do grafo com peso e testar se há caminho de um vértice do gráfico para todos os demais vértices imprimindo o total de distância deste vértice para cada um dos outros vértices.

**RESPOSTAS**

# Ex- 1)

import java.util.\*;

public class Grafo {

    private int numVertices;

    private List<String> vertices;

    private List<int[]> arestas;

    public Grafo(int numVertices) {

        this.numVertices = numVertices;

        this.vertices = **new** ArrayList<>();

        this.arestas = **new** ArrayList<>();

    }

    public void adicionarVertice(String vertice) {

        vertices.add(vertice);

    }

    public void adicionarAresta(int vertice1, int vertice2) {

        arestas.add(**new** int[]{vertice1, vertice2});

    }

    public void exibirMatrizAdjacencia() {

        int[][] matrizAdj = **new** int[numVertices][numVertices];

        for (int[] aresta : arestas) {

            int v1 = aresta[0];

            int v2 = aresta[1];

            matrizAdj[v1][v2] = 1;

            matrizAdj[v2][v1] = 1; // Para grafos não direcionados

        }

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

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

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

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

            }

            System.out.println();

        }

    }

    public void exibirMatrizIncidencia() {

        int[][] matrizIncidencia = **new** int[numVertices][arestas.size()];

        for (int j = 0; j < arestas.size(); j++) {

            int[] aresta = arestas.get(j);

            int v1 = aresta[0];

            int v2 = aresta[1];

            matrizIncidencia[v1][j] = 1;

            matrizIncidencia[v2][j] = 1; // Para grafos não direcionados

        }

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

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

            for (int j = 0; j < arestas.size(); j++) {

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

            }

            System.out.println();

        }

    }

    public void exibirGrauVertices() {

        int[] grau = **new** int[numVertices];

        for (int[] aresta : arestas) {

            grau[aresta[0]]++;

            grau[aresta[1]]++; // Para grafos não direcionados

        }

        System.out.println("Grau dos Vértices:");

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

            System.out.println("Vértice " + vertices.get(i) + " tem grau " + grau[i]);

        }

    }

    public static void main(String[] args) {

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

        System.out.print("Digite o número de vértices: ");

        int numVertices = scanner.nextInt();

        scanner.nextLine(); // Consumir nova linha

        Grafo grafo = **new** Grafo(numVertices);

        System.out.println("Digite os vértices:");

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

            System.out.print("Vértice " + (i + 1) + ": ");

            String vertice = scanner.nextLine();

            grafo.adicionarVertice(vertice);

        }

        System.out.print("Digite o número de arestas: ");

        int numArestas = scanner.nextInt();

        System.out.println("Digite as arestas (pares de índices de vértices começando de 0):");

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

            System.out.print("Aresta " + (i + 1) + " - Vértice 1: ");

            int v1 = scanner.nextInt();

            System.out.print("Aresta " + (i + 1) + " - Vértice 2: ");

            int v2 = scanner.nextInt();

            grafo.adicionarAresta(v1, v2);

        }

        grafo.exibirMatrizAdjacencia();

        grafo.exibirMatrizIncidencia();

        grafo.exibirGrauVertices();

        scanner.close();

    }

}

# Ex-2)

import java.util.\*;

public class Grafo {

    private int numVertices;

    private List<String> vertices;

    private List<int[]> arestas;

    private Map<String, Integer> mapaVertices;

    public Grafo(int numVertices) {

        this.numVertices = numVertices;

        this.vertices = **new** ArrayList<>();

        this.arestas = **new** ArrayList<>();

        this.mapaVertices = **new** HashMap<>();

    }

    public void adicionarVertice(String vertice) {

        vertices.add(vertice);

        mapaVertices.put(vertice, vertices.size() - 1);

    }

    public void adicionarAresta(int vertice1, int vertice2) {

        arestas.add(**new** int[]{vertice1, vertice2});

    }

    public boolean existeCaminhoParaTodos(int verticeInicial) {

        boolean[] visitados = **new** boolean[numVertices];

        dfs(verticeInicial, visitados);

        for (boolean visitado : visitados) {

            if (!visitado) {

                return false;

            }

        }

        return true;

    }

    private void dfs(int vertice, boolean[] visitados) {

        visitados[vertice] = true;

        for (int[] aresta : arestas) {

            int v1 = aresta[0];

            int v2 = aresta[1];

            if (v1 == vertice && !visitados[v2]) {

                dfs(v2, visitados);

            } else if (v2 == vertice && !visitados[v1]) {

                dfs(v1, visitados);

            }

        }

    }

    public static void main(String[] args) {

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

        System.out.print("Digite o número de vértices: ");

        int numVertices = scanner.nextInt();

        scanner.nextLine(); // Consumir nova linha

        Grafo grafo = **new** Grafo(numVertices);

        System.out.println("Digite os vértices:");

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

            System.out.print("Vértice " + (i + 1) + ": ");

            String vertice = scanner.nextLine();

            grafo.adicionarVertice(vertice);

        }

        System.out.print("Digite o número de arestas: ");

        int numArestas = scanner.nextInt();

        System.out.println("Digite as arestas (pares de índices de vértices começando de 0):");

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

            System.out.print("Aresta " + (i + 1) + " - Vértice 1: ");

            int v1 = scanner.nextInt();

            System.out.print("Aresta " + (i + 1) + " - Vértice 2: ");

            int v2 = scanner.nextInt();

            grafo.adicionarAresta(v1, v2);

        }

        System.out.print("Digite o vértice inicial para testar o caminho: ");

        scanner.nextLine(); // Consumir nova linha

        String verticeInicial = scanner.nextLine();

        int indiceVerticeInicial = grafo.mapaVertices.get(verticeInicial);

        boolean caminhoParaTodos = grafo.existeCaminhoParaTodos(indiceVerticeInicial);

        if (caminhoParaTodos) {

            System.out.println("Há caminho do vértice " + verticeInicial + " para todos os outros vértices.");

        } else {

            System.out.println("Não há caminho do vértice " + verticeInicial + " para todos os outros vértices.");

        }

        scanner.close();

    }

}

# Ex – 3)

import java.util.\*;

public class Grafo {

    private int numVertices;

    private List<String> vertices;

    private int[][] matrizAdjacencia;

    private Map<String, Integer> mapaVertices;

    public Grafo(int numVertices) {

        this.numVertices = numVertices;

        this.vertices = **new** ArrayList<>();

        this.matrizAdjacencia = **new** int[numVertices][numVertices];

        for (int[] row : matrizAdjacencia) {

            Arrays.fill(row, Integer.MAX\_VALUE);

        }

        this.mapaVertices = **new** HashMap<>();

    }

    public void adicionarVertice(String vertice) {

        vertices.add(vertice);

        mapaVertices.put(vertice, vertices.size() - 1);

    }

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

        matrizAdjacencia[vertice1][vertice2] = peso;

        matrizAdjacencia[vertice2][vertice1] = peso; // Para grafos não direcionados

    }

    public void exibirMatrizAdjacencia() {

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

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

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

                if (matrizAdjacencia[i][j] == Integer.MAX\_VALUE) {

                    System.out.print("∞ ");

                } else {

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

                }

            }

            System.out.println();

        }

    }

    public void dijkstra(int verticeInicial) {

        int[] distancias = **new** int[numVertices];

        boolean[] visitados = **new** boolean[numVertices];

        Arrays.fill(distancias, Integer.MAX\_VALUE);

        distancias[verticeInicial] = 0;

        PriorityQueue<int[]> heap = **new** PriorityQueue<>(Comparator.comparingInt(a -> a[1]));

        heap.add(**new** int[]{verticeInicial, 0});

        while (!heap.isEmpty()) {

            int[] atual = heap.poll();

            int vertice = atual[0];

            if (visitados[vertice]) continue;

            visitados[vertice] = true;

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

                if (matrizAdjacencia[vertice][i] != Integer.MAX\_VALUE && !visitados[i]) {

                    int novaDistancia = distancias[vertice] + matrizAdjacencia[vertice][i];

                    if (novaDistancia < distancias[i]) {

                        distancias[i] = novaDistancia;

                        heap.add(**new** int[]{i, novaDistancia});

                    }

                }

            }

        }

        System.out.println("Distâncias a partir do vértice " + vertices.get(verticeInicial) + ":");

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

            if (distancias[i] == Integer.MAX\_VALUE) {

                System.out.println("Vértice " + vertices.get(i) + " é inacessível.");

            } else {

                System.out.println("Distância até " + vertices.get(i) + ": " + distancias[i]);

            }

        }

    }

    public static void main(String[] args) {

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

        System.out.print("Digite o número de vértices: ");

        int numVertices = scanner.nextInt();

        scanner.nextLine(); // Consumir nova linha

        Grafo grafo = **new** Grafo(numVertices);

        System.out.println("Digite os vértices:");

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

            System.out.print("Vértice " + (i + 1) + ": ");

            String vertice = scanner.nextLine();

            grafo.adicionarVertice(vertice);

        }

        System.out.print("Digite o número de arestas: ");

        int numArestas = scanner.nextInt();

        System.out.println("Digite as arestas (pares de índices de vértices e peso):");

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

            System.out.print("Aresta " + (i + 1) + " - Vértice 1: ");

            int v1 = scanner.nextInt();

            System.out.print("Aresta " + (i + 1) + " - Vértice 2: ");

            int v2 = scanner.nextInt();

            System.out.print("Peso da aresta: ");

            int peso = scanner.nextInt();

            grafo.adicionarAresta(v1, v2, peso);

        }

        grafo.exibirMatrizAdjacencia();

        System.out.print("Digite o vértice inicial para testar o caminho: ");

        scanner.nextLine(); // Consumir nova linha

        String verticeInicial = scanner.nextLine();

        int indiceVerticeInicial = grafo.mapaVertices.get(verticeInicial);

        grafo.dijkstra(indiceVerticeInicial);

        scanner.close();

    }

}