Prática de Laboratório 7

Objetivos:

- Tipos Abstratos de Dados - TADs.
- Implementação de Grafos.

1. Implemente as classes abaixo:

Vértice

```
 7  public class Vertice {
 8
 9      protected String ID;
10      protected float peso;
11      ArrayList<Vertice> listaAdjacentes;
12
13      public Vertice(String ID, float peso) {
14          this.ID = ID;
15          this.peso = peso;
16          this.listaAdjacentes = new ArrayList<Vertice>();
17      }
18
19      public String getID() {
20          return ID;
21      }
22
23      public float getPeso() {
24          return peso;
25      }
26
27      public void setPeso(int peso) {
28          this.peso = peso;
29      }
30
31      public ArrayList<Vertice> getListaAdjacentes() {
32          return listaAdjacentes;
33      }
34
35      public void addAdjacente(Vertice n) {
36          this.listaAdjacentes.add(n);
37      }
38
39      public void removeAdjacente(Vertice n) {
40          this.listaAdjacentes.remove(n);
41      }
42
43  }
```

Aresta

```java
public class Aresta {

    private float peso;
    private String ID;
    private Vertice v1, v2;
    private boolean orientada;

    public Aresta(Vertice v1, Vertice v2, float peso, String ID, boolean orientada) {
        this.v1 = v1;
        this.v2 = v2;
        this.peso = peso;
        this.ID = ID;
        this.orientada = orientada;
    }

    public float peso() {
        return this.peso;
    }

    public Vertice getV1() {
        return this.v1;
    }

    public Vertice getV2() {
        return this.v2;
    }

    public float getPeso() {
        return peso;
    }

    public String getID() {
        return this.ID;
    }

    public boolean isOrientada() {
        return orientada;
    }
}
```

Grafo

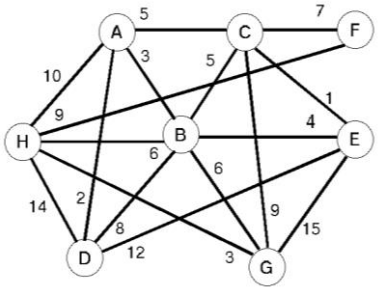
```java
public class Grafo {

    protected ArrayList<Vertice> V;
    protected ArrayList<Aresta> E;

    public Grafo() {
        this.V = new ArrayList<>();
        this.E = new ArrayList<>();
    }

    public ArrayList<Vertice> getVertices() {
        return V;
    }

    public ArrayList<Aresta> getArestas() {
        return E;
    }

    public Vertice searchVertice(String ID) {
        for (Vertice v : this.V) {
            if (v.getID().equals(ID))
                return v;
        }
        return null;
    }

    public void addVertice(String ID, float peso) {
        V.add(new Vertice(ID, peso));
    }

    public void addAresta(float peso, String ID, String idV1, String idV2, boolean orientada) {
        // Busca o vértice V1
        Vertice v1 = searchVertice(idV1);
        // Busca o vértice V2
        Vertice v2 = searchVertice(idV2);
        this.E.add(new Aresta(v1, v2, peso, ID, orientada));
        // Quando adiciona uma aresta, deve-se atualizar a lista de adjacências
        if (orientada) {
            v1.addAdjacente(v2);
        } else {
            v1.addAdjacente(v2);
            v2.addAdjacente(v1);
        }
    }

    public void printGraph() {
        System.out.println("Lista de vértices: ");
        for (Vertice vertice : V) {
            System.out.println("ID: " + vertice.getID() + " Peso: " + vertice.getPeso());
        }
        System.out.println("Lista de arestas: ");
        for (Aresta aresta : E) {
            System.out.println("ID: " + aresta.getID() + " Peso: " + aresta.getPeso());
        }
    }

}
```

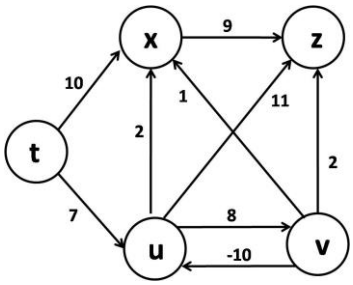
melhoresanos.unibh.br unibh	**Instituto de Engenharia e Tecnologia - IET**
Curso: Ciência da Computação	**Disciplina:** Algoritmos e Estruturas de Dados
Professor(a): Jaqueline Faria de Oliveira	

Exercícios:

Faça os exercícios considerando a TAD grafos:

1. Crie os grafos abaixo e imprima ao final a lista de vértices e arestas, considerando o ID das arestas o ID do vértice de origem e o ID do vértice de destino.

Grafo A = 

Grafo B = 

2. Insira as seguintes validações na TAD Grafo:
   a. Não permitir inserir vértice com ID repetido
   b. Não permitir inserir aresta com ID repetido
   c. Não permitir inserir aresta que não tenha os vértices de origem e destino
3. Implemente uma função para imprimir todos os adjacentes de um vértice.
4. Implemente uma função que recebe o ID de dois vértices e retorna True se esses forem adjacentes e False caso não sejam adjacentes.

Bom trabalho!