Prática 1 – Aplicações de Teoria dos Grafos

Grupo de 5 aluno(a)s

Objetivo: Projetar e desenvolver uma biblioteca Java para manipular grafos. A biblioteca deverá ser capaz de representar grafos e um conjunto de algoritmos clássicos.

O que será avaliado?

A corretude e design da biblioteca. Portanto, **busquem desenvolver a melhor implementação possível!!**

Plágios que forem identificados sofrerão as devidas punições.

Descrição:

Sua biblioteca deve possuir, pelo menos, os seguintes métodos que serão acessados externamente:

- **readGraph(path)**. A biblioteca deve ler um grafo a partir de um arquivo de texto. O grafo será descrito segundo o seguinte formato: a primeira linha informa o número de vértices do grafo. Cada linha subsequente informa as arestas do mesmo.

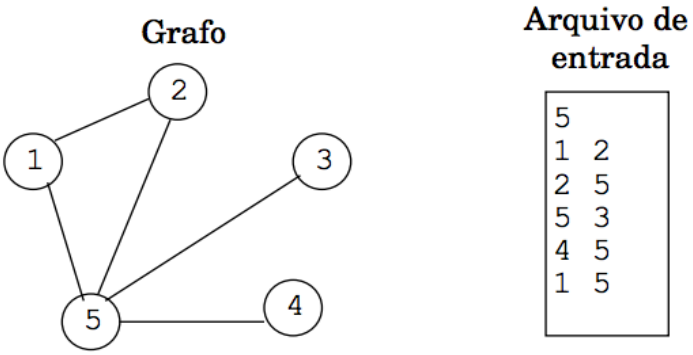

Figure 1 - Exemplo de grafo de entrada

- **readWeightedGraph(path).** Grafos com pesos. A biblioteca deve ser capaz de representar e manipular grafos não-direcionados que possuam pesos nas arestas. Os pesos, que serão representados por valores reais, devem estar associados às arestas. O arquivo de entrada agora terá uma terceira coluna, que representa o peso da aresta.

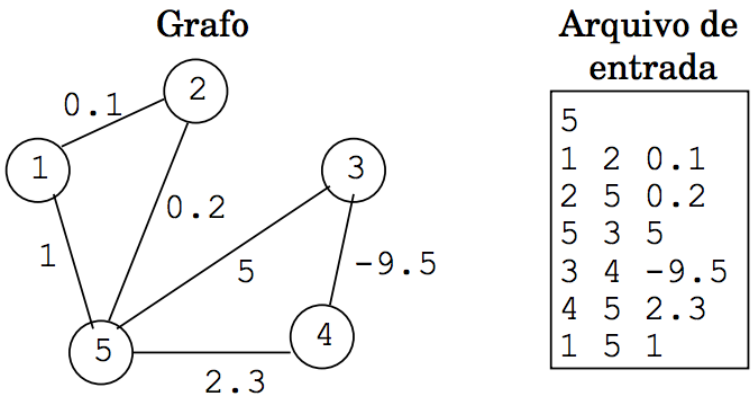

Figure 2 - Exemplo de Grafo de entrada com pesos

- **int getVertexNumber (graph), int getEdgeNumber (graph)** e **float getMeanEdge (graph)**. A biblioteca deve ser calcular as seguintes informações sobre o grafo: número de vértices, número de arestas e grau médio.

- **String graphRepresentation (graph, type)**. Representação de grafos. A biblioteca deve ser capaz de representar grafos utilizando tanto uma matriz de adjacência (type = "AM"), quanto uma lista de adjacência (type = "AL"). O usuário poderá escolher a representação a ser utilizada.

  Saída esperada para o grafo exemplo (Figura 1) para o tipo (AM):

```
  1 2 3 4 5
1 0 1 0 0 1
2 1 0 0 0 1
3 0 0 0 1 1
4 0 0 1 0 1
5 1 1 1 4 0
```

  Saída esperada para o grafo exemplo (Figura 1) para o tipo (AL):

```
1 - 2 3
2 - 1 5
3 - 4 5
4 - 3 5
5 - 1 2 3 4
```

- **String BFS(graph, v)** e **DFS(graph, v).** Busca em grafos: largura e profundidade. A biblioteca deve ser capaz de percorrer o grafo utilizando busca em largura e busca em profundidade. O vértice inicial será fornecido pelo usuário da biblioteca. A respectiva árvore de busca deve ser gerada assim como o nível de cada vértice na árvore (nível da raiz é zero). Para descrever a árvore gerada, basta informar o pai de cada vértice e seu nível na String de saída.

  Saída esperada para o grafo exemplo (Figura 1) para a execução do BFS:

```
1 - - 0
2 - 1 1
3 - 2 5
```

4 - 2 5
5 - 1 1

- **String SCC(graph)**. Componentes fortemente conectados. A biblioteca deve ser capaz descobrir os componentes fortemente conexos de um grafo. Saída: cada linha contém a lista de vértices pertencentes a um componente. Os componentes devem estar listados em ordem decrescente de tamanho (listar primeiro o componente com o maior número de vértices, etc).

  Saída esperada para o grafo exemplo (Figura 1):
  1 2 5
  3
  4

- **String shortestPath(v1, v2)**. Caminho mínimo. A biblioteca deve ser capaz de encontrar o caminho mais curto entre dois vértices.

  Saída esperada para o grafo exemplo (Figura 1), v1 = 1, v2 = 5:
  1 2 5

- **String mst(graph)**. Arvore geradora mínima. A biblioteca deve ser capaz de encontrar uma árvore geradora mínima de um grafo. Você deve escolher um algoritmo apropriado para este problema. Para descrever a árvore gerada, basta informar o pai de cada vértice e seu nível na String de saída (semelhante ao método BFS(graph, v)).

OBS.: O grupo poderá, se desejar, adicionar outras funcionalidades à biblioteca. Estas serão contabilizadas como bônus para a nota do grupo.

Crie uma suíte de testes JUnit que tenha, ao menos, um teste para cada método da sua biblioteca. Esses testes devem demonstrar o funcionamento da biblioteca.

O que deve ser entregue?

- Um representante da equipe deve anexar a atividade postada no Classroom:
  - O .jar da biblioteca criada e sua documentação.
  - O link para o repositório com a implementação da biblioteca e testes.

O que será apresentado em sala de aula?

- A demonstração do código desenvolvido
- As ideias usadas para desenvolvimento da biblioteca