Centro Federal de Educação Tecnológica de Minas Gerais - Departamento de Computação

Laboratório de Algoritmos e Estruturas de Dados II (2ECOM.025)

Lista de Exercícios 8

Professores: Amadeu Almeida e Thiago Rodrigues **Data de entrega:** 21 de Outubro de 2019

Instruções: leia com atenção **todas** as orientações abaixo antes de começar a lista.

- Os exercícios deverão ser feitos individualmente ou em dupla e valem 5 pontos.
- Submeta no SIGAA um arquivo compactado chamado **Lista8.zip**. Ele deverá conter:
  - As classes em Java que implementam as soluções das tarefas.
  - Um relatório em **PDF** que inclui as árvores e as tabelas obtidas nos experimentos e as respostas das questões do item 6 do exercício 3.
- Caso a tarefa seja feita em dupla, apenas um aluno precisa submetê-la.
- O prazo de entrega é até às 23:59 do dia 21 de Outubro.
- Códigos fonte em PDF e submissões em atraso serão desconsiderados.
- O enunciado desta lista possui 3 exercícios e 4 páginas.

Exercício 1: classe Grafo

Esta tarefa objetiva implementar uma classe que manipula um grafo que pode ser orientado ou não e que é representado por uma matriz de adjacências. Um exemplo de grafo não direcionado contendo 4 vértices e 6 arestas é apresentado na figura 1 enquanto a matriz de adjacências deste grafo é exibida na tabela 1.

Grafo:

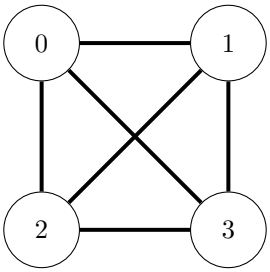

Figura 1: Grafo com 4 vértices e 6 arestas.

Matriz de adjacência:

Vértice	0	1	2	3
0	0	5	6	3
1	5	0	6	3
2	6	6	0	1
3	3	3	1	0

Tabela 1: Matriz de adjacências do grafo mostrado na figura 1

1. Abra o projeto **ListasDeGrafos** e o arquivo Grafo.java.
2. Adicione o método a seguir na classe **Grafo**:
   (a) public void insereArestaNaoOrientada(int vertice1, int vertice2, int peso) - adiciona as seguintes arestas na matriz de adjacência:
      i. vertice1 $--> $ vertice2
      ii. vertice2 $--> $ vertice1

      Observação: a junção das duas arestas orientadas incluídas no método acima resulta em uma aresta não orientada

Exercício 2: classe AlgoritmosEmGrafos

O objetivo deste exercício é acrescentar o algoritmo de Prim ou o de Kruskal na biblioteca de procedimentos em grafos que está sendo construída na classe **AlgoritmosEmGrafos**.

1. Acesse o arquivo **AlgoritmosEmGrafos.java**.
2. Insira os seguintes atributos na classe **AlgoritmosEmGrafos**:
   - **private final** $ArrayList < Pair < Integer, Integer >>$ **arestasArvoreGeradoraMinima;** - armazena as arestas que estão na árvore geradora mínima do grafo.
   - **private final int[] verticeAntecessorAGM;** - guarda o vértice antecessor de cada nó pertencente à árvore geradora mínima (AGM). Um nó antecessor é aquele que está na outra extremidade da aresta que conectou o vértice corrente à AGM durante a execução do algoritmo.
3. Elabore os métodos a seguir:
   - **public AlgoritmosEmGrafos(int vertices)** - o método construtor deve ser atualizado para inicializar os novos atributos da classe juntamente com os antigos.
   - **public int iniciaArvoreGeradoraMinima(int vertice)** - inicia o algoritmo que irá encontrar a árvore geradora mínima do grafo. Neste método, os valores de todos os vértices antecessores devem ser iniciados com -1.
   - **private int arvoreGeradoraMinima(int vertice)** - implementa o algoritmo de Prim ou o de Kruskal e retorna o peso da árvore geradora mínima.
   - **public** $ArrayList < Pair < Integer, Integer >>$ **getArestasArvore()**
   - **public int[] getVerticeAntecessorAGM()**
4. Explique o objetivo de todos os métodos implementados no item 3 por meio de comentários no próprio código.

Exercício 3: experimentos computacionais

Este exercício visa testar o funcionamento do algoritmo implementado na tarefa anterior em diferentes tipos de grafos não orientados.

1. Antes de começar os experimentos, faça o download dos dois grafos que estão anexados junto a esta lista no SIGAA. A primeira linha de cada arquivo possui, respectivamente, a quantidade de vértices e arestas dos grafos, enquanto as demais contém uma das arestas do grafo. O nó de saída, o de entrada e o peso da aresta são retratados, respectivamente, no primeiro, no segundo e no terceiro número de cada linha.

Por exemplo, o arquivo que define o grafo mostrado na primeira página deste documento é organizado da seguinte forma:

```
4 6
0 1 5
0 3 3
1 3 3
2 1 6
2 2 6
3 2 1
```

2. Experimento 1:
   (a) Execute o algoritmo de construção da árvore geradora mínima iniciando-o com vértice 0 de cada um dos dois grafos.
   (b) Para cada grafo, retorne:
      i. O peso total da árvore geradora mínima.
      ii. As arestas do grafo que estão na árvore.
      iii. O vetor de nós antecessores.
3. Experimento 2:
   (a) Rode o procedimento de obtenção da árvore geradora mínima nos dois grafos partindo de um vértice a sua escolha.
   (b) Para cada grafo, devolva:
      i. A soma das arestas que estão na árvore geradora mínima.
      ii. As arestas do grafo que pertencem à árvore.
      iii. A lista de vértices predecessores.
4. Crie duas tabelas que retratam os resultados dos experimentos 1 e 2 para o primeiro grafo. Cada tabela possui três colunas contendo respectivamente:
   - O número de cada vértice.
   - O vértice predecessor ao atual na árvore
   - O peso da aresta que conecta o nó antecessor ao atual

   Observação: o peso da aresta que liga o vértice inicial a seu antecessor é igual a 0.

   Modelo de Tabela:

Vértice	Antecessor	Peso

   Tabela que caracteriza a árvore geradora mínima do grafo exibido na primeira página desta lista:

Vértice	Antecessor	Peso
0	-1	0
1	3	3
2	3	1
3	0	3

5. Desenhe quatro **árvores**:

   (a) Duas que apresentem, para o primeiro grafo, as árvores geradoras mínimas resultantes dos experimentos 1 e 2. Elas devem conter todos os vértices do grafo original e o conjunto de arestas da árvore com seus respectivos pesos.

   (b) Duas que exibam, para o segundo grafo, as árvores geradoras mínimas obtidas durante as execuções dos experimentos 1 e 2. Elas devem mostrar todos os vértices do grafo original, o conjunto de arestas presentes na árvore e o peso de cada uma destas arestas.

6. Responda as seguintes questões relativas às quatro árvores geradoras mínimas (AGMs) ilustradas no item anterior:

   (a) As AGMs obtidas para determinado grafo no decorrer dos experimentos 1 e 2 tem o mesmo peso e quantidade de arestas? Por quê?

   (b) Para um mesmo grafo, as AGMs retornadas após a execução de cada um dos dois experimentos é igual? Porque?

   (c) Um grafo pode ter duas ou mais árvores geradoras mínimas diferentes? Porque?

   (d) Uma árvore geradora mínima é uma árvore binária? Porque?

Guarde o projeto contendo as classes Grafo e AlgoritmosEmGrafos para as aulas seguintes, pois ele será utilizado nas próximas listas de exercícios.