PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e Informática
Lista de Exercícios Prática – 2/2018
Valor: 5,0 pontos

Curso : *Sistemas de Informação*
Disciplina : *Algoritmos em Grafos*
Professora : *Michelle Nery Nascimento*

Regras Básicas:

- Todo o código deve ser desenvolvido na linguagem **C#**.
- Essa lista de exercícios pode ser resolvida em grupos de até **três componentes**.
- A **entrega** da resolução dessa lista de exercícios será feita exclusivamente pelo **SGA**, em pasta própria. Resoluções entregues após o prazo estabelecido não serão consideradas.
- Os alunos devem **comentar todos os métodos** implementados.
- O código deverá ser desenvolvido observando-se o formato de entrada especificado.
- Cópias de trabalho, de colegas ou de soluções da internet, se existirem, serão zeradas.

Tarefas:

Implementar os seguintes métodos, em uma **classe Grafo**:

1) Para **grafos não-dirigidos**:

- bool isAdjacente (Vertice v1, Vertice v2);
- int getGrau (Vertice v1);
- bool isIsolado (Vertice v1);
- bool isPendente (Vertice v1);
- bool isRegular ();
- bool isNulo ();
- bool isCompleto ();
- bool isConexo ();
- bool isEuleriano ();
- bool isUnicursal ();
- Grafo getComplementar ();
- Grafo getAGMPrim (Vertice v1): esse método deve retornar, para um grafo conexo, sua **Árvore Geradora Mínima** obtida por meio da aplicação do **algoritmo de Prim**. Nesse método, deve também ser impressa uma linha de saída contendo a ordem em que o algoritmo de Prim inseriu na AGM os vértices do grafo original, sendo que o vértice inicial corresponde ao vértice passado como parâmetro. Além disso, se tivermos duas arestas com o mesmo peso, escolha aquela cuja soma dos índices dos vértices seja menor. Se tivermos um novo empate, escolha aquela incidente ao vértice de menor índice.
- Grafo getAGMKruskal (Vertice v1): esse método deve retornar, para um grafo conexo, sua **Árvore Geradora Mínima** obtida por meio da aplicação do **algoritmo de Kruskal**. Nesse método, deve também ser impressa uma linha de saída contendo a ordem em que o algoritmo de Kruskal inseriu na AGM os vértices do grafo original, sendo que o vértice inicial corresponde ao vértice passado como parâmetro. Além disso, se tivermos duas arestas com o mesmo peso, escolha aquela cuja soma dos índices dos vértices seja menor. Se tivermos um novo empate, escolha aquela incidente ao vértice de menor índice.

- int getCutVertices (): esse método deve retornar, para um grafo conexo, seu número de *cut*-vértices.

2) Para **grafos dirigidos**:

- int getGrauEntrada (Vertice v1);
- int getGrauSaida (Vertice v1);
- bool hasCiclo ();

Formato do Arquivo de Entrada:

Para a realização dos testes e avaliação do código desenvolvido, será fornecido um arquivo texto de entrada que apresentará, na primeira linha, o **número de vertices** do grafo. As linhas seguintes desse arquivo de entrada conterão, cada uma, **informações sobre cada aresta** do grafo, no seguinte formato: **vértice v_1; vértice v_2; peso da aresta; direção da aresta**. Apenas grafos dirigidos apresentarão esse último valor em cada linha do arquivo de entrada. Se o valor desse parâmetro for 1, a aresta é direcionada de v_1 para v_2. Se o valor desse parâmetro for -1, a aresta tem a direção contrária, sendo direcionada, portanto, de v_2 para v_1.
Seguem exemplos de arquivos de entrada:

1) Grafo não-dirigido:

```
3
1; 2; 4
1; 3; 7
2; 3; 10
```

No exemplo acima, temos um grafo não-dirigido de três vertices. A aresta que conecta os vértives v_1 e v_2 tem peso 4; a aresta que conecta v_1 a v_3 tem peso 7; e a aresta que conecta os vertices v_2 e v_3 tem peso 10.

2) Grafo dirigido:

```
3
1; 2; 4; 1
1; 2; 11; -1
1; 3; 7; 1
2; 3; 10; -1
```

No exemplo acima, temos um grafo dirigido de três vertices. Há uma aresta dirigida do vértice v_1 ao vértice v_2 com peso 4; há uma aresta dirigida de v_2 para v_1 com peso 11; há uma aresta dirigida do vértice v_1 ao vértice v_3 com peso 7; e uma aresta dirigida do vértice v_3 ao vértice v_2 com peso 10.

Seu grupo deve criar seus próprios arquivos de entrada para testes, mas eles devem seguir o formato especificado acima, pois irei executar o código implementado com os meus arquivos de teste (nesse formato) durante a correção dessa lista de exercícios.