FCT/Unesp – Presidente Prudente
Teoria dos Grafos
Prof. Danilo Medeiros Eler

Trabalho Prático 01

Instruções de Envio: enviar para **daniloelerunesp@gmail.com** (por favor, enviar **somente** para esse e-mail).
No assunto do email você deve colocar: **[Grafos2019] Trabalho Prático 01.** Anexar o código fonte, executável e, se houver, dados utilizados. No corpo do email você deve identificar os integrantes do grupo, no caso do trabalho ter sido desenvolvido em dupla ou em trio.

DICA: não envie como ZIP ou RAR para o gmail não barrar o seu email. Você deve **renomear** o arquivo, por exemplo, trabalho01-fulanoCiclanoBeltrano.renomearParaZIP; ou então colocar em algum lugar para eu fazer download (github, google drive ou outros).

O trabalho deverá ser desenvolvido de forma individual ou em grupo de no máximo três pessoas. Essa regra é rígida e não deverá ser alterada.

Caso haja evidencia de **cópia** (de trabalhos deste ano ou de anos anteriores), os trabalhos envolvidos terão **nota zero**.

Data máxima para envio: O trabalho deve ser enviado por email até o dia **12/05/2019**.

Especificações do trabalho

1 – Implementar a representação computacional de Matriz de Adjacência e de Lista de Adjacência para a representação computacional de **grafos e dígrafos**. **Atenção**: todos os algoritmos implementados devem ser executados com qualquer um desses dois tipos de representação computacional de grafos. O seu programa deve disponibilizar para o usuário, na interface, a opção de escolher com qual representação ele irá trabalhar.

2 – Implementar a busca em profundidade para ser executada a partir de um nó raiz fornecido pelo usuário. **Em seguida**, exibir o tempo de chegada (cinza) e o tempo de finalização (preto) de cada vértice. Note que o **vértice inicial é escolhido pelo usuário**, mas a busca em profundidade varrerá todo o grafo (ou dígrafo).

3 – Implementar a busca em largura. O usuário deve fornecer o nó raiz para iniciar a busca em largura. **Após a busca ser realizada**, exibir o caminho percorrido para chegar em cada vértice do grafo e o tamanho do caminho (distância até a raiz). Para isso, utilize as estruturas auxiliares empregadas pela busca em largura.

4 – Implementar a verificação de um caminho entre dois vértices ***u*** e ***v***. O usuário indicará quais são esses vértices e escolherá entre a busca em profundidade e a busca em largura para verificar a existência desse caminho. Se existir o caminho entre ***u*** e ***v***, o programa deverá

exibir esse caminho; caso contrário, o programa dever informar que não há um caminho entre os vértices indicados.

5 – Implementar uma função para verificar se um **grafo** é conexo. Se ele não for conexo, deve ser apresentado o número de componentes conexas que formam o grafo. Exiba a qual componente cada vértice pertence. Note que esse exercício é só para grafo.

6 – Fazer um programa que implemente os algoritmos de Prim e de Kruskal para computar uma árvore geradora mínima em **grafos ponderados**. No caso do Prim, **o usuário escolherá o vértice inicial**. Note que esse exercício é só para grafo.

7 – Fazer um programa que implemente os algoritmos de Dijkstra e de Bellman-Ford para encontrar o caminho mínimo a partir de um vértice inicial escolhido pelo usuário. Após a execução, o programa deverá exibir o caminho mínimo desse vértice para todos os outros vértices do grafo e também a distância. Esse algoritmo deverá ser implementado para **dígrafos ou grafos ponderados**. O **usuário deverá escolher o vértice inicial**.

Considere que o **grafo** (ou o **dígrafo**) será lido de um arquivo texto (ver abaixo). A primeira linha do arquivo indica se é um grafo (0) ou um dígrafo (1); a segunda linha indica o número de vértices; e as demais linhas indicam as arestas – vértice inicial, vértice final e peso. Por favor, utilize esse formato de arquivo, pois é o que será utilizado para testar o seu programa Se o programa não ler o arquivo, não poderá ser testado.

Exemplo de formato do arquivo de entrada

```
-------------------------------------------------
0
5
0 2 4
0 4 60
0 3 23
2 3 4
3 1 10
4 2 15
-------------------------------------------------
```

Observações:

- Organize a comunicação com o usuário da melhor forma possível, tanto a entrada de dados como a saída, ou seja, faça uma boa interface. Utilize também menus para o usuário escolher dentre as diversas opções do seu programa.
    - Se julgar necessário, por favor, faça um manual simples indicando como utilizar o programa.
- As apresentações do resultado dos algoritmos deverão ser emitidas de modo textual, o que não impede a utilização de uma interface gráfica. Portanto, quem utilizar uma interface gráfica, poderá exibir as saídas em caixas de texto.

- Novamente, utilize interface com o usuário, pois o programa não deve ser testado com a adição de informações no seu código para ser compilado em seguida.