Universidade Federal de Juiz de Fora

Departamento de Ciência da Computação

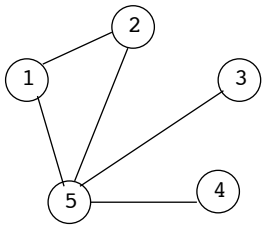
Disciplina: DCC059 Turma A 2016-3 – Prof. Stênio Sã.

Trabalho Prático: Parte 1

Desenvolva um TAD ou uma classe que implemente o conjunto de funcionalidades sobre Grafos descritas abaixo. Seu TAD ou classe deve ser capaz de representar grafos (possivelmente digrafos, ponderados ou não) utilizando matriz de adjacência ou lista de adjacência. Para definição do Grafo, você poderá precisar de outros TADs ou classes, como Nó e Aresta.

A entrada do algoritmo será através de arquivo texto, sendo formato dependente do tipo de grafo. Para grafos não ponderados, a entrada segue o formato abaixo:

Grafo



Arquivo de entrada

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

Seu algoritmo deve ser capaz de escrever um grafo em arquivo texto, no mesmo formato que a entrada indicada.

Seu algoritmo deve ser capaz de armazenar o grafo tanto através de lista de adjacência como através de matriz de adjacência, sendo que o usuário deverá especificar qual tipo de armazenamento escolherá qual estrutura.

Todo o código deve ser desenvolvido usando padrão Ansi. A biblioteca será compilada usando gcc ou g++ e a execução do programa será feita em ambiente Linux. Assim, evite includes com endereço absoluto. **Nota:** O professor/tutor não fará qualquer alteração no código na tentativa de fazer compilar o código.

Produto: ao final deste trabalho, deve ser enviado ao professor o que segue:

i) O código fonte (em C ou C++) do TAD ou da classe, que deve estar bem comentado/documentado (código sem documentação é penalizado na nota);

ii) Relatório conforme o modelo a ser enviado, que deve informar as decisões de projeto e de implementação (fundamentadas nas dificuldades enfrentadas), além do detalhamento de cada funcionalidade requerida neste documento. O relatório não deve ultrapassar 15 páginas, não deve ter listagem de código fonte e todo pseudocódigo inserido deve ter as linhas numeradas.

iii) O trabalho deve ser feito em grupo de no máximo quatro integrantes, sendo de responsabilidade do grupo a distribuição de tarefas e o compromisso de cada integrante. **A nota de cada membro depende de si e do grupo, mas a nota é individual, podendo acontecer de se ter diferentes notas entre membros de um mesmo grupo.**

Iv) O TAD ou classe Grafo deve ter os seguintes atributos mínimos:

n: número de nós (ou ordem do grafo);

m: número de arestas;

flagDir: indica se o grafo é direcionado ou não direcionado;

D: grau do grafo;

v) O TAD ou classe Nó deve ter os seguintes atributos mínimos:

d: grau (ou valência) do nó;

w: peso do nó;

vi) O TAD ou classe Aresta deve ter os seguintes atributos mínimos:

w: peso da aresta;

vii) O TAD ou classe Grafo deve apresentar as seguintes funcionalidades mínimas:

~~1 - incluir e excluir nó e aresta;~~

~~2- retornar o grau de nó passado no parâmetro;~~

~~3- Retornar a sequência de graus do grafo;~~

~~4 - verificar se o grafo é k-regular;~~

~~5 - verificar se o grafo é completo;~~

~~6 - verificar se dois nós são adjacentes;~~

~~7 - Busca em grafos: largura e profundidade. Seu algoritmo deve ser capaz de percorrer o grafo utilizando busca em largura e busca em profundidade. O vértice inicial será dado como parâmetro. A respectiva árvore de busca deve ser gerada, assim como o nível de cada vértice na árvore (nível da raiz é zero). Estas informações devem ser impressas em arquivo. Para descrever a árvore gerada, basta informar o pai de cada vértice e seu nível.~~

~~8 - verificar se o grafo é conexo;~~

~~9 - verificar se dois nós estão em uma mesma componente conexa;~~

~~10 – verificar se um dado nó é de articulação;~~

~~11 – verificar se uma dada aresta é ponte;~~

~~12 – retornar a vizinhança (aberta/fechada) de um dado nó;~~

13 – retornar o fechamento transitivo direto/indireto de um dado nó;

~~14 – Retornar a ordenação topológica do DAG;~~

15 – Determinar o caminho mínimo entre dois vértices usando algoritmo de Dijkstra;

~~16 – Determinar o caminho mínimo entre qualquer par de vértices usando o algoritmo de Floyd;~~

~~17 – Retornar o subgrafo induzido por um dado subconjunto de vértices;~~

~~18 – Retornar as componentes conexas do grafo;~~

20 – Retornar o grafo gerado pelo produto cartesiano entre o grafo e outro grafo;

21 – Retornar á árvore geradora mínima usando o algoritmo de PRIM;

20 – Retornar a árvore geradora mínima usando o algoritmo de Kruskal;

22 – Verificar se o grafo é k-conexo;

~~23 – Verificar se o grafo é eulerianos.~~

Bom trabalho!