Departamento de Ciência da Computação - UFJF
DCC059 - Teoria dos Grafos
Prof. Gabriel Souza

Trabalho Grafos Parte 1 Versão 2

Da Entrega

Deve ser enviado **pelo classroom da turma** em um único arquivo .zip até dia **11/02/2025** com a seguinte estrutura de pasta e arquivos **(e nomes dos arquivos)**:

```
TrabalhoGrafosGrupoX.zip
|       docs/
|       |       documentacao.pdf
|       |       apresentacao.pdf
|       |       contribuidores.pdf     (o que cada membro fez, junto com o link do github)
|       |       descricao.pdf          (este documento)
|       include/
|       |       *.h
|       |       *.hpp
|       |       *.tpp
|       src/
|       |       *.c
|       |       *.cpp
|       entradas/
|       |       *.txt
|       main.cpp
|       README      (explicação simples e considerações sobre a execução do programa)
```

Da apresentação

Será mantida a nota da apresentação já feita pela turma.

Departamento de Ciência da Computação - UFJF
DCC059 - Teoria dos Grafos
Prof. Gabriel Souza

Da avaliação

O trabalho será avaliado em 100 pontos e depois será ponderado conforme seu valor na nota final da disciplina.

O calculo da nota será dado por: $\sqrt{T * \sqrt{A * P}}$ onde A, T e P valem 100 pontos e representam:

- A - Apresentação
    - Clareza da apresentação
    - Escolha dos tópicos a apresentar
    - Resposta as perguntas do professor
    - Conhecimento sobre o assunto

- T - Avaliação Geral do trabalho
    - Estrutura de arquivos e organização (5 pts)
    - Uso correto de hierarquia (20 pts)
    - Código compilando e retornando as saidas esperadas (20 pts)
    - Estrutura e implementação nos arquivos *cpp, *tpp (25 pts)
    - Corretude de alocação e acesso de memoria (20 pts)
    - Documentação (10 pts)

- P - Participação do aluno no desenvolvimento do código

Departamento de Ciência da Computação - UFJF
DCC059 - Teoria dos Grafos
Prof. Gabriel Souza

Do código

Implementar uma classe abstrada “grafo” em C++ com 2 classes filhas para diferentes estruturas de armazenamento para nós e arestas:

- grafo_matriz - uso de matriz de adjacência para representar arestas (quando grafo não direcionado deve ser usado a representação linear de matriz triangular). Além disso, os vertices e arestas devem ser estáticos.

- grafo_lista - uso de lista encadeada tanto para vertices quanto arestas usando alocação dinamica.

Funções com implementação **exclusivamente** na classe abstrata:

- n_conexo - função que indica a quantidade de componentes conexas
- get_grau - função que retorna o grau do grafo
- get_ordem - função que retorna a ordem do grafo
- eh_direcionado - função que retorna se o grafo é direcionado ou não
- vertice_ponderado - função que informa se os vertices do grafo tem peso
- aresta_ponderada - função que informa se as arestas do grafo tem peso
- eh_completo - função que diz se um grafo é completo ou não
- carrega_grafo - função que lê um arquivo txt com um grafo e carrega ele

Observações:

- Outras funções podem (e devem) ser inclusas para melhor estruturação do código seguindo os preceitos de Orientação a Objetos.
- **Apenas as funções de acesso as estruturas de vertices e arestas devem estar nas classes filhas. Como por exemplo as funções get_vizinhos, get_arestas...**
- **(DICA)** Implemente a classe de lista encadeada de forma separada.
- Deve ser respeitado os conceitos de herança e encapsulamento durante o desenvolvimento do código.
- O uso de memória será verificado com o Valgrind.
- Bibliotecas permitidas: fstream, iostream, iomanip, cmath, cstdlib, cstdarg, ctime, string
- Os grafos não devem aceitar arestas multiplas ou laços

Departamento de Ciência da Computação - UFJF
DCC059 - Teoria dos Grafos
Prof. Gabriel Souza

Da execução

Após compilado, o código deve ser executado via terminal com as seguintes linhas de comando:

Caso 1: main.out -d -m grafo.txt

- Imprime descrição do grafo apos carregá-lo com matriz de adjacencia como no exemplo abaixo:

```
grafo.txt

Grau: 3
Ordem: 3
Direcionado: Sim
Vertices ponderados: Sim
Arestas ponderadas: Sim
Completo: Sim
```

Caso 2: main.out -d -l grafo.txt

- Mesma coisa que o caso 1, mas carregando o grafo com lista encadeada.

Dos arquivos

Grafo.txt

```
3 1 1 1 // numero de nos, direcionado, ponderado vertices, ponderado arestas
2 3 7 // peso dos nos (apenas se ponderado nos vertices)
1 2 6 // origem, destino, peso (peso apenas se ponderado na aresta)
2 1 4 // origem, destino, peso (peso apenas se ponderado na aresta)
2 3 -5 // origem, destino, peso (peso apenas se ponderado na aresta)
…
```

Obs.: o id dos vertices deve começar em 1, o arquivo não deve conter os comentarios do exemplo acima, eles são apenas descritivos para esse documento