Apresentação - Trabalho 2 de Grafos

UNIVERSIDADE FEDERAL DE JUIZ DE FORA

Isabela Salvador Romão - MAT 202165065AB

Isabella Mourão dos Santos Dias - MAT 202165066AC

Mateus Alves da Silva - MAT 202076023

Professor: Gabriel Henrique de Souza

Fevereiro 11, 2025

1 - Introdução

O objetivo do Trabalho 2 de Grafos é aprimorar a estrutura de grafos desenvolvida anteriormente, incorporando novas funcionalidades e refinamentos, tais como:

- Manipulação dinâmica do grafo, com adição e remoção de nós e arestas;
- Alocação dinâmica da matriz de adjacência;
- Recálculo de IDs ao remover nós na matriz, mantendo um grafo isomorfo;
- Cálculo da menor distância entre dois nós utilizando algoritmos eficientes.

2 - Funções Implementadas

Classe Grafo (Abstrata)

A classe Grafo fornece a interface base para a implementação dos grafos.

Principais métodos:

- novo_no(int id): Adiciona um novo nó ao grafo.
- nova_aresta(int origem, int destino, float peso = 1.0): Adiciona uma aresta.
- deleta_no(int id): Remove um nó e reorganiza a estrutura.
- deleta_aresta(int origem, int destino): Remove uma aresta.
- menor_distancia(int origem, int destino): Retorna a menor distância entre dois nós.

Classe GrafoMatriz

A matriz de adjacência agora é alocada dinamicamente, iniciando com tamanho 10x10 e dobrando sua capacidade quando necessário.

Métodos implementados:

- novo_no(int id): Adiciona um nó redimensionando a matriz se necessário.

- deleta_no(int id): Remove um nó e recalcula IDs.
- nova_aresta(int origem, int destino, float peso): Adiciona uma aresta.
- deleta_aresta(int origem, int destino): Remove uma aresta.
- menor_distancia(int origem, int destino): Utiliza Floyd-Warshall para encontrar o caminho mínimo.

Classe GrafoLista

A implementação via listas encadeadas permite operações eficientes.

Métodos implementados:

- novo_no(int id): Adiciona um novo nó.
- deleta_no(int id): Remove um nó e ajusta conexões.
- nova_aresta(int origem, int destino, float peso): Adiciona uma aresta.
- deleta_aresta(int origem, int destino): Remove uma aresta.
- menor_distancia(int origem, int destino): Utiliza Dijkstra para encontrar a menor distância.

3 - Restrições

- O grafo não aceita arestas múltiplas.
- O grafo não aceita laços.

4 - Desenvolvimento

Ferramentas e Linguagens Utilizadas:

- Linguagem: C++
- Compilador: g++
- Bibliotecas: fstream, iostream, cstdlib, string
- IDE: Visual Studio Code, Code Blocks, CLion

5 - Conclusão

Este projeto expandiu o nosso entendimento sobre estruturas dinâmicas de grafos, alocação de memória e algoritmos de busca e caminhos mínimos. A alocação dinâmica da matriz e a reestruturação de IDs fortaleceram conhecimentos sobre gestão eficiente de memória.