



PUC Minas

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE
MINAS GERAIS**

Instituto de Ciências Exatas e de Informática

Relatório de Grafos

Alessandra Faria Rodrigues¹

Enzo Marques Pylo²

Débora Luiza de Paula Silva³

Douglas Nicolas Silva Gomes⁴

Gabriel Chaves Mendes⁵

Guilherme Henrique da Silva Teodoro⁶

Maria Eduarda P. Martins⁷

Suzane Lemos de Lima⁸

Belo Horizonte, 30 de agosto de 2025

Resumo

Este relatório apresenta o desenvolvimento e a implementação de estruturas para a manipulação de grafos em linguagem C/C++. A implementação abrange quatro tipos distintos de grafos: (i) não direcionado não ponderado; (ii) não direcionado ponderado; (iii) direcionado não ponderado; e (iv) direcionado ponderado. São discutidos os detalhes da implementação, com ênfase na escolha da lista de adjacência como estrutura de dados para a representação dos grafos, e as funcionalidades básicas desenvolvidas para a manipulação de vértices e arestas em cada uma das abordagens.

Palavras-chave: Grafos. Lista de Adjacência. Grafos Ponderados. Grafos Direcionados.

1 Introdução

A representação de grafos é um conceito fundamental em Ciência da Computação, permitindo a modelagem de sistemas complexos e a relação entre diversas áreas. Um grafo é definido como um conjunto de vértices e arestas, cujas características, como direção e peso, determinam o tipo e a finalidade da estrutura. A implementação dessas diferentes variações de grafos requer a escolha de uma representação de dados eficiente, como a lista de adjacência, que impacta diretamente o desempenho e a funcionalidade para a resolução de problemas.

A principal diferença entre os tipos de grafos está na natureza das suas ligações. Um grafo não direcionado é um conjunto de vértices e arestas, onde as arestas não têm direção e conectam dois vértices de forma simétrica, permitindo a ligação entre os nós em ambos os sentidos. Por outro lado, em um grafo direcionado as ligações têm um sentido único, onde uma ligação de A para B não garante uma ligação de volta de B para A, semelhante a uma rua de sentido único.

Outra classificação dos grafos baseia-se na presença ou ausência de valores nas suas arestas. Em um grafo não ponderado, as arestas não possuem um valor numérico ou peso associado a elas. Já um grafo ponderado atribui um “peso” ou “custo” a cada aresta, que pode representar distância, tempo ou qualquer outra métrica. Esta característica é crucial para problemas de otimização, como encontrar o caminho mais curto entre dois pontos num mapa.

Neste trabalho, diversas responsabilidades foram distribuídas aos membros do grupo. Os integrantes Enzo Pylo, Débora Luiza, Douglas Nicolas e Guilherme Teodoro ficaram responsáveis pelo desenvolvimento do código e implementação dos grafos, os integrantes Gabriel Chaves e Maria Eduarda pelo desenvolvimento do relatório e as integrantes Alessadra Faria e Suzane Lemos pela revisão e alteração do relatório.

2 Metodologia e Implementação

Neste trabalho foi utilizada apenas uma única abordagem.

2.1 Abordagem por Lista de Adjacência

Esta abordagem de representação consiste em associar cada vértice do grafo a uma lista contendo todos os seus vizinhos, ou seja, aqueles diretamente conectados por uma aresta. Apesar de exigir uma busca sequencial para verificar a adjacência entre dois vértices específicos, sua principal vantagem está na eficiência de memória, especialmente para grafos esparsos (aqueles que têm o número de arestas proporcional ao número de vértices). Na implementação desta abordagem, são realizadas as seguintes operações:

- Utilização de um vetor ou arranjo principal, onde cada índice corresponde a um vértice do grafo.
- Para cada vértice v , o armazenamento de uma lista (geralmente uma lista encadeada) contendo os identificadores de todos os vértices u para cada aresta existente (v, u) .
- Em grafos ponderados, a lista armazena pares (vértice, peso), representando não apenas o vizinho, mas também o peso da aresta que os conecta.

2.2 Implementação

2.2.1 Grafo Direcionado Não Ponderado

Em um vetor, o índice representa o vértice de origem e cada índice contém uma lista armazenando todos os vértices de destino conectados por uma aresta. Ao chamar a função `adicionarAresta(v1, v2)`, ela cria uma aresta que conecta $v1$ até $v2$ e adiciona $v2$ à lista de $v1$, por meio da função `adj[v1].push_back(v2)`, existindo assim um caminho de $v1$ para $v2$, mas não ao contrário.

2.2.2 Grafo Direcionado Ponderado

Assim como o anterior, em um vetor o índice representa um vértice de origem e cada índice contém uma lista armazenando todos os vértices de destino. Entretanto, ao adicionar uma nova aresta, `adicionarAresta(v1, v2, peso)`, um novo parâmetro (peso) é passado para a lista de adjacência de $v1$, armazenando um par contendo o vértice de destino e o peso da aresta: `adjPeso[v1].emplace_back(v2, peso)`.

2.2.3 Grafo Não Direcionado Não Ponderado

Para a implementação desse grafo, quando é chamada a função `adicionarAresta(v1, v2)`, são adicionadas duas arestas direcionadas: uma de $v1$ para $v2$ e outra de $v2$ para $v1$, por meio das funções `adj[v1].push_back(v2)` e `adj[v2].push_back(v1)`. Assim, simula-se um grafo não direcionado.

2.2.4 Grafo Não Direcionado Ponderado

Assim como o grafo não direcionado não ponderado, são adicionadas duas arestas direcionadas, uma de $v1$ para $v2$ e outra de $v2$ para $v1$. Entretanto, é passado um parâmetro a mais (peso), adicionando o par $(v2, peso)$ na lista de $v1$ e o par $(v1, peso)$ na lista de $v2$.

3 Conclusão

Este trabalho concluiu a implementação de quatro tipos de grafos, demonstrando que a representação por lista de adjacência é uma abordagem eficaz. A estratégia de reutilizar a estrutura do grafo direcionado para construir o não direcionado mostrou-se uma solução prática e de fácil desenvolvimento para a construção dos grafos.

Referências

- <https://pt.stackoverflow.com/questions/216105/todos-poss%C3%ADveis-caminhos-em-grafos>
- <https://www.geeksforgeeks.org/dsa/graph-and-its-representations>
- GOODRICH, M. T.; TAMASSIA, R.; GOLDWASSER, M. H. Data Structures and Algorithms in Python (Estruturas de Dados e Algoritmos em Python). 2^a ed. Wiley, 2013
- <https://www.facom.ufu.br/~madriana/ED2/roteirolab2.pdf>
- <https://blog.tiagomadeira.com/2006/01/grafos-ponderados>
- GOMES, Paulo César Rodacki. Grafos: conceitos fundamentais, algoritmos e aplicações. Blumenau, SC: Editora do Instituto Federal Catarinense, 2022