Relatório Técnico – Etapa 1 do Projeto de Grafos

Disciplina: Algorítmos Grafos – GCC218

Alunos: Helder Ávila , Lucas Reis Silvino

Data: Abril de 2025

Arquivo de entrada: DI-NEARP-n422-Q8k.dat

1. Apresentação

Este relatório documenta o desenvolvimento da Etapa 1 do projeto prático da disciplina de Grafos, cujo objetivo é representar e analisar uma rede de transporte utilizando estruturas de dados baseadas em grafos.
A rede em questão é modelada a partir de um arquivo de dados que segue o padrão do Capacitated Arc Routing Problem (CARP), estendido para permitir a presença de vértices, arestas e arcos com ou sem demanda.
Esta etapa tem um papel fundamental no projeto como um todo, pois oferece a base estrutural e analítica para as próximas fases, que envolvem algoritmos de otimização e rotas em grafos mistos.

2. Objetivos da Etapa

O foco principal desta etapa foi montar uma base sólida de dados em forma de grafo para análises futuras. Pra isso, o trabalho se dividiu em algumas partes:
- Interpretar corretamente o arquivo de entrada e identificar seus diferentes elementos;
- Representar o grafo com estruturas adequadas e eficientes (listas de adjacência);
- Implementar algoritmos para calcular estatísticas essenciais do grafo;
- Construir as matrizes de caminhos mais curtos e predecessores;
- Validar os resultados com métodos alternativos.

3. Organização dos Dados

3.1 Tipos de Elementos PresentesO grafo representado neste projeto é um grafo misto, ou seja, contém:
- Arestas (conexões bidirecionais entre vértices)
- Arcos (conexões unidirecionais)
- Vértices com ou sem demanda
- Elementos requeridos (devem obrigatoriamente ser considerados)
- Elementos opcionais (não obrigatórios, mas disponíveis)

3.2 Estrutura do Arquivo

O arquivo de entrada está dividido em seções com cabeçalhos específicos, tais como:
- #Nodes: — número total de vértices
- ReN. — vértices com demanda
- ReE. — arestas requeridas
- ReA. — arcos requeridos
- EDGE — arestas opcionais
- ARC — arcos opcionais

Cada linha contém os dados estruturados com posições fixas, geralmente contendo o identificador, vértices envolvidos, custo de percurso, demanda e custo de serviço.

4. Estrutura de Implementação

4.1 Linguagem e Estilo

O projeto foi implementado em C++, com foco em modularidade, clareza e eficiência. As estruturas de dados e algoritmos foram organizados em funções separadas para facilitar manutenção e expansão.

4.2 Estruturas Criadas

Foram definidas duas estruturas principais para representar os elementos do grafo:
struct Aresta { int de, para, custo, demanda, custo_servico; };
struct Arco   { int de, para, custo, demanda, custo_servico; };

Além disso, foram utilizadas listas de adjacência para modelar a topologia do grafo:
- adj_arestas → para arestas (não direcionadas)
- adj_arcos → para arcos (direcionadas)

5. Leitura e Processamento do Arquivo

A função lerArquivo() faz a leitura do arquivo DI-NEARP-n422-Q8k.dat, linha por linha, identificando a seção atual por meio de uma enumeração de estado.
Cada tipo de linha é processado de forma específica e os dados são armazenados em suas respectivas estruturas. A leitura é robusta e trata corretamente rótulos como E1, A5, N8, ignorando-os quando necessário.
A leitura também realiza a contagem e classificação de elementos (requeridos e opcionais), armazenando-os em vetores globais para uso posterior.

6. Cálculo das Estatísticas

Após a construção do grafo, foram implementadas funções para calcular as seguintes métricas estruturais:
- Quantidade de vértices
- Quantidade total de arestas (requeridas + opcionais)
- Quantidade total de arcos
- Quantidade de vértices requeridos
- Quantidade de arestas requeridas
- Quantidade de arcos requeridos
- Densidade do grafo
- Componentes conectados
- Grau mínimo
- Grau máximo
- Intermediação
- Caminho médio
- Diâmetro do grafo

7. Algoritmos Utilizados

7.1 Floyd-Warshall

Foi implementado para gerar a matriz de caminhos mínimos (dist[i][j]) e predecessores (pred[i][j]), utilizando os custos reais das arestas e arcos. Essa matriz é usada em várias análises posteriores.

7.2 DFS (Busca em Profundidade)

Usada para detectar componentes conectados no grafo não direcionado, verificando se todos os vértices estão interligados.

8. Conclusão

A Etapa 1 foi implementada com sucesso. O grafo foi representado corretamente a partir do arquivo de entrada, todas as estatísticas foram calculadas corretamente, e os algoritmos aplicados foram adequados. O sistema está preparado para as próximas etapas do projeto, como planejamento de rotas e otimização. O código é modular, confiável e validado com ferramentas externas.