**FACULDADE DE TECNOLOGIA JOSÉ CRESPO GONZALES**

**ANALISE E DESENVOLVIMENTO DE SISTEMAS**

ISADORA CORREA DE CARVALHO

MARIA JULIA LOUREIRO DA COSTA ALVES

MIKE WILLY FRANGUELLI

RAFAELA SILVA RIGANTI

RONALDO ARAUJO SILVA JUNIOR

SOPHIA MARCELINO DE ARAUJO

**RELATÓRIO DO PROJETO:**

**ANÁLISE DE REDES DE TRANSPORTE**

Sorocaba - SP

2025

ISADORA CORREA DE CARVALHO

MARIA JULIA LOUREIRO DA COSTA ALVES

MIKE WILLY FRANGUELLI

RAFAELA SILVA RIGANTI

RONALDO ARAUJO SILVA JUNIOR

SOPHIA MARCELINO DE ARAUJO

**RELATÓRIO DO PROJETO:**

**ANÁLISE DE REDES DE TRANSPORTE**

Sorocaba - SP

2025

**SUMÁRIO**

[1 INTRODUÇÃO 4](#_Toc212064635)

[2 COMPLEXIDADE 4](#_Toc212064638)

[3 RESULTADOS OBTIDOS 6](#_Toc212064648)

[4 REPRESENTAÇÃO GRÁFICA 8](#_Toc212064649)

[REFERÊNCIAS 9](#_Toc212064650)

# 1 INTRODUÇÃO

# Este projeto tem como objetivo realizar a análise de uma rede de transporte urbano, representada por um grafo ponderado, utilizando código em Java.

# A aplicação lê arquivos contendo as estações (nós) e conexões (arestas) da rede, constrói a estrutura do grafo e aplica algoritmos de busca e caminho mínimo para extrair informações relevantes, como rotas mais curtas e distâncias totais.

Os dados são fornecidos em dois arquivos:

estacoes.txt: contém o identificador e o nome de cada ponto da rede.

conexoes.txt: define as conexões entre as estações e o peso (distância) de cada ligação.

# 2 COMPLEXIDADE

# No projeto Análise de Redes de Transporte, foram implementadas rotinas que constroem e manipulam um grafo ponderado, aplicando algoritmos clássicos como Busca em Largura (BFS) e Dijkstra para exploração e cálculo de rotas.

# Construção do Grafo

# O processo de leitura dos arquivos estacoes.txt e conexoes.txt realiza a criação de cada estação (vértice) e a definição das conexões (arestas) com seus respectivos pesos.

# Esse procedimento percorre todas as linhas dos arquivos uma única vez.

# Complexidade Temporal: O(V + E)

# Cada estação (V) e conexão (E) é processada apenas uma vez, o que garante tempo linear.

# Complexidade Espacial: O(V + E)

# O armazenamento das listas de adjacência ocupa espaço proporcional ao total de elementos da rede.

# Esse tempo linear torna o processo de leitura e construção altamente eficiente, mesmo quando o número de estações e conexões aumenta significativamente.

Busca em Largura (BFS – Breadth-First Search)

A BFS é utilizada para explorar a rede e verificar a conectividade entre as estações. Ela percorre o grafo de forma sistemática, expandindo todos os nós vizinhos antes de avançar para o próximo nível.

Complexidade Temporal: O(V + E)

Cada estação e cada conexão é visitada no máximo uma vez.

Complexidade Espacial: O(V)

A estrutura de fila e o vetor de “visitados” ocupam espaço proporcional ao número de vértices.

Esse algoritmo é extremamente eficiente para verificar se o sistema de transporte é totalmente conectado isto é, se existe um caminho entre quaisquer duas estações.

Além disso, a BFS serve como base para análises de alcance e componentes conectados, úteis em redes maiores, como sistemas metroviários ou rodoviários urbanos.

Algoritmo de Dijkstra

O Dijkstra é o núcleo do projeto, sendo responsável pelo cálculo do menor caminho entre duas estações, considerando as distâncias como pesos positivos. Ele mantém um conjunto de estações já visitadas e atualiza continuamente as menores distâncias conhecidas, utilizando uma fila de prioridade.

Complexidade Temporal:

O((V + E) log V) quando implementado com PriorityQueue, estrutura utilizada no código Java.

O termo logarítmico decorre das operações de inserção e atualização de prioridade.

Complexidade Espacial: O(V)

São armazenados vetores de distâncias, predecessores e o heap de prioridades.

O Dijkstra é um algoritmo altamente eficiente para grafos esparsos (poucas conexões por vértice) e garante precisão absoluta no cálculo de rotas.

No contexto de transporte, ele representa a base de sistemas de navegação, como o cálculo de trajetos em aplicativos de mobilidade urbana.

# 3 RESULTADOS OBTIDOS

O desempenho do sistema é diretamente proporcional ao tamanho e densidade da rede — ou seja, ao número de estações (V) e conexões (E).

Em redes pequenas, como a deste projeto (8 nós e 10 conexões), a execução ocorre de forma imediata.

Porém, à medida que a rede cresce, alguns fatores passam a influenciar o desempenho:

* Crescimento linear na leitura e construção do grafo:
* O tempo de leitura aumenta proporcionalmente ao número de entradas nos arquivos.
* O impacto é pequeno, pois a operação é linear e facilmente paralelizável.

Escalabilidade da BFS:

Permanece eficiente mesmo em grafos com milhares de nós, já que sua complexidade é O(V + E).

É ideal para análise de conectividade e mapeamento de regiões acessíveis.

Custo computacional do Dijkstra:

É o algoritmo com maior custo no sistema, crescendo com O((V + E) log V).

Em redes densas (muitos caminhos possíveis), o número de operações cresce rapidamente, exigindo estruturas de dados otimizadas.

Otimizações possíveis:

A (A-estrela): introduz heurísticas que estimam a distância ao destino, reduzindo o número de nós visitados.

Dijkstra com heap de Fibonacci: melhora o tempo teórico para O(E + V log V).

Busca bidirecional: executa Dijkstra a partir do ponto de origem e do destino simultaneamente, reduzindo o espaço de busca.

O sistema desenvolvido é altamente eficiente em redes de pequeno e médio porte, como sistemas municipais de transporte ou redes logísticas regionais.

Com o uso de estruturas otimizadas, ele pode ser ampliado para redes maiores sem comprometer a precisão dos resultados.

# 4 REPRESENTAÇÃO GRÁFICA

|  |
| --- |
|  |

# REFERÊNCIAS

OLIVEIRA, Lucas. **Entendendo o Algoritmo Breadth-first search (BFS)**. DEV Community. Disponível em: <https://dev.to/lucas\_oliveira\_b978d8c21c/entendendo-o-algoritmo-breadth-first-search-bfs-43h8>. Acesso em: 23 out. 2025.

DIAS, Cayo. **Algoritmo de caminho de custo mínimo de Dijkstra - uma introdução detalhada e visual**. freeCodeCamp.org. Disponível em: <https://www.freecodecamp.org/portuguese/news/algoritmo-de-caminho-de-custo-minimo-de-dijkstra-uma-introducao-detalhada-e-visual/>. Acesso em: 23 out. 2025.

‌

‌

‌

‌