Relatório do trabalho da disciplina de Estrutura de Dados Avançada

# Projeto EDA Fase Final

Maksym Yavorenko - 25989

Engenharia de Sistemas Informáticos

Maio de 2025

# Índice

# Introdução

O presente capítulo pretende apresentar o enquadramento do projeto, a sua motivação, objetivos gerais, a metodologia aplicada e o plano de desenvolvimento seguido.

## Resumo

Este relatório apresenta o desenvolvimento de um projeto dividido em duas fases, com o objetivo de representar e analisar uma rede de antenas distribuídas numa cidade. Na Fase 1, foram aplicadas estruturas de listas ligadas para gerir antenas numa matriz lida de ficheiro. Na Fase 2, o sistema foi expandido para um grafo dirigido, permitindo a análise de conexões e percursos entre antenas com base na distância Manhattan. Foram implementados algoritmos como DFS e BFS, bem como funcionalidades para análise de caminhos e interações entre tipos distintos. O trabalho resultou numa aplicação modular, documentada e preparada para extensões futuras.

## Motivação

O projeto proposto na unidade curricular de Estruturas de Dados Avançadas foi dividido em duas fases complementares, com o objetivo de simular a análise e organização de antenas espalhadas numa cidade.

Na Fase 1, foi desenvolvido um programa em C que utiliza listas ligadas para representar as antenas detetadas numa matriz lida de um ficheiro. Cada antena armazena as suas coordenadas, tipo e frequência de ressonância, sendo possível realizar operações como inserção, pesquisa e listagem dos dados.

Na Fase 2, a estrutura foi evoluída para um grafo dinâmico, onde cada antena passa a ser um vértice, e ligações entre antenas do mesmo tipo são representadas como arestas dirigidas, com base na distância de Manhattan. Esta estrutura permite aplicar algoritmos de percurso (DFS e BFS), análise de caminhos múltiplos entre duas antenas, e identificação de possíveis interferências entre antenas de tipos distintos.

Estas duas fases complementam-se: a primeira foca-se na gestão e leitura da informação, e a segunda na exploração relacional e topológica entre as antenas. Em conjunto, oferecem uma solução robusta e escalável para representar e analisar uma rede urbana de comunicações.

## Objetivos

- Implementar uma estrutura de dados dinâmica com listas ligadas para representar uma matriz de antenas.
- Evoluir essa estrutura para um grafo dirigido, permitindo a análise de conectividade e percursos.
- Aplicar algoritmos clássicos de grafos (DFS, BFS, caminhos múltiplos).
- Desenvolver uma solução modular, documentada e preparada para expansão.

## Metodologia de Trabalho

A metodologia seguida baseou-se num desenvolvimento incremental e modular. Na Fase 1, foram implementadas estruturas e funções fundamentais de leitura e manipulação de dados com listas ligadas. Na Fase 2, a estrutura foi reformulada como grafo, sendo aplicados algoritmos de percurso e análise relacional. O código foi dividido em módulos bem definidos, seguindo boas práticas de programação em C e documentado com Doxygen. As funcionalidades foram testadas com ficheiros de exemplo representando cenários reais de distribuição de antenas.

# Estado da Arte

Este capítulo apresenta o enquadramento teórico e prático do projeto, com base nas estruturas de dados utilizadas e nas soluções técnicas existentes.

## Estruturas de Dados Relevantes

As listas ligadas são estruturas fundamentais para armazenamento dinâmico de dados, permitindo inserções e remoções eficientes. Já os grafos são essenciais para modelar relações entre elementos, sendo amplamente utilizados em redes de comunicação, transporte e análise de caminhos. A representação por listas de adjacência oferece eficiência tanto em memória quanto em tempo de acesso.

## Soluções Técnicas Existentes

Diversos algoritmos clássicos são utilizados para análise de grafos. O DFS (Depth-First Search) e o BFS (Breadth-First Search) são os mais comuns para percorrer grafos. Ambos foram adaptados neste projeto para explorar caminhos entre antenas. Além disso, técnicas como backtracking e uso de distâncias Manhattan foram aplicadas para permitir uma análise mais realista da conectividade urbana.

Em projetos académicos e aplicações práticas, bibliotecas como NetworkX (Python) ou Boost Graph (C++) são comumente usadas para manipular grafos. Este projeto, no entanto, optou por uma implementação manual e educacional em linguagem C, permitindo maior compreensão e controle sobre as estruturas.

# Fase 1 – Listas Ligadas

## Desenvolvimento

O programa oferece diversas funcionalidades para a gestão das antenas na matriz. A seguir, estão as principais operações que podem ser realizadas:

### Inserção de Antenas

A função insert_antenna permite adicionar novas antenas na matriz. Cada antena possui coordenadas (x, y), um tipo representado por um caractere e um valor de ressonância.

### Remoção de Antenas

A função delete_antenna permite remover uma antena específica da matriz, a partir de suas coordenadas.

### Deteção e Remoção de Antenas Nefastas

A função detect_nefasto identifica elementos "nefastos" na matriz com base nas antenas que estão presentes.

A função remove_nefasto é uma função auxiliar para remover os nefastos calculados anteriormente, para melhor precisão na deteção de alterações.

## Impressão da Matriz

A função print_matrix imprime no terminal a matriz atualizada, representando as antenas com letras maiúsculas e os nefastos com '#' e espaços vazios com '.' .

## Impressão das Antenas

A função print_antennas exibe todas as antenas existentes na matriz, listando suas coordenadas e tipos, facilitando a visualização dos elementos presentes.

## Leitura da Matriz de um Ficheiro

A função read_matrix_from_file permite carregar uma matriz a partir de um ficheiro de texto.

## Libertação de Memória

Para evitar vazamento de memória, a função deallocate é utilizada para desalocar todas as estruturas antes do encerramento do programa.

# Fase 2 – Grafos

## Desenvolvimento

O programa oferece diversas funcionalidades para a gestão das antenas utilizando grafos. A seguir, estão as principais operações que podem ser realizadas:

### Representação do Grafo

O grafo é representado através de:

- Uma lista ligada de vértices (Vertex), cada um correspondendo a uma antena;
- Um vetor de listas de adjacência (Edge** adjList), onde cada posição representa as conexões do vértice correspondente.

Cada vértice armazena: ID único, coordenadas (linha, coluna) e o tipo de antena ('A', '0', etc.).

### Busca em Profundidade (DFS)

A função *dfs* realiza uma procura em profundidade a partir de uma antena indicada por coordenadas. As antenas visitadas são impressas na ordem em que são alcançadas, explorando o caminho até o fim antes de retroceder. A implementação utiliza recursão e um vetor de marcação *visited*.

### Busca em Largura (BFS)

A função *bfs* executa a procura em largura, explorando todas as antenas adjacentes antes de prosseguir para o próximo nível. A estrutura de dados usada é uma fila (*Queue*).

### Caminhos Possíveis entre Antenas

A função *find_all_paths* enumera todos os caminhos possíveis entre duas antenas, considerando apenas caminhos dirigidos e conexões entre antenas do mesmo tipo. Usa DFS com

backtracking, armazenando e imprimindo cada caminho completo como uma sequência de coordenadas.

### Interseção de Antenas com Tipos Diferentes

A função *find_intersections* permite verificar todas as possíveis interações entre antenas de tipos distintos, dentro de uma determinada distância Manhattan. Para cada par que satisfaz o critério, são listadas as coordenadas e a distância entre elas. Essa função é útil para visualizar áreas de potencial interferência.

### Libertação de Memória

A função *free_graph* desaloca toda a memória associada ao grafo, incluindo vértices, listas de adjacência e estruturas auxiliares, prevenindo vazamentos de memória.

## Conclusão

Este projeto, dividido em duas fases complementares, proporcionou uma abordagem prática e progressiva à manipulação de dados utilizando estruturas dinâmicas na linguagem C, com aplicação ao contexto de antenas distribuídas numa cidade.

Na Fase 1, foi implementado um sistema baseado em listas ligadas, capaz de representar, armazenar e manipular antenas numa matriz lida de ficheiro. Nesta fase, o foco esteve na estruturação da informação, permitindo a organização e pesquisa eficiente dos dados.

Na Fase 2, essa estrutura foi estendida para um grafo dirigido, onde cada antena passou a ser um vértice e as ligações entre antenas do mesmo tipo foram representadas como arestas condicionadas pela distância de Manhattan. Foram implementadas funcionalidades avançadas como percursos DFS e BFS, listagem de todos os caminhos possíveis entre duas antenas e deteção de interações entre antenas de tipos diferentes.

- Durante a realização deste trabalho, foram adquiridos os seguintes conhecimentos:
- Definição e manipulação de estruturas de dados dinâmicas, como listas ligadas e grafos;
- Utilização de listas de adjacência para representar grafos dirigidos;
- Implementação de algoritmos de percurso em grafos (DFS e BFS);
- Resolução de problemas com recursividade e backtracking (ex: todos os caminhos possíveis);
- Aplicação de distâncias Manhattan para modelar conexões no grafo;
- Leitura e parsing de ficheiros de texto para construção dinâmica de estruturas;
- Modularização do código em ficheiros .h e .c, com reutilização de funções;

- Gestão de memória dinâmica: alocação, libertação e prevenção de fugas de memória;
- Criação de documentação técnica com Doxygen para maior clareza e manutenção futura.

Todas as funcionalidades propostas foram implementadas com sucesso, cumprindo os objetivos da unidade curricular. O projeto encontra-se estruturado, modular e preparado para futuras extensões, como interface gráfica, exportação de resultados ou visualização de grafos. Esta experiência revelou-se essencial para o domínio prático das estruturas fundamentais de zados e sua aplicação em contextos reais.

## Link Repositório

https://github.com/Maksym2204/ProjetoEDA.git

## Bibliografia


- CProgramming. (2023). Short introduction to linked lists in C [Vídeo]. YouTube. Disponível em: Short introduction to linked lists in C
- CProgramming. (2023). Iterating over a linked list in C [Vídeo]. YouTube. Disponível em: Iterating over a linked list in C
- CProgramming. (2023). Adding elements to a linked list [Vídeo]. YouTube. Disponível em: Adding elements to a linked list
- CProgramming. (2023). Deallocating a linked list [Vídeo]. YouTube. Disponível em: Deallocating a linked list
- CProgramming. (2023). Removing an element from a linked list [Vídeo]. YouTube. Disponível em: Removing an element from a linked list
- Ghosh, A. (2023). Graphs in C – Basics and Implementation. GeeksforGeeks. Disponível em: https://www.geeksforgeeks.org/graph-and-its-representations
- Saini, P. (2023). Breadth-First Search Algorithm in C. GeeksforGeeks. Dispnível em: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph
- Tiwari, P. (2023). Depth-First Search with Recursion in C. GeeksforGeeks. Disponível em: https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph
- GFG Editorial. (2023). Pathfinding with DFS. GeeksforGeeks. Disponível em: https://www.geeksforgeeks.org/find-paths-given-source-destination
- W3Schools. (2023). C Programming: Dynamic Memory Management. Disponível em: https://www.w3schools.com/c/c_memory.asp
- Williams, B. (2022). BFS & DFS Animated Explanation [Vídeo]. YouTube. Disponível em: 5.1 Graph Traversals - BFS & DFS -Breadth First Search and Depth First Search