Trabalho Prático - Fase 1 | Fase 2

# 

**Estruturas de Dados Avançadas**

**Docente: Luís Gonzaga Martins Ferreira**

**Aluno: Rodrigo Lopes Ferreira Nº 31525**

Índice

[Índice de Figuras 3](#_Toc198505842)

[Resumo 4](#_Toc198505843)

[1 - Introdução 5](#_Toc198505844)

[1.1 - Motivação 5](#_Toc198505845)

[1.2 - Enquadramento 5](#_Toc198505846)

[1.3 - Objetivos 5](#_Toc198505847)

[1.4 - Metodologia de Trabalho 6](#_Toc198505848)

[1.5 - Plano de Trabalho 6](#_Toc198505849)

[1.6 - Estrutura do Documento 6](#_Toc198505850)

[2 - Estado da Arte, Enquadramento Teórico e Prático 7](#_Toc198505851)

[2.1 - Fundamentos Teóricos e Tecnologias a Explorar 7](#_Toc198505852)

[2.1.1 – Listas Ligadas 7](#_Toc198505853)

[2.1.2 – Estrutura de Grafos 8](#_Toc198505854)

[2.1.3 – Algoritmos de Procura 8](#_Toc198505855)

[2.1.4 – Interferência de Sinal e Efeito Nefasto 8](#_Toc198505856)

[2.2 - Soluções Técnicas Existentes e Análise Comparativa 9](#_Toc198505857)

[3 - Trabalho Desenvolvido 9](#_Toc198505858)

[3.1 - Fase 1 9](#_Toc198505859)

[3.1.1 - Análise e Especificação 10](#_Toc198505860)

[3.1.3 - Implementação 12](#_Toc198505861)

[3.2 - Fase 2 12](#_Toc198505862)

[3.2.2 – Análise e Especificação 13](#_Toc198505863)

[3.2.3 - Análise e Especificação 14](#_Toc198505864)

[3.3 - Tecnologias e Ferramentas Utilizadas 14](#_Toc198505865)

[3.3 - Organização do Código 15](#_Toc198505866)

[4 - Análise e Discussão de Resultados 16](#_Toc198505867)

[5 - Conclusão 17](#_Toc198505868)

[6 - Glossário e Siglas 18](#_Toc198505869)

[7 - Referências 19](#_Toc198505870)

# Índice de Figuras

[Figura 1 - Quadro das classificações das abordagens 9](#_Toc198506993)

[Figura 2 - Lista Ligada 10](#_Toc198506994)

[Figura 3 - Estrutura Antena 10](#_Toc198506995)

[Figura 4 - Estrutura Efeito Nefasto 11](#_Toc198506996)

[Figura 5 - Output das funções *1* 11](#_Toc198506997)

[Figura 6 - Lista de adjacências 11](#_Toc198506998)

# Resumo

O presente trabalho teve como objetivo principal o desenvolvimento de uma aplicação modular em linguagem ***C*** para representação e análise de redes de antenas com base em estruturas de dados dinâmicas. O problema consistia em identificar potenciais interferências entre antenas com a mesma frequência, posicionadas em diferentes coordenadas de uma grelha.

A solução foi concebida em duas fases distintas. A Fase 1 focou-se na leitura dos dados de antenas a partir de um ficheiro, utilizando listas ligadas para armazenamento e operações como a deteção de localizações com efeito nefasto.

Na Fase 2 foi feita a transição para uma abordagem baseada em grafos, onde cada antena passou a ser representada como um vértice, e as ligações entre antenas como arestas. Nesta fase foram implementados algoritmos clássicos como ***Depth-First Search (DFS)*** e ***Breadth-First Search (BFS)***, bem como outras funcionalidades semelhantes às da Fase 1.

Este relatório documenta a metodologia aplicada, as decisões técnicas tomadas e os resultados obtidos. Conclui-se que a transição de listas para grafos trouxe vantagens consideráveis em termos de escalabilidade e clareza no assunto de estruturas de dados dinâmicas.

# 1 - Introdução

O presente capítulo pretende introduzir o problema abordado neste trabalho, o seu contexto académico, os objetivos estabelecidos, a metodologia de desenvolvimento, o plano de trabalho seguido, e ainda a estrutura do documento.

## 1.1 - Motivação

O aumento da densidade de dispositivos de comunicação tem vindo a tornar mais evidente a importância da gestão eficiente de redes de antenas. Antenas com a mesma frequência, localizadas em posições próximas, podem interferir entre si, comprometendo o desempenho da rede.

O desenvolvimento de ferramentas que permitam representar, analisar e simular estas situações, recorrendo a estruturas de dados adequadas, permite melhorar a compreensão do problema e contribuir para a sua resolução. Este projeto procura responder a essa necessidade, aplicando conceitos de estruturas de dados avançadas no desenvolvimento de uma aplicação capaz de identificar interferências e relações entre antenas.

## 1.2 - Enquadramento

O trabalho foi realizado no âmbito da unidade curricular de Estruturas de Dados Avançadas (EDA), e a proposta do mesmo consiste na construção de um sistema dividido em duas fases. A primeira focou-se na utilização de listas ligadas para a análise de antenas a partir de ficheiro. A segunda fase introduziu estruturas de grafos, permitindo uma análise mais abrangente das interligações entre antenas.

## 1.3 - Objetivos

O principal objetivo deste projeto foi desenvolver uma aplicação modular em linguagem C para representar e analisar redes de antenas. Entre os objetivos específicos, destacam-se:

* Representar antenas com listas ligadas e grafos;
* Identificar coordenadas repetidas e localizações com efeito nefasto;
* Aplicar algoritmos de pesquisa em grafos, como DFS e BFS;
* Identificar caminhos possíveis entre duas antenas;
* Listar intersecções entre antenas com frequências distintas na mesma posição;
* Guardar o grafo em formato binário para posterior leitura.

## 1.4 - Metodologia de Trabalho

O trabalho foi desenvolvido de forma incremental. Na primeira fase, implementaram-se operações básicas sobre listas ligadas, como a leitura de ficheiro, inserção ordenada e verificação de repetições.

Na segunda fase, os dados passaram a ser representados com grafos, utilizando listas de adjacência para modelar as ligações entre antenas. Foram implementadas funções para percorrer o grafo, identificar caminhos e listar relações entre antenas com diferentes frequências.

Todo o código foi organizado modularmente e acompanhado de documentação gerada com ***Doxygen***.

## 1.5 - Plano de Trabalho

O desenvolvimento decorreu em duas fases:

* **Fase 1** – Listas Ligadas: leitura do ficheiro de antenas, deteção de coordenadas repetidas, identificação de localizações nefastas.
* **Fase 2** – Grafos: representação das antenas como vértices, ligação entre antenas da mesma frequência, execução de pesquisas em largura e profundidade, identificação de caminhos e interseções, e exportação binária do grafo.

## 1.6 - Estrutura do Documento

O presente relatório está organizado da seguinte forma:

* **Resumo** – Apresenta uma visão global do trabalho, destacando o problema abordado.
* **Capítulo 1** **– Introdução** – Introduz o tema e o contexto do trabalho, incluindo a motivação, enquadramento, objetivos, metodologia, plano de trabalho e a atual estrutura do documento.
* **Capítulo 2** **– Estado da Arte** – Explora os fundamentos teóricos e práticos relevantes.
* **Capítulo 3** **– Trabalho Desenvolvido** – Descreve as etapas práticas do projeto, sendo dividido em duas secções principais: a análise e especificação do problema, e a implementação técnica da solução.
* **Capítulo 4** **– Análise e Discussão de Resultados** – Apresenta os testes realizados, os dados recolhidos, os resultados obtidos e a sua interpretação e as limitações encontradas.
* **Capítulo 5** **– Conclusão** – Resume os principais pontos do trabalho, destacando os objetivos alcançados e algumas aprendizagens sobre o tema em questão.
* **Capítulo 6** **– Glossário e siglas** – Lista os termos técnicos utilizados ao longo do documento, segundo o estilo definido no regulamento.
* **Referências** – Lista as fontes bibliográficas referidas ao longo do documento, segundo o estilo definido no regulamento.

# 2 - Estado da Arte, Enquadramento Teórico e Prático

Este capítulo apresenta os fundamentos teóricos e técnicos essenciais para a concretização do trabalho. Inicia-se com a análise das principais estruturas de dados envolvidas, nomeadamente listas ligadas e grafos, e são também abordados os algoritmos de procura clássicos. Em seguida, são descritas soluções técnicas semelhantes no domínio, com uma análise comparativa.

## 2.1 - Fundamentos Teóricos e Tecnologias a Explorar

### 2.1.1 – Listas Ligadas

As **listas ligadas** são estruturas de dados dinâmicas compostas por elementos encadeados, nos quais cada nó aponta para o seguinte. Ao contrário dos ***arrays***, estas estruturas não necessitam de memória contígua, permitindo flexibilidade na inserção e remoção de elementos. No contexto deste trabalho, as listas ligadas foram utilizadas para armazenar as localizações das antenas do ficheiro, sendo cada nó correspondente a uma antena com frequência (A ou 0) e coordenadas associadas.

### 2.1.2 – Estrutura de Grafos

Um **grafo** é uma estrutura de dados dinâmica composta por um conjunto de vértices (ou nós) e arestas que estabelecem ligações entre pares de vértices. Os grafos podem ser direcionados ou não- direcionados. Neste trabalho, adotou-se um grafo não-dirigido, representando a interligação entre antenas com base na sua proximidade e frequência.

A representação selecionada foi a **lista de adjacência**, que se revelou mais eficiente em termos de memória para grafos esparsos, além de permitir uma implementação mais direta das operações de procura e manipulação das ligações.

### 2.1.3 – Algoritmos de Procura

Haviam sido pedidos dois algoritmos de procura, clássicos, que foram implementados para explorar os grafos: a **procura em profundidade** **(Depth-First Search, DFS)** e a **procura em largura (Breadth-First Search, BFS)**.

* O ***DFS*** percorre o grafo explorando o mais profundamente possível cada ramo antes de retroceder. Foi implementado de forma iterativa, sem recurso a recursividade, com o uso de uma pilha manual.
* O ***BFS*** percorre o grafo por níveis, utilizando uma fila para visitar todos os vizinhos de um vértice antes de passar para os seus descendentes.

Estas abordagens permitiram identificar componentes conexas e caminhos entre antenas, sendo fundamentais para o desenvolvimento de funcionalidades avançadas como a análise de conectividade e intersecções.

### 2.1.4 – Interferência de Sinal e Efeito Nefasto

Considera-se existir **efeito nefasto** quando duas antenas da mesma frequência estão situadas a uma distância inferior ou igual a dois. Este conceito foi explorado na Fase 1 do projeto, através da análise direta de antenas em lista ligada, e adaptado na Fase 2 ao contexto de grafos.

## 2.2 - Soluções Técnicas Existentes e Análise Comparativa

Consultaram-se exemplos de listas ligadas e grafos em repositórios do ***GitHub*** (GitHub, 2024) e materiais fornecidos pelo docente (Acosta, 2023). Vídeos explicativos complementares foram analisados no ***Youtube***(YouTube, 2024). O website ***StackOverflow*** foi utilizado bastantes vezes, tendo sido um complemento bastante importante para o desenvolvimento deste projeto.Este quadro comparativo, avalia diferentes abordagens com base em clareza, modularidade e eficiência.

|  |  |  |  |
| --- | --- | --- | --- |
| **Solução** | **Clareza** | **Modularidade** | **Eficiência** |
| Repositórios *GitHub* | Alta | Alta | Alta |
| Aula ACosta | Média | Média | Alta |
| StackOverflow | Alta | Alta | Alta |
| Tutoriais *Youtube* | Média | Média | Alta |

Figura 1 - Quadro das classificações das abordagens

A análise das soluções revela que, embora existam várias abordagens válidas, poucas estão orientadas especificamente para a gestão de antenas e vérices. A abordagem aqui proposta distingue-se pela flexibilidade da lista ligada e dos grafos.

Conclui-se que tanto as listas ligadas como os grafos são uma escolha eficaz para representar dados dinâmicos e dispersos. O trabalho desenvolvido baseia-se em boas práticas identificadas na pesquisa própria e nas aulas, incorporando as aprendizagens com vista a uma solução funcional e expansível.

# 3 - Trabalho Desenvolvido

O presente capítulo descreve detalhadamente o trabalho realizado no âmbito do projeto, com foco na análise, especificação e implementação de uma aplicação para representação e manipulação de antenas utilizando estruturas de dados do tipo grafo. O objetivo foi aplicar os conhecimentos adquiridos na unidade curricular de Estruturas de Dados Avançadas, nomeadamente listas ligadas, grafos e listas de adjacência.

## 3.1 - Fase 1

Uma lista ligada é uma lista onde cada elemento - chamado de nó - contém um valor e um ponteiro para o elemento seguinte. Assim, sabendo onde está o primeiro elemento da lista, podemos chegar a qualquer outro elemento.

No tema do trabalho, seriam várias antenas inseridas numa lista com as respetivas informações, em que cada antena apontava para a próxima.

A black screen with white text

AI-generated content may be incorrect.

Figura 2 - Lista Ligada

### 3.1.1 - Análise e Especificação

Definiram-se duas estruturas ***(structs)***, uma para se utilizar para as operações normais com Antenas, e depois uma para a localização dos efeitos nefastos, com os seguintes campos:

Antena:

* ***char frequencia***: indica a frequência da antena;
* ***int x, y***: coordenadas na matriz;
* ***Antena\* proxima***: ponteiro para a antena seguinte.

A black background with white text

AI-generated content may be incorrect.

Figura 3 - Estrutura Antena

Efeito Nefasto:

* ***int x, y***: coordenadas na matriz;
* ***struct LocalizacaoNefasta\* proxima***:

***Antena\* proxima***: ponteiro para a antena seguinte.

A screenshot of a computer code

AI-generated content may be incorrect.

Figura 4 - Estrutura Efeito Nefasto

## A screen shot of a computer AI-generated content may be incorrect.3.1.3 - Implementação

As funcionalidades desenvolvidas incluem:

* Carregamento de antenas a partir de um ficheiro;
* Inserção de nova antena na lista;
* Remoção de antena por coordenadas;
* Listagem organizada de todas as antenas.

Figura 5 - Output das funções *1*

## 3.2 - Fase 2

Uma lista de adjacência é uma estrutura de dados para representar grafos. Numa representação de uma lista de adjacência, podemos manter, para cada vértice do grafo, uma lista de todos os outros vértices com os quais ele tem uma aresta (a "lista de adjacência", deste vértice).

A black background with white numbers and arrows

AI-generated content may be incorrect.

Figura 6 - Lista de adjacências

## 3.2.2 – Análise e Especificação

Para a fase 2, definiu-se 4 estruturas ***(structs)***, uma para se utilizar nas operações com vértices e uma para operações com adjacências.

Vértice:

* ***struct Antena info:*** Informação da antena importada da ***struct*** Antena;
* ***struct Adj\* adj:*** Apontador para a próxima aresta na lista*;*
* ***struct Vertice próximo:*** Apontador para o próximo vértice na lista;
* ***int visitado:*** Verificação de se um vértice já foi visitado ou não;

A screen shot of a computer code

AI-generated content may be incorrect.

Figura 7 - Estrutura Vértice

Adjacência:

* ***struct Vertice\* origem:*** Apontador para o vértice de origem da adjacência;
* ***struct Vertice destino:*** Apontador para o vértice de destino da adjacência;
* ***struct Adj\* proxima:*** Apontador para a próxima aresta do vértice;

A black background with white text

AI-generated content may be incorrect.

Figura 8 - Estrutura Adj

### 3.2.3 - Análise e Especificação

As funcionalidades desenvolvidas incluem:

* ***A screen shot of a computer

  AI-generated content may be incorrect.A screen shot of a computer

  AI-generated content may be incorrect.***Carregamento de grafo e ligações entre vértices com a mesma frequência a partir de um ficheiro;

Figura 9 - Output das funções *2*

* Listagem de vértices com respetivas coordenadas;
* Listagem de lista de adjacências;
* Remoção de vértices e adjacências;
* Inserção de novo vértice, adjacência no grafo;
* ***DFS (Depth first search)*** e ***BFS (Breath first search)***;
* Interseções entre vértices com frequências diferentes;
* Gravação de ficheiro de texto (.c) em ficheiro binário;
* Libertação de memória;

## 3.3 - Tecnologias e Ferramentas Utilizadas

A implementação foi realizada na linguagem de programação ***C***, pela sua eficiência e controlo de baixo nível, fundamentais para a gestão manual de memória exigida por estruturas de dados dinâmicas como as listas ligadas. Utilizar ***C*** permitiu um maior domínio sobre operações como a alocação e libertação de memória, através das funções ***malloc()*** e ***free()****,* assegurando o correto funcionamento do programa.

A organização modular do código foi estruturada em ficheiros ***.c*** e ***.h***, garantindo uma separação clara entre a implementação e os protótipos das funções, facilitando a manutenção e legibilidade. Para automatizar o processo de compilação, foi utilizado um ***Makefile***, o que permitiu simplificar a construção do projeto e assegurar consistência entre ***builds***.

A documentação do projeto foi gerada com a ferramenta ***Doxygen***, que produz documentação a partir de comentários no código, assegurando que qualquer utilizador ou programador futuro compreenda a estrutura e funcionalidade das diferentes partes do sistema. Esta abordagem segue as boas práticas de desenvolvimento de *software*, reforçando a clareza e profissionalismo do trabalho.

## 3.3 - Organização do Código

O projeto foi dividido em múltiplos ficheiros e duas pastas, sendo estas duas Fase 1 e Fase 2 e estando todos dentro da pasta ***src (source)*** da pasta do projeto:

**Fase 1:**

* ***main.c***: função principal e interação com o utilizador;
* ***funcoes.c* e *funcoes.h***: corpo e implementação das funções principais;
* ***struct.h***: definição das *structs*;
* ***Makefile***: automatização da compilação do programa com o uso de comandos.

**Fase 2:**

* ***main.c***: função principal e interação com o utilizador;
* ***grafos.c* e *grafos.h***: corpo e implementação das funções principais;
* ***structs.h***: definição das ***structs***;
* ***Makefile***: automatização da compilação do programa com o uso de comandos.

A screen shot of a computer

AI-generated content may be incorrect.

Figura 10 - Organização dos ficheiros

# 4 - Análise e Discussão de Resultados

O trabalho desenvolvido cumpriu com sucesso os requisitos funcionais definidos para o mesmo, nomeadamente a criação e manipulação de uma lista ligada para representar antenas em contexto urbano, e da gestão e manipulação de uma lista de adjacências para representar antenas no mesmo contexto da primeira fase. As funcionalidades implementadas — carregamento a partir de ficheiro, inserção, remoção e listagem — demonstraram um comportamento correto em todos os testes realizados, validando a fiabilidade da estrutura de dados escolhida.

A clareza e a modularidade do código, associadas ao uso de ferramentas de documentação como o ***Doxygen***, contribuíram significativamente para a organização e manutenibilidade do projeto. A automatização do processo de compilação através de ***Makefile*** permitiu garantir consistência entre diferentes execuções, demonstrando a importância de práticas profissionais na construção de software académico. No entanto, a funcionalidade de deteção de efeitos nefastos foi implementada apenas no período designado para a segunda fase.

Na segunda fase, conseguiu-se cumprir com quase todos os objetivos, ficando apenas por fazer a função para encontrar todos os caminhos entre dois determinados vértices. Muitas das funções feitas na segunda fase, foram inspiradas e até adaptadas da primeira, o que refletiu uma maior compreensão sob as mesmas, fazendo-se encontrar novas e mais simples formas de chegar ao mesmo objetivo, e também dando uma boa ajuda em termos de organização de código.

A screenshot of a graph

AI-generated content may be incorrect.

Figura 11 - Comparação de desempenho

# 5 - Conclusão

A primeira fase do projeto representou um marco significativo na consolidação de competências em estruturas de dados dinâmicas, com ênfase na implementação de listas ligadas simples em linguagem ***C***. Através da criação de um sistema capaz de gerir informações sobre antenas, incluindo operações de carregamento, inserção, remoção e listagem, alcançaram-se os principais objetivos estabelecidos no início do trabalho.

Já na segunda fase, foi onde o que se teria aprendido previamente fosse aprofundado, e feito de uma forma completamente diferente, desta vez, utilizou-se grafos.

A estrutura modular do código, aliada à utilização de ferramentas como ***Doxygen*** para documentação, garante não apenas a funcionalidade imediata, mas também a possibilidade de expansão para análises mais complexas nas fases subsequentes.

O trabalho evoluiu, entre a primeira e a segunda fase, de uma simples gestão de antenas com listas ligadas para uma análise em grafo capaz de representar ligações, percorrer a rede e sinalizar interferências. Na fase inicial demonstrou-se que a leitura de dados, a inserção, a remoção e a listagem podiam ser realizadas de forma consistente em ***C***, tirando partido de código modular e da geração automática de documentação com ***Doxygen***. Sobre essa base adicionou-se, na fase 2, a conversão de cada antena num vértice, o estabelecimento automático de arestas entre dispositivos de frequência idêntica e a implementação das buscas ***DFS*** e ***BFS***, ferramentas que permitem explorar rapidamente a topologia formada. A deteção dos chamados “efeitos nefastos” foi concluída, mas outras sobraram.

Em síntese, o projeto cumpriu com a maioria dos objetivos propostos: introduziu estruturas de dados dinâmicas, aplicou conceitos de grafos e entregou um protótipo funcional capaz de evoluir. A experiência consolidou práticas de programação modular, documentação e planeamento de tarefas, proporcionando uma base sólida para aprofundamentos futuros em algoritmos de gestão de estruturas de dados dinâmicas. O projeto contribuiu para o aprofundamento do entendimento sobre a aplicação prática de listas ligadas e de grafos em ***C***, evidenciando as suas vantagens em cenários dinâmicos e a importância de uma implementação cuidadosa para garantir eficiência e escalabilidade.

Em síntese, ambas as fases do projeto foram bem-sucedidas, cumprindo os objetivos iniciais e abrindo caminho para desenvolvimentos futuros.

# 6 - Glossário e Siglas

A seguir apresenta-se um glossário com os principais termos e siglas utilizados ao longo do relatório, por ordem alfabética.

* ***Antena***: Elemento da rede responsável pela emissão e receção de sinais de rádio.
* ***Doxygen***: Ferramenta de geração de documentação a partir do código fonte.
* ***char***: Tipo de dados em C que armazena um único caracter.
* ***Arrays***: Conjunto de dados em C.
* ***malloc***: Função em C usada para alocar memória dinamicamente.
* ***struct***: Estrutura de dados composta em C que permite agrupar diferentes tipos.
* ***builds***: Processos de compilação de código-fonte
* ***Depth-First Search (DFS)***: Algoritmo de procura em profundidade.
* ***Breadth-First Search (BFS)***: Algoritmo de procura em largura.
* ***src(source)***: Pasta onde se guarda os ficheiros de um projeto.

# 7 - Referências

**Acosta, L. (2023).** Lista Ligada em C. Notas de aula do IPCA.

***StackOverflow. (2024).*** Website para ver excertos de código de outras pessoas. Disponível em:  
<https://stackoverflow.com/>

***GitHub*. (2024).** Repositório com relatório de inspiração. Disponível em:

[**https://github.com/LFtech6/cryptrail-app/blob/master/Relato%CC%81rio-PAP.docx**](https://github.com/LFtech6/cryptrail-app/blob/master/Relato%CC%81rio-PAP.docx)

***GitHub*. (2024).** Repositório das aulas. Disponível em:

[***https://github.com/luferIPCA/EDA-LESI-2024-2025***](https://github.com/luferIPCA/EDA-LESI-2024-2025)

***GitHub*. (2024).** Repositório próprio de projeto anterior. Disponível em:

[***https://github.com/LFtech6/PI-tp***](https://github.com/LFtech6/PI-tp)

***ChatGPT. (2024).***

[***https://chatgpt.com***](https://chatgpt.com)

***Grok AI. (2024).***

[***https://grok.com***](https://grok.com)

***YouTube*. (2024).** Vídeo explicativo sobre listas ligadas. Disponível em:

[***https://www.youtube.com/watch?v=N6dOwBde7-M***](https://www.youtube.com/watch?v=N6dOwBde7-M)

**Knuth. D. E. (1997).** The Art of Computer Programming:

Volume 1 - Fundamental Algorithms. Addison-Wesley.