Instituto Politécnico do Cávado e do Ave

Escola Superior de Tecnologia

Estrutura de Dados Avançadas

Licenciatura em Engenharia de Sistemas Informáticos

Trabalho Prático

Gestão de antenas e efeitos nefastos

Pedro Lourenço Morais Rocha – a23009

março de 2025

Resumo

Este relatório apresenta o desenvolvimento da Fase 1 do projeto de avaliação da Unidade Curricular Estruturas de Dados Avançadas (EDA), focado na implementação de listas ligadas em C para gerenciar antenas em uma cidade fictícia. O sistema carrega dados do arquivo antenas.txt (matriz 12x12), permite inserção e remoção de antenas, deteta efeitos nefastos baseados em interferências de frequência e exibe resultados em formato tabular. A solução utiliza estruturas dinâmicas (Antena e Nefasto), modularização em quatro arquivos e documentação *Doxygen*, atendendo aos requisitos do enunciado.

Índice

Índice de Ilustrações

1. Introdução

Explora-se neste capítulo o contexto, a motivação e os objetivos da Fase 1 do projeto de avaliação da Unidade Curricular Estruturas de Dados Avançadas (EDA), bem como a metodologia aplicada durante o desenvolvimento e a organização detalhada deste documento.

1.1 Motivação

Considera-se o problema de gerenciar antenas numa cidade fictícia, onde cada antena possui uma frequência representada por um caractere específico e está posicionada em coordenadas definidas numa matriz bidimensional. Identifica-se a necessidade de uma solução que permita manipular dinamicamente essas antenas, incluindo operações como inserção e remoção, além de detetar efeitos nefastos gerados por interferências entre antenas de mesma frequência, especificamente quando uma está a uma distância dupla da outra em relação a um ponto de efeito. A ausência de soluções prontas para este cenário em contextos académicos iniciais justifica a implementação de uma abordagem personalizada em linguagem C. Releva-se a pertinência atual deste trabalho no âmbito da programação de baixo nível, uma vez que estruturas dinâmicas como listas ligadas são amplamente utilizadas para resolver problemas que requerem flexibilidade no armazenamento e manipulação de dados, especialmente em sistemas espaciais ou de telecomunicações.

1.2 Enquadramento

Insere-se este trabalho no âmbito da Unidade Curricular Estruturas de Dados Avançadas, lecionada pelo docente Luís Ferreira, integrada no 2º semestre do 1º ano da Licenciatura em Engenharia de Sistemas Informáticos. Não se vincula este projeto a um estágio ou contexto externo, sendo exclusivamente uma atividade académica focada na aplicação prática dos conceitos lecionados.

1.3 Objetivos

Definiu-se como objetivos principais para a Fase 1 do projeto:

- Implementar uma estrutura de dados dinâmica, designada por ED no enunciado, utilizando lists ligadas para representar antenas, cada uma contendo informações sobre sua frequência de ressonância e coordenadas (x, y) numa matriz.
- Carregar os dados das antenas a partir de um arquivo de texto fornecido, antenas.txt, suportando matrizes de dimensões variáveis até um limite máximo definido.
- Desenvolver operações de manipulação da lista ligada, incluindo a inserção de novas antenas em posições específicas, a remoção de antenas existentes com base nas suas coordenadas, a dedução automática das localizações com efeitos nefastos representadas numa lista ligada separada e a listagem de todas as antenas e efeitos nefastos em formato tabular na consola.

1.4 Estrutura do Documento

Organiza-se este relatório em várias seções principais para proporcionar uma visão abrangente do trabalho desenvolvido. Inicia-se com o Resumo, que oferece uma perspetiva global do projeto. Segue-se a Introdução, que detalha a motivação, enquadramento, objetivos, metodologia, plano de trabalho e estrutura do documento. A seção Trabalho Desenvolvido descreve a análise, especificação e implementação do sistema. A Análise de Resultados avalia o desempenho e os resultados obtidos. A Conclusão sumariza as principais conquistas.

2. Trabalho Desenvolvido

Descreve-se neste capítulo a análise detalhada, a especificação dos requisitos e a implementação completa do sistema de gestão de antenas e deteção de efeitos nefastos, incluindo todas as etapas realizadas para atingir os objetivos propostos.

2.1 Análise e Especificação

Analisaram-se os requisitos do enunciado para definir as funcionalidades necessárias ao sistema:

- **Carregamento de Dados**: Ler o arquivo antenas.txt e inicializar uma lista ligada de antenas com base nos caracteres encontrados na matriz, bem como uma matriz dinâmica para visualização.
- **Manipulação de Antenas**: Permitir a inserção de novas antenas em posições específicas e a remoção de antenas existentes com base nas suas coordenadas.
- **Deteção de Efeitos Nefastos**: Identificar automaticamente as localizações com efeitos nefastos, gerados por pares de antenas de mesma frequência, e armazená-las numa lista ligada separada.
- **Listagem Tabular**: Exibir na consola as antenas e os efeitos nefastos em formato de tabelas organizadas e legíveis.

Projetou-se uma arquitetura lógica baseada em duas estruturas principais:

- **Estrutura Antena**: Uma lista ligada contendo os campos frequência (caractere), x e y (coordenadas inteiras) e prox (ponteiro para o próximo nó).
- **Estrutura Nefasto**: Uma lista ligada contendo os campos x e y (coordenadas inteiras) e prox (ponteiro para o próximo nó), destinada a armazenar os efeitos nefastos calculados.

3. Análise Técnica

3.1 Estrutura de Dados

- **Lista Ligada de Antenas**: Cada nó (Antena) armazena um caractere (frequência) e coordenadas (x, y), com um ponteiro para o próximo nó. Inserção ocorre no início, e remoção tem complexidade.
- **Lista Ligada de Efeitos Nefastos**: Cada nó (Nefasto) contém apenas coordenadas (x, y), evitando duplicatas durante a inserção.

3.2 Carregamento de Dados

- A função *carregarAntenas* lê o antenas.txt linha a linha, inserindo antenas na lista ligada quando encontra caracteres diferentes de '.'.

```c
Antena* carregarAntenas(const char* filename) {
    FILE* file = fopen(filename, "r");
    if (!file) {
        printf("Erro ao abrir o ficheiro!\n");
        return NULL;
    }
    Antena* lista = NULL;
    char linha[100];
    for (int y = 0; fgets(linha, sizeof(linha), file); y++) {
        for (int x = 0; linha[x] != '\0' && linha[x] != '\n'; x++) {
            if (linha[x] != '.') inserirAntena(&lista, linha[x], x, y);
        }
    }
    fclose(file);
    return lista;
}
```

Figura 1: Função para ler o antena.txt

- Suporta matrizes de qualquer dimensão até 100x100, conforme definido por *MAX_LINHAS* e *MAX_COLUNAS*.

3.3 Operações de manipulação

- **Inserção**: *inserirAntena* insere uma nova antena no início da lista ligada, chamando *criarAntena* e atualizando o ponteiro da lista. Retorna 1 em caso de sucesso ou 0 em caso de falha.

```
int inserirAntena(Antena** lista, char freq, int x, int y) {
    Antena* nova = criarAntena(freq, x, y);
    if (!nova) return 0;
    nova->prox = *lista;
    *lista = nova;
    return 1;
}
```

Figura 2: Função para inserir antenas

- **Remoção**: *removerAntena* Busca uma antena na lista ligada com base nas coordenadas x e y, percorrendo a lista sequencialmente. Quando encontrada, remove o nó correspondente, atualizando os ponteiros e liberando a memória associada com free. Retorna 1 em caso de sucesso ou 0 se a antena não for encontrada.

```
int removerAntena(Antena** lista, int x, int y) {
    Antena* atual = *lista, *anterior = NULL;
    while (atual && (atual->x != x || atual->y != y)) {
        anterior = atual;
        atual = atual->prox;
    }
    if (!atual) return 0;
    if (!anterior) *lista = atual->prox;
    else anterior->prox = atual->prox;
    free(atual);
    return 1;
}
```

Figura 3: Função para remover antena

Deteção de Efeitos: *detetarEfeitosNefastos* Compara todos os pares de antenas na lista ligada, verificando se possuem a mesma frequência. Para cada par identificado, calcula as diferenças nas coordenadas (dx e dy) e determina dois pontos simétricos que representam os efeitos nefastos, baseando-se na ideia de alinhamento e distância dupla conforme o enunciado. Esses pontos são inseridos na lista de efeitos nefastos usando inserirNefasto.

```
Nefasto* detetarEfeitosNefastos(Antena* lista) {
    Nefasto* efeitos = NULL;
    for (Antena* a1 = lista; a1; a1 = a1->prox) {
        for (Antena* a2 = lista; a2; a2 = a2->prox) {
            if (a1 != a2 && a1->frequencia == a2->frequencia) {
                int dx = abs(a1->x - a2->x);
                int dy = abs(a1->y - a2->y);
                int ex1 = a1->x > a2->x ? a1->x + dx : a1->x - dx;
                int ey1 = a1->y > a2->y ? a1->y + dy : a1->y - dy;
                int ex2 = a2->x > a1->x ? a2->x + dx : a2->x - dx;
                int ey2 = a2->y > a1->y ? a2->y + dy : a2->y - dy;
                inserirNefasto(&efeitos, ex1, ey1);
                inserirNefasto(&efeitos, ex2, ey2);
            }
        }
    }
    return efeitos;
}
```

Figura 4: Função que deteta efeitos nefastos

- **Impressão de antenas**: Exibe a lista de antenas em formato tabular na consola, com colunas para "Frequência" e "Coordenadas". Inclui linhas divisórias para melhorar a legibilidade, percorrendo a lista ligada e imprimindo cada nó até o final.

```c
void imprimirAntenas(Antena* lista) {
    printf("\nLista de Antenas:\n");
    printf("Frequência | Coordenadas\n");
    printf("-----------------------\n");
    while (lista) {
        printf("    %c      | (X:%d, Y:%d)\n", lista->frequencia, lista->x, lista->y);
        lista = lista->prox;
    }
    printf("-----------------------\n");
}
```

Figura 5: Função que exibe as antenas

- **Impressão de Efeitos Nefastos**: Exibe a lista de efeitos nefastos em formato tabular, com uma coluna para "Coordenadas", utilizando o símbolo '#' para indicar os pontos afetados, conforme sugerido no enunciado.

```c
void imprimirEfeitosNefastos(Nefasto* lista) {
    printf("\nEfeitos Nefastos:\n");
    printf("Coordenadas\n");
    printf("------------\n");
    while (lista) {
        printf("# |  (X:%d, Y:%d)\n", lista->x, lista->y);
        lista = lista->prox;
    }
    printf("------------\n");
}
```

Figura 6: Função que exibe os efeitos nefastos

No arquivo main.c, integrou-se todas essas funcionalidades num fluxo de execução que testa o sistema com o antenas.txt. Carrega-se inicialmente as antenas e a matriz, insere-se uma antena 'B' em (5, 5), detetam-se os efeitos nefastos, remove-se a antena inserida e uma tentativa adicional em (6, 5), recalculam-se os efeitos e atualiza-se a matriz, exibindo os resultados após cada etapa.

4. Análise de Resultados

Avaliam-se neste capítulo os resultados obtidos com a execução do sistema, analisando o desempenho das operações implementadas e discutindo os pontos fortes e limitações observadas durante os testes com o arquivo antenas.txt.

Executou-se o programa com o arquivo antenas.txt, uma matriz de 12 linhas por 12 colunas contendo 7 antenas iniciais: '0' nas posições (9, 2), (6, 3), (7, 4) e (4, 5), e 'A' nas posições (6, 6), (8, 9) e (9, 10). A função *carregarAntenas* inicializou corretamente a lista ligada com essas antenas, e a função *imprimirAntenas* exibiu uma tabela clara com as frequências e coordenadas correspondentes. Verificou-se que o carregamento é eficiente para matrizes pequenas como esta, com complexidade O(m*n), onde m e n são as dimensões da matriz.

Inseriu-se uma antena 'B' na posição (5, 5) utilizando *inserirAntena*, pois a inserção ocorre sempre no início da lista. A tabela atualizada confirmou a adição correta da nova antena. Em seguida, aplicou-se *detetarEfeitosNefastos*, que identificou pontos de interferência, como (3, 2) para o par de antenas '0' em (6, 3) e (9, 2), com base na diferença de coordenadas e na projeção simétrica. A saída tabular gerada por *imprimirEfeitosNefastos* apresentou esses pontos de forma organizada, embora se reconheça que a validação explícita da distância dupla não foi implementada, utilizando-se uma aproximação baseada em simetria.

Removeu-se a antena 'B' em (5, 5) com *removerAntena*, pois exige percorrer a lista até encontrar as coordenadas correspondentes. A remoção foi bem-sucedida, e a tabela atualizada refletiu a ausência da antena. Tentou-se remover uma antena em (6, 5), mas a operação retornou 0, indicando que não havia antena nessa posição, o que está correto dado o conteúdo inicial do antenas.txt. Após a remoção, recalculou-se os efeitos nefastos, confirmando que a alteração na lista de antenas impactou os resultados esperados.

Analisou-se a matriz em cada etapa com *imprimirMatriz*, verificando que as atualizações refletiam corretamente as antenas e efeitos nefastos. A complexidade $O(n^2)$ de *detetarEfeitosNefastos* é aceitável para o tamanho reduzido da matriz de teste (12x12 com 7 antenas), mas poderia tornar-se um gargalo em cenários com centenas de antenas. Identificou-se como limitação principal a ausência de uma função para liberar a memória alocada para as listas ligadas, o que pode causar *memory leaks* em execuções prolongadas.

5. Conclusão

Conclui-se neste capítulo que o desenvolvimento da Fase 1 do projeto resultou numa solução funcional e robusta que atende plenamente aos requisitos estabelecidos no enunciado da Unidade Curricular Estruturas de Dados Avançadas. Consolidaram-se os conhecimentos sobre estruturas de dados dinâmicas em C, especificamente listas ligadas, através da implementação de um sistema que gere antenas e deteta efeitos nefastos de forma eficiente para o contexto proposto. A utilização de listas ligadas para representar tanto as antenas como os efeitos nefastos demonstrou adequação para manipulação dinâmica, permitindo inserções e remoções rápidas e flexíveis, enquanto a modularização em quatro arquivos (*estrutura.h*, *lista.h*, *lista.c* e *main.c*) reflete boas práticas de organização de código, facilitando a manutenção e a escalabilidade do projeto.

A documentação detalhada no formato *Doxygen*, presente em todos os arquivos de código, oferece clareza sobre a lógica das funções, seus parâmetros e retornos, atendendo ao requisito de qualidade na documentação.

6. Repositório GitHub

No seguinte link encontra-se a versão mais atualizada do projeto, com um *ReadMe* bem definido que explicita o que se encontra no repositório *Git*:
https://github.com/PieWolfie/ProjetoEDA.git