Gerenciamento de Antenas: Estruturas de Dados e Algoritmos

João Loureiro

Março de 2025

Abstract

Este trabalho apresenta um sistema para gerenciamento de antenas em uma cidade, utilizando listas ligadas simples. O sistema permite carregar, inserir, remover e listar antenas, além de detectar interferências e exibir um mapa representativo. Foram aplicadas técnicas de manipulação de listas dinâmicas para otimizar a eficiência das operações. Link do repositorio Git: https://github.com/JoaoLoureiro06/EDA-Project.git

Contents

1 Introdução

O gerenciamento de antenas é um desafio na organização de redes de comunicação. Este projeto propõe um sistema baseado em estruturas de dados dinâmicas para armazenar e manipular informações sobre antenas, otimizando a detecção de interferências e o controle de posicionamento.

2 Metodologia

Utilizamos uma lista ligada simples para armazenar os dados das antenas. As operações implementadas incluem:

- Inserção de novas antenas.
- Remoção de antenas existentes.
- Listagem de todas as antenas.
- Detecção de interferências com base na proximidade.
- Exibição de um mapa representando as antenas e áreas problemáticas.

3 Estrutura de Dados Utilizada

A estrutura básica da antena foi definida conforme abaixo:

```
typedef struct Antena {
    char freq;
    int x, y;
    struct Antena* next;
} Antena;
```

Listing 1: Estrutura da Antena

Cada antena possui uma frequência representada por um caractere, além das coordenadas (x, y). A lista ligada armazena os elementos de maneira dinâmica.

4 Implementação do Sistema

4.1 Criação e Inserção de Antenas

```
Antena* criarAntena(char freq, int x, int y) {
    Antena* nova = (Antena*)malloc(sizeof(Antena));
    nova->freq = freq;
    nova->x = x;
    nova->y = y;
    nova->next = NULL;
    return nova;
```

```
}
```

Listing 2: Função para criar uma antena

A inserção ocorre no início da lista:

```
void inserirAntena(Antena** head, char freq, int x, int y) {
    Antena* nova = criarAntena(freq, x, y);
    nova->next = *head;
    *head = nova;
}
```

Listing 3: Função para inserir antenas

4.2 Remoção de Antenas

A remoção de antenas é realizada buscando pelo par de coordenadas:

```
void removerAntena(Antena** head, int x, int y) {
    Antena* temp = *head, *prev = NULL;
    while (temp != NULL && (temp->x != x || temp->y != y)) {
        prev = temp;
        temp = temp->next;
    }
    if (temp == NULL) return;
    if (prev == NULL) *head = temp->next;
    else prev->next = temp->next;
    free(temp);
}
```

Listing 4: Função para remover uma antena

4.3 Listagem de Antenas

A listagem das antenas percorre a estrutura e exibe as informações:

```
void listarAntenas(Antena* head) {
    while (head != NULL) {
        printf("Frequ ncia: %c | Posi   o: (%d, %d)\n", head->freq, head
            ->x, head->y);
        head = head->next;
    }
}
```

Listing 5: Função para listar antenas

4.4 Detecção de Interferências

A detecção ocorre comparando pares de antenas de mesma frequência:

```
void deduzirEfeitoNefasto(Antena* head) {
    for (Antena* a1 = head; a1 != NULL; a1 = a1->next) {
        for (Antena* a2 = a1->next; a2 != NULL; a2 = a2->next) {
```

```
            if (a1->freq == a2->freq) {
                int mx = (a1->x + a2->x) / 2;
                int my = (a1->y + a2->y) / 2;
                printf("Efeito nefasto em (%d, %d)\n", mx, my);
            }
        }
    }
}
```

Listing 6: Função para detectar interferências

5 Resultados e Discussão

Os testes realizados mostraram que a lista ligada proporciona uma manipulação eficiente das antenas. A complexidade das operações principais é:

- Inserção: $O(1)$
- Remoção: $O(n)$
- Listagem: $O(n)$
- Detecção de interferências: $O(n^2)$

Os resultados indicam que, para grandes volumes de dados, a detecção de interferências pode ser otimizada com estruturas adicionais, como árvores ou tabelas hash.

6 Conclusão

O sistema desenvolvido demonstrou eficiência na manipulação e análise de antenas. A estrutura de lista ligada possibilitou um controle dinâmico das antenas, mas melhorias podem ser implementadas para otimizar a detecção de interferências.

7 Referências

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. *Introduction to Algorithms*, MIT Press, 2009.