Integrantes do grupo:

- Tomás Lenzi (2220711)
- Gabriel Emile (2220498)

1. Introdução

Este relatório descreve a implementação de um Tipo Abstrato de Dados (TAD) para uma Árvore B de ordem 5, como parte do trabalho de Estruturas de Dados Avançadas (EDA) da PUC-Rio. O programa implementa funções para inserir chaves em uma Árvore B, imprimir a árvore em ordem simétrica após cada inserção e uma função especial para imprimir chaves em um determinado intervalo. O objetivo é

criar uma árvore B que possa eficientemente gerenciar inserções e buscas, características essenciais para estruturas de dados avançadas.

2. Estrutura do Programa

O programa é dividido em três arquivos principais: funcoes.h, funcoes.c, e main.c.

funcoes.h: Este arquivo de cabeçalho define a estrutura de um nó da árvore B (t_no) e os protótipos das funções utilizadas no programa. Cada nó da árvore contém um número de descritores (ndesc), um array de chaves e um array de ponteiros para outros nós.

funcoes.c: Este arquivo contém a implementação das funções definidas em funcoes.h. Inclui a função cria_no, que aloca memória para um novo nó, e insere, que insere uma chave na árvore B. Também implementa funções adicionais para gerenciar a inserção de chaves e a divisão de nós, garantindo as propriedades da árvore B.

main.c: O arquivo principal do programa, onde as funções definidas são utilizadas para criar a árvore B e inserir as chaves. Após cada inserção, a árvore é impressa em ordem simétrica. Este arquivo também testa a função imprime_intervalo com diferentes intervalos de chaves.

3. Solução

A solução implementada para a Árvore B envolve a inserção e manutenção de chaves de acordo com as regras específicas para árvores B. Abaixo estão trechos do código que ilustram a implementação das funções principais:

Trecho de funcoes.c

```
// Definição da função cria_no
t_no* cria_no() {
    t_no* novo = (t_no*)malloc(sizeof(t_no));
    if (novo) {
        novo->ndesc = 0;
        for (int i = 0; i < MAX + 1; i++) {
            novo->ramo[i] = NULL;
        }
    }
    return novo;
}
```

Este trecho mostra a função cria_no, que aloca memória para um novo nó da árvore B e inicializa seus membros.

Trecho de main.c

```
// Inserção e impressão das chaves
int main() {
    int chaves[] = { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150,
160, 170, 180, 190, 200, 210 };
    int n = sizeof(chaves) / sizeof(chaves[0]);

    t_no* raiz = NULL;

    for (int i = 0; i < n; i++) {
        insere(&raiz, chaves[i]);
        printf("Arvore B apos insercao de %d em ordem simetrica:\\n", chaves[i]);
        imprime_simetrico(raiz);
    }
}
```

Este trecho ilustra como as chaves são inseridas na árvore B e como a árvore é impressa em ordem simétrica após cada inserção.

4. Observações e Conclusões

Durante a implementação, foram encontradas algumas dificuldades, principalmente relacionadas à manutenção do balanceamento da árvore durante as inserções. No entanto, a estrutura final funcionou conforme o esperado, com inserções e impressões em ordem simétrica sendo realizadas corretamente.

5. Teste da Função imprime_intervalo

A função imprime_intervalo foi testada com diferentes intervalos de valores. Os resultados demonstraram que a função é capaz de percorrer a árvore B e imprimir as chaves dentro do intervalo especificado de forma eficiente.