Relatório: Implementação de Árvore AVL

Introdução

Este documento descreve uma implementação de uma árvore binária de busca (BST) com balanceamento AVL em C. A árvore AVL é uma forma de árvore binária de busca que automaticamente se mantém balanceada, garantindo complexidade de busca (O(\log n)) em todos os casos. O código possui as seguintes funcionalidades principais: criação, inserção, remoção e balanceamento de nós.

Estrutura dos Arquivos

O código está dividido em três arquivos principais:

1. **lib_bsttree.h**: Define as estruturas de dados e protótipos das funções para manipulação da BST.
2. **lib_avltree.h**: Define funções específicas para manutenção do balanceamento AVL, como rotações e cálculo do fator de balanceamento.
3. **Implementação C**: Os arquivos .c contêm a implementação das funções definidas nos cabeçalhos.

Estrutura do Código

Estrutura do Nó

A estrutura do nó é definida pelos seguintes campos:

```
struct node {
    int key; // Valor do nó
    struct node *left; // Ponteiro para nó a esquerda
    struct node *right; // Ponteiro para nó a direita
    struct node *father; // Ponteiro para o nó pai
    int height; // Altura do nó
};
```

Funções de BST

- `create_node(int value_key)` : Cria um novo nó inicializado com a chave fornecida.
- `insert_node(struct node *root, int value_key)` : Insere um novo nó na BST, preservando a propriedade de árvore de busca.

- `delete_node(struct node *root, int value_key)` : Remove um nó da árvore, mantendo a estrutura correta da BST.
- `inoder(struct node *root, int height_node)` : Realiza a travessia in-order da árvore, imprimindo os nós em ordem crescente.

Funções de AVL

- `node_balance(struct node *node_tree)` : Retorna a altura do nó ou -1 se for nulo, usado para calcular o fator de balanceamento.
- `balancing_test(struct node *node_tree)` : Calcula a diferença entre a altura dos filhos esquerdo e direito para determinar o balanceamento.
- **Rotações**:
  - `rotate_left(struct node *node_tree)` e `rotate_right(struct node *node_tree)` : Realizam rotações à esquerda e à direita para balancear a árvore.
- `tree_balancing(struct node *node_tree)` : Realiza o balanceamento do nó conforme o fator de balanceamento, aplicando rotações simples ou duplas.

Lógica de Balanceamento AVL

O balanceamento AVL é realizado após cada inserção ou remoção de nós. O fator de balanceamento (‘balancing_test’) é a diferença entre as alturas dos filhos esquerdo e direito de um nó. Se o fator for maior que 1 ou menor que -1, é necessária uma rotação para reequilibrar a árvore:

- **Rotação Simples**: É aplicada quando o desequilíbrio é linear (esquerda-esquerda ou direita-direita).
- **Rotação Dupla**: Ocorre quando o desequilíbrio é “cruzado” (esquerda-direita ou direita-esquerda).

A função `tree_balancing` é responsável por ajustar os ponteiros e manter a árvore balanceada após cada operação.

Conclusão

A implementação da árvore binária AVL, apresentada, oferece uma estrutura eficiente para armazenar e buscar elementos. O uso de balanceamento automático garante que a altura da árvore se mantenha aproximamente `log N`, mantendo as operações de busca, inserção e remoção eficientes. O código exemplifica as principais operações de uma árvore AVL, incluindo inserções, remoções e rotações para balanceamento.