Relatório Técnico: Implementação e Análise de Árvores Balanceadas

Disciplina: Estruturas de Dados Básicas II
Autor: João Miguel Sousa Oliveira e Pedro Melo Arruda **GitHub:** `https://github.com/seu-usuario/seu-repositorio.git`

1. Introdução

Este relatório documenta o projeto de implementação e análise de duas estruturas de dados fundamentais: a Árvore AVL e a Árvore Rubro-Negra. O objetivo foi aplicar os conceitos teóricos de balanceamento em uma implementação funcional, interativa e devidamente testada, permitindo a comparação prática entre as duas abordagens.

---

2. Estruturas de Dados e Estratégias de Balanceamento

2.1. Árvore AVL

A Árvore AVL é uma árvore de busca binária que mantém seu balanceamento de forma rigorosa. A estratégia central é garantir que, para qualquer nó, a diferença de altura entre suas sub-árvores esquerda e direita (o "Fator de Balanceamento") nunca seja maior que 1.

- **Manutenção:** Após cada inserção ou remoção, o fator de balanceamento dos nós no caminho de volta à raiz é verificado.
- **Operações:** Caso um desequilíbrio seja detectado, o balanço é restaurado através de **rotações simples (LL, RR)** ou **duplas (LR, RL)**, que reestruturam a árvore localmente para atender à propriedade de balanceamento.

2.2. Árvore Rubro-Negra

A Árvore Rubro-Negra adota uma abordagem de balanceamento menos estrita, utilizando um sistema de cores (Vermelho e Preto) para garantir que o caminho mais longo da raiz a uma folha não seja mais que o dobro do caminho mais curto. Isso é assegurado por um conjunto de 5 propriedades fundamentais.

- **Manutenção:** A violação de uma dessas propriedades após uma inserção ou remoção dispara um processo de correção.
- **Operações:** O equilíbrio é restaurado primariamente por **recoloração** de nós, uma operação computacionalmente barata. Quando a recoloração não é suficiente, **rotações** são realizadas para reestruturar a árvore e restabelecer as propriedades.

---

3. Funções e Módulos Implementados

O projeto foi estruturado de forma modular para separar as responsabilidades:

- `main.py`**:** Contém a interface de linha de comando (CLI), responsável por toda a interação com o usuário. Permite escolher a árvore, selecionar operações e visualizar os resultados.

- **`src/arvore_avl.py`:** Implementação completa da Árvore AVL.
- **`src/arvore_rn.py`:** Implementação completa da Árvore Rubro-Negra.

Ambas as implementações fornecem uma API consistente com as seguintes funcionalidades:

- `inserir(valor)` e `remover(valor)`: Para manipulação dos dados.
- `buscar(valor)`: Para consulta de elementos.
- `imprimir()`: Para visualização da estrutura da árvore.
- `em_ordem()` e `obter_tamanho()`: Para análise e obtenção de informações.

---

4. Desafios na Implementação

Um dos maiores desafios foi a **criação de testes unitários confiáveis**. Testar uma estrutura de dados complexa e "viva" como uma árvore balanceada não é uma tarefa trivial. Não basta apenas verificar se um valor foi inserido; é preciso garantir que, após a inserção, a árvore inteira ainda obedece às regras estritas de sua estrutura (seja o fator de balanceamento da AVL ou as propriedades de cor da Rubro-Negra).

Isso exigiu um planejamento cuidadoso para cobrir uma vasta gama de cenários. Foi preciso pensar em casos de uso simples, como a inserção em uma árvore vazia, mas também em casos complexos, como a remoção de um nó raiz que exige múltiplas rotações para rebalancear a árvore. Criar testes para cada um desses "casos de borda" (edge cases) foi um exercício desafiador, que demandou mais tempo e lógica do que a própria implementação de algumas das funcionalidades principais.

---

5. Testes e Validação

Para garantir a corretude e robustez das implementações, foi utilizada a ferramenta `pytest`.

- **Estratégia:** Uma suíte de 10 testes automatizados foi desenvolvida para validar as operações de Inserção, Remoção e Busca. Os testes foram projetados para verificar não apenas o resultado da operação, mas também a manutenção das propriedades de balanceamento de cada árvore após as modificações.
- **Resultado:** Após a depuração e refatoração do código, a suíte de testes executa com **100% de aprovação**, indicando que ambas as implementações estão funcionando conforme o esperado para os cenários testados.
