UNIVERSIDADE FEDERAL DE MATO GROSSO
CAMPUS DO ARAGUAIA BARRA DO GARÇAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DANILO BARBOSA DA SILVA

GUSTAVO CAMERINO

Análise Experimental e Comparativa de Árvores Binárias de Busca

Barra do Garças - MT

2025

DANILO BARBOSA DA SILVA
GUSTAVO CAMERINO

Análise Experimental e Comparativa de Árvores Binárias de Busca

Este trabalho apresenta uma análise comparativa de três estruturas de árvores binárias de busca (**BST clássica**, **AVL** e **Rubro-Negra**), utilizando dados reais do repositório de Princeton (words.utf-8.txt). O estudo focou em medir a eficiência dessas estruturas em operações de **inserção** e **busca,** A proposta visa aplicar, na prática, os conceitos de Estruturas de Dados II, demonstrando como o balanceamento automático (AVL/RN) impacta no desempenho em comparação à BST padrão, especialmente em cenários com grandes volumes de dados (645 mil palavras)

Docente: Prof ° Dr. Robson .
Discente: Gustavo Camerino e Danilo Barbosa

Barra do Garças - MT

2025

1. Introdução

Árvores binárias de busca (BST) são estruturas de dados fundamentais para organização hierárquica de informações. Entretanto, sua eficiência depende fortemente da ordem dos dados inseridos. Este trabalho realiza uma análise comparativa entre três tipos de árvores: BST clássica, AVL e Rubro-Negra (RN), utilizando um dataset real contendo 645 mil palavras. As estruturas foram avaliadas quanto ao tempo de inserção, tempo de busca, altura da árvore e número de rotações realizadas.

2. Fundamentação Teórica

2.1 BST (Árvore Binária de Busca)

- **Propriedade:** Para cada nó, valores menores à esquerda e maiores à direita.
- **Complexidade:**
- Melhor caso (balanceada): O(logn) para inserção/busca.
- Pior caso (degenerada): O(n) (equivale a uma lista ligada)

2.2 AVL e Rubro-Negra

- **AVL:** Balanceamento rígido (altura das subárvores difere no máximo em 1).
- **RN:** Balanceamento flexível (regras de cores), com menos rotações que AVL.

3. Metodologia

3.1 Dados e Ambiente

- Dataset: words.utf-8.txt (645 mil palavras)
- Ambiente: Implementações próprias em linguagem C
- Os dados foram utilizados em sua ordem original (ordenada). Foi implementada uma lógica para evitar inserções duplicadas.

3.2 Métricas

Métrica	Descrição
Tempo de Inserção	Tempo para inserir todas as palavras.
Tempo de Busca	Média de 100 buscas aleatórias.
Altura	Altura máxima da árvore.
Rotações (AVL/RN)	Número total de rotações.

4. Resultados

4.1 Tabela Comparativa

Estrutura	Inserção (s)	Busca (s)	Altura	Rotações
BST	651.15	0.094	204475	N/A
AVL	0.15	0.00021	20	731743
RN	0.16	0.00019	34	767491

4.2 Análise da Lentidão da BST

A BST é ineficiente quando os dados estão ordenados. Neste caso, ela se torna uma lista ligada:

- Altura = n (645 mil)
    - Tempo de inserção: 651.15s

  Com balanceamento (AVL/RN):

    - Altura = 20 (AVL), 34 (RN)
    - Tempo de inserção: ~0.15s

  5. Discussão

  5.1 Eficiência da AVL e RN

    - Ambas garantem complexidade O(log n) para inserção e busca.
    - AVL: Altura mínima, mas mais rotações.
    - RN: Balanceamento mais leve, menos rotações.

5. Discussão

5.1 Por que AVL e RN são eficientes?

- **Balanceamento automático:**
    - AVL mantém altura mínima com rotações rigorosas.
    - RN usa regras de cores para balanceamento "aproximado".
- **Complexidade garantida:** O(logn) para todas as operações.

5.2 Quando usar cada estrutura?

Estrutura	Melhor Cenário	Pior Cenário
BST	Dados aleatórios/pequenos	Dados ordenados/grandes
AVL	Buscas frequentes	Inserções dinâmicas
RN	Inserções/remoções constantes	---

6. Conclusão

- **BST clássica** é **impraticável para dados ordenados** devido à degeneração.
- **AVL e RN** mantêm desempenho estável, com vantagens distintas:
    - AVL é mais rígida (altura mínima).
    - RN é mais flexível (menos rotações).
    - **Recomendação:** Usar AVL/RN para datasets grandes ou dinâmicos; BST apenas para dados aleatórios.