

# Análise de Árvores em Estruturas de Dados: Conceitos, Complexidade e Aplicações

Gabriel Yan, Nathan David, Pauline Fernandes, Rayna Livia

Centro Universitário Unieuro

{g.yan150506, nathandavidoliveiras2, paulinefernandesbraz10, raynalivia12}@gmail.com

**Resumo**—Este artigo explora conceitos fundamentais de árvores em estruturas de dados, abordando definições, operações em BSTs, técnicas de balanceamento (AVL e Red-Black) e aplicações práticas. Apresenta análise rigorosa de complexidade (Big O), comparativo entre técnicas de balanceamento e aplicações em sistemas críticos, demonstrando a eficiência de  $O(\log n)$  em operações essenciais.

**Index Terms**—Estruturas de dados, árvores, BST, AVL, Red-Black, complexidade, Big O, análise comparativa

## I. INTRODUÇÃO

As estruturas arbóreas representam componentes fundamentais na ciência da computação, fornecendo eficiência superior em operações de busca e organização de dados quando comparadas a estruturas lineares. Este artigo analisa sistematicamente árvores binárias de busca (BST), técnicas de balanceamento (AVL e Red-Black) e suas aplicações práticas. O objetivo principal é examinar conceitos teóricos, analisar complexidade computacional e demonstrar aplicações em sistemas reais. A metodologia inclui análise assintótica (Big O), comparação de técnicas e avaliação de casos de uso críticos.

## II. CONCEITOS E PROPRIEDADES

### A. Definição e Tipos

Árvores são estruturas hierárquicas não lineares caracterizadas por:

- **Nó raiz:** Elemento superior na hierarquia
- **Subárvores:** Estruturas aninhadas conectadas
- **Propriedade fundamental:**

$$\text{left-child} < \text{parent} < \text{right-child}$$

#### Tipos principais:

- Árvores binárias
- BST (Árvores Binárias de Busca)
- Árvores balanceadas (AVL, Red-Black)
- Árvores B/B+ (otimizadas para sistemas de armazenamento)

### B. Análise Big O de Operações

## III. ÁRVORES BINÁRIAS DE BUSCA (BST)

### A. Operações Fundamentais

#### Estrutura do nó:

```
typedef struct Node {  
    int key;
```

Tabela I  
COMPLEXIDADE DAS OPERAÇÕES EM BSTs

Operação	BST Balanceada	BST Degenerada
Inserção	$O(\log n)$	$O(n)$
Busca	$O(\log n)$	$O(n)$
Remoção	$O(\log n)$	$O(n)$

```
    struct Node *left, *right;  
} Node;
```

#### Busca recursiva:

```
Node* search(Node *root, int key) {  
    if (!root || root->key == key)  
        return root;  
    if (key < root->key)  
        return search(root->left, key);  
    return search(root->right, key);  
}
```

### B. Análise de Complexidade

- **Melhor caso:**  $O(1)$  (chave na raiz)
- **Caso médio:**  $O(\log n)$  (árvore balanceada)
- **Pior caso:**  $O(n)$  (árvore degenerada em lista)

## IV. APLICAÇÕES PRÁTICAS

### A. Bancos de Dados

- **Estrutura:** Árvores B+
- **Operação:** Indexação de registros
- **Eficiência:** Busca em  $O(\log_k n)$  ( $k$  = fator de ramificação)

### B. Árvores de Decisão (IA)

- **Aplicação:** Classificação de dados
- **Complexidade:**  $O(d \times \log n)$  ( $d$  = profundidade)

### C. Sistemas de Arquivos

- **Exemplo:** Sistemas EXT4 (Linux)
- **Vantagem:** Acesso  $O(\log n)$  vs  $O(n)$  em listas

## V. ÁRVORES BALANCEADAS

### A. AVL (Balanceamento por Altura)

#### Rotação à direita:

```
Node* rightRotate(Node *y) {
```

```

Node *x = y->left;
Node *T2 = x->right;
x->right = y;
y->left = T2;
// Atualização de alturas
y->height = max(height(y->left), height(y->right));
x->height = max(height(x->left), height(x->right));
return x;
}

```

#### Regras de balanceamento:

- if (balance  $\geq 1$  && key  $\leq$  left-key)  $\rightarrow$  Rotação direita
- if (balance  $\leq -1$  && key  $\geq$  right-key)  $\rightarrow$  Rotação esquerda

#### B. Red-Black

##### Propriedades essenciais:

- 1) Todo nó é vermelho ou preto
- 2) Raiz sempre preta
- 3) Folhas (NIL) pretas
- 4) Nós vermelhos têm filhos pretos
- 5) Caminhos igualmente pretos da raiz às folhas

##### Complexidade:

- Inserção:  $O(\log n)$
- Remoção:  $O(\log n)$
- Balanceamento:  $O(1)$  por operação

#### C. Análise Comparativa

Tabela II  
COMPARAÇÃO AVL VS RED-BLACK

Característica	AVL	Red-Black
Fator de balanceamento	Altura	Cor
Rotações (pior caso)	$O(1)$	$O(1)$
Complexidade inserção	$O(\log n)$	$O(\log n)$
Aplicação típica	Buscas intensivas	Inserções frequentes
Balanceamento	Mais rígido	Mais flexível

#### D. Otimizações

- **AVL:** Armazenamento do fator de balanceamento para evitar recálculos
- **Red-Black:** Uso de nós sentinelas para simplificar operações
- **Geral:** Algoritmos iterativos para redução de overhead

### VI. IMPLEMENTAÇÃO E TESTES

#### A. Abordagem

Implementamos estruturas AVL e Red-Black em C, seguindo as especificações teóricas. Para garantir eficiência:

- Otimização de acesso à memória
- Algoritmos iterativos para operações críticas
- Testes de estresse com até  $10^6$  elementos

#### B. Resultados de Desempenho

##### Observações:

- AVL mostra superioridade em operações de busca
- Red-Black apresentou melhor desempenho em inserções sequenciais
- Degeneração de BST impacta diretamente o desempenho

Tabela III  
COMPARAÇÃO DE DESEMPENHO (TEMPO EM MS)

Elementos	BST	AVL
1,000	2.1	0.3
10,000	24.7	1.9
100,000	152.8	12.4

### VII. CONCLUSÃO

Este trabalho demonstrou que:

- **BSTs** fornecem operações eficientes ( $O(\log n)$ ) quando balanceadas
- Técnicas de **balanceamento (AVL/Red-Black)** são essenciais para evitar degeneração em  $O(n)$
- **AVL** é ideal para cenários com buscas intensivas
- **Red-Black** apresenta melhor desempenho em atualizações frequentes
- Aplicações críticas como bancos de dados dependem de **árvores B+** para garantir  $O(\log n)$  em operações de disco

O repositório com análise completa e implementação está disponível em: <https://github.com/seugrupo/arvores-ed>

### REFERÊNCIAS

- [1] T. H. Cormen, *Introduction to Algorithms*. MIT Press, 2009.
- [2] D. E. Knuth, *The Art of Computer Programming, Vol. 3*. Addison-Wesley, 1998.
- [3] R. Bayer, "Symmetric Binary B-Trees," *Acta Informatica*, vol. 1, pp. 290-306, 1972.
- [4] R. Sedgewick, *Algorithms in C*. Addison-Wesley, 1990.
- [5] G. Adelson-Velskii, E. Landis, "An algorithm for the organization of information," *Proceedings of the USSR Academy of Sciences*, 1962.