AED2 - Algoritmos e Estr. de Dados II

Aula 7: Árvores AVL

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Fevereiro de 2026

Sumário

1. O Problema da BST "Degenerada"

Como vimos, se inserirmos `1, 2, 3, 4` em uma BST, ela vira uma Linha Reta com altura N. A busca degrada para $O(N)$. Para resolver isso, G.M. **A**delson-**V**elsky e E.M. **L**andis criaram, em 1962, a primeira árvore balanceada auto-ajustável.

2. Invariante da AVL

BST é AVL se $|FB| \leq 1$ para todo nó.

$$FB = h(\text{dir}) - h(\text{esq}) \in \{-1, 0, +1\}$$

Se $|FB| > 1$ após inserção/remoção, aplicar rotação.

3. Implementação: Calculando Altura

Diferente da BST comum, cada nó da AVL precisa armazenar sua própria altura para evitar recalcular recursivamente (o que seria $O(N)$).

```
class Node {
    int key, height;
    Node left, right;

    Node(int d) {
        key = d;
        height = 1; // Nó novo é folha, altura 1
    }
}

int height(Node N) {
    if (N == null) return 0;
    return N.height;
}
```

4. As Rotações

As rotações são as operações mágicas que reduzem a altura da árvore localmente mantendo a ordem da BST ($mq < Raiz < Mq$).

4.1. Rotação Simples à Direita (Right Rotate)

Usada quando o desbalanceamento é na **Esquerda-Esquerda** (O filho esquerdo está pesado na esquerda).

```
graph TD
    y((y)) --> x((x))
    y --> T3
    x --> T1
    x --> T2
```

Vira:

```
graph TD
    x((x)) --> T1
    x --> y((y))
```

5. Inserção Completa

```
Node insert(Node node, int key) {
    // 1. Inserção normal de BST
    if (node == null) return (new Node(key));

    if (key < node.key)
        node.left = insert(node.left, key);
    else if (key > node.key)
        node.right = insert(node.right, key);
    else
        return node; // Chaves duplicadas não permitidas

    // 2. Atualiza altura do ancestral
    node.height = 1 + Math.max(height(node.left), height(node.right)
        );

    // 3. Verifica Balanceamento
    int balance = getBalance(node);
```

6. AVL vs Árvore Rubro-Negra

Característica	AVL	Rubro-Negra
Balanceamento	Rígido ($\|FB\| \leq 1$)	Relaxado
Altura	$\approx 1{,}44 \log N$	$\approx 2 \log N$
Busca	Mais rápida	Levemente mais lenta
Inserção/Remoção	Mais rotações	Menos rotações
Uso	Muita leitura	TreeMap, std::map

Conclusão

A AVL é excelente para cenários $_{R}ead - Heavy_{(}muitasbuscas, poucasalteraçõ es).Sehouvermuitasescritas, aÁrvoreRubro - Negraépreferível.$