Algoritmos e Estruturas de Dados II

Capítulo da Aula 8: Árvores Rubro-Negras

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br
CEFET-MG - Campus Timóteo

Fevereiro de 2026

Introdução e Contexto Histórico

As Árvores Rubro-Negras (*Red-Black Trees*) representam um dos marcos mais importantes no estudo de estruturas de dados auto-balanceadas. Inventadas em 1978 por Leonidas Guibas e Robert Sedgewick, elas foram inspiradas nas "Árvores Simétricas Binárias B" de Rudolf Bayer (1972), hoje conhecidas como Árvores 2-3-4.

Diferente das Árvores AVL, que buscam um balanceamento rígido (diferença de altura máxima de 1), as Rubro-Negras permitem um balanceamento mais "relaxado". Elas utilizam um bit extra de informação em cada nó — sua **cor** (Vermelho ou Preto) — para impor regras que garantem que nenhum caminho da raiz até uma folha seja mais que o dobro de qualquer outro caminho. Essa flexibilidade resulta em um custo menor de reequilíbrio (menos rotações) durante inserções e remoções, tornando-as a escolha predileta para implementações de bibliotecas de sistema e kernels.

A Motivação: O Fim das Listas Encadeadas Disfarçadas

O grande problema das Árvores Binárias de Busca (BST) simples é o cenário de pior caso. Se inserirmos chaves em ordem crescente, a árvore torna-se uma linha reta. Para um conjunto de $N = 1.000.000$ de elementos, uma busca em uma BST degenerada exigiria até 1 milhão de comparações ($O(N)$), o que é computacionalmente inviável para aplicações de tempo real.

Em contraste, as soluções de balanceamento automático garantem performance logarítmica ($O(\log N)$):

- **AVL:** Altura máxima $\approx 1,44 \log_2 N$. Para $N = 10^6$, a altura é ≈ 29.
- **Rubro-Negra:** Altura máxima $\leq 2 \log_2(N + 1)$. Para $N = 10^6$, a altura é ≈ 40.

Embora a Rubro-Negra seja ligeiramente mais alta que a AVL, a diferença de 11 níveis é insignificante frente ao ganho de eficiência nas operações de escrita (inserção e remoção), que exigem **no máximo 2 ou 3 rotações** para reequilibrar a estrutura, enquanto a AVL pode exigir $O(\log N)$ rotações em remoções.

Definição e as 5 Propriedades Fundamentais

Uma Árvore Rubro-Negra é uma BST na qual cada nó é colorido de vermelho ou preto. Para manter o balanceamento, as seguintes 5 propriedades de cor devem ser satisfeitas **sempre**:

1. **Cor:** Todo nó deve ser obrigatoriamente **Vermelho** ou **Preto**.
2. **Raiz:** A raiz deve ser sempre de cor **Preta**.
3. **Folhas (NIL):** Todas as folhas nulas (`null`) são consideradas **Pretas**.
4. **Vermelho-Vermelho:** Se um nó é Vermelho, ambos os seus filhos **devem** ser Pretos. Em outras palavras, não pode haver dois nós vermelhos consecutivos em qualquer caminho.
5. **Altura Negra:** Para qualquer nó x, todos os caminhos simples de x até folhas descendentes contêm o **mesmo número de nós Pretos**.

• Teoria

Por que essas regras funcionam? Imagine o caminho mais curto possível da raiz até a folha (composto apenas por nós pretos) e o caminho mais longo possível (onde alternamos Preto-Vermelho-Preto). Pela regra da Altura Negra, ambos devem ter o mesmo número de pretos. Pela regra Vermelho-Vermelho, não podemos ter dois vermelhos seguidos. Logo, o caminho mais longo nunca terá mais que o dobro do comprimento do caminho mais curto.

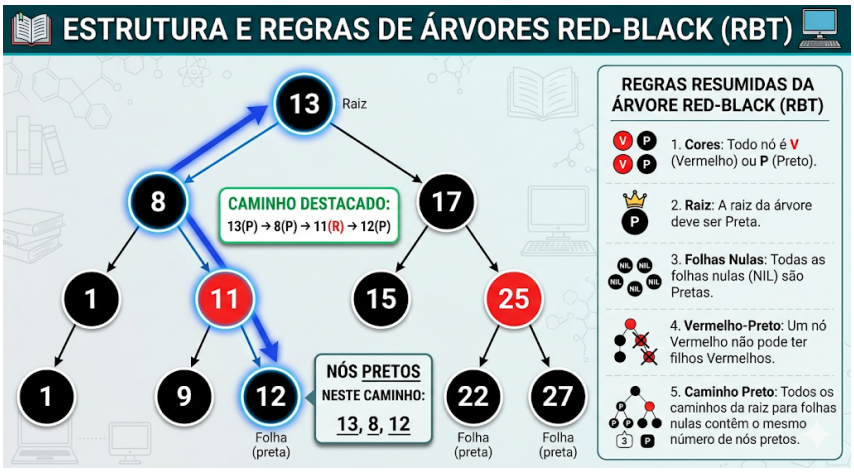

Figure 1: Propriedades rubro-negras garantindo altura balanceada e custo logarítmico.

Implementação Técnica: O Segredo do Nó Sentinela NIL

Uma das maiores dificuldades ao implementar Árvores Rubro-Negras é lidar com os diversos checks de `null` ao acessar o pai ou os filhos durante rotações e recolorações. A solução elegante herdada do livro CLRS (*Introduction to Algorithms*) é o uso de um **Nó Sentinela NIL**.

O NIL é um único objeto estático, de cor negra, para o qual apontam todas as referências que seriam nulas (`root.parent`, `folhas.left`, ...). Isso permite que o algoritmo trate folhas como nós reais, simplificando drasticamente o código.

Listing 1: Implementação da Estrutura do Nó e Sentinela

```
class Node {
    int key;
    boolean isRed = true; // Novos nos nascem vermelhos
    Node left, right, parent;

    Node(int key) {
        this.key = key;
    }
}

// Singleton NIL (Preto por definicao)
static final Node NIL = new Node(0);

static {
    NIL.isRed = false;
    NIL.left = NIL.right = NIL.parent = NIL;
}
```

Operações Fundamentais: Rotações

As rotações são operações locais que preservam a ordem da BST ($E < Raiz < D$), mas alteram a estrutura para diminuir a altura de um lado e aumentar a do outro.

Listing 2: Implementação das Rotações

```
void rotateLeft(Node x) {
    Node y = x.right;
    x.right = y.left;

    if (y.left != NIL) {
        y.left.parent = x;
    }

    y.parent = x.parent;
```

```
    if (x.parent == NIL) {
        root = y;
    } else if (x == x.parent.left) {
        x.parent.left = y;
    } else {
        x.parent.right = y;
    }

    y.left = x;
    x.parent = y;
}

void rotateRight(Node y) {
    Node x = y.left;
    y.left = x.right;

    if (x.right != NIL) {
        x.right.parent = y;
    }

    x.parent = y.parent;

    if (y.parent == NIL) {
        root = x;
    } else if (y == y.parent.right) {
        y.parent.right = x;
    } else {
        y.parent.left = x;
    }

    x.right = y;
    y.parent = x;
}
```

Lógica de Inserção e Casos de Correção

Todo novo nó Z ingressa arquiteturalmente na árvore pintado estritamente de **Vermelho**. Essa escolha de engenharia é genial: fundir um novo nó avermelhado **nunca** ataca nem altera a matriz da Altura Negra (Propriedade 5). No entanto, se o pai da folha recém-chegada (P) já detinha precariamente a coloração Vermelha, disparamos um curto-circuito arquitetural por quebra física direta (*double-red violation*).

Nesses milissegundos críticos, a engine aciona emergencialmente o fixador de balanceamento. A gigantesca sacada para solucionar o entrave é observar o parentesco adjacente de forma lateral: nós destrinchamos e checamos obrigatoriamente a coloração térmica

residente no **Tio** (U) do nó inserido.

Os 3 Domínios da Inserção

O Domínio 1: Tio Vermelho (Propagação Abstrata por Recoloração)
Se o Pai (P) e o **Tio** (U) do nó problemático compartilham ambos da estrita cor vermelha, subentende-se irrefutavelmente que por lei o Avô pivô local (G) repousou pintado de negro. Para diluir o atrito físico estourado entre as linhas sem mexer ou adulterar a gravidade assintótica bruta de Alturas no galho intersecional inteiro, esguichamos um reposicionamento térmico: **Recolorimos** P e U injetando a resina Negra isoladora e aquecemos retroativamente o nó avô G transferindo sua cor fria para estourar o limite **Vermelho**.
Como o próprio Avô subjacente aglutinador G acabou de amanhecer em chamas puras rubras, sufocamos a inflamação nas folhas originárias, mas perigosamente propensos a chocar esse novo avô recém avermelhado contra ligações primitivas superiores (o bisavô da triagem)! Caso sim, o framework processual simplesmente impulsiona o cursor e a inflamação sobe livre estagnando recursivamente degraus adiante e dissipando o calor no ápice mor isolante absoluto Supremo (Raiz `NIL` estagnada em Negro inquebrável).

Os Domínios 2 e 3: Tio Preto ou `NIL` (Resolução Físico/Dinâmica por Rotação)
Se o estático ramo marginal do **Tio** (U) absorver e conter todo envoltório imposto se revestindo das massas cinzentas **Pretas** da base (ou abismado pela Folha limite fantasma terminal `NIL`), a banal via da "recoloração purista" está matematicamente morta, pois a inversão térmica arruinaria a proporção blindada das ramificações matrizes da Altura Negra global para as alas do Avô. Assim a engrenagem exige torções de translocamento físico base atreladas às quebras das ligações:

- **A Etapa Preparatória (O Joellho "Zig-Zag" / Caso 3):** Se a malha ramificada descender engasgada "quebrada" pelo meio nas entranhas (ex: o pai P ancora à esquerda estrita do Avô pivô, mas injetamos o anomalo Z cravado reverso para a aresta direta dele), a máquina detona micro-rotações primárias espirais. Ela subverte o ângulo cravando a chave do nó base no trânsito reverso para tracionar P. Estourando os cabos de arestas ele injeta a topologia para o formato limpo, transferindo perfeitamente o estrago irregular e transformando o aglomerado para o temido Caso 2 puro.
- **A Cura Desbalanceada (O Alinhamento Linear "Zig-Zig" / Caso 2):** Com todas as entidades e pontas do Trio (Avô, Pai e Neto) desdobradas retilíneas pelo escopo de um único horizonte físico estrito, despeja-se a brutal rotação gravitacional focada inteira pelo pivô base oposto de retenção no Avô (G). Reverte-se imune de uma só carga as películas das tintas dos blocos desmembrados atritantes: pinta a raiz substituta (P) estrita para blindagem da carga na cor final **Preta**, e relaxa o antigo ancestral deposto (G) para **Vermelho**. Esta manobra majestosa esmaga permanentemente em pífio ciclo curto o nó atípico cravando $O(1)$ garantias processuais plenas por base estrutural isolada e interrompendo recursões massivas superiores.

O Desafio Assintótico: A Remoção e o "Duplo-Preto"

Diferente do mar aberto e fluido das injeções construtivas recrescidas rubras, as estritas operações de **extração de carga e deleção** materializam em forma isolada os algoritmos mais impiedosos e intrincados da lógica computacional sistêmica. O processo engole fardos massivos em testes agressivos de resistência corriqueiros aplicados livremente pela depuração nos Kernels de pesados hipervisores.

Excluir e remover um sub-nó atômico unicamente e passivo portador de essência **Vermelha** da estrutura representa apenas calmaria e alívio processual; sua estirpação passiva jamais fere, rasga ou afeta as raízes matemáticas ativas retentoras da malha profunda da Altura Negra.

A catástrofe abismal nas rubro-negras desponta implacavelmente apenas quando disparam compulsoriamente a ruptura na casca protetora excluindo por fim uma âncora inteiriça da classe pesada contendo a coloração matriz purista **Preta**. Desprezando este limite engessado, quebra-se severamente todos os escoamentos topológicos pendentes submersos que descendiam vitais correndo e afunilando através de suas estáticas vias, roubando fisicamente massas irrecuperáveis e rebaixando o saldo contábil da trilha restritiva global reduzindo tudo de vez para um degrau negativo obscuro absoluto que rompe universalmente a base inteira da Altura Negra para qualquer percurso da aresta podada.

Para estancar e mascarar fisicamente a hemorragia topológica severa, a engine emula impiedosamente a carga estourada num peso virtual artificial alucinatório: o rebento filho do deserdado recebe por herança integral todo o vácuo temporal obscuro do extinto, abarcando as duas massas combinatórias da carga subornada no assustador espectro batizado de entidade do sub-nó fantasma **Duplo-Preto**.

Este evento monstruoso dispara um frenético radar vasculhador que orquestrará a análise de pesagens de colorações focadas de reestruturação de forma integral contra um ecossistema estático de "irmãos", operando torções insanas até 4 eixos dinâmicos de transferências estruturais simultâneas (o irmão, o irmão da ramificação dos filhos do irmão e transferências de pais atrelados ao vazio). Esses reparos garantem que, em poucos pulsos absolutos milimétricos com um estrago $\mathcal{O}(\log N)$, a árvore retome total flexibilidade balanceadora sem sucumbir sistemicamente a degenerações brutas comuns enfrentadas pelas restrições físicas travadas das subamostras binárias simples!

Sistemas Reais e Alta Performance

O uso das Árvores Rubro-Negras não é apenas acadêmico. Elas são o motor de sistemas de alta performance:

- **Linux CFS Scheduler:** O kernel do Linux mantém uma RB-Tree de processos ativos ordenada pelo `vruntime` (tempo virtual de CPU). O processo com menor vruntime (mais à esquerda na árvore) é o próximo a ser executado. Como processos entram e saem da fila de execução milhares de vezes por segundo, o custo baixo de rotação da RB-Tree é vital.
- **Java HashMap (Bifurcação de Buckets):** A partir do Java 8, quando uma lista de colisões em um `HashMap` atinge o tamanho de 8 elementos, ela é automaticamente con-

vertida em uma RB-Tree para garantir que ataques de colisão maliciosa não degradem a performance de $O(1)$ para $O(N)$, limitando o pior caso em $O(\log N)$.

Comparativo Final: Quando usar?

Estrutura	Vantagem Principal	Caso de Uso Ideal
`HashMap`	Velocidade média $O(1)$	Tabelas de símbolos simples.
`AVL Tree`	Busca mais rápida (h menor)	Bases de dados de leitura intensiva.
`RB-Tree`	Menos rotações na escrita	Kernels, sistemas de arquivos, mapas genéricos.

Table 1: Comparação de trade-offs entre estruturas balanceadas.

Exercícios Propostos

Conceituais e Teóricos

1 Explique por que a cor inicial de um nó ao ser inserido deve ser vermelha e não preta. Refira-se às 5 propriedades.

2 Demonstre, através de um desenho, como o Caso 1 (recoloração) pode gerar uma nova violação em um nível superior da árvore.

Analíticos e Práticos

1 Desenhe a árvore rubro-negra resultante da inserção dos valores: 2, 1, 4, 5, 9, 3, 6, 7. Mostre os passos de rotação e recoloração.

2 No Java, quais as vantagens de um `TreeSet` sobre um `HashSet` em um sistema que precisa exibir os usuários ativos em ordem alfabética?