

ESD 2

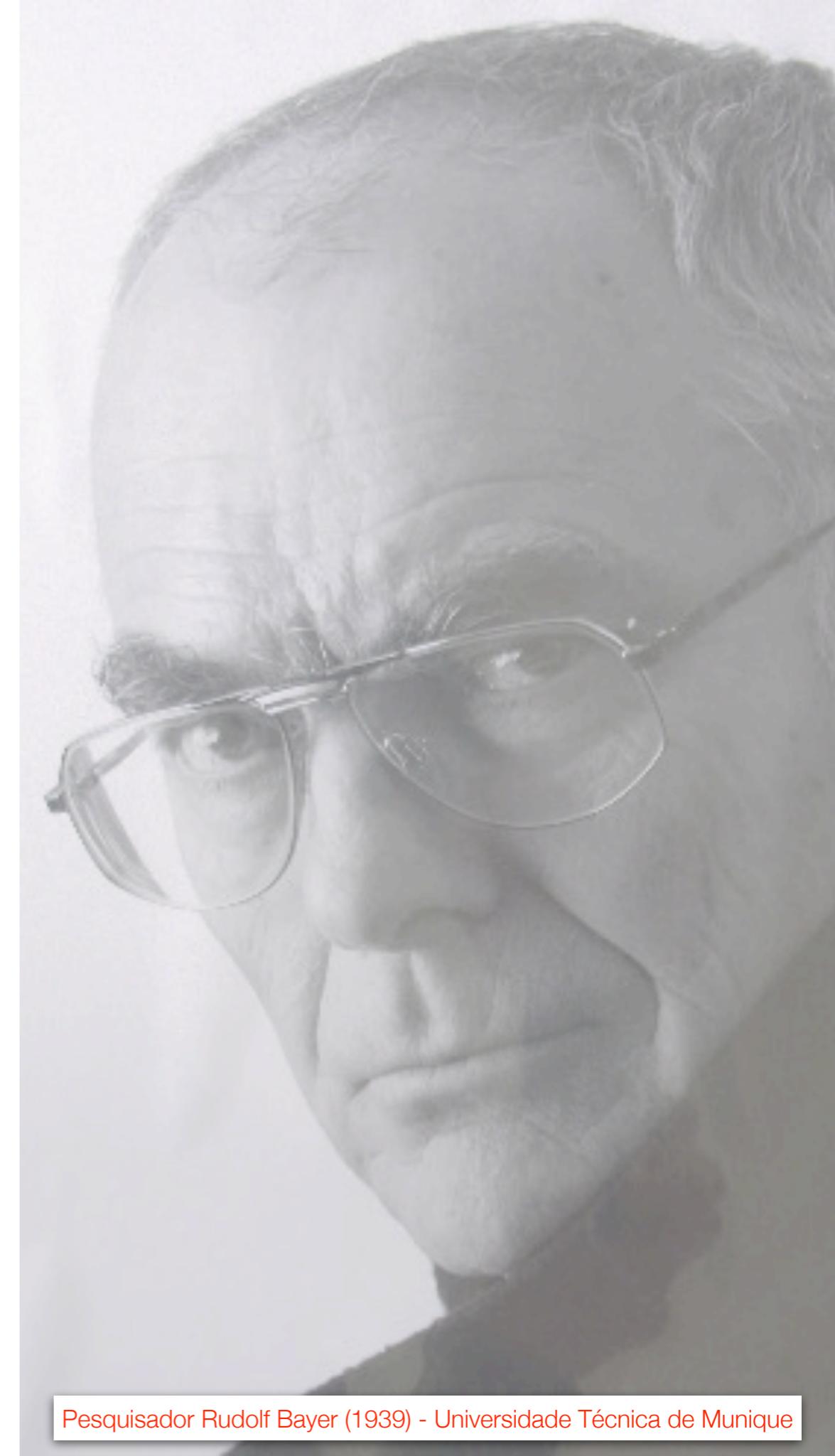
Estrutura de
Dados

Árvores Rubro-Negras

Prof. Fernando Sambinelli

Introdução

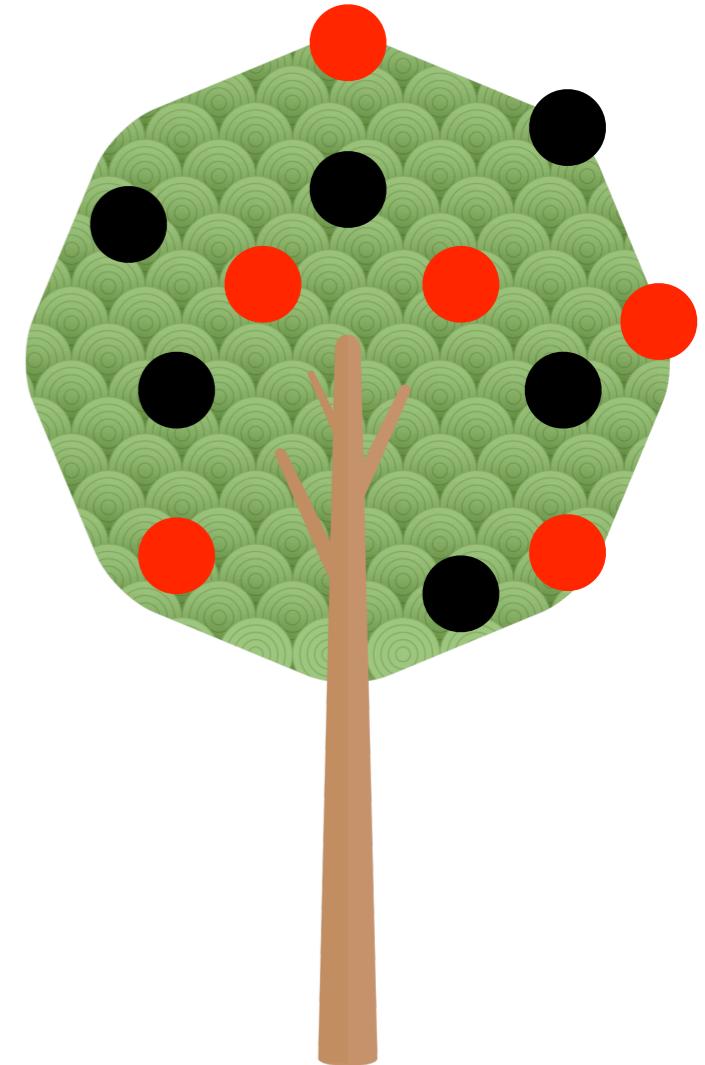
- Uma árvore rubro-negra (ARN) é um tipo de árvore binária de busca balanceada, assim como a AVL.
- Foi originalmente inventada, em 1972, por Rudolf Bayer, da Universidade Técnica de Monique (Alemanha).
- A árvore rubro-negra tem um bom pior caso, em termos de complexidade de execução - O $(\log n)$, quando comparada a uma árvore binária de busca comum.



Pesquisador Rudolf Bayer (1939) - Universidade Técnica de Munique

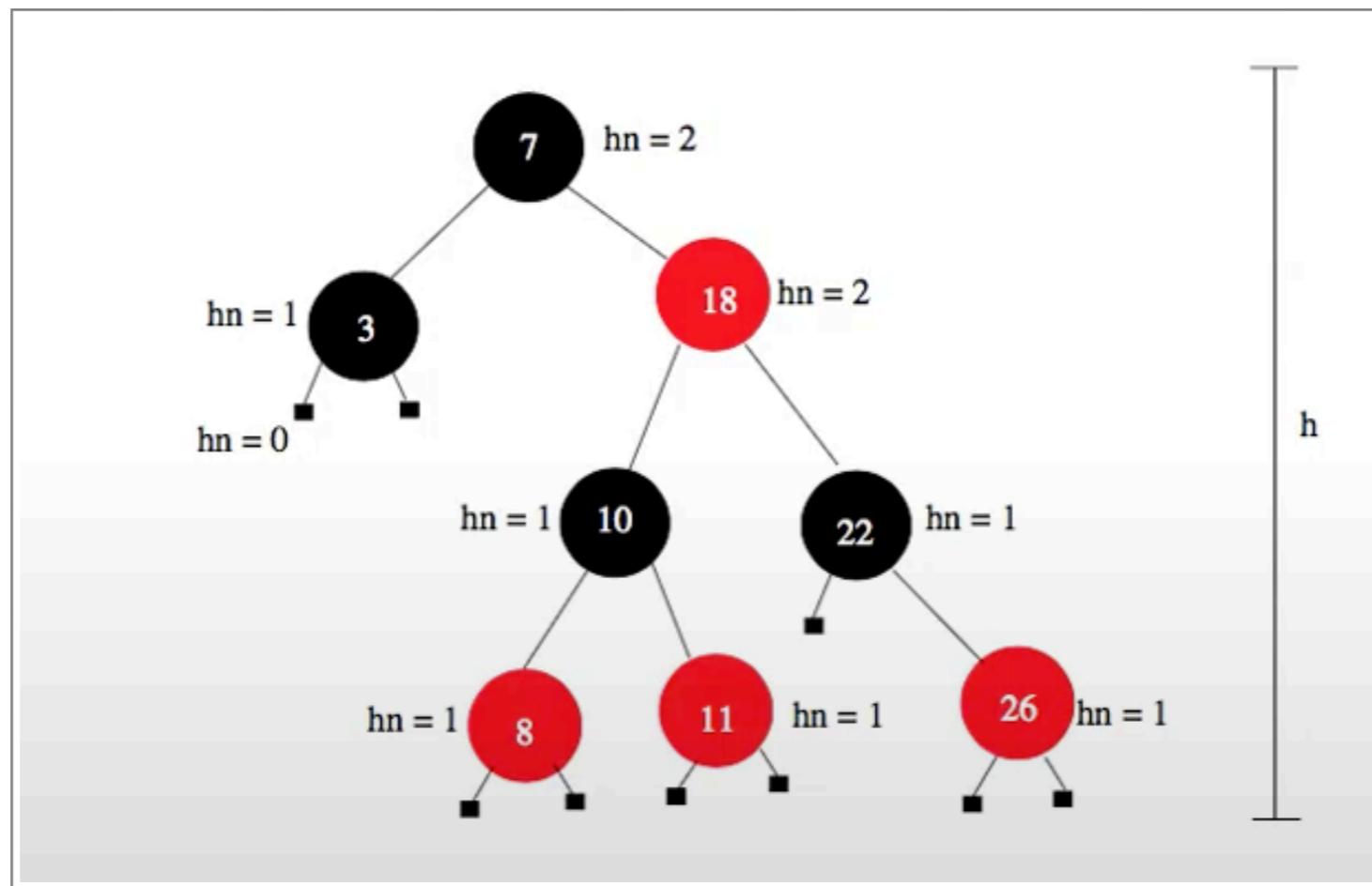
Conceitos sobre ARN

- É uma ABB com **um atributo extra de cor** para cada nó.
- **Propriedades:**
 1. Todo nó da árvore ou é vermelho ou é preto
 2. A raiz é preta
 3. Toda folha (null) é preta
 4. Se um nó é vermelho, então ambos os filhos são pretos.
 5. Para todo nó, todos os caminhos do nós até as folhas descendentes contém o mesmo número de nós pretos (altura negra)



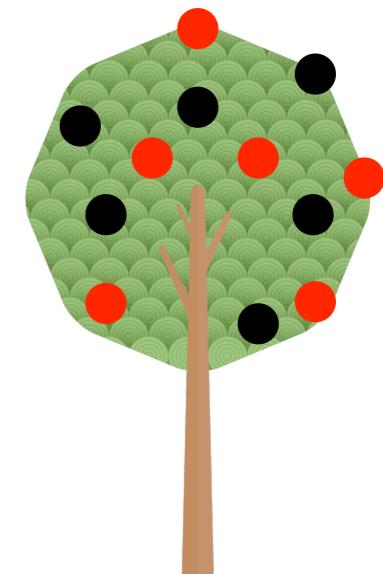
Altura Negra

- **Definição:** A altura negra de um nó x , $h_n(x)$ é o número de nós pretos de um caminho até uma folha descendente, excluindo x .



Altura da ARN

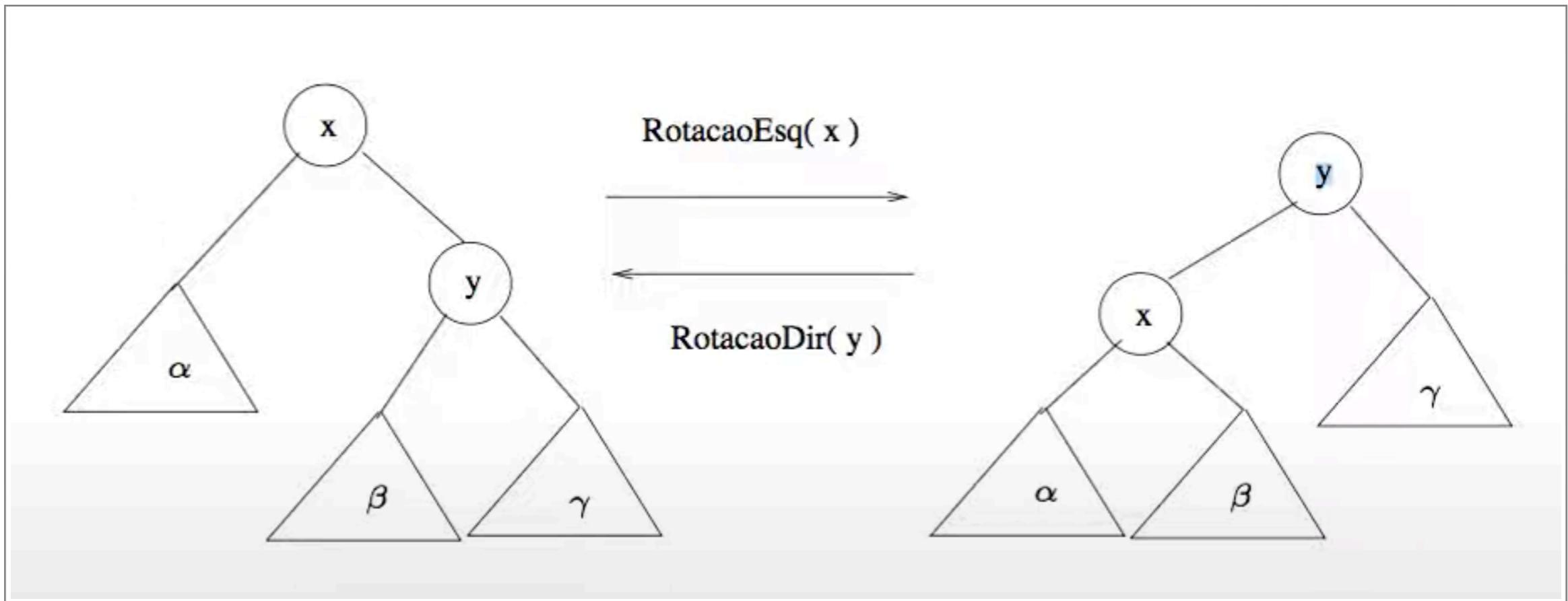
- **Definição:** A árvore rubro-negra com n chaves tem altura $h < 2 \log(n + 1)$
- **Exemplo:**
 - uma árvore com 1023 chaves (valores) terá altura máxima 19
 - $h < 2 \log(1023 + 1) \Rightarrow h < 2 \log(1024) \Rightarrow h < 2 * 10 \Rightarrow h < 20$
 - Logo, neste exemplo, essa ARN terá um $h < 20$, ou seja, no máximo 19



Assim como a AVL, o desafio da ARN é mantê-la balanceada, por meio de rotações, conforme os nós são adicionados ou removidos

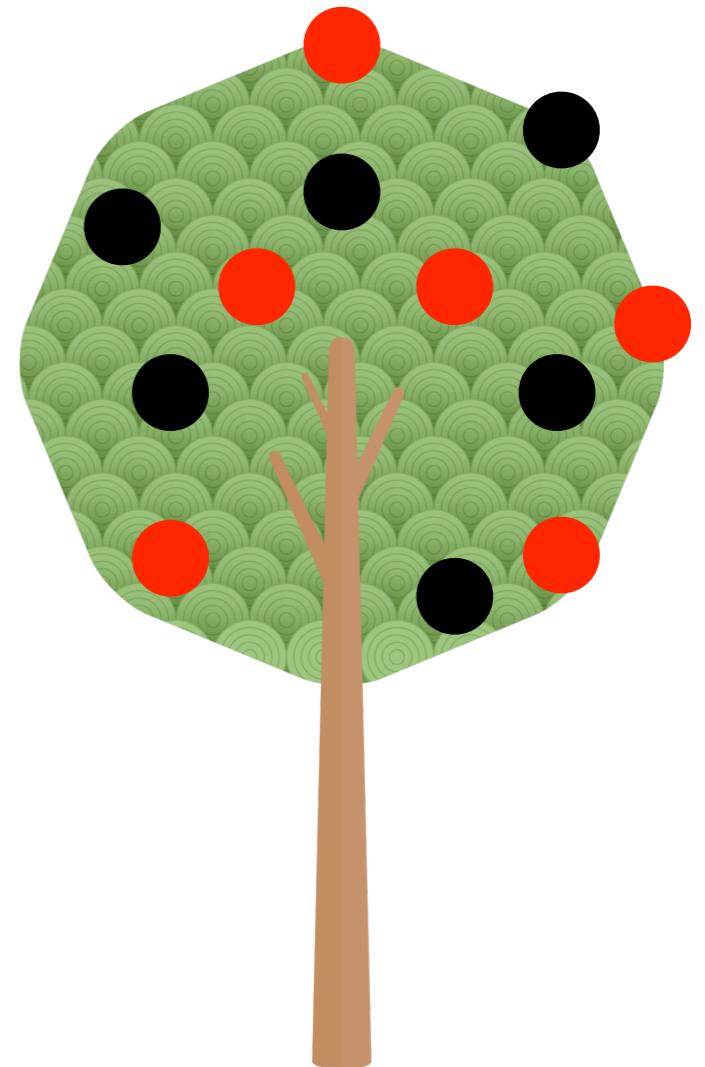


Rotações



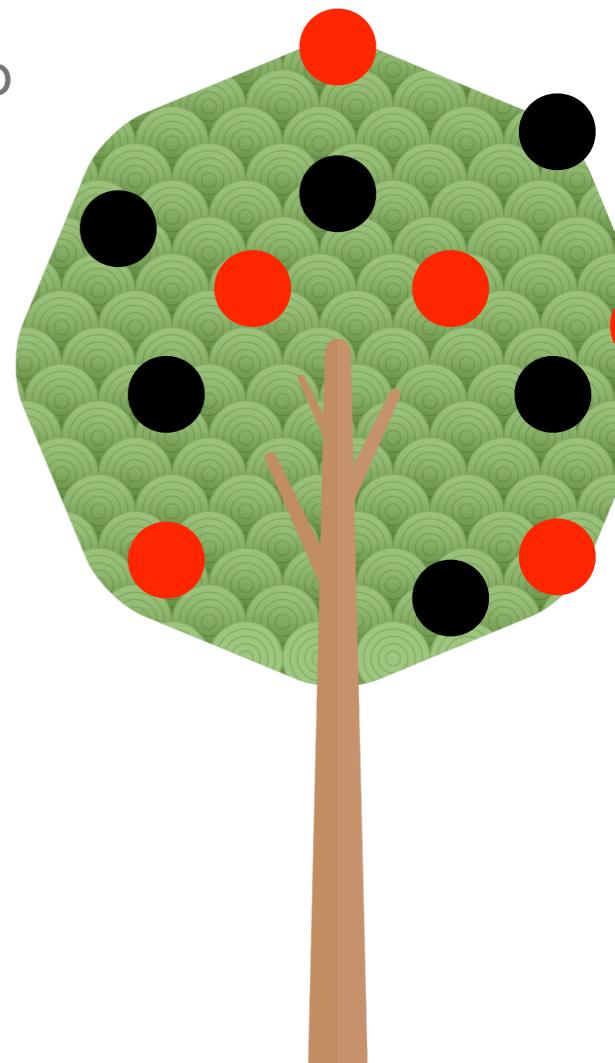
Operação de Inclusão

- A inclusão em uma ARN deve seguir os seguintes passos:
 - Insira um nó X como em uma ABB
 - Pinte o nó X (recém adicionado) de **VERMELHO**
 - Restaure as propriedades rubro-negras
 - Use rotações e recoloração

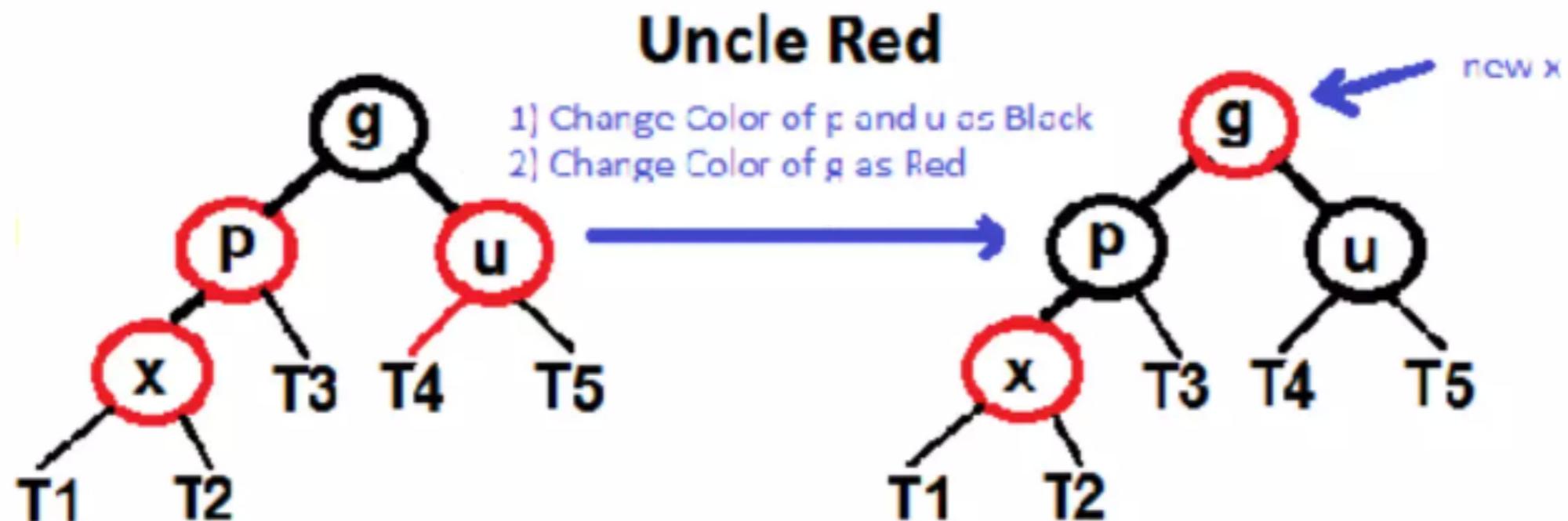


Operação de Inclusão: Casos Especiais

- Há 5 casos a considerar quando X e o pai de X são vermelhos:
 - Caso 1: X tem um tio U vermelho
 - Caso 2: X tem tio U preto, o pai P é filho esquerdo e X é filho direito
 - Caso 3: X tem tio U preto, o pai P é filho esquerdo e X é filho esquerdo
 - Caso 4: X tem tio U preto, o pai P é filho direito e X é filho esquerdo
 - Caso 5: X tem tio U preto, o pai P é filho direito e X é filho direito
- Observações:
 - O pai de X não pode ser a raiz da árvore, pois é vermelho



Caso 1: X tem tio U **vermelho**



x: Current Node, p: Parent; u: Uncle, g: Grandparent

T1, T2, T3, T4 and T5 are subtrees

```
noReferenciaU.setCor(PRETO);
```

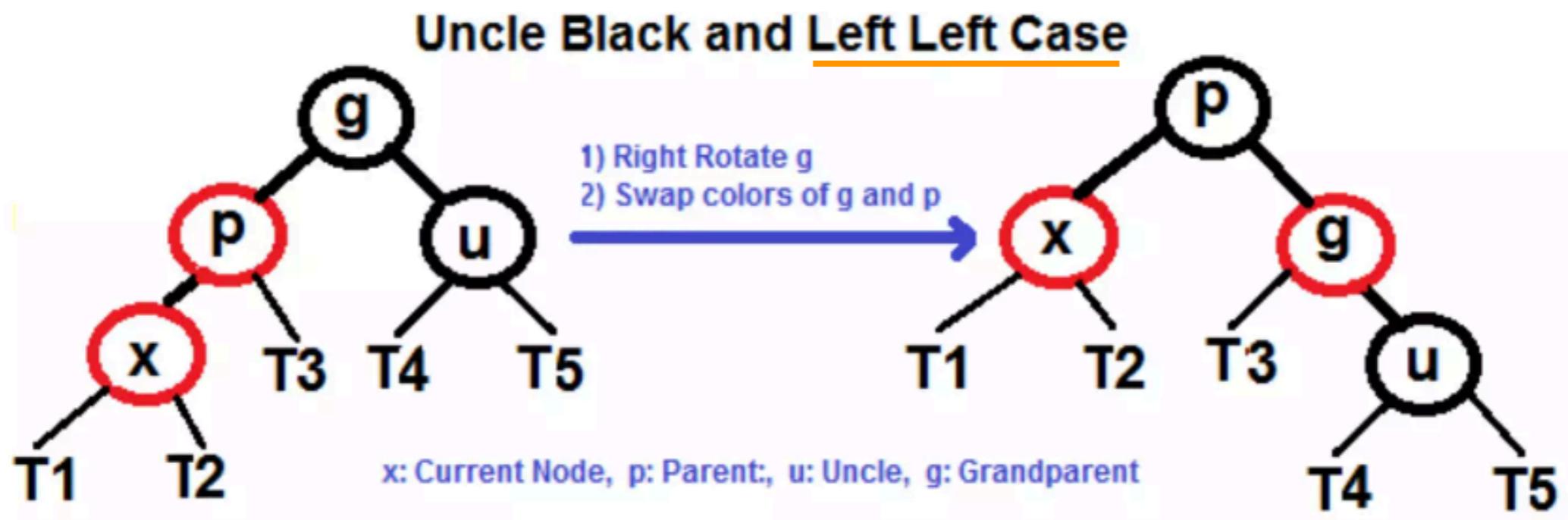
```
noReferenciaX.getPai().setCor(PRETO);
```

```
noReferenciaX.getPai().getPai().setCor(VERMELHO)
```

=> Mude X = avô de X, repita os passos anteriores

*observações:
(1) se G fosse raiz, então, G seria preto.
(2) X é o nó recém adicionado
(3) tem caso simétrico

Caso 3: X tem tio U preto, o pai P é filho esquerdo e X é filho esquerdo



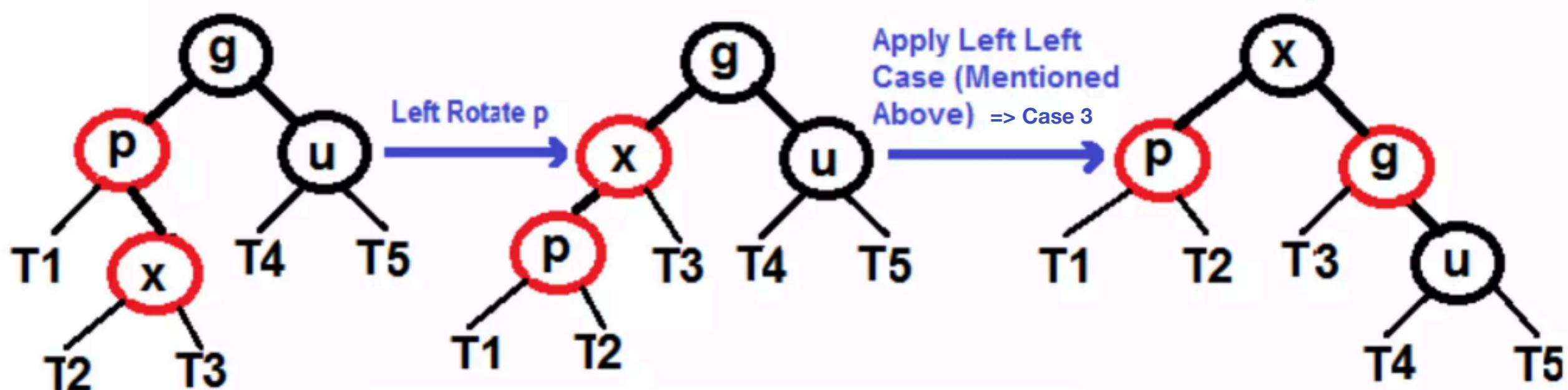
```
noReferenciaX.getPai().setCor(PRETO);
```

```
noReferenciaX.getPai().getPai().setCor(VERMELHO);
```

```
RotacaoDireita (noReferenciaX.getPai().getPai())
```

Caso 2: X tem tio U preto, o pai P é filho esquerdo e X é filho direito

Uncle Black and Left Right Case



```
noReferenciaX = noReferenciaX.getPai();
```

```
RotacaoEsquerda(noReferenciaX);
```

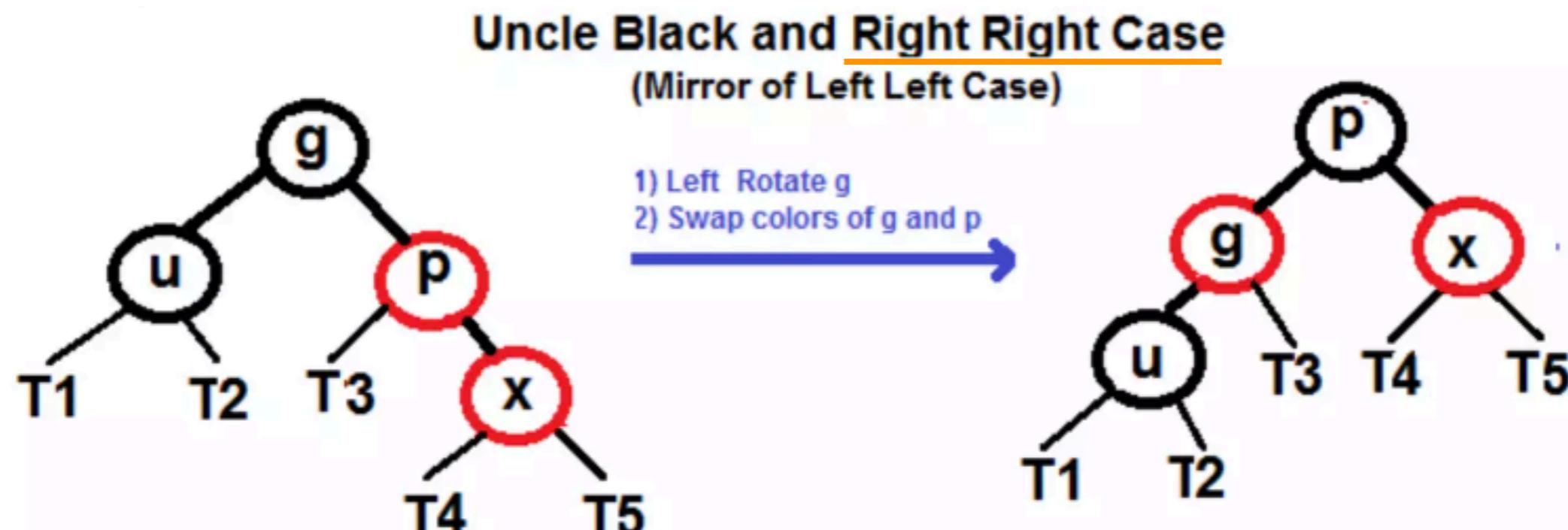
** em seguida, sempre aplicar **Caso 3**

```
noReferenciaX.getPai().setCor(PRETO);
```

```
noReferenciaX.getPai().getPai().setCor(VERMELHO);
```

```
RotacaoDireita (noReferenciaX.getPai().getPai())
```

Caso 5: X tem tio U preto, o pai P é filho direito e X é filho direito

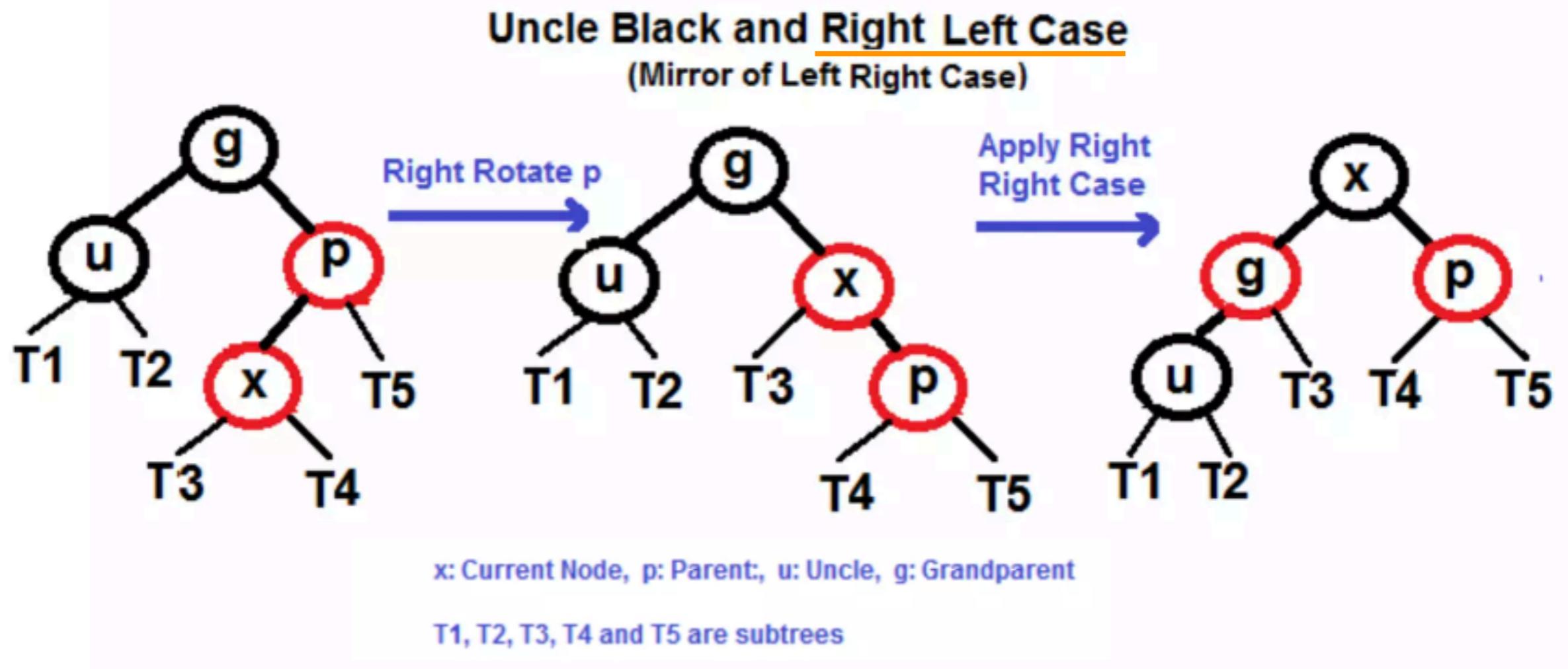


```
noReferenciaX.getPai().setCor(PRETO);
```

```
noReferenciaX.getPai().getPai().setCor(VERMELHO);
```

```
RotacaoEsquerda (noReferenciaX.getPai().getPai())
```

Caso 4: X tem tio U preto, o pai P é filho direito e X é filho esquerdo



```
noReferenciaX = noReferenciaX.getPai();
```

```
RotacaoDireita(noReferenciaX);
```

** em seguida, sempre aplicar **Caso 5**

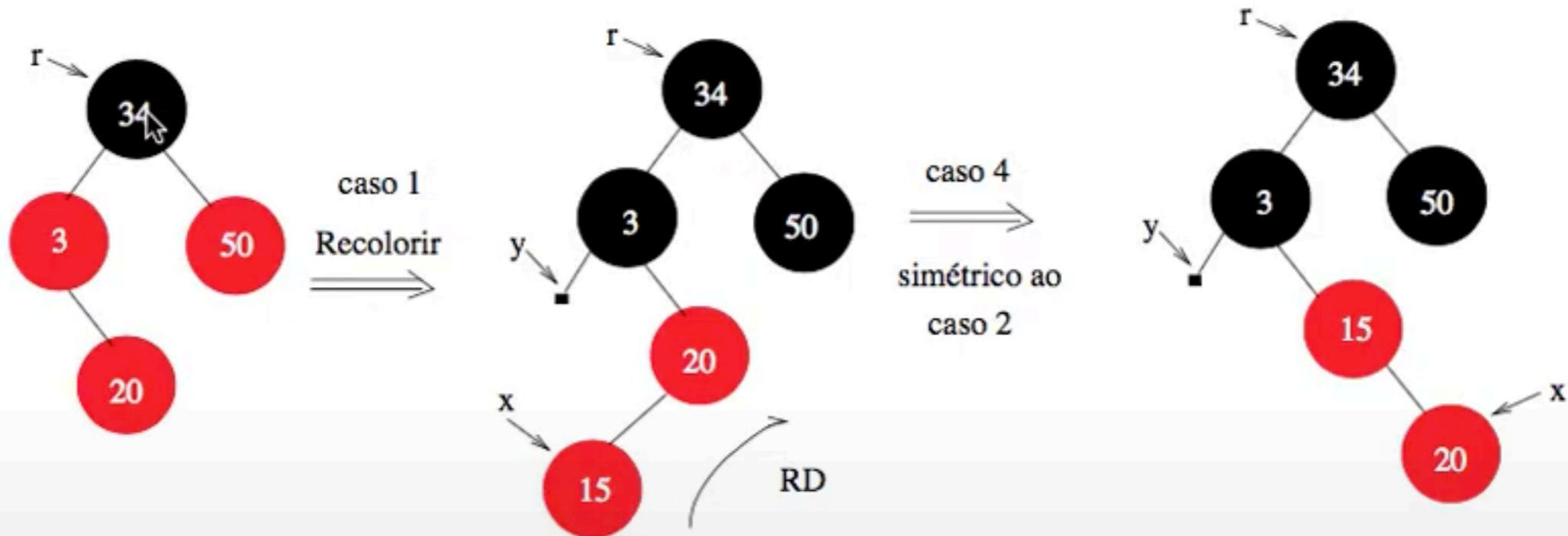
```
noReferenciaX.getPai().setCor(PRETO);
```

```
noReferenciaX.getPai().getPai().setCor(VERMELHO);
```

```
RotacaoEsquerda (noReferenciaX.getPai().getPai())
```

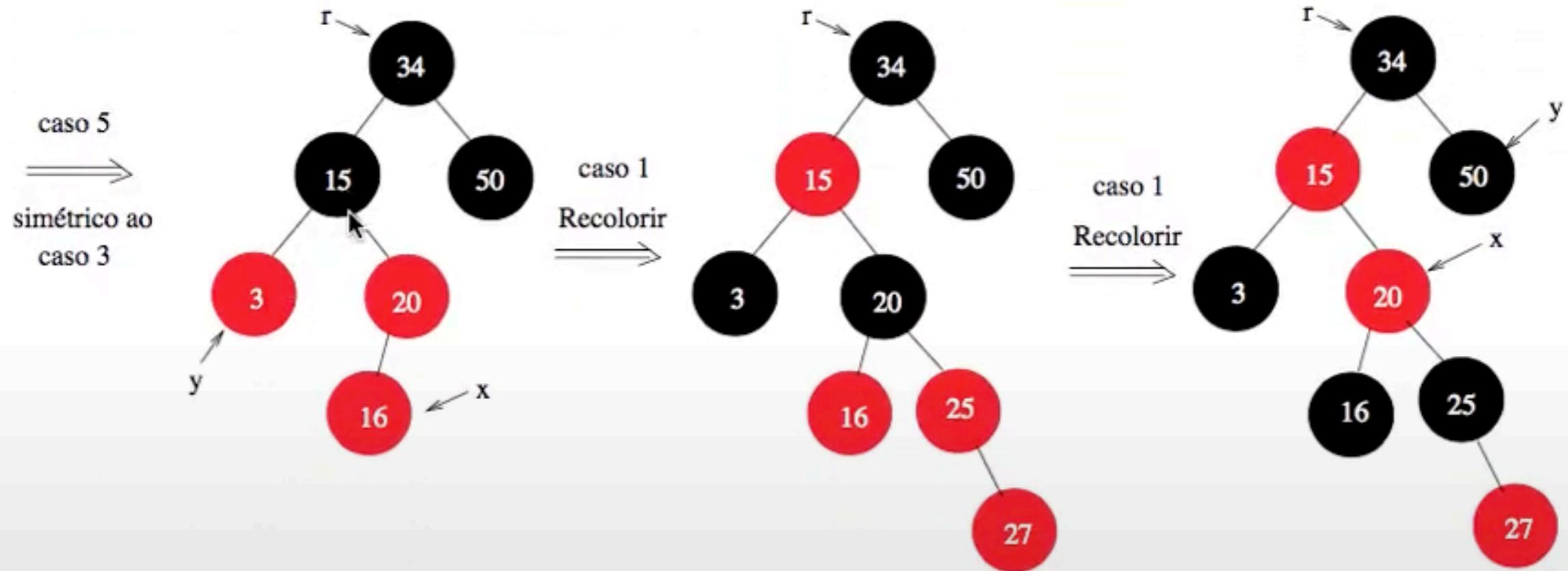
Exemplo de Inclusão

Ordem de inserção: 34, 3, 50, 20, 15, 16, 25, 27



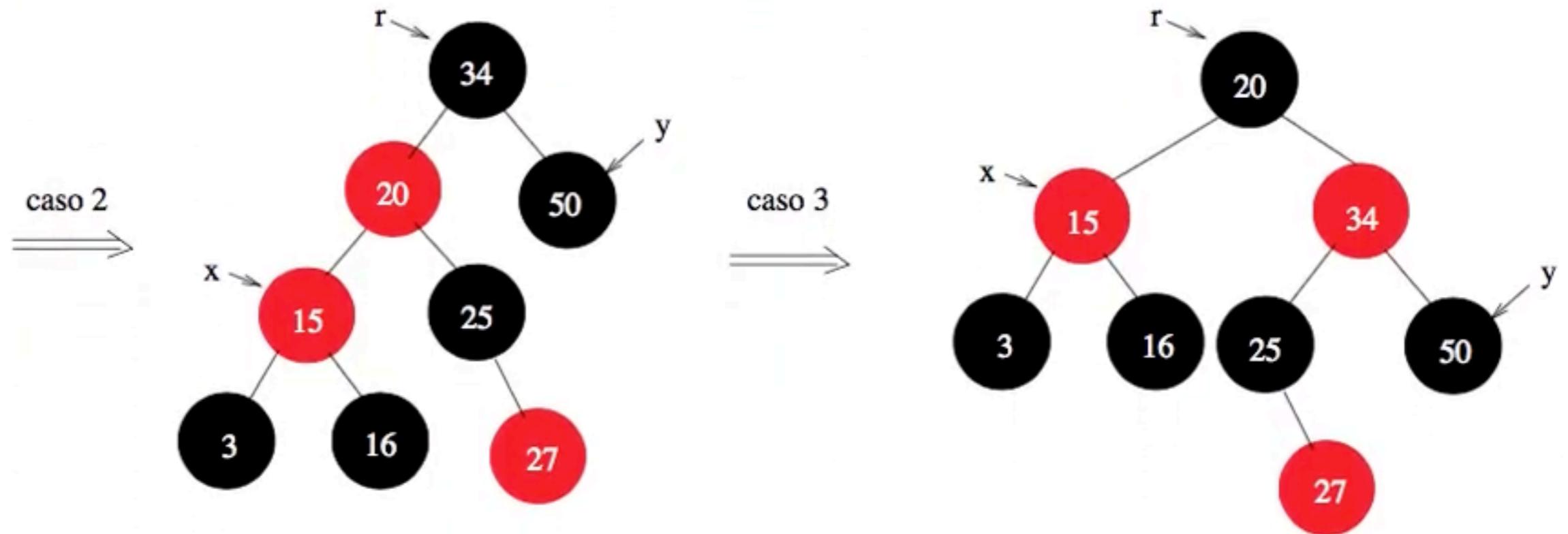
Exemplo de Inclusão

Ordem de inserção: 34, 3, 50, 20, 15, 16, 25, 27



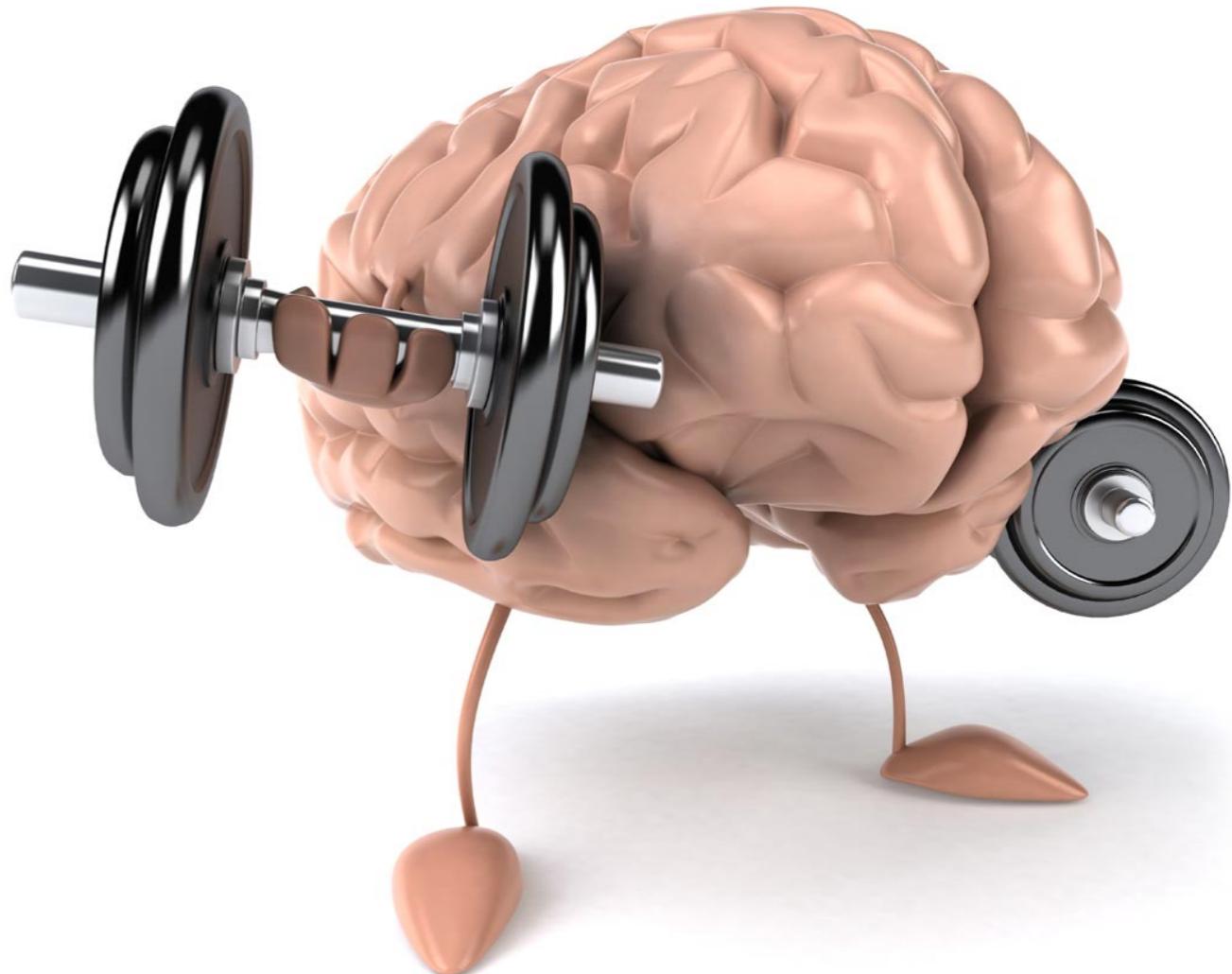
Exemplo de Inclusão

Ordem de inserção: 34, 3, 50, 20, 15, 16, 25, 27



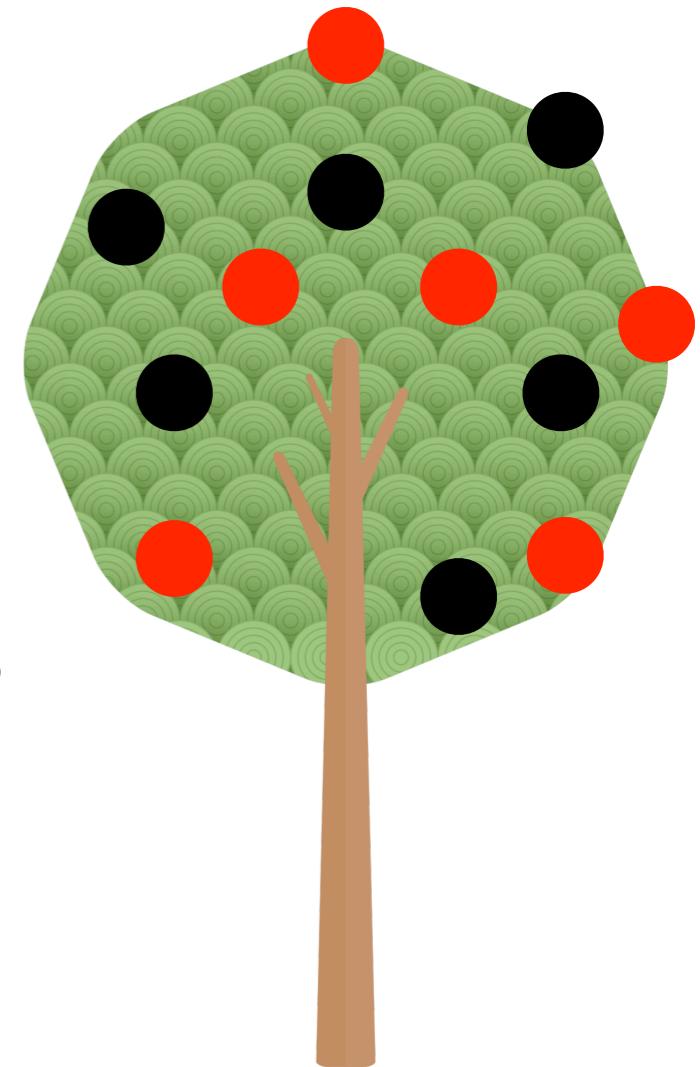
Exercício

- Vamos inserir na ARN a seguinte sequência:
 - 41, 38, 31, 12, 19, 8, 45, 60 e 62.



Operação de Exclusão

- O objetivo da operação de exclusão é similar a inclusão, no sentido de manter a árvore atendendo as cinco propriedades que definem uma ARN.
- Ou seja, depois de remover um nó da árvore, pode ser necessário realizar operações de recoloração e de rotação para garantir que ela ainda seja uma ARN.
- Em suma, a exclusão em uma ARN ocorre do mesmo modo que a ABB, porém, no caso 3 (dois filhos), copiamos o conteúdo do menor valor, e não a cor. Depois, é preciso realizar várias análises na árvore resultante para garantir as propriedade da ARN.



ARN Caída para Esquerda

- Criada por Robert Sedgewick em 2008
- Do inglês, “Left leaning red black tree”
- É uma variante da árvore rubro-negra
- Garante a mesma complexidade de operações, mas possui uma implementação mais simples da inserção e remoção
- A principal diferença é que, em uma ARN Caída para a Esquerda, todos os nós vermelhos estão à esquerda de seus pais. Isso simplifica a implementação em comparação com a ARN tradicional, que tem regras mais complexas para manter o equilíbrio.



AVL vs. ARN



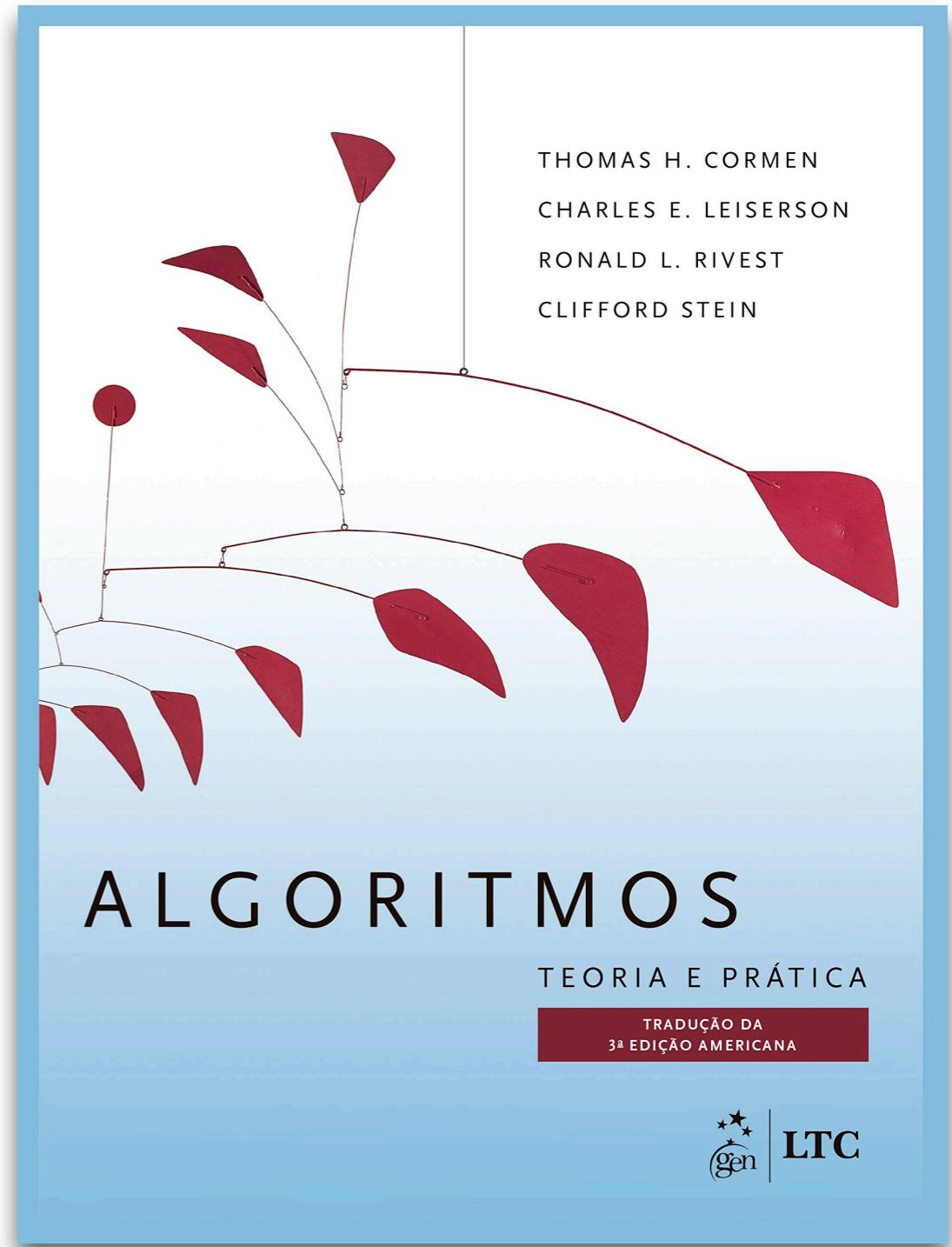
- Na teoria, ambas possuem a mesma complexidade computacional, $O \log(N)$, para inserção, remoção e busca.
- Na prática, a árvore AVL é mais rápida na operação de busca e mais lenta nas operações de inserção e remoção.
- A AVL é “mais balanceada” que a ARN, o que acelera a operação de busca. Porém, a AVL é mais lenta em relação a ARN nas operações de inserção e remoção.
- A ARN é mais comumente usada do que a AVL
 - É usada em diversas aplicações e bibliotecas de linguagem de programação, por exemplo:
 - **Java:** java.util.TreeMap, java.util.TreeSet
 - **C++ STL:** map, multimap, multiset
 - **Linux kernel:** completely fair scheduler, linux/rbtree.h

Simulador Árvores RN

[https://www.inf.ufsc.br/~aldo.vw/
estruturas/simulador/RB.html](https://www.inf.ufsc.br/~aldo.vw/estruturas/simulador/RB.html)



Referência da Aula





Prof. Fernando Sambinelli
sambinelli@ifsp.edu.br



$$j = \frac{p}{4\pi r^2}$$

