

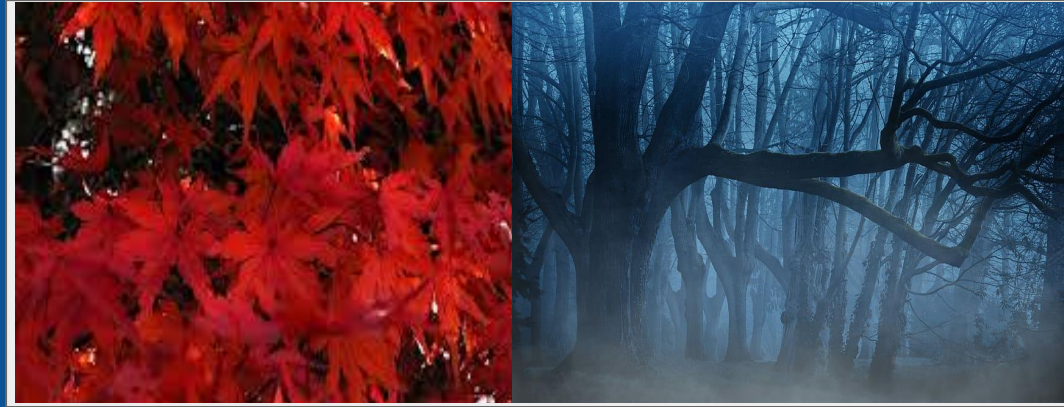
ÁRVORES RUBRO NEGRAS

Prof. Joaquim Uchôa
Profa. Juliana Gregghi
Prof. Renato Ramos



- Visão geral
- Ementa
- Material de Apoio
- Plano de Curso
- Metodologia de ensino
- Recuperação e
Frequência

VISÃO GERAL



ÁRVORES VERMELHAS

Por que as folhas das árvores se tornam avermelhadas antes de caírem no outono, e por que sua cor é mais viva em alguns anos? Uma pesquisa desenvolvida na Universidade de Wisconsin-Madison (EUA) propõe uma explicação simples para o fenômeno: "Os pigmentos vermelhos chamados antocianinas que se acumulam nas folhas funcionam como uma proteção contra a radiação solar intensa", diz William Hoch, coordenador do estudo publicado na revista Tree Physiology.

ÁRVORES VERMELHAS

Os pigmentos vermelhos protegem o tecido que realiza a fotossíntese. Durante o outono, as árvores reabsorvem os nutrientes das folhas; para recolher o máximo de nutrientes antes que as folhas caiam, elas precisam da energia gerada na fotossíntese. Contudo, os sistemas que participam da fotossíntese -- muito utilizados no verão -- também estão sendo decompostos e absorvidos no outono. Além dessa decomposição, a fotossíntese pode ser inibida ainda por uma luminosidade muito intensa.

ÁRVORES VERMELHAS

Por isso, logo que a reabsorção de nutrientes se inicia no outono, a concentração das antocianinas aumenta na superfície das folhas. "Esses pigmentos absorvem grande parte da luz que chega às folhas", afirma Hoch. Dessa forma, preserva-se a limitada habilidade das árvores de produzir energia durante o outono. Além da luz abundante, baixas temperaturas e outros fatores de estresse também provocam o acúmulo das antocianinas nas folhas.

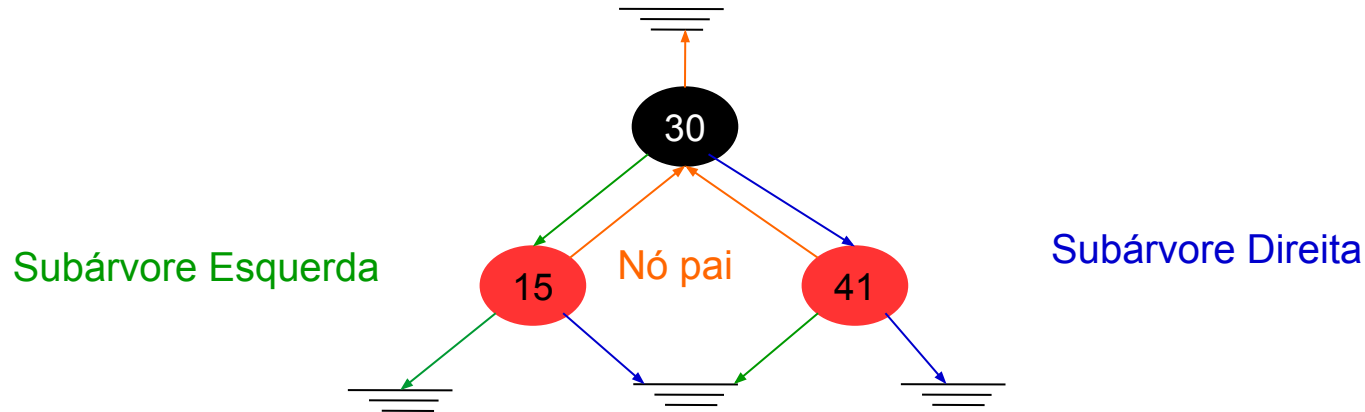
ÁRVORES VERMELHAS

A descoberta da equipe de Hoch confirma as observações de que as cores do outono são mais vivas em dias mais claros e nas folhas situadas na parte mais externa das árvores. "As regiões em que o outono é ensolarado e frio exibem folhas muito vermelhas nessa estação", diz Hoch.

Fonte: Ciência Hoje On-line e
<http://www.jardimdeflores.com.br/CURIOSIDADES/A25folhasvermelhas.htm>

ÁRVORE RUBRO-NEGRA

- Árvore binária de busca, onde cada nó é constituído por:
 - Valor de uma chave;
 - Cor, que pode ser **vermelha** ou **preta**;
 - Ponteiros para subárvores direita e esquerda e para o nó pai (em caso de duplo encadeamento).

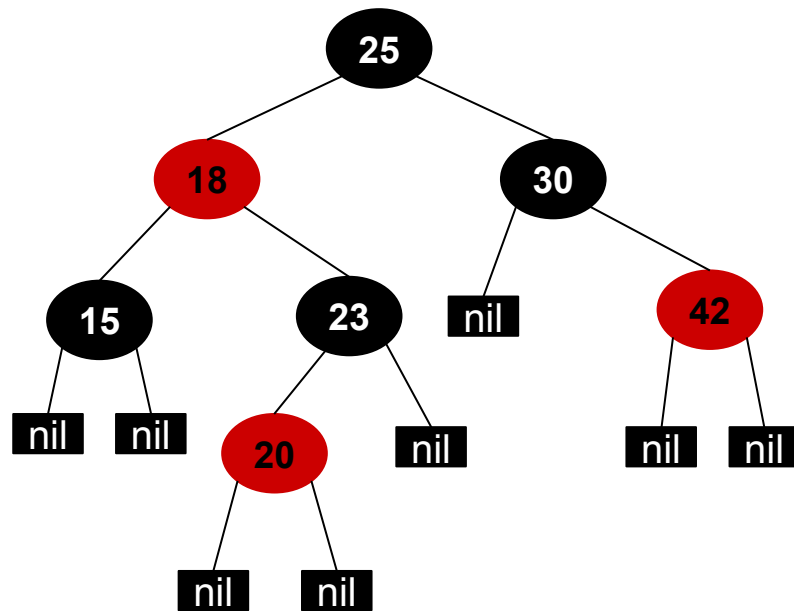


PROPRIEDADES

- Todo nó da árvore é vermelho ou preto;
- A raiz é preta;
- Toda folha (*nil*) é preta;
- Um nó vermelho tem ambos os filhos pretos;
- Para todo nó, todo caminho do nó até as folhas contém o mesmo número de nós pretos.

Todos os nós internos contêm chaves e o nós folhas são *nil*.

EXEMPLO



OBSERVAÇÕES GERAIS PARA IMPLEMENTAÇÃO

A implementação de uma árvore rubro-negra é similar à de uma árvore binária tradicional, com ajustes para balanceamento quando da inserção ou remoção de nós.

Os nós possuem um atributo adicional para armazenamento da cor.

OBSERVAÇÕES GERAIS PARA IMPLEMENTAÇÃO

A maior parte das implementações utiliza um nó sentinela NIL, preto, para o qual todas as folhas apontam. Sem isso, é necessário considerar nós nulos como folha de cor preta.

OPERAÇÃO DE INSERÇÃO EM ÁRVORES RUBRO-NEGRAS



INSERÇÃO

Para garantir o balanceamento, todas as inserções em uma árvore rubro-negra são feitas com nós vermelhos, à exceção da raiz.

Quando o pai do nó inserido é preto, não há problemas e a inserção é feita naturalmente.

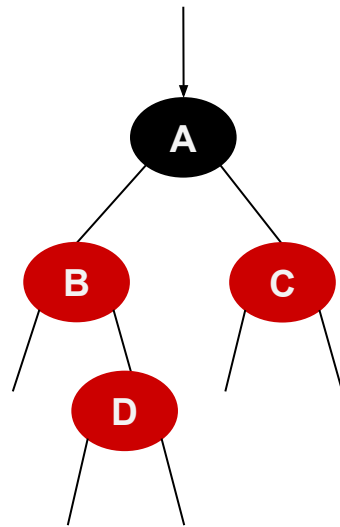
Quando o pai do nó inserido também é vermelho, é necessário investigar o caminho, avaliando-se além do nó pai, o nó tio e o nó avô.

INSERÇÃO (PAI É VERMELHO) - CASO 1

Caso se tenha a seguinte situação:

- ❑ pai vermelho
- ❑ tio vermelho
- ❑ avô preto

Basta alterar as cores do pai,
tio e avô

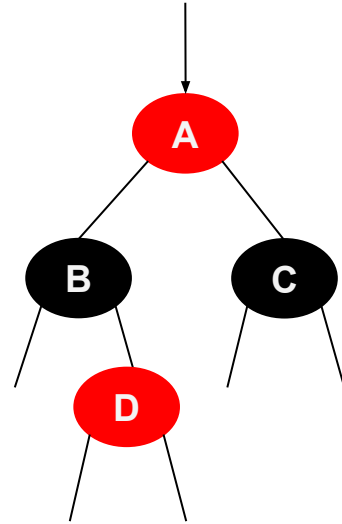


INSERÇÃO (PAI É VERMELHO) - CASO 1

Caso se tenha a seguinte situação:

- ❑ pai vermelho
- ❑ tio vermelho
- ❑ avô preto

Basta alterar as cores do pai,
tio e avô

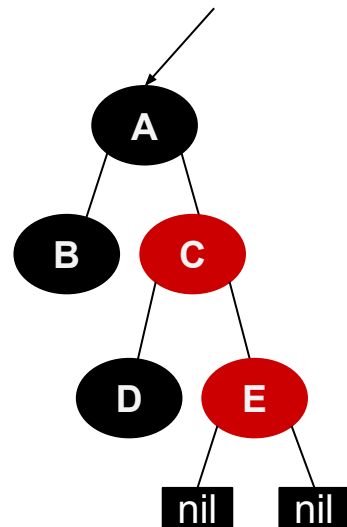


INSERÇÃO (PAI É VERMELHO) - CASO 2

Caso se tenha a seguinte situação:

- ❑ pai vermelho
- ❑ tio preto
- ❑ avô preto
- ❑ nó inserido à direita do pai e pai à direita do avô

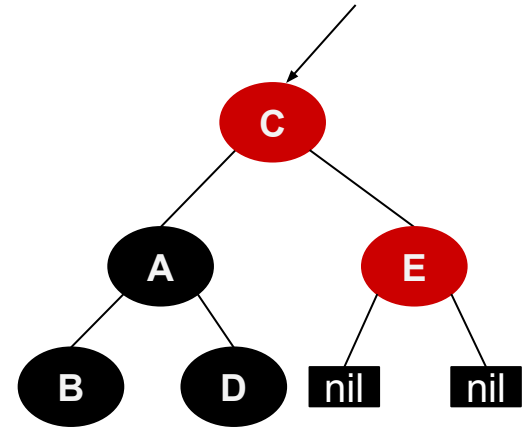
Rotação à esquerda, com pai tornando-se nova raiz da subárvore, seguida de alteração das cores do pai e do avô.



INSERÇÃO (PAI É VERMELHO) - CASO 2

Caso se tenha a seguinte situação:

- ❑ pai vermelho
- ❑ tio preto
- ❑ avô preto
- ❑ nó inserido à direita do pai e pai à direita do avô

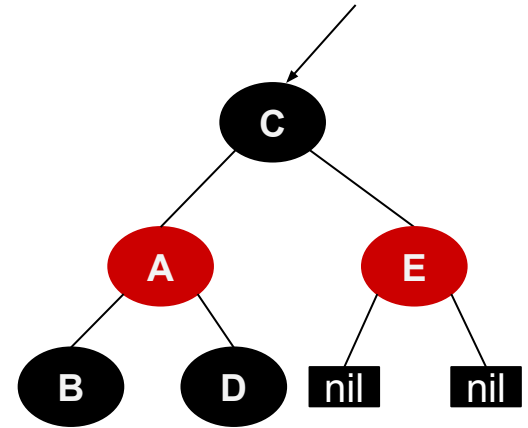


Rotação à esquerda, com pai tornando-se nova raiz da subárvore, seguida de alteração das cores do pai e do avô.

INSERÇÃO (PAI É VERMELHO) - CASO 2

Caso se tenha a seguinte situação:

- ❑ pai vermelho
- ❑ tio preto
- ❑ avô preto
- ❑ nó inserido à direita do pai e pai à direita do avô



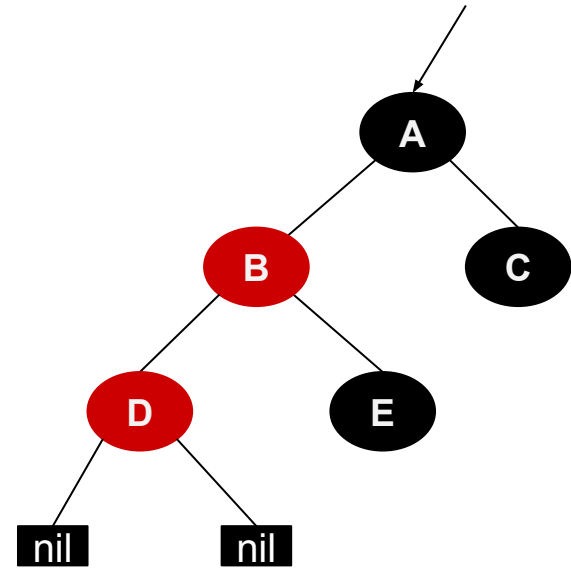
Rotação à esquerda, com pai tornando-se nova raiz da subárvore, seguida de alteração das cores do pai e do avô.

INSERÇÃO (PAI É VERMELHO) - CASO 3

Caso se tenha a seguinte situação:

- ❑ pai vermelho
- ❑ tio preto
- ❑ avô preto
- ❑ nó inserido à esquerda do pai e pai à esquerda do avô

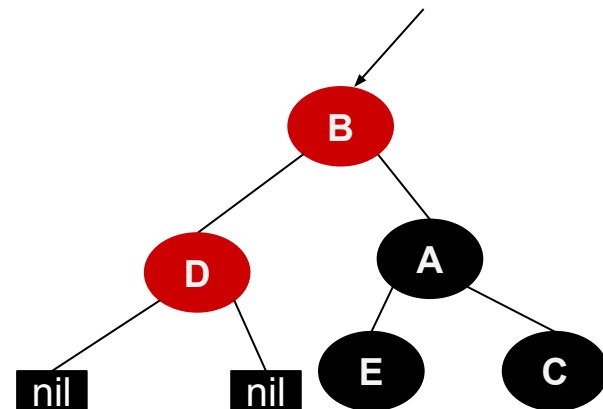
Rotação à direita, com pai
Tornando-se nova raiz da subárvore,
seguida de alteração das cores do pai e do avô.



INSERÇÃO (PAI É VERMELHO) - CASO 3

Caso se tenha a seguinte situação:

- ❑ pai vermelho
- ❑ tio preto
- ❑ avô preto
- ❑ nó inserido à esquerda do pai e pai à esquerda do avô



Rotação à direita, com pai

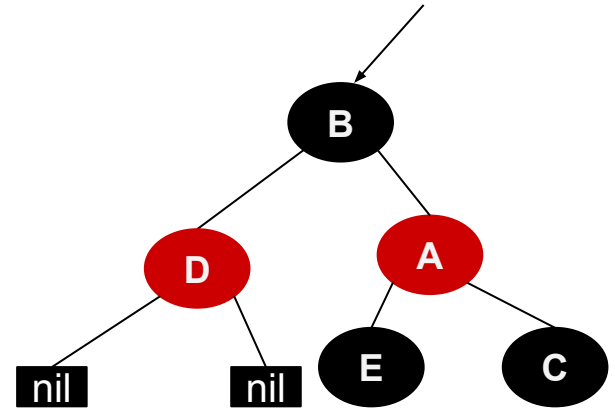
Tornando-se nova raiz da subárvore,

seguida de alteração das cores do pai e do avô.

INSERÇÃO (PAI É VERMELHO) - CASO 3

Caso se tenha a seguinte situação:

- ❑ pai vermelho
- ❑ tio preto
- ❑ avô preto
- ❑ nó inserido à esquerda do pai e pai à esquerda do avô



Rotação à direita, com pai

Tornando-se nova raiz da subárvore,

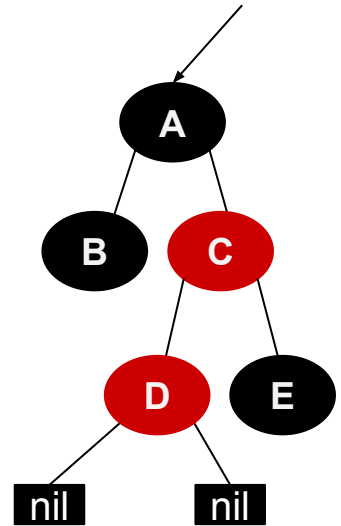
seguida de alteração das cores do pai e do avô.

INSERÇÃO (PAI É VERMELHO) - CASO 4

Caso se tenha a seguinte situação:

- ❑ pai vermelho
- ❑ tio preto
- ❑ avô preto
- ❑ nó inserido à esquerda do pai e pai à direita do avô

Rotação dupla, a primeira à direita, segunda à esquerda, seguida de alteração das cores do nó e do avô.

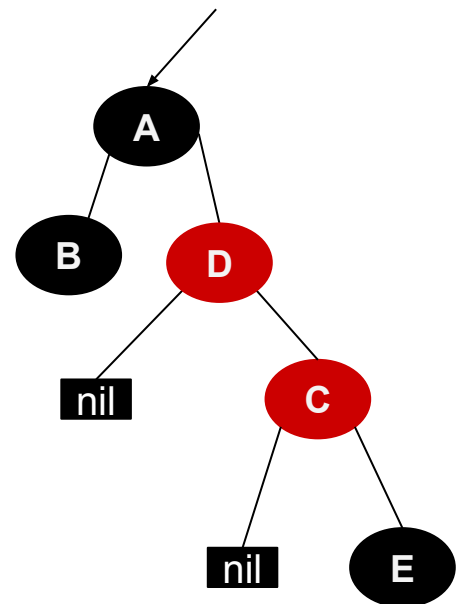


INSERÇÃO (PAI É VERMELHO) - CASO 4

Caso se tenha a seguinte situação:

- ❑ pai vermelho
- ❑ tio preto
- ❑ avô preto
- ❑ nó inserido à esquerda do pai e pai à direita do avô

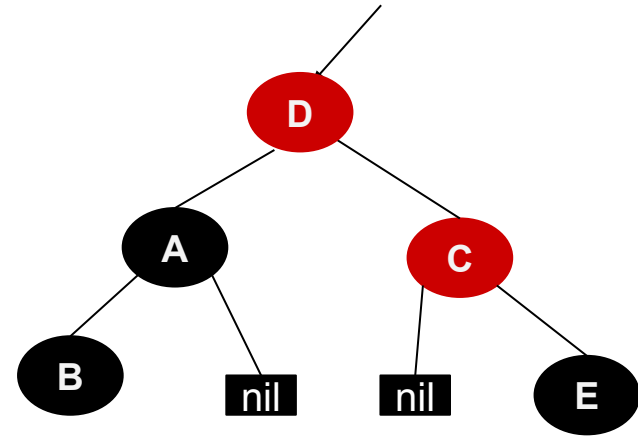
Rotação dupla, a primeira à direita, segunda à esquerda, seguida de alteração das cores do nó e do avô.



INSERÇÃO (PAI É VERMELHO) - CASO 4

Caso se tenha a seguinte situação:

- ❑ pai vermelho
- ❑ tio preto
- ❑ avô preto
- ❑ nó inserido à esquerda do pai e pai à direita do avô



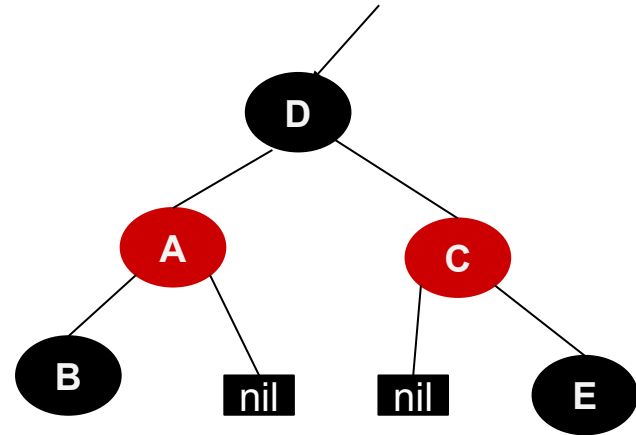
Rotação dupla, a primeira à direita, segunda à esquerda, seguida de alteração das cores do nó e do avô.

INSERÇÃO (PAI É VERMELHO) - CASO 4

Caso se tenha a seguinte situação:

- ❑ pai vermelho
- ❑ tio preto
- ❑ avô preto
- ❑ nó inserido à esquerda do pai e pai à direita do avô

Rotação dupla, a primeira à direita, segunda à esquerda, seguida de alteração das cores do nó e do avô.

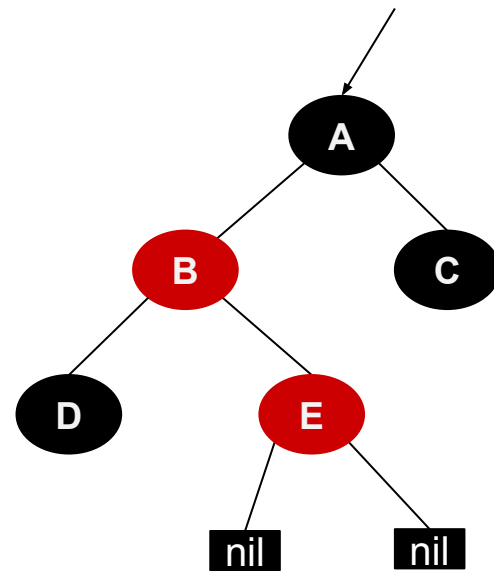


INSERÇÃO (PAI É VERMELHO) - CASO 5

Caso se tenha a seguinte situação:

- ❑ pai vermelho
- ❑ tio preto
- ❑ avô preto
- ❑ nó inserido à direita do pai e pai à esquerda do avô

Rotação dupla, a primeira à esquerda, segunda à direita, seguida de alteração das cores do nó e do avô.

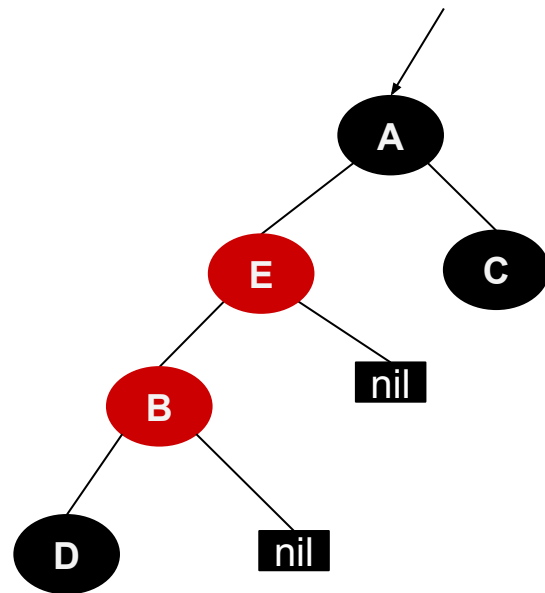


INSERÇÃO (PAI É VERMELHO) - CASO 5

Caso se tenha a seguinte situação:

- ❑ pai vermelho
- ❑ tio preto
- ❑ avô preto
- ❑ nó inserido à direita do pai e pai à esquerda do avô

Rotação dupla, a primeira à esquerda, segunda à direita, seguida de alteração das cores do nó e do avô.

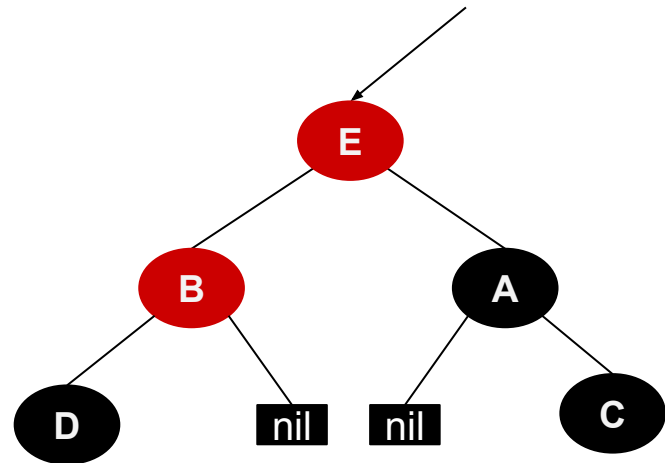


INSERÇÃO (PAI É VERMELHO) - CASO 5

Caso se tenha a seguinte situação:

- ☐ pai vermelho
- ☐ tio preto
- ☐ avô preto
- ☐ nó inserido à direita do pai e pai à esquerda do avô

Rotação dupla, a primeira à esquerda, segunda à direita, seguida de alteração das cores do nó e do avô.

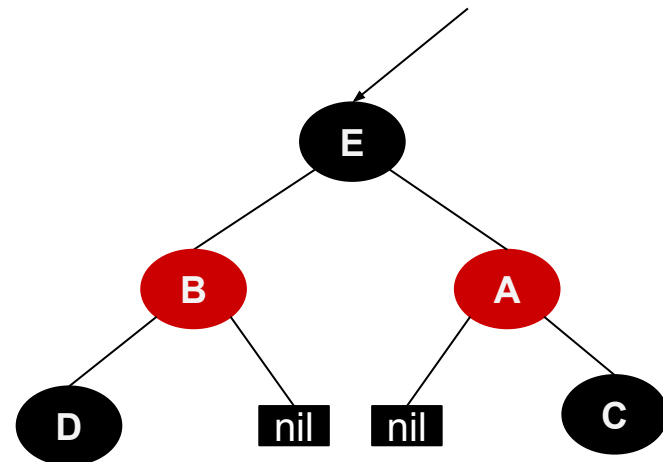


INSERÇÃO (PAI É VERMELHO) - CASO 5

Caso se tenha a seguinte situação:

- ☐ pai vermelho
- ☐ tio preto
- ☐ avô preto
- ☐ nó inserido à direita do pai e pai à esquerda do avô

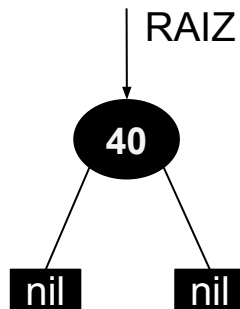
Rotação dupla, a primeira à esquerda, segunda à direita, seguida de alteração das cores do nó e do avô.



EXEMPLO - INSERÇÃO - 1/27

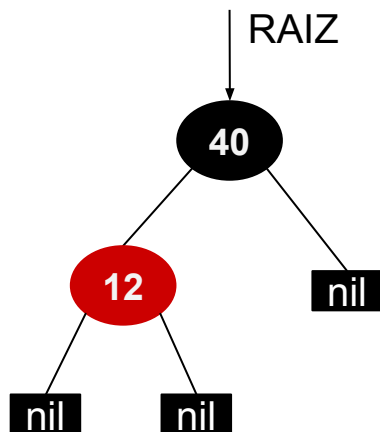
Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65

- Todo nó inserido deve ser vermelho, **exceto** a raiz.



EXEMPLO - INSERÇÃO - 2/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso base

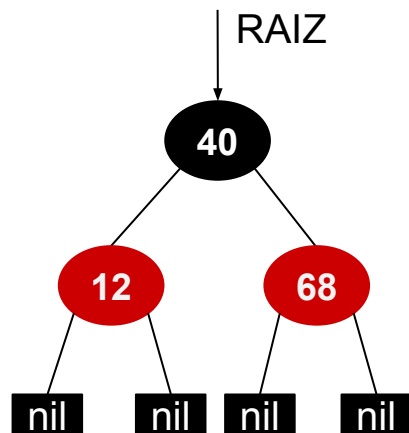
- o pai é preto

- * Inserir nó vermelho

- O novo nó possui dois filhos *nil*;

EXEMPLO - INSERÇÃO - 3/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65

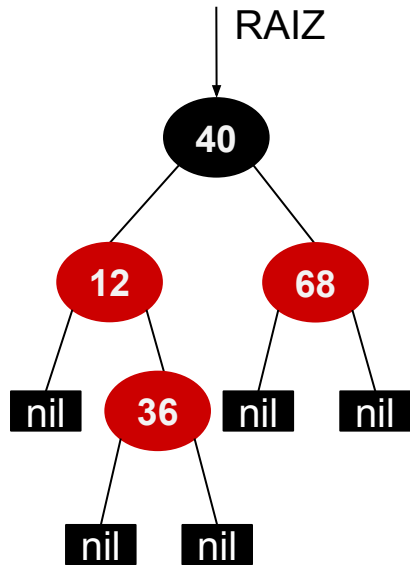


Operação

- Encontra-se a posição correta;
- Substitui-se o nó *nil* pelo novo nó;
- O novo nó possui dois filhos *nil*;

EXEMPLO - INSERÇÃO - 4/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso 1

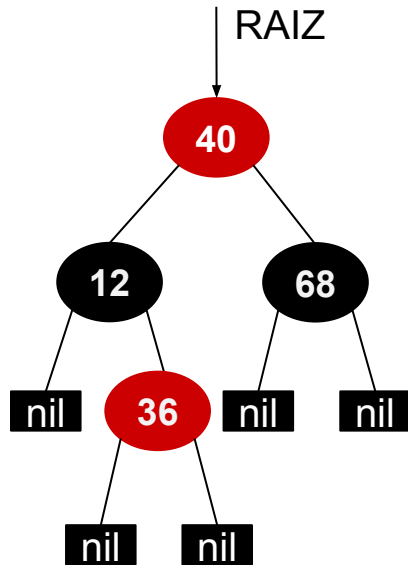
* o pai é vermelho

* o avô é preto

* tio é vermelho

EXEMPLO - INSERÇÃO - 5/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso 1

* o pai é vermelho

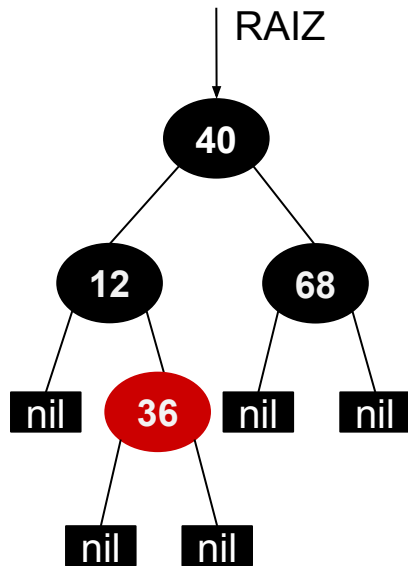
* o avô é preto

* tio é vermelho

→ **Alterar as cores do pai, tio e avô**

EXEMPLO - INSERÇÃO - 6/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso 1

* o pai é vermelho

* o avô é preto

* tio é vermelho

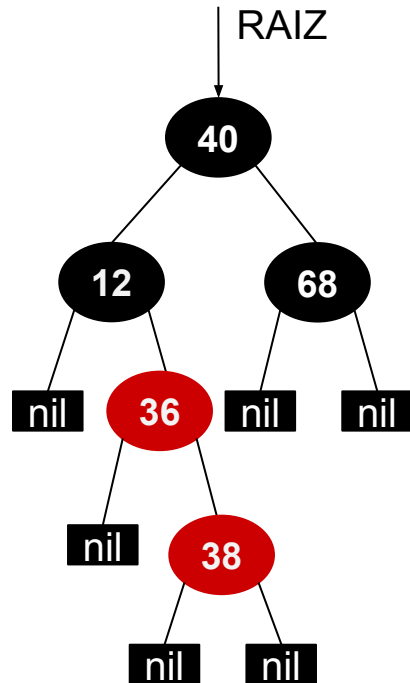
→ **Alterar as cores do pai, tio e avô**

EXCEÇÃO: avô é raiz!

→ **alterar cor do avô**

EXEMPLO - INSERÇÃO - 7/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso 2

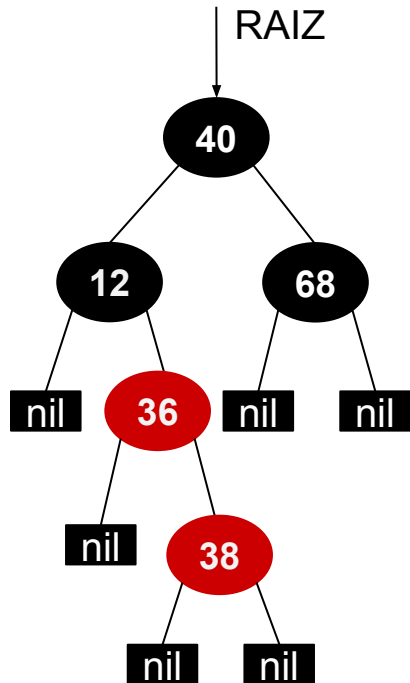
* o pai é vermelho

* o avô é preto

* tio é preto

EXEMPLO - INSERÇÃO - 8/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso 2

* o pai é vermelho

* o avô é preto

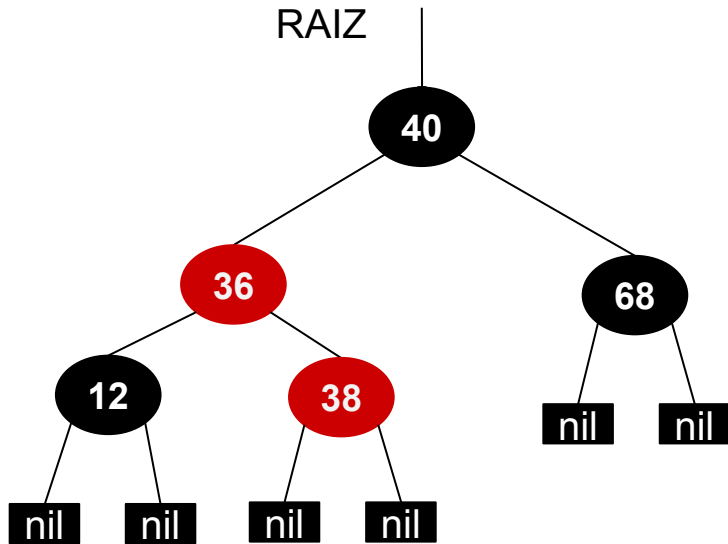
* tio é preto

→ **Rotação à esquerda**

EXEMPLO - INSERÇÃO - 9/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65

RAIZ



Caso 2

* o pai é vermelho

* o avô é preto

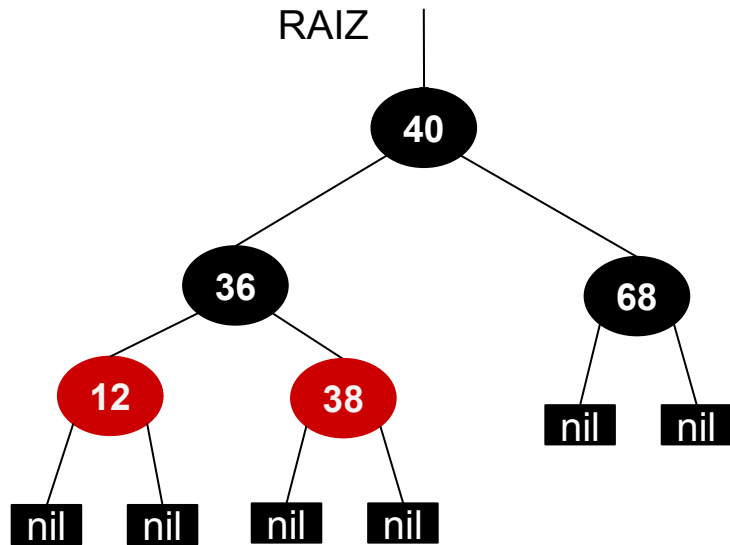
* tio é preto

→ **Rotação à esquerda**

EXEMPLO - INSERÇÃO - 10/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65

RAIZ



Caso 2

* o pai é vermelho

* o avô é preto

* tio é preto

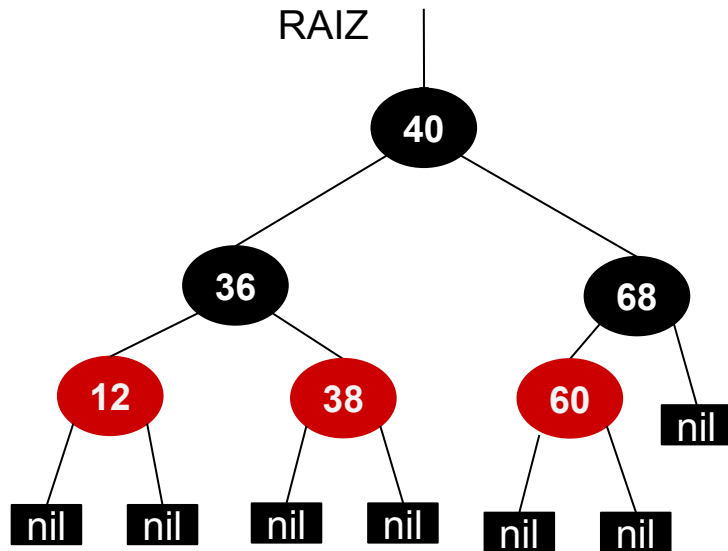
→ Rotação à esquerda

→ Trocar a cor do pai e avô

EXEMPLO - INSERÇÃO - 11/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65

RAIZ

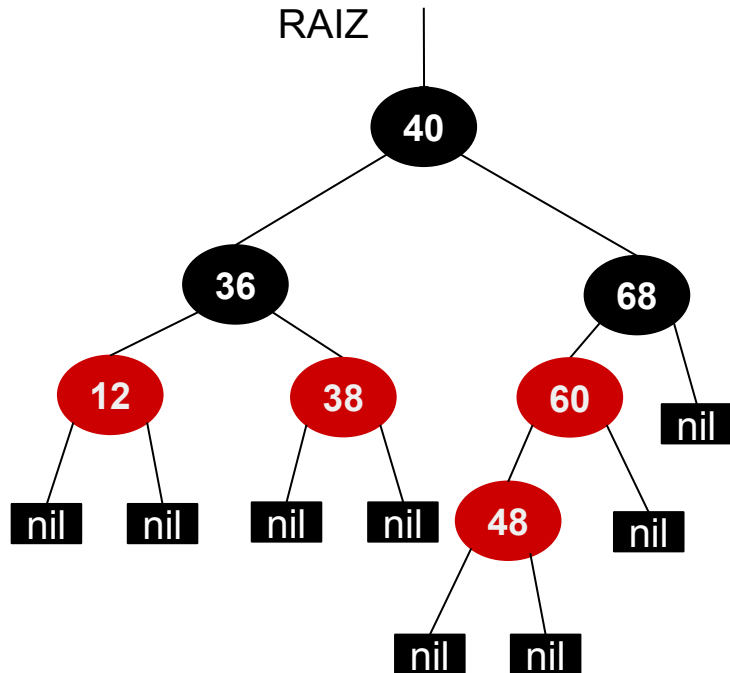


Inserção simples

EXEMPLO - INSERÇÃO - 12/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65

RAIZ



Caso 3

* o pai é vermelho

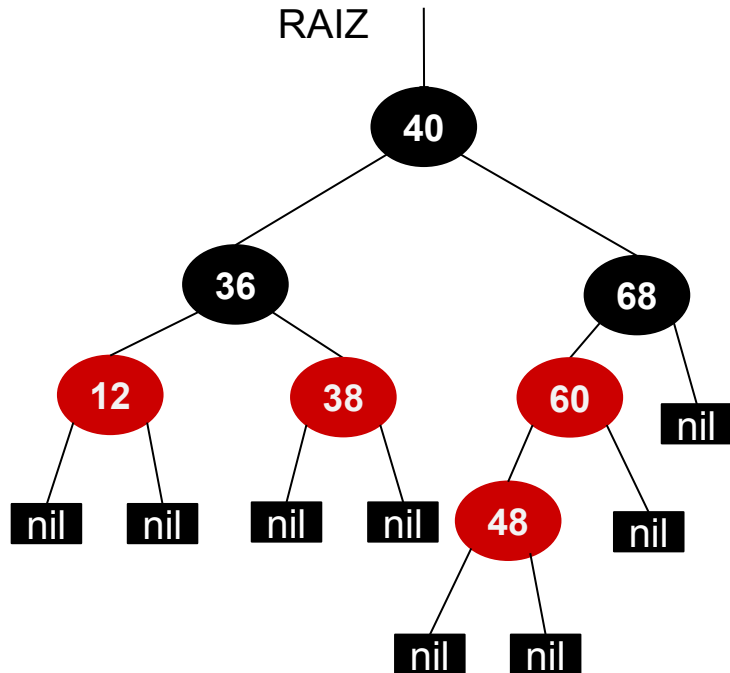
* o avô é preto

* tio é preto

EXEMPLO - INSERÇÃO - 13/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65

RAIZ



Caso 3

* o pai é vermelho

* o avô é preto

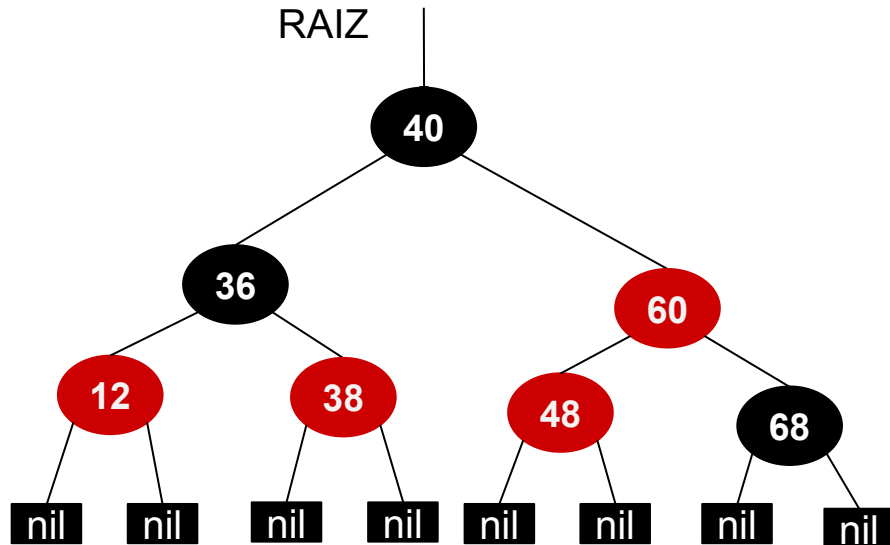
* tio é preto

→ Rotação à direita

EXEMPLO - INSERÇÃO - 14/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65

RAIZ



Caso 3

* o pai é vermelho

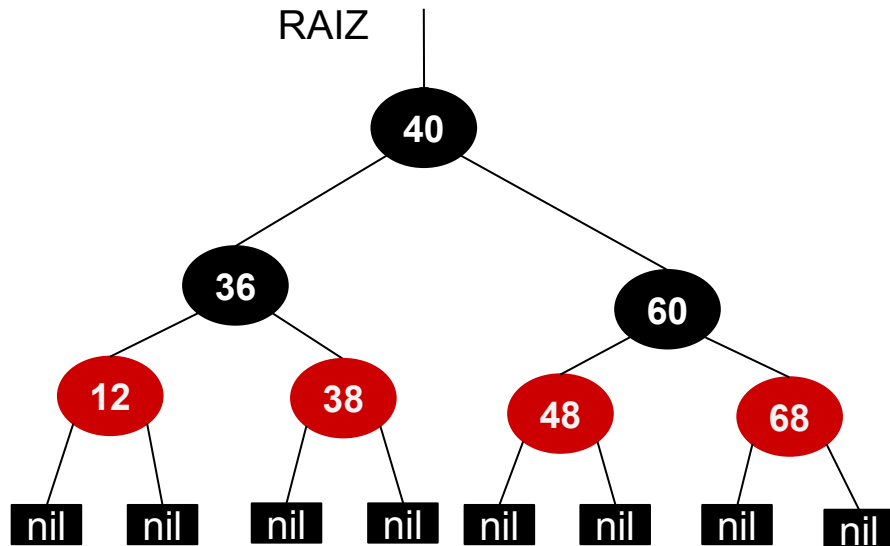
* o avô é preto

* tio é preto

→ Rotação à direita

EXEMPLO - INSERÇÃO - 15/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso 3

* o pai é vermelho

* o avô é preto

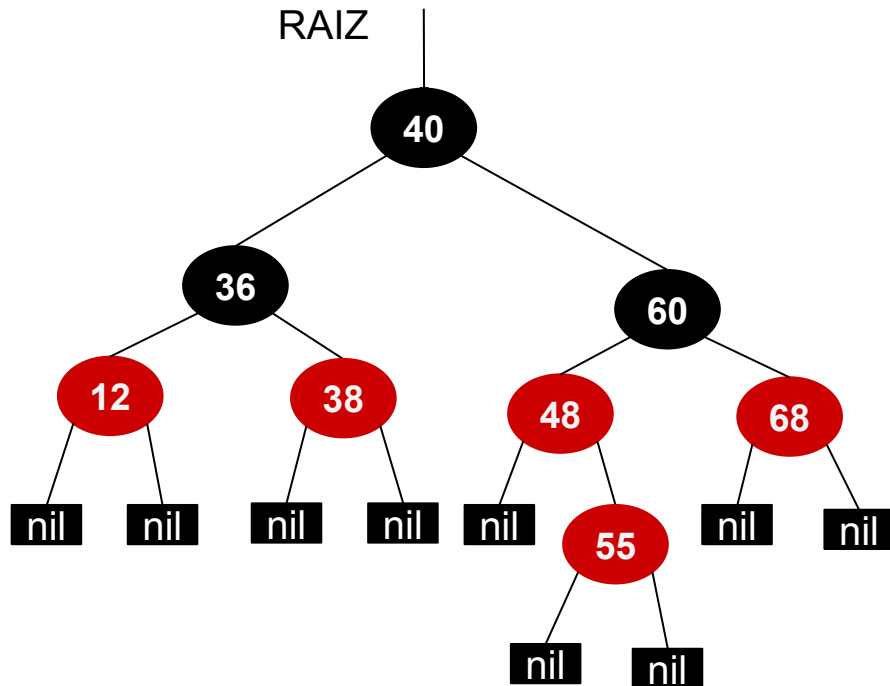
* tio é preto

→ Rotação à direita

→ Trocar a cor do pai e do avô

EXEMPLO - INSERÇÃO - 16/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso 1

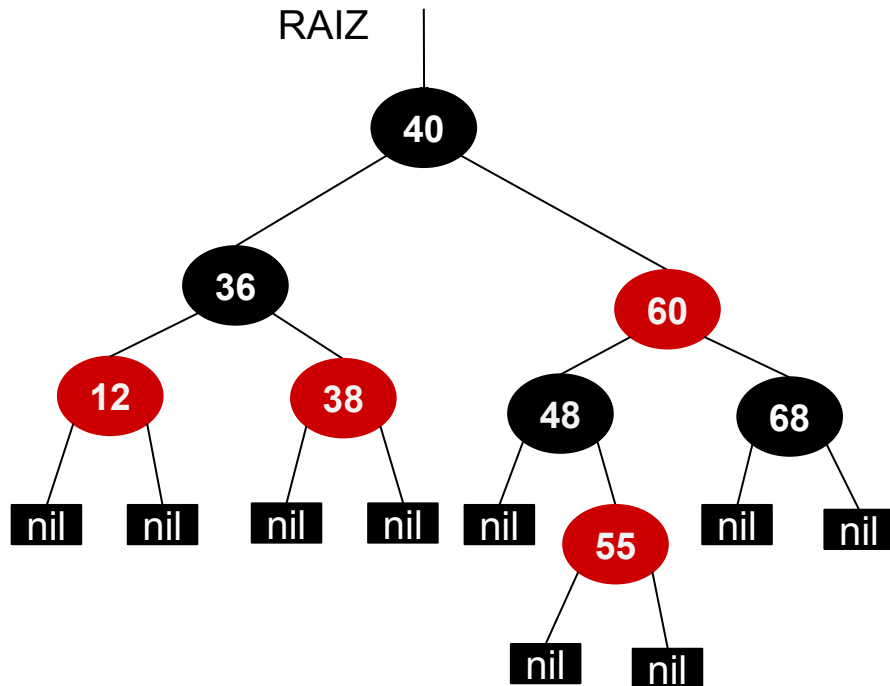
* o pai é vermelho

* o avô é preto

* tio é vermelho

EXEMPLO - INSERÇÃO - 17/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso 1

* o pai é vermelho

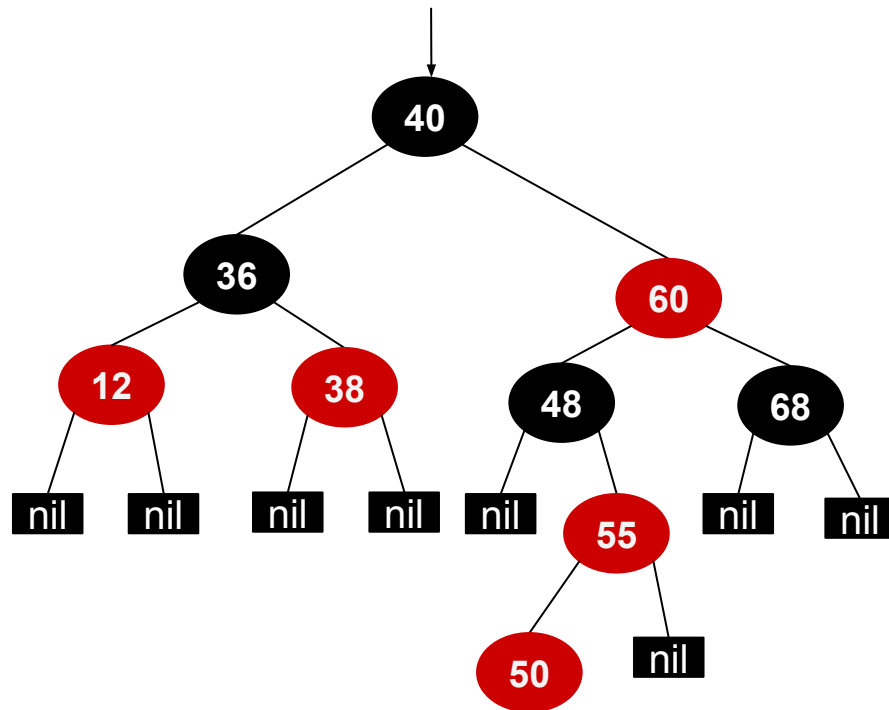
* o avô é preto

* tio é vermelho

→ **Alterar as cores do pai, tio e avô**

EXEMPLO - INSERÇÃO - 18/27

Sequência de inserção: 40 12 68 36 38 60 48 55 **50** 62 65



Caso 4

* o pai é vermelho

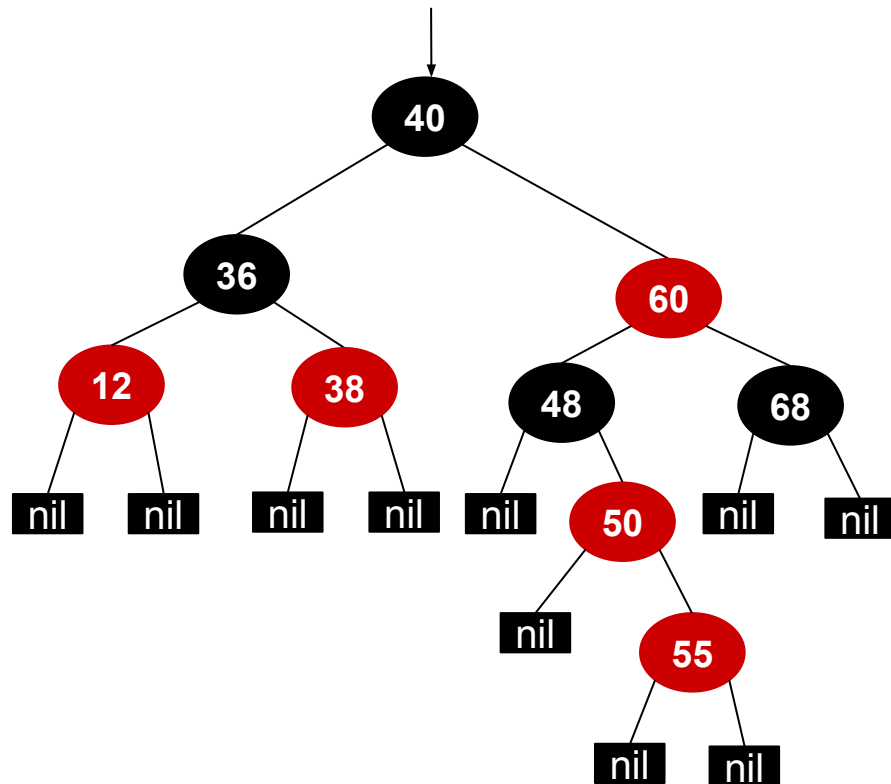
* o avô é preto

* tio é preto

→ Rotação dupla

EXEMPLO - INSERÇÃO - 19/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso 4

* o pai é vermelho

* o avô é preto

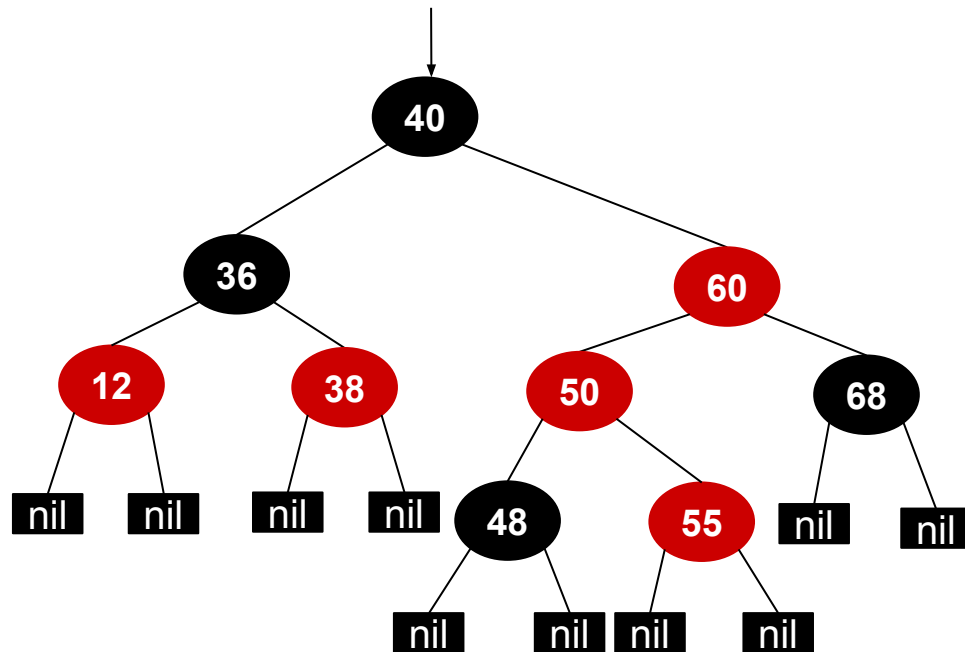
* tio é preto

→ Rotação dupla

* rotação à direita

EXEMPLO - INSERÇÃO - 20/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso 4

* o pai é vermelho

* o avô é preto

* tio é preto

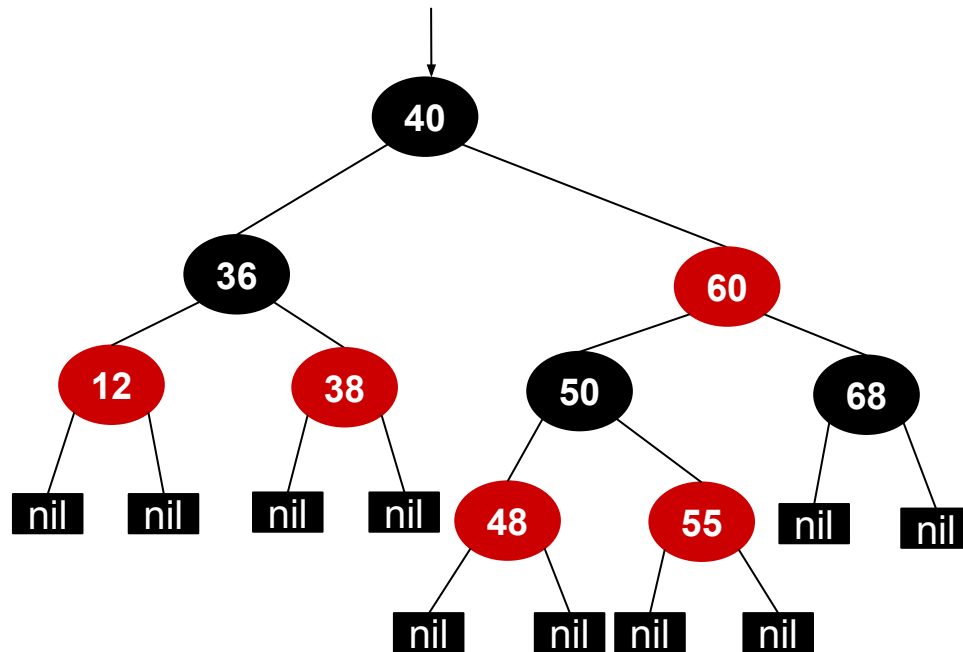
→ Rotação dupla

* rotação à direita

* rotação à esquerda

EXEMPLO - INSERÇÃO - 21/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso 4

* o pai é vermelho

* o avô é preto

* tio é preto

→ Rotação dupla

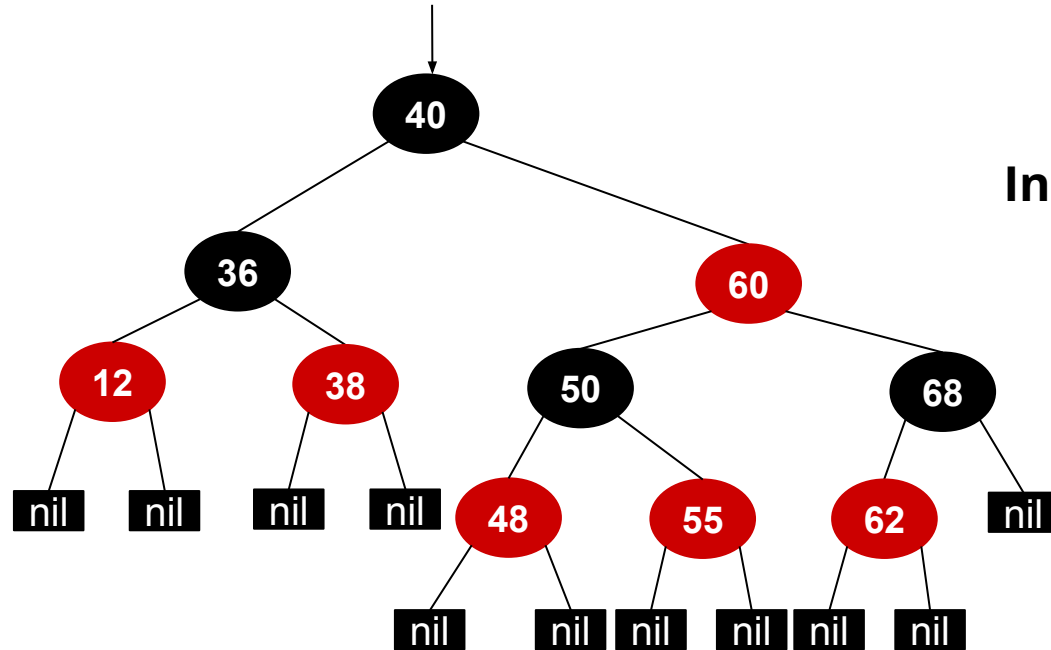
* rotação à direita

* rotação à esquerda

* trocar as cores

EXEMPLO - INSERÇÃO - 22/27

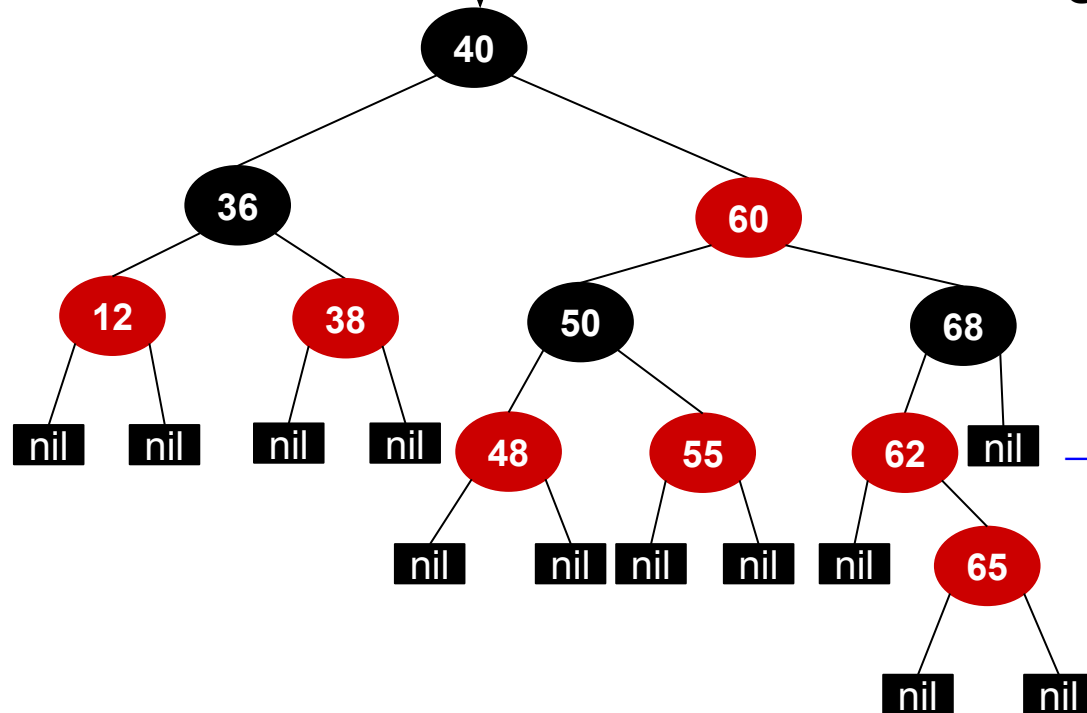
Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Inserção simples

EXEMPLO - INSERÇÃO - 23/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso 5

* o pai é vermelho

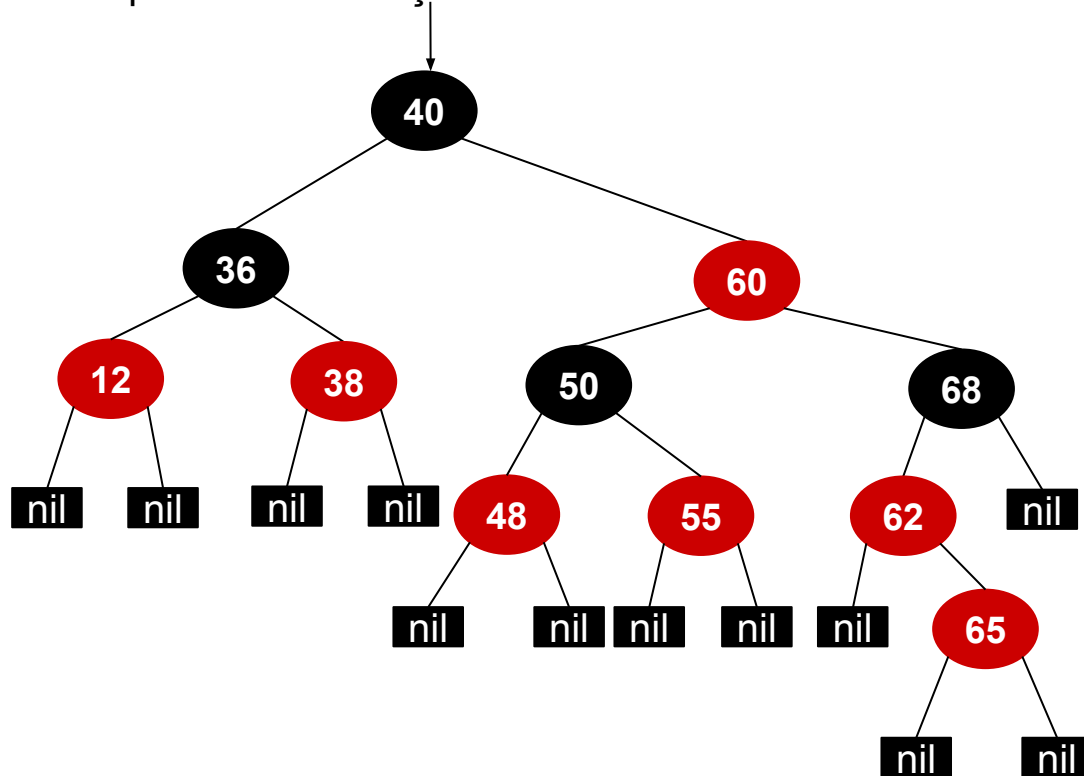
* o avô é preto

* tio é preto

→ Rotação dupla

EXEMPLO - INSERÇÃO - 24/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso 5

* o pai é vermelho

* o avô é preto

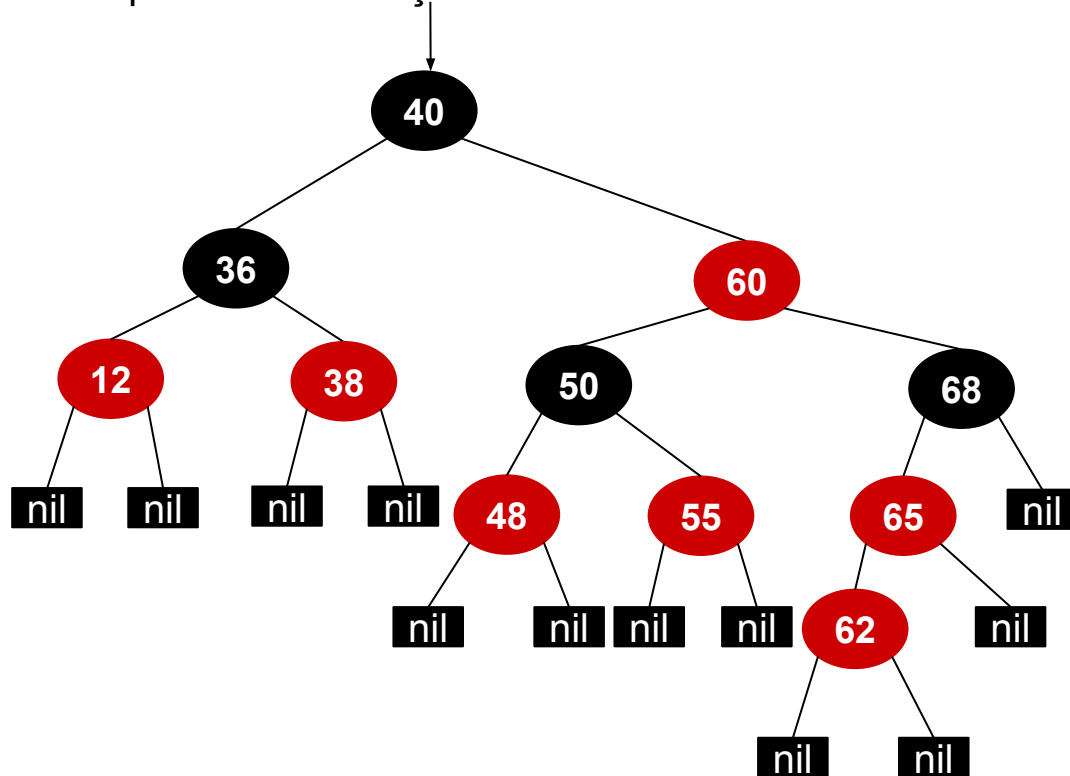
* tio é preto

→ Rotação dupla

* rotação à esquerda

EXEMPLO - INSERÇÃO - 25/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65



Caso 5

* o pai é vermelho

* o avô é preto

* tio é preto

→ Rotação dupla

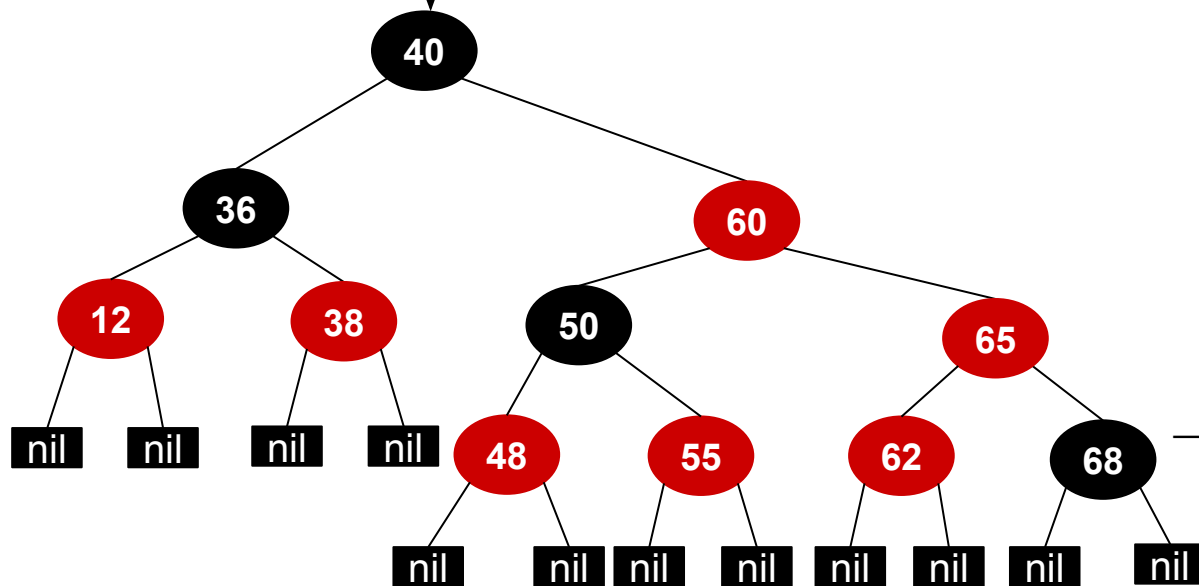
* rotação à esquerda

* rotação à direita

EXEMPLO - INSERÇÃO - 26/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65

Caso 5



* o pai é vermelho

* o avô é preto

* tio é preto

→ **Rotação dupla**

* rotação à esquerda

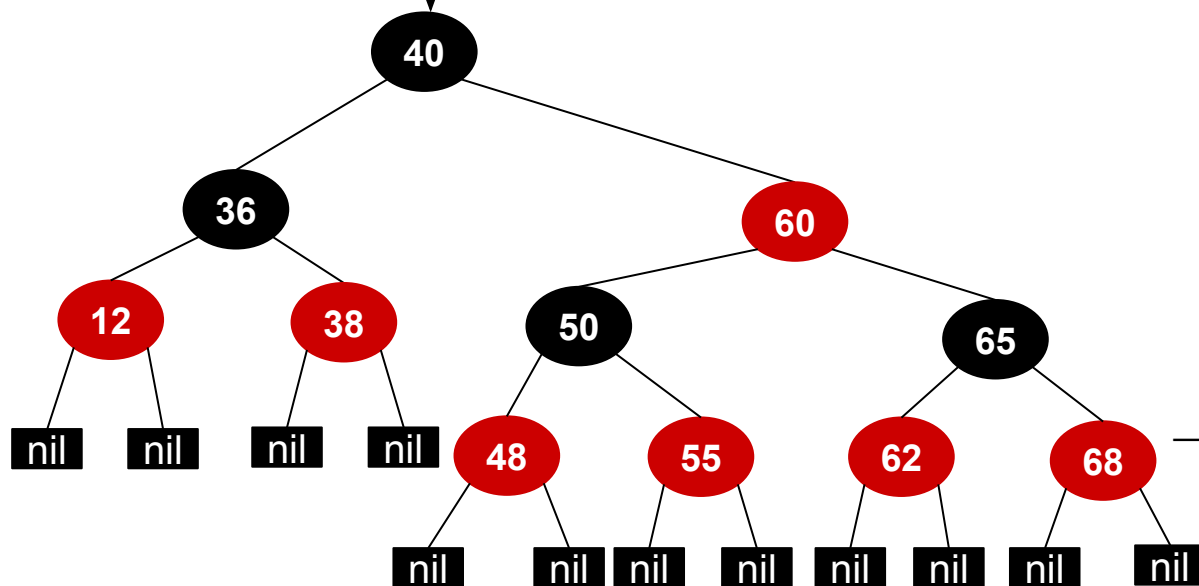
* rotação à direita

* trocar as cores

EXEMPLO - INSERÇÃO - 27/27

Sequência de inserção: 40 12 68 36 38 60 48 55 50 62 65

Caso 5



* o pai é vermelho

* o avô é preto

* tio é preto

→ Rotação dupla

* rotação à esquerda

* rotação à direita

* trocar as cores

IMPLEMENTAÇÃO DA INSERÇÃO EM UMA RB



ÁRVORE RN - INSERÇÃO ITERATIVA - PSEUDOCÓDIGO - 1

inserirIterativamente(umValor):

novo ← criar_noh(umValor); // cria um nó com o valor

se (raiz = NIL) {

 raiz ← novo;

} senão {

 atual ← raiz;

// percorre a árvore para encontrar o ponto de inserção

// anterior irá marcar o ponto de inserção

enquanto (atual ≠ NIL) {

ÁRVORE RN - INSERÇÃO ITERATIVA - PSEUDOCÓDIGO - 2

```
// percorre a árvore para encontrar o ponto de inserção
// anterior irá marcar o ponto de inserção
enquanto (atual  $\neq$  NIL) {
    anterior  $\leftarrow$  atual;
    // use  $\geq$  para permitir valores repetidos
    se (atual.valor > umValor) {
        atual  $\leftarrow$  atual.esquerdo; // segue pelo filho esquerdo
    } senão {
        atual  $\leftarrow$  atual.direito; // segue pelo filho direito
    }
}
```

ÁRVORE RN - INSERÇÃO ITERATIVA - PSEUDOCÓDIGO - 3

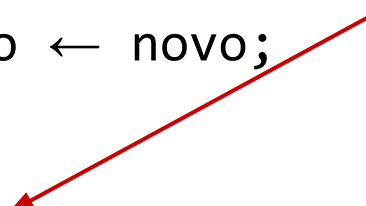
```
// encontrou o ponto, agora é inserir
novo.pai ← anterior;
se (anterior.valor > novo.valor) {
    anterior.esquerdo ← novo;
} senão {
    anterior.direito ← novo;
}
}
```

```
arrumaInsercaoRN(umNoh);
```

ÁRVORE RN - INSERÇÃO ITERATIVA - PSEUDOCÓDIGO - 3

```
// encontrou o ponto, agora é inserir
novo.pai ← anterior;
se (anterior.valor > novo.valor) {
    anterior.esquerdo ← novo;
} senão {
    anterior.direito ← novo;
}
}
```

*Alteração de código da ABB para
suporte ao balanceamento
Vermelho e Preto*



```
arrumaInsercaoRN(novo);
```

ÁRVORE RN-ARRUMAR INSERÇÃO- PSEUDOCÓDIGO - I

arrumarInserçãoRN(*umNoh*):

```
// percorre a árvore do nó até a raiz,  
// ou ter arrumado o balanceamento  
enquanto ((umNoh ≠ raiz) e (umNoh.pai.cor = VERMELHO)) {  
    //encontrando o tio  
    se (umNoh.pai = raiz) {  
        tio ← NIL;  
    } senão se { (umNoh.pai = umNoh.pai.pai.esq) }  
        tio ← umNoh.pai.pai.dir;  
    } senão {  
        tio ← umNoh.pai.pai.esq;  
    }
```

ÁRVORE RN-ARRUMAR INSERÇÃO- PSEUDOCÓDIGO - II

```
se (tio.cor = VERMELHO) { // CASO 1
    tio.cor ← PRETO;
    umNoh.pai.cor ← PRETO;
    umNoh.pai.pai.cor ← VERMELHO;
    // Atualiza umNoh com o avô para continuar o teste
    umNoh ← umNoh.pai.pai;
} senão { // tio é preto, pai é vermelho, precisa de rotação
    // para saber o tipo de rotação precisa verificar
    // se o nó está à direita ou esquerda do pai
    // e se o pai está à direita ou esquerda do avô
    se ((umNoh.pai = umNoh.pai.pai.dir) e ...
```


ÁRVORE RN-ARRUMAR INSERÇÃO - PSEUDOCÓDIGO - III

```
se ((umNoh.pai = umNoh.pai.pai.dir) e
(umNoh = umNoh.pai.dir)) { // CASO 2
    umNoh.pai.cor ← PRETO;
    umNoh.pai.pai.cor ← VERMELHO;
    rotacaoEsquerda(umNoh.pai.pai);
} senão se ((umNoh.pai = umNoh.pai.pai.esquerdo) e
(umNoh = umNoh.pai.esquerdo)) { // CASO 3
    umNoh.pai.cor ← PRETO;
    umNoh.pai.pai.cor ← VERMELHO;
    rotacaoDireita(umNoh.pai.pai);
} senão se ((umNoh.pai == umNoh.pai.pai.esq) e ...
```

ÁRVORE RN-ARRUMAR INSERÇÃO- PSEUDOCÓDIGO - IV

```
} então se ((umNoh.pai = umNoh->pai.pai.dir) e
// (umNoh == umNoh.pai.esq)) { // CASO 4
    umNoh.cor ← PRETO;
    avo ← umNoh.pai.pai; // para manter após primeira rotação
    avo.cor ← VERMELHO;
    girarDireita(umNoh.pai);
    girarEsquerda(avo);
    // atualiza umNoh para verificação no enquanto
    umNoh ← umNoh.pai;
} então { // ((umNoh.pai = umNoh->pai.pai.esq) e
// (umNoh == umNoh.pai.dir)) --- CASO 5
    ...
```

ÁRVORE RN-ARRUMAR INSERÇÃO- PSEUDOCÓDIGO - V

```
} senão { // ((umNoh.pai = umNoh->pai.pai.esq) e  
// (umNoh == umNoh.pai.dir)) --- CASO 5  
    umNoh.cor ← PRETO;  
    avo ← umNoh.pai.pai; // para manter após primeira rotação  
    avo.cor ← VERMELHO;  
    girarEsquerda(umNoh.pai);  
    girarDireita(avo);  
    // atualiza umNoh para verificação no enquanto  
    umNoh ← umNoh.pai;  
}
```

ÁRVORE RN-ARRUMAR INSERÇÃO- PSEUDOCÓDIGO - VI

```
    } // fim do senão do “se (tio.cor = VERMELHO)”  
  } // fim do enquanto  
raiz.cor ← PRETO; // arruma cor da raiz, caso tenha alterado-a
```

OUTROS



VISÃO GERAL DA REMOÇÃO - 1/3

O processo de remoção em uma árvore rubro-negra é mais complexo. Segue-se uma visão geral do processo:

1. Caso o nó removido seja vermelho, não há problemas no balanceamento, indiferente do processo de remoção.
2. Caso o nó seja preto e ele seja substituído por um filho vermelho, basta trocar a cor desse filho para preto.

VISÃO GERAL DA REMOÇÃO - 2/3

3. Caso o nó seja preto e substituído pelo sucessor ou antecessor vermelho, também não há problemas no balanceamento, basta trocar a cor do antecessor ou sucessor para preto.

VISÃO GERAL DA REMOÇÃO - 3/3

O problema da remoção ocorre quando o nó removido é preto, e ele será substituído por um outro nó preto (incluindo NIL). Aquele ramo da árvore ficará desbalanceado, precisando de rotações para corrigir o problema.

Há um conjunto de situações a serem observadas, deixamos ao aprendiz a verificação em textos clássicos de Estruturas de Dados, ex. (Cormen, 2002), sobre o algoritmo completo de remoção em árvores rubro-negras.

ÁRVORES LLRB - 1/3

As árvores rubro-negras foram originalmente propostas por Rudolf Bayer em 1972, chamadas inicialmente de árvores binárias simétricas.

Elas adquiriram o nome atual em um trabalho de 1978 de Leonidas J. Guibas e Robert Sedgwick.

Em 2008, Sedgwick propôs uma variante da árvore rubro-negra, chamada de árvore LLRB (Left Leaning Red Black), ou árvore rubro-negra caída para a esquerda.

ÁRVORES LLRB - 2/3

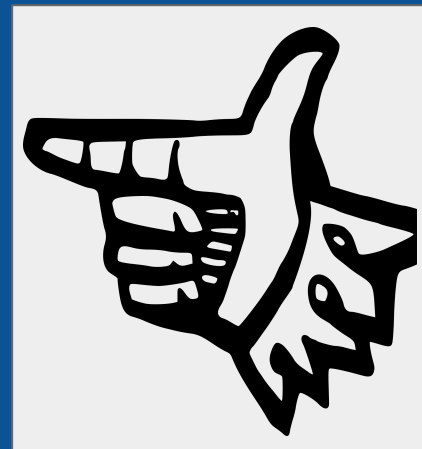
A árvore LLRB é uma variação da árvore rubro-negra que garante a mesma complexidade de operações, mas possui uma implementação mais simples e curta da inserção e remoção. A variação inclui uma restrição adicional na rubro-negra padrão (a queda para a esquerda).

Uma árvore rubro-negra é caída para a esquerda se, para todo nó com um único filho vermelho, este nó filho está à esquerda do nó pai. Ou seja, um nó só possui um filho vermelho à direita se ele também tiver um filho vermelho à esquerda.

ÁRVORES LLRB - 3/3

A LLRB permite, entre outros elementos, simular uma árvore 2-3 e uma árvore 2-3-4 como uma rubro-negra, o que garante similaridade na complexidade das operações de inserção, remoção e busca nessas árvores.

SOBRE O MATERIAL



SOBRE ESTE MATERIAL

Material produzido coletivamente, principalmente pelos seguintes professores do DCC/UFLA:

- Joaquim Quinteiro Uchôa
- Juliana Galvani Greggi
- Renato Ramos da Silva

Inclui contribuições de outros professores do setor de Fundamentos de Programação do DCC/UFLA.

Esta obra está licenciado com uma Licença [Creative Commons Atribuição 4.0 Internacional](https://creativecommons.org/licenses/by/4.0/).