Universidade Federal do Pará
Instituto de Ciências Exatas e Naturais
Programa de Pós-Graduação em Ciência da Computação

Árvores Balanceadas (Rubro-Negra)

Autor: Nelson Cruz Sampaio Neto

Carlos Gustavo Resque dos Santos
gustavoresqueufpa@gmail.com

15 de março de 2023

Árvores Vermelho-Preto

- As árvores vermelho-preto (ou rubro-negras) constituem um entre muitos esquemas de árvores de busca binária que são “balanceadas”.

- Uma árvore rubro-negra é uma árvore de busca binária com um *bit* extra de armazenamento por nó que indica sua cor: **PRETA** ou **VERMELHA**.

- Essa estrutura de dados é complexa, mas eficiente na prática, ao garantir que suas operações demorem $O(\log n)$ no pior caso.

- Foram inventadas por Bayer sob o nome “Árvores Binárias Simétricas” em 1972, 10 anos depois das árvores AVL.

Árvores Vermelho-Preto

- Uma árvore de busca binária é uma árvore rubro-negra se ela satisfaz as seguintes propriedades:

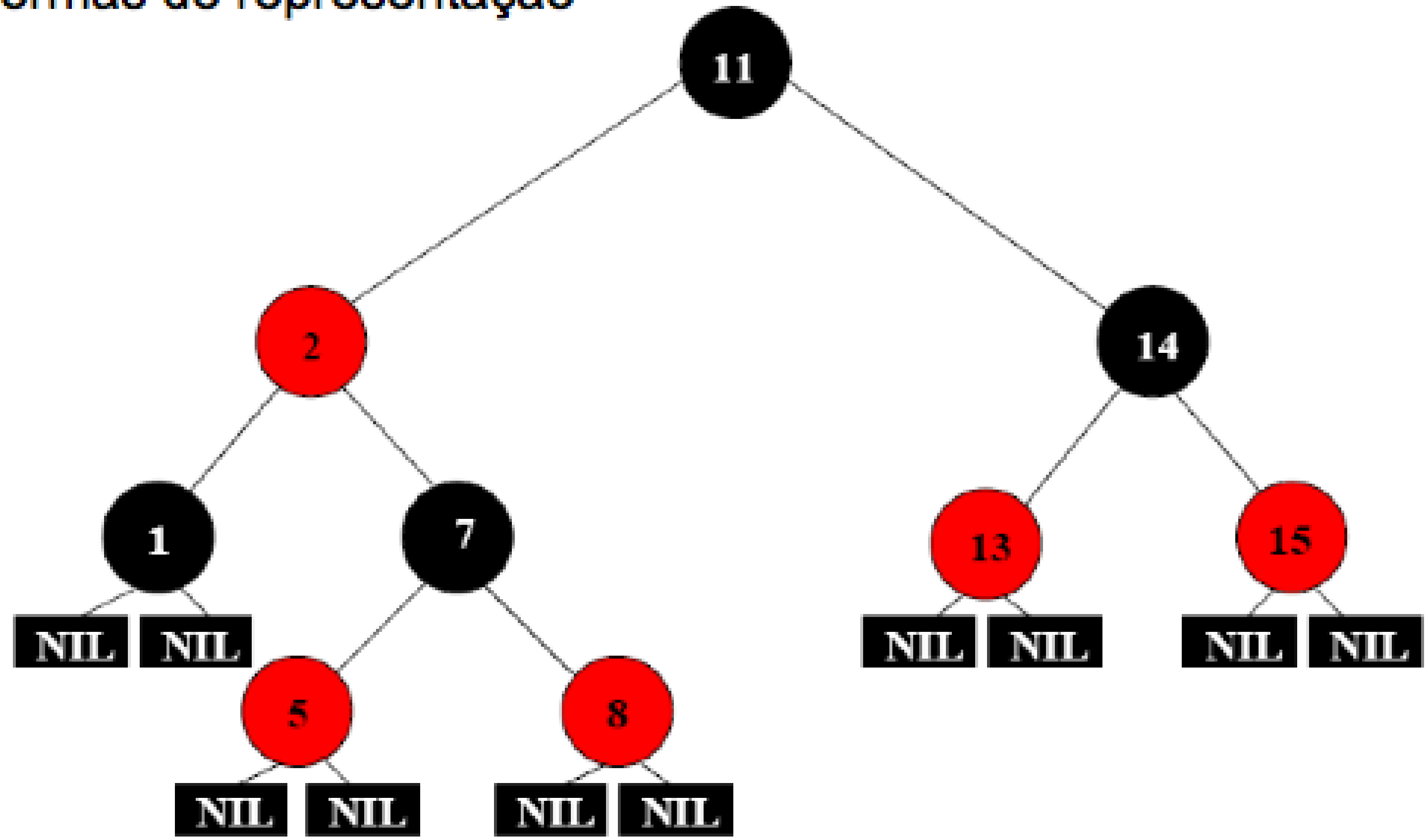
- 1. Todo nó é vermelho ou preto;
- 2. A raiz é preta;
- 3. Todo nó externo (NIL) é preto;
- 4. Se um nó é vermelho, então ambos os seus filhos são pretos;
- 5. Todos os caminhos desde um nó até os nós externos descendentes contêm o mesmo número de nós pretos.

- Um nó que satisfaz as propriedades é denominado equilibrado, caso contrário é dito desequilibrado. Na árvore rubro-negra todos os nós estão equilibrados.

- Em um caminho da raiz até uma sub-árvore vazia não pode existir dois nós vermelhos consecutivos.

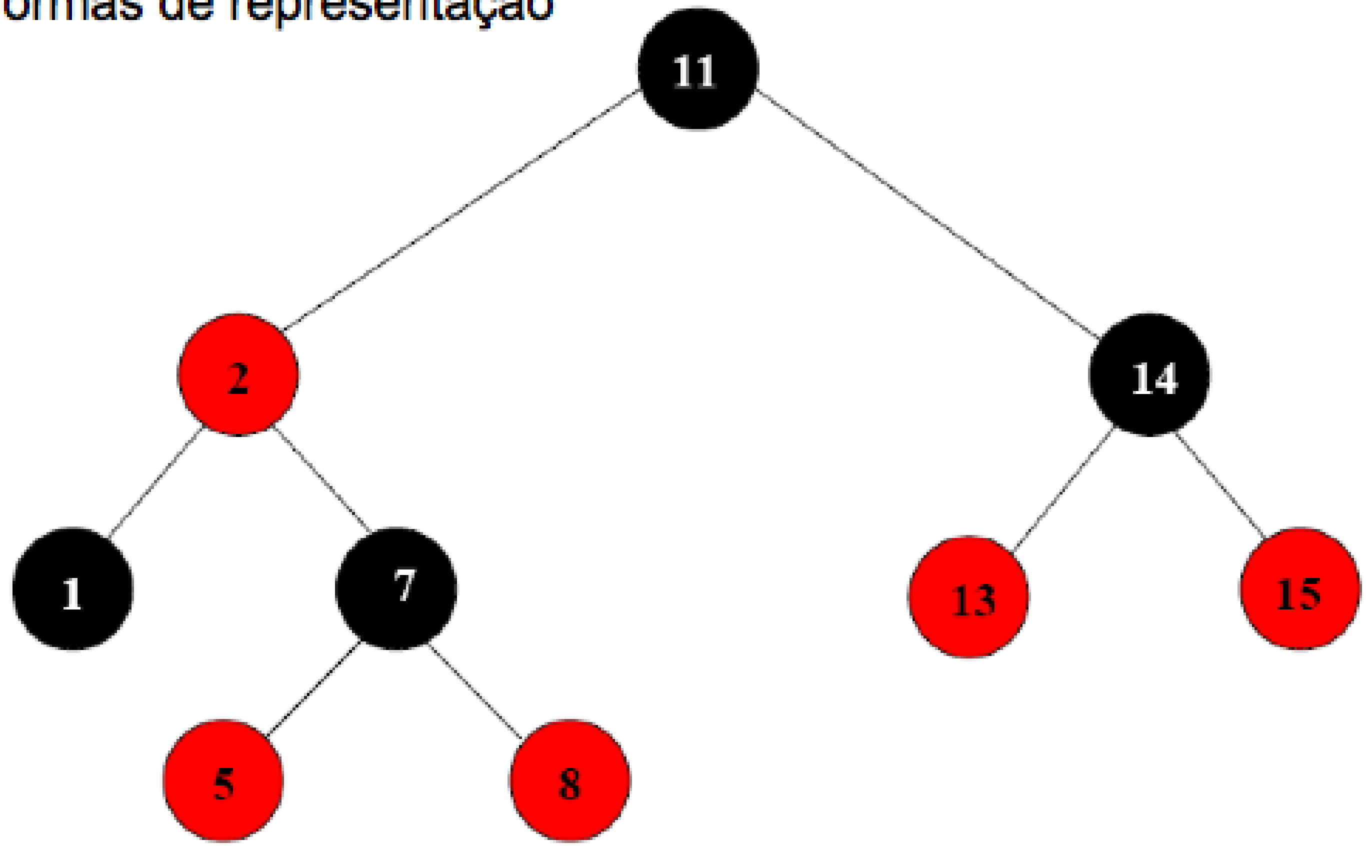
Árvores Vermelho-Preto

Formas de representação



Árvores Vermelho-Preto

Formas de representação



Árvores Vermelho-Preto

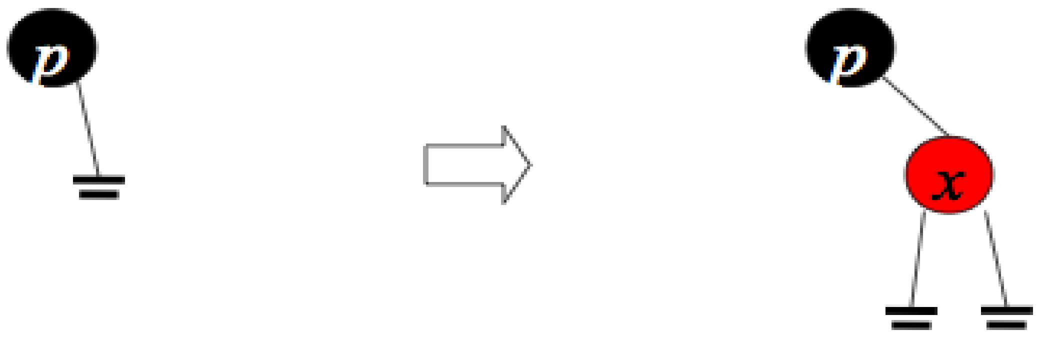
- Lema:
- "Uma árvore vermelho-preto com *n* nós internos tem altura no máximo 2 *log* (*n* + 1)."
- A prova do Lema pode ser conferida na página 222 do livro texto.
- O lema mostra que as árvores vermelho-preto constituem “boas" árvores de busca, visto que sua altura é *O*(*log n*).
-

Árvores Vermelho-Preto

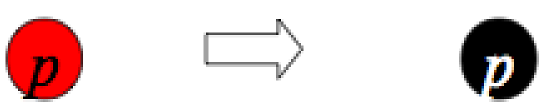
- As operações **Inserir** e **Remover** são mais complicadas nas árvores rubro-negras porque elas podem ferir alguma propriedade desse tipo de árvore.
- Como veremos, essas operações podem ser implementadas de forma bastante parecida com as respectivas operações nas árvores binárias de busca, bastando apenas modificar as cores dos nós e trocar ponteiros para que as propriedades das árvores rubro-negras sejam satisfeitas.
- Como a inserção e remoção propriamente ditas já foram vistas para árvores binárias de busca, veremos apenas o que é necessário para acertar as cores da árvore.

Árvores Vermelho-Preto

- Um nó é inserido sempre na cor vermelha.

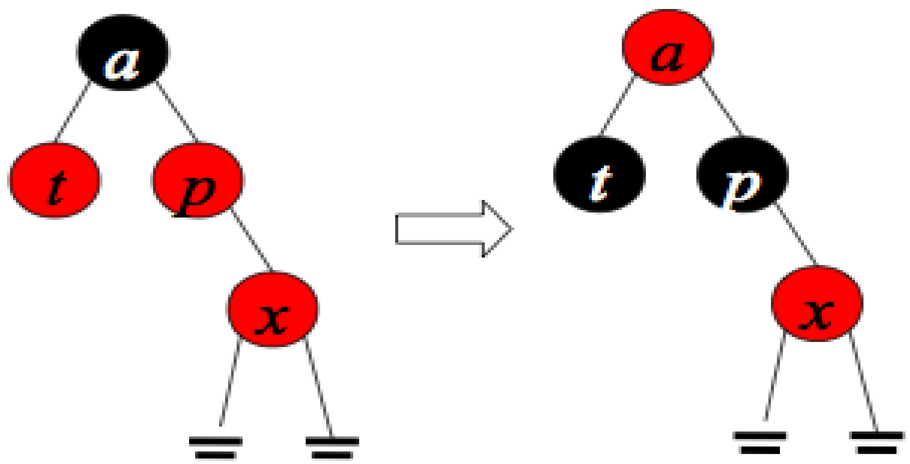


- **Caso 1**: Caso a inserção seja feita em uma árvore vazia, basta alterar a cor do nó para preto, para manter a propriedade 2.



Árvores Vermelho-Preto

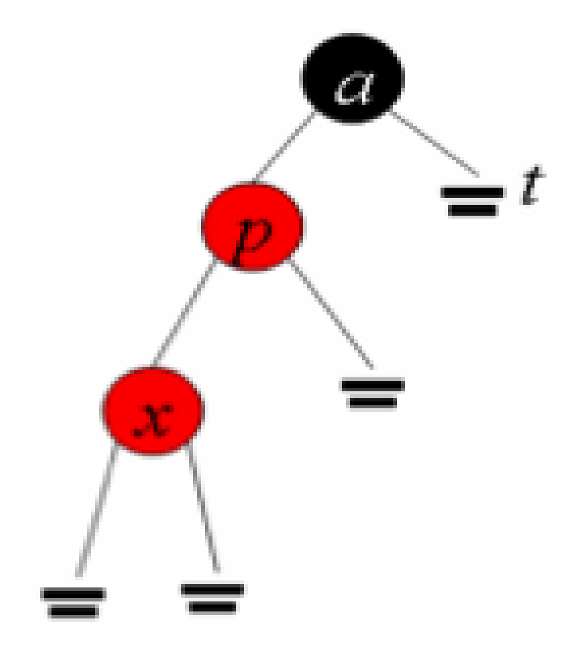
- **Caso 2**: Ao inserir *x*, se o tio de *x* é vermelho, é necessário fazer a recoloração de *a*, *t* e *p*.



- Se o pai do nó *a* é vermelho, o rebalanceamento tem que ser feito novamente, considerando o nó *a* como inserido.
- Se o nó *a* é raiz, então ele deve ser preto.

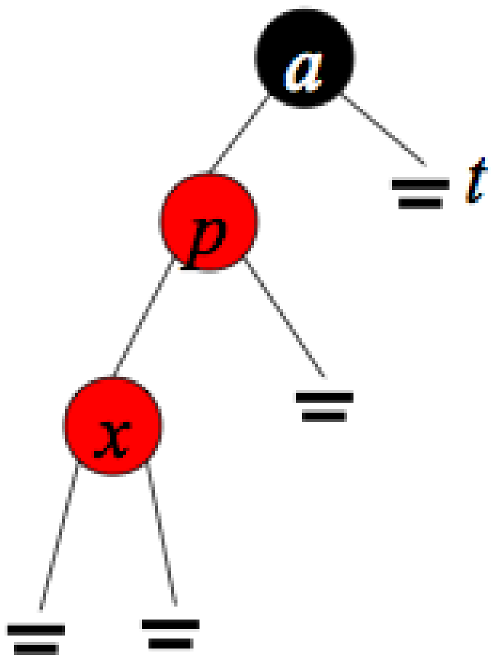
Árvores Vermelho-Preto

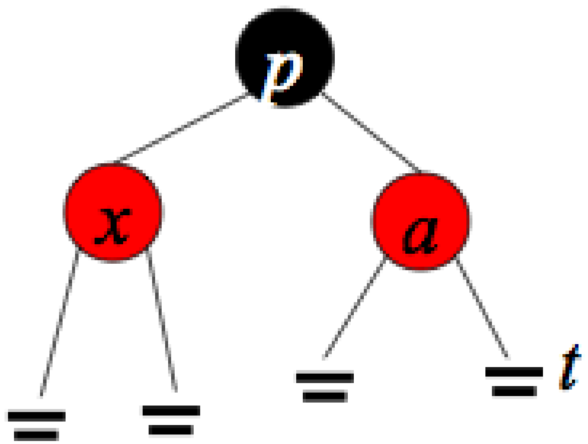
- **Caso 3**: Suponha que o tio do elemento inserido *x* seja preto. Nesse caso, para manter a propriedade 4 é preciso fazer rotações envolvendo *a*, *t*, *p* e *x*.
- Há 4 subcasos que correspondem às 4 rotações possíveis.



Árvores Vermelho-Preto

- **Caso 3a**: Rotação à direita.

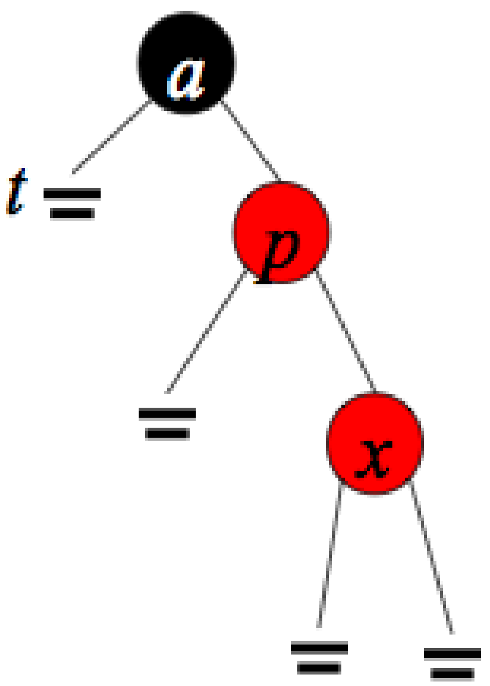


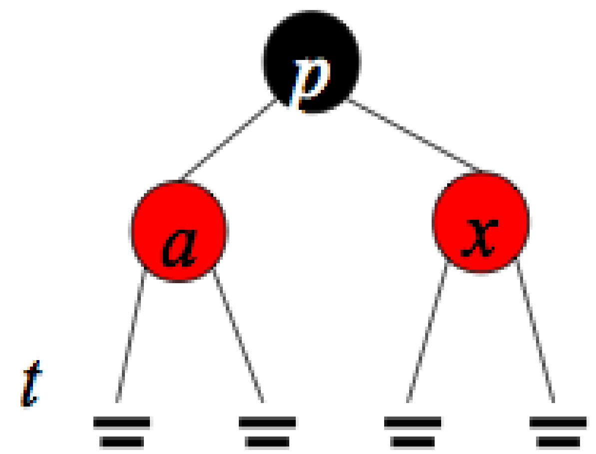


Recoloração de p e a

Árvores Vermelho-Preto

- **Caso 3b**: Rotação à esquerda.

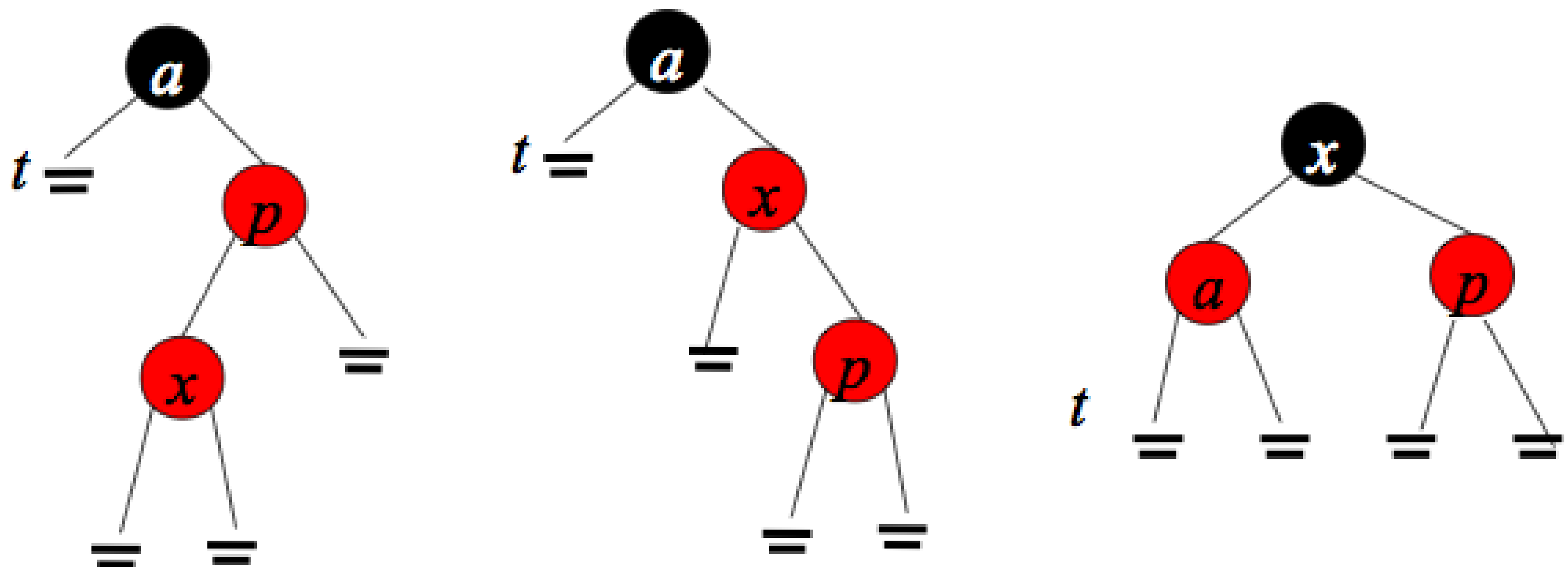




Recoloração de p e a

Árvores Vermelho-Preto

- **Caso 3c**: Rotação dupla à esquerda.



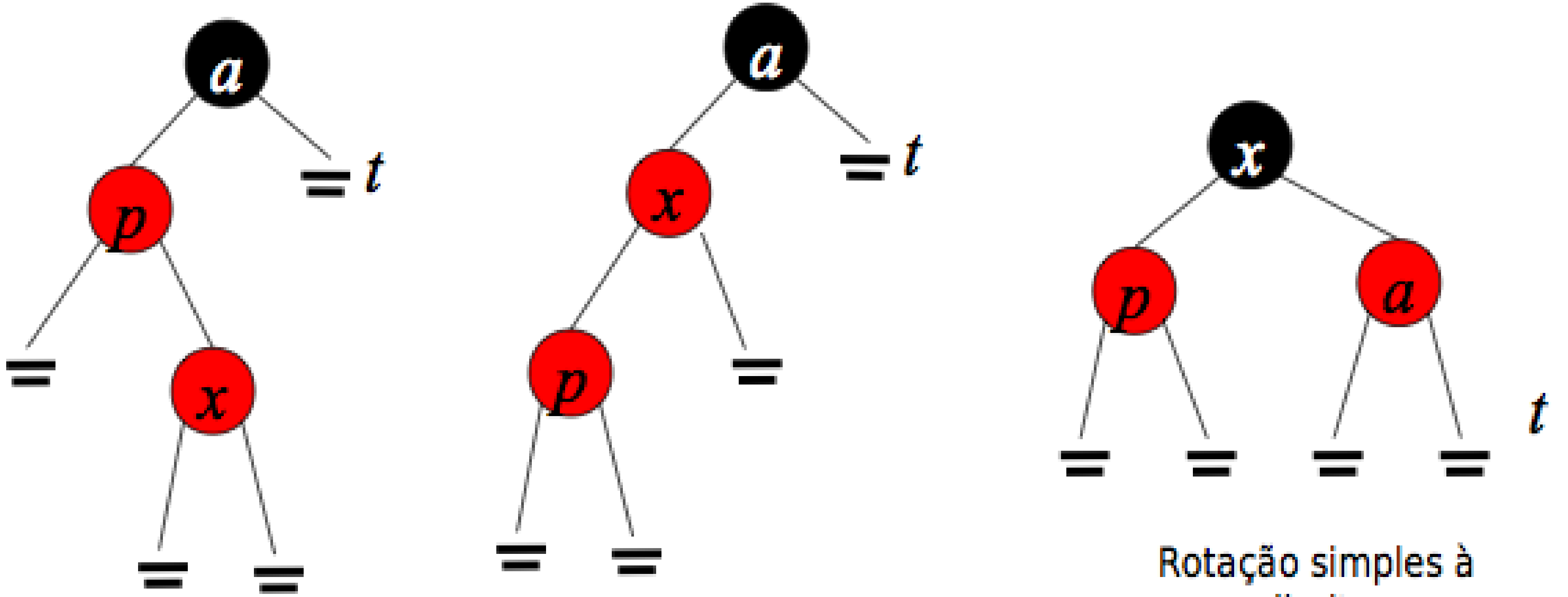
Rotação simples à direita

Rotação simples à esquerda

Recoloração de x e a

Árvores Vermelho-Preto

- **Caso 3d**: Rotação dupla à direita.

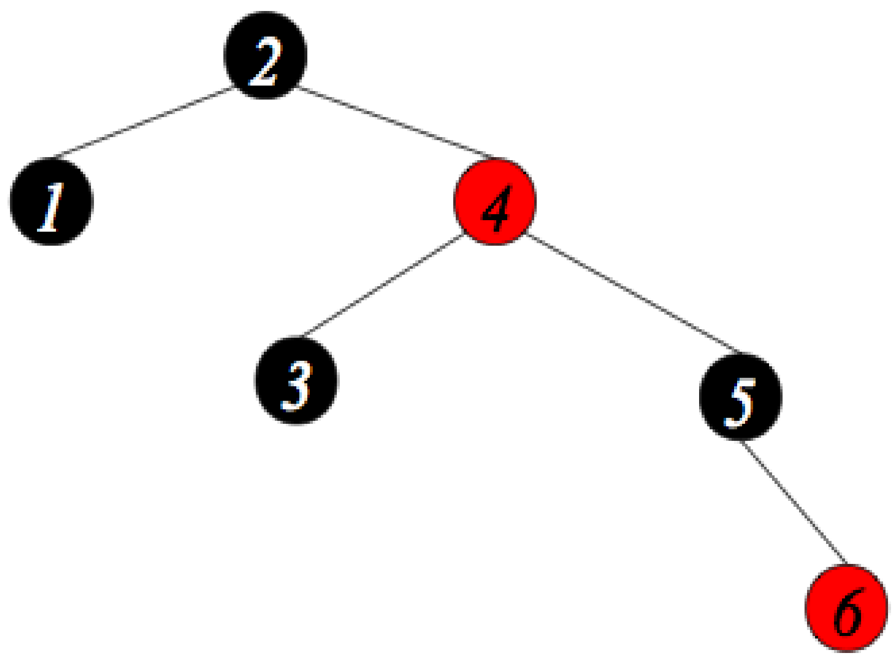


Rotação simples à esquerda

Rotação simples à direita
Recoloração de x e a

Exercício

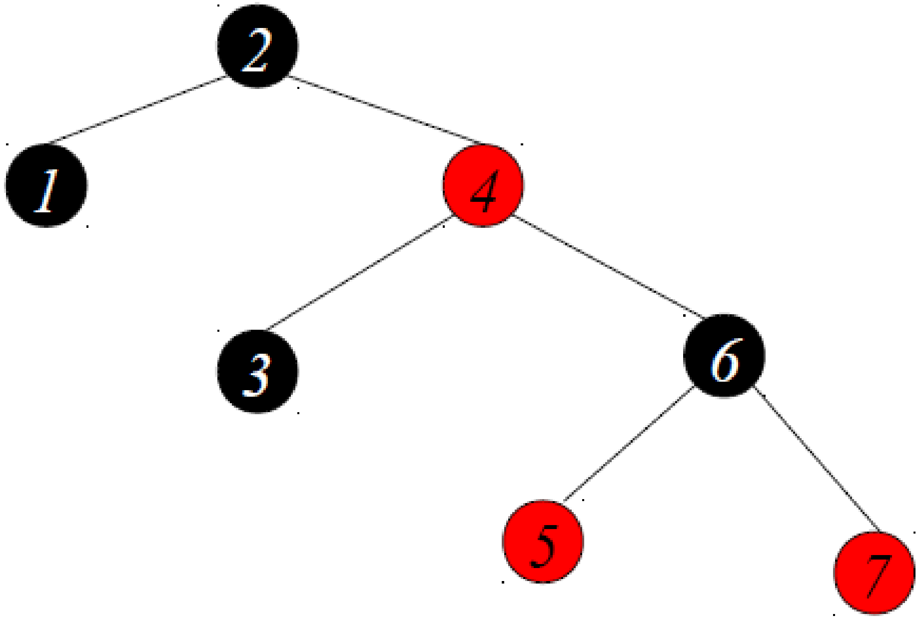
- Insira um nó com chave 7 na árvore rubro-negra abaixo e trabalhe para manter seu balanceamento.



Solução

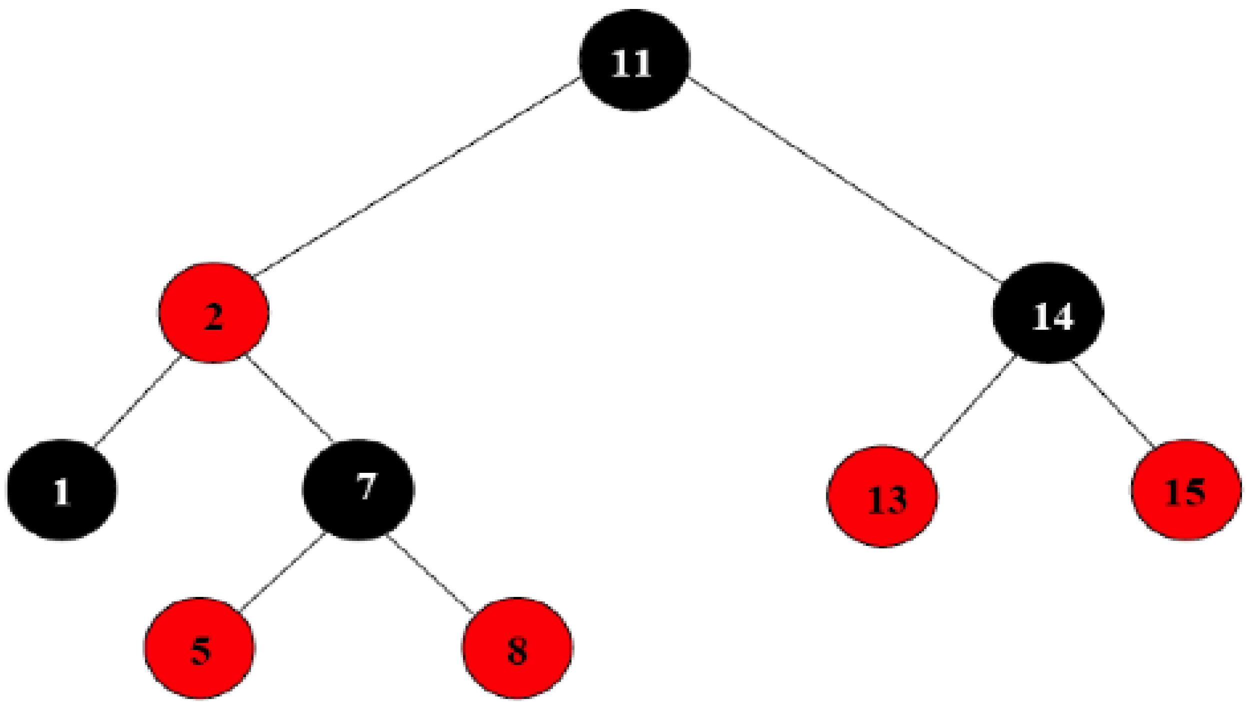
- Aplicar o **Caso 3b**:

- 1. Rotação à esquerda dos nós 5, 6 e 7.
- 2. Alteração da cor dos nós 5 e 6.



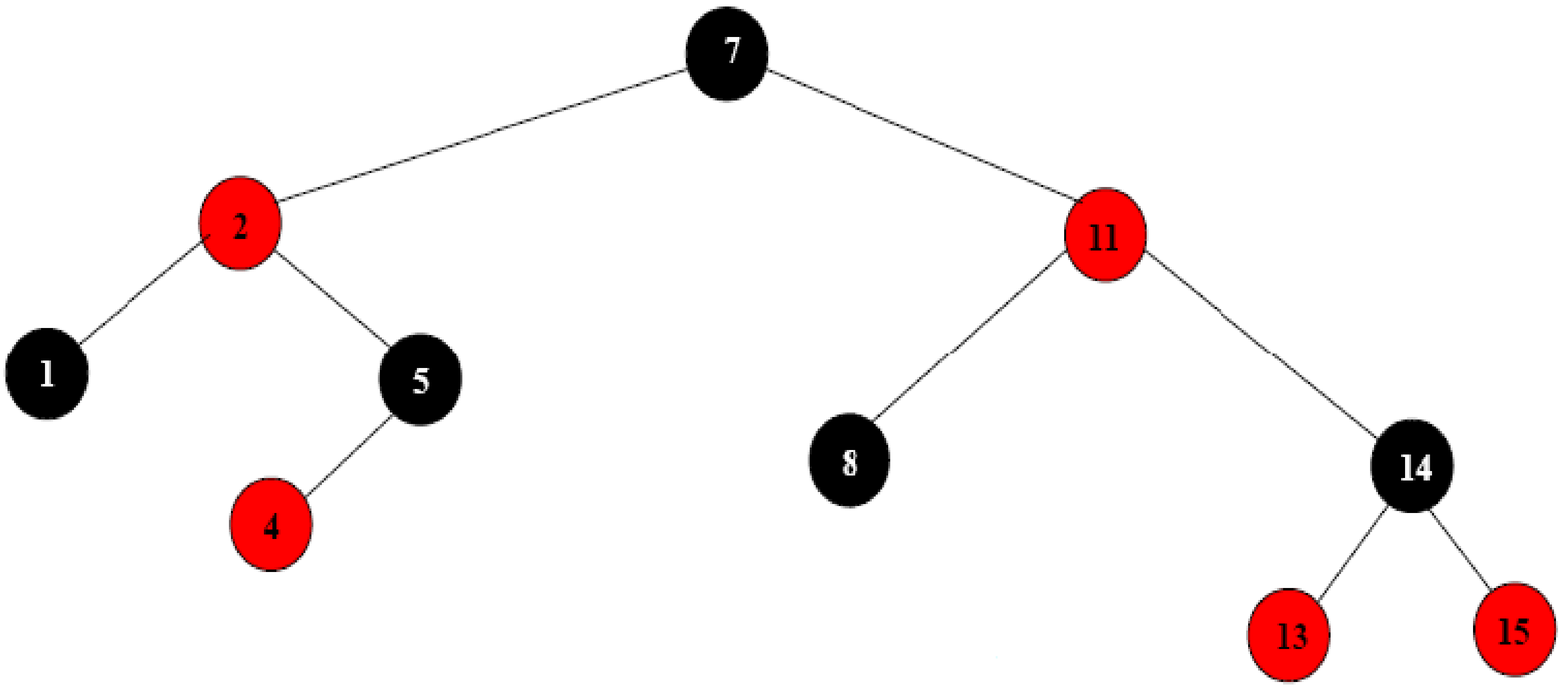
Exercício

- Insira um nó com chave 4 na árvore rubro-negra abaixo e trabalhe para manter seu balanceamento.



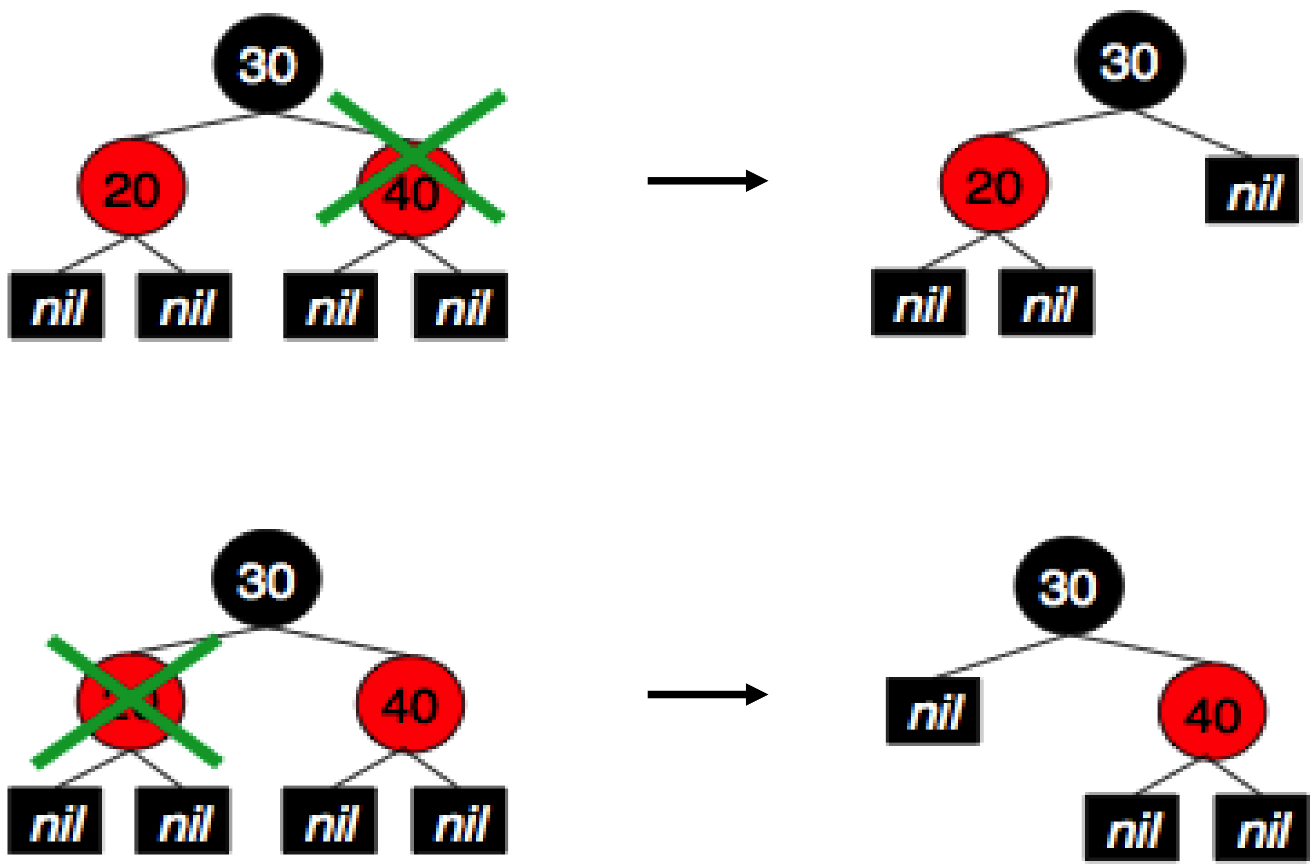
Solução

- Alteração de cor dos nós 5, 7 e 8 → **Caso 2**
- Rotação dupla à direita dos nós 2, 7 e 11 → **Caso 3d**



Árvores Vermelho-Preto

- A remoção de um nó vermelho **não** altera o balanceamento da árvore rubro-negra.

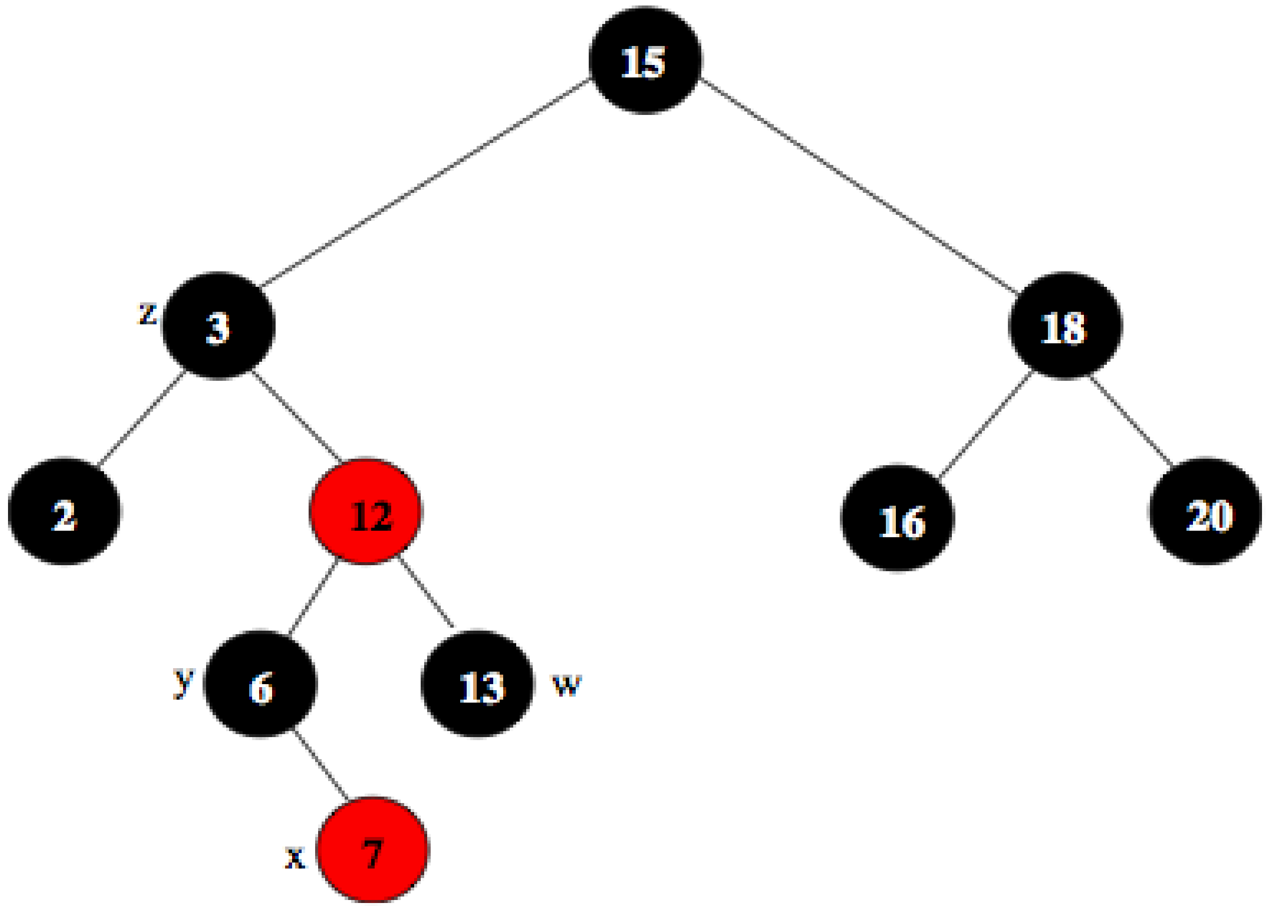


Árvores Vermelho-Preto

- Existem casos para corrigir as cores **após** a remoção de um nó preto, mas antes de defini-los, identificaremos alguns nós:

  - Seja *z* o nó a ser removido.
  - Seja *y* = *z*, se *z* possui um ou nenhum filho,
  - ou *y* = *sucessor(z)*, se *z* possui dois filho.
  - Seja *x* o filho de *y* antes da remoção de *z*, ou NIL, caso *y* não possua filho.
  - Seja *w* o tio de *x* antes da remoção de *z*.

Árvores Vermelho-Preto



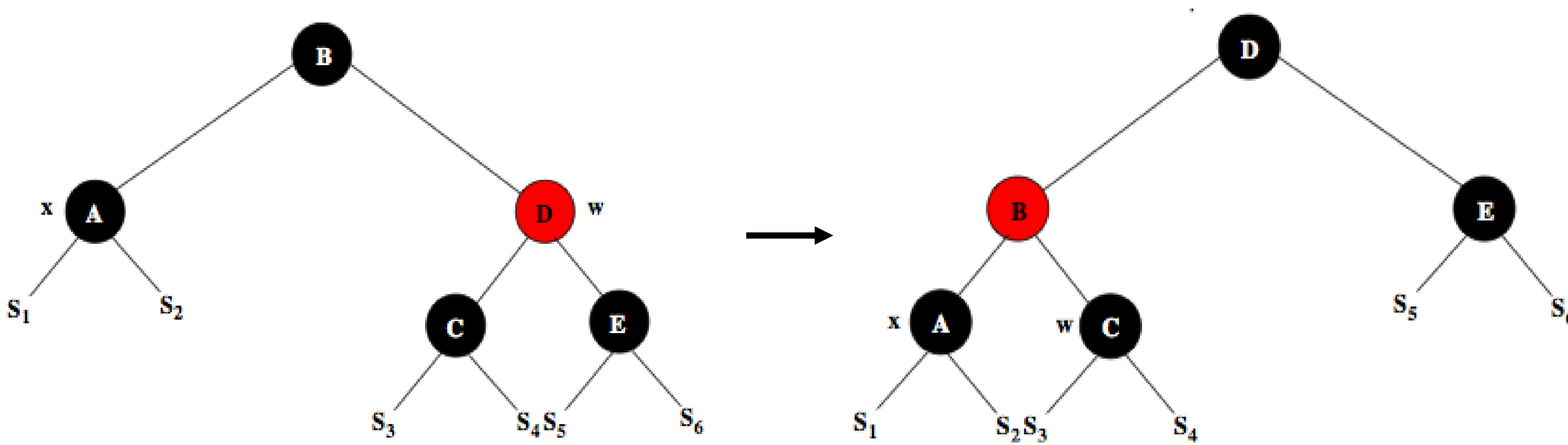
Árvores Vermelho-Preto

1. Caso o nó *z* tenha dois filhos e o nó *y* seja vermelho, basta realizar o procedimento de remoção do nó *z* e depois atribuir a cor preta ao nó *y* que a árvore estará balanceada.
2. Se após a remoção do nó *z* o nó *x* for a raiz da árvore ou da cor vermelha, então basta alterar sua cor para preto que a árvore estará balanceada. Por exemplo, na remoção do nó com chave 3 da árvore do slide anterior.
3. Mas, enquanto o nó *x* for diferente da raiz da árvore e sua cor for preta, 4 (quatro) casos serão repetidos no intuito de restaurar as propriedades vermelho-preto da árvore de busca.

   OBS: Quando w estiver a esquerda espelhe os casos.
4. Quando o critério de parada (descrito no passo 2) for satisfeito, a cor preta é atribuída ao nó *x* e a árvore estará balanceada.

Árvores Vermelho-Preto

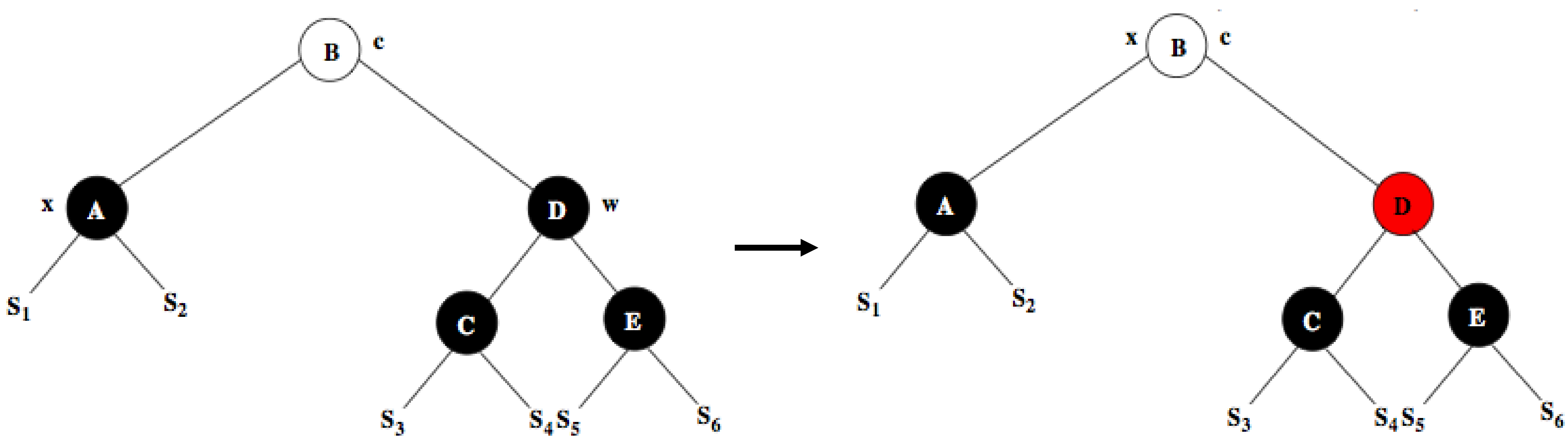
- **Caso 1**: O irmão *w* de *x* é vermelho.



- O caso 1 é transformado no caso 2, 3 ou 4 pela troca de cores dos nós B e D e pela execução de uma rotação à esquerda.

Árvores Vermelho-Preto

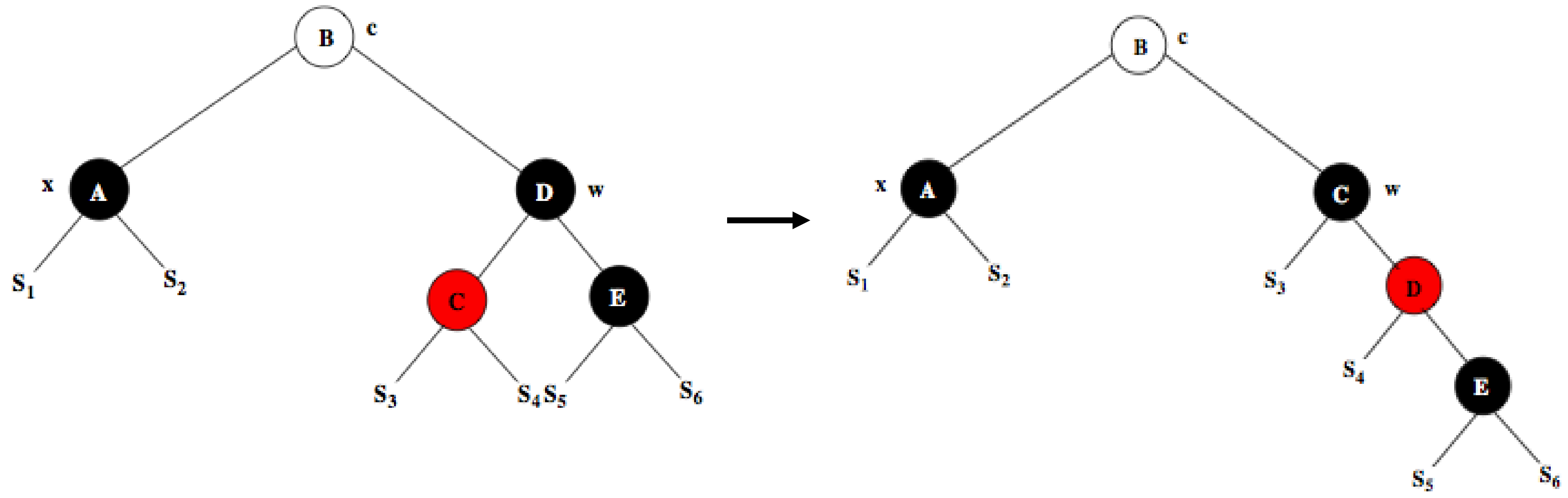
- **Caso 2**: O nó *w* é preto, e ambos os filhos de *w* são pretos.



- Se entrarmos no caso 2 vindo do caso 1, o critério de parada (o nó x será vermelho) será atendido após a transformação.

Árvores Vermelho-Preto

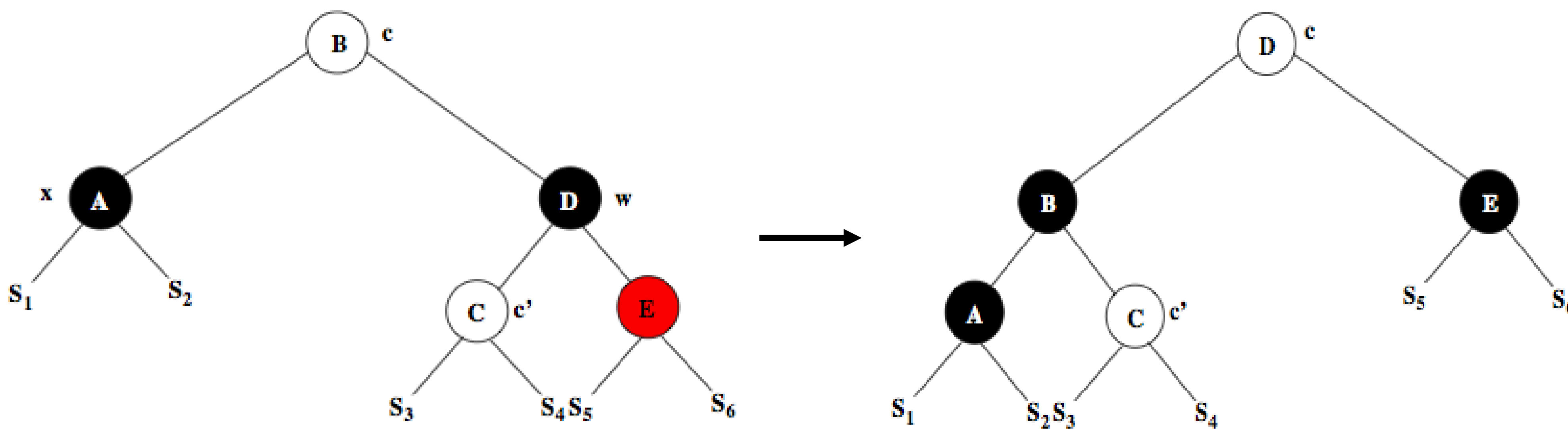
- **Caso 3**: O nó *w* é preto, e o filho da esquerda de *w* é vermelho e o filho da direita de *w* é preto.



- O caso 3 é transformado no caso 4 pela troca de cores dos nós C e D e pela execução de uma rotação à direita.

Árvores Vermelho-Preto

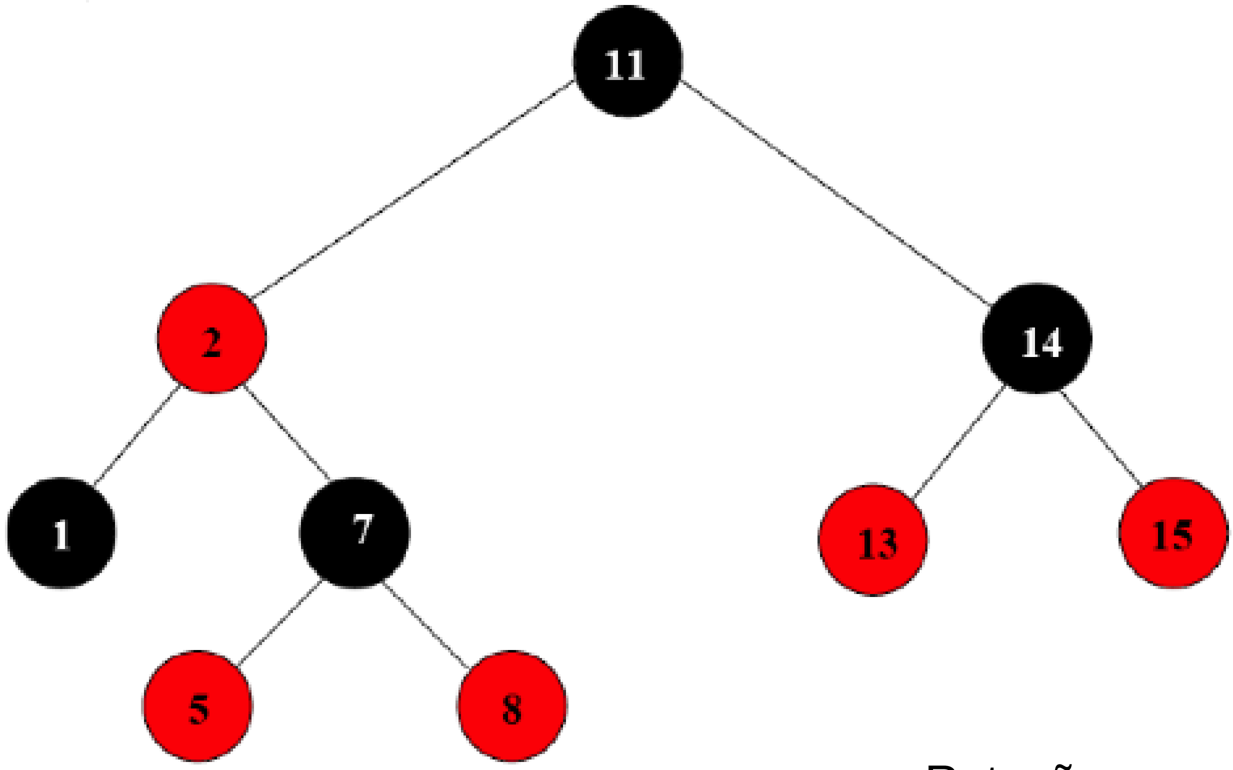
- **Caso 4**: O nó *w* é preto, e o filho da direita de *w* é vermelho.



- O critério de parada será atendido após a transformação, já que o novo *x* será a raiz da árvore.

Exercício

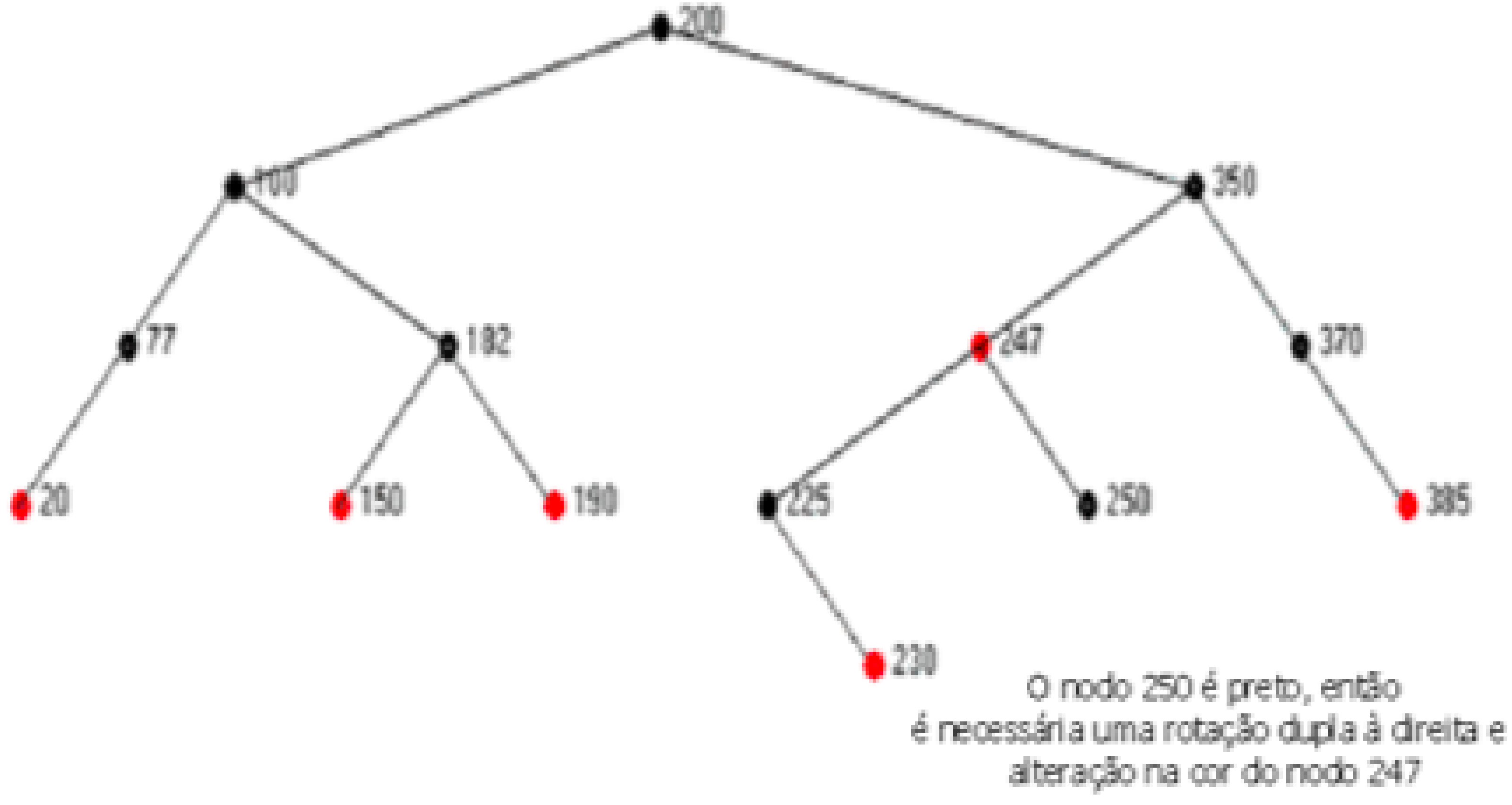
1. 1. Remova o nó com chave 1 da árvore rubro-negra abaixo e trabalhe para manter seu balanceamento.



Rotação a esquerda e
alteração
na cor dos nós 2, 7 e 8

Exercício

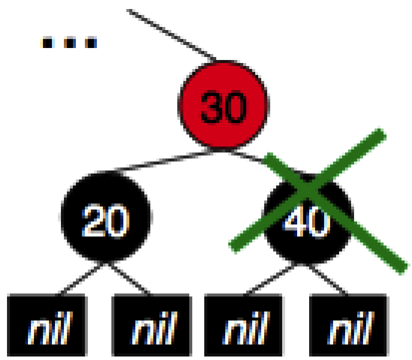
1. 2. Remova o nó com chave 250 da árvore rubro-negra abaixo e trabalhe para manter seu balanceamento.



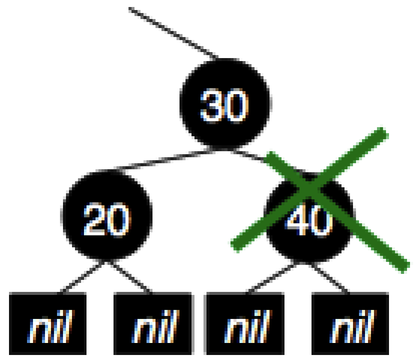
Exercício

- 3. Explique o procedimento realizado para a remoção da chave 40 nas situações (i) e (ii) abaixo.

- (i)



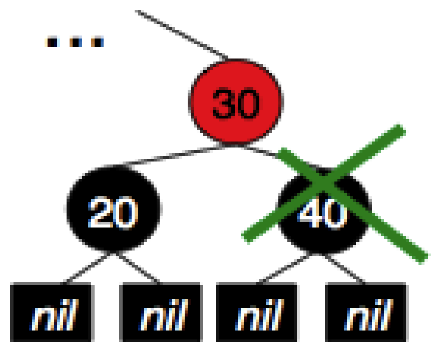
- (ii)



Exercício

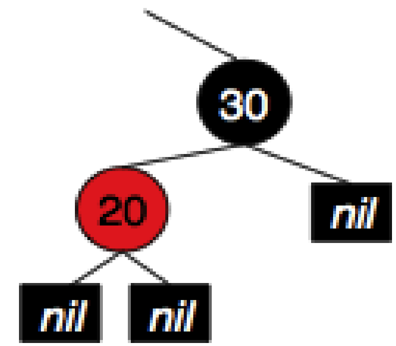
- 3. Explique o procedimento realizado para a remoção da chave 40 nas situações (i) e (ii) abaixo.

- (i)

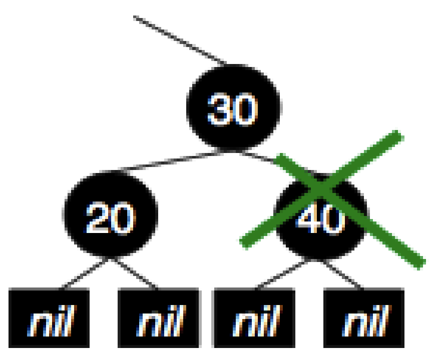


Caso 2

---->

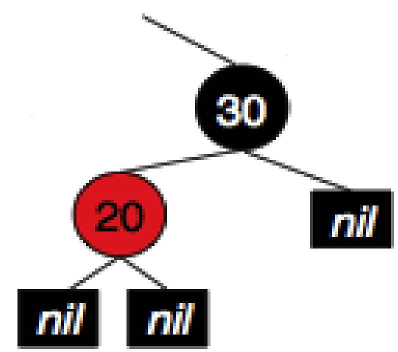


- (ii)



Caso 2

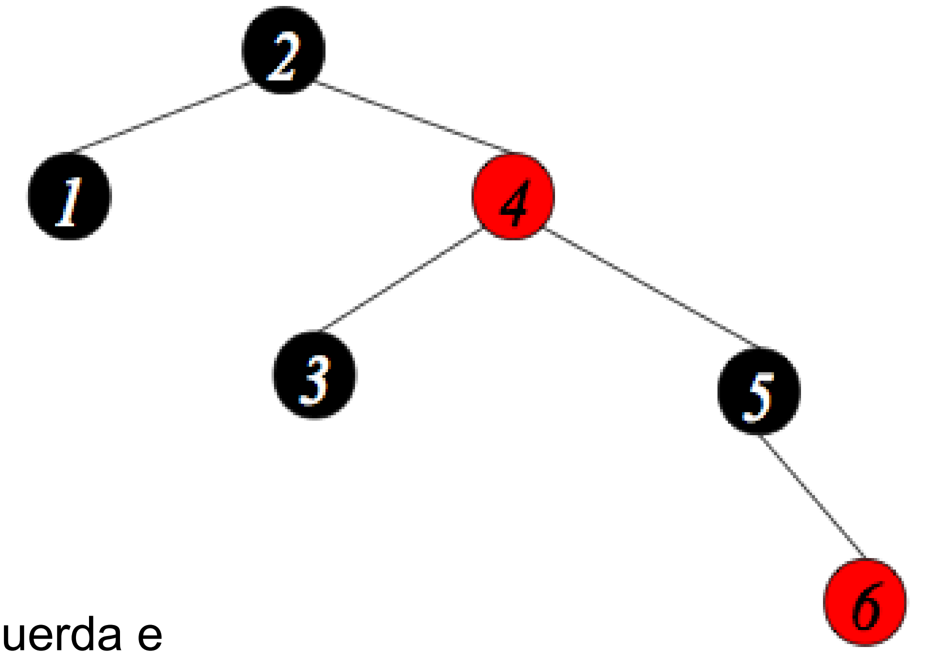
---->



...

Exercício

- 4. Remova o nó com chave 1 da árvore rubro-negra abaixo e trabalhe para manter seu balanceamento.



Rotação a esquerda e
alteração
na cor dos nós 3 e 4.

Soluções

- **Exercício 1:** Aplicar o Caso 4.
- **Exercício 2:** Aplicar o Caso 3 e, em seguida, o Caso 4.
- **Exercício 3:**
  - (i) Aplicar o Caso 2. Como o novo *x* (ou seja, a chave 30) é vermelho, o critério de parada foi satisfeito. Logo, basta mudar a cor do novo *x* de vermelho para preto.
  - (ii) Aplicar o Caso 2 e seguir com a análise.
- **Exercício 4:** Aplicar o Caso 1 e, em seguida, o Caso 2. Após aplicar o Caso 2, o novo *x* (ou seja, a chave 2) é vermelho, logo, a cor do novo *x* muda de vermelho para preto.