Análise de Algoritmos

Parte 3: Árvores Vermelho-Preto

Eduardo Freire Nakamura
eduardo.nakamura@fucapi.br

CESF – FUCAPI
Ciência da Computação, Engenharia da Computação e Sistemas de Informação

Árvores vermelho-preto (rubro-negras)

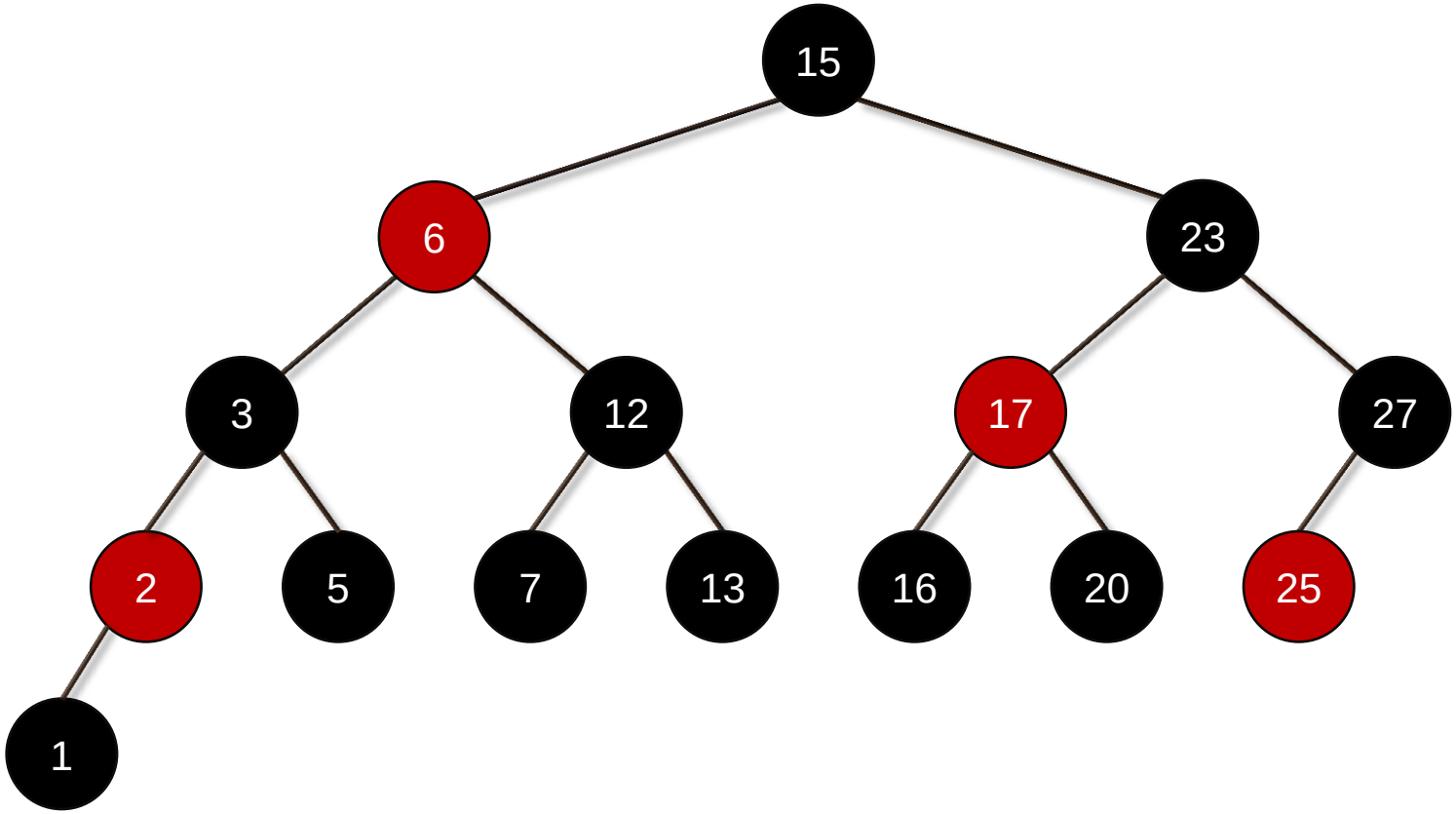
- O que é
  - Árvore de pesquisa binária
  - Um bit a mais por nó: VERMELHO ou PRETO

- Esquema bem definido para coloração
  - Nenhum caminho será 2x maior que o comprimento de qualquer outro
  - Árvore balanceada (aproximadamente)
  - A altura será no máximo 2 log(n+1)

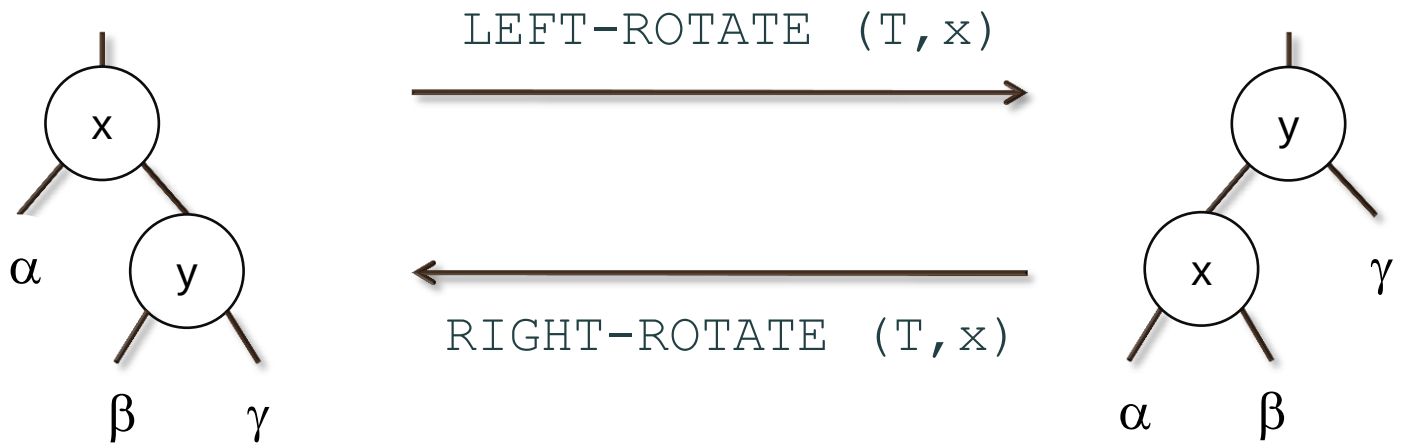
Regras

- Regras para balanceamento
  - O nó raiz é sempre preto
  - Nós vermelhos possuem apenas filhos pretos
  - Para cada nó, todos os caminhos do nodo até qualquer folha passa pelo mesmo número de nós pretos

Um exemplo

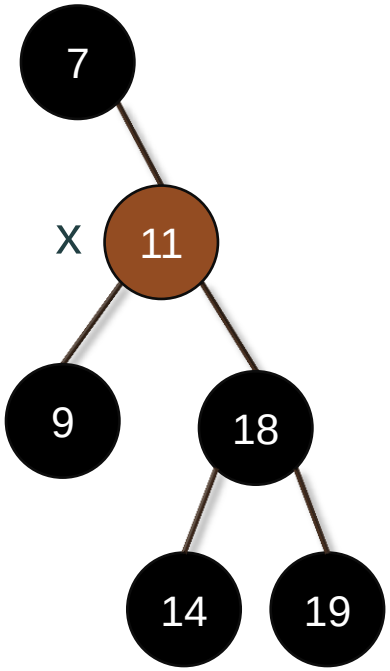


Rotação



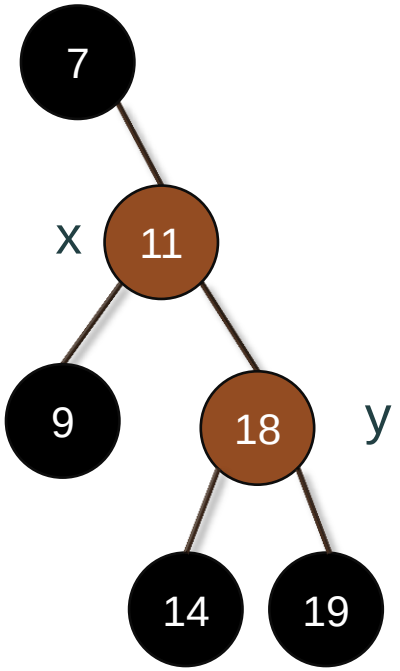
Rotação

```
LEFT-ROTATE(T,x)
 1: y ← x.dir;
 2: x.dir ← y.esq;
 3: (y.esq).pai ← x;
 4: if x.pai = NIL then
 5:    T.raiz ← y;
 6: else
 7:    if x = (x.pai).esq then
 8:       (x.pai).esq ← y;
 9:    else
10:       (x.pai).dir ← y;
11:    end if
12: end if
13: y.esq ← x;
14: x.pai ← y;
```



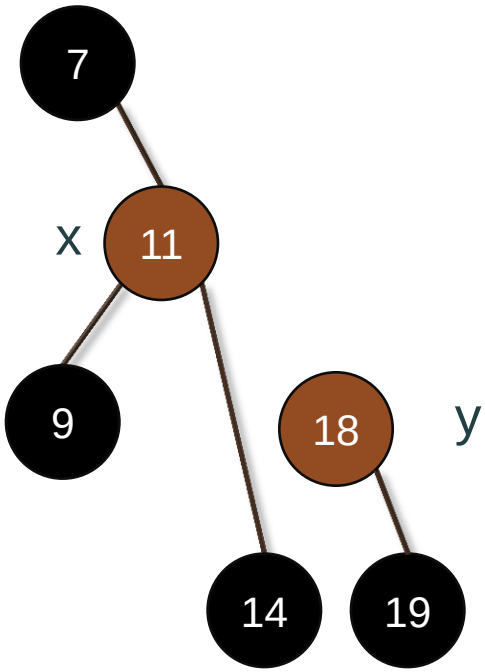
Rotação

```
LEFT-ROTATE(T,x)
 1: y ← x.dir;
 2: x.dir ← y.esq;
 3: (y.esq).pai ← x;
 4: if x.pai = NIL then
 5:    T.raiz ← y;
 6: else
 7:    if x = (x.pai).esq then
 8:       (x.pai).esq ← y;
 9:    else
10:       (x.pai).dir ← y;
11:    end if
12: end if
13: y.esq ← x;
14: x.pai ← y;
```



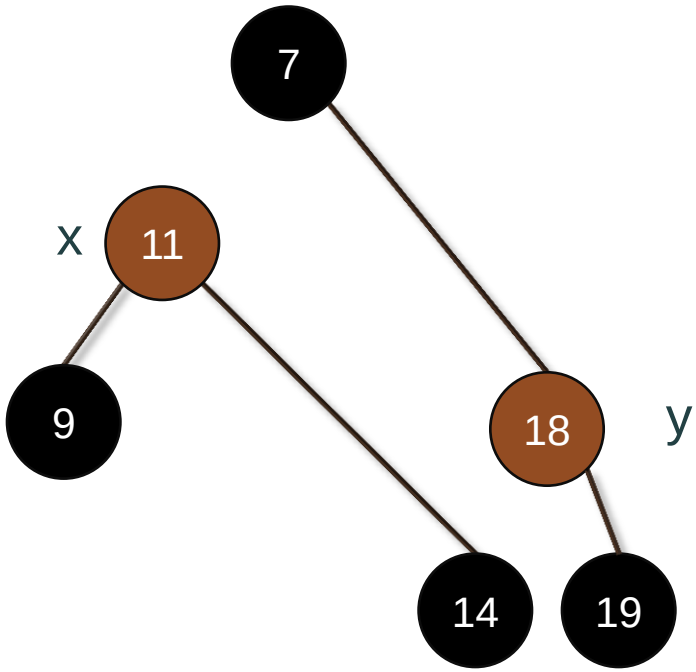
Rotação

```
LEFT-ROTATE(T,x)
 1: y ← x.dir;
 2: x.dir ← y.esq;
 3: (y.esq).pai ← x;
 4: if x.pai = NIL then
 5:    T.raiz ← y;
 6: else
 7:    if x = (x.pai).esq then
 8:       (x.pai).esq ← y;
 9:    else
10:       (x.pai).dir ← y;
11:    end if
12: end if
13: y.esq ← x;
14: x.pai ← y;
```



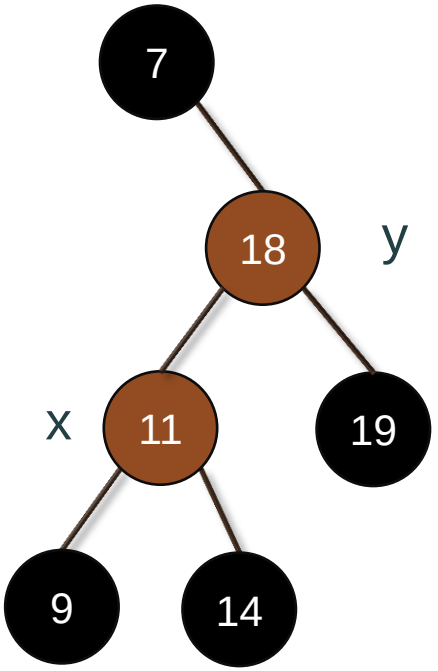
Rotação

```
LEFT-ROTATE(T,x)
 1: y ← x.dir;
 2: x.dir ← y.esq;
 3: (y.esq).pai ← x;
 4: if x.pai = NIL then
 5:    T.raiz ← y;
 6: else
 7:    if x = (x.pai).esq then
 8:       (x.pai).esq ← y;
 9:    else
10:       (x.pai).dir ← y;
11:    end if
12: end if
13: y.esq ← x;
14: x.pai ← y;
```



Rotação

```
LEFT-ROTATE(T,x)
 1: y ← x.dir;
 2: x.dir ← y.esq;
 3: (y.esq).pai ← x;
 4: if x.pai = NIL then
 5:    T.raiz ← y;
 6: else
 7:    if x = (x.pai).esq then
 8:       (x.pai).esq ← y;
 9:    else
10:       (x.pai).dir ← y;
11:    end if
12: end if
13: y.esq ← x;
14: x.pai ← y;
```



Rotação

```
LEFT-ROTATE(T,x)
 1: y ← x.dir;
 2: x.dir ← y.esq;
 3: (y.esq).pai ← x;
 4: if x.pai = NIL then
 5:    T.raiz ← y;
 6: else
 7:    if x = (x.pai).esq then
 8:       (x.pai).esq ← y;
 9:    else
10:       (x.pai).dir ← y;
11:    end if
12: end if
13: y.esq ← x;
14: x.pai ← y;
```

Qual é o custo desta operação?

Rotação

```
LEFT-ROTATE(T,x)
 1: y ← x.dir;
 2: x.dir ← y.esq;
 3: (y.esq).pai ← x;
 4: if x.pai = NIL then
 5:    T.raiz ← y;
 6: else
 7:    if x = (x.pai).esq then
 8:       (x.pai).esq ← y;
 9:    else
10:       (x.pai).dir ← y;
11:    end if
12: end if
13: y.esq ← x;
14: x.pai ← y;
```

Como seria o RIGHT-ROTATE?

Inserção

```
VP-INSERT(T,z)
 1: TREE-INSERT(T,z);
 2: z.cor ← VERMELHO;
 3: VP-INSERT-FIX(T,z);
```

Inserção

```
VP-INSERT-FIX(T,z)
 1: while (z.pai).cor = VERMELHO do
 2:   if z.pai = ((z.pai).pai).esq then
 3:     y ← ((z.pai).pai).dir;
 4:     if y.cor = VERMELHO then
 5:       (z.pai).cor ← PRETO;
 6:       y.cor ← PRETO;
 7:       ((x.pai).pai).cor ← VERMELHO;
 8:       z ← (z.pai).pai;
 9:     else if z = (z.pai).dir then
10:       z ← z.pai;
11:       LEFT-ROTATE(T,z);
12:       (z.pai).cor ← PRETO;
13:       ((z.pai).pai).cor ← VERMELHO;
14:       RIGHT-ROTATE(T,(z.pai).pai);
15:   else (igual ao "if" trocando "dir" e "esq")
16:   end if
17: (T.raiz).cor ← PRETO;
```

Inserção

```
VP-INSERT-FIX(T,z)
 1: while (z.pai).cor = VERMELHO do
 2:   if z.pai = ((z.pai).pai).esq then
 3:     y ← ((z.pai).pai).dir;
 4:     if y.cor = VERMELHO then
 5:       (z.pai).cor ← PRETO;
 6:       y.cor ← PRETO;
 7:       ((x.pai).pai).cor ← VERMELHO;
 8:       z ← (z.pai).pai;
 9:     else if z = (z.pai).dir then
10:       z ← z.pai;
11:       LEFT-ROTATE(T,z);
12:       (z.pai).cor ← PRETO;
13:       ((z.pai).pai).cor ← VERMELHO;
14:       RIGHT-ROTATE(T,(z.pai).pai);
15:   else (igual ao "if" trocando "dir" e "esq")
16:   end if
17: (T.raiz).cor ← PRETO;
```

O tio y de z é vermelho

Inserção

```
VP-INSERT-FIX(T,z)
 1: while (z.pai).cor = VERMELHO do
 2:   if z.pai = ((z.pai).pai).esq then
 3:     y ← ((z.pai).pai).dir;
 4:     if y.cor = VERMELHO then
 5:       (z.pai).cor ← PRETO;
 6:       y.cor ← PRETO;
 7:       ((x.pai).pai).cor ← VERMELHO;
 8:       z ← (z.pai).pai;
 9:     else if z = (z.pai).dir then
10:       z ← z.pai;
11:       LEFT-ROTATE(T,z);
12:       (z.pai).cor ← PRETO;
13:       ((z.pai).pai).cor ← VERMELHO;
14:       RIGHT-ROTATE(T,(z.pai).pai);
15:   else (igual ao "if" trocando "dir" e "esq")
16:   end if
17: (T.raiz).cor ← PRETO;
```

O tio y de z é preto
e z é um filho da direita

Inserção

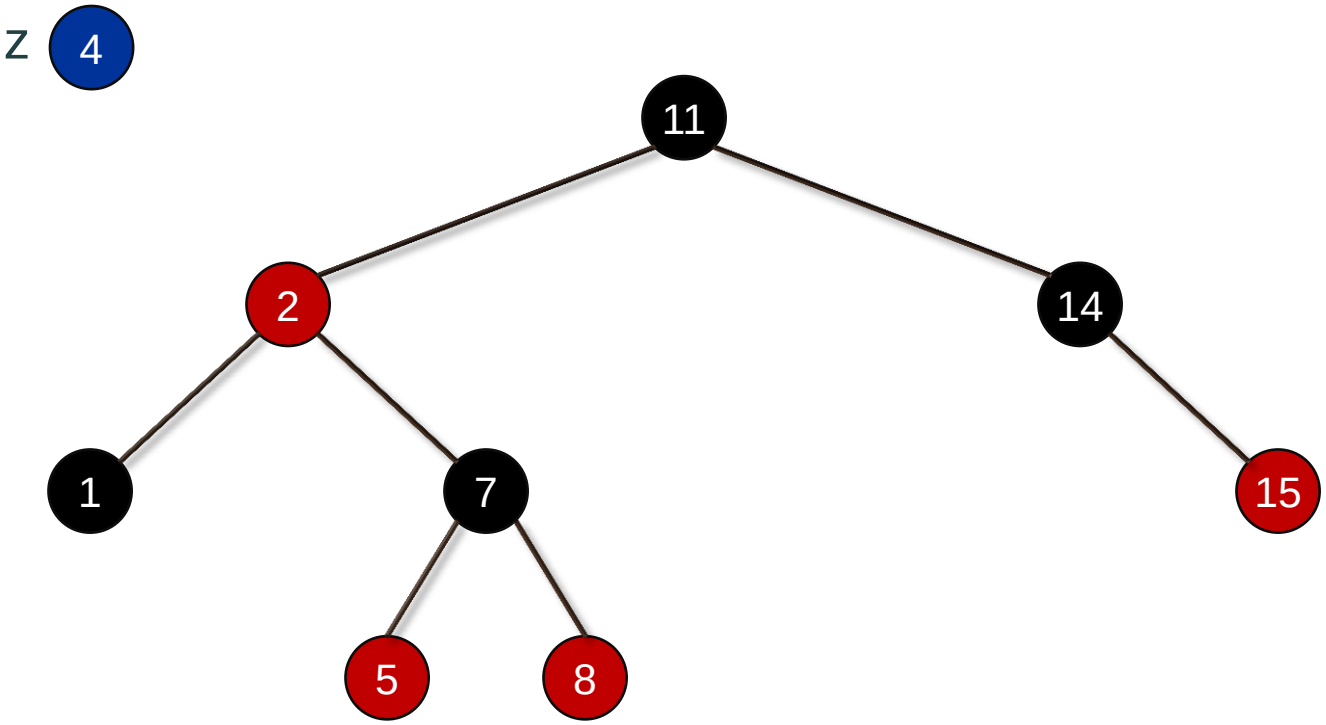
```
VP-INSERT-FIX(T,z)
 1: while (z.pai).cor = VERMELHO do
 2:   if z.pai = ((z.pai).pai).esq then
 3:     y ← ((z.pai).pai).dir;
 4:     if y.cor = VERMELHO then
 5:       (z.pai).cor ← PRETO;
 6:       y.cor ← PRETO;
 7:       ((x.pai).pai).cor ← VERMELHO;
 8:       z ← (z.pai).pai;
 9:     else if z = (z.pai).dir then
10:       z ← z.pai;
11:       LEFT-ROTATE(T,z);
12:       (z.pai).cor ← PRETO;
13:       ((z.pai).pai).cor ← VERMELHO;
14:       RIGHT-ROTATE(T,(z.pai).pai);
15:   else (igual ao "if" trocando "dir" e "esq")
16:   end if
17: (T.raiz).cor ← PRETO;
```

O tio y de z é preto e z é um filho da esq.

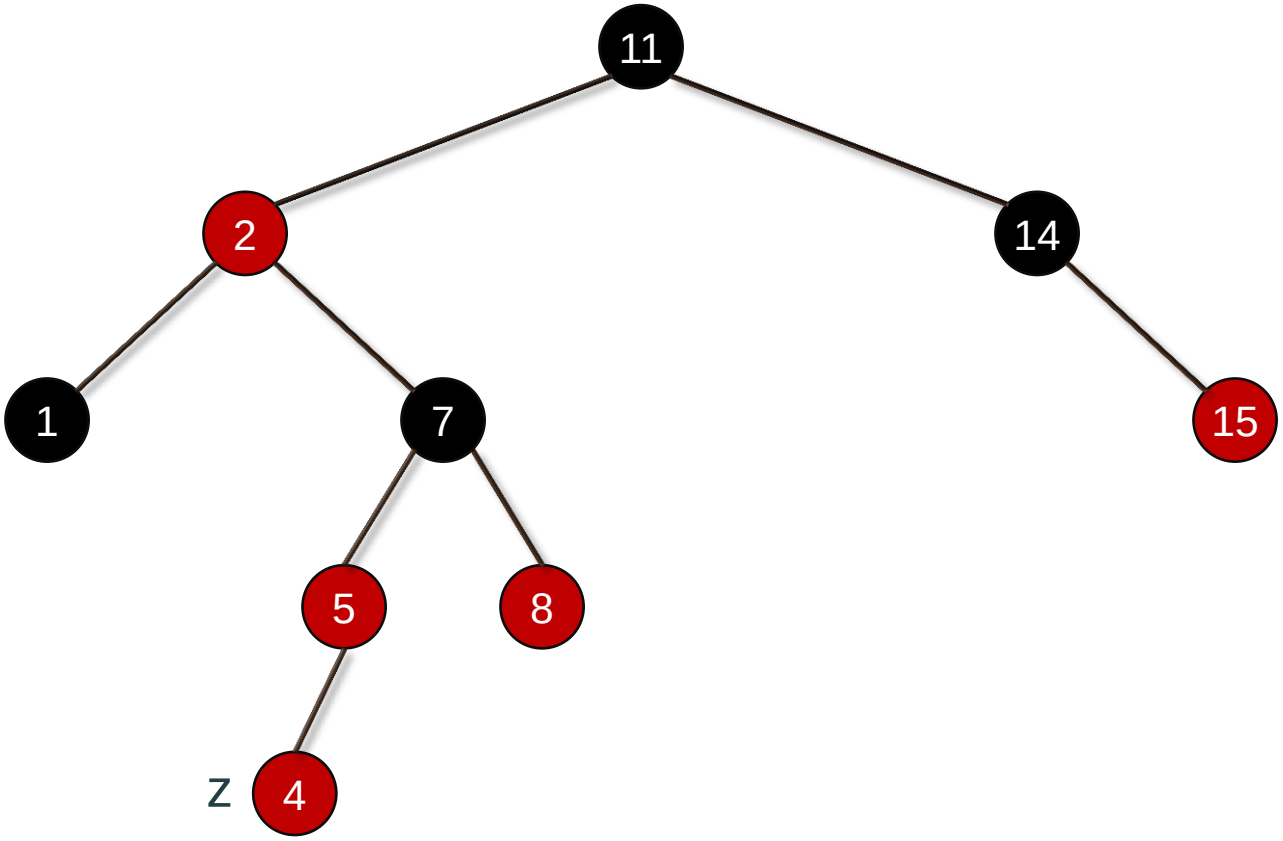
Inserção

```
VP-INSERT-FIX(T,z)
 1: while (z.pai).cor = VERMELHO do
 2:   if z.pai = ((z.pai).pai).esq then
 3:     y ← ((z.pai).pai).dir;
 4:     if y.cor = VERMELHO then
 5:       (z.pai).cor ← PRETO;
 6:       y.cor ← PRETO;
 7:       ((z.pai).pai).cor ← VERMELHO;
 8:       z ← (z.pai).pai;
 9:     else if z = (z.pai).dir then
10:       z ← z.pai;
11:       LEFT-ROTATE(T,z);
12:       (z.pai).cor ← PRETO;
13:       ((z.pai).pai).cor ← VERMELHO;
14:       RIGHT-ROTATE(T,(z.pai).pai);
15:   else (igual ao "if" trocando "dir" e "esq")
16:   end if
17: (T.raiz).cor ← PRETO;
```

Inserção

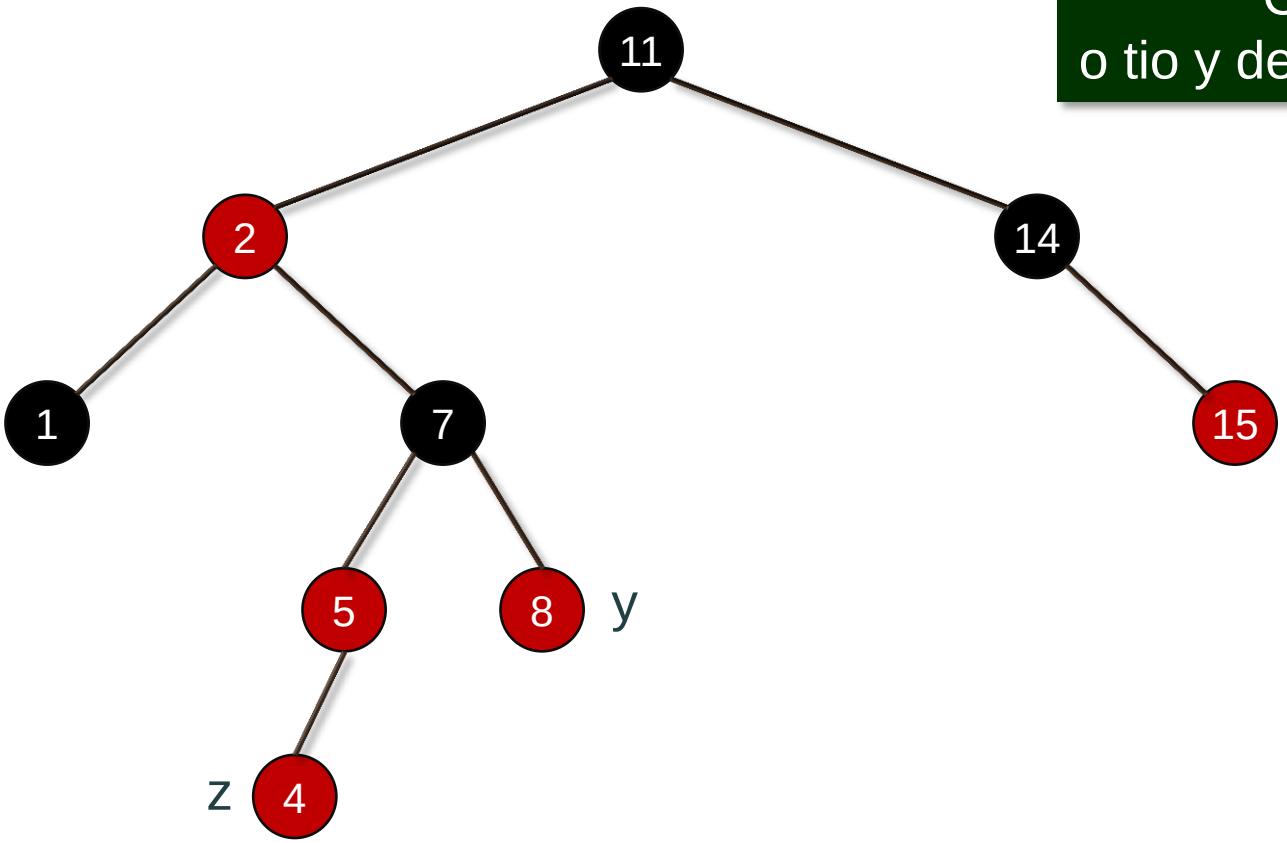


Inserção

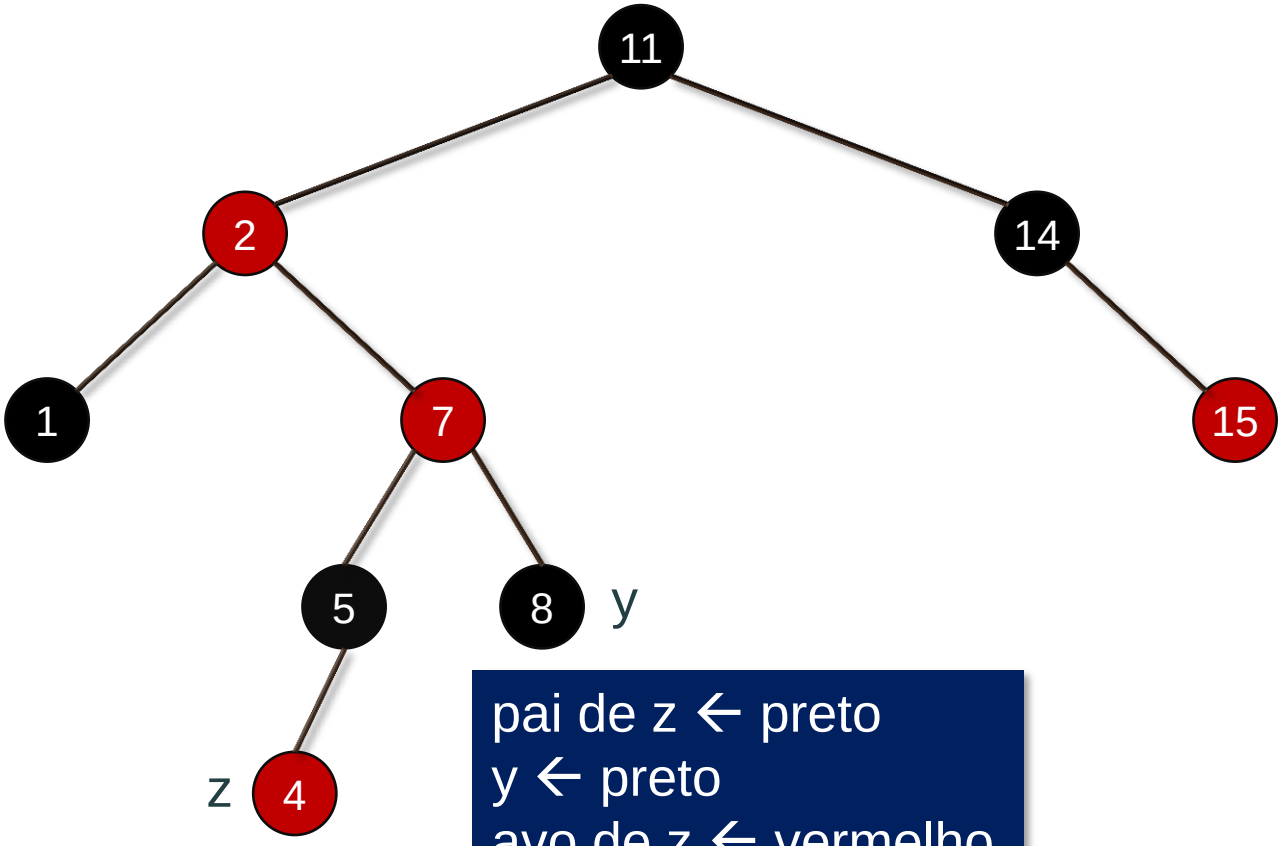


Inserção

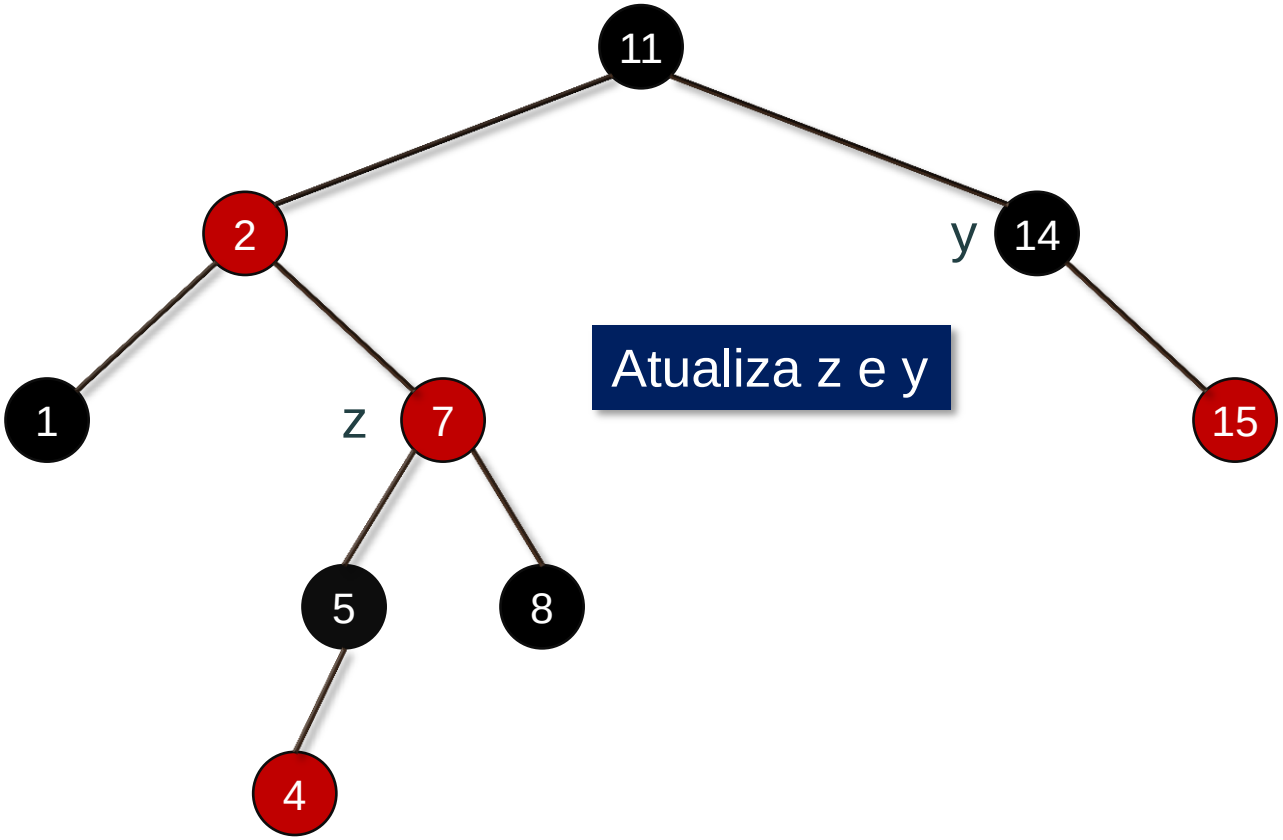
Caso 1
o tio y de z é vermelho



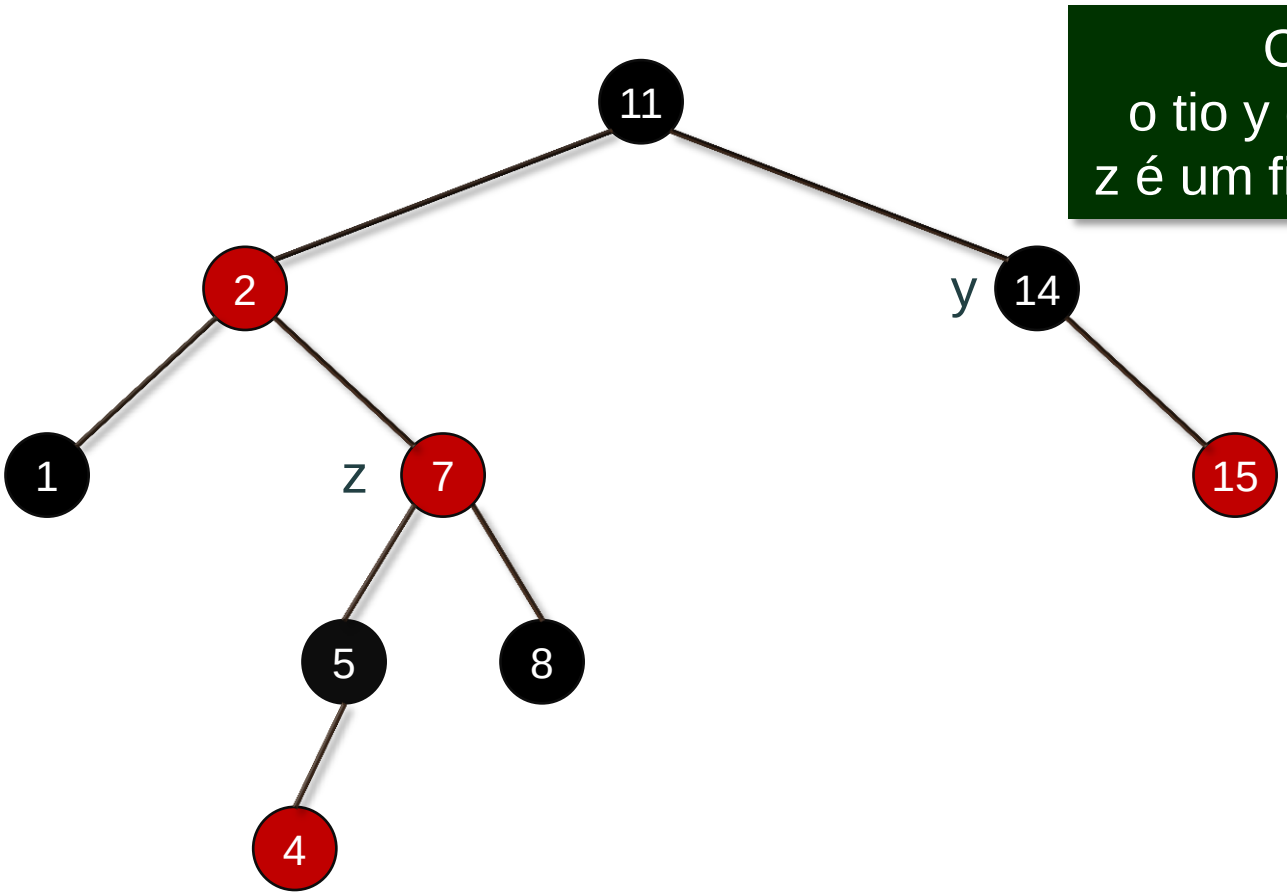
Inserção



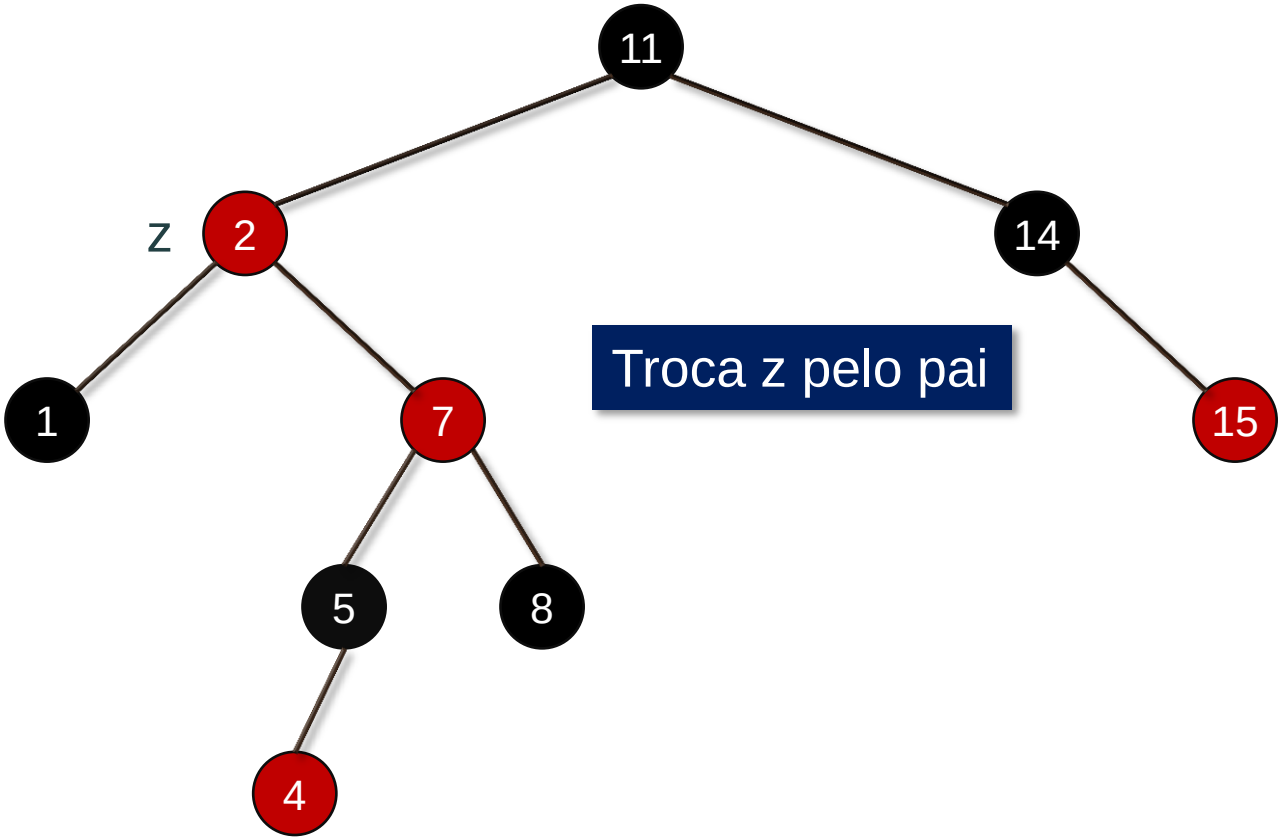
Inserção



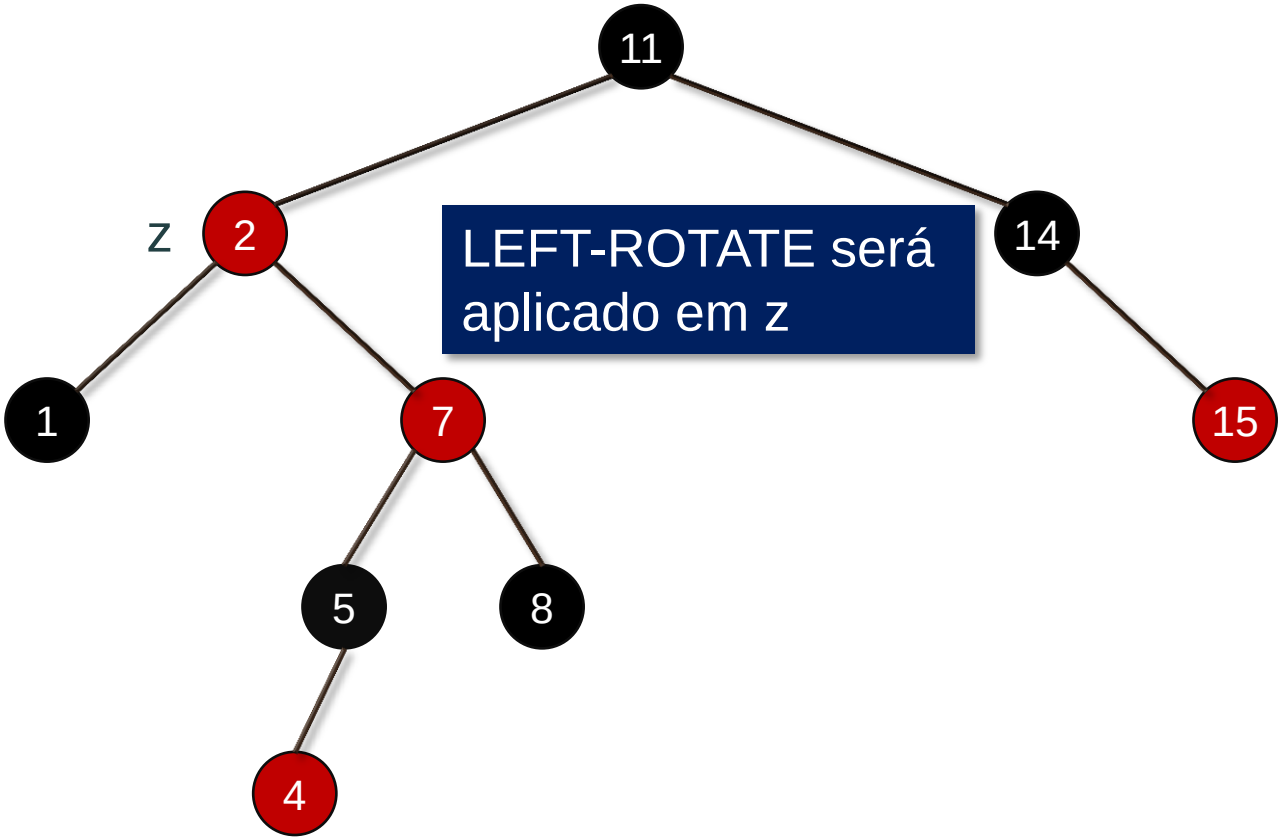
Inserção


Caso 2
o tio y de z é preto
z é um filho da direita
11
2
y
14
1
z
7
15
5
8
4

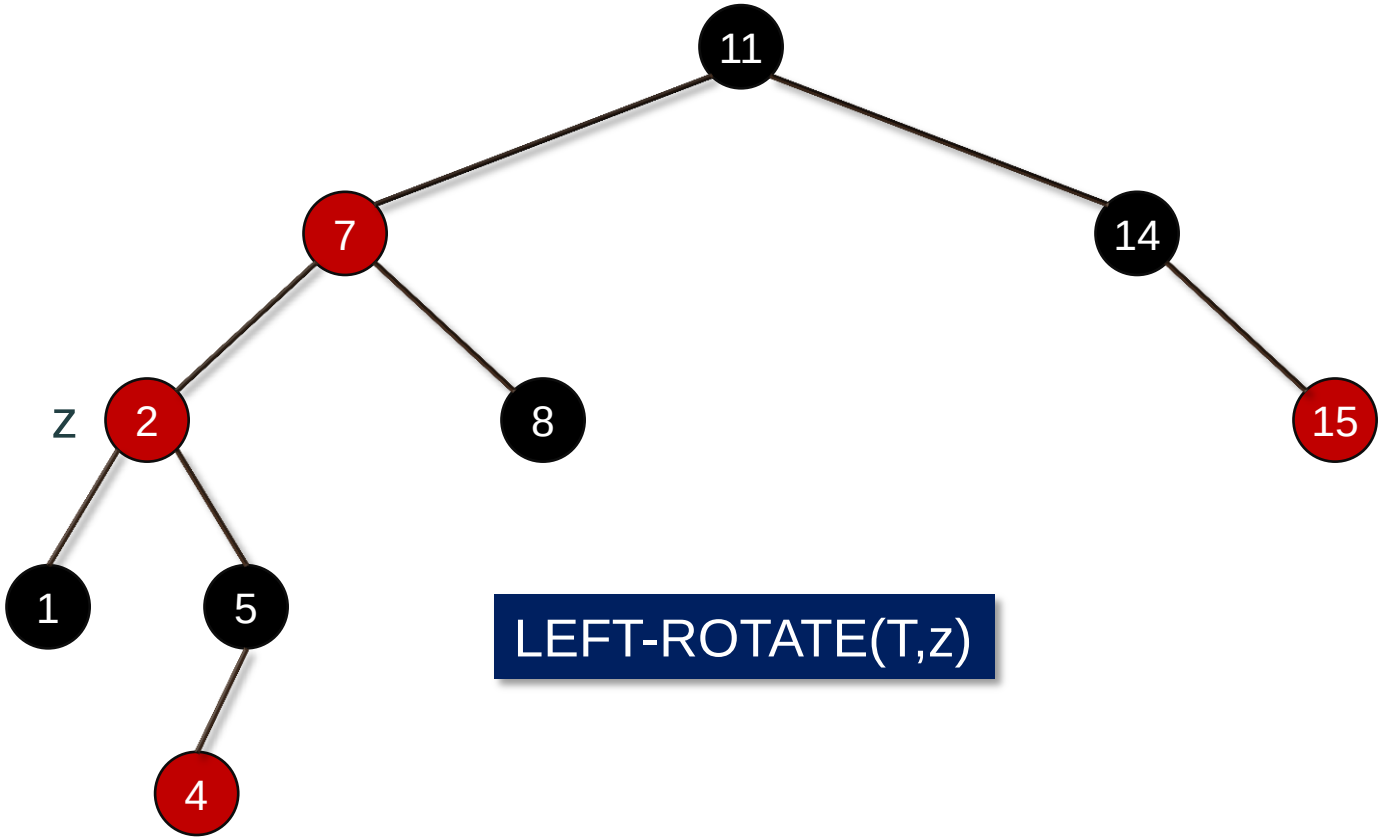
Inserção



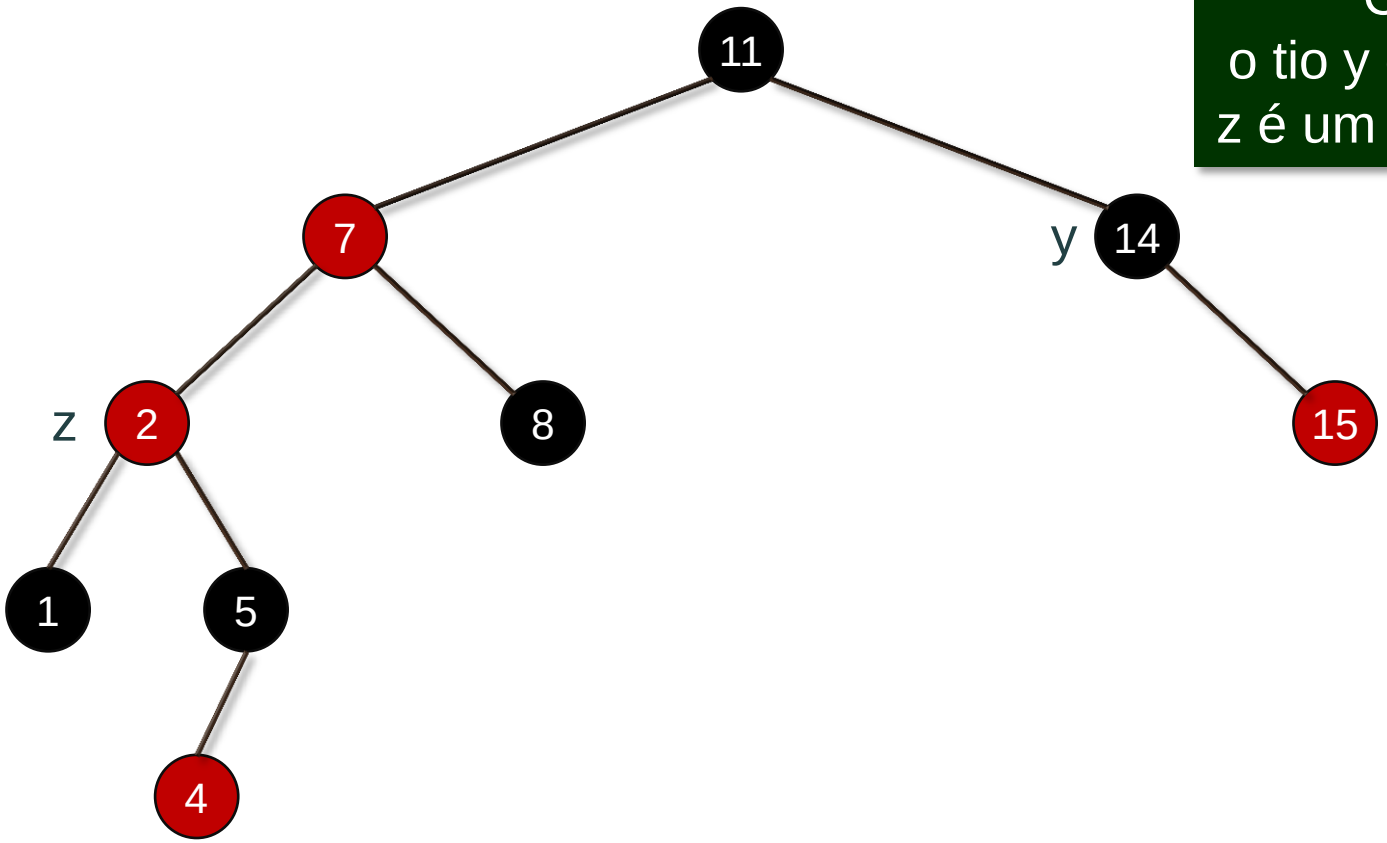
Inserção



Inserção

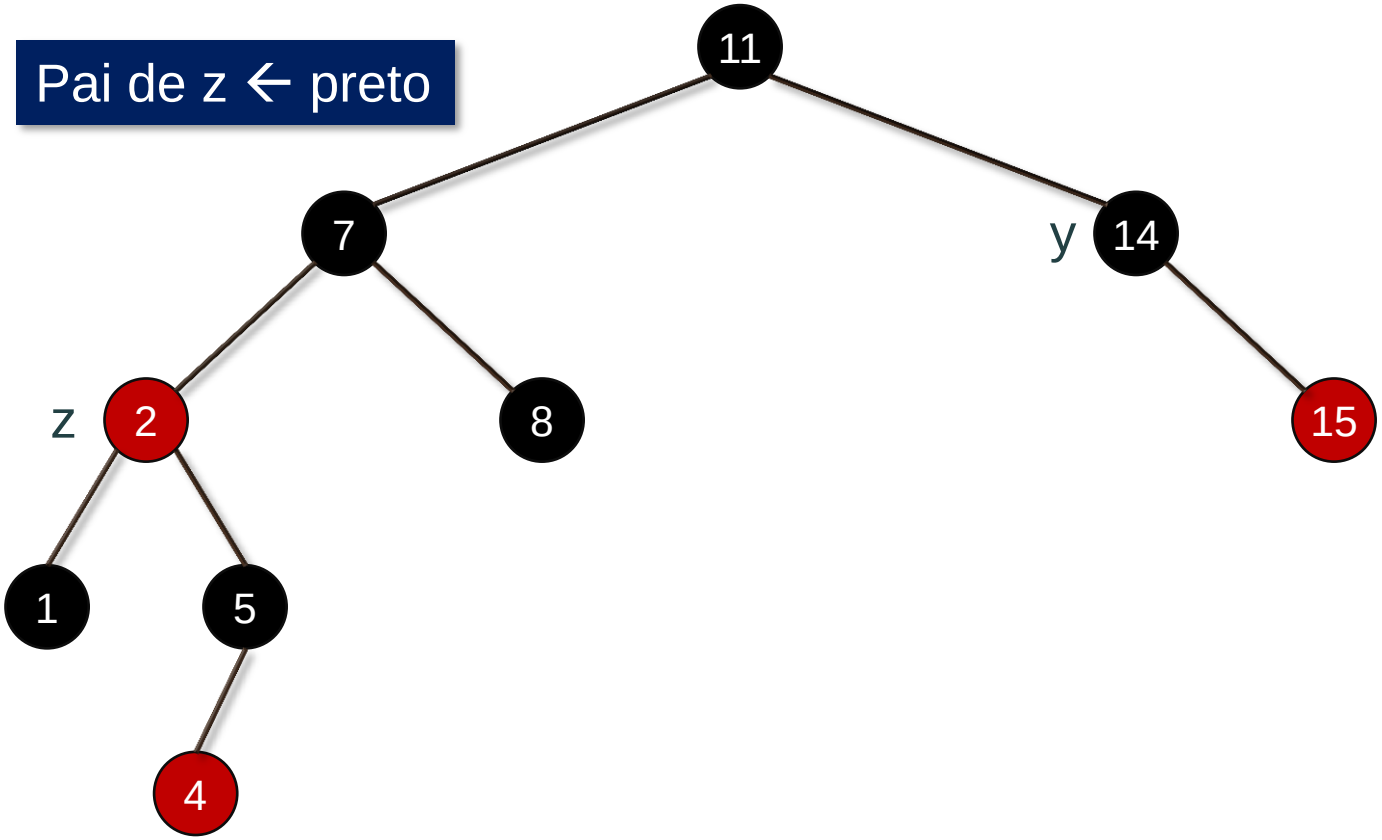


Inserção

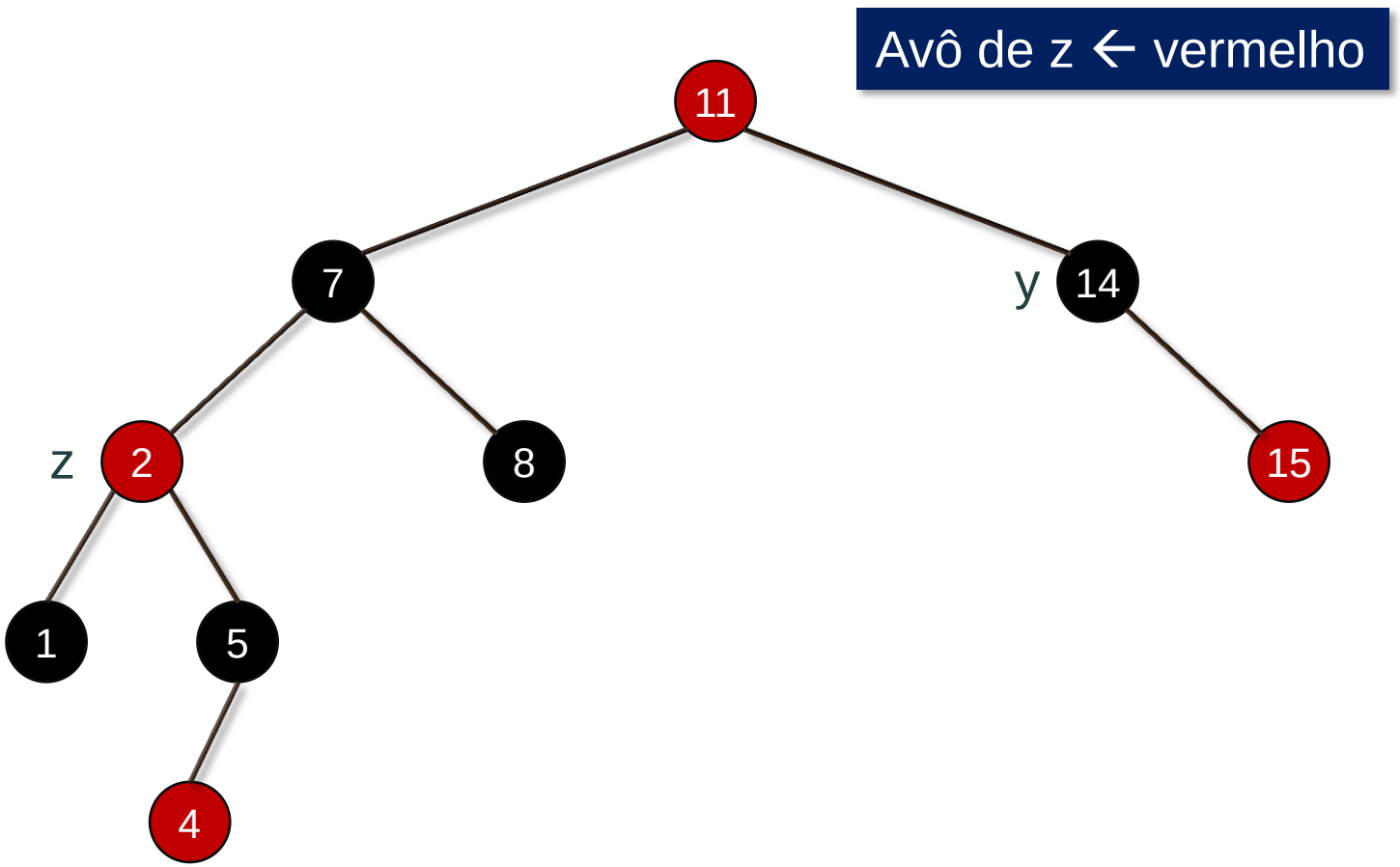


Caso 3
o tio y de z é preto
z é um filho da esq.

Inserção

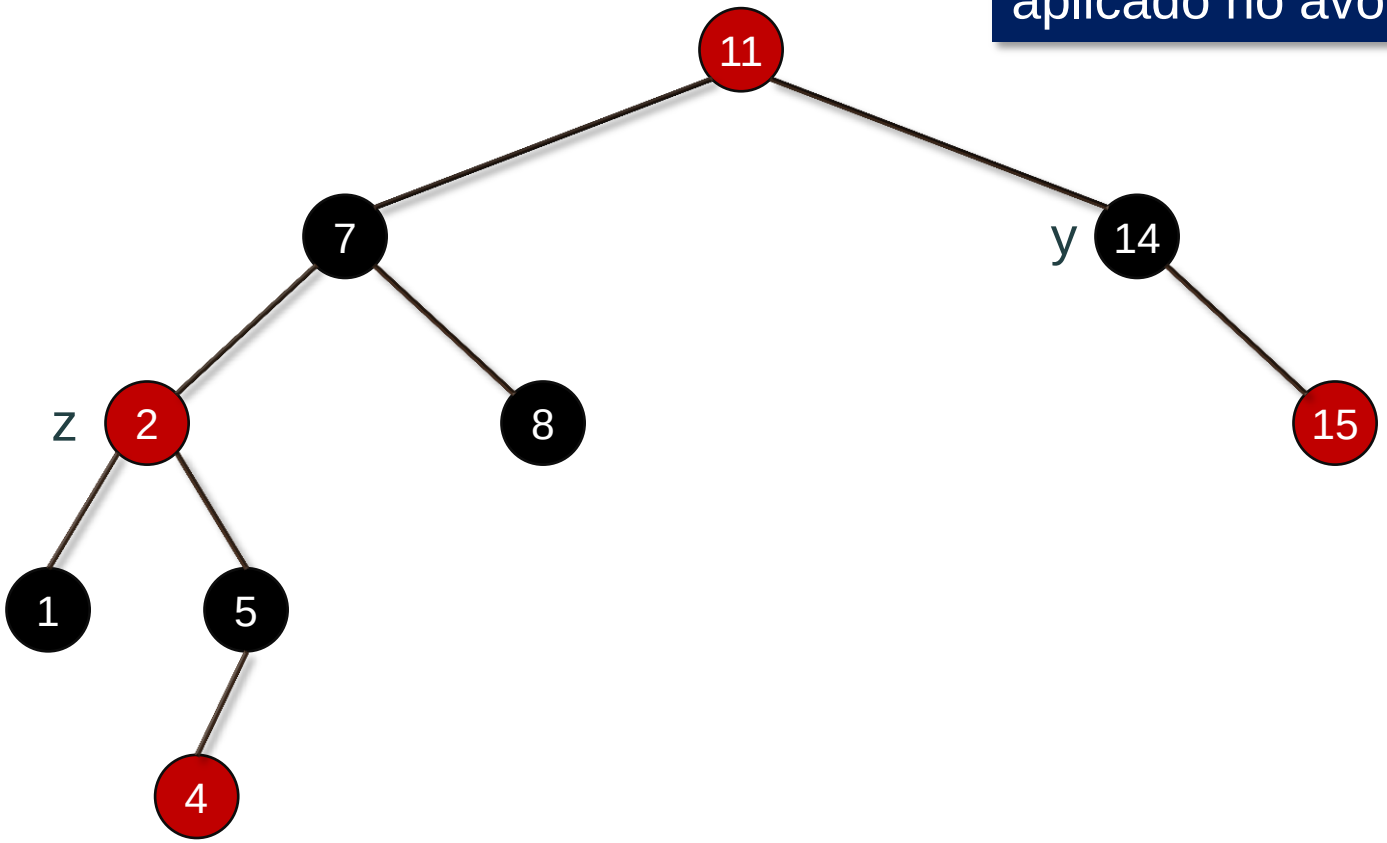


Inserção

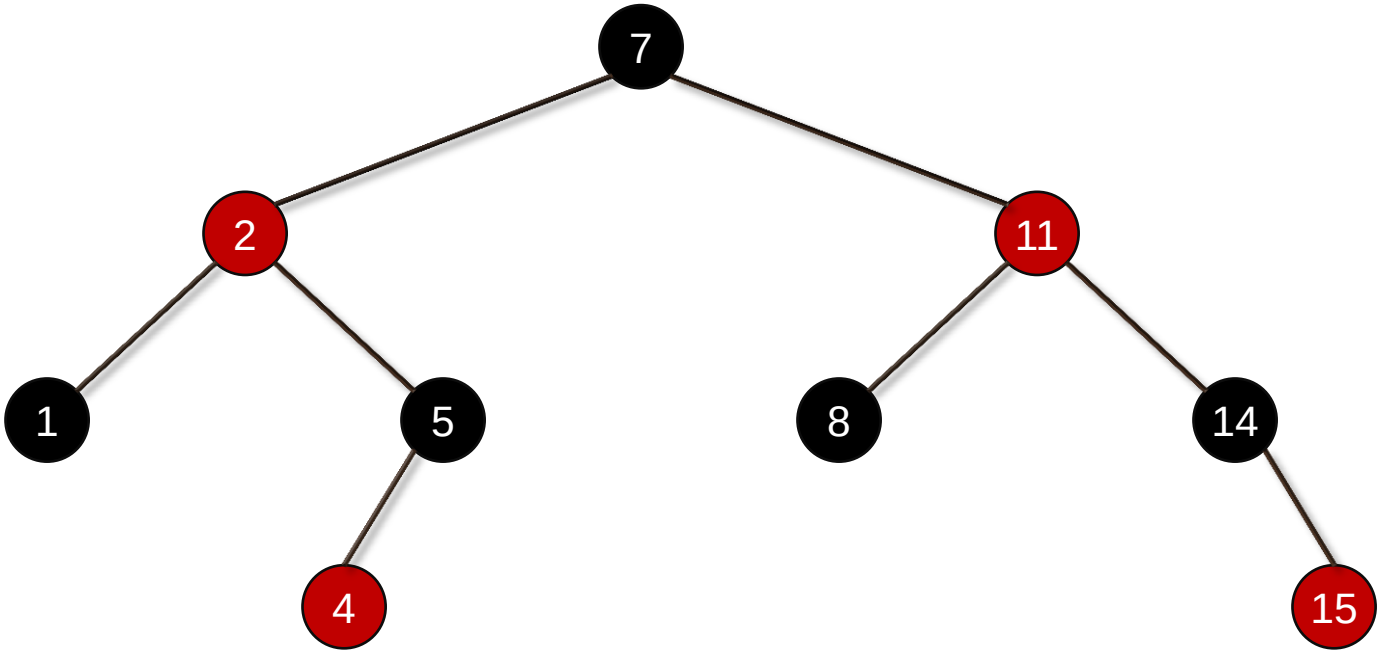


Inserção

RIGHT-ROTATE será aplicado no avô de z



Inserção



RIGHT-ROTATE(T,(z.pai).pai))

Remoção

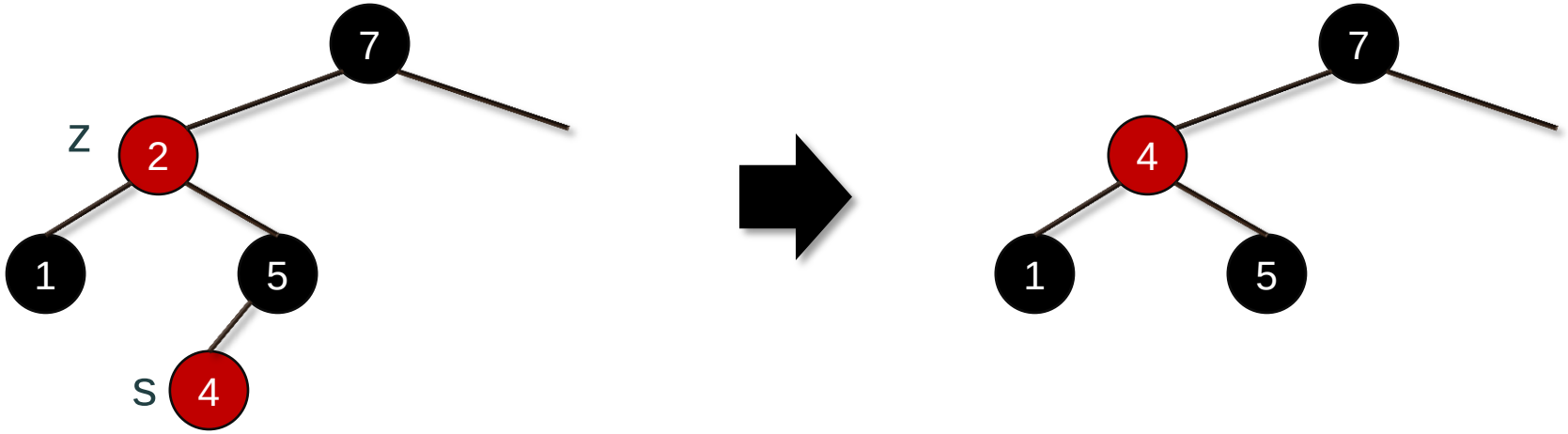
- É um pouco mais complexa do que a inserção
- Executa a remoção simples da árvore de pesquisa binária e resolve inconsistências
- Passos:
  - Encontre o nó v a ser removido
  - Remova o nó v da mesma forma que em uma árvore binária de pesquisa
  - Ajuste os critérios da árvore rubro-negra

Possibilidades

possibilidade	z	sucessor
1	vermelho	vermelho
2	preto	vermelho
3	preto	preto
4	vermelho	preto

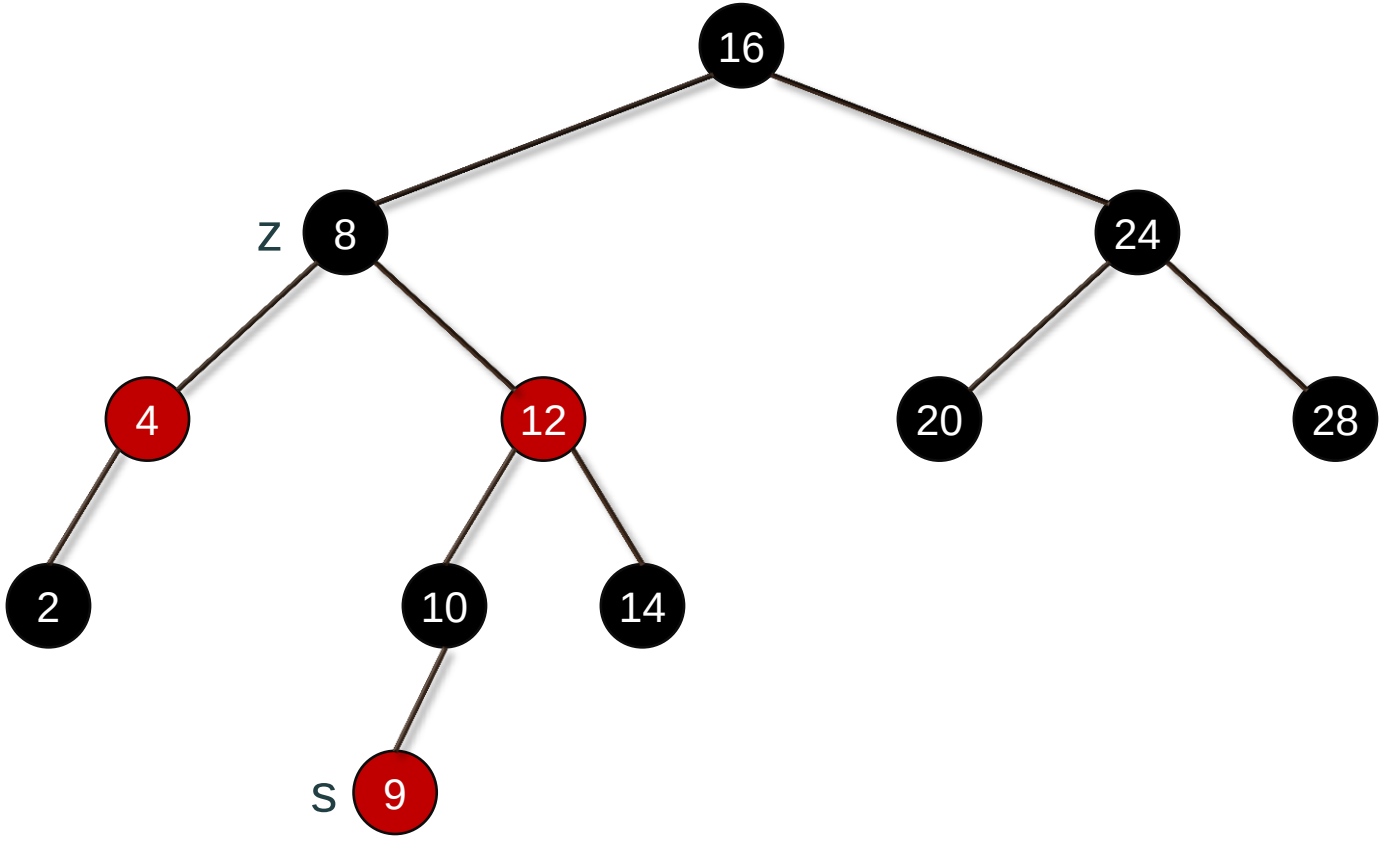
Possibilidade 1

- z é vermelho e seu sucessor também
  - Remoção simples



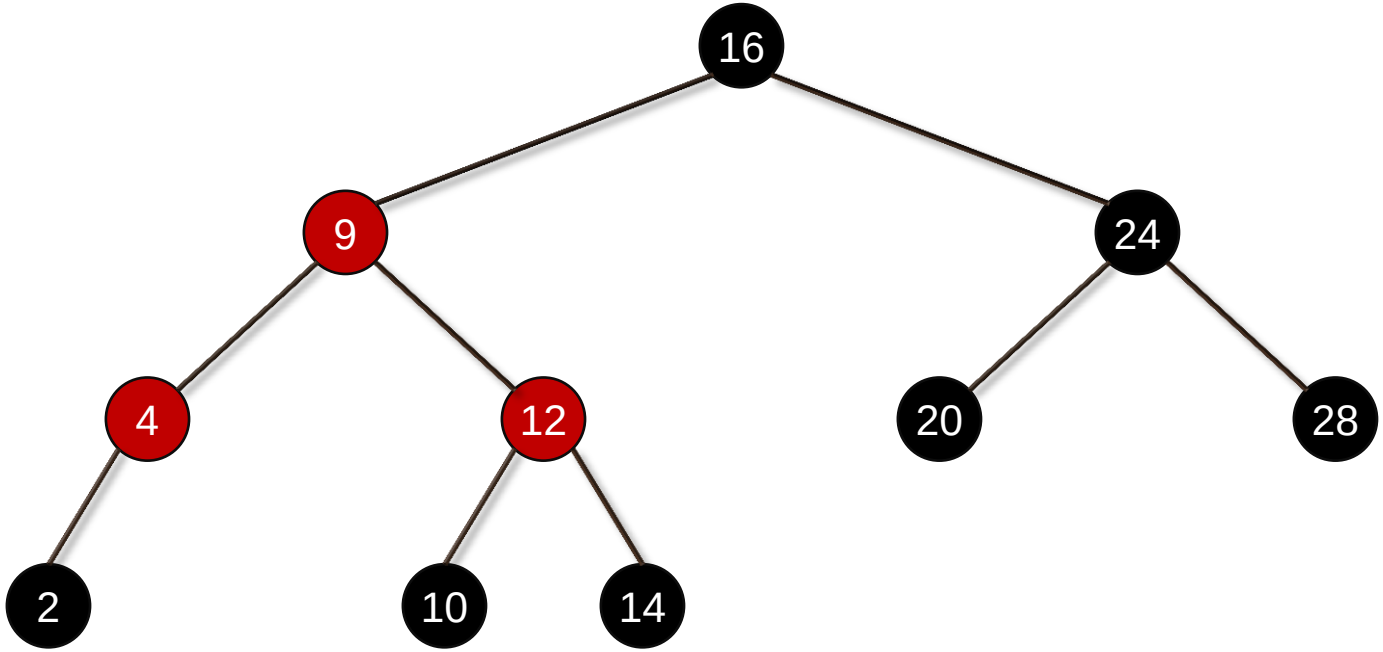
Possibilidade 2

- z é preto e seu sucessor é vermelho



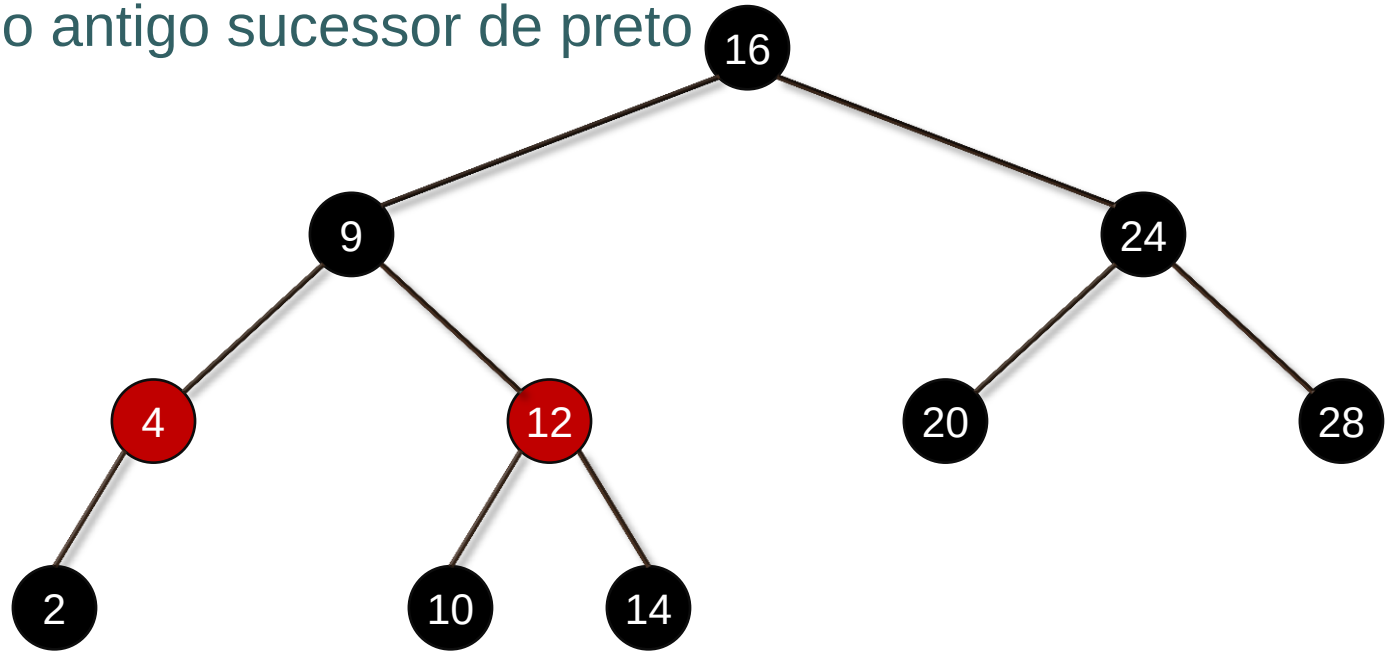
Possibilidade 2

- z é preto e seu sucessor é vermelho
  - Remove z



Possibilidade 2

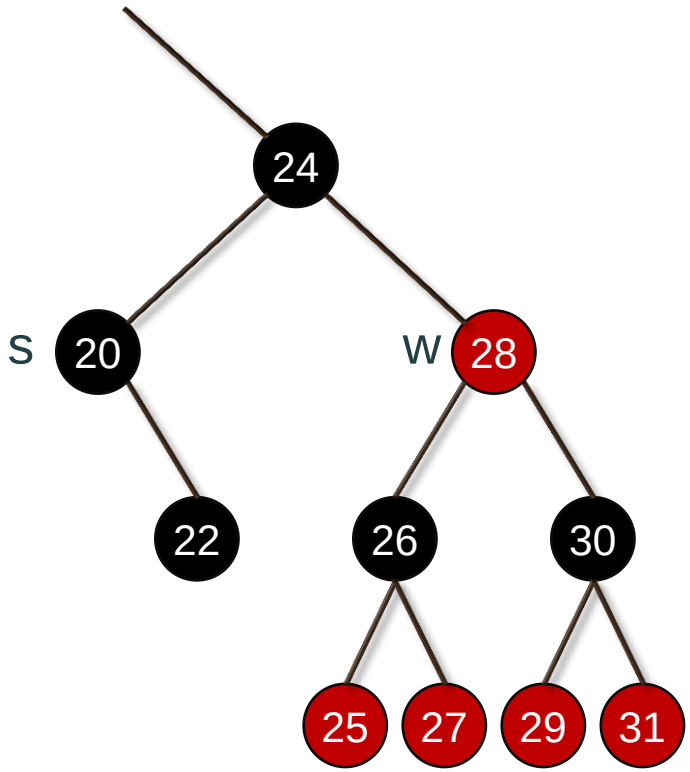
- z é preto e seu sucessor é vermelho
  - Remove z
  - Pinta o antigo sucessor de preto



Possibilidade 3

- z e seu sucessor s são pretos
- Caso 1
  - s é preto
  - O irmão w de s é vermelho
  - O pai de s é preto
- Procedimento
  - Faça uma rotação a esquerda sobre pai de s
  - Pinte w de preto
  - Pinte o pai de s de vermelho
  - Remova z

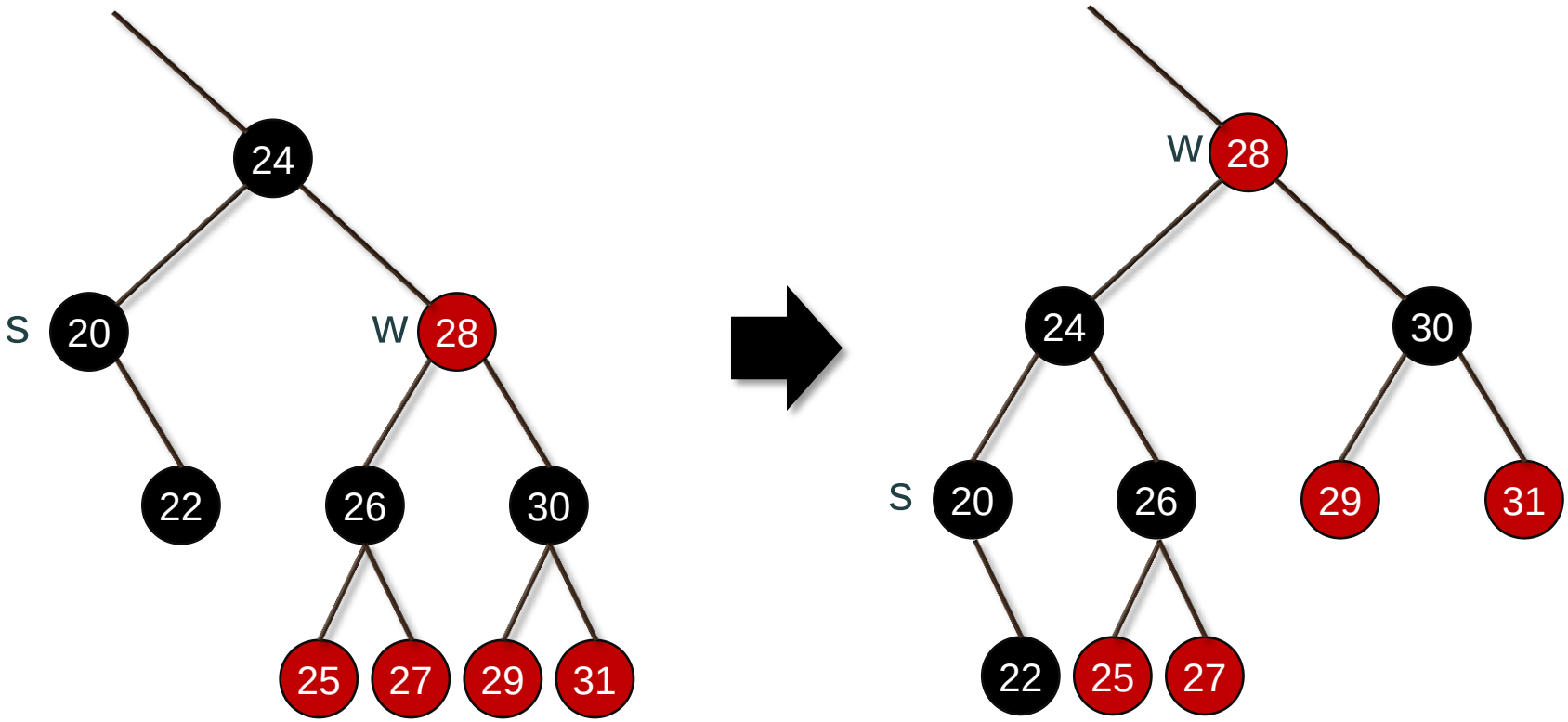
Possibilidade 3



- Caso 1
  - s é preto
  - O irmão w de s é vermelho
  - O pai de s é preto

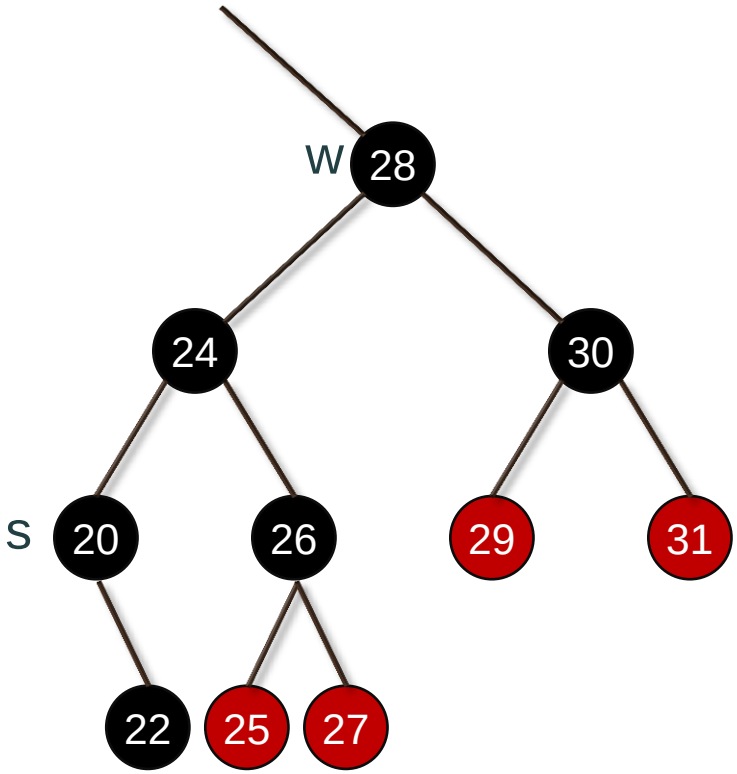
Possibilidade 3

- Rotação a esquerda sobre pai de s



Possibilidade 3

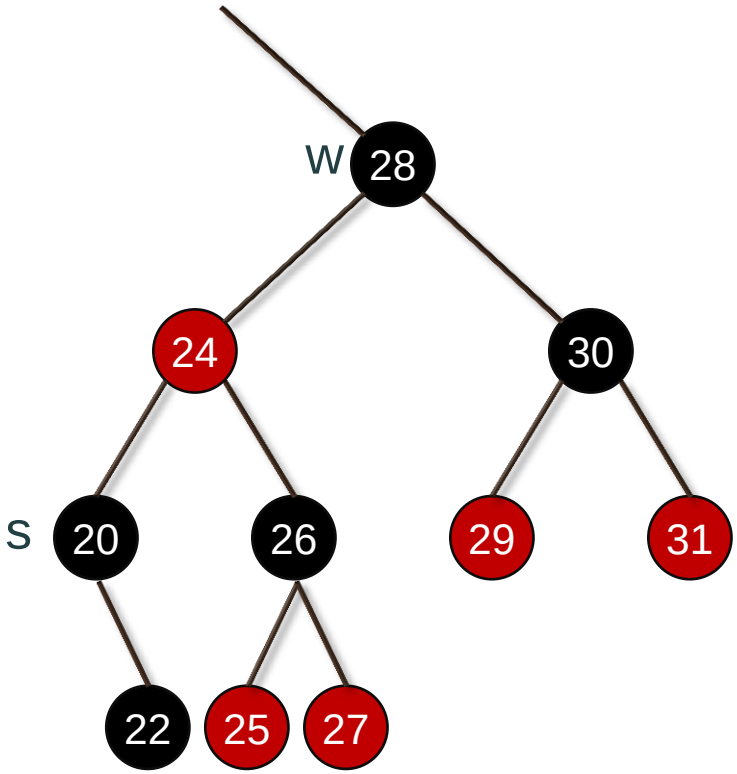
- Rotação a esquerda sobre pai de s
- Pinta w de preto



Possibilidade 3

- Rotação a esquerda sobre pai de s
- Pinta w de preto
- Pinta o pai de s de vermelho

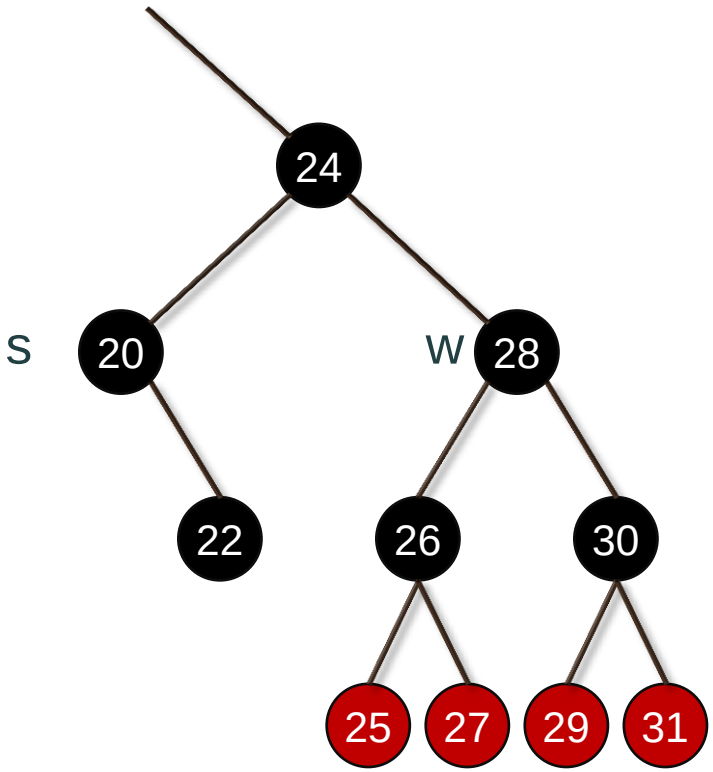
Agora z pode ser removido



Possibilidade 3

- z e seu sucessor s são pretos
- Caso 2a
  - s é preto
  - O irmão w de s é preto
  - Os filhos de w são pretos
  - O pai de s é preto
- Procedimento
  - Pinte w de vermelho
  - Remova z

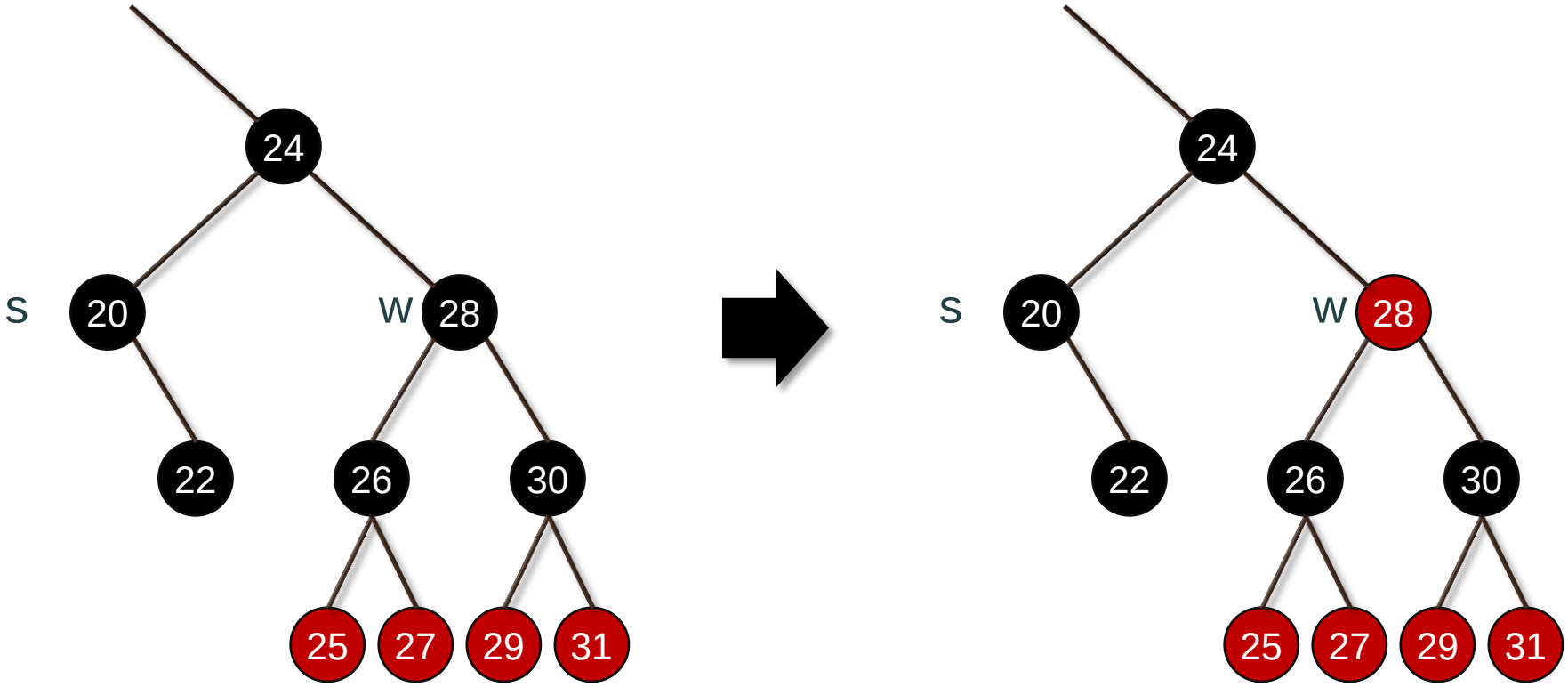
Possibilidade 3



- Caso 2a
  - s é preto
  - O irmão w de s é preto
  - Os filhos de w são pretos
  - O pai de s é preto

Possibilidade 3

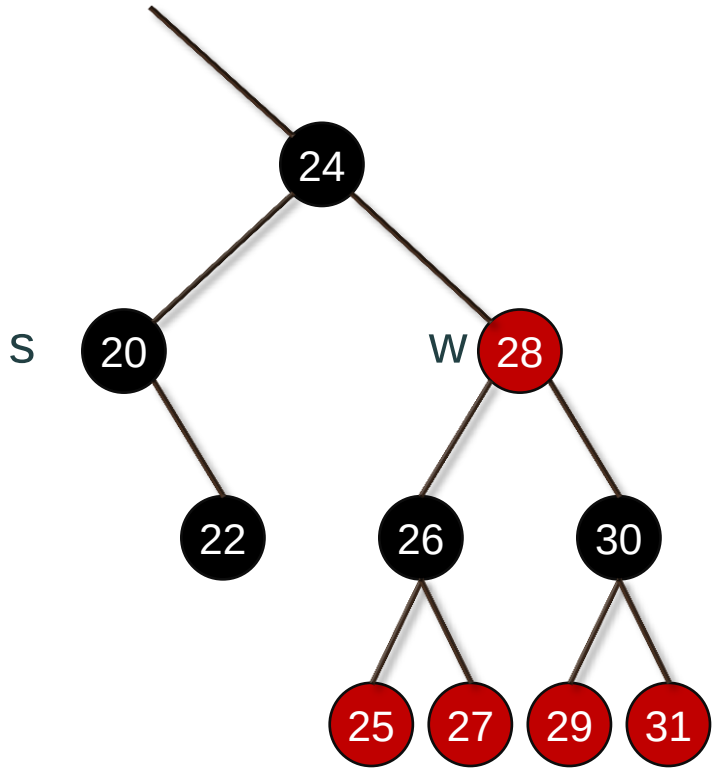
- Pinte w de vermelho



Possibilidade 3

- Pinte w de vermelho

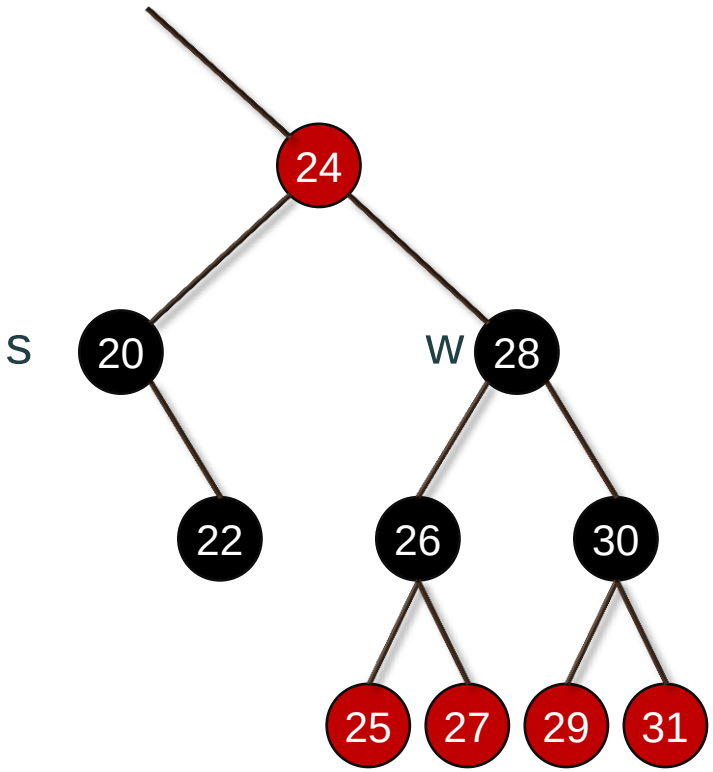
Agora z pode ser removido



Possibilidade 3

- z e seu sucessor s são pretos
- Caso 2b
  - s é preto
  - O irmão w de s é preto
  - Os filhos de w são pretos
  - O pai de s é vermelho
- Procedimento
  - Pinte w de vermelho
  - Pinte o pai de s de preto
  - Remova z

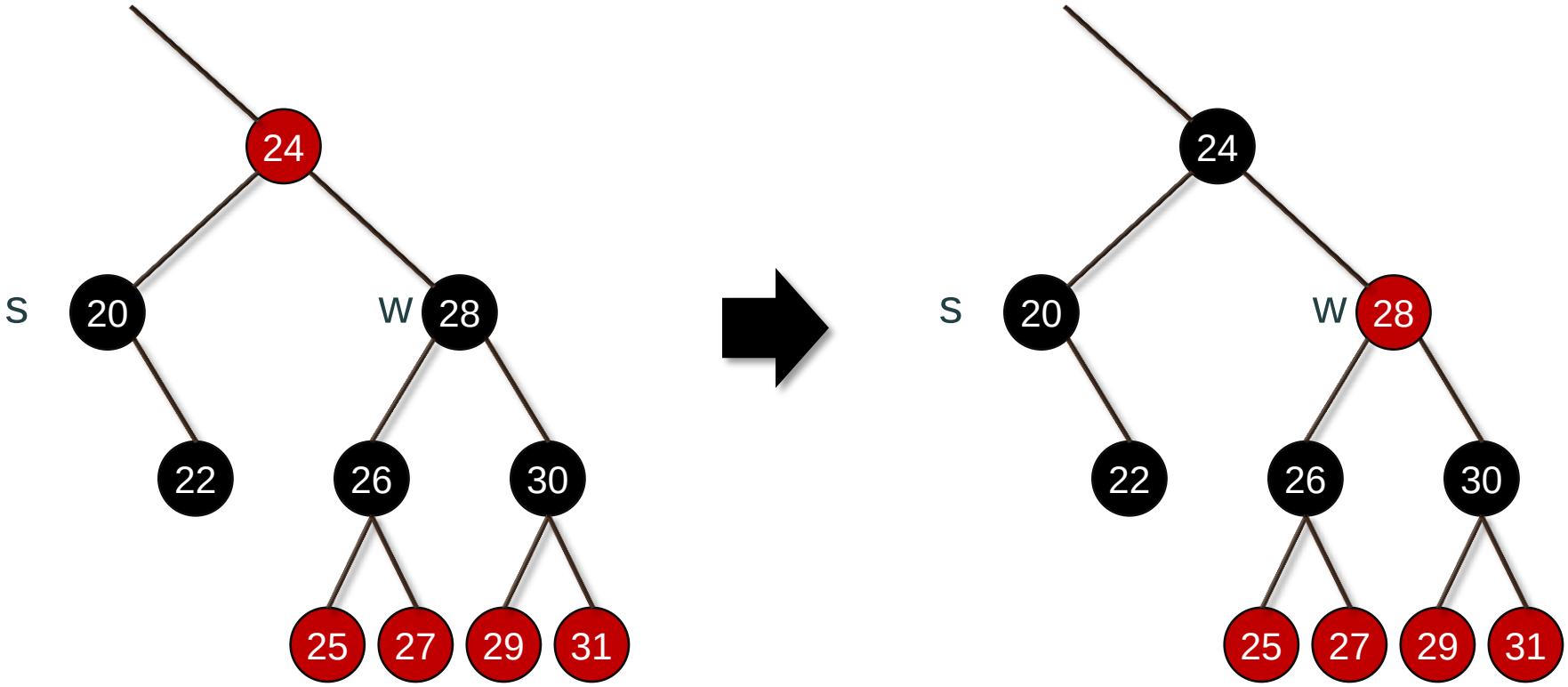
Possibilidade 3



- Caso 2b
  - s é preto
  - O irmão w de s é preto
  - Os filhos de w são pretos
  - O pai de s é vermelho

Possibilidade 3

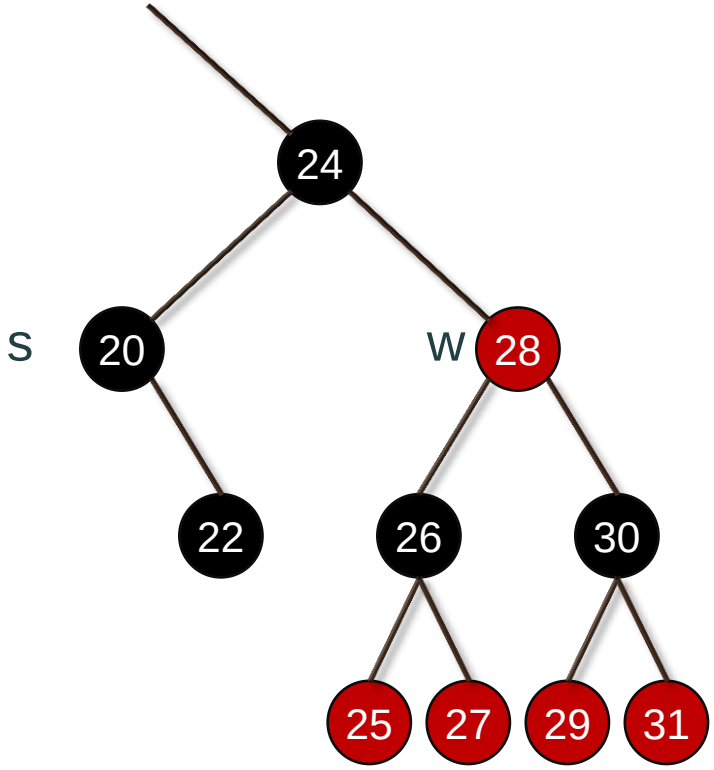
- Pinte w de vermelho e o pai de s de preto



Possibilidade 3

- Pinte w de vermelho e o pai de s de preto

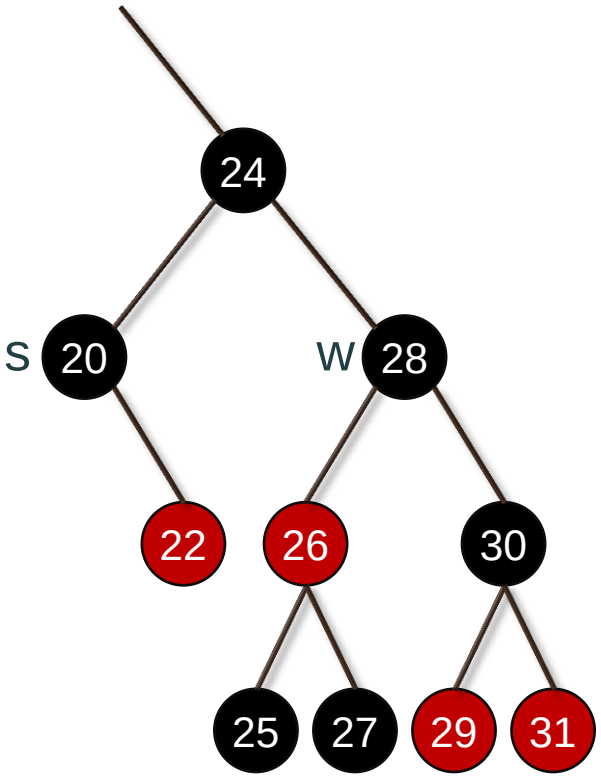
Agora z pode ser removido



Possibilidade 3

- z e seu sucessor s são pretos
- Caso 3
  - s é preto
  - O irmão w de s é preto
  - O filho esquerdo de w é vermelho e o direto preto
  - O pai de s é vermelho ou preto
- Procedimento
  - Trocar as cores de w com seu filho esquerdo
  - Rotação a direita sobre w
  - Remova z

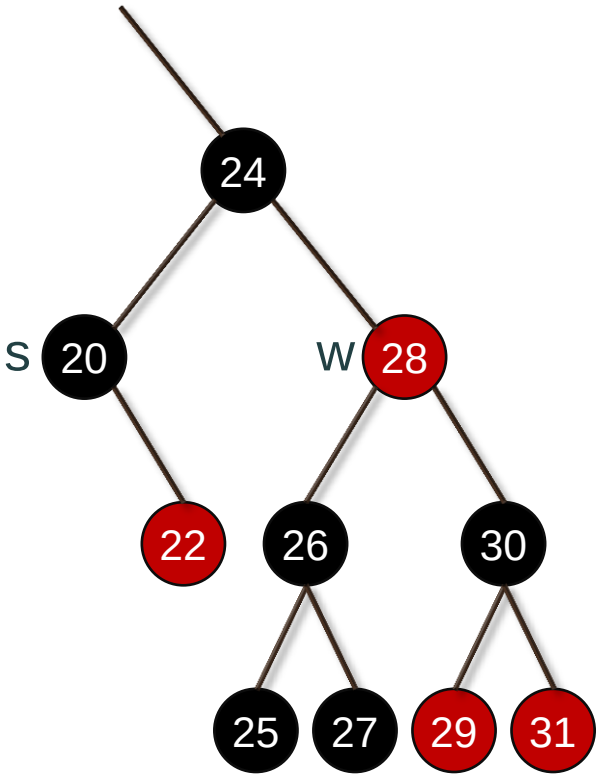
Possibilidade 3



- Caso 3
  - s é preto
  - O irmão w de s é preto
  - O filho esquerdo de w é vermelho e o direto preto
  - O pai de s é vermelho ou preto

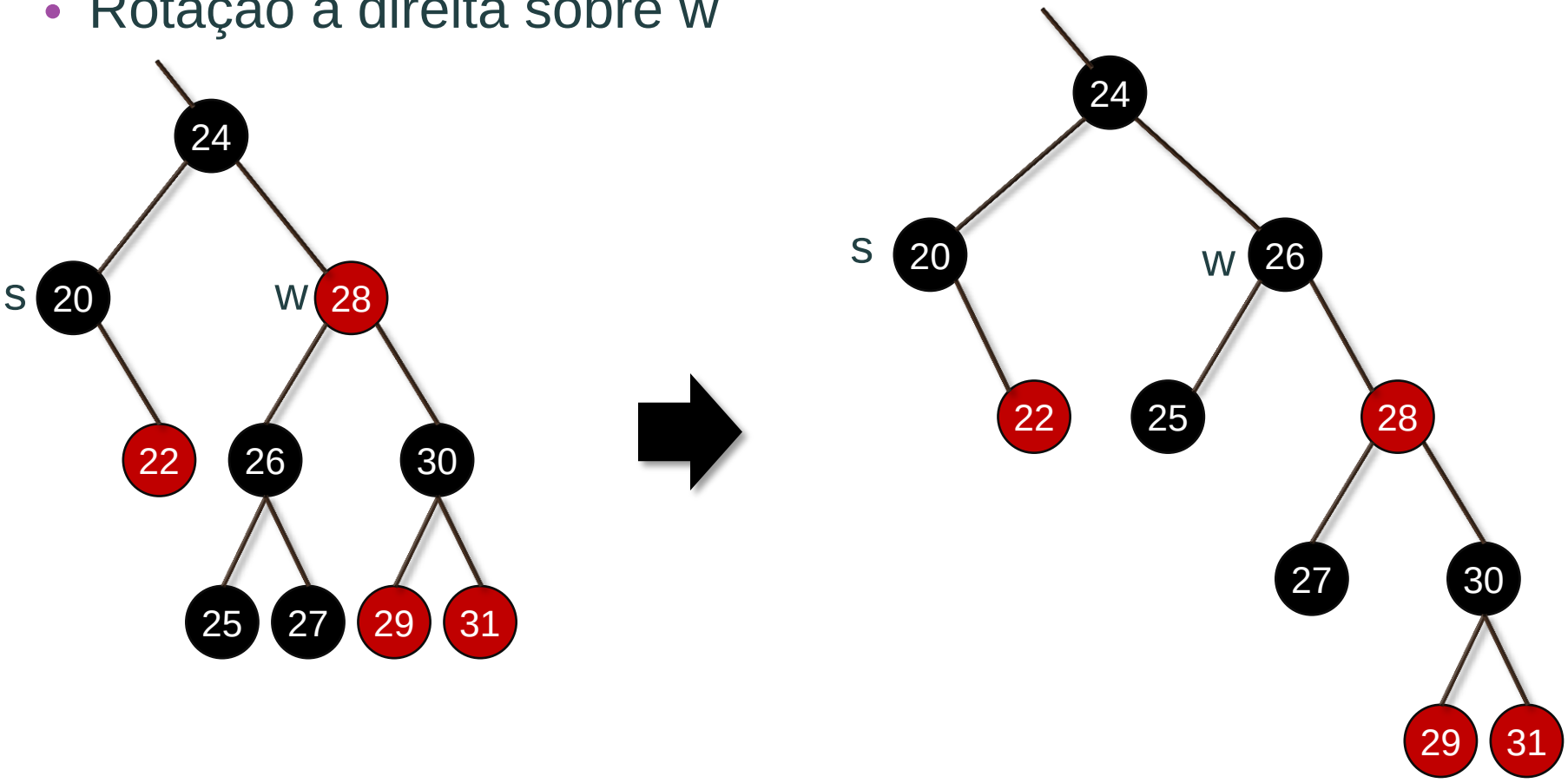
Possibilidade 3

- Troca a cor de w com o filho da esquerda



Possibilidade 3

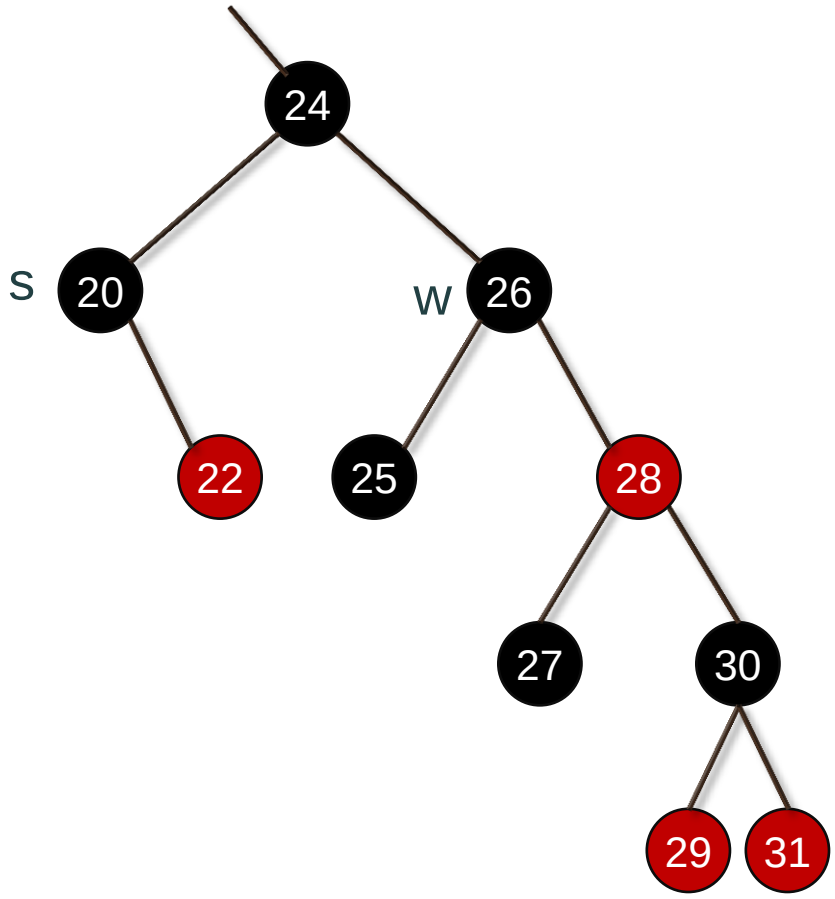
- Troca a cor de w com o filho da esquerda
- Rotação a direita sobre w



Possibilidade 3

- Troca a cor de w com o filho da esquerda
- Rotação a direita sobre w

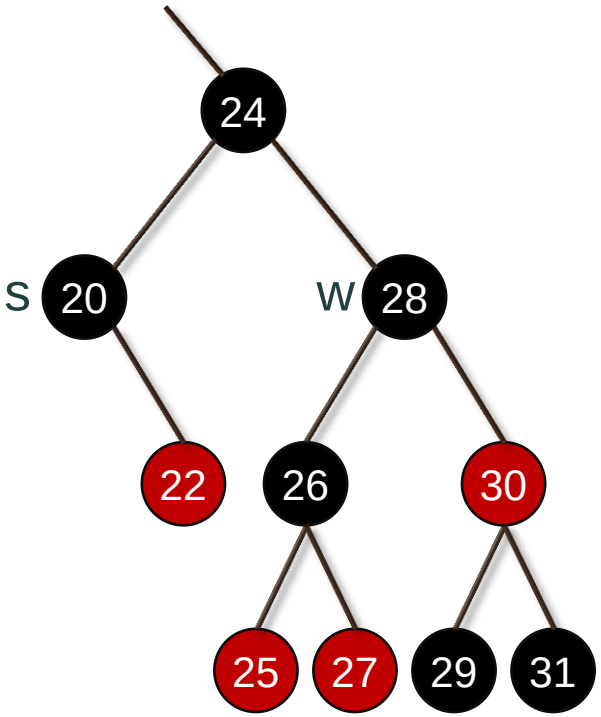
Agora z pode ser removido



Possibilidade 3

- z e seu sucessor s são pretos
- Caso 4
  - s é preto
  - O irmão w de s é preto
  - O filho esquerdo de w é de qualquer cor
  - O filho direito de w é vermelho
  - O pai de s é preto
- Procedimento
  - Rotação a esquerda sobre pai de s
  - Pinte o pai de preto
  - Pinte w com a cor anterior do pai
  - Pinte o filho direito de w de preto
  - Remova z

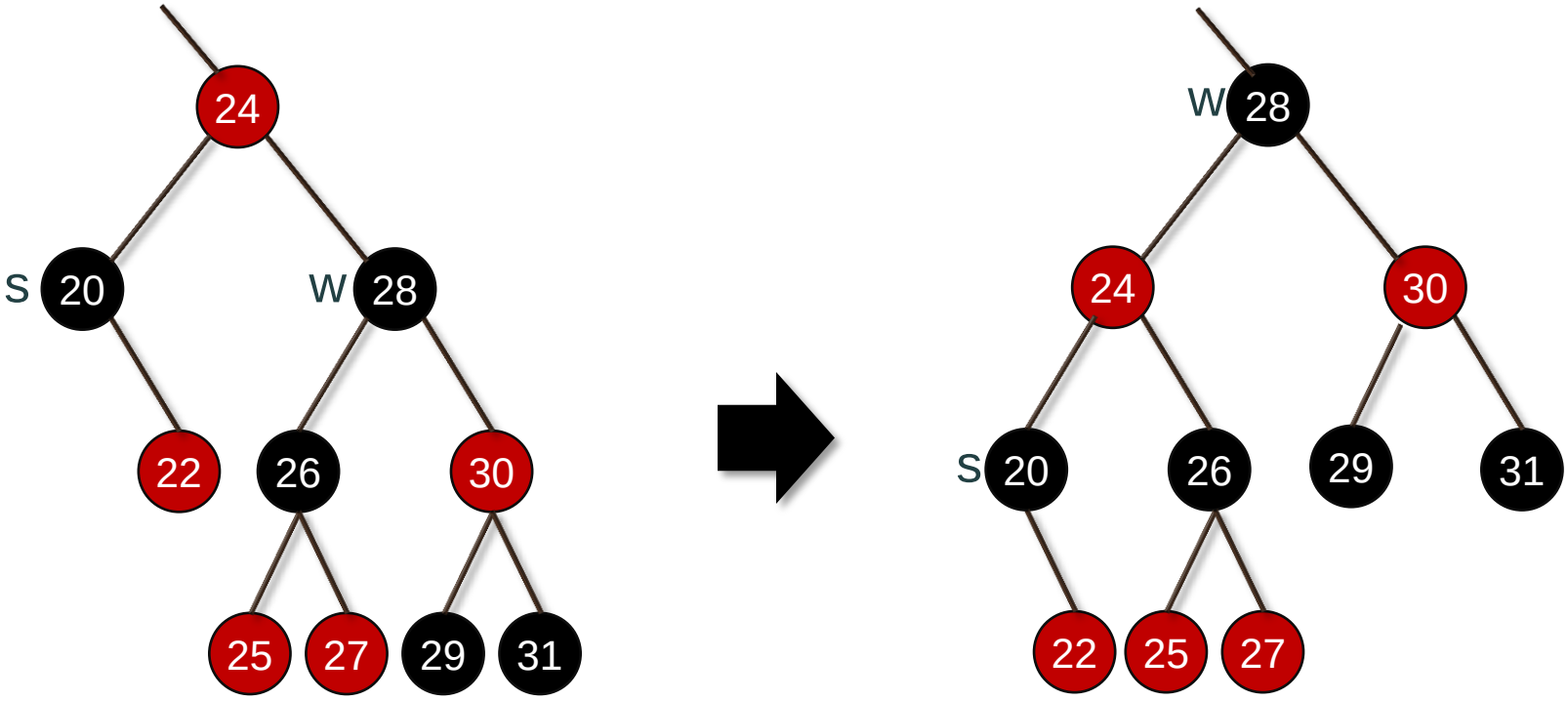
Possibilidade 3



- Caso 4
  - s é preto
  - O irmão w de s é preto
  - O filho esquerdo de w é de qualquer cor
  - O filho direito de w é vermelho
  - O pai de s é preto

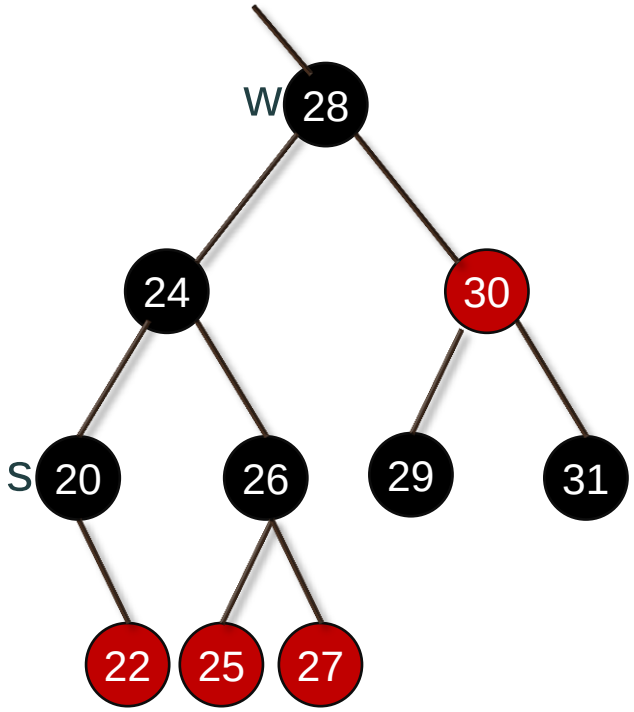
Possibilidade 3

- Rotação a esquerda sobre pai de s



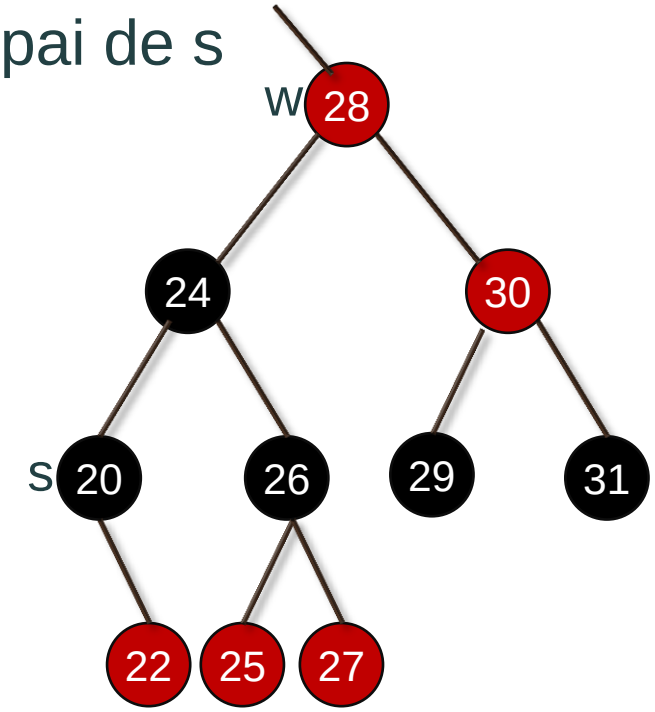
Possibilidade 3

- Rotação a esquerda sobre pai de s
- Pinte o pai de s de preto



Possibilidade 3

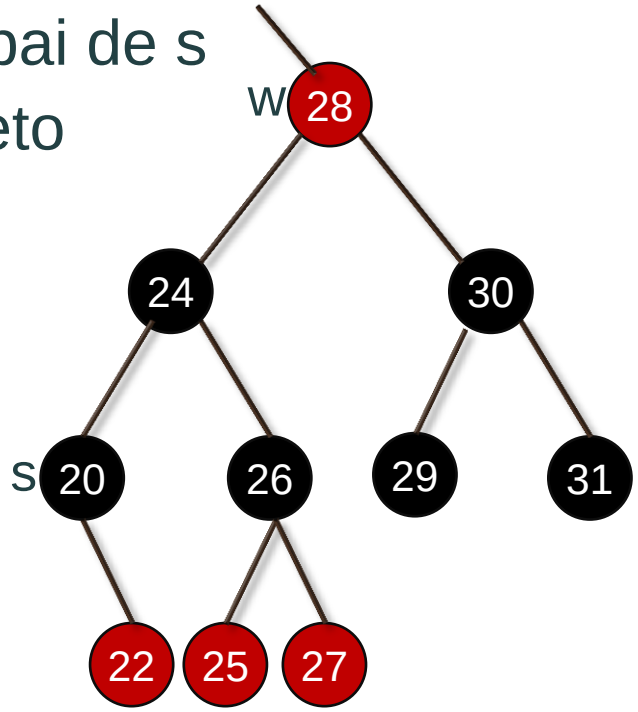
- Rotação a esquerda sobre pai de s
- Pinte o pai (vermelho) de s de preto
- Pinte w com a cor anterior do pai de s



Possibilidade 3

- Rotação a esquerda sobre pai de s
- Pinte o pai (vermelho) de s de preto
- Pinte w com a cor anterior do pai de s
- Pinte o filho direito de w de preto

Agora z pode ser removido



Possibilidade 4

- z é vermelho e seu sucessor s é preto
- Similar à possibilidade 3
- Procedimento
  - Pinte s de vermelho
  - Proceda como na Possibilidade 3