CC BY

“Você poderia me dizer que caminho devo seguir?”
“Isso depende muito de onde você quer chegar” – disse o Gato.
“Eu não me importo pra onde” – disse Alice.
“Então qualquer caminho serve” (Alice no país das Maravilhas).

Árvores
Red-Black

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR

Árvores de busca binária

Uma árvore de busca binária de altura h suporta as operações buscar, excluir, mínimo, ... com um custo $O(h)$.

Se $h \to n$, então uma árvore não é melhor do que uma lista encadeada.

Uma árvore Red-Black (rubro-negra) mantém a árvore balanceada para que $h \approx log_2 n$.

Logo, as operações são $O(h) = O(log_2 n)$ no pior caso.

Red-Black

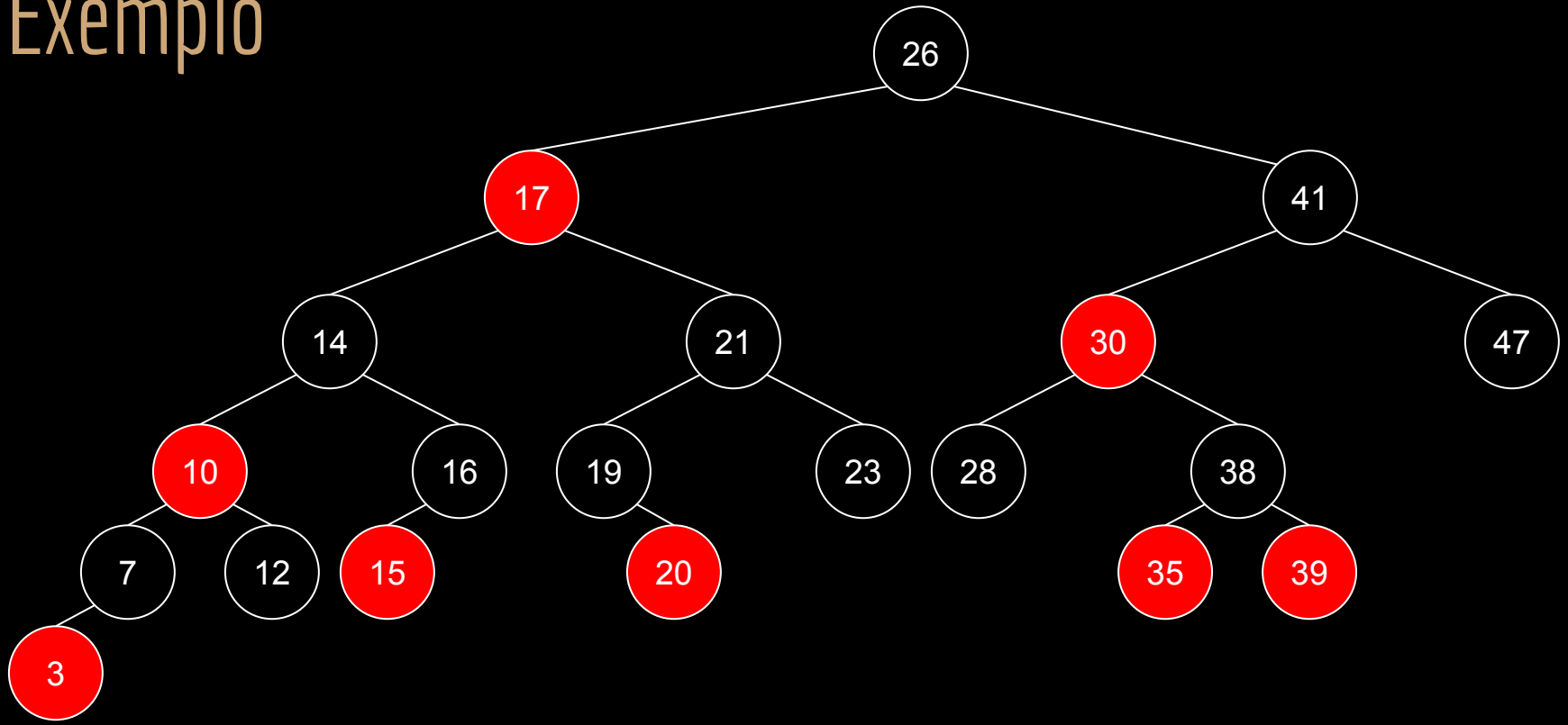
Árvore de busca binária com um bit extra por nodo.

Bit representa a cor do nodo: vermelho (red) ou preto (black).

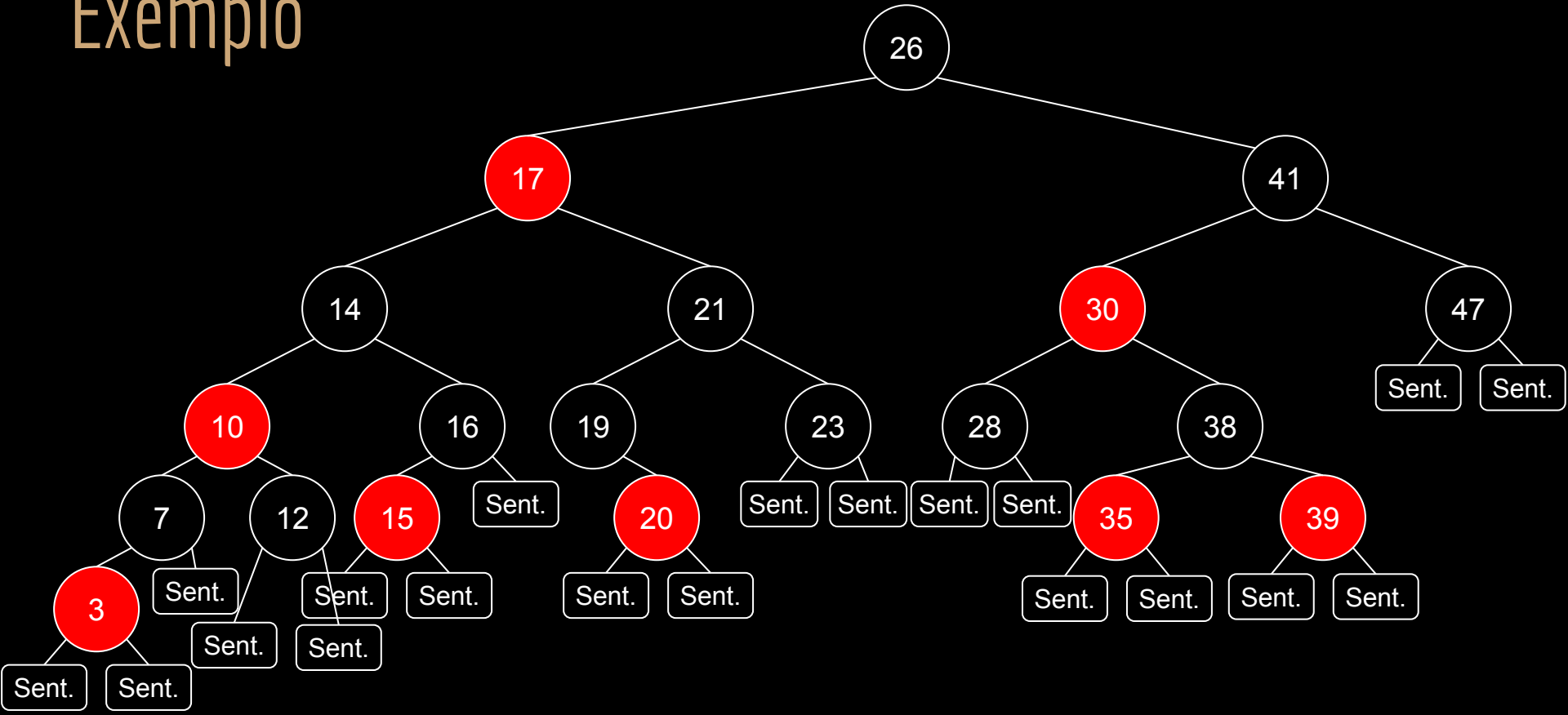
Propriedades

1. Todo nodo é vermelho ou preto;
2. A raiz é preta;
3. Toda folha é um **sentinela** preto;
4. Se um nodo é vermelho, ambos filhos são pretos;
5. Todos caminhos simples (sem repetição de nodos) de um nodo qualquer até suas folhas descendentes têm o mesmo número de nodos pretos.

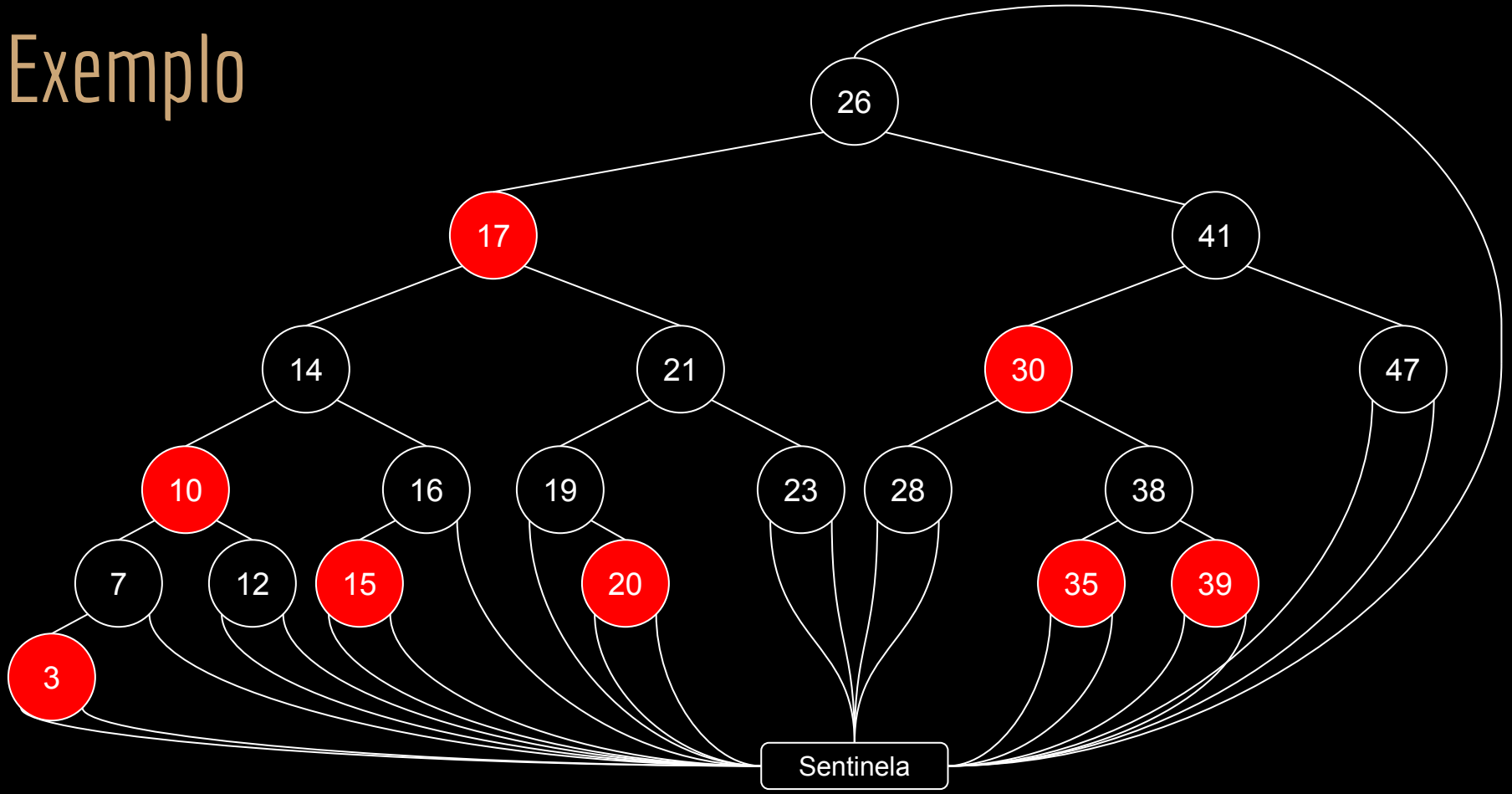
Exemplo



Exemplo

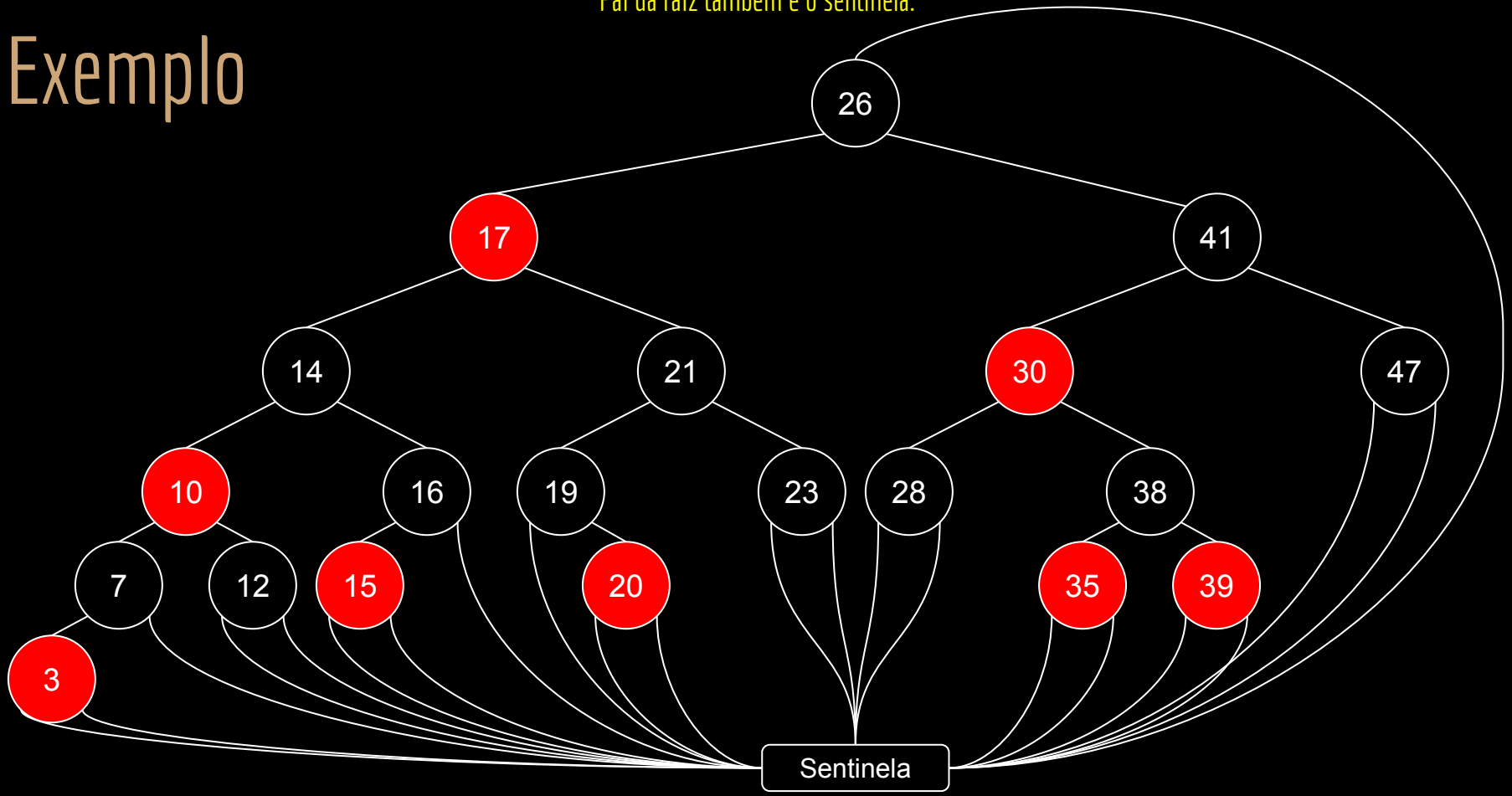

26
17
41
14
21
30
47
Sent.
Sent.
10
16
19
23
28
38
7
12
15
Sent.
20
Sent.
Sent.
Sent.
Sent.
35
39
3
Sent.
Sent.
Sent.
Sent.
Sent.
Sent.
Sent.
Sent.
Sent.
Sent.
Sent.
Sent.
Sent.
Sent.
Sent.
Sent.

Exemplo



Exemplo

Pai da raiz também é o sentinela.



Altura Preta

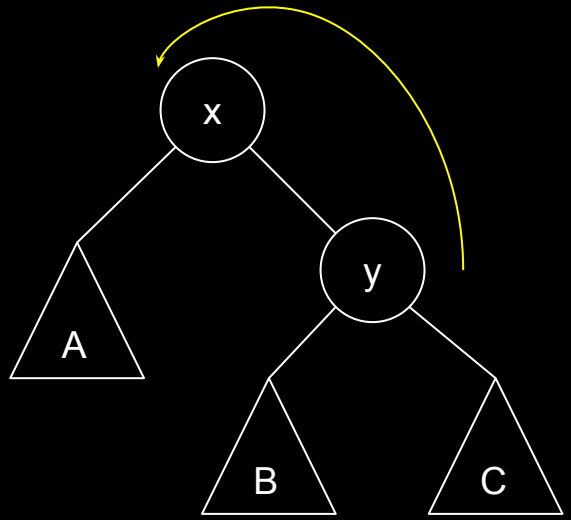
Considere um caminho simples descendente entre um nodo x qualquer e uma folha.

A altura preta, denotada por *bh(x)*, é o numero de nodos pretos entre *x* e a folha. O nodo x não é considerado no caminho.

Pela propriedade 5, a altura preta da árvore será a altura preta da raiz.

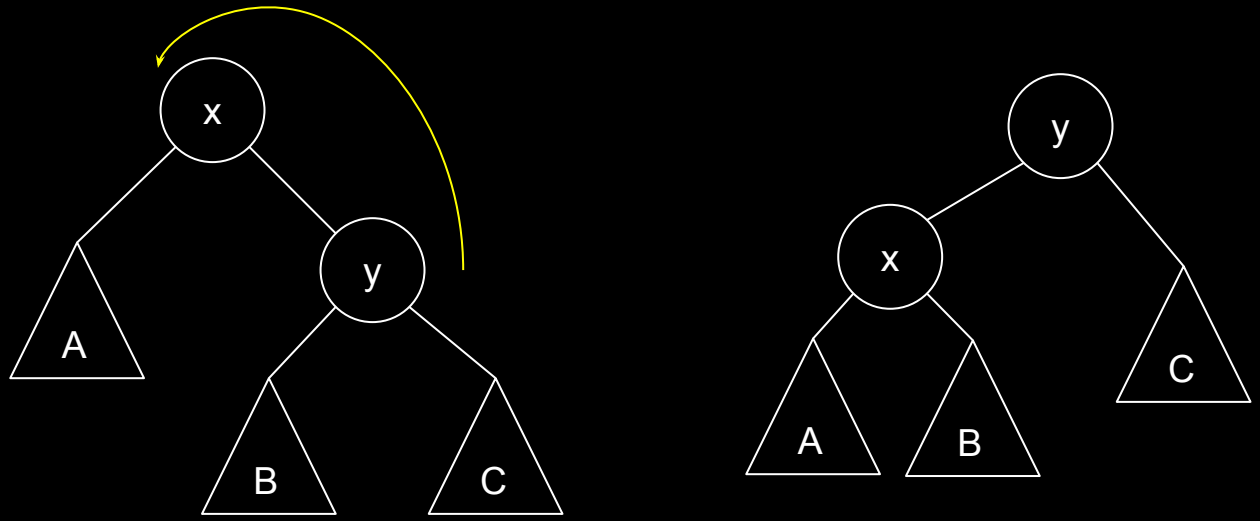
Rotações

Rotação para esquerda.



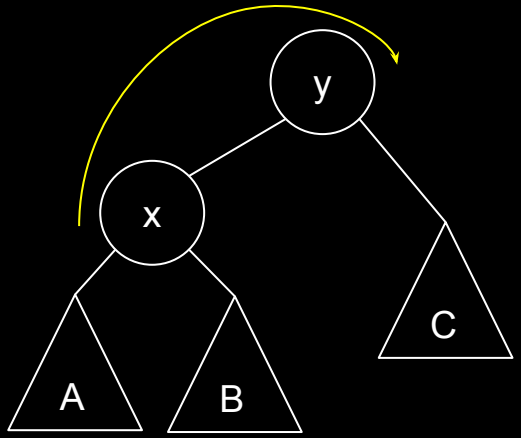
Rotações

Rotação para esquerda.



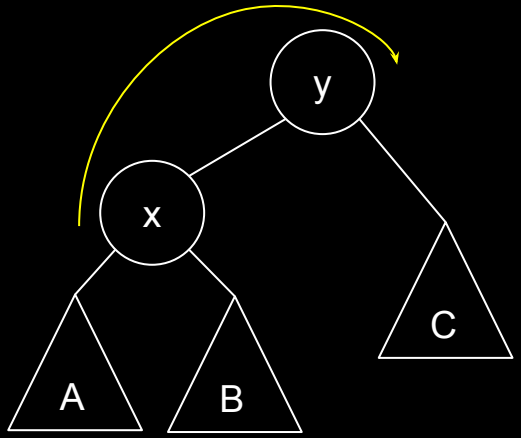
Rotações

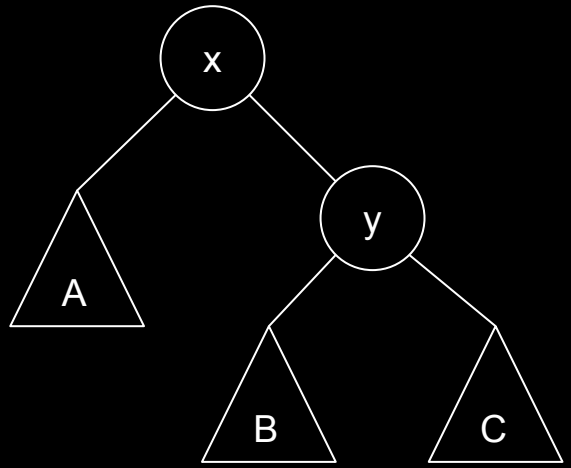
Rotação para direita.



Rotações

Rotação para direita.





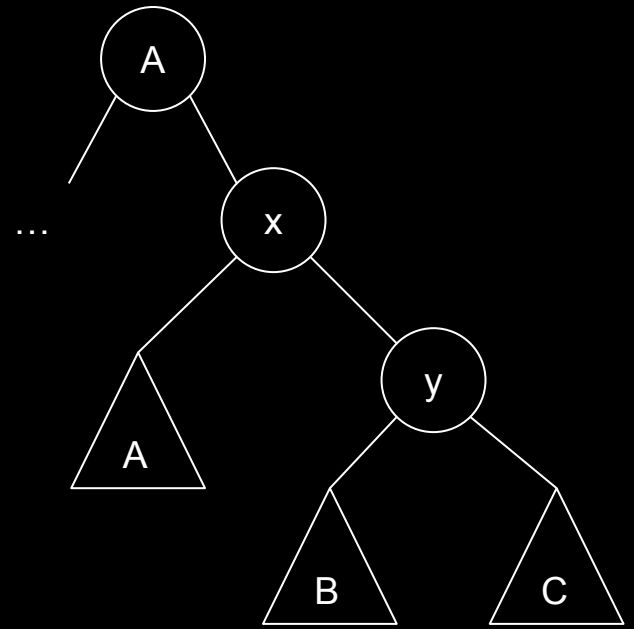
Rotação Esquerda

```
função rotacaoEsquerda(T,x)
entrada: nodo x a ser rotacionado e a árvore T
saída: o nodo x é rotacionado a esquerda.

y = x.fd
x.fd = y.fe
se y.fe ≠ sentinela
     y.fe.pai = x
y.p = x.p
se x.p == sentinela
     T.raiz = y
senão
     se x == x.p.fe
          x.p.fe = y
     senão
          x.p.fd = y
y.fe = x
x.p = y
```

Teste de Mesa

```
função rotacaoEsquerda(T,x)
entrada: nodo x a ser rotacionado e a árvore T
saída: o nodo x é rotacionado a esquerda.

y = x.fd
x.fd = y.fe
se y.fe ≠ sentinela
     y.fe.pai = x
y.p = x.p
se x.p == sentinela
     T.raiz = y
senão
     se x == x.p.fe
          x.p.fe = y
     senão
          x.p.fd = y
y.fe = x
x.p = y
```

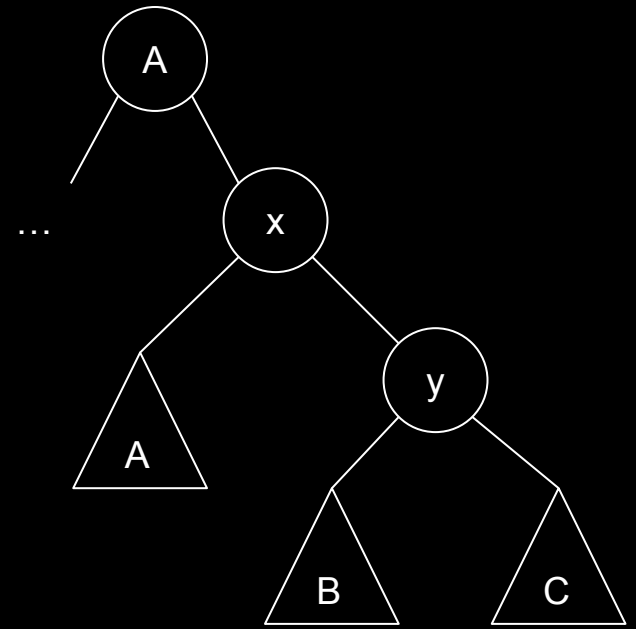
rotacaoEsquerda(T,x)



Teste de Mesa

rotacaoEsquerda(T,x)

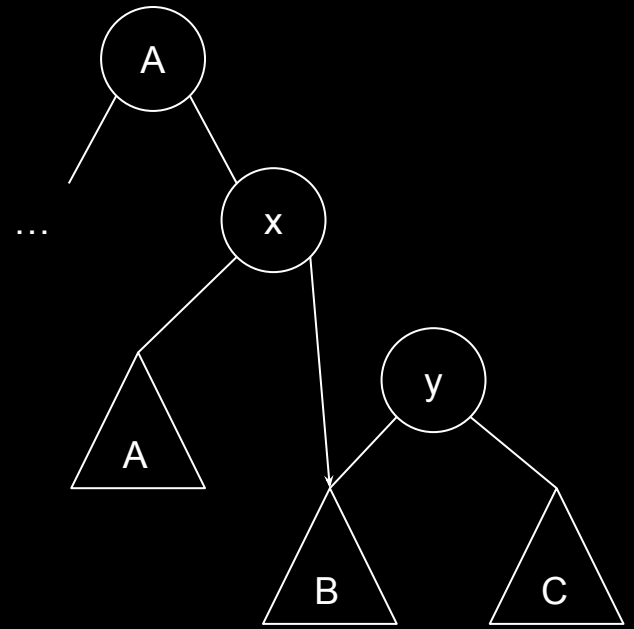
```
função rotacaoEsquerda(T,x)
entrada: nodo x a ser rotacionado e a árvore T
saída: o nodo x é rotacionado a esquerda.

y = x.fd
x.fd = y.fe
se y.fe ≠ sentinela
     y.fe.pai = x
y.p = x.p
se x.p == sentinela
     T.raiz = y
senão
     se x == x.p.fe
          x.p.fe = y
     senão
          x.p.fd = y
y.fe = x
x.p = y
```



Teste de Mesa

```
função rotacaoEsquerda(T,x)
entrada: nodo x a ser rotacionado e a árvore T
saída: o nodo x é rotacionado a esquerda.

y = x.fd
x.fd = y.fe
se y.fe ≠ sentinela
     y.fe.pai = x
y.p = x.p
se x.p == sentinela
     T.raiz = y
senão
     se x == x.p.fe
          x.p.fe = y
     senão
          x.p.fd = y
y.fe = x
x.p = y
```

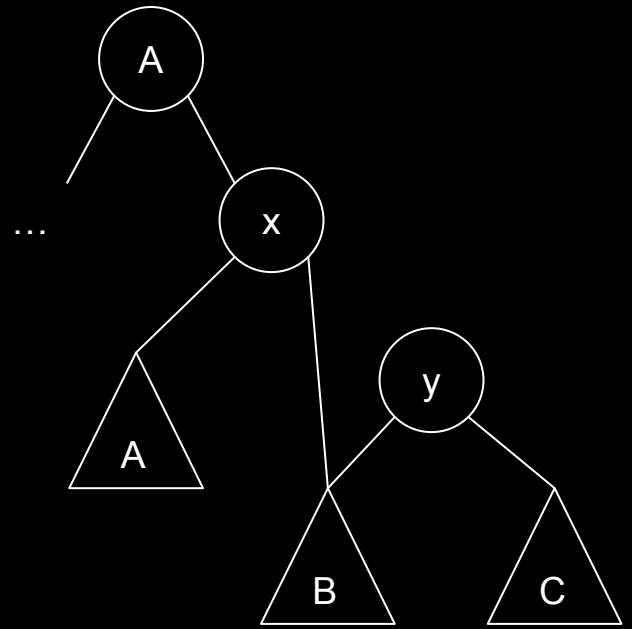
rotacaoEsquerda(T,x)



Teste de Mesa

```
função rotacaoEsquerda(T,x)
entrada: nodo x a ser rotacionado e a árvore T
saída: o nodo x é rotacionado a esquerda.

y = x.fd
x.fd = y.fe
se y.fe ≠ sentinela
     y.fe.pai = x
y.p = x.p
se x.p == sentinela
     T.raiz = y
senão
     se x == x.p.fe
          x.p.fe = y
     senão
          x.p.fd = y
y.fe = x
x.p = y
```

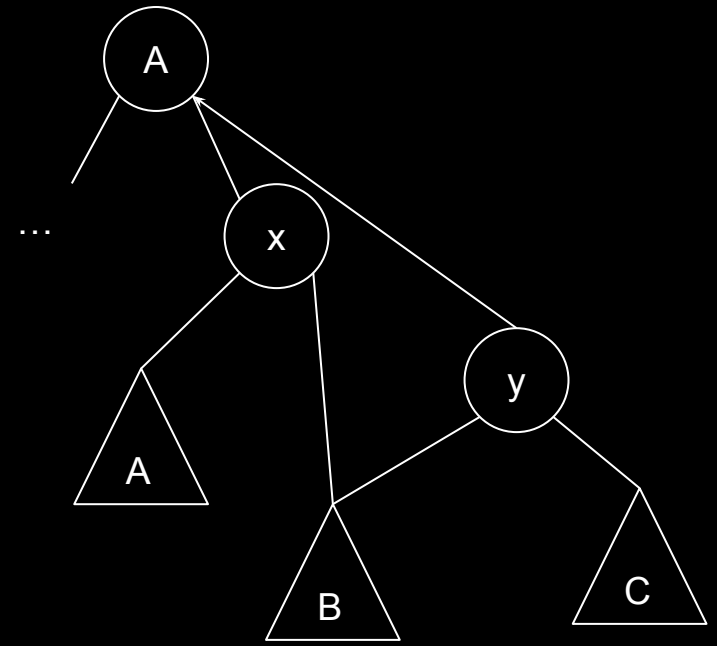
rotacaoEsquerda(T,x)



Teste de Mesa

```
função rotacaoEsquerda(T,x)
entrada: nodo x a ser rotacionado e a árvore T
saída: o nodo x é rotacionado a esquerda.

y = x.fd
x.fd = y.fe
se y.fe ≠ sentinela
     y.fe.pai = x
y.p = x.p
se x.p == sentinela
     T.raiz = y
senão
     se x == x.p.fe
          x.p.fe = y
     senão
          x.p.fd = y
y.fe = x
x.p = y
```

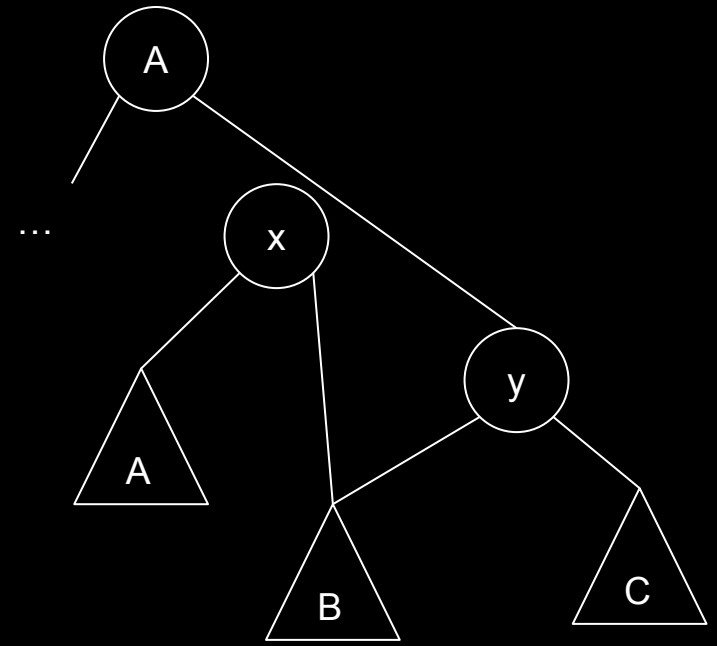
rotacaoEsquerda(T,x)



Teste de Mesa

```
função rotacaoEsquerda(T,x)
entrada: nodo x a ser rotacionado e a árvore T
saída: o nodo x é rotacionado a esquerda.

y = x.fd
x.fd = y.fe
se y.fe ≠ sentinela
     y.fe.pai = x
y.p = x.p
se x.p == sentinela
     T.raiz = y
senão
     se x == x.p.fe
          x.p.fe = y
     senão
          x.p.fd = y
y.fe = x
x.p = y
```

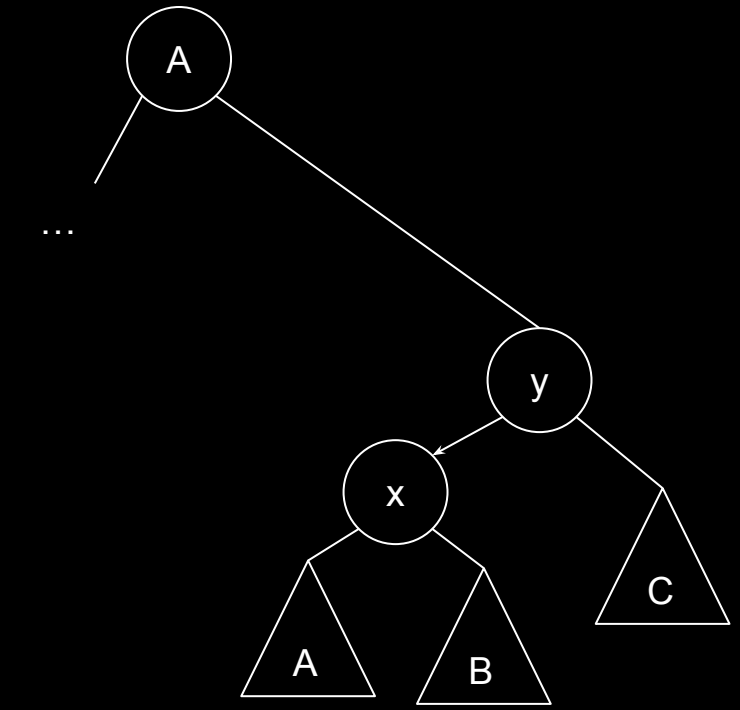
rotacaoEsquerda(T,x)



Teste de Mesa

rotacaoEsquerda(T,x)

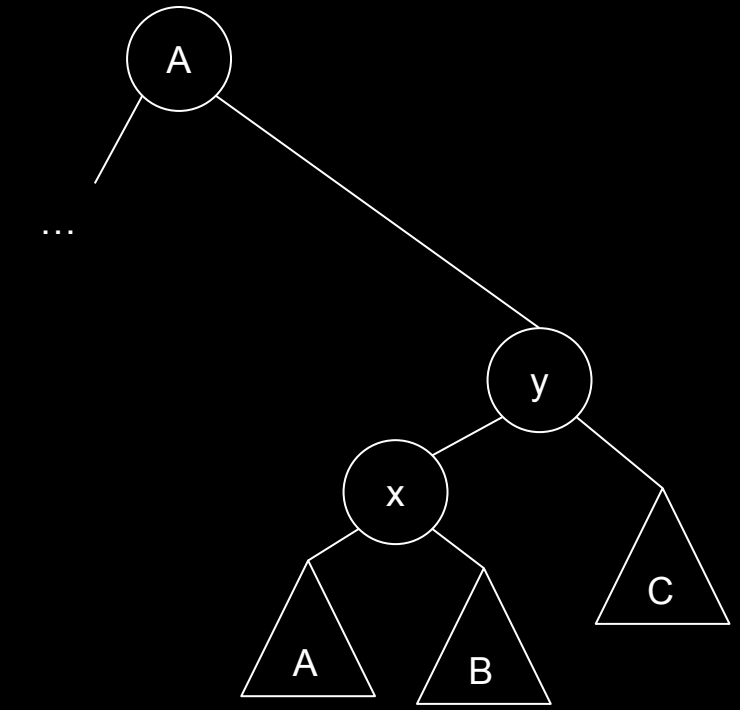
```
função rotacaoEsquerda(T,x)
entrada: nodo x a ser rotacionado e a árvore T
saída: o nodo x é rotacionado a esquerda.

y = x.fd
x.fd = y.fe
se y.fe ≠ sentinela
     y.fe.pai = x
y.p = x.p
se x.p == sentinela
     T.raiz = y
senão
     se x == x.p.fe
          x.p.fe = y
     senão
          x.p.fd = y
y.fe = x
x.p = y
```



Teste de Mesa

```
função rotacaoEsquerda(T,x)
entrada: nodo x a ser rotacionado e a árvore T
saída: o nodo x é rotacionado a esquerda.

y = x.fd
x.fd = y.fe
se y.fe ≠ sentinela
     y.fe.pai = x
y.p = x.p
se x.p == sentinela
     T.raiz = y
senão
     se x == x.p.fe
          x.p.fe = y
     senão
          x.p.fd = y
y.fe = x
x.p = y
```

rotacaoEsquerda(T,x)



Rotação Direita

Rotação direita é simétrica à rotação esquerda.

Inserção

Na inserção:

1. Insere-se o nodo normalmente, como em uma árvore de busca binária;
    a. O nodo inserido é vermelho;
2. Chamar `redBlackInsertFixup` para corrigir a árvore;
    a. Mudar colorações e efetuar rotações.

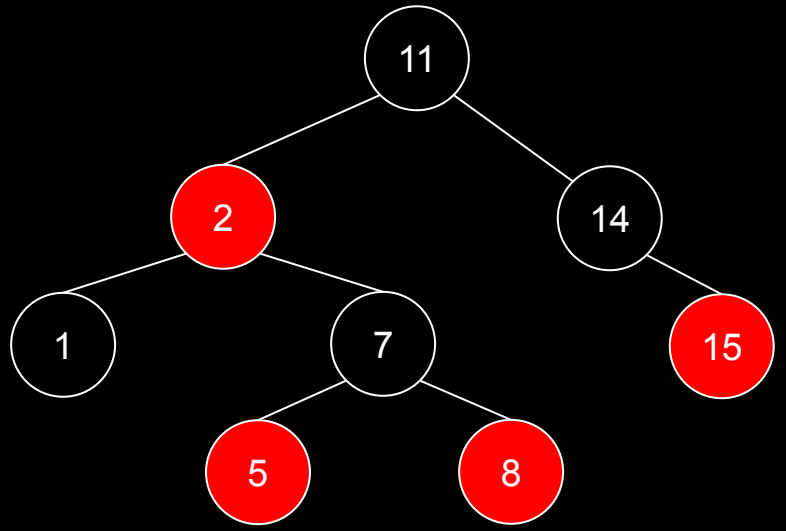
Inserção

```
função criarNodo(c)
entrada: chave c
saída: nodo z para ser inserido na árvore
red-black.

z.chave = c
z.fe = sentinela
z.fd = sentinela
z.cor = vermelho

retorne z
```

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```

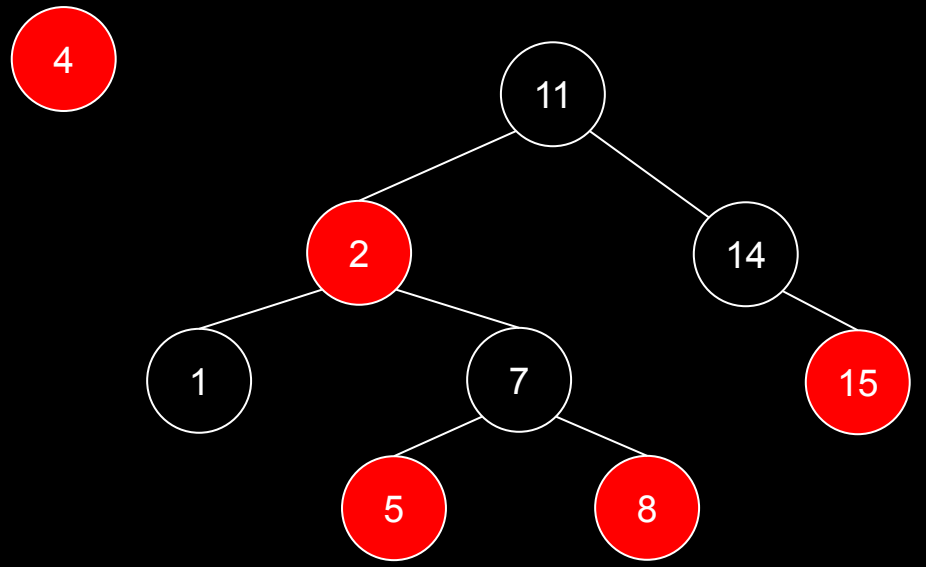
redBlackInsert(T,4)

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```



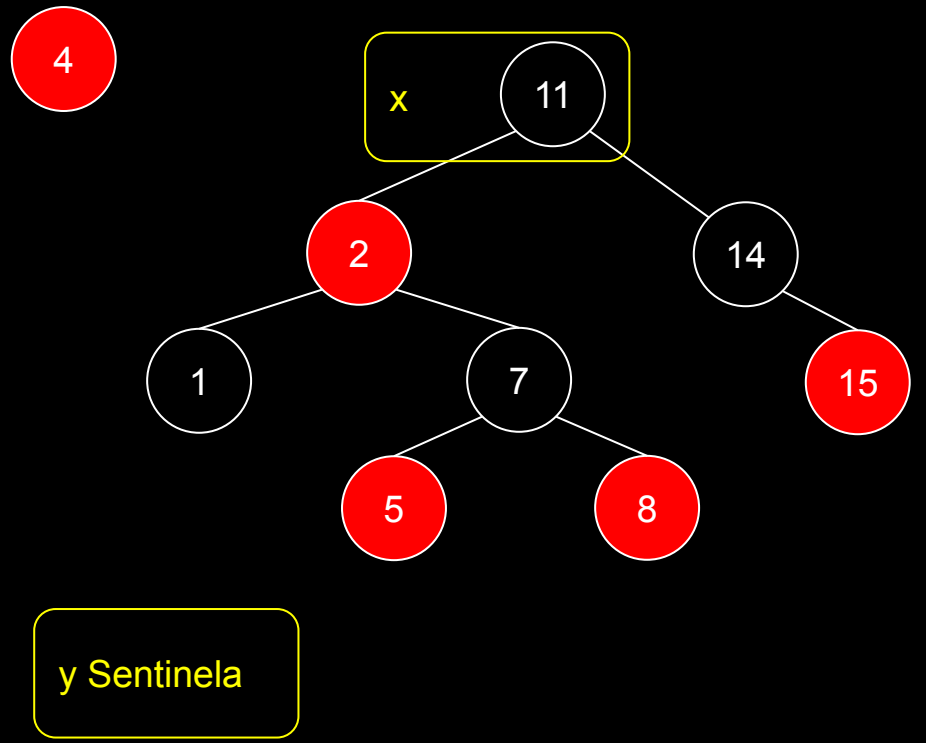
redBlackInsert(T,4)

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```



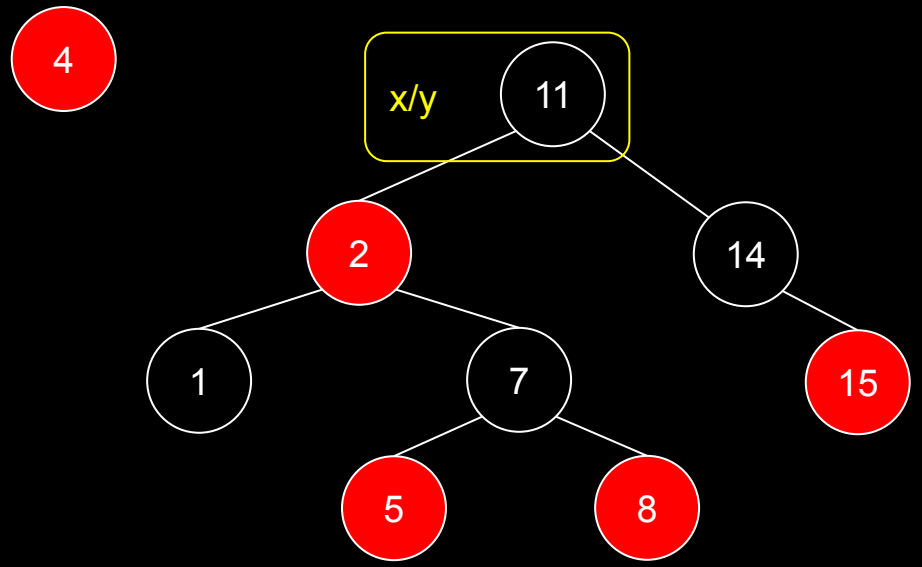
redBlackInsert(T,4)

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```



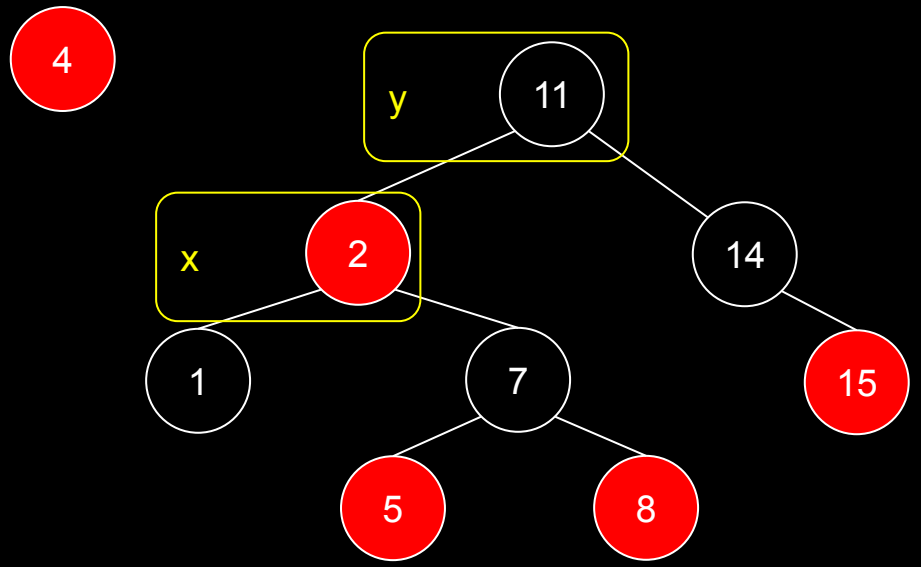
redBlackInsert(T,4)

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```



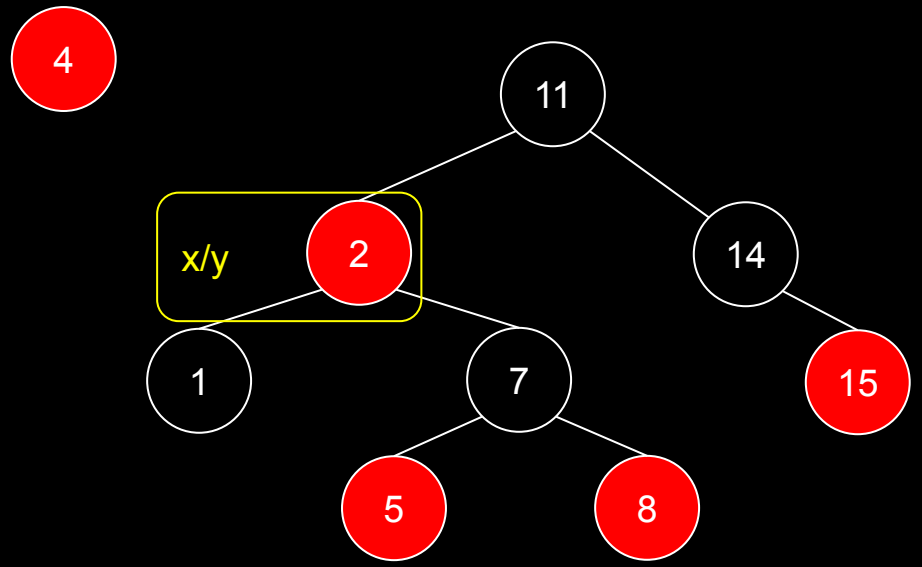
redBlackInsert(T,4)

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```



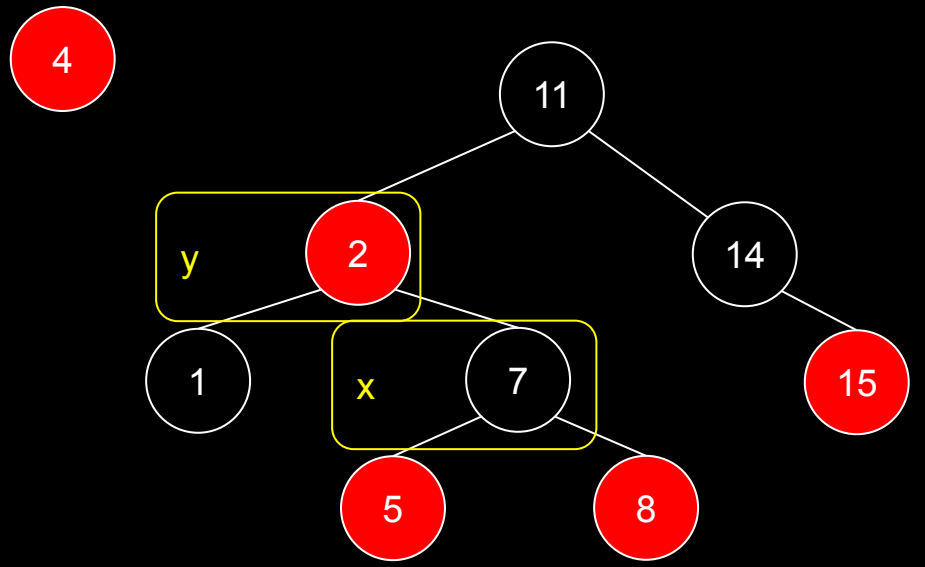
redBlackInsert(T,4)

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```



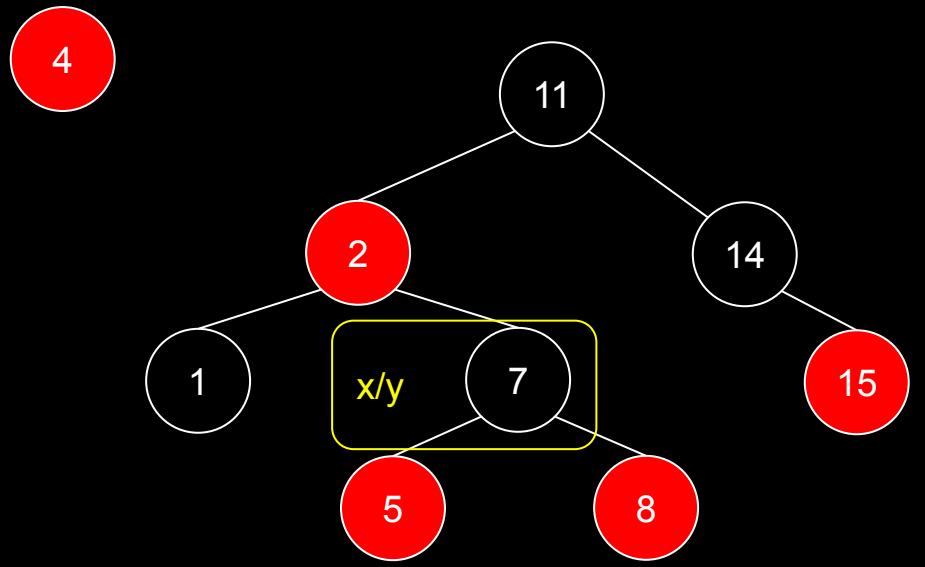
redBlackInsert(T,4)

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```



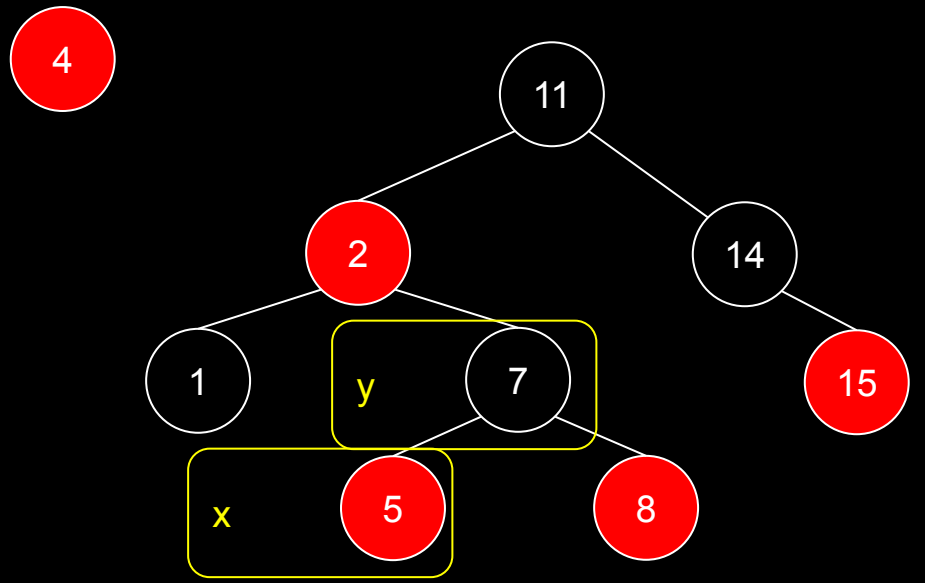
redBlackInsert(T,4)

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```



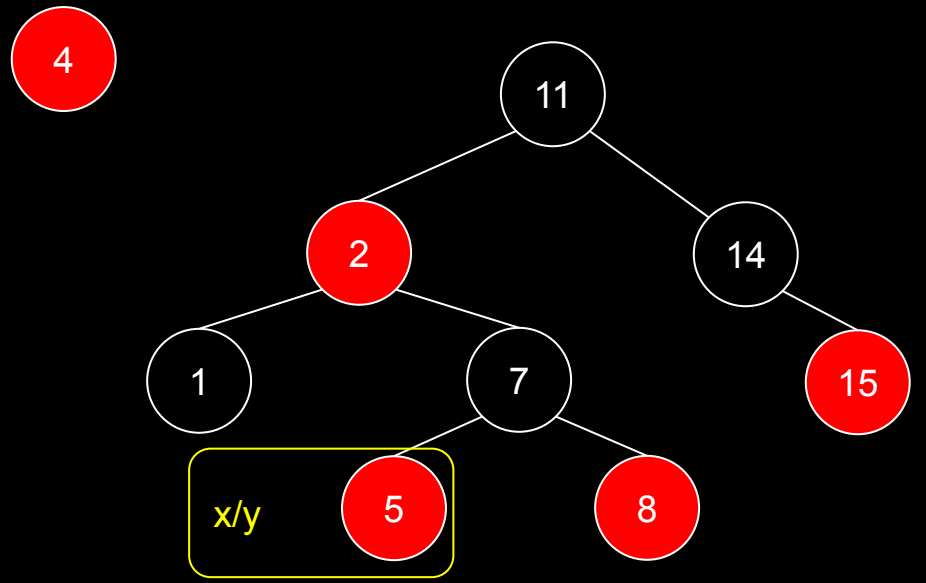
redBlackInsert(T,4)

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```



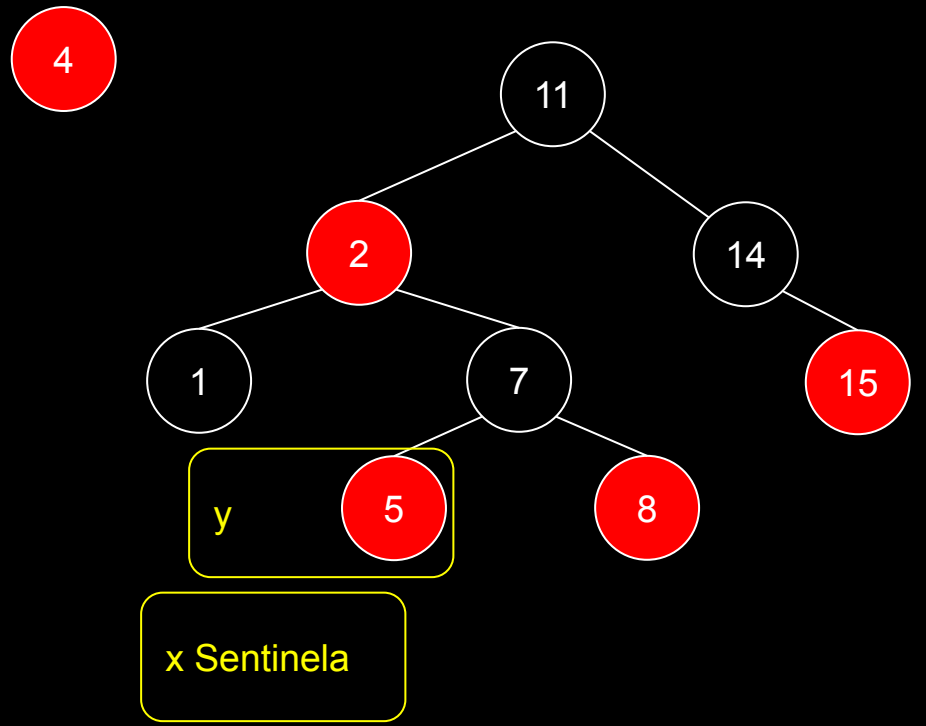
redBlackInsert(T,4)

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```



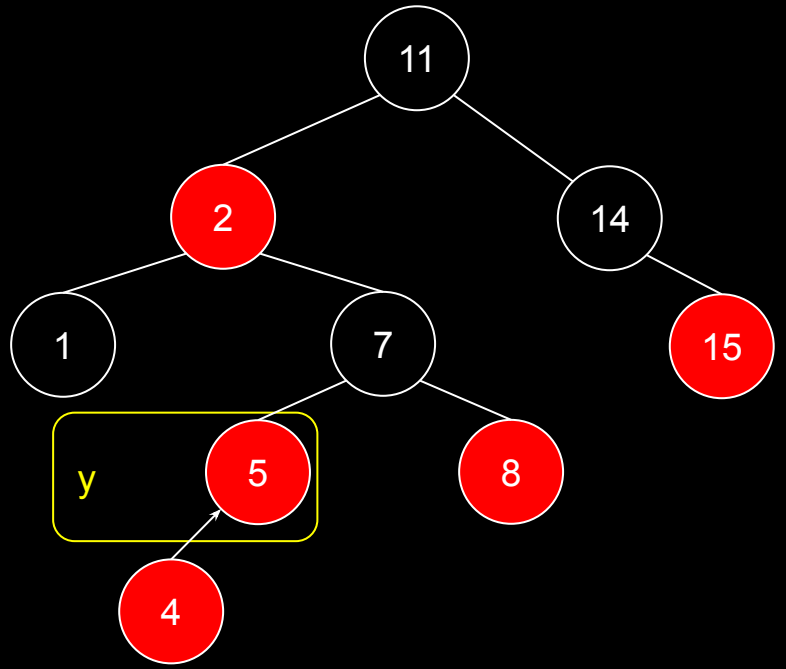
redBlackInsert(T,4)

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```



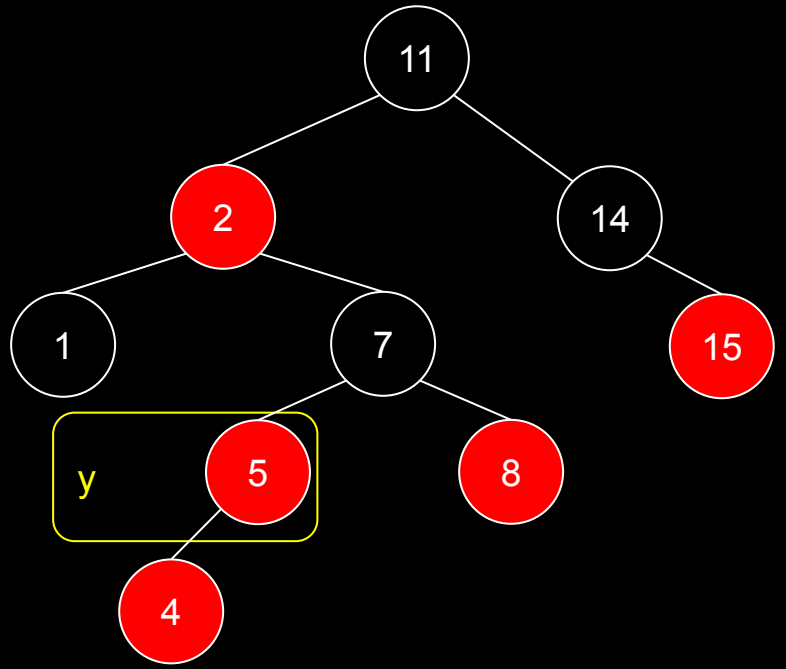
redBlackInsert(T,4)

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```



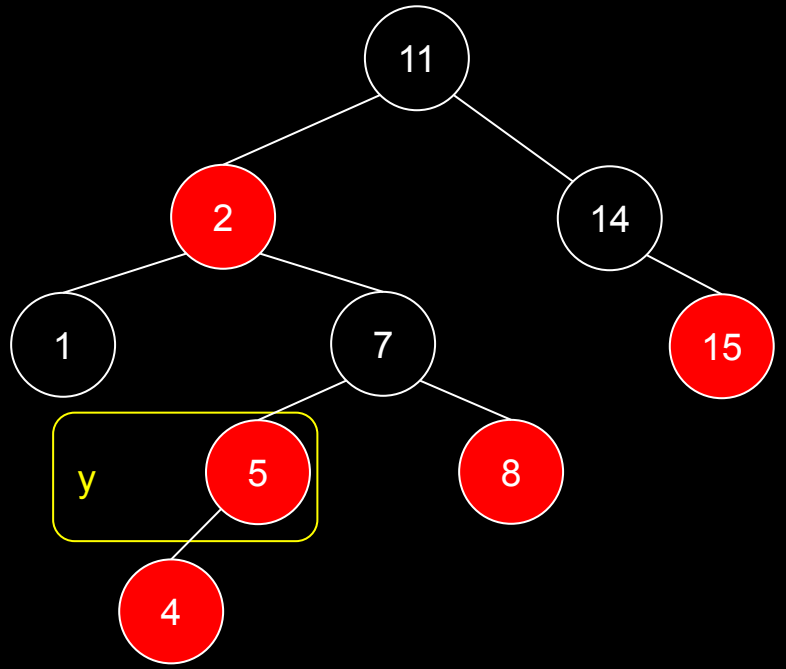
redBlackInsert(T,4)

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```



redBlackInsert(T,4)

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z) ???
```



Inserção

Após a inserção, quais propriedades podem ter sido quebradas?

1. Todo nodo é vermelho ou preto;
2. A raiz é preta;
3. Toda folha é um **sentinela** preto;
4. Se um nodo é vermelho, ambos filhos são pretos;
5. Todos caminhos simples (sem repetição de nodos) de um nodo qualquer até suas folhas descendentes têm o mesmo número de nodos pretos.

```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```

Inserção

Após a inserção, quais propriedades podem ter sido quebradas?

1. Todo nodo é vermelho ou preto;
2. A raiz é preta;
3. Toda folha é um **sentinela** preto;
4. Se um nodo é vermelho, ambos filhos são pretos;
5. Todos caminhos simples (sem repetição de nodos) de um nodo qualquer até suas folhas descendentes têm o mesmo número de nodos pretos.

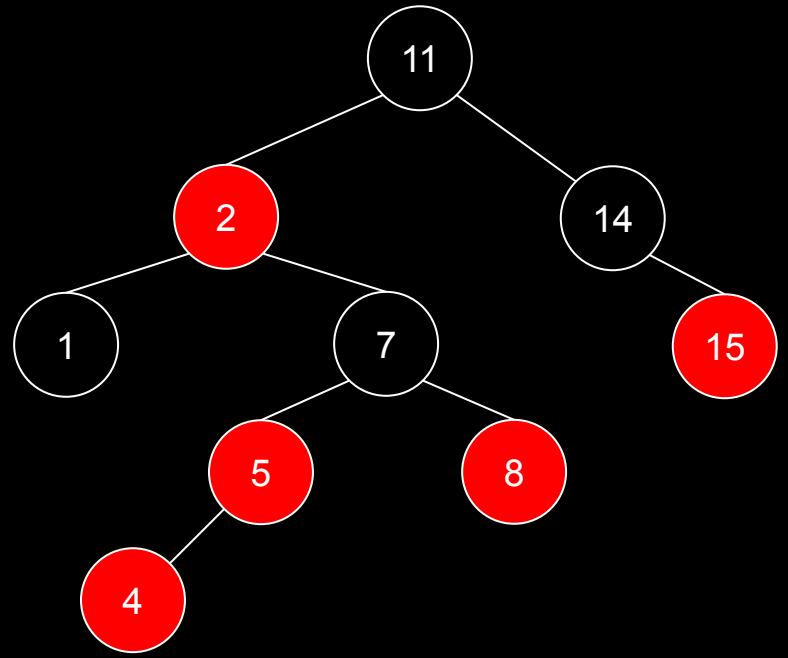
```
função redBlackInsert(T,c)
entrada: chave c a ser inserida na árvore T
saída: z é inserido na árvore red-black de
forma a manter as propriedades da árvore.

z = criarNodo(c)
x = T.raiz
y = sentinela
enquanto x ≠ sentinela
     y = x
     se z.chave < x.chave
          x = x.fe
     senão
          x = x.fd
z.p = y
se y == sentinela
     T.raiz = z
senão
     se z.chave < y.chave
          y.fe = z
     senão
          y.fd = z
redBlackInsertFixup(T,z)
```

Fixup

```
função redBlackInsertFixup(T,z)
entrada: nodo z inserido na árvore T
saída: a árvore é modificada de forma a manter as propriedades da
red-black.

enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe //o pai era um filho esquerdo?
          y = z.pai.pai.fd    //y é o tio de z
          se y.cor == vermelho //o tio e o pai eram vermelhos?
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
          //por exemplo, troca rotaçãoDireita por rotaçãoEsquerda
          //veja o algoritmo completo em Cormen et. al. (2022).
T.raiz.cor = preto
```

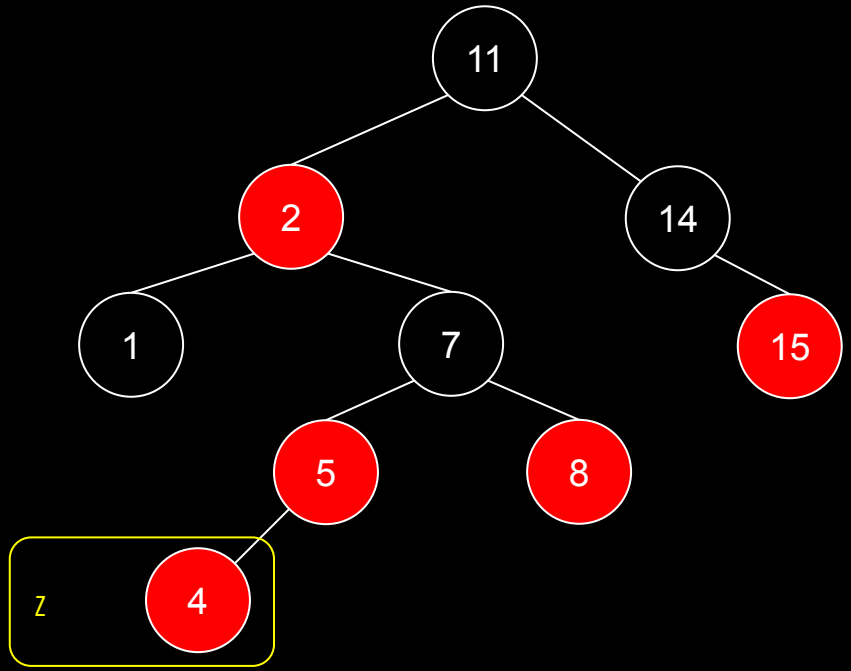
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



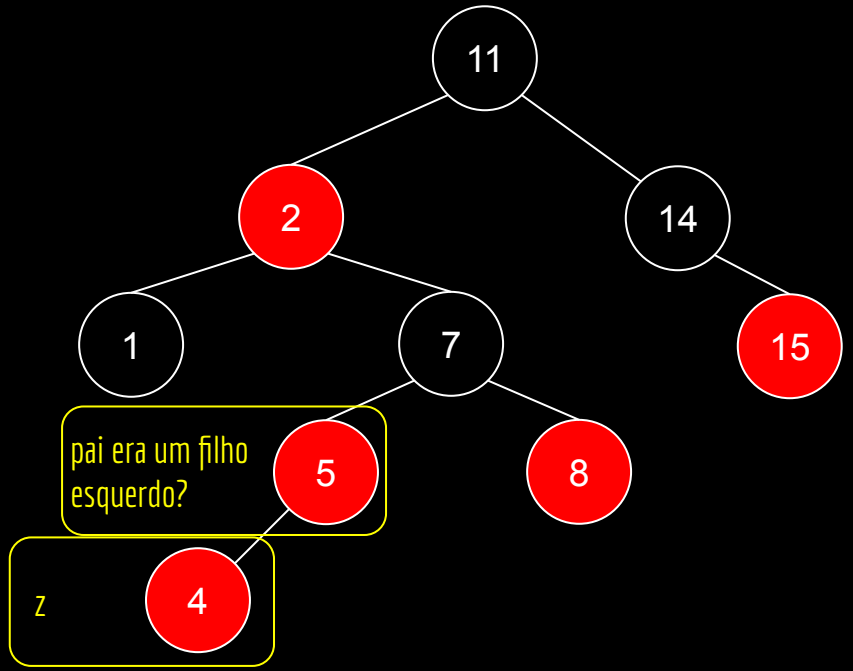
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



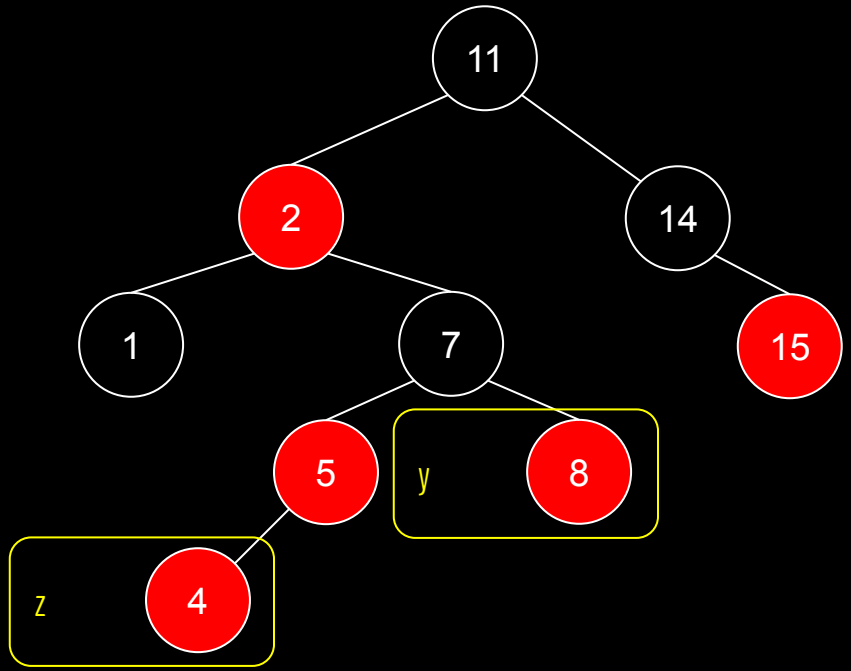
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso "se"
T.raiz.cor = preto
```



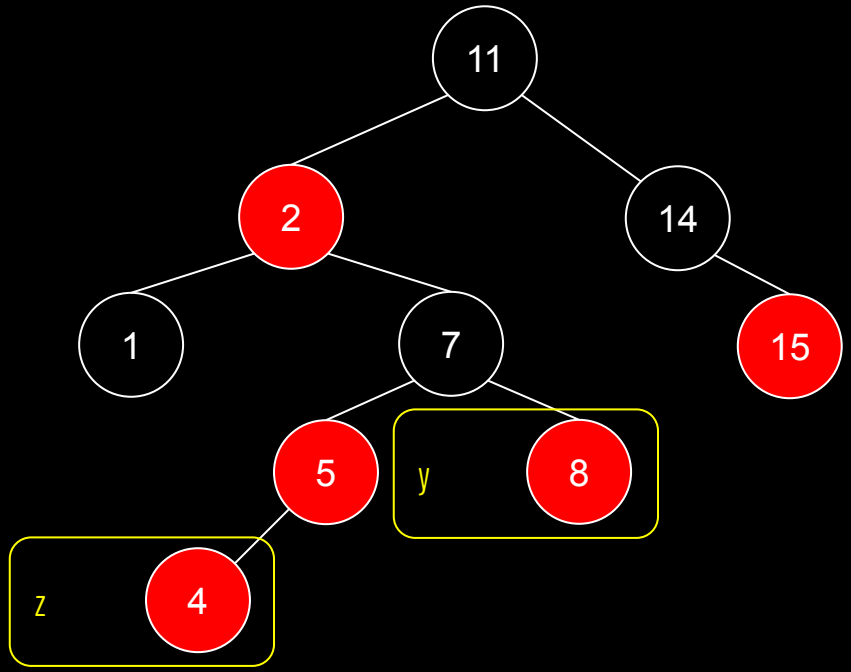
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



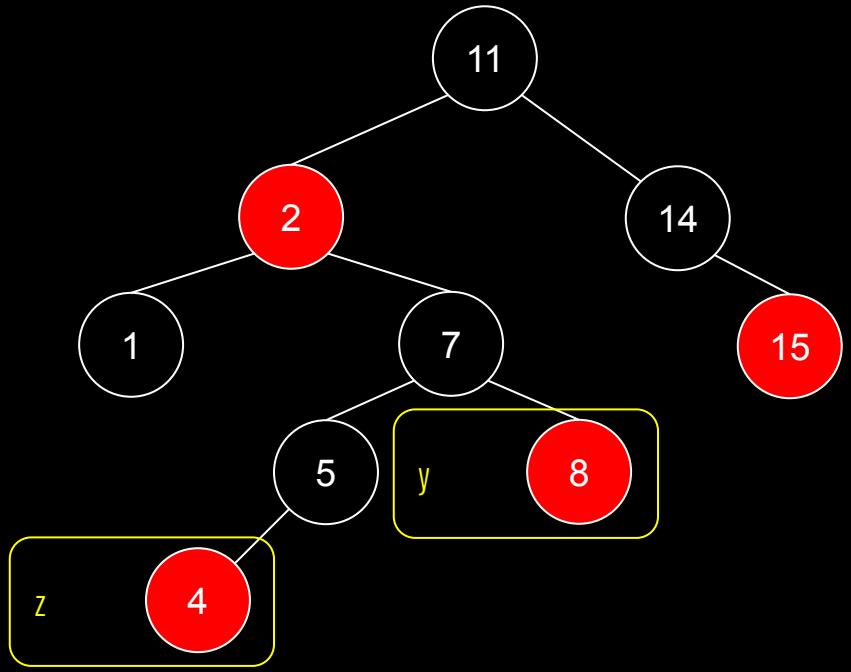
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



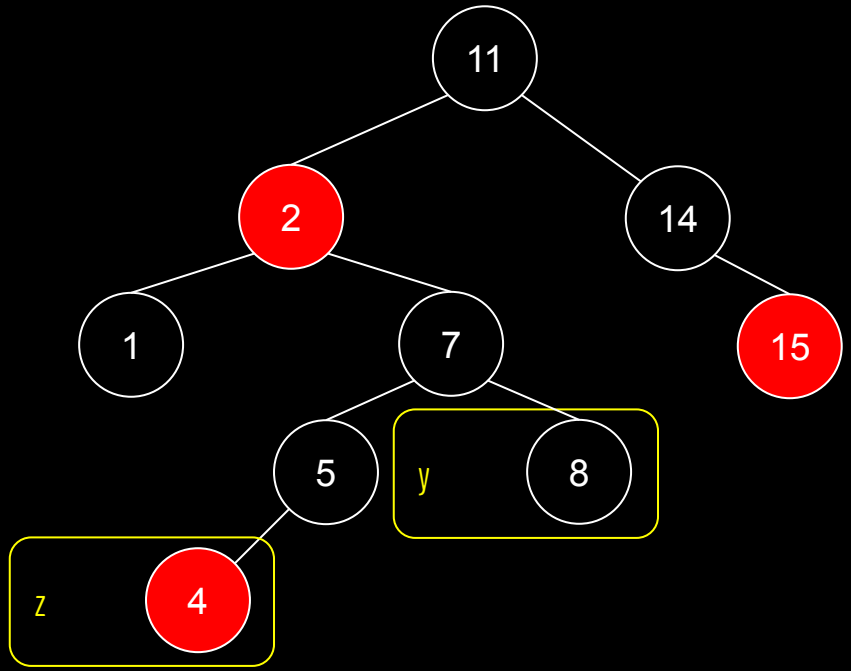
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso "se"
T.raiz.cor = preto
```



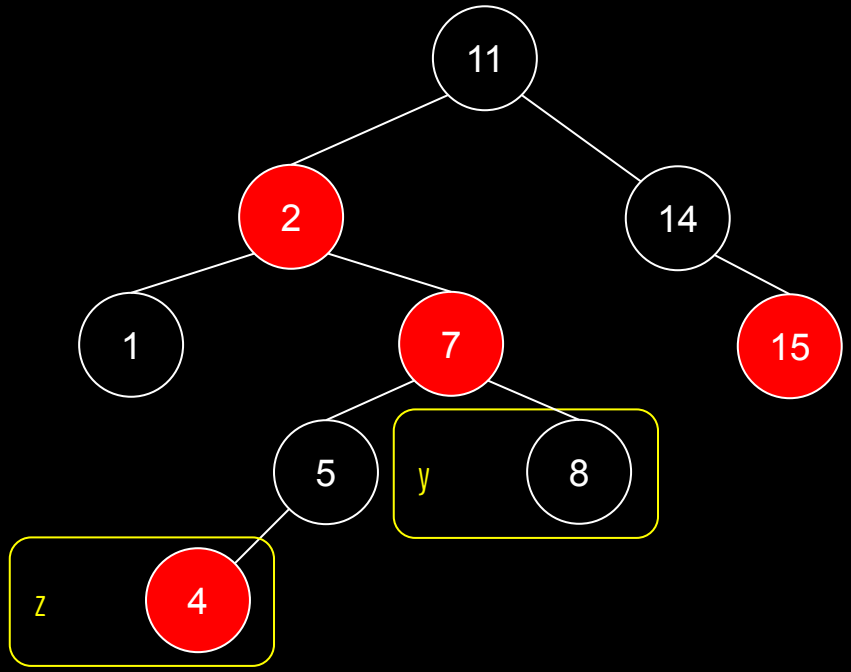
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



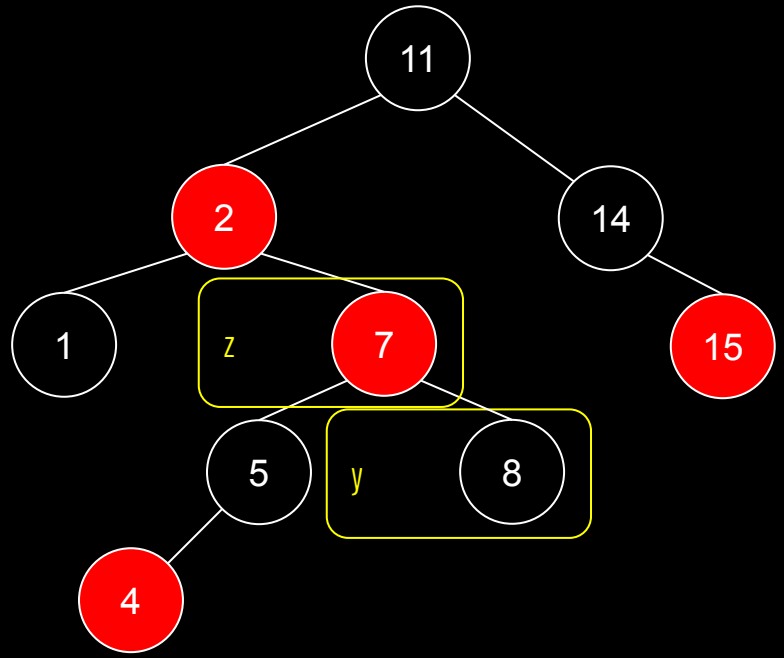
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



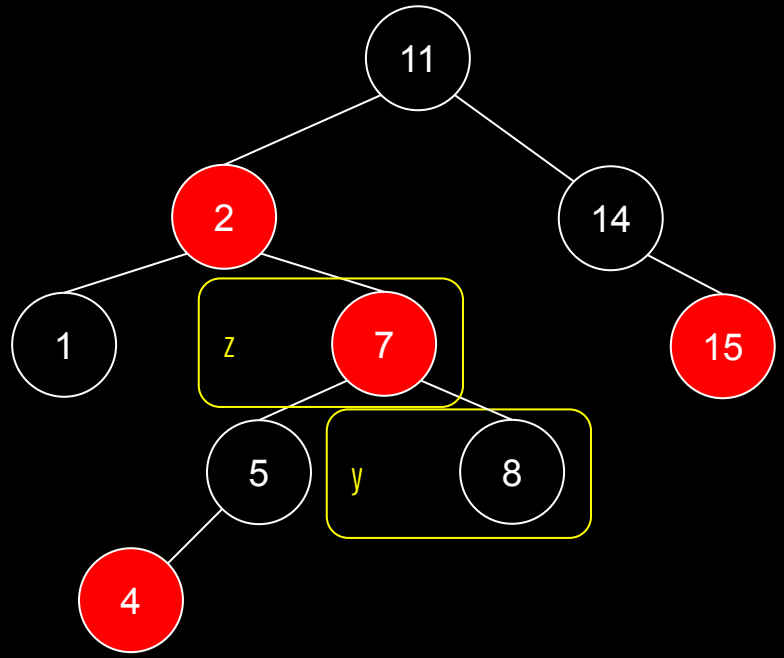
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



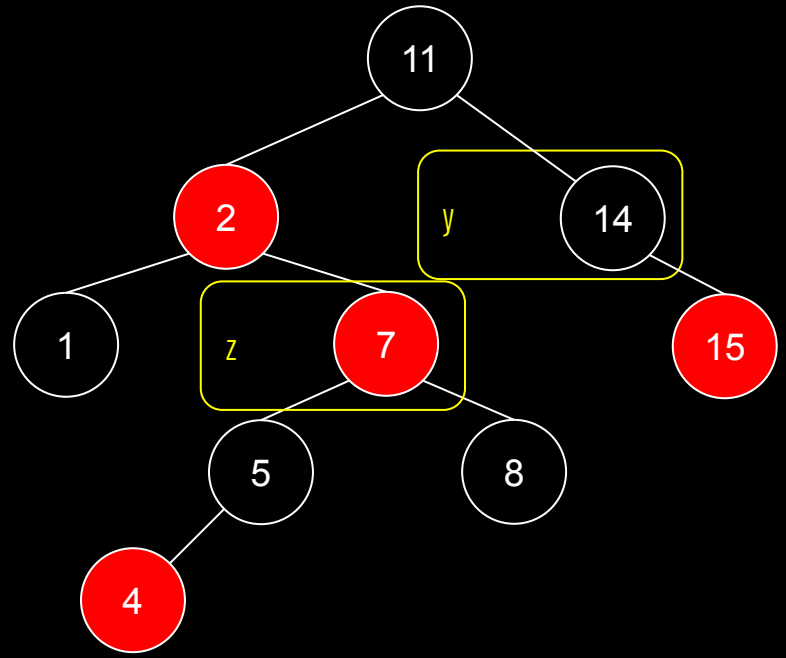
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



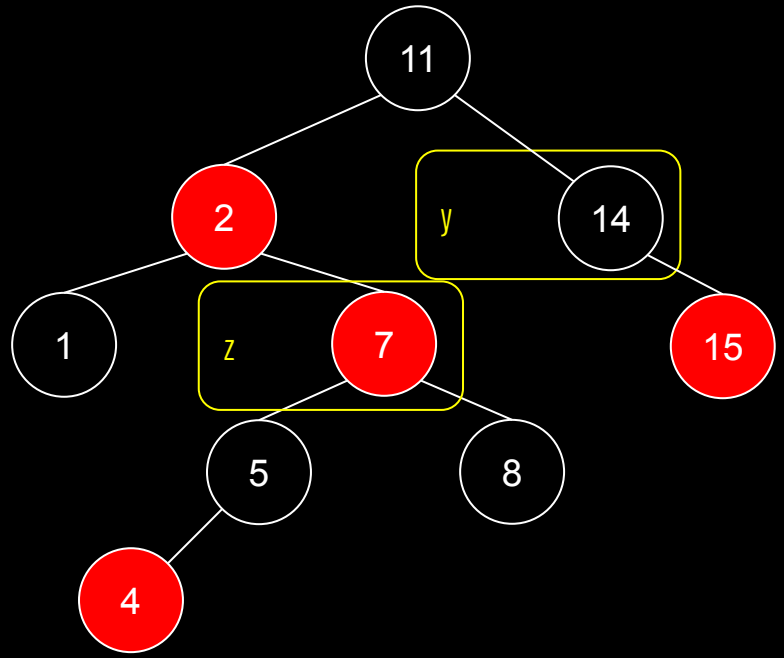
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



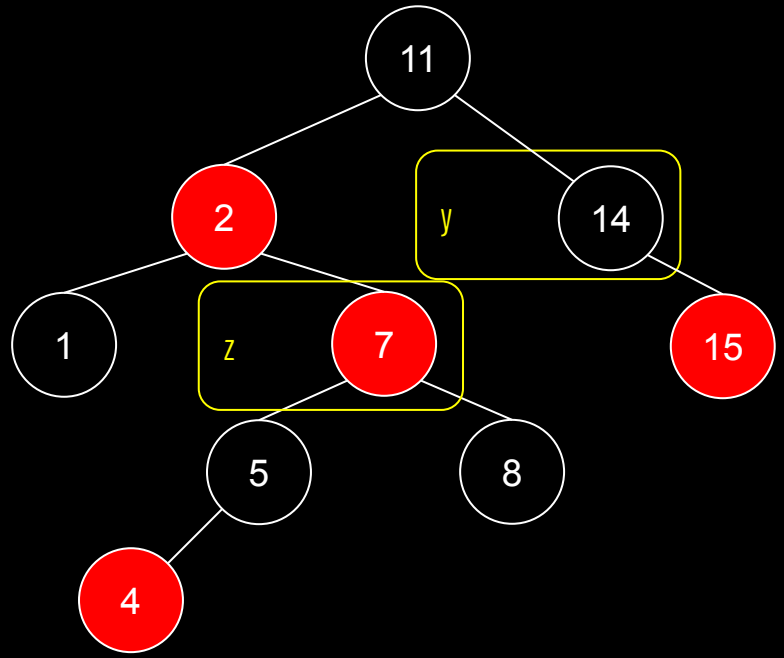
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



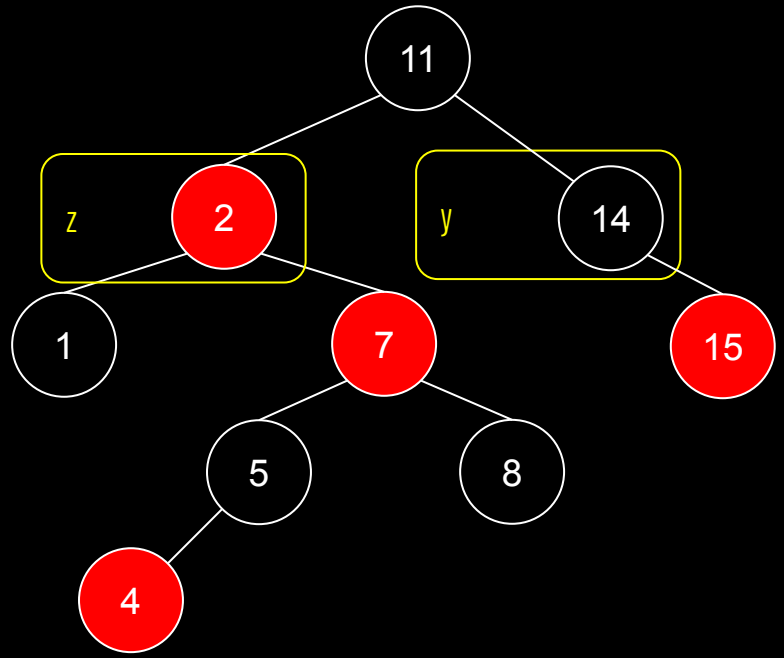
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



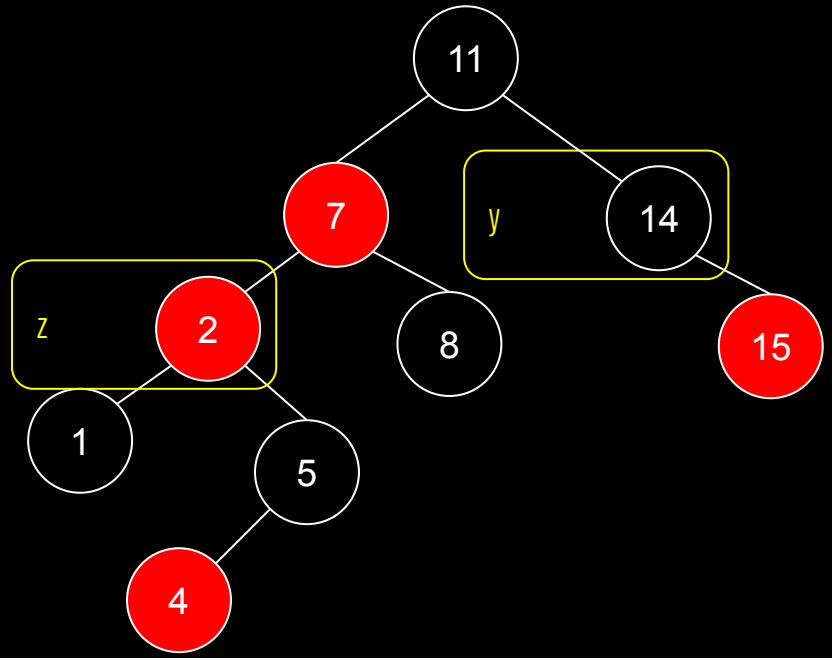
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



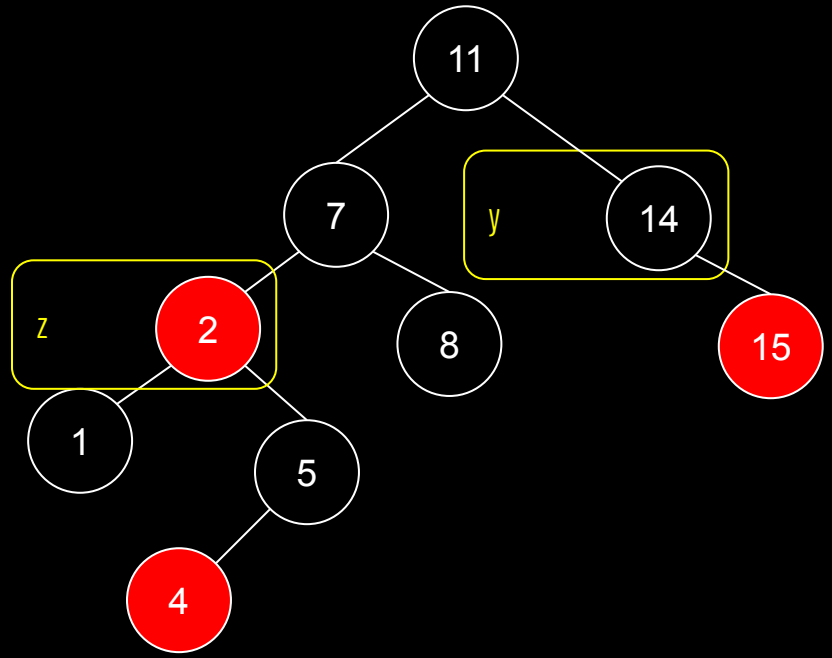
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



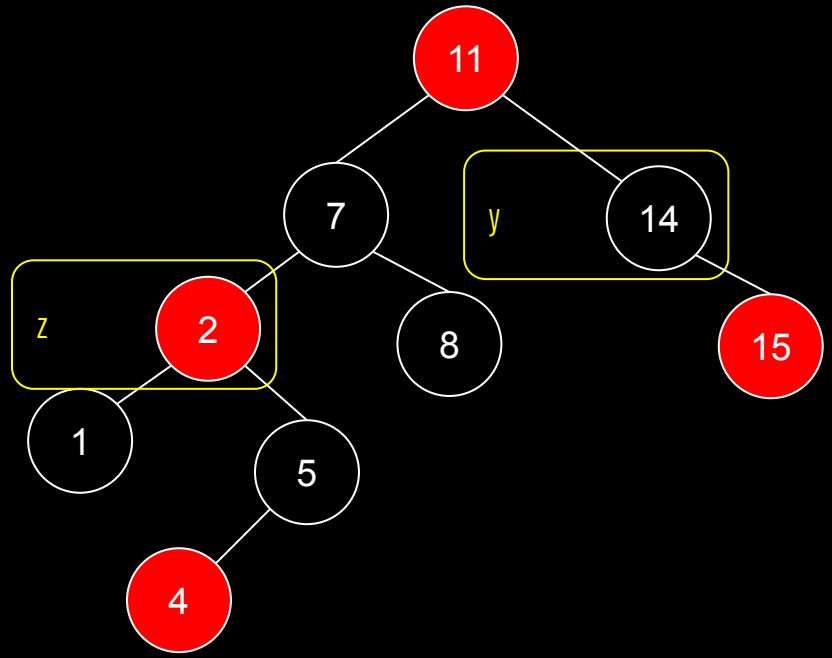
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



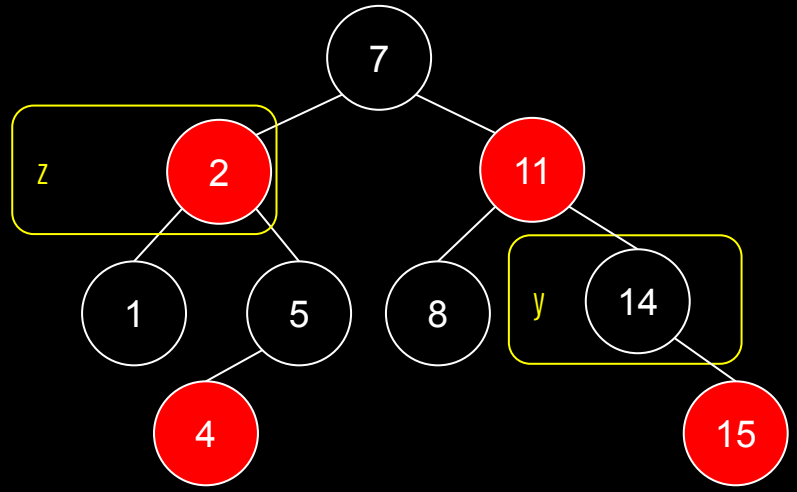
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



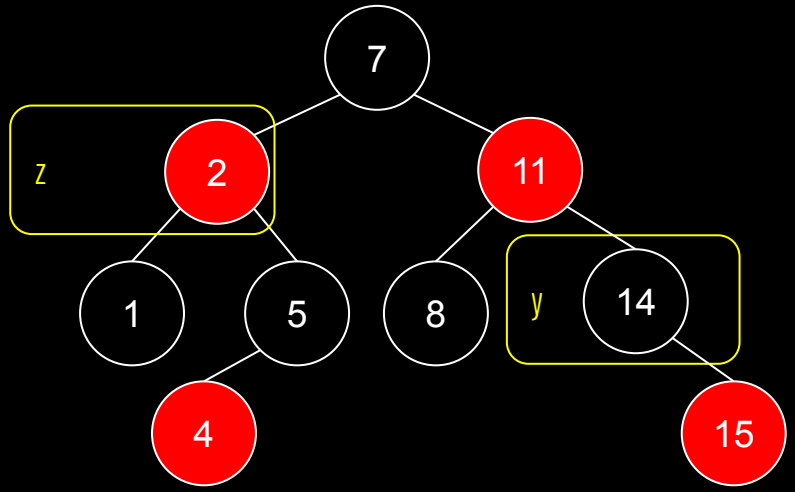
Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



Exemplo

```
função redBlackInsertFixup(T,z)
enquanto z.pai.cor == vermelho
     se z.pai == z.pai.pai.fe
          y = z.pai.pai.fd
          se y.cor == vermelho
               z.pai.cor = preto
               y.cor = preto
               z.pai.pai.cor = vermelho
               z = z.pai.pai
          senão
               se z == z.pai.fd
                    z = z.pai
                    rotacaoEsquerda(T,z)
               z.pai.cor = preto
               z.pai.pai.cor = vermelho
               rotacaoDireita(T,z.pai.pai)
     senão  //o pai era um filho direito?
          //faz as operações espelhadas do caso “se”
T.raiz.cor = preto
```



Formalismo

Um pouco de formalismo:

Vamos mostrar que uma red-black com n nodos internos tem uma altura de no máximo $2log_2(n+1)$.

Formalismo

Prova que toda subárvore com raiz em um nodo *x* possui pelo menos $2^{bh(x)} - 1$ nodos internos, onde *bh(x)* é a altura preta de *x*.

Prova por indução na altura de *x*.

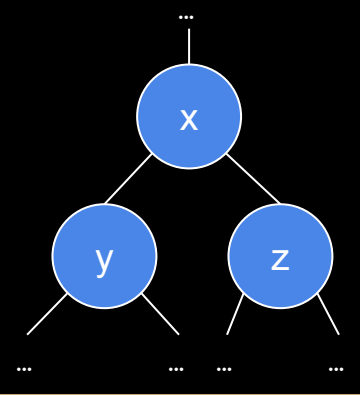
Formalismo

- Se a altura de *x* é zero, então x é um sentinela (folha), e a árvore de fato possui $2^{bh(x)} - 1 = 2^0 - 1 = 0$ nodos internos.

Sentinela

Formalismo

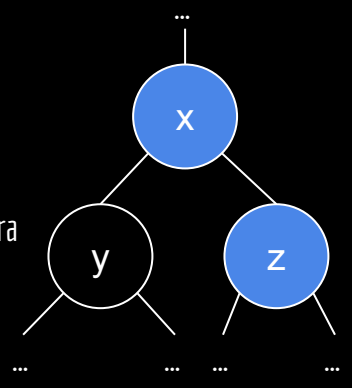
- Se a altura de *x* é zero, então x é um sentinela (folha), e a árvore de fato possui $2^{bh(x)} - 1 = 2^0 - 1 = 0$ nodos internos.
- Considere um nodo interno x com altura positiva.



Formalismo

- Se a altura de *x* é zero, então x é um sentinela (folha), e a árvore de fato possui $2^{bh(x)} - 1 = 2^0 - 1 = 0$ nodos internos.
- Considere um nodo interno x com altura positiva.
  - Se um filho de *x* é preto, então ele contribui *1* para a altura preta de *x*, mas não contribui para sua própria altura preta.

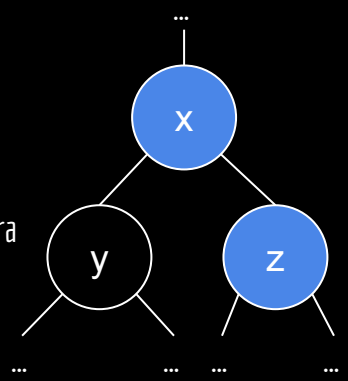
y contribui com 1 para a altura preta de x.



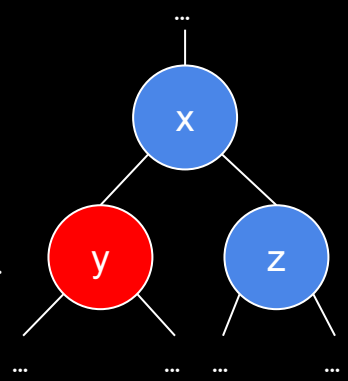
Formalismo

- Se a altura de *x* é zero, então x é um sentinela (folha), e a árvore de fato possui $2^{bh(x)} - 1 = 2^0 - 1 = 0$ nodos internos.
- Considere um nodo interno x com altura positiva.
  - Se um filho de *x* é preto, então ele contribui *1* para a altura preta de *x*, mas não contribui para sua própria altura preta.
  - Se um filho de *x* é vermelho, então ele não contribui *1* altura preta de *x*, nem para sua própria altura preta.

y contribui com 1 para a altura preta de x.



Não contribui.



Formalismo

- Se a altura de *x* é zero, então x é um sentinela (folha), e a árvore de fato possui $2^{bh(x)} - 1 = 2^0 - 1 = 0$ nodos internos.
- Considere um nodo interno x com altura positiva.
  - Se um filho de *x* é preto, então ele contribui *1* para a altura preta de *x*, mas não contribui para sua própria altura preta.
  - Se um filho de *x* é vermelho, então ele não contribui *1* altura preta de *x*, nem para sua própria altura preta.
- Logo, cada filho de x tem uma altura preta de:
  - bh(x)-1 se for preto,
  - bh(x) se for vermelho.

Formalismo

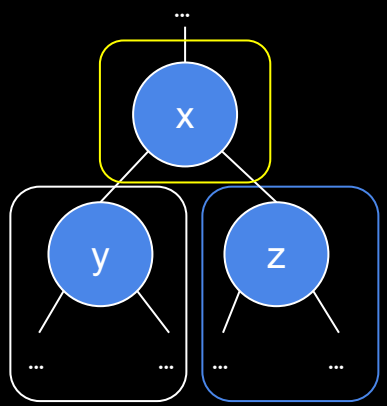
- Se a altura de *x* é zero, então x é um sentinela (folha), e a árvore de fato possui $2^{bh(x)} - 1 = 2^0 - 1 = 0$ nodos internos.
- Considere um nodo interno x com altura positiva.
  - Se um filho de *x* é preto, então ele contribui *1* para a altura preta de *x*, mas não contribui para sua própria altura preta.
  - Se um filho de *x* é vermelho, então ele não contribui *1* altura preta de *x*, nem para sua própria altura preta.
- Logo, cada filho de x tem uma altura preta de:
  - bh(x)-1 se for preto,
  - bh(x) se for vermelho.
- Sabemos que a altura de qualquer filho de *x* é menor do que a altura de *x*. Logo, aplicando o passo da indução, cada filho deve ter $2^{bh(x)-1} - 1$ nodos internos.

Formalismo

- Dessa forma, a subárvore com raiz em x possui pelo menos $(2^{bh(x)-1} - 1) + (2^{bh(x)-1} - 1) + 1$ nodos internos, completando a prova.

Formalismo

- Dessa forma, a subárvore com raiz em x possui pelo menos $(2^{bh(x)-1} - 1) + (2^{bh(x)-1} - 1) + 1$ nodos internos, completando a prova.



Formalismo

- Seja *h* a altura da da árvore.
- De acordo com a propriedade 4.
  - Se um nodo é vermelho, ambos os filhos são pretos.
  - Logo, pelo menos metade dos nodos nodos devem ser pretos.

Formalismo

- Seja *h* a altura da da árvore.
- De acordo com a propriedade 4.
  - Se um nodo é vermelho, ambos os filhos são pretos.
  - Logo, pelo menos metade dos nodos nodos devem ser pretos.
- Logo, a altura preta da raiz deve ser pelo menos h/2, e assim:

$$n \geq 2^{h/2} - 1$$

Formalismo

- Seja *h* a altura da da árvore.
- De acordo com a propriedade 4.
  - Se um nodo é vermelho, ambos os filhos são pretos.
  - Logo, pelo menos metade dos nodos nodos devem ser pretos.
- Logo, a altura preta da raiz deve ser pelo menos h/2, e assim:

  $n \geq 2^{h/2} - 1$
- Assim:

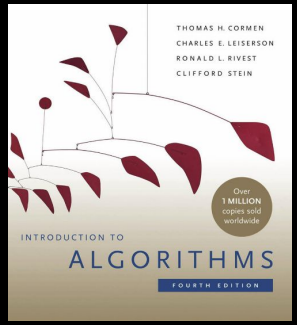
  $n + 1 \geq 2^{h/2}$

  $\log_2(n + 1) \geq h/2$

  $h \leq 2\log_2(n + 1)$

  ou seja, $h = O(\log_2 n)$

Veja essa prova em Cormen et. al. (2022).



Exercícios

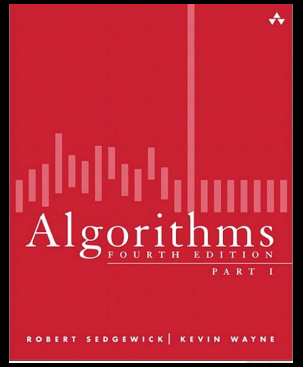
1. Implemente os algoritmos discutidos em C.

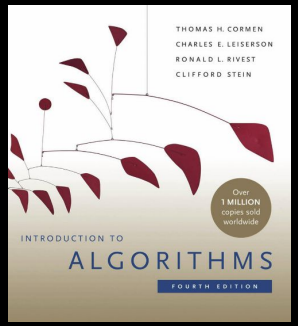
Referências

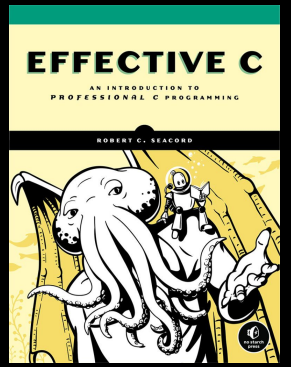
R. Sedgewick, K. Wayne. Algorithms Part I. 4a ed. 2011.

T. Cormen, C. Leiserson, R. Rivest, C. Stein. Algoritmos: Teoria e Prática. 4a ed. 2022.

Seacord, R. C. Effective C: An ntroduction to Professional C Programming. 2020.







Licença

Esta obra está licenciada com uma Licença Creative Commons Atribuição 4.0 Internacional.