"Condições excepcionalmente boas ou ruins não duram para sempre" (Benjamin Graham).

Exclusão em Red-Blacks

Paulo Ricardo Lisboa de Almeida

DSBD

UFPR

Excluir

Assim como a inserção, a exclusão em uma Red-Black tem custo $O(\log_2 n)$.

Transplante

Vamos precisar da rotina de transplante da BST com algumas modificações simples.

```
função rbTransplantar(T,u,v)
entrada: árvore Red-Black T, uma subárvore u e uma subárvore v.
saída: a subárvore v irá tomar o lugar da subárvore u em T.

se u.p é sentinela //u estava na raiz
     T.raiz = v
senão
     se u == u.pai.fe //u é um filho esquerdo?
          u.pai.fe = v
     senão
          u.pai.fd = v
v.pai = u.pai
```

Compare

Compare com o algoritmo de transplante de uma BST. Quais as diferenças? Por quê?

Red-Black

```
função rbTransplantar(T,u,v)

se u.p é sentinela //u estava na raiz
     T.raiz = v
senão
     se u == u.pai.fe
          u.pai.fe = v
     senão
          u.pai.fd = v
v.pai = u.pai
```

Árvore de Busca Binária

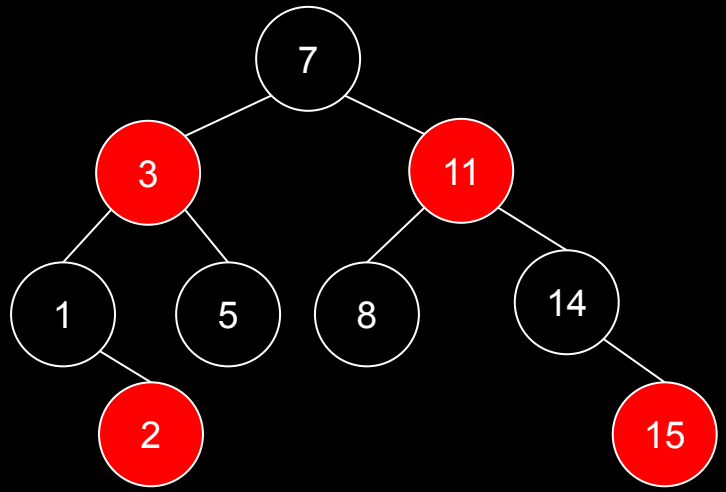
```
função transplantar(T,u,v)

se u.p é NULO //u estava na raiz
     T.raiz = v
senão
     se u == u.pai.fe
          u.pai.fe = v
     senão
          u.pai.fd = v
se v não é NULO
     v.pai = u.pai
```

Excluir

```
função rbExcluir(T,z)
entrada: árvore Red-Black T, e o nodo a ser excluído z.
saída: o nodo z é excluído de forma a manter a propriedade da Red-Black.

y = z
corOriginal = y.cor
se z.fe == sentinela
     x = z.fd
     rbTransplantar(T,z,z.fd)
senão
     se z.fd == sentinela
          x = z.fe
          rbTransplantar(T,z,z.fe)
     senão
          y = minimo(z.fd)
          corOriginal = y.cor
          x = y.fd
          se y ≠ z.fd
               rbTransplantar(T,y,y.fd)
               y.fd = z.fd
               y.fd.pai = y
          senão
               x.pai = y
          rbTransplantar(T,z,y)
          y.fe = z.fe
          y.fe.pai = y
          y.cor = z.cor
se corOriginal == preto
     redBlackDeleteFixup(T,x)
```

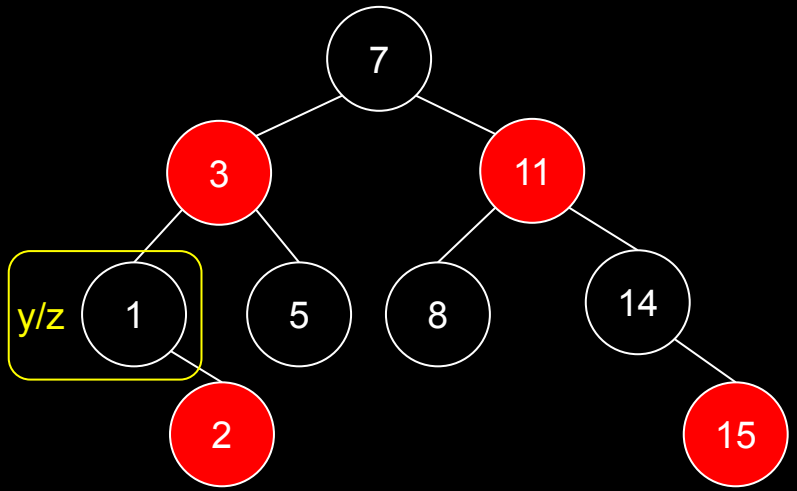
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```

rbExcluir(T,1)

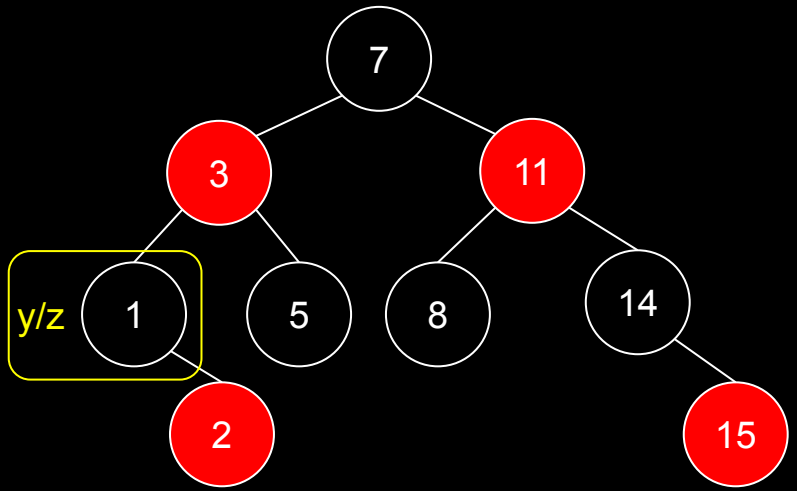


```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```

corOriginal



```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```

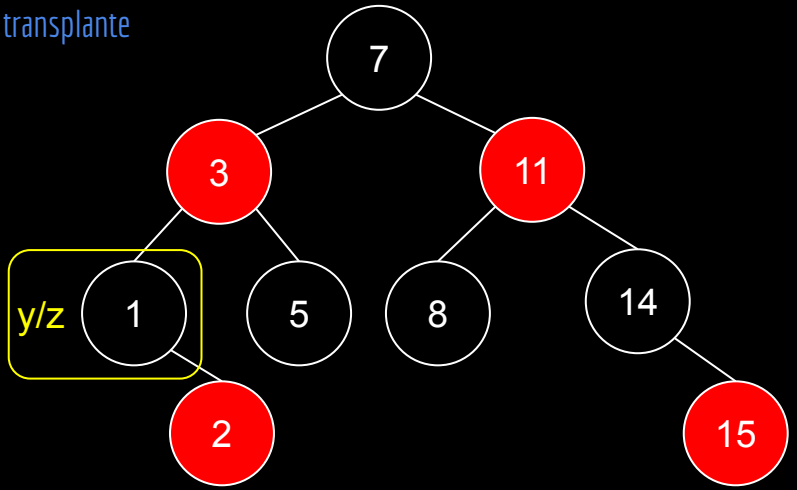


corOriginal

preto

```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```

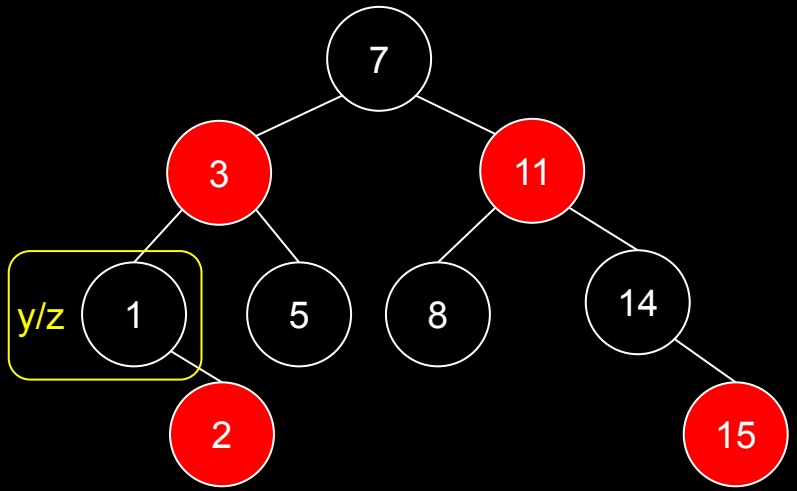
Se o nodo a ser excluído possui somente uma subárvore, ou nenhuma subárvore, um transplante resolve a exclusão.



corOriginal

preto

```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



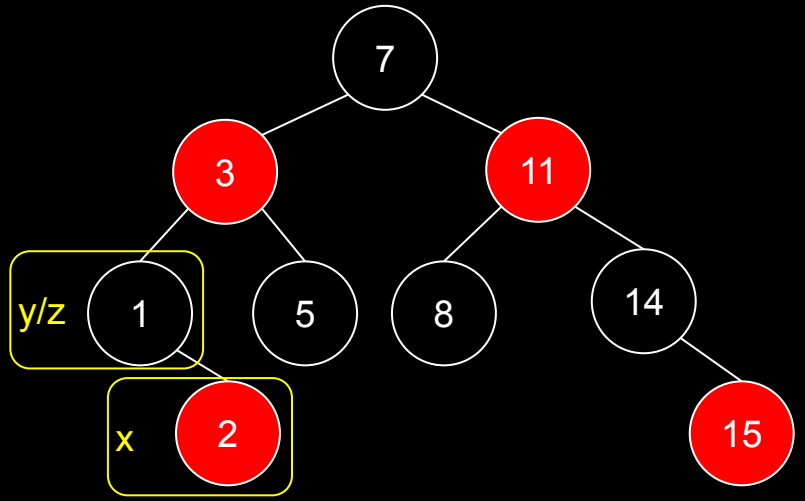
corOriginal

preto

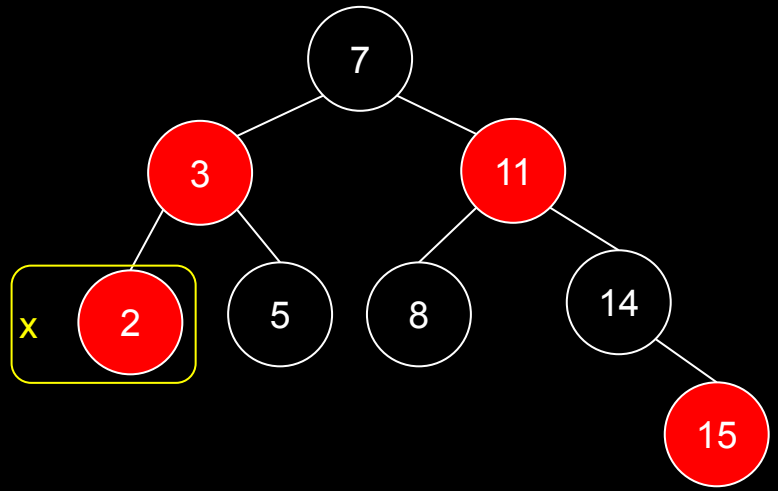
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```

corOriginal

preto



```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



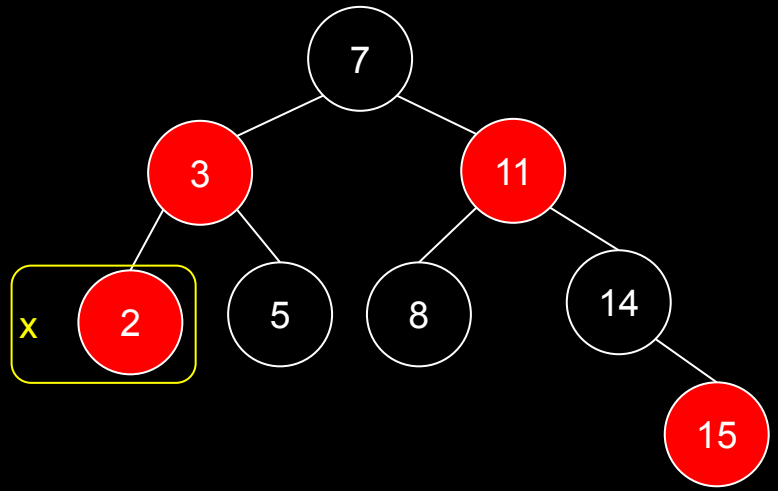
corOriginal

preto

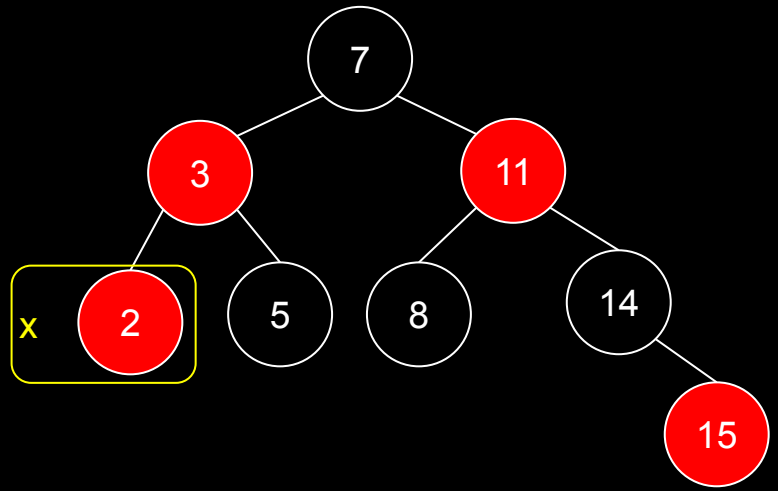
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```

corOriginal

preto



```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```

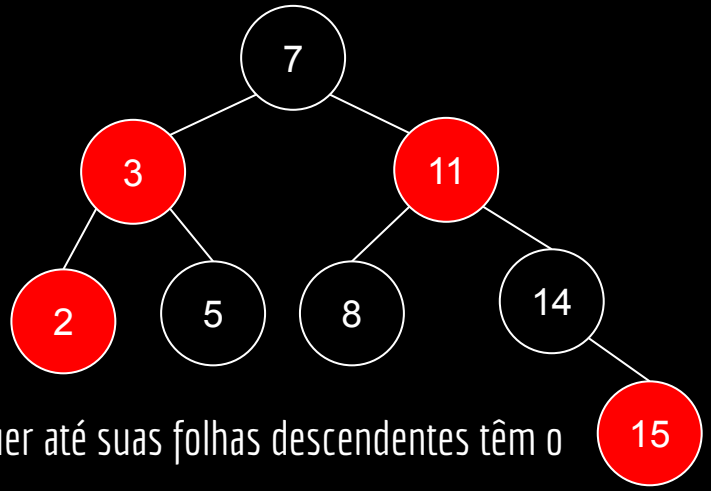


corOriginal

preto

Propriedades

Quais propriedades foram violadas, que agora precisarão ser corrigidas?

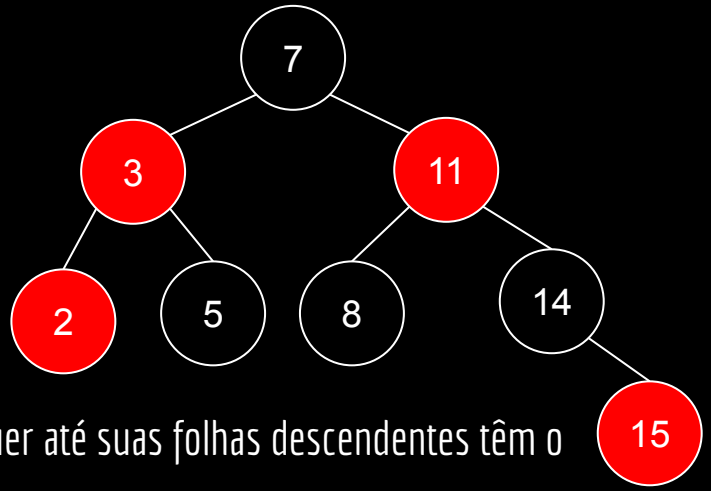


1. Todo nodo é vermelho ou preto;
2. A raiz é preta;
3. Toda folha é um sentinela preto;
4. Se um nodo é vermelho, ambos filhos são pretos;
5. Todos caminhos simples (sem repetição de nodos) de um nodo qualquer até suas folhas descendentes têm o mesmo número de nodos pretos.

Propriedades

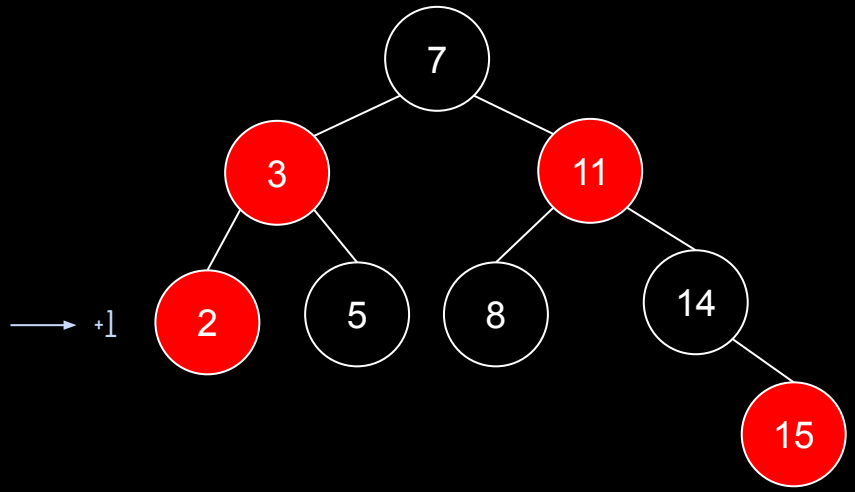
Quais propriedades foram violadas, que agora precisarão ser corrigidas?

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.



Propriedades

O lado esquerdo ficou 1 nodo preto mais curto. Vamos precisar colocar esse 1 em algum nodo na hierarquia. Por enquanto, vai ficar destacado aqui.

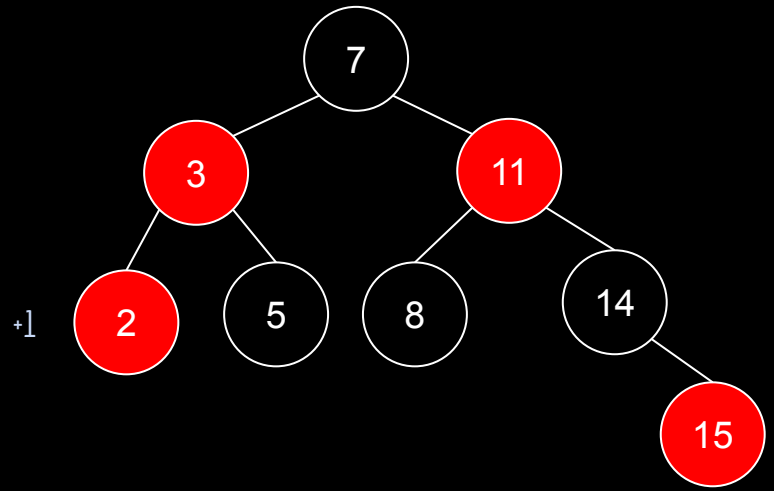


redBlackDeleteFixup

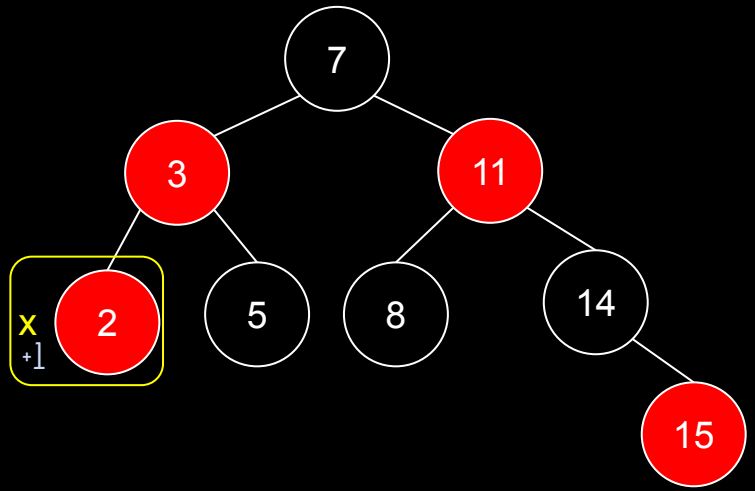
```
função redBlackDeleteFixup(T,x)
entrada: nodo x que tomou a posição do nodo excluído na árvore T
saída: a árvore é modificada de forma a manter as propriedades
da red-black.
enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            w = x.pai.fd
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoEsquerda(T,x.pai)
                  w = x.pai.fd
            se w.fe.cor == preto e w.fd.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fd.cor == preto
                        w.fe.cor = preto
                        w.cor = vermelho
                        rotacaoDireita(T,w)
                        w = x.pai.fd
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fd.cor = preto
                  rotacaoEsquerda(T,x.pai)
                  x = T.raiz
      senão
            //o mesmo que o se, mas espelhado
x.cor = preto
```

redBlackDeleteFixup(T,2)

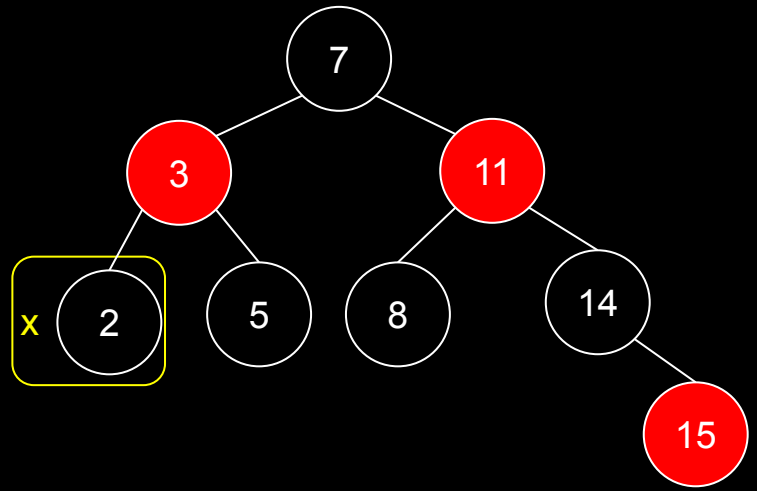
```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            w = x.pai.fd
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoEsquerda(T,x.pai)
                  w = x.pai.fd
            se w.fe.cor == preto e w.fd.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fd.cor == preto
                        w.fe.cor = preto
                        w.cor = vermelho
                        rotacaoDireita(T,w)
                        w = x.pai.fd
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fd.cor = preto
                  rotacaoEsquerda(T,x.pai)
                  x = T.raiz
      senão
            //o mesmo que o se, mas espelhado
x.cor = preto
```



```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            w = x.pai.fd
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoEsquerda(T,x.pai)
                  w = x.pai.fd
            se w.fe.cor == preto e w.fd.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fd.cor == preto
                        w.fe.cor = preto
                        w.cor = vermelho
                        rotacaoDireita(T,w)
                        w = x.pai.fd
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fd.cor = preto
                  rotacaoEsquerda(T,x.pai)
                  x = T.raiz
      senão
            //o mesmo que o se, mas espelhado
x.cor = preto
```



```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            w = x.pai.fd
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoEsquerda(T,x.pai)
                  w = x.pai.fd
            se w.fe.cor == preto e w.fd.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fd.cor == preto
                        w.fe.cor = preto
                        w.cor = vermelho
                        rotacaoDireita(T,w)
                        w = x.pai.fd
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fd.cor = preto
                  rotacaoEsquerda(T,x.pai)
                  x = T.raiz
      senão
            //o mesmo que o se, mas espelhado
x.cor = preto
```

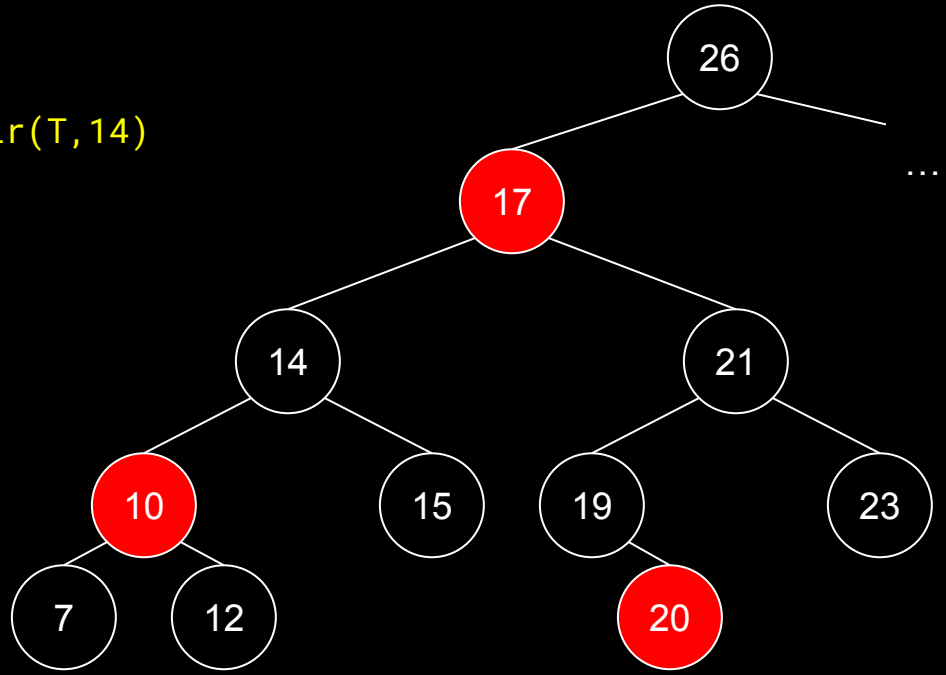


Outro exemplo

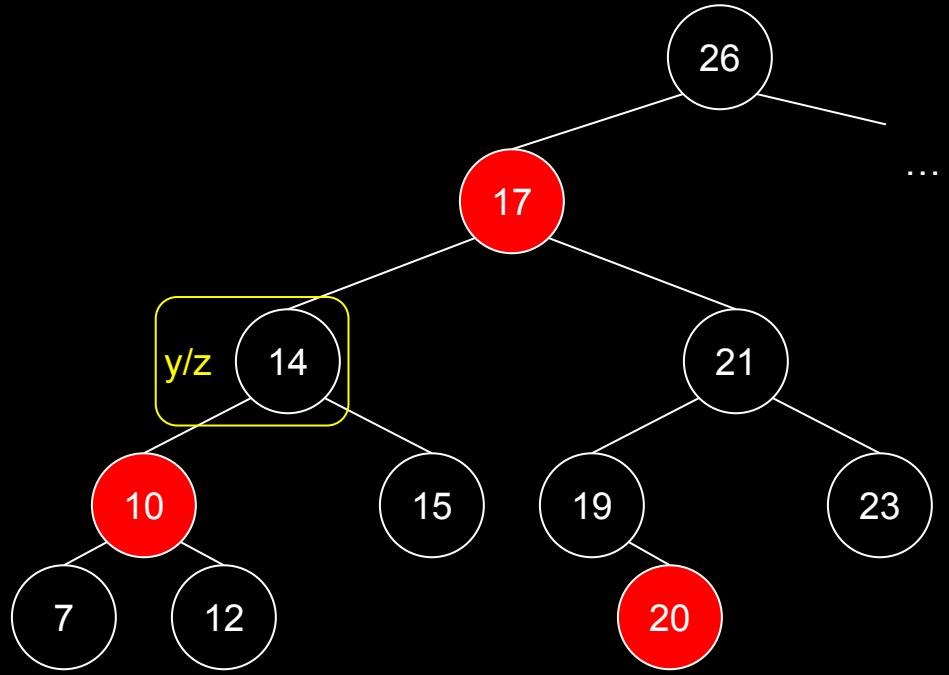
Vamos a outro exemplo...

```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```

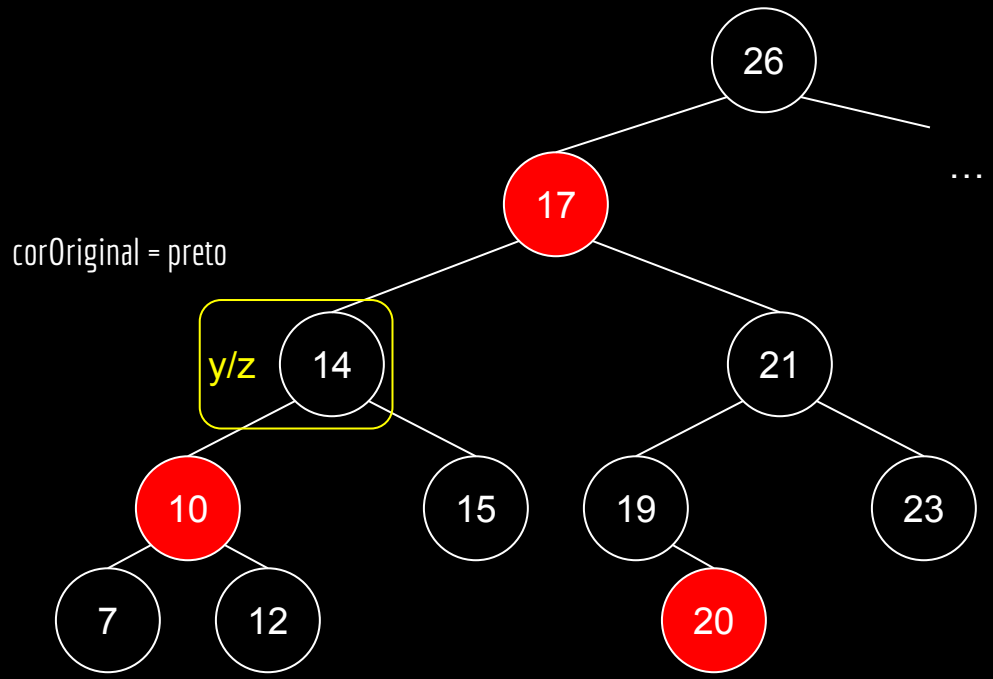
rbExcluir(T,14)



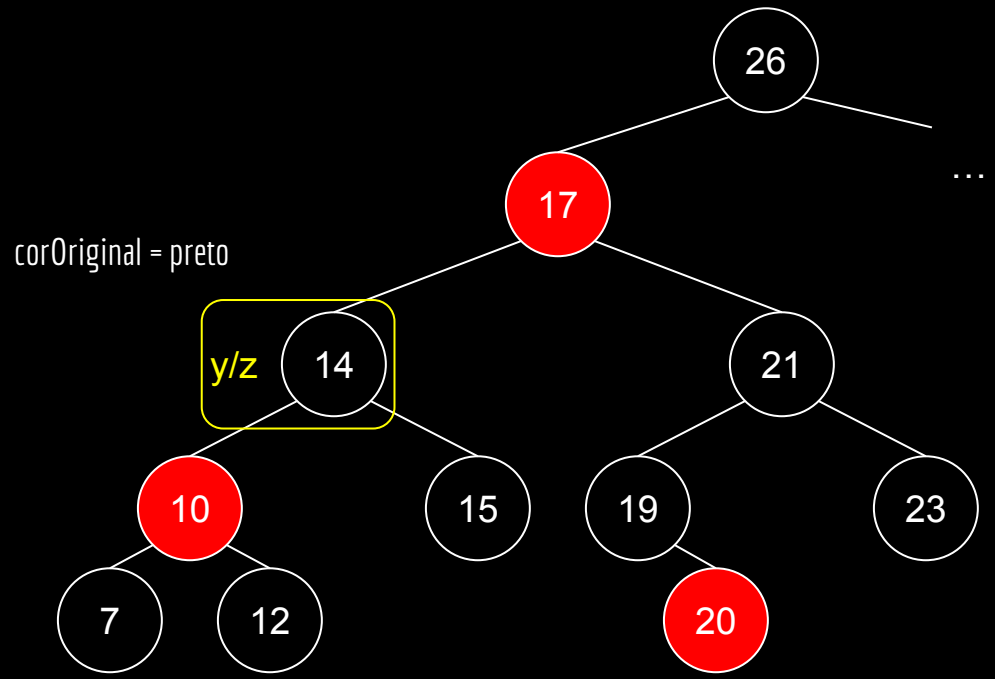
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



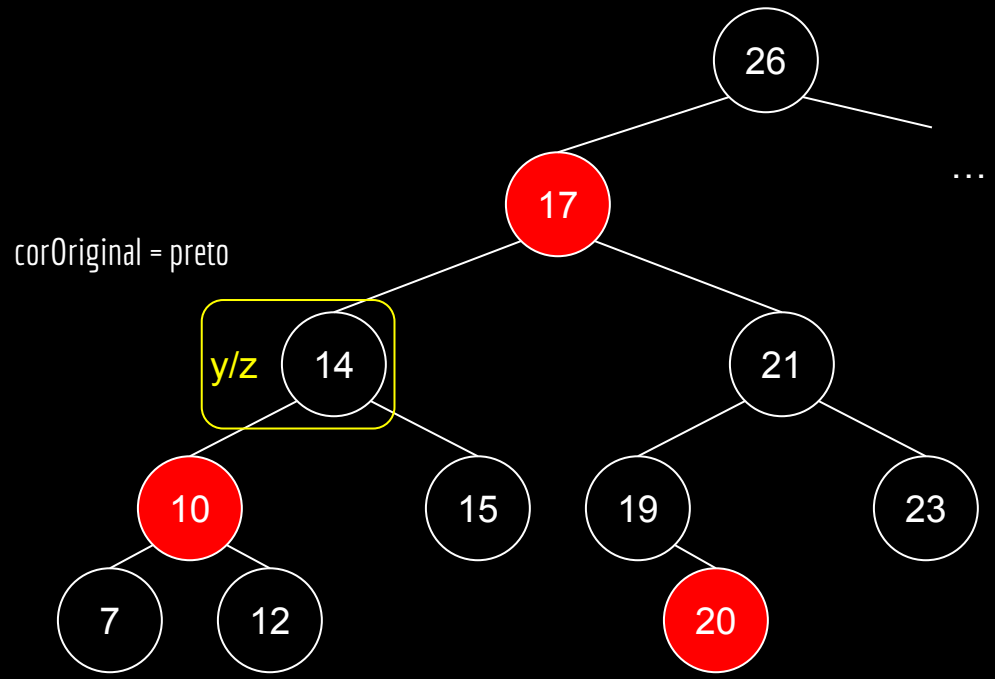
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



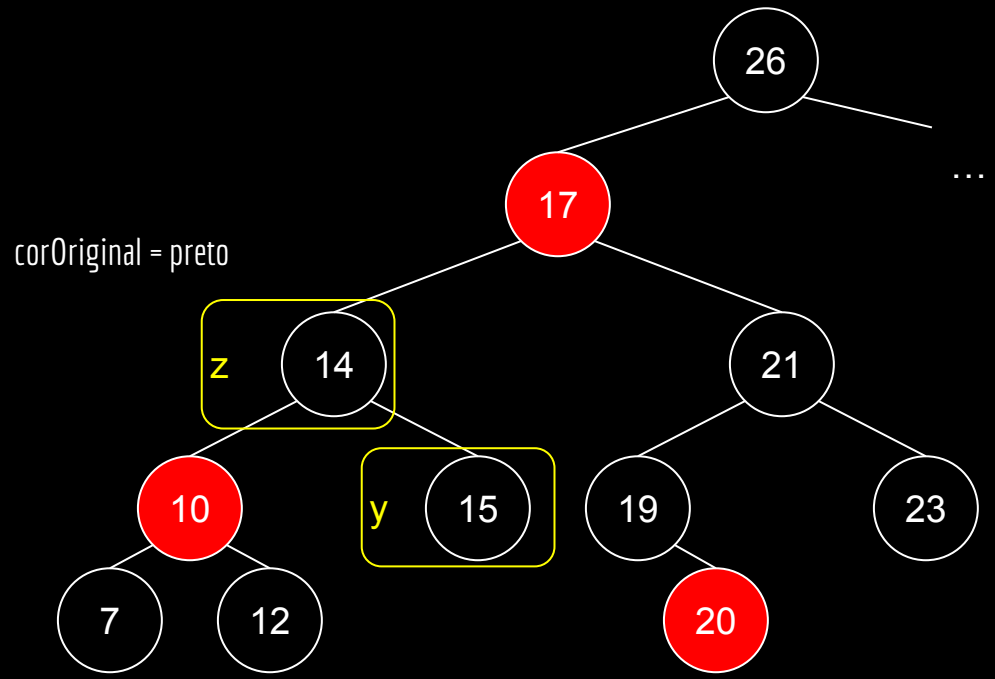
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



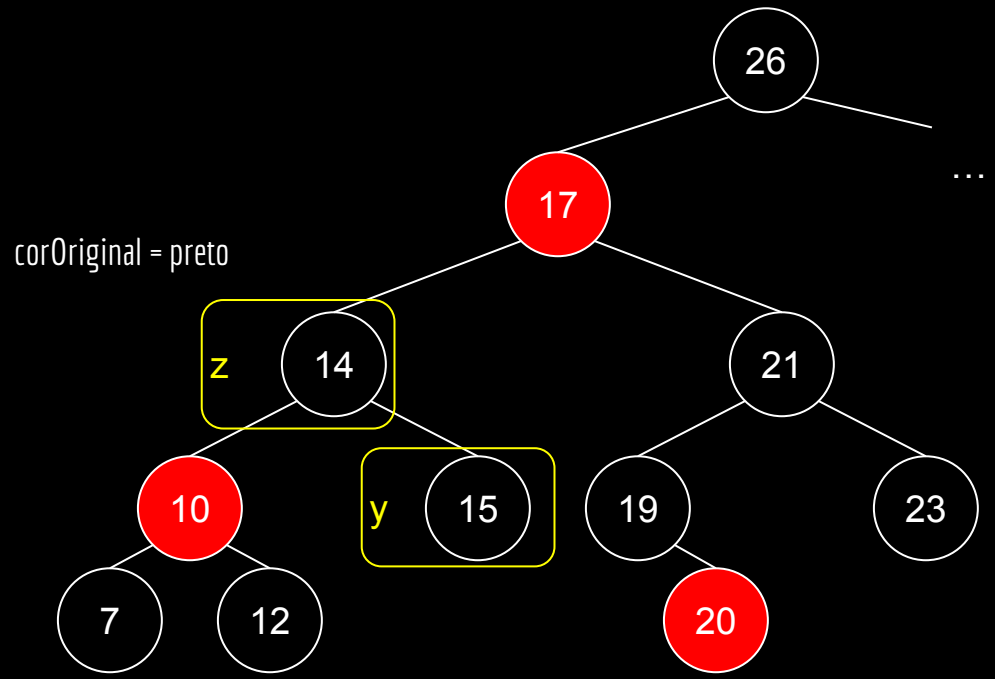
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



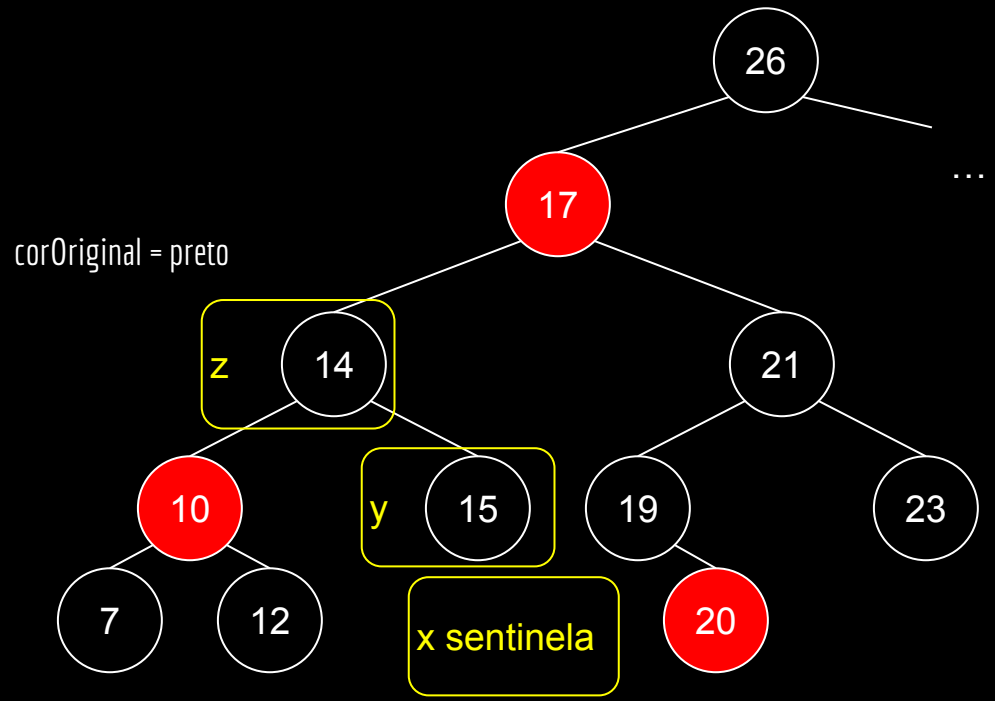
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



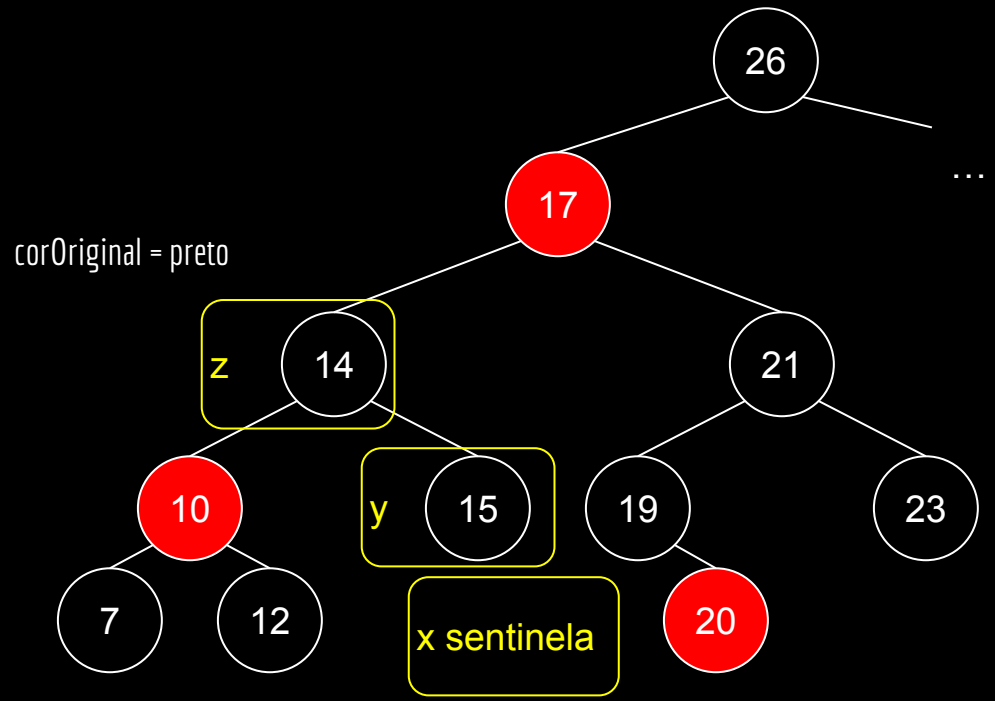
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



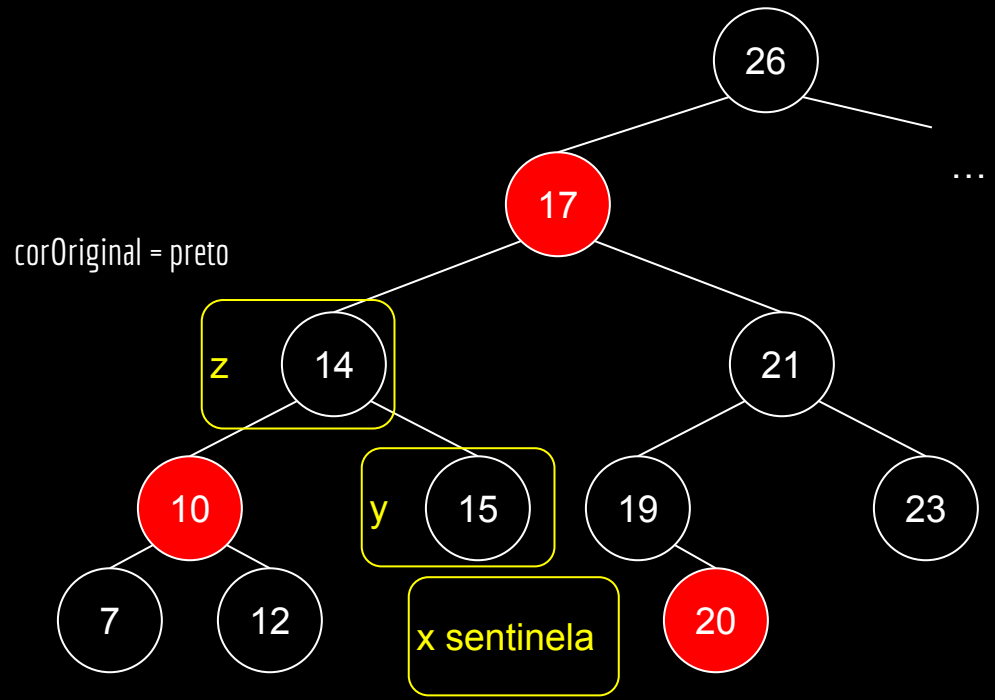
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



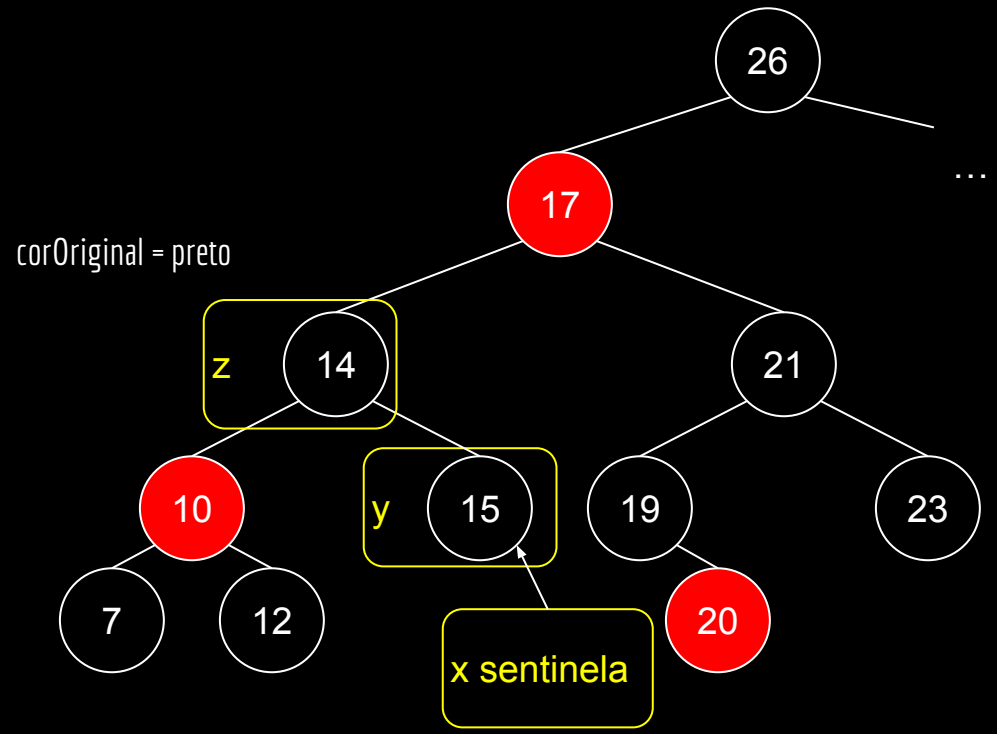
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```

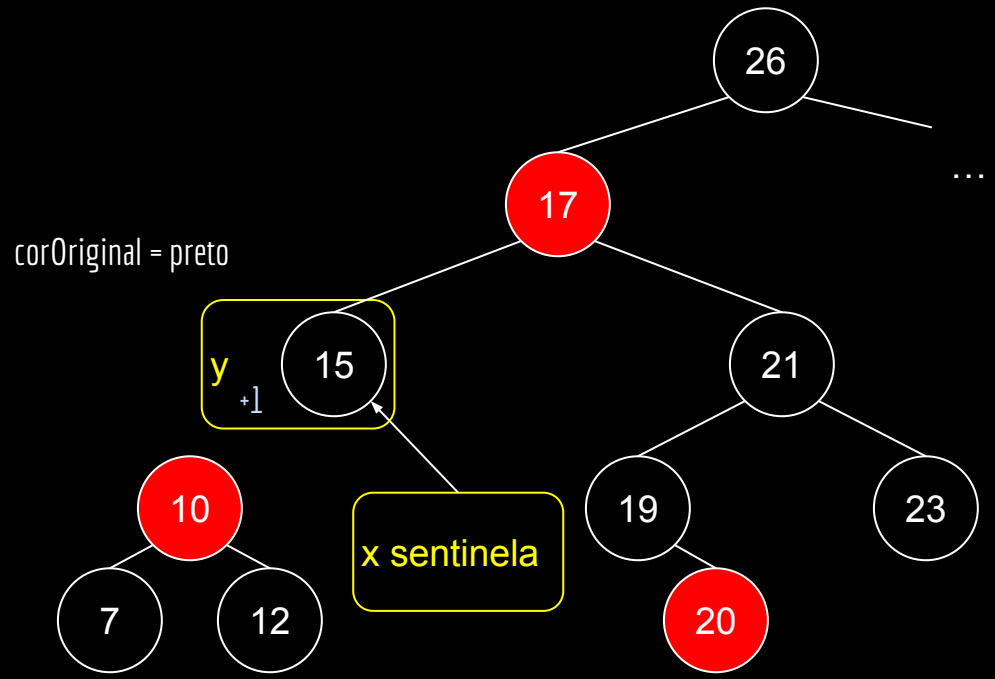


```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```

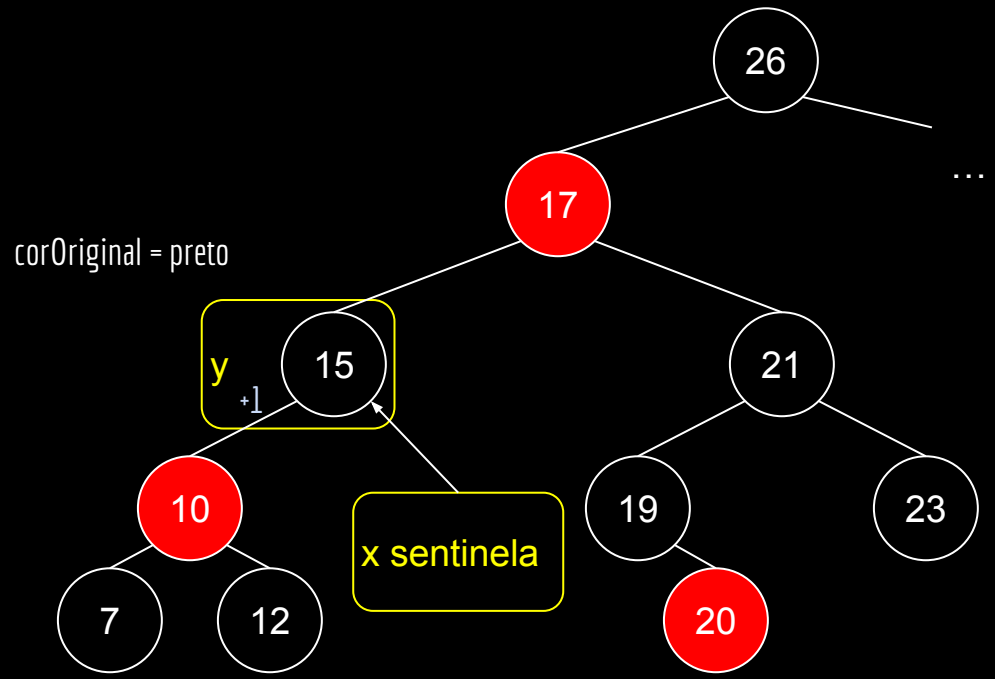


Parece estranho. Mas caso x seja um sentinela, precisamos garantir que ele aponta para y como pai nesse momento.

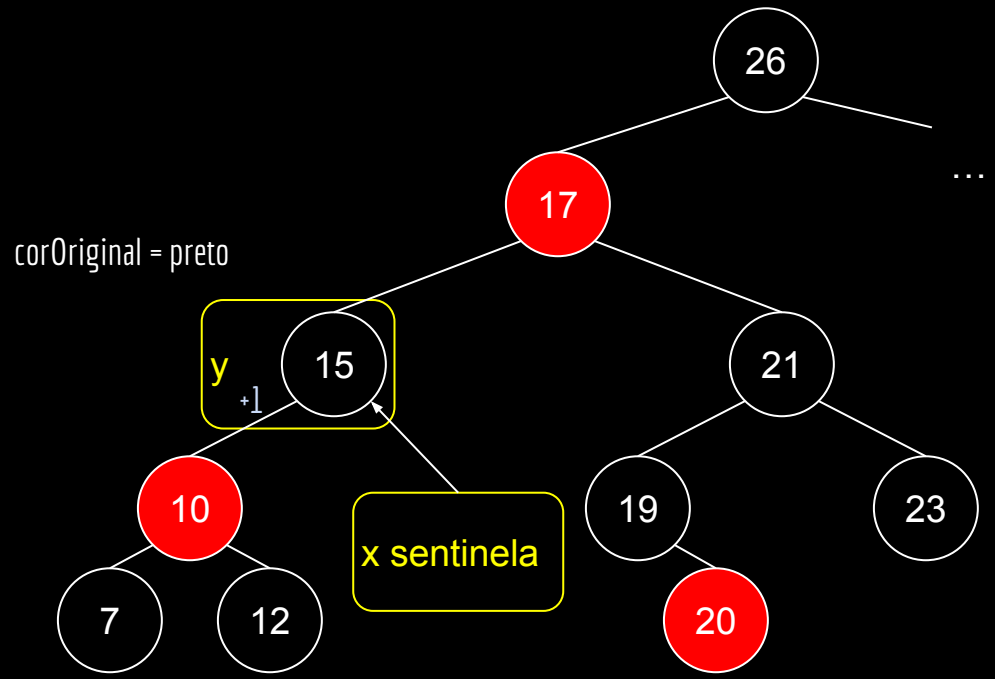
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



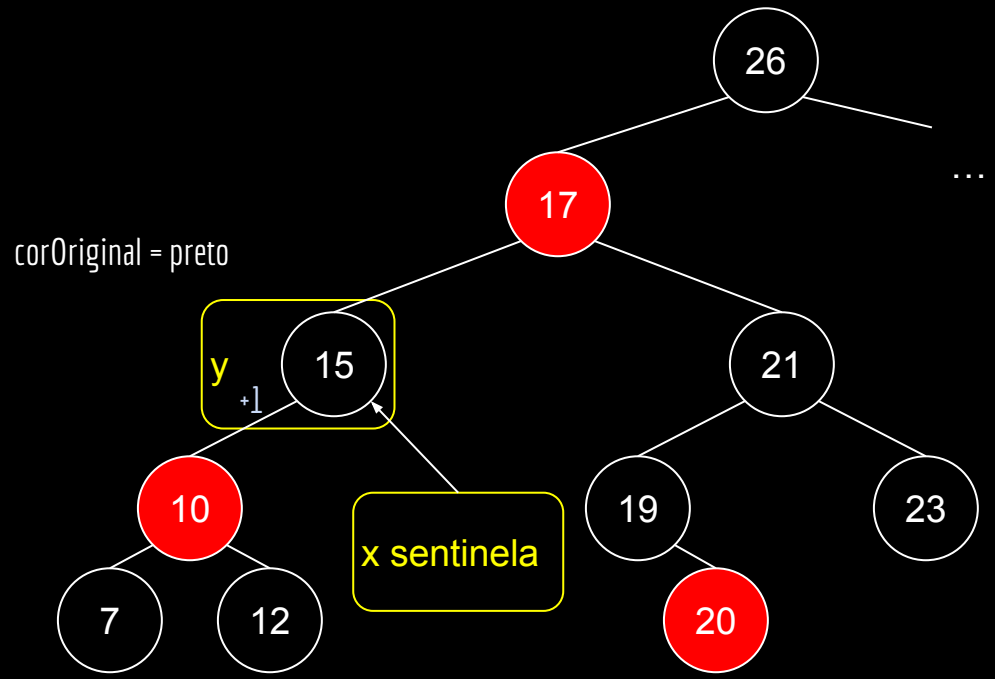
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



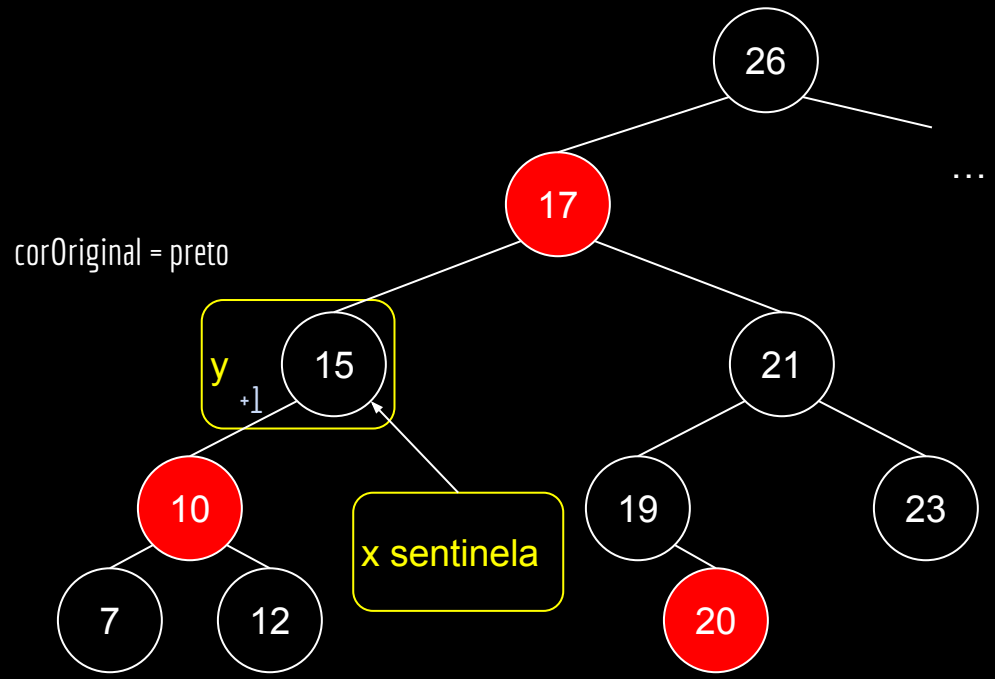
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



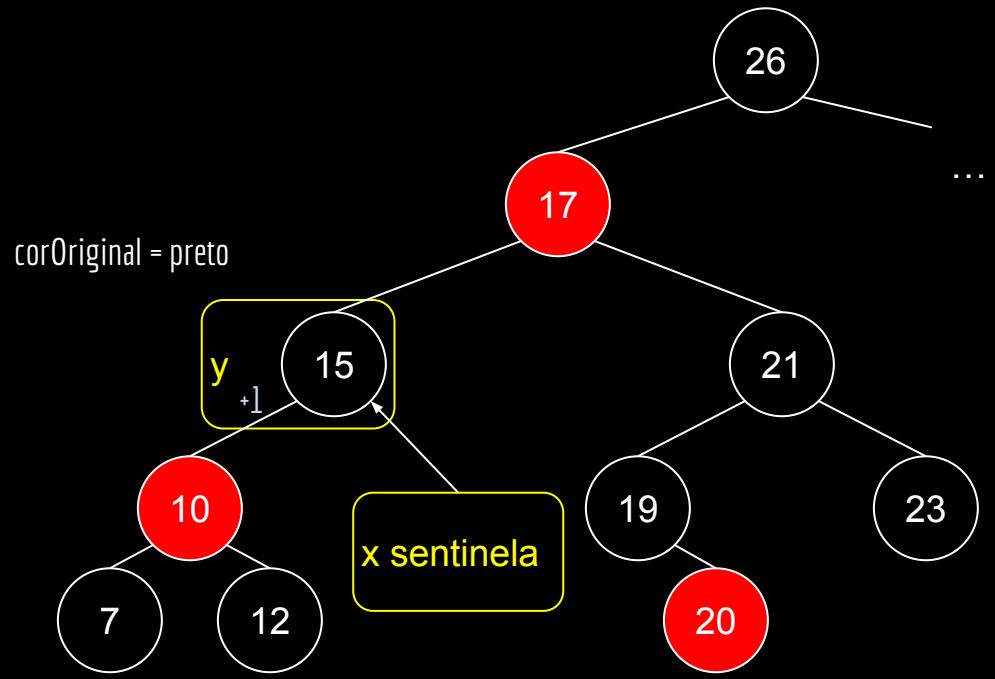
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



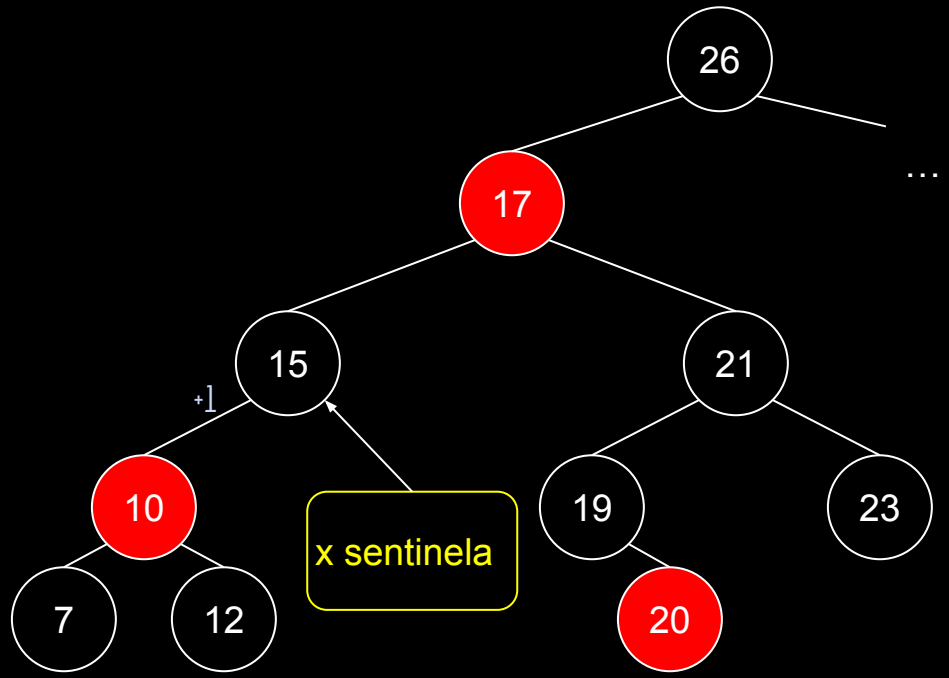
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



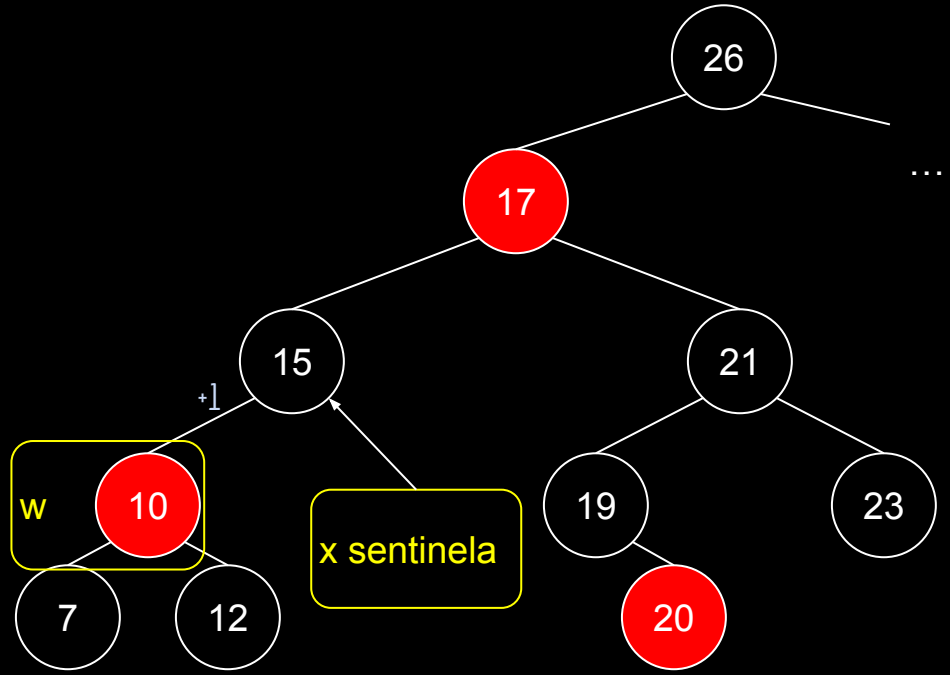
```
função rbExcluir(T,z)

y = z
corOriginal = y.cor
se z.fe == sentinela
      x = z.fd
      rbTransplantar(T,z,z.fd)
senão
      se z.fd == sentinela
            x = z.fe
            rbTransplantar(T,z,z.fe)
      senão
            y = minimo(z.fd)
            corOriginal = y.cor
            x = y.fd
            se y ≠ z.fd
                  rbTransplantar(T,y,y.fd)
                  y.fd = z.fd
                  y.fd.pai = y
            senão
                  x.pai = y
            rbTransplantar(T,z,y)
            y.fe = z.fe
            y.fe.pai = y
            y.cor = z.cor
se corOriginal == preto
      redBlackDeleteFixup(T,x)
```



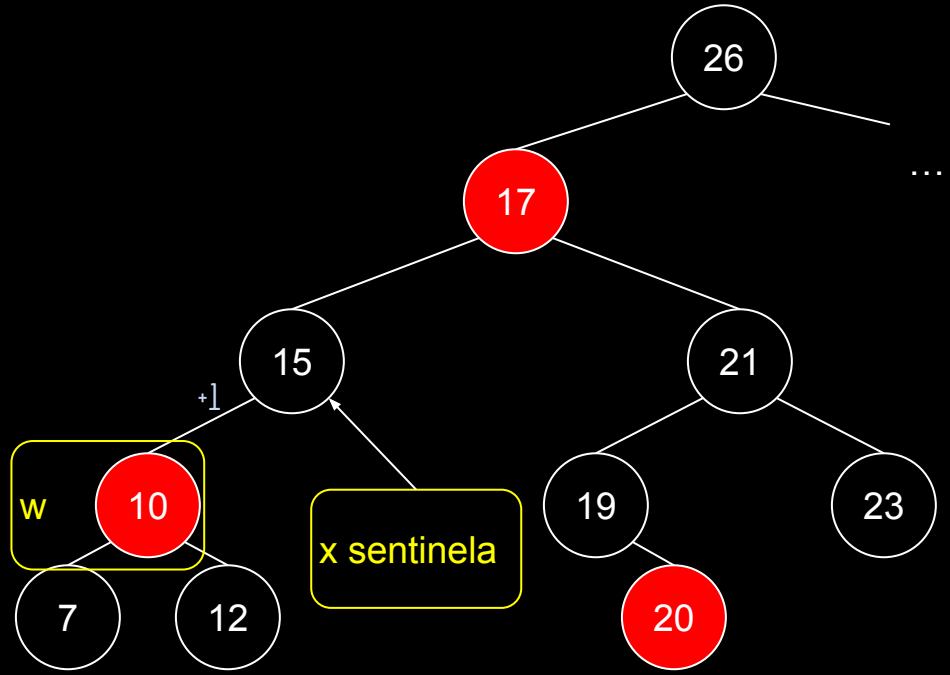
```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
     se x == x.pai.fe //era um filho esquerdo?
          //espelho do senão
     senão
          w = x.pai.fe
          se w.cor == vermelho
               w.cor = preto
               x.pai.cor = vermelho
               rotacaoDireita(T,x.pai)
               w = x.pai.fe
          se w.fd.cor == preto e w.fe.cor == preto
               w.cor = vermelho
               x = x.pai
          senão
               se w.fe.cor == preto
                    w.fd.cor = preto
                    w.cor = vermelho
                    rotacaoEsquerda(T,w)
                    w = x.pai.fe
               w.cor = x.pai.cor
               x.pai.cor = preto
               w.fe.cor = preto
               rotacaoDireita(T,x.pai)
               x = T.raiz
x.cor = preto
```



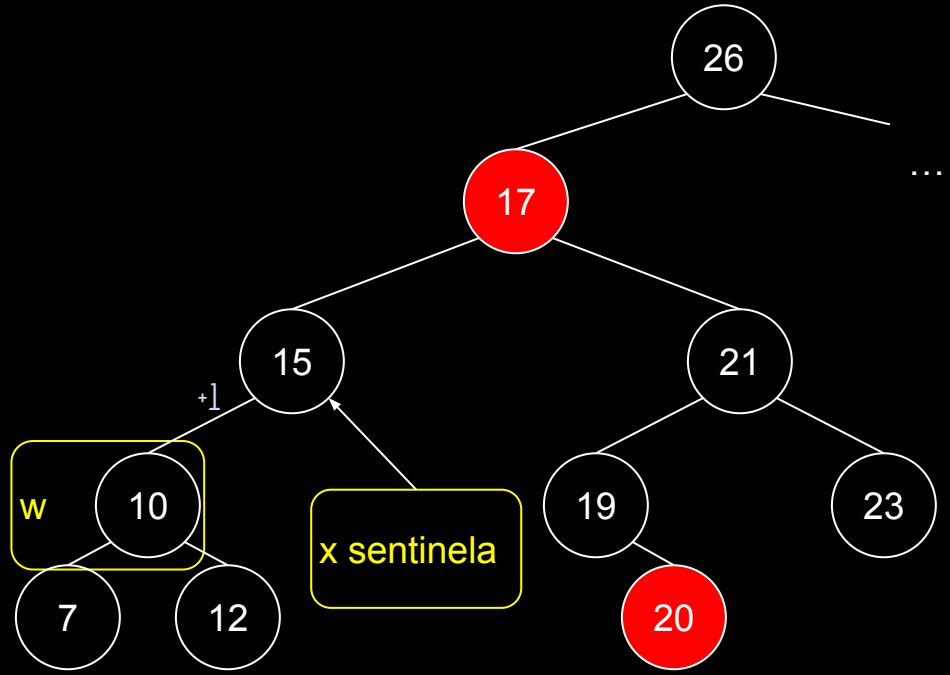
```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            //espelho do senão
      senão
            w = x.pai.fe
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoDireita(T,x.pai)
                  w = x.pai.fe
            se w.fd.cor == preto e w.fe.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fe.cor == preto
                        w.fd.cor = preto
                        w.cor = vermelho
                        rotacaoEsquerda(T,w)
                        w = x.pai.fe
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fe.cor = preto
                  rotacaoDireita(T,x.pai)
                  x = T.raiz
x.cor = preto
```



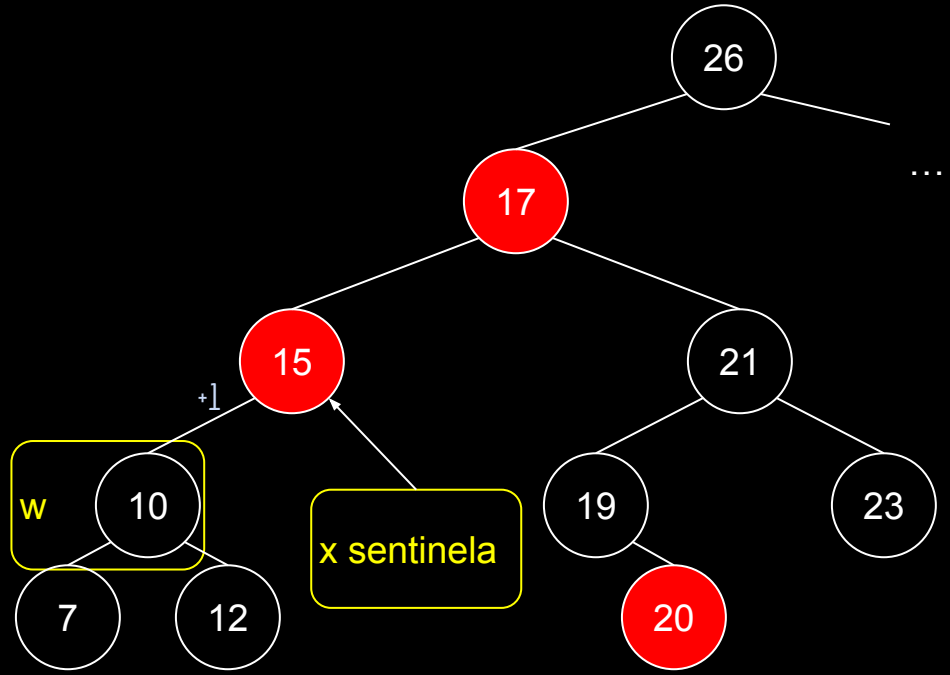
```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            //espelho do senão
      senão
            w = x.pai.fe
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoDireita(T,x.pai)
                  w = x.pai.fe
            se w.fd.cor == preto e w.fe.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fe.cor == preto
                        w.fd.cor = preto
                        w.cor = vermelho
                        rotacaoEsquerda(T,w)
                        w = x.pai.fe
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fe.cor = preto
                  rotacaoDireita(T,x.pai)
                  x = T.raiz
x.cor = preto
```



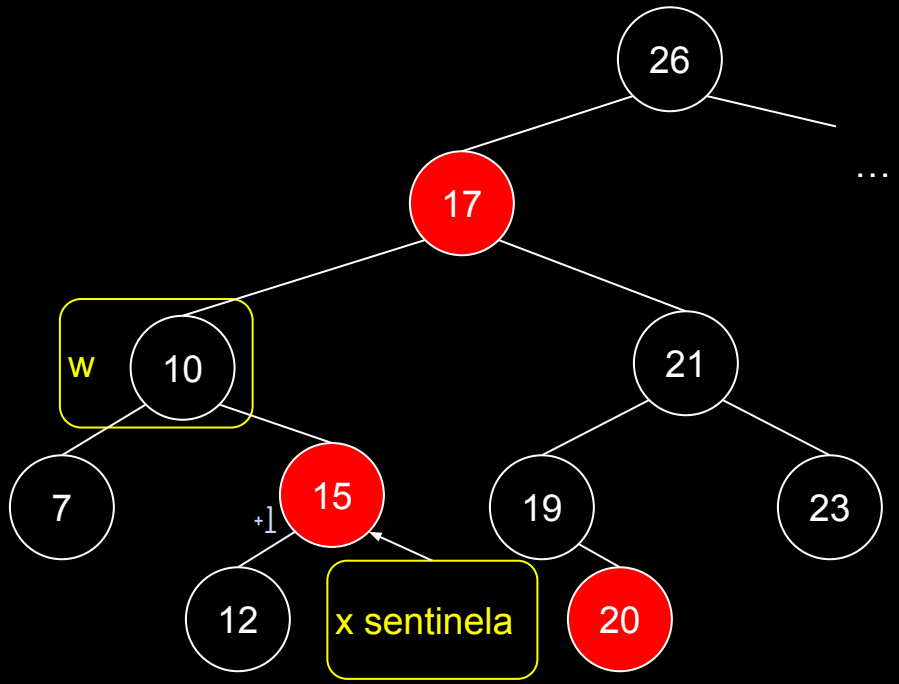
```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            //espelho do senão
      senão
            w = x.pai.fe
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoDireita(T,x.pai)
                  w = x.pai.fe
            se w.fd.cor == preto e w.fe.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fe.cor == preto
                        w.fd.cor = preto
                        w.cor = vermelho
                        rotacaoEsquerda(T,w)
                        w = x.pai.fe
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fe.cor = preto
                  rotacaoDireita(T,x.pai)
                  x = T.raiz
x.cor = preto
```



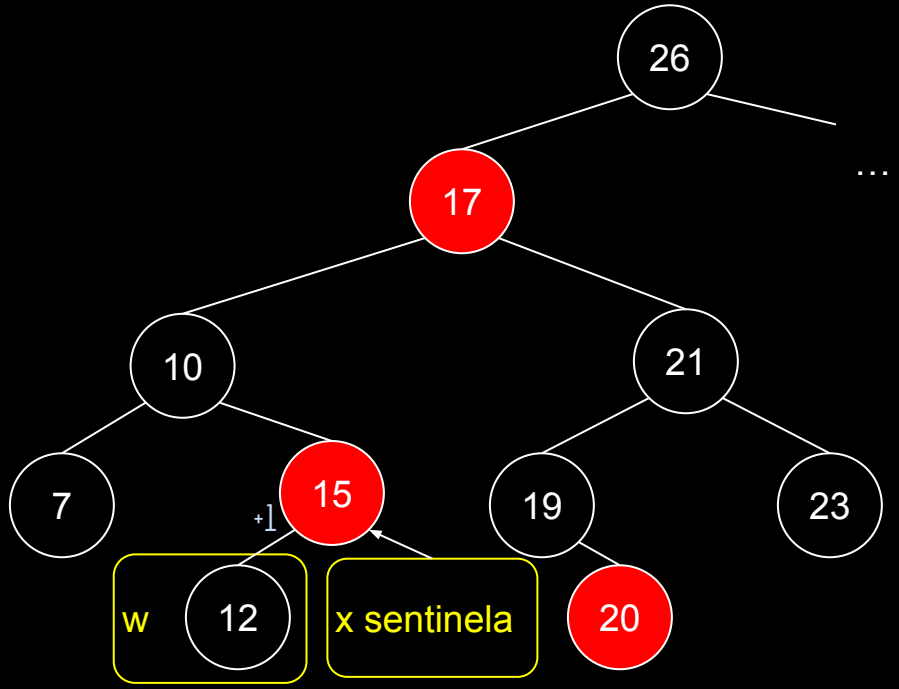
```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            //espelho do senão
      senão
            w = x.pai.fe
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoDireita(T,x.pai)
                  w = x.pai.fe
            se w.fd.cor == preto e w.fe.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fe.cor == preto
                        w.fd.cor = preto
                        w.cor = vermelho
                        rotacaoEsquerda(T,w)
                        w = x.pai.fe
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fe.cor = preto
                  rotacaoDireita(T,x.pai)
                  x = T.raiz
x.cor = preto
```



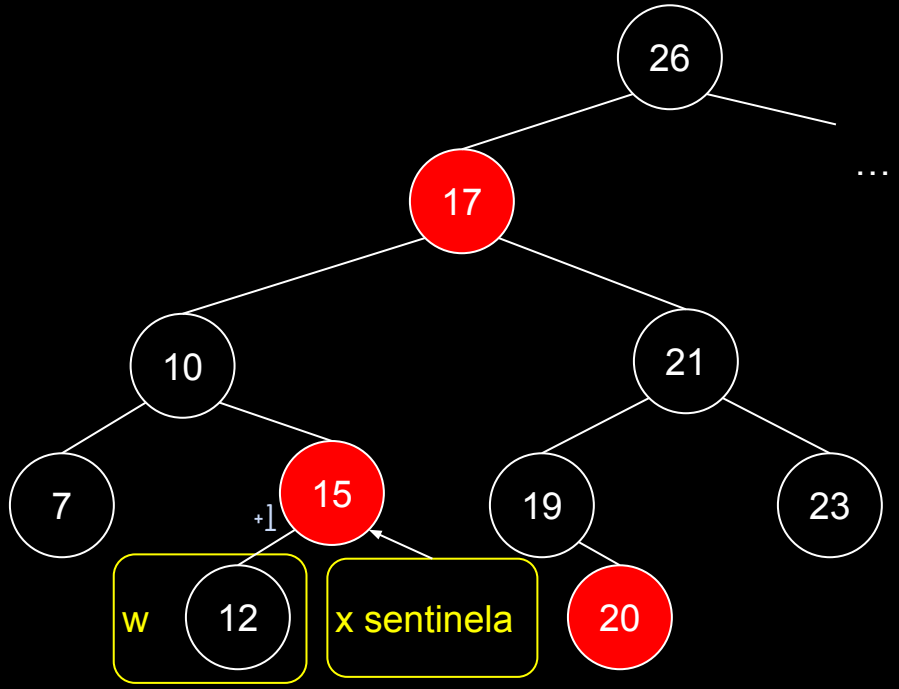
```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            //espelho do senão
      senão
            w = x.pai.fe
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoDireita(T,x.pai)
                  w = x.pai.fe
            se w.fd.cor == preto e w.fe.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fe.cor == preto
                        w.fd.cor = preto
                        w.cor = vermelho
                        rotacaoEsquerda(T,w)
                        w = x.pai.fe
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fe.cor = preto
                  rotacaoDireita(T,x.pai)
                  x = T.raiz
x.cor = preto
```



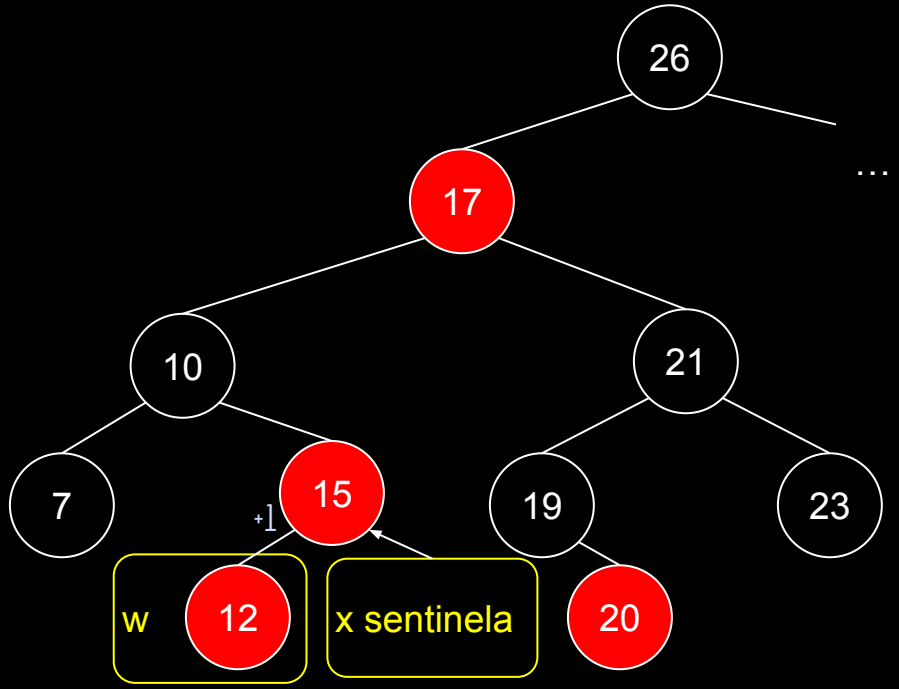
```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            //espelho do senão
      senão
            w = x.pai.fe
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoDireita(T,x.pai)
                  w = x.pai.fe
            se w.fd.cor == preto e w.fe.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fe.cor == preto
                        w.fd.cor = preto
                        w.cor = vermelho
                        rotacaoEsquerda(T,w)
                        w = x.pai.fe
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fe.cor = preto
                  rotacaoDireita(T,x.pai)
                  x = T.raiz
x.cor = preto
```



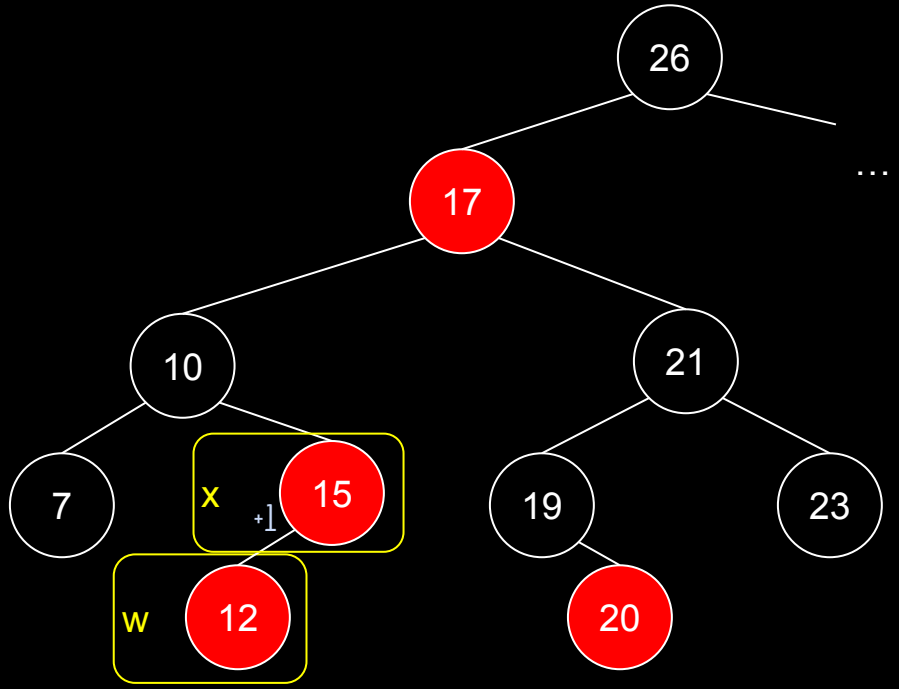
```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            //espelho do senão
      senão
            w = x.pai.fe
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoDireita(T,x.pai)
                  w = x.pai.fe
            se w.fd.cor == preto e w.fe.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fe.cor == preto
                        w.fd.cor = preto
                        w.cor = vermelho
                        rotacaoEsquerda(T,w)
                        w = x.pai.fe
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fe.cor = preto
                  rotacaoDireita(T,x.pai)
                  x = T.raiz
x.cor = preto
```



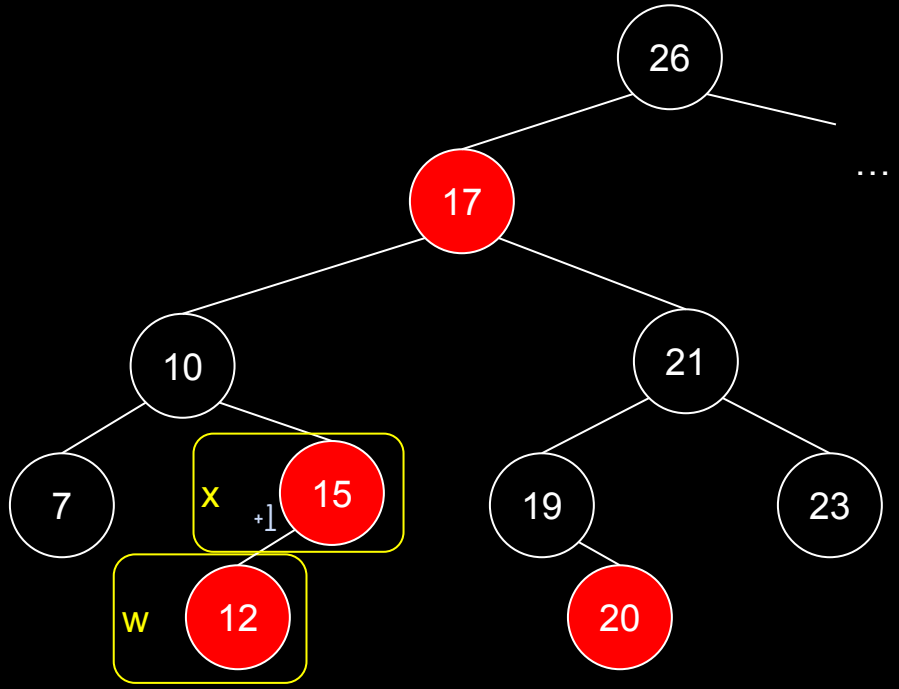
```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            //espelho do senão
      senão
            w = x.pai.fe
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoDireita(T,x.pai)
                  w = x.pai.fe
            se w.fd.cor == preto e w.fe.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fe.cor == preto
                        w.fd.cor = preto
                        w.cor = vermelho
                        rotacaoEsquerda(T,w)
                        w = x.pai.fe
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fe.cor = preto
                  rotacaoDireita(T,x.pai)
                  x = T.raiz
x.cor = preto
```



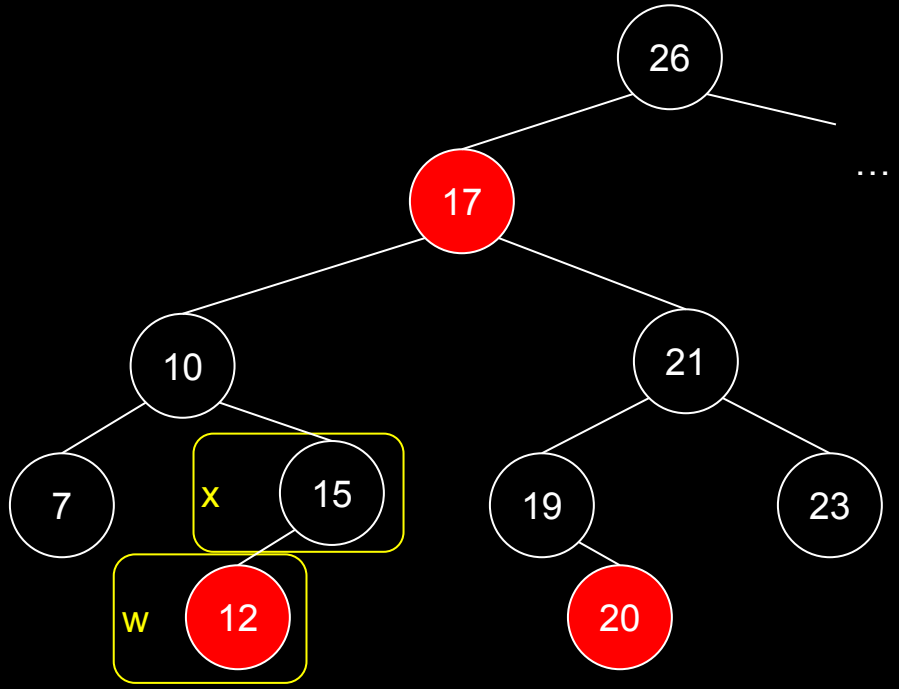
```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            //espelho do senão
      senão
            w = x.pai.fe
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoDireita(T,x.pai)
                  w = x.pai.fe
            se w.fd.cor == preto e w.fe.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fe.cor == preto
                        w.fd.cor = preto
                        w.cor = vermelho
                        rotacaoEsquerda(T,w)
                        w = x.pai.fe
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fe.cor = preto
                  rotacaoDireita(T,x.pai)
                  x = T.raiz
x.cor = preto
```



```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            //espelho do senão
      senão
            w = x.pai.fe
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoDireita(T,x.pai)
                  w = x.pai.fe
            se w.fd.cor == preto e w.fe.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fe.cor == preto
                        w.fd.cor = preto
                        w.cor = vermelho
                        rotacaoEsquerda(T,w)
                        w = x.pai.fe
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fe.cor = preto
                  rotacaoDireita(T,x.pai)
                  x = T.raiz
x.cor = preto
```



```
função redBlackDeleteFixup(T,x)

enquanto x ≠ T.raiz e x.cor == preto
      se x == x.pai.fe //era um filho esquerdo?
            //espelho do senão
      senão
            w = x.pai.fe
            se w.cor == vermelho
                  w.cor = preto
                  x.pai.cor = vermelho
                  rotacaoDireita(T,x.pai)
                  w = x.pai.fe
            se w.fd.cor == preto e w.fe.cor == preto
                  w.cor = vermelho
                  x = x.pai
            senão
                  se w.fe.cor == preto
                        w.fd.cor = preto
                        w.cor = vermelho
                        rotacaoEsquerda(T,w)
                        w = x.pai.fe
                  w.cor = x.pai.cor
                  x.pai.cor = preto
                  w.fe.cor = preto
                  rotacaoDireita(T,x.pai)
                  x = T.raiz
x.cor = preto
```



Exemplo em C++

Veja mais em en.cppreference.com/w/cpp/container/map

`std::map` is a sorted associative container that contains key-value pairs with unique keys. Keys are sorted by using the comparison function `Compare`. Search, removal, and insertion operations have logarithmic complexity. Maps are usually implemented as Red–black trees.

```cpp
#include <map>
#include <string>
#include <iostream>

struct pessoa{
 std::string nome;
 unsigned char idade;
};

int main(){
   //dicionario onde as chaves são ulongs, e os dados satélite são pessoas
   std::map<unsigned long, struct pessoa> dicionario;
   dicionario[1234] = pessoa{"Joao", 20};
   dicionario[5678] = pessoa{"Maria", 19};

   std::cout << dicionario[1234].nome << "\n";

   return 0;
}
```

Para compilar:

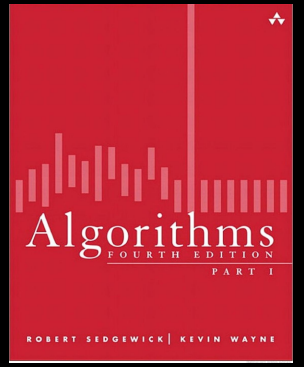
```
g++ exemplo.cpp
```

Exercícios

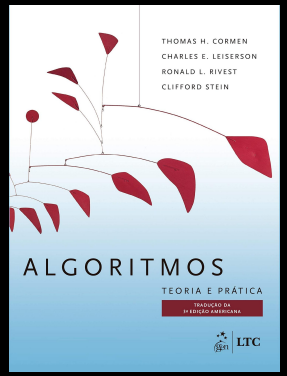
1. Na função `rbExcluir`, nada precisa ser feito se a cor original era vermelha. Por quê? Argumente sobre como nesse caso nenhuma propriedade da red-black é violada. Compare sua resposta com a discussão em Cormen et al.
2. Implemente os algoritmos discutidos em C.

Referências

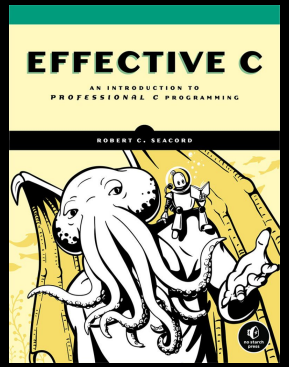
R. Sedgewick, K. Wayne. Algorithms Part I. 4a ed. 2011.



T. Cormen, C. Leiserson, R. Rivest, C. Stein. Algoritmos: Teoria e Prática. 3a ed. 2012.



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



Licença