Estruturas de Dados

Cristina Gomes Fernandes

ABBs rubro-negras

Uma ABB é rubro-**negra** se

1. todo nó é rubro ou **negro**
2. toda folha (NIL) é **negra**
3. se um nó é rubro, então seus dois filhos são **negros**
4. todo caminho de um nó x até uma folha sua descendente tem o mesmo número de nós **negros**.

ABBs rubro-negras

Uma ABB é rubro-**negra** se

1. todo nó é rubro ou **negro**
2. toda folha (NIL) é **negra**
3. se um nó é rubro, então seus dois filhos são **negros**
4. todo caminho de um nó x até uma folha sua descendente tem o mesmo número de nós **negros**.

$rn(x)$: número de nós **negros** no caminho de um filho de x até uma folha descendente de x.

ABBs rubro-negras

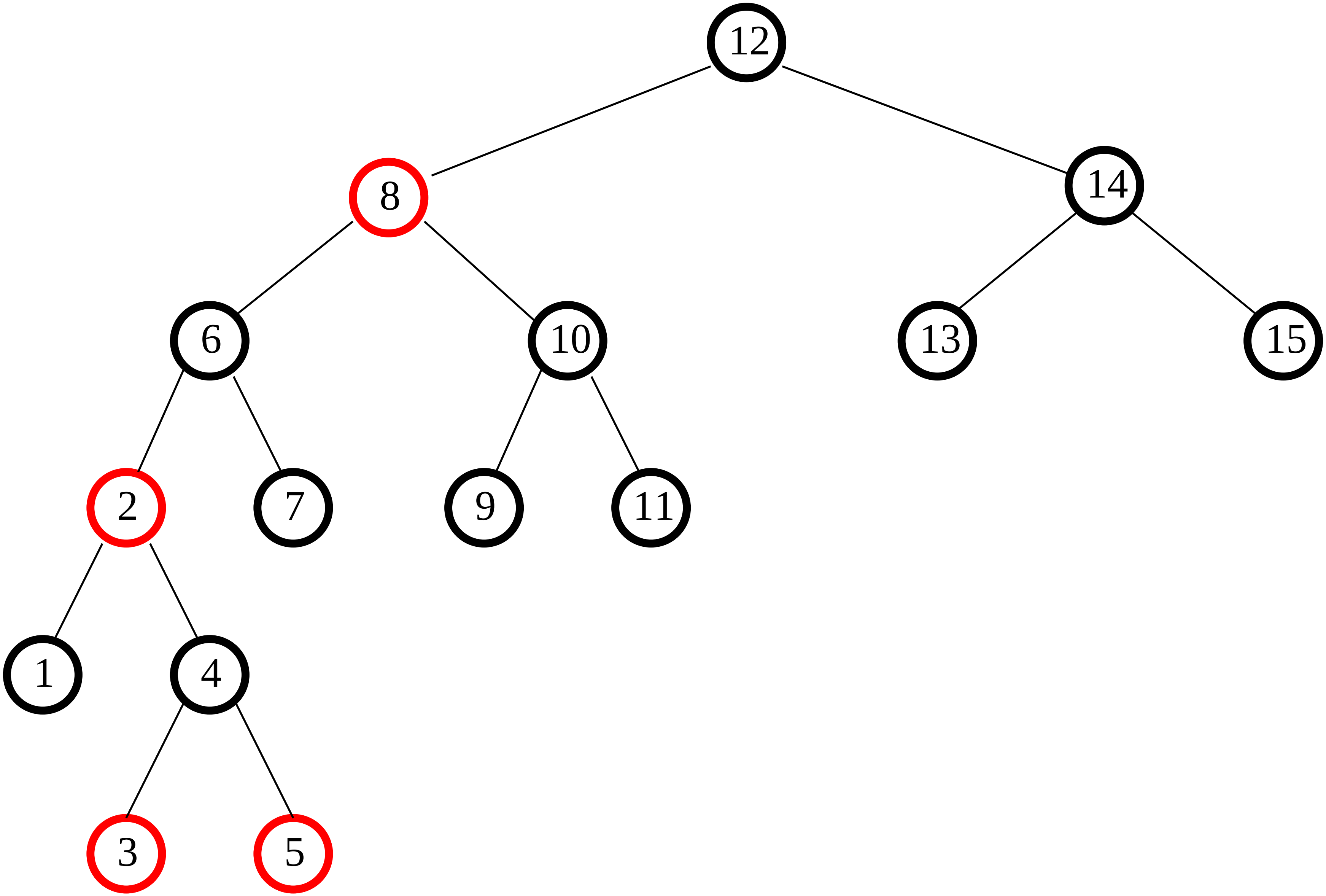
Uma ABB é rubro-**negra** se

1. todo nó é rubro ou **negro**
2. toda folha (NIL) é **negra**
3. se um nó é rubro, então seus dois filhos são **negros**
4. todo caminho de um nó x até uma folha sua descendente tem o mesmo número de nós **negros.**

$rn(x)$: número de nós **negros** no caminho de um filho de x até uma folha descendente de x.

Lema: Uma ABB rubro-**negra** com n nós internos tem altura no máximo $2\lg(n+1)$.

Exemplo de ABB rubro-negra



Rotações

O nó p é tal que $dir(p) \neq \text{NIL}$ e $cor(dir(p)) = $ RUBRO.

ROTACIONEESQ (p)

1 $q \leftarrow dir(p)$
2 $dir(p) \leftarrow esq(q)$
3 $esq(q) \leftarrow p$
4 $cor(q) \leftarrow cor(p)$
5 $cor(p) \leftarrow$ RUBRO
6 **devolva** q

Rotações

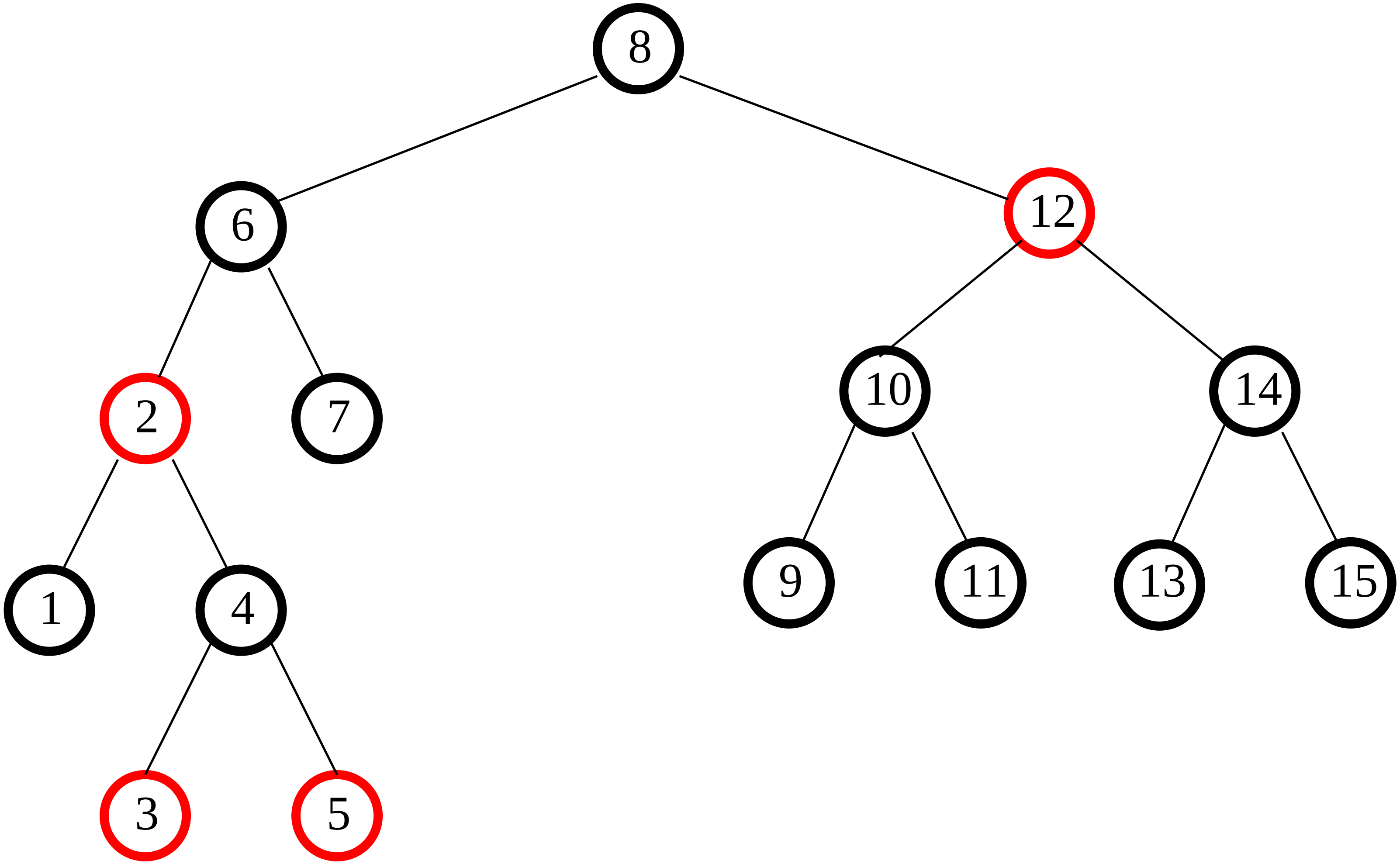
O nó p é tal que $dir(p) \neq \text{NIL}$ e $cor(dir(p)) = $ RUBRO.

ROTACIONEESQ (p)
1 $q \leftarrow dir(p)$
2 $dir(p) \leftarrow esq(q)$
3 $esq(q) \leftarrow p$
4 $cor(q) \leftarrow cor(p)$
5 $cor(p) \leftarrow$ RUBRO
6 **devolva** q

Exercício: Escreva o ROTACIONEDIR.

Exercício: Ajuste estas rotinas para que mantenham o campo pai.

Rotação à direira na raiz



Ajusta cores

O nó p é interno.

TROQUECORES (p)
1 $cor(p) \leftarrow$ OUTRACOR$(cor(p))$
2 $cor(esq(p)) \leftarrow$ OUTRACOR$(cor(esq(p)))$
3 $cor(dir(p)) \leftarrow$ OUTRACOR$(cor(dir(p)))$

Ajusta cores

O nó p é interno.

TROQUECORES (p)
1 $cor(p) \leftarrow$ OUTRACOR$(cor(p))$
2 $cor(esq(p)) \leftarrow$ OUTRACOR$(cor(esq(p)))$
3 $cor(dir(p)) \leftarrow$ OUTRACOR$(cor(dir(p)))$

OUTRACOR (c)
1 **se** $c =$ RUBRO
2 **então devolva** NEGRO
3 **senão devolva** RUBRO

Inserção em ABB rubro-negra

RUBRO (p)

se $p = \text{NIL}$

  então devolva FALSO

  senão se $cor(p) =$ RUBRO

    então devolva VERDADE

    senão devolva FALSO

NEGRO (p)

devolva não RUBRO(p)

Inserção em ABB rubro-negra

```
RUBRO (p)
se p = NIL
    então devolva FALSO
    senão se cor(p) = RUBRO
              então devolva VERDADE
              senão devolva FALSO
```

```
NEGRO (p)
devolva não RUBRO(p)
```

```
INSIRA (T, x)
T ← INSIRAREC(T, x)
cor(T) ← NEGRO      ▷ a raiz é sempre negra
```

Árvores 2-3

Uma árvore 2-3 é uma extensão de uma ABB:

1. todo nó tem uma ou duas chaves, sendo que um nó interno com k chaves tem $k+1$ filhos;
2. todas as folhas estão no mesmo nível;
3. um nó interno com uma única chave c_1 tem como filhos p_1 e p_2, onde p_1 é a raiz de uma subárvore 2-3 cujas chaves são todas menores que c_1, e p_2 é a raiz de uma subárvore 2-3 cujas chaves são todas maiores que c_1.
4. um nó interno com duas chaves c_1 e c_2 tem como filhos p_1, p_2 e p_3, onde p_1 é a raiz de uma subárvore 2-3 cujas chaves são todas menores que c_1, p_2 é a raiz de uma subárvore 2-3 cujas chaves são todas maiores que c_1 e menores que c_2, e p_3 é a raiz de uma subárvore 2-3 cujas chaves são todas maiores que c_2.

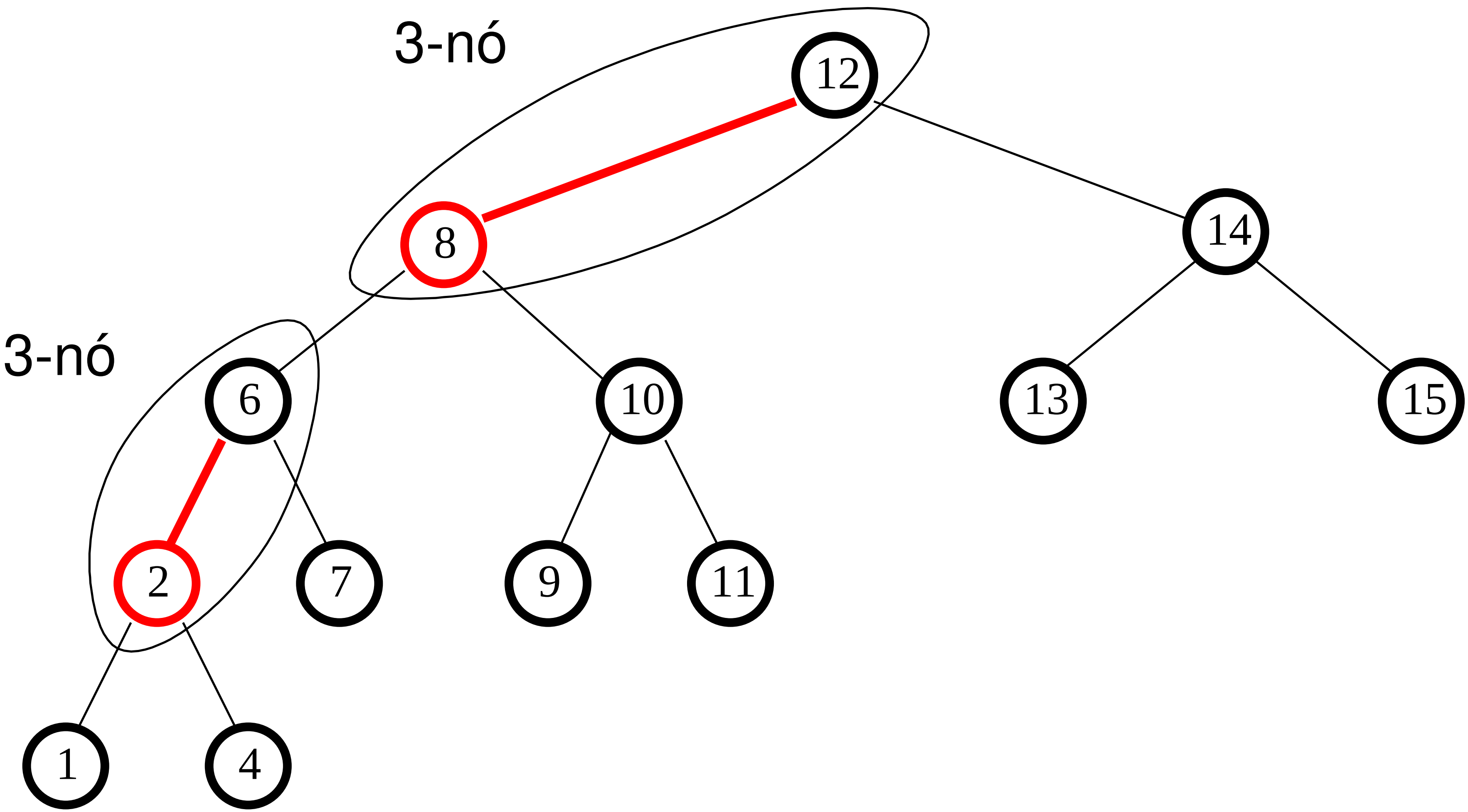
Árvores 2-3 $\times$ rubro-negras

Um nó de uma árvore 2-3 com duas chaves é chamado de 3-nó, e um com uma, é chamado de 2-nó.

Uma árvore 2-3 pode ser representada por uma árvore rubro-negra em que todo nó negro tem no máximo um filho rubro.

A representação é caída para a esquerda (*left leaning*) se, quando um nó negro tem um filho rubro, este é sempre o esquerdo.

Árvores 2-3 × rubro-negras



Inserção em ABB rubro-negra 2-3

Esta é a inserção para árvores 2-3:

INSIRAREC (T, x)

se $T = \text{NIL}$
  então $q \leftarrow$ NOVACÉLULA$(x, \text{NIL}, \text{NIL},$ RUBRO$)$
    devolva q
se $x < \mathit{info}(T)$
  então $\mathit{esq}(T) \leftarrow$ INSIRAREC$(\mathit{esq}(T), x)$
  senão $\mathit{dir}(T) \leftarrow$ INSIRAREC$(\mathit{dir}(T), x)$
se RUBRO$(\mathit{dir}(T))$ **e** NEGRO$(\mathit{esq}(T))$
  então $T \leftarrow$ ROTACIONEESQ(T)
se RUBRO$(\mathit{esq}(T))$ **e** RUBRO$(\mathit{esq}(\mathit{esq}(T)))$
  então $T \leftarrow$ ROTACIONEDIR(T)
se RUBRO$(\mathit{esq}(T))$ e RUBRO$(\mathit{dir}(T))$
  então TROQUECORES(T)
devolva T

Inserção em ABB rubro-negra 2-3

Mostre por indução na altura de T que,
no InsiraRec anterior,
se a árvore T for rubro-negra 2-3 exceto pela raiz,
que pode ser rubra, então a árvore T devolvida satisfaz:

- se a raiz de T era negra, então T é rubro-negra 2-3 exceto pela raiz, que pode ser rubra.
- se a raiz de T era rubra, então T é rubro-negra 2-3 exceto pela raiz, que pode ser rubra, e pode ter o filho esquerdo rubro.

(Fizemos isso na aula. Refaça para ver se você entendeu e se é capaz de reproduzir.)

Conclua que, se a T for uma árvore rubro-negra 2-3, então a árvore devolvida por Insira(T, x) é rubro-negra 2-3.

Árvores 2-3-4

Uma árvore 2-3-4 é uma extensão de uma árvore 2-3 em que os nós podem ter uma, duas ou três chaves e, se internos, respectivamente dois, três ou quatro filhos.

Uma árvore rubro-negra nada mais é do que uma implementação por uma ABB de uma árvore 2-3-4.

Um nó negro com dois filhos rubros é um 4-nó.

Mantenha isso em mente quando estudar os algoritmos sobre ABBs rubro-negras, pois isso ajuda. Por exemplo, note a seguir como a inserção em ABB rubro-negra difere da inserção em árvore 2-3 apenas pela mudança de duas linhas de código do lugar.

Inserção em ABB rubro-negra

Esta é a inserção para árvores 2-3-4:

InsiraRec (T, x)
se $T = \text{NIL}$
  então $q \leftarrow$ NovaCélula$(x, \text{NIL}, \text{NIL},$ RUBRO$)$
      devolva q
se Rubro$(esq(T))$ e Rubro$(dir(T))$
  então TroqueCores(T)
se $x < info(T)$
  então $esq(T) \leftarrow$ InsiraRec$(esq(T), x)$
  senão $dir(T) \leftarrow$ InsiraRec$(dir(T), x)$
se Rubro$(dir(T))$ **e** Negro$(esq(T))$
  então $T \leftarrow$ RotacioneEsq(T)
se Rubro$(esq(T))$ **e** Rubro$(esq(esq(T)))$
  então $T \leftarrow$ RotacioneDir(T)
devolva T

Inserção em ABB rubro-negra

Faça uma análise da correção da segunda versão do INSIRAREC, como fizemos para a primeira versão.

Conclua da sua análise que, se a T for uma árvore rubro-negra 2-3-4, então a árvore devolvida por INSIRA(T, x) é rubro-negra 2-3-4.

(Fizemos isso na aula. Refaça.)

Dica: ajuste as duas condições da prova por indução anterior para árvores rubro-negras 2-3-4, fortalecendo a primeira. Refaça a análise de casos cuidadosamente, para ver se os seus ajustes estão corretos. (Em outras palavras, faça a prova por indução.)

Remoção em ABBs

REMOVA (T, q)

se $esq(q) = \text{NIL}$ **ou** $dir(q) = \text{NIL}$
    então $p \leftarrow q$
    senão $p \leftarrow$ MÍNIMO$(dir(q))$ $info(q) \leftarrow info(p)$
se $esq(p) = \text{NIL}$
    então $f \leftarrow dir(p)$
    senão $f \leftarrow esq(p)$
se $f \neq \text{NIL}$
    então $pai(f) \leftarrow pai(p)$
se $pai(p) = \text{NIL}$
    então $T \leftarrow f$
    senão se $p = esq(pai(p))$
            então $esq(pai(p)) \leftarrow f$
            senão $dir(pai(p)) \leftarrow f$
LIBERACÉLULA(p)

Versão recursiva da remoção

Supõe que x está presente na ABB.

REMOVA (T, x)

$T \leftarrow$ REMOVAREC(T, x)

REMOVAREC (T, x)

se $x < info(T)$
  então $esq(T) \leftarrow$ REMOVAREC$(esq(T), x)$
  senão se $x > info(T)$
    então $dir(T) \leftarrow$ REMOVAREC$(dir(T), x)$
    senão se $dir(T) = \text{NIL}$
      então devolva $esq(T)$
      $p \leftarrow$ MÍNIMO$(dir(T))$
      $info(T) \leftarrow info(p)$
      $dir(T) \leftarrow$ REMOVAMIN$(dir(T))$
devolva T

Versão recursiva da remoção

REMOVAMIN (T)

1 **se** $esq(T) = \text{NIL}$
2 **então devolva** $dir(T)$
3 **senão** $esq(T) \leftarrow$ REMOVAMIN$(esq(T))$
4 **devolva** T

Versão recursiva da remoção

REMOVAMIN (T)

se $esq(T) = \text{NIL}$
  então devolva $dir(T)$
  senão $esq(T) \leftarrow$ REMOVAMIN$(esq(T))$
    devolva T

Exercício: Modifique as rotinas acima para que a remoção funcione mesmo que x não esteja na ABB dada.

Exercício: Ajuste estas rotinas para que mantenham o campo pai atualizado.

Exercício: A remoção iterativa do REMOVA pode resultar em algo diferente da versão recursiva apresentada. Modifique o REMOVAREC para que ela faça a remoção exatamente como a iterativa.

RemovaMin em árvores 2-3

REMOVAMIN (T)

se $esq(T) = \text{NIL}$
  então devolva NIL ▷ $dir(T) = \text{NIL}$ aqui
se NEGRO$(esq(T))$ **e** NEGRO$(esq(esq(T)))$
  então $T \leftarrow$ MOVARUBROESQ(T)
$esq(T) \leftarrow$ REMOVAMIN$(esq(T))$
▷ Fixup
se RUBRO$(dir(T))$ **e** NEGRO$(esq(T))$
  então $T \leftarrow$ ROTACIONEESQ(T)
se RUBRO$(esq(T))$ **e** RUBRO$(dir(T))$
  então TROQUECORES(T)
devolva T

RemovaMin em árvores 2-3

MOVARUBROESQ (p)

TROQUECORES(p)
se RUBRO$(esq(dir(p)))$
    então $dir(p) \leftarrow$ ROTACIONEDIR$(dir(p))$
        $p \leftarrow$ ROTACIONEESQ(p)
        TROQUECORES(p)
devolva p

RemovaMin em árvores 2-3

MOVARUBROESQ (p)

```
TROQUECORES(p)
se RUBRO(esq(dir(p)))
   então dir(p) ← ROTACIONEDIR(dir(p))
         p ← ROTACIONEESQ(p)
         TROQUECORES(p)
devolva p
```

Mostre que o REMOVAMIN funciona quando T é uma árvore rubro-negra 2-3 exceto pela raiz, que pode ser rubra, ou é uma árvore rubro-negra em que o filho esquerdo da raiz rubro.

(Fizemos isso na aula. Refaça.)