Árvores Rubro-Negras

Estrutura de Dados Avançada — QXD0015

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

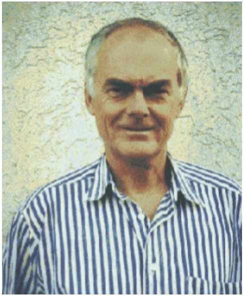
Prof. Atílio Gomes Luiz
gomes.atilio@ufc.br

Universidade Federal do Ceará

1º semestre/2024

Árvores rubro-negras (Red-Black trees)

- Originalmente criada por Rudolf Bayer em 1972.
  - Chamadas de Árvores Binárias Simétricas.
- Adquiriu o seu nome atual em um trabalho de Leonidas J. Guibas e Robert Sedgewick, em 1978.



R. Bayer

R. Sedgewick

Árvores rubro-negras na prática

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Árvores rubro-negras são utilizadas em diversas aplicações e bibliotecas de linguagens de programação.
- Exemplos:
  - **Java:** java.util.TreeMap, java.util.TreeSet
  - **C++ STL:** std::set, std::map, std::multiset
  - **Linux kernel:** completely fair scheduler, linux/rbtree.h

Árvores rubro-negras – Definição

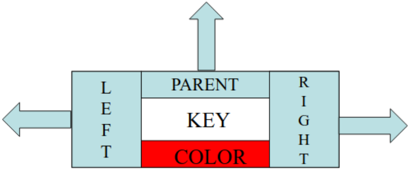
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Uma árvore rubro-negra é uma árvore binária de busca com propriedades adicionais.

Árvores rubro-negras – Definição

- Uma árvore rubro-negra é uma árvore binária de busca com propriedades adicionais.

- Cada nó de uma árvore rubro negra tem os seguintes campos:
  - color – Indica se o nó é **vermelho** ou **preto**.
  - key – Campo chave. Cada nó tem uma chave única.
  - right – Subárvore direita.
  - left – Subárvore esquerda.
  - p – Ponteiro para o pai do nó.



Árvores rubro-negras – Definição (cont.)

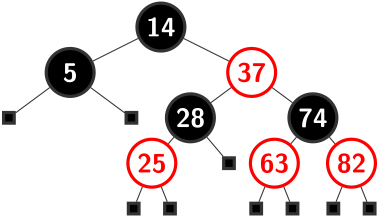
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Se o filho ou o pai de um nó não existir, o campo aponta para um nó especial chamado NIL que possui cor **preta**.

Árvores rubro-negras – Definição (cont.)

- Se o filho ou o pai de um nó não existir, o campo aponta para um nó especial chamado NIL que possui cor **preta**.

- Toda folha da árvore é um nó NIL e é também chamado nó externo. Os demais nós são chamados nós internos.



Árvores rubro-negras – Definição (cont.)

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Uma árvore rubro-negra também satisfaz as seguintes propriedades:

Árvores rubro-negras – Definição (cont.)

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Uma árvore rubro-negra também satisfaz as seguintes propriedades:

(P1) Todo nó da árvore ou é **vermelho** ou é **preto**.

Árvores rubro-negras – Definição (cont.)

Uma árvore rubro-negra também satisfaz as seguintes propriedades:

(P1) Todo nó da árvore ou é **vermelho** ou é **preto**.

(P2) A raiz é **preta**.

Árvores rubro-negras – Definição (cont.)

Uma árvore rubro-negra também satisfaz as seguintes propriedades:

(P1) Todo nó da árvore ou é **vermelho** ou é **preto**.

(P2) A raiz é **preta**.

(P3) Toda folha (NIL) é **preta**.

Árvores rubro-negras – Definição (cont.)

Uma árvore rubro-negra também satisfaz as seguintes propriedades:

(P1) Todo nó da árvore ou é **vermelho** ou é **preto**.

(P2) A raiz é **preta**.

(P3) Toda folha (NIL) é **preta**.

(P4) Se um nó é **vermelho**, então ambos os filhos são **pretos**.

Árvores rubro-negras – Definição (cont.)

Uma árvore rubro-negra também satisfaz as seguintes propriedades:

(P1) Todo nó da árvore ou é **vermelho** ou é **preto**.

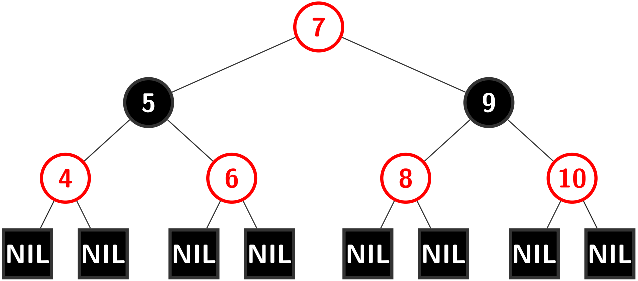
(P2) A raiz é **preta**.

(P3) Toda folha (NIL) é **preta**.

(P4) Se um nó é **vermelho**, então ambos os filhos são **pretos**.

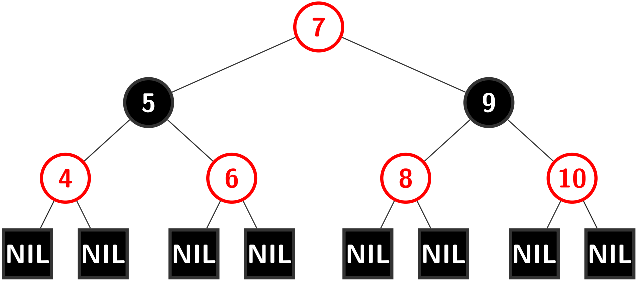
(P5) Para todo nó, todos os caminhos do nó até as folhas descendentes contêm o mesmo número de nós **pretos**.

Árvores rubro-negras – Reconhecendo 1



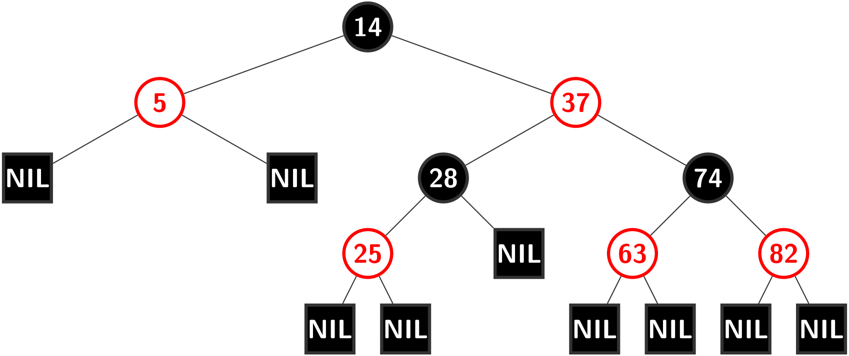
- É uma árvore rubro-negra?

Árvores rubro-negras – Reconhecendo 1



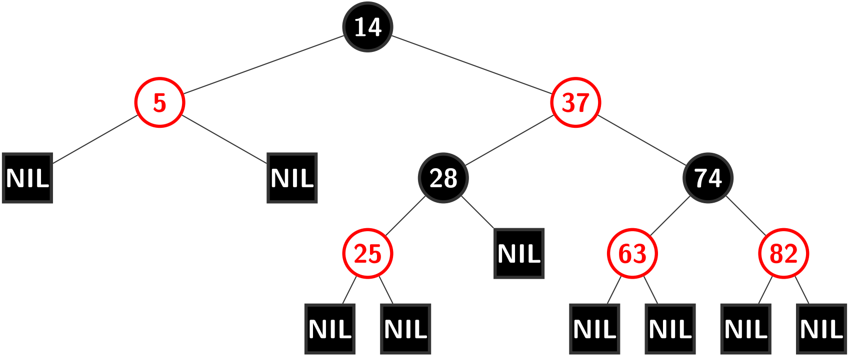
- É uma árvore rubro-negra?
- Não. Viola a Propriedade (2).

Árvores rubro-negras – Reconhecendo 2



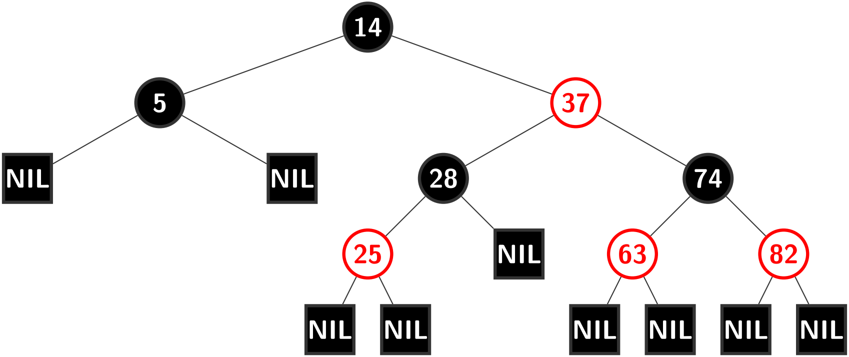
- É uma árvore rubro-negra?

Árvores rubro-negras – Reconhecendo 2



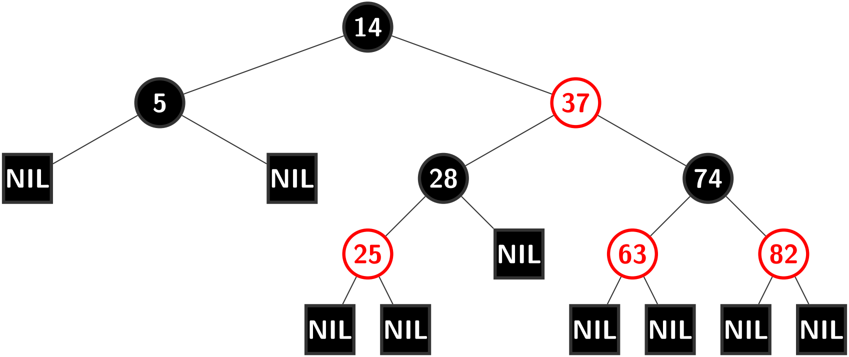
- É uma árvore rubro-negra?
- Não. Viola a Propriedade (5).

Árvores rubro-negras – Reconhecendo 3



- É uma árvore rubro-negra?

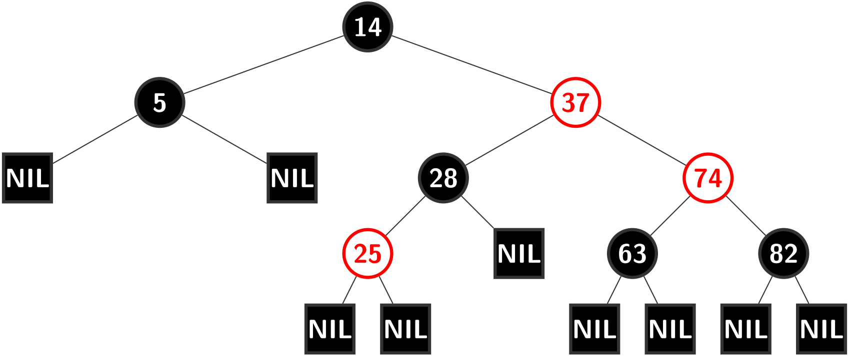
Árvores rubro-negras – Reconhecendo 3



- É uma árvore rubro-negra?
- SIM!!!

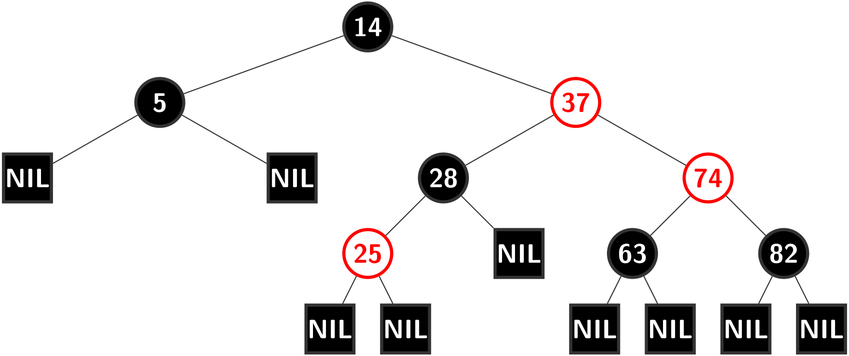
Árvores rubro-negras – Reconhecendo 4

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



- É uma árvore rubro-negra?

Árvores rubro-negras – Reconhecendo 4



- É uma árvore rubro-negra?
- NÃO. Viola a Propriedade (4).

Qual a ideia por trás da definição?

Qual a intuição por trás das propriedades da árvore rubro-negra?
Como elas levam ao balanceamento?

Qual a ideia por trás da definição?

Qual a intuição por trás das propriedades da árvore rubro-negra?
Como elas levam ao balanceamento?

- Restringindo a maneira com que os nós podem ser coloridos do caminho da raiz até qualquer uma das suas folhas, as árvores rubro-negras:

Qual a ideia por trás da definição?

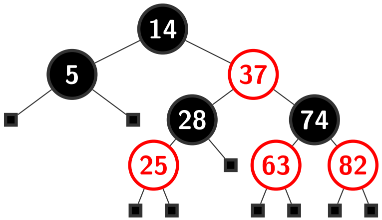
Qual a intuição por trás das propriedades da árvore rubro-negra?
Como elas levam ao balanceamento?

- Restringindo a maneira com que os nós podem ser coloridos do caminho da raiz até qualquer uma das suas folhas, as árvores rubro-negras:
    - Garantem que nenhum dos caminhos será maior que 2 vezes o comprimento de qualquer outro.

Qual a ideia por trás da definição?

Qual a intuição por trás das propriedades da árvore rubro-negra?
Como elas levam ao balanceamento?

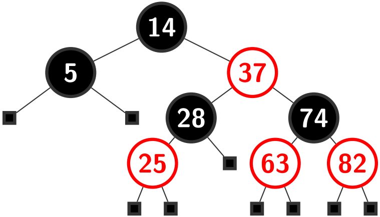
- Restringindo a maneira com que os nós podem ser coloridos do caminho da raiz até qualquer uma das suas folhas, as árvores rubro-negras:
  - Garantem que nenhum dos caminhos será maior que 2 vezes o comprimento de qualquer outro.
  - Isso garante que a árvore é balanceada ($h = O(\lg n)$).



Prova do balanceamento

Definição — Altura Negra

A altura negra de um nó v é o número de nós pretos visitados em qualquer caminho de v até suas folhas (sem incluir o próprio v).

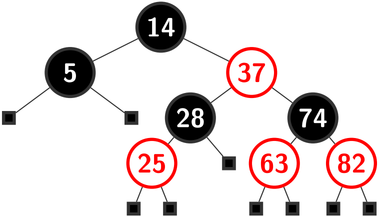


- A altura negra do nó v é denotada por $bh(v)$.

Definição — Altura Negra

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

A altura negra de um nó v é o número de nós pretos visitados em qualquer caminho de v até suas folhas (sem incluir o próprio v).



- A altura negra do nó v é denotada por $bh(v)$.
- Pela Propriedade (5), $bh(v)$ é bem-definida para todo nó v da árvore. A altura negra da árvore rubro-negra é definida como sendo $bh(root)$.

Teste rápido

Seja T uma árvore rubro-negra e sejam v e w nós de T tais que v é pai de w.

1. Se v é preto e w é vermelho, então qual das opções abaixo é verdadeira?
   (a) $bh(v) = bh(w) - 1$
   (b) $bh(v) = bh(w)$
   (c) $bh(v) = bh(w) + 1$

Teste rápido

Seja T uma árvore rubro-negra e sejam v e w nós de T tais que v é pai de w.

1. Se v é preto e w é vermelho, então qual das opções abaixo é verdadeira?
   (a) $bh(v) = bh(w) - 1$
   (b) $bh(v) = bh(w)$
   (c) $bh(v) = bh(w) + 1$
2. Se v é vermelho e w é preto, então qual das opções abaixo é verdadeira?
   (a) $bh(v) = bh(w) - 1$
   (b) $bh(v) = bh(w)$
   (c) $bh(v) = bh(w) + 1$

Teste rápido

Seja T uma árvore rubro-negra e sejam v e w nós de T tais que v é pai de w.

1. Se v é preto e w é vermelho, então qual das opções abaixo é verdadeira?
   (a) $bh(v) = bh(w) - 1$
   (b) $bh(v) = bh(w)$
   (c) $bh(v) = bh(w) + 1$
2. Se v é vermelho e w é preto, então qual das opções abaixo é verdadeira?
   (a) $bh(v) = bh(w) - 1$
   (b) $bh(v) = bh(w)$
   (c) $bh(v) = bh(w) + 1$
3. Se ambos v e w são pretos, então qual das opções abaixo é verdadeira?
   (a) $bh(v) = bh(w) - 1$
   (b) $bh(v) = bh(w)$
   (c) $bh(v) = bh(w) + 1$

Teste rápido

Seja T uma árvore rubro-negra e sejam v e w nós de T tais que v é pai de w.

1. Se v é preto e w é vermelho, então qual das opções abaixo é verdadeira?
   (a) $bh(v) = bh(w) - 1$
   (b) $bh(v) = bh(w)$
   (c) $bh(v) = bh(w) + 1$
2. Se v é vermelho e w é preto, então qual das opções abaixo é verdadeira?
   (a) $bh(v) = bh(w) - 1$
   (b) $bh(v) = bh(w)$
   (c) $bh(v) = bh(w) + 1$
3. Se ambos v e w são pretos, então qual das opções abaixo é verdadeira?
   (a) $bh(v) = bh(w) - 1$
   (b) $bh(v) = bh(w)$
   (c) $bh(v) = bh(w) + 1$

Respostas: 1(b), 2(c), 3(c)

Lemas sobre árvores rubro-negras

Lema 1

Seja T uma árvore rubro-negra e seja v um nó de T com altura h. Então,

$$bh(v) \geq \frac{h(v) - 1}{2}.$$

Atenção: nas minhas aulas, eu uso a seguinte definição de altura:

A altura de um nó v é o número de nós no caminho de v até uma de suas folhas descendentes no nível de maior profundidade.

Lemas sobre árvores rubro-negras

Demonstração do Lema 1:

Lemas sobre árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Demonstração do Lema 1:

Seja v um nó de T com altura $h(v)$.

Pela **Propriedade 5**, todos os caminhos de v até suas folhas descendentes contém o mesmo número de nós pretos.

Lemas sobre árvores rubro-negras

Demonstração do Lema 1:

Seja v um nó de T com altura $h(v)$.

Pela **Propriedade 5**, todos os caminhos de v até suas folhas descendentes contém o mesmo número de nós pretos.

Note que o maior caminho possível saindo de v até uma folha descendente deve intercalar nós vermelhos e nós pretos (**Propriedade 4**).

Lemas sobre árvores rubro-negras

Universidade Federal do Ceará
Campus Quixadá

Demonstração do Lema 1:

Seja v um nó de T com altura $h(v)$.

Pela **Propriedade 5**, todos os caminhos de v até suas folhas descendentes contém o mesmo número de nós pretos.

Note que o maior caminho possível saindo de v até uma folha descendente deve intercalar nós vermelhos e nós pretos (**Propriedade 4**).

Caso 1: v é vermelho. Neste caso, no maior caminho possível de v até uma de suas folhas descendentes, todo nó preto tem um pai vermelho. Assim, $h(v) \leq 2 \cdot bh(v)$.

Lemas sobre árvores rubro-negras

Universidade Federal do Ceará
Campus Quixadá

Demonstração do Lema 1:

Seja v um nó de T com altura $h(v)$.

Pela **Propriedade 5**, todos os caminhos de v até suas folhas descendentes contém o mesmo número de nós pretos.

Note que o maior caminho possível saindo de v até uma folha descendente deve intercalar nós vermelhos e nós pretos (**Propriedade 4**).

Caso 1: v é vermelho. Neste caso, no maior caminho possível de v até uma de suas folhas descendentes, todo nó preto tem um pai vermelho. Assim, $h(v) \leq 2 \cdot bh(v)$.

Caso 2: v é preto. Neste caso, no maior caminho possível de v até uma de suas folhas descendentes, todo nó vermelho tem um filho preto e o filho de v nesse caminho, chame-o w, é vermelho. Assim, $h(v) \leq h(w) + 1 \leq 2 \cdot bh(w) + 1 = 2 \cdot bh(v) + 1$.

Lemas sobre árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Demonstração do Lema 1:

Seja v um nó de T com altura $h(v)$.

Pela **Propriedade 5**, todos os caminhos de v até suas folhas descendentes contém o mesmo número de nós pretos.

Note que o maior caminho possível saindo de v até uma folha descendente deve intercalar nós vermelhos e nós pretos (**Propriedade 4**).

Caso 1: v é vermelho. Neste caso, no maior caminho possível de v até uma de suas folhas descendentes, todo nó preto tem um pai vermelho. Assim, $h(v) \leq 2 \cdot bh(v)$.

Caso 2: v é preto. Neste caso, no maior caminho possível de v até uma de suas folhas descendentes, todo nó vermelho tem um filho preto e o filho de v nesse caminho, chame-o w, é vermelho. Assim,
$h(v) \leq h(w) + 1 \leq 2 \cdot bh(w) + 1 = 2 \cdot bh(v) + 1$.

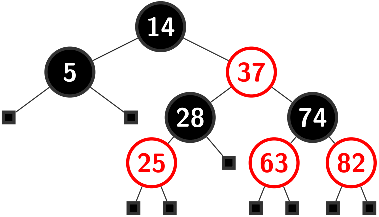
Dos Casos 1 e 2, concluímos que $h(v) \leq 2 \cdot bh(v) + 1$, o que implica que $bh(v) \geq \frac{h(v)-1}{2}$. $\square$

Lemas sobre árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Lema 2

A subárvore enraizada em um nó x qualquer contém pelo menos $2^{bh(x)} - 1$ nós internos.



Lemas sobre árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Lema 2

A subárvore enraizada em um nó x qualquer contém pelo menos $2^{bh(x)} - 1$ nós internos.

Lemas sobre árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Lema 2

A subárvore enraizada em um nó x qualquer contém pelo menos $2^{bh(x)} - 1$ nós internos.

Demonstração:

Lemas sobre árvores rubro-negras

Universidade Federal do Ceará
Campus Quixadá

Lema 2

A subárvore enraizada em um nó x qualquer contém pelo menos $2^{bh(x)} - 1$ nós internos.

Demonstração:

- Por indução na altura h do vértice x.

Lemas sobre árvores rubro-negras

Lema 2

A subárvore enraizada em um nó x qualquer contém pelo menos $2^{bh(x)} - 1$ nós internos.

Demonstração:

- Por indução na altura h do vértice x.
- **Caso base:** Se a altura h de x é 1, então x é um nó externo (NIL). Então, a árvore enraizada em x contém pelo menos $2^{bh(x)} - 1 = 2^0 - 1 = 0$ nós internos.

Conclusão da demonstração

- **Passo indutivo:** Considere um nó x com altura $h > 1$. Note que x é um nó interno e possui dois filhos.

Conclusão da demonstração

- **Passo indutivo:** Considere um nó x com altura $h > 1$. Note que x é um nó interno e possui dois filhos.

- Cada filho de x tem altura negra igual a $bh(x)$ ou $bh(x) - 1$, dependendo se sua cor é **vermelha** ou **preta**, respectivamente.

Conclusão da demonstração

- **Passo indutivo:** Considere um nó x com altura $h > 1$. Note que x é um nó interno e possui dois filhos.

- Cada filho de x tem altura negra igual a $bh(x)$ ou $bh(x) - 1$, dependendo se sua cor é **vermelha** ou **preta**, respectivamente.

- Como a altura de um filho de x é menor que a altura de x, pela **hipótese de indução**, cada filho de x tem pelo menos $2^{bh(x)-1} - 1$ nós internos.

Conclusão da demonstração

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- **Passo indutivo:** Considere um nó x com altura $h > 1$. Note que x é um nó interno e possui dois filhos.
- Cada filho de x tem altura negra igual a $bh(x)$ ou $bh(x) - 1$, dependendo se sua cor é **vermelha** ou **preta**, respectivamente.
- Como a altura de um filho de x é menor que a altura de x, pela **hipótese de indução**, cada filho de x tem pelo menos $2^{bh(x)-1} - 1$ nós internos.
- Seja n o número de nós internos da subárvore enraizada em x. Então,

$$n \geq (2^{bh(x)-1} - 1) + (2^{bh(x)-1} - 1) + 1 = 2^{bh(x)} - 1.$$

e o resultado segue. ■

Árvores rubro-negras são balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Teorema 3

A altura de uma árvore rubro-negra com n nós internos é no máximo $2\lg(n+1)+1$.

Árvores rubro-negras são balanceadas

Universidade Federal do Ceará
Campus Quixadá

Teorema 3

A altura de uma árvore rubro-negra com n nós internos é no máximo $2\lg(n+1)+1$.

Demonstração:

Árvores rubro-negras são balanceadas

Universidade Federal do Ceará
Campus Quixadá

Teorema 3

A altura de uma árvore rubro-negra com n nós internos é no máximo $2\lg(n+1)+1$.

Demonstração:

- Seja T uma árvore rubro-negra com altura h, com n nós, e seja v sua raiz.

Árvores rubro-negras são balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Teorema 3

A altura de uma árvore rubro-negra com n nós internos é no máximo $2\lg(n+1)+1$.

Demonstração:

- Seja T uma árvore rubro-negra com altura h, com n nós, e seja v sua raiz.
- Pelo **Lema 1**, $bh(v) \geq \frac{h(v)-1}{2} = \frac{h-1}{2}$.

Árvores rubro-negras são balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Teorema 3

A altura de uma árvore rubro-negra com n nós internos é no máximo $2\lg(n+1)+1$.

Demonstração:

- Seja T uma árvore rubro-negra com altura h, com n nós, e seja v sua raiz.
- Pelo **Lema 1**, $bh(v) \geq \frac{h(v)-1}{2} = \frac{h-1}{2}$.
- Logo, pelo **Lema 2**, temos que $n \geq 2^{(h-1)/2} - 1$.

Árvores rubro-negras são balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Teorema 3

A altura de uma árvore rubro-negra com n nós internos é no máximo $2\lg(n+1)+1$.

Demonstração:

- Seja T uma árvore rubro-negra com altura h, com n nós, e seja v sua raiz.
- Pelo **Lema 1**, $bh(v) \geq \frac{h(v)-1}{2} = \frac{h-1}{2}$.
- Logo, pelo **Lema 2**, temos que $n \geq 2^{(h-1)/2} - 1$.
- $n \geq 2^{(h-1)/2} - 1 \Longrightarrow$

Árvores rubro-negras são balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Teorema 3

A altura de uma árvore rubro-negra com n nós internos é no máximo $2\lg(n+1)+1$.

Demonstração:

- Seja T uma árvore rubro-negra com altura h, com n nós, e seja v sua raiz.
- Pelo **Lema 1**, $bh(v) \geq \frac{h(v)-1}{2} = \frac{h-1}{2}$.
- Logo, pelo **Lema 2**, temos que $n \geq 2^{(h-1)/2} - 1$.
- $n \geq 2^{(h-1)/2} - 1 \Longrightarrow n+1 \geq 2^{(h-1)/2} \Longrightarrow$

Árvores rubro-negras são balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Teorema 3

A altura de uma árvore rubro-negra com n nós internos é no máximo $2\lg(n+1)+1$.

Demonstração:

- Seja T uma árvore rubro-negra com altura h, com n nós, e seja v sua raiz.
- Pelo **Lema 1**, $bh(v) \geq \frac{h(v)-1}{2} = \frac{h-1}{2}$.
- Logo, pelo **Lema 2**, temos que $n \geq 2^{(h-1)/2} - 1$.
- $n \geq 2^{(h-1)/2} - 1 \implies n+1 \geq 2^{(h-1)/2} \implies \lg(n+1) \geq \lg 2^{(h-1)/2}$
  $\implies$

Árvores rubro-negras são balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Teorema 3

A altura de uma árvore rubro-negra com n nós internos é no máximo $2\lg(n+1)+1$.

Demonstração:

- Seja T uma árvore rubro-negra com altura h, com n nós, e seja v sua raiz.
- Pelo **Lema 1**, $bh(v) \geq \frac{h(v)-1}{2} = \frac{h-1}{2}$.
- Logo, pelo **Lema 2**, temos que $n \geq 2^{(h-1)/2} - 1$.
- $n \geq 2^{(h-1)/2} - 1 \Longrightarrow n+1 \geq 2^{(h-1)/2} \Longrightarrow \lg(n+1) \geq \lg 2^{(h-1)/2}$ $\Longrightarrow \lg(n+1) \geq \frac{h-1}{2} \Longrightarrow$

Árvores rubro-negras são balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Teorema 3

A altura de uma árvore rubro-negra com n nós internos é no máximo $2\lg(n+1)+1$.

Demonstração:

- Seja T uma árvore rubro-negra com altura h, com n nós, e seja v sua raiz.
- Pelo **Lema 1**, $bh(v) \geq \frac{h(v)-1}{2} = \frac{h-1}{2}$.
- Logo, pelo **Lema 2**, temos que $n \geq 2^{(h-1)/2} - 1$.
- $n \geq 2^{(h-1)/2} - 1 \implies n+1 \geq 2^{(h-1)/2} \implies \lg(n+1) \geq \lg 2^{(h-1)/2}$
  $\implies \lg(n+1) \geq \frac{h-1}{2} \implies h-1 \leq 2\lg(n+1) \implies$

Árvores rubro-negras são balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Teorema 3

A altura de uma árvore rubro-negra com n nós internos é no máximo $2\lg(n+1)+1$.

Demonstração:

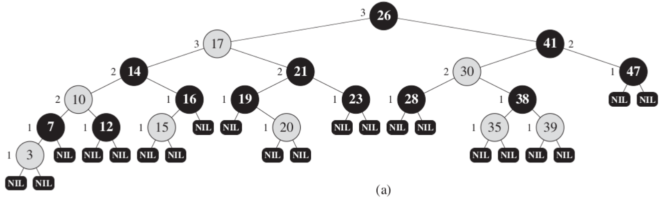
- Seja T uma árvore rubro-negra com altura h, com n nós, e seja v sua raiz.
- Pelo **Lema 1**, $bh(v) \geq \frac{h(v)-1}{2} = \frac{h-1}{2}$.
- Logo, pelo **Lema 2**, temos que $n \geq 2^{(h-1)/2} - 1$.
- $n \geq 2^{(h-1)/2} - 1 \Longrightarrow n+1 \geq 2^{(h-1)/2} \Longrightarrow \lg(n+1) \geq \lg 2^{(h-1)/2}$ $\Longrightarrow \lg(n+1) \geq \frac{h-1}{2} \Longrightarrow h-1 \leq 2\lg(n+1) \Longrightarrow h \leq 2\lg(n+1)+1$. ■

Corolário

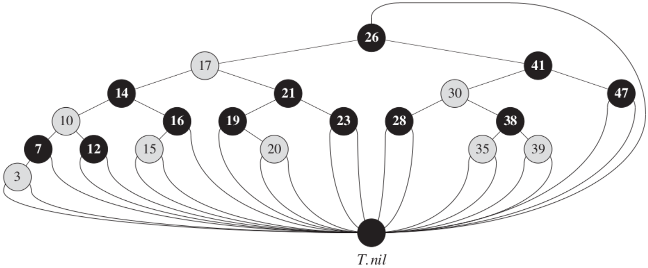
Corolário 4

As operações de Busca, Mínimo, Máximo, Sucessor e Predecessor podem ser efetuadas em tempo $O(\lg(n))$ em uma árvore rubro-negra. $\square$

O nó T.NIL



(a)



Vamos considerar que cada `NIL` é substituído por um único nó chamado `T.NIL`, que sempre tem cor preta. O nó `T.NIL` também é o pai de `T.root`

Rotações

Rotações

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Antes de vermos como fazer inserções em uma árvore rubro-negra, é preciso apresentar o conceito de rotações.

Rotações

- Antes de vermos como fazer inserções em uma árvore rubro-negra, é preciso apresentar o conceito de rotações.
- Usamos as rotações para consertar parte do estrago feito pelas operações já conhecidas de inserção e remoção nas propriedades rubro-negras.

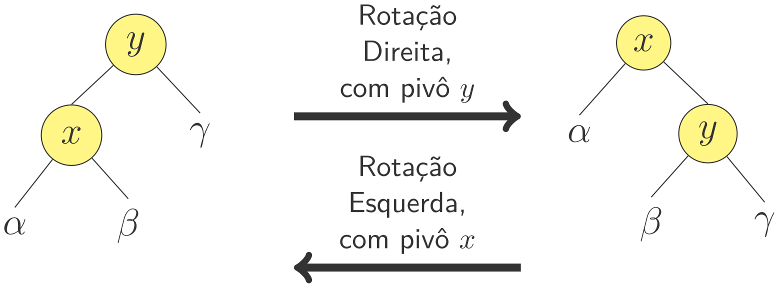
Rotações

- Antes de vermos como fazer inserções em uma árvore rubro-negra, é preciso apresentar o conceito de rotações.
- Usamos as rotações para consertar parte do estrago feito pelas operações já conhecidas de inserção e remoção nas propriedades rubro-negras.
- O resto do estrago é consertado utilizando recoloração de nós.

Rotações

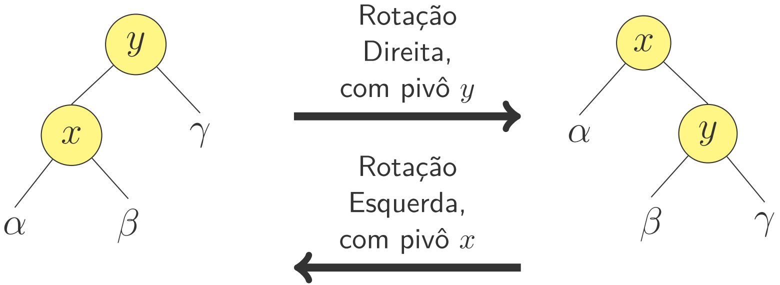
- Antes de vermos como fazer inserções em uma árvore rubro-negra, é preciso apresentar o conceito de rotações.
- Usamos as rotações para consertar parte do estrago feito pelas operações já conhecidas de inserção e remoção nas propriedades rubro-negras.
- O resto do estrago é consertado utilizando recoloração de nós.
- Rotações são operações locais: alteram um número pequeno e constante de ponteiros.

Rotações Esquerda e Direita



- As letras α, β, γ representam subárvores quaisquer, que podem ou não ser vazias.

Rotações Esquerda e Direita



- As letras α, β, γ representam subárvores quaisquer, que podem ou não ser vazias.
- Note que as duas rotações preservam a propriedade da árvore binária de busca.

Rotação à esquerda — Pseudocódigo

Rotação à esquerda — Pseudocódigo

$\textsc{Left-Rotate}(T, x)$

```
y = x.right
x.right = y.left
if y.left ≠ T.NIL
        y.left.p = x
y.p = x.p
if x.p == T.NIL
        T.root = y
elseif x == x.p.left
        x.p.left = y
else
        x.p.right = y
y.left = x
x.p = y
```

Essa função supõe que `x->right != T.NIL` e que `T.root.p == T.NIL`.

Rotação à esquerda — Pseudocódigo

LEFT-ROTATE(T, x)

```
y = x.right
x.right = y.left
if y.left ≠ T.NIL
      y.left.p = x
y.p = x.p
if x.p == T.NIL
      T.root = y
elseif x == x.p.left
      x.p.left = y
else
      x.p.right = y
y.left = x
x.p = y
```

Essa função supõe que `x->right != T.NIL` e que `T.root.p == T.NIL`.

- O pseudocódigo para `Right-Rotate(T,y)` é simétrico e é deixado como exercício.

Inserção

Visão geral da inserção

- Um nó que satisfaz as regras de uma árvore rubro-negra é denominado regulado, caso contrário é dito desregulado.
- Em uma árvore rubro-negra todos os nós estão regulados.

Visão geral da inserção

Universidade Federal do Ceará
Campus Quixadá

- Um nó que satisfaz as regras de uma árvore rubro-negra é denominado regulado, caso contrário é dito desregulado.
- Em uma árvore rubro-negra todos os nós estão regulados.
- Uma consequência direta das propriedades é que em qualquer caminho da raiz até uma folha não existem dois nós vermelhos consecutivos.

Visão geral da inserção

- Um nó que satisfaz as regras de uma árvore rubro-negra é denominado regulado, caso contrário é dito desregulado.

- Em uma árvore rubro-negra todos os nós estão regulados.

- Uma consequência direta das propriedades é que em qualquer caminho da raiz até uma folha não existem dois nós vermelhos consecutivos.

- Cada vez que uma operação de inserção/remoção for realizada na árvore, o conjunto de propriedades é verificado em busca de violações.

Visão geral da inserção

- Um nó que satisfaz as regras de uma árvore rubro-negra é denominado regulado, caso contrário é dito desregulado.
- Em uma árvore rubro-negra todos os nós estão regulados.
- Uma consequência direta das propriedades é que em qualquer caminho da raiz até uma folha não existem dois nós vermelhos consecutivos.
- Cada vez que uma operação de inserção/remoção for realizada na árvore, o conjunto de propriedades é verificado em busca de violações.
- Caso alguma propriedade tenha sido violada, realizamos rotações e ajustamos as cores dos nós para que todas as propriedades continuem válidas.

Inserções em árvores rubro-negras

- Árvores rubro-negras são árvores de busca binária com propriedades adicionais:

(P1) Um nó é vermelho ou é **preto**.
(P2) A raiz é **preta**.
(P3) Toda folha (NIL) é **preta**.
(P4) Se um nó é vermelho então ambos os seus filhos são **pretos**.
(P5) Para cada nó p, todos os caminhos desde p até as folhas contêm o mesmo número de nós pretos.

Inserções em árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Árvores rubro-negras são árvores de busca binária com propriedades adicionais:

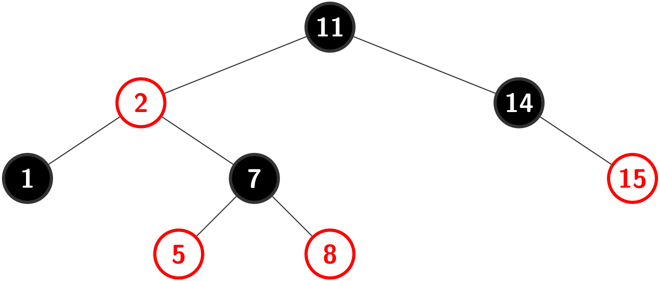
(P1) Um nó é vermelho ou é **preto**.
(P2) A raiz é **preta**.
(P3) Toda folha (NIL) é **preta**.
(P4) Se um nó é vermelho então ambos os seus filhos são **pretos**.
(P5) Para cada nó p, todos os caminhos desde p até as folhas contêm o mesmo número de nós pretos.

- Se utilizarmos o algoritmo que já conhecemos para a inserção em uma árvore rubro-negra, corremos o risco de quebrar algumas dessas regras. Mas quais?

Inserções em árvores rubro-negras

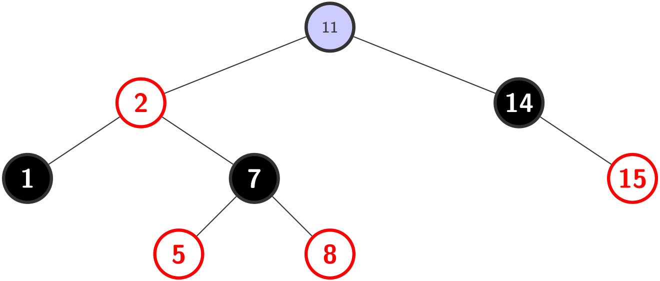
Revisando a inserção em árvore binária de busca

Exemplo: Inserindo a chave 4



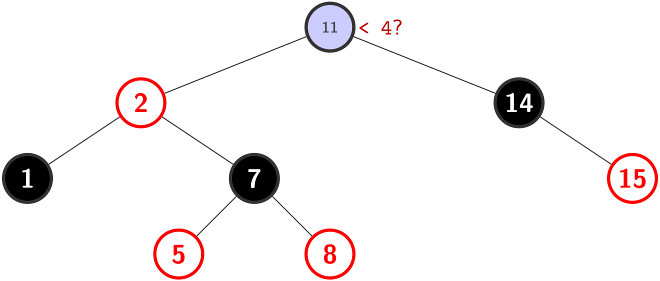
Inserções em árvores rubro-negras

Revisando a inserção em árvore binária de busca
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

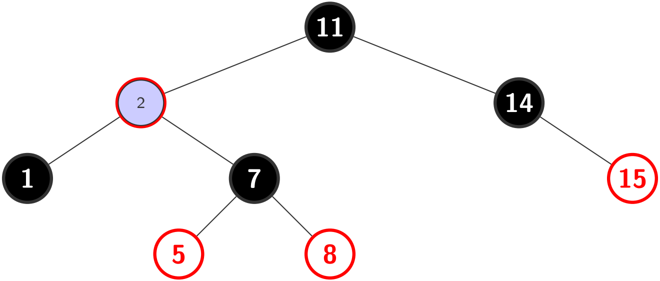
Revisando a inserção em árvore binária de busca
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

Revisando a inserção em árvore binária de busca

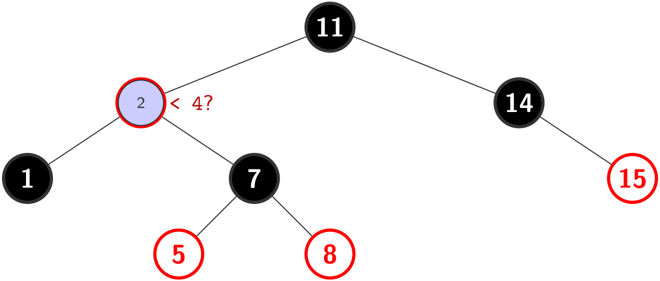
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

Revisando a inserção em árvore binária de busca

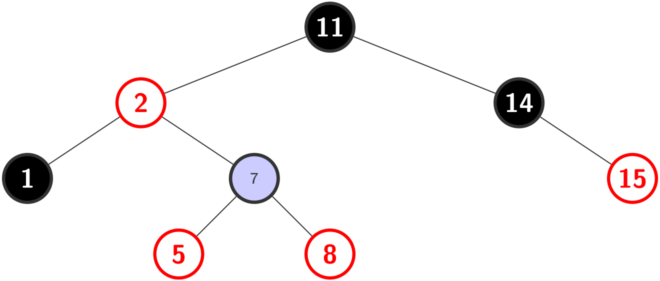
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

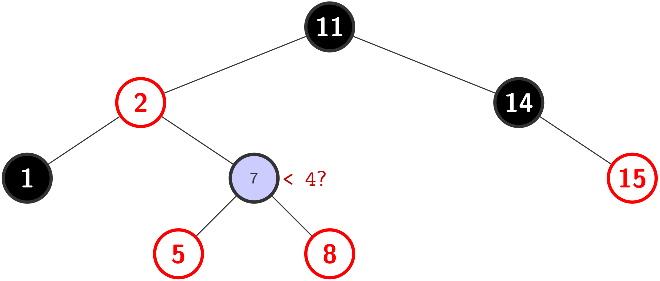
Revisando a inserção em árvore binária de busca

Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

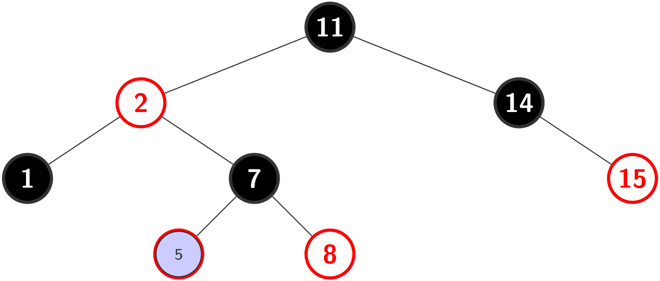
Revisando a inserção em árvore binária de busca
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

Revisando a inserção em árvore binária de busca

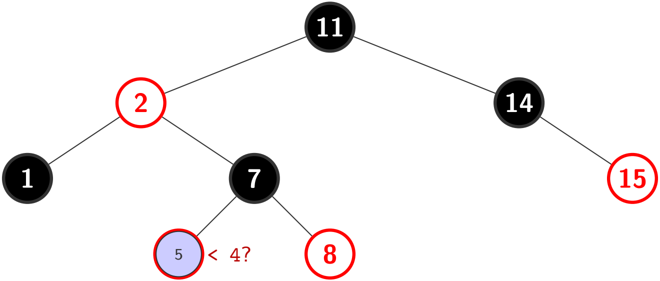
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

Revisando a inserção em árvore binária de busca

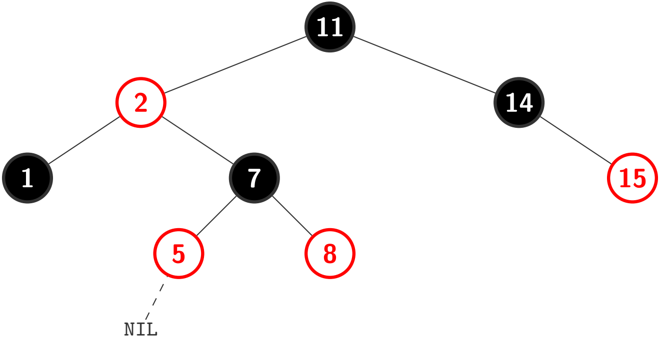
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

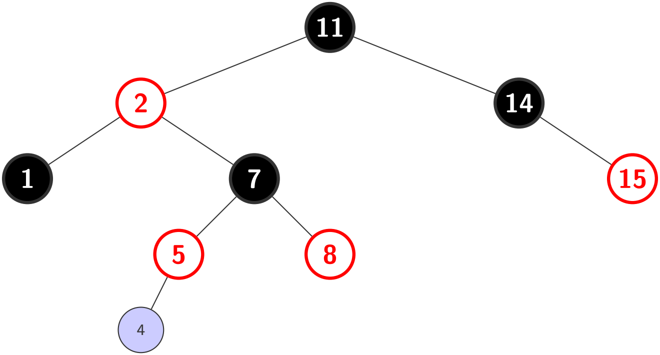
Revisando a inserção em árvore binária de busca

Exemplo: Inserindo a chave 4



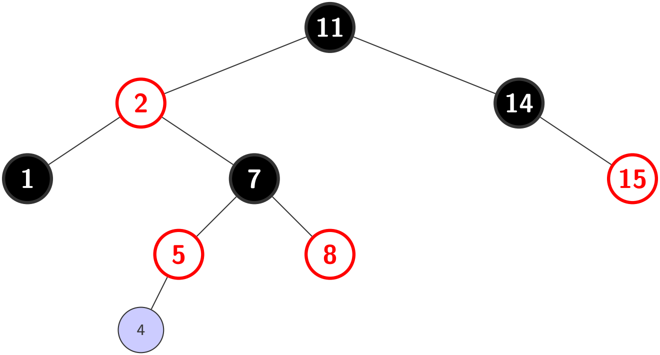
Inserções em árvores rubro-negras

Revisando a inserção em árvore binária de busca
Exemplo: Inserindo a chave 4



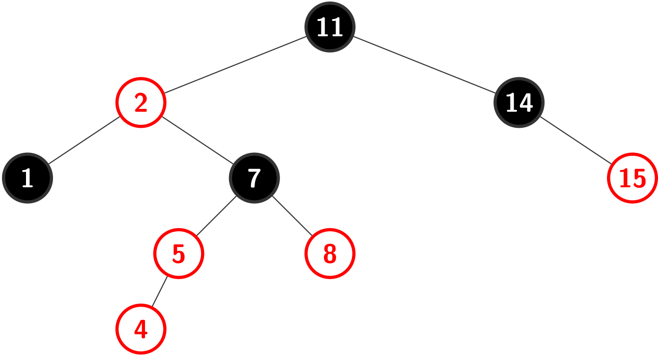
- Quebra P1 — Todo nó deve ser vermelho ou **preto**.

Inserções em árvores rubro-negras



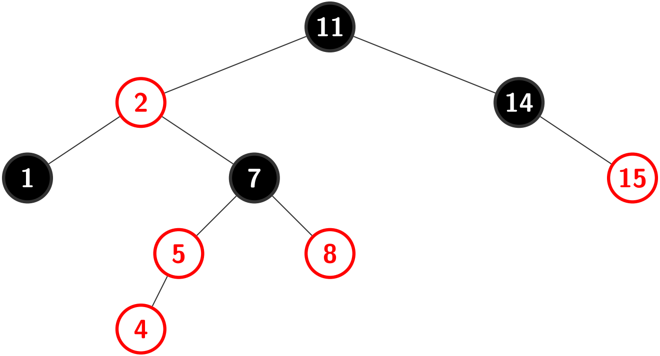
- É preciso decidir, qual desregula menos, colocar um nó vermelho ou um **preto**?

Inserções em árvores rubro-negras



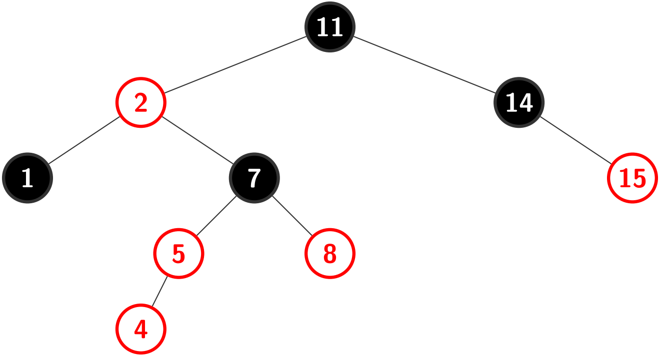
- É preciso decidir, qual desregula menos, colocar um nó vermelho ou um **preto**?
  - Em algumas situações, o vermelho pode não quebrar nada.
  - O preto sempre vai desequilibrar a altura negra dos seus ancestrais.

Inserções em árvores rubro-negras



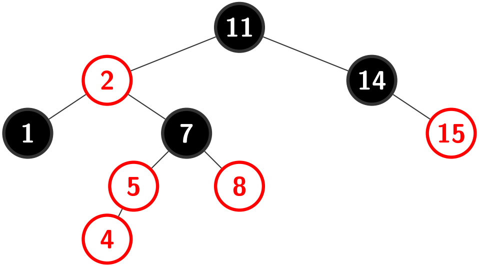
- Propriedade (1) satisfeita, sempre insiro um nó com a cor vermelha.

Inserções em árvores rubro-negras



- Propriedade (1) satisfeita, sempre insiro um nó com a cor vermelha.
- E agora, quais regras eu posso ter quebrado?

Avaliando quebras das propriedades



(P1) Um nó é vermelho ou é **preto**.

(P2) A raiz é **preta**.

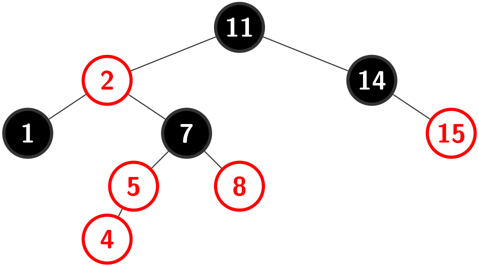
(P3) Toda folha (NIL) é **preta**.

(P4) Se um nó é vermelho então ambos os seus filhos são **pretos**.

(P5) Para cada nó p, todos os caminhos desde p até as folhas contêm o mesmo número de nós pretos.

Avaliando quebras das propriedades

Universidade Federal do Ceará
Campus Quixadá



(P1) ~~Um nó é vermelho ou é **preto**.~~

(P2) A raiz é **preta**.

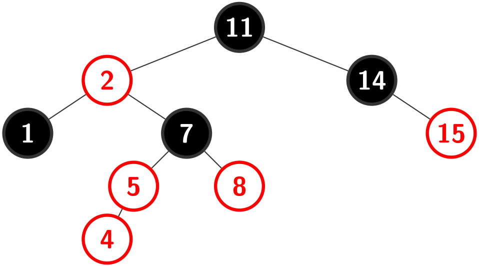
(P3) Toda folha (NIL) é **preta**.

(P4) Se um nó é vermelho então ambos os seus filhos são **pretos**.

(P5) Para cada nó p, todos os caminhos desde p até as folhas contêm o mesmo número de nós pretos.

Avaliando quebras das propriedades

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



(P1) ~~Um nó é vermelho ou é **preto**.~~

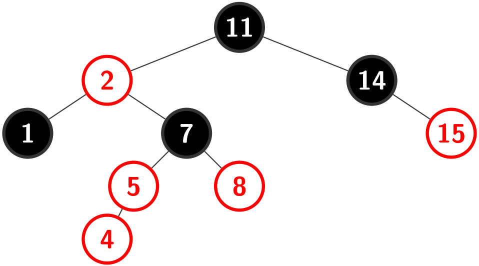
(P2) A raiz é **preta**.

(P3) ~~Toda folha (NIL) é **preta**.~~

(P4) Se um nó é vermelho então ambos os seus filhos são **pretos**.

(P5) Para cada nó p, todos os caminhos desde p até as folhas contêm o mesmo número de nós pretos.

Avaliando quebras das propriedades



(P1) ~~Um nó é vermelho ou é **preto**.~~

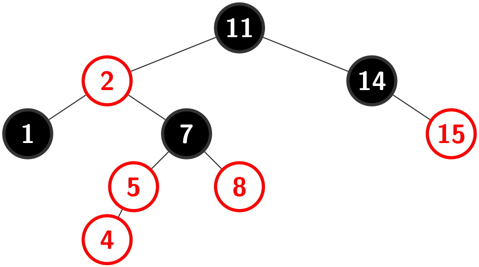
(P2) A raiz é **preta**.

(P3) ~~Toda folha (NIL) é **preta**.~~

(P4) Se um nó é vermelho então ambos os seus filhos são **pretos**.

(P5) ~~Para cada nó p, todos os caminhos desde p até as folhas contêm o mesmo número de nós pretos.~~

Avaliando quebras das propriedades



- Sabemos que podemos quebrar apenas as propriedades (P2) e (P4).
- Precisaremos de uma estratégia para recolorir os nós e rebalancear a árvore.
- Porém, já sabemos inserir o nó. Vamos ver o pseudocódigo da inserção!

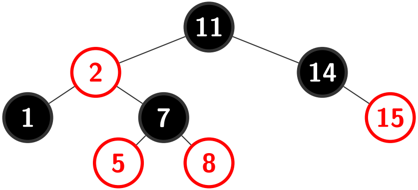
Inserindo um nó – Pseudocódigo

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

RB-INSERT(T, z)

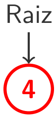
```
x = T.root
y = T.NIL
while x ≠ T.NIL
    y = x
    if z.key < x.key
        x = x.left
    elseif z.key > x.key
        x = x.right
    else
        return
z.p = y
if y == T.NIL
    T.root = z
elseif z.key < y.key
    y.left = z
else
    y.right = z
z.left = T.NIL
z.right = T.NIL
z.color = RED
RB-INSERT-FIXUP(T, z)
```

Obs.: Esta função recebe um nó z como argumento e insere z na árvore T. O nó z foi criado fora desta função e já contém a chave a ser inserida na árvore.



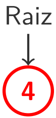
Resolvendo violação da Propriedade 2

Propriedade 2: A raiz é **preta**.

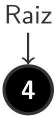


Resolvendo violação da Propriedade 2

Propriedade 2: A raiz é **preta**.



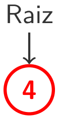
- Quando a raiz for vermelha, basta pintá-la de preto.



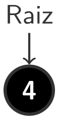
- Isto não quebra nenhuma outra propriedade. Por quê?

Resolvendo violação da Propriedade 2

Propriedade 2: A raiz é **preta**.



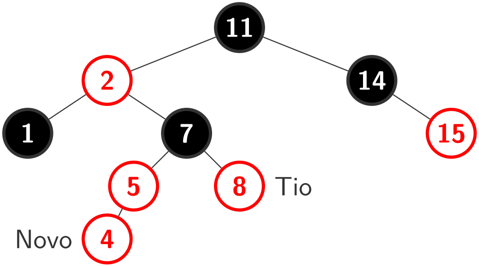
- Quando a raiz for vermelha, basta pintá-la de preto.



- Isto não quebra nenhuma outra propriedade. Por quê?
- Agora, vamos tentar garantir a Propriedade 4

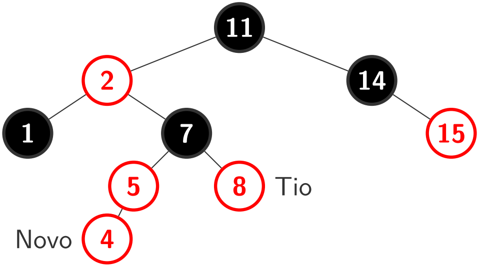
Resolvendo violação da Propriedade 4

O pai do nó Novo é Vermelho. Então, Novo tem um avô e ele é preto.



Resolvendo violação da Propriedade 4

O pai do nó Novo é Vermelho. Então, Novo tem um avô e ele é preto.

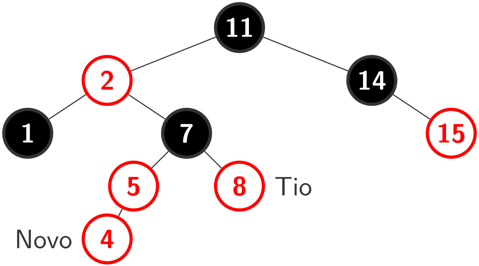


Dividiremos a análise em 3 casos:

- Caso 1: O tio do nó Novo é vermelho.

Resolvendo violação da Propriedade 4

O pai do nó Novo é Vermelho. Então, Novo tem um avô e ele é preto.

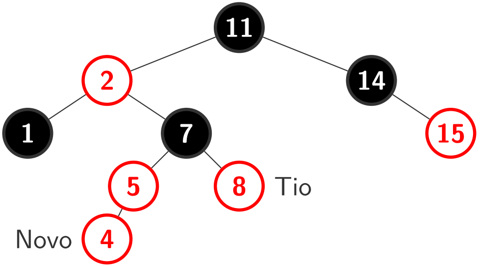


Dividiremos a análise em 3 casos:

- Caso 1: O tio do nó Novo é vermelho.
- Caso 2: O tio do nó Novo é **preto** e Novo é filho da direita.

Resolvendo violação da Propriedade 4

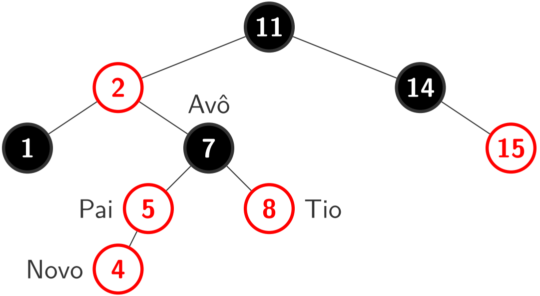
O pai do nó Novo é Vermelho. Então, Novo tem um avô e ele é preto.



Dividiremos a análise em 3 casos:

- Caso 1: O tio do nó Novo é vermelho.
- Caso 2: O tio do nó Novo é **preto** e Novo é filho da direita.
- Caso 3: O tio do nó Novo é **preto** e Novo é filho da esquerda.

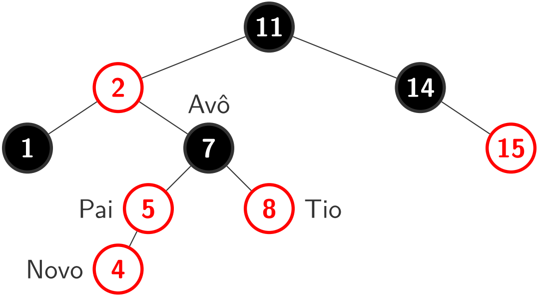
Inserção — Caso 1



Caso 1: O tio do nó Novo é vermelho.

- Como o pai e o tio são vermelhos, então o avô é obrigatoriamente preto.

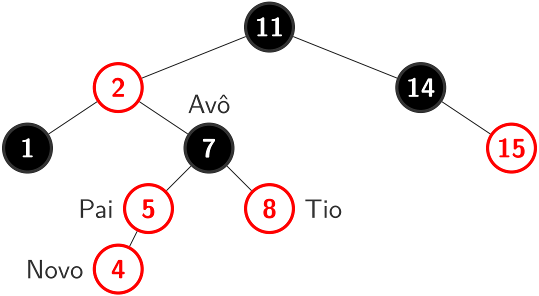
Inserção — Caso 1



Caso 1: O tio do nó Novo é vermelho.

- Como o pai e o tio são vermelhos, então o avô é obrigatoriamente preto.
- Troque a cor do pai e do tio para **preto**.

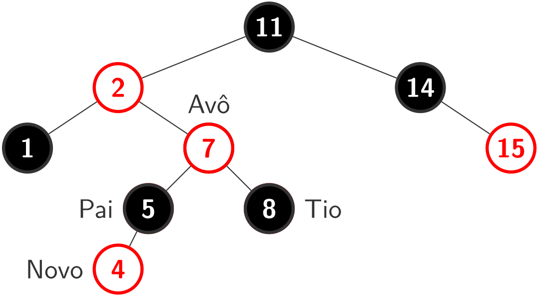
Inserção — Caso 1



Caso 1: O tio do nó Novo é vermelho.

- Como o pai e o tio são vermelhos, então o avô é obrigatoriamente preto.
- Troque a cor do pai e do tio para **preto**.
- Troque a cor do avô para vermelho.

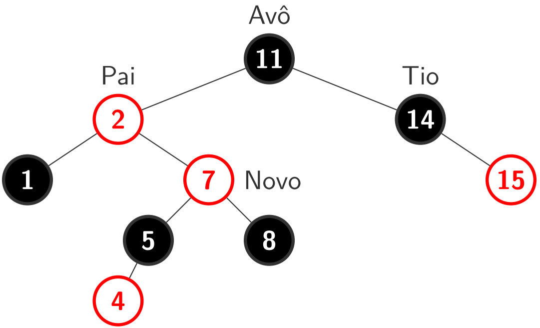
Inserção — Caso 1



Caso 1: O tio do nó Novo é vermelho.

- Como o pai e o tio são vermelhos, então o avô é obrigatoriamente preto.
- Troque a cor do pai e do tio para **preto**.
- Troque a cor do avô para vermelho.
- Neste ponto, consertamos o problema do nó Novo, mas possivelmente estragamos o avô.

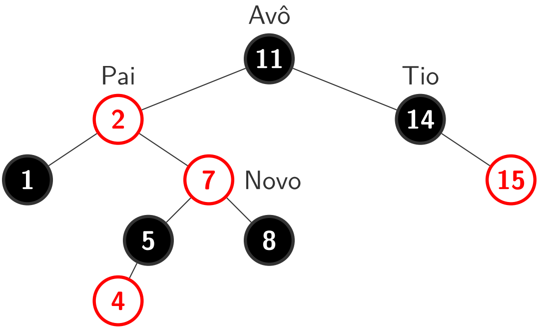
Inserção em árvores rubro-negras



Continuamos o conserto, agora tomando como referência o antigo avô, o nó com o valor 7.

- Caso 1: O tio de Novo é vermelho.
- Caso 2: O tio de Novo é **preto** e Novo é filho da direita.
- Caso 3: O tio de Novo é **preto** e Novo é filho da esquerda.

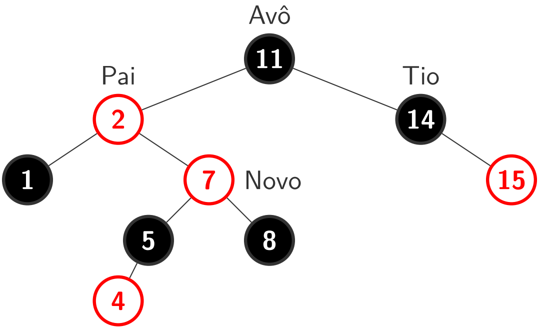
Inserção em árvores rubro-negras



Continuamos o conserto, agora tomando como referência o antigo avô, o nó com o valor 7.

- ~~Caso 1: O tio de Novo é vermelho.~~
- Caso 2: O tio de Novo é **preto** e Novo é filho da direita.
- Caso 3: O tio de Novo é **preto** e Novo é filho da esquerda.

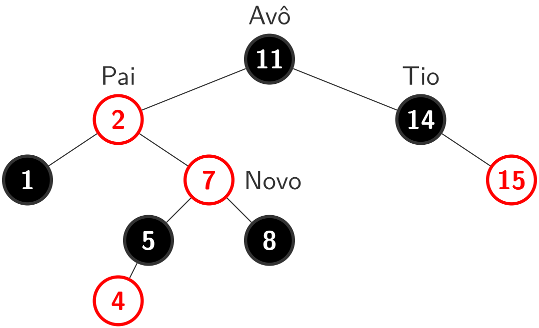
Inserção em árvores rubro-negras



Continuamos o conserto, agora tomando como referência o antigo avô, o nó com o valor 7.

- ~~Caso 1: O tio de Novo é vermelho.~~
- Caso 2: O tio de Novo é **preto** e Novo é filho da direita.
- ~~Caso 3: O tio de Novo é **preto** e Novo é filho da esquerda.~~

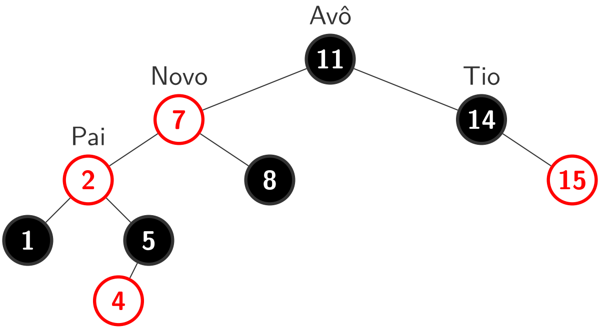
Inserção — Caso 2



Caso 2: O tio de Novo é **preto** e Novo é filho da direita.

- Executa uma rotação à esquerda tendo como pivô o pai.

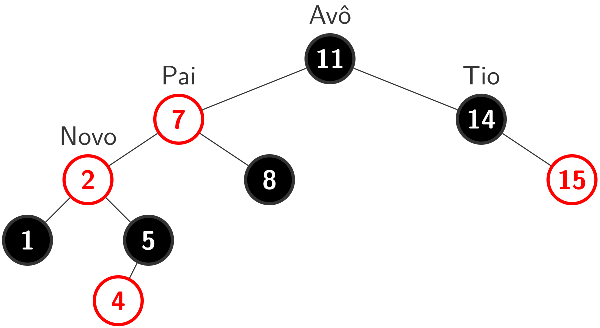
Inserção — Caso 2



Caso 2: O tio de Novo é **preto** e Novo é filho da direita.

- Executa uma rotação à esquerda tendo como pivô o pai.
- Neste ponto, consertamos o problema no Novo, mas possivelmente estragamos o nó Pai.

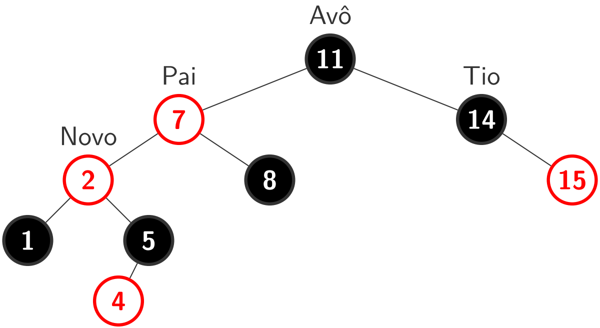
Inserções em árvores rubro-negras



Continuamos o conserto, agora tomando como referência o antigo pai, o nó com a chave 2 na figura acima.

- Caso 1: O tio de Novo é vermelho.
- Caso 2: O tio de Novo é **preto** e Novo é filho da direita.
- Caso 3: O tio de Novo é **preto** e Novo é filho da esquerda.

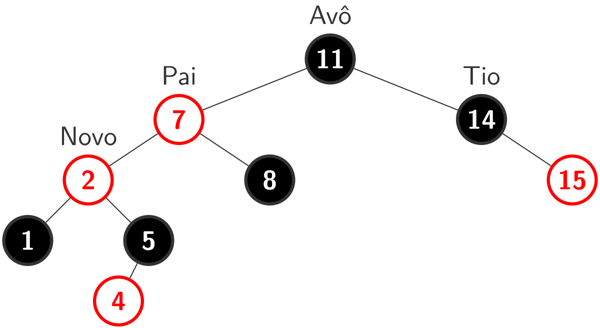
Inserções em árvores rubro-negras



Continuamos o conserto, agora tomando como referência o antigo pai, o nó com a chave 2 na figura acima.

- ~~Caso 1: O tio de Novo é vermelho.~~
- Caso 2: O tio de Novo é **preto** e Novo é filho da direita.
- Caso 3: O tio de Novo é **preto** e Novo é filho da esquerda.

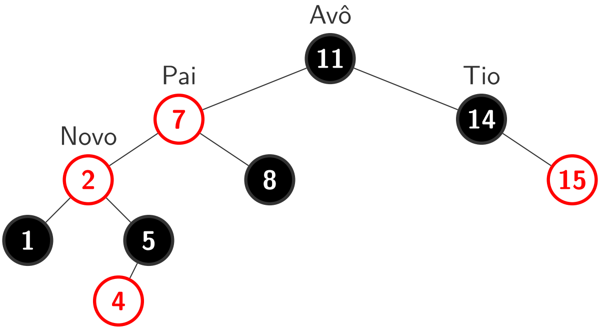
Inserções em árvores rubro-negras



Continuamos o conserto, agora tomando como referência o antigo pai, o nó com a chave 2 na figura acima.

- ~~Caso 1: O tio de Novo é vermelho.~~
- ~~Caso 2: O tio de Novo é **preto** e Novo é filho da direita.~~
- Caso 3: O tio de Novo é **preto** e Novo é filho da esquerda.

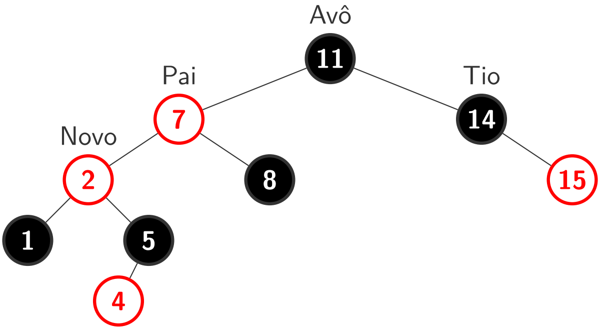
Inserção — Caso 3



Caso 3: O tio de Novo é **preto** e Novo é filho da esquerda.

- Troca a cor do pai para **preto**.
- Troca a cor do avô para **vermelho**.
- Executa uma rotação à direita tendo como pivô o avô.
- Neste ponto a árvore voltou a ser uma árvore rubro-negra válida.

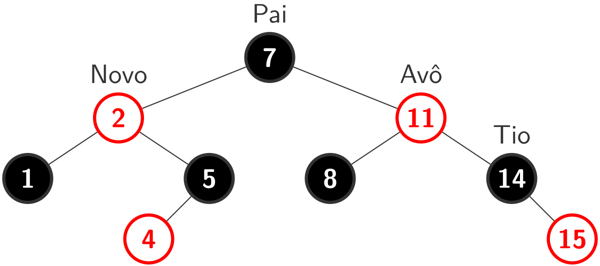
Inserção — Caso 3



Caso 3: O tio de Novo é **preto** e Novo é filho da esquerda.

- Troca a cor do pai para **preto**.
- Troca a cor do avô para **vermelho**.
- Executa uma rotação à direita tendo como pivô o avô.
- Neste ponto a árvore voltou a ser uma árvore rubro-negra válida.

Inserção — Caso 3



Caso 3: O tio de Novo é **preto** e Novo é filho da esquerda.

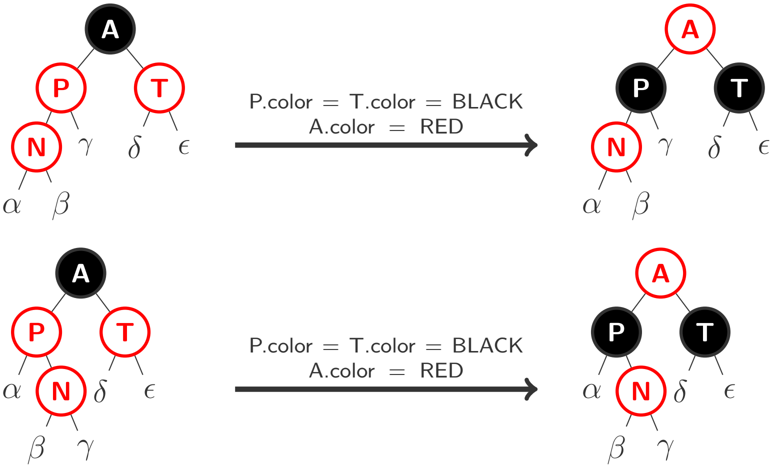
- Troca a cor do pai para **preto**.
- Troca a cor do avô para **vermelho**.
- Executa uma rotação à direita tendo como pivô o avô.
- Neste ponto a árvore voltou a ser uma árvore rubro-negra válida.

Corretude do Caso 1

> Vamos verificar que a recoloração realizada no Caso 1 mantém a altura negra da árvore!

- Vamos verificar apenas o caso em que o pai do nó Novo é filho esquerdo do avô de Novo.
- O caso em que o pai do nó Novo é filho direito do avô de Novo é simétrico.

Corretude do Caso 1



Propriedade 5 é mantida: Todos os caminhos saindo de um nó até uma de suas folhas possuem o mesmo número de nós pretos. Note que $\alpha, \beta, \gamma, \delta, \epsilon$ possuem todos a mesma altura negra (antes e depois).

Corretude dos Casos 2 e 3

O tio do nó inserido é preto

Vamos verificar que as rotações e recolorações realizadas nos Casos 2 e 3 mantêm a altura negra da árvore!

Corretude dos Casos 2 e 3

O tio do nó inserido é preto

Vamos verificar que as rotações e recolorações realizadas nos Casos 2 e 3 mantêm a altura negra da árvore!

- Se o tio é preto, então existem 4 configurações possíveis:

Corretude dos Casos 2 e 3

O tio do nó inserido é preto

Vamos verificar que as rotações e recolorações realizadas nos Casos 2 e 3 mantêm a altura negra da árvore!

- Se o tio é preto, então existem 4 configurações possíveis:

1. Configuração Esq-Dir (pai é filho esquerdo do avô e Novo é filho direito de pai) — Caso 2(a)

Corretude dos Casos 2 e 3

O tio do nó inserido é preto

Vamos verificar que as rotações e recolorações realizadas nos Casos 2 e 3 mantêm a altura negra da árvore!

- Se o tio é preto, então existem 4 configurações possíveis:

  1. Configuração Esq-Dir (pai é filho esquerdo do avô e Novo é filho direito de pai) — Caso 2(a)
  2. Configuração Esq-Esq (pai é filho esquerdo do avô e Novo é filho esquerdo de pai) — Caso 3(a)

Corretude dos Casos 2 e 3

O tio do nó inserido é preto

Vamos verificar que as rotações e recolorações realizadas nos Casos 2 e 3 mantêm a altura negra da árvore!

- Se o tio é preto, então existem 4 configurações possíveis:

  1. Configuração Esq-Dir (pai é filho esquerdo do avô e Novo é filho direito de pai) — Caso 2(a)
  2. Configuração Esq-Esq (pai é filho esquerdo do avô e Novo é filho esquerdo de pai) — Caso 3(a)
  3. Configuração Dir-Esq (simétrico da 1) — Caso 2(b)

Corretude dos Casos 2 e 3

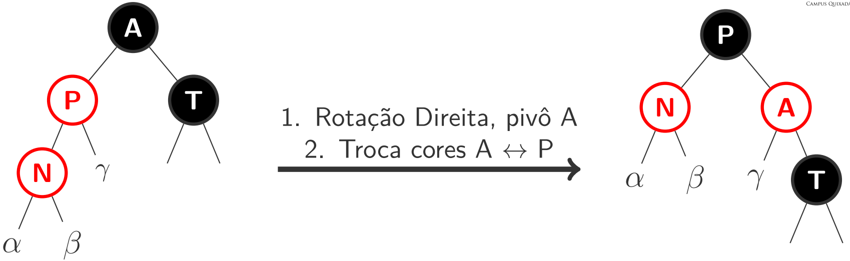
O tio do nó inserido é preto

Vamos verificar que as rotações e recolorações realizadas nos Casos 2 e 3 mantêm a altura negra da árvore!

- Se o tio é preto, então existem 4 configurações possíveis:

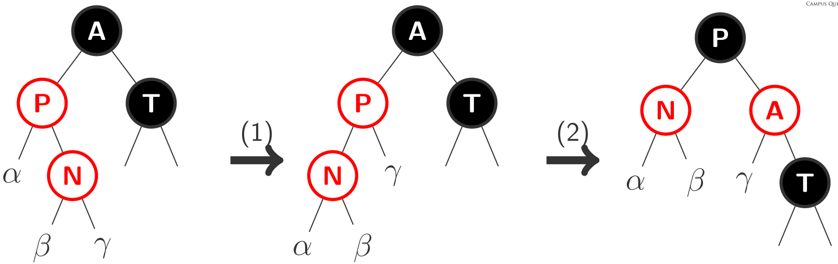
1. Configuração Esq-Dir (pai é filho esquerdo do avô e Novo é filho direito de pai) — Caso 2(a)
2. Configuração Esq-Esq (pai é filho esquerdo do avô e Novo é filho esquerdo de pai) — Caso 3(a)
3. Configuração Dir-Esq (simétrico da 1) — Caso 2(b)
4. Configuração Dir-Dir (simétrico da 2) — Caso 3(b)

Configuração Esq-Esq (Caso 3a)



- A = Avô
- P = Pai
- T = Tio
- N = Novo
- α, β, γ são subárvores.
- As subárvores α, β, γ e T possuem a mesma altura negra (antes e depois).

Configuração Esq-Dir (Caso 2a + Caso 3a)



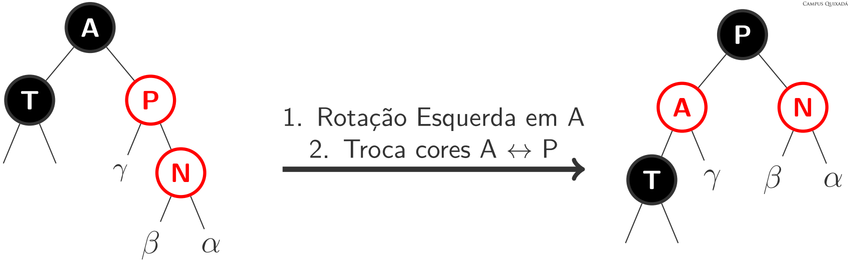
- A = Avô; P = Pai; T = Tio; N = Novo;
- α, β e γ são subárvores.
- As subárvores α, β, γ e T possuem a mesma altura negra (antes e depois).

(1) Efetua rotação à esquerda em torno do pai e troca $P \leftrightarrow N$.

(2) Aplica a configuração Esq-Esq (Caso 3a).

Configuração Dir-Dir (Caso 3b)

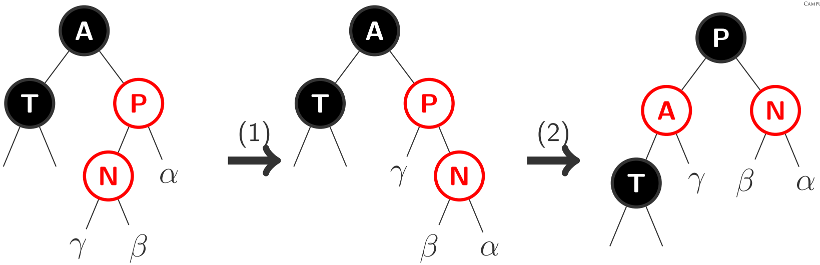
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



- A = Avô
- P = Pai
- T = Tio
- N = Novo
- α, β e γ são subárvores.
- As subárvores α, β, γ e T possuem a mesma altura negra (antes e depois).

Configuração Dir-Esq (Caso 2b + Caso 3b)

Universidade Federal do Ceará
Campus Quixadá



- A = Avô; P = Pai; T = Tio; N = Novo;
- α, β e γ são subárvores.
- As subárvores α, β, γ e T possuem a mesma altura negra (antes e depois).

(1) Efetua rotação à direita em torno do pai e troca $N \leftrightarrow P$.

(2) Aplica a configuração Dir-Dir (Caso 3b).

Código (incompleto) do conserto pós-inserção

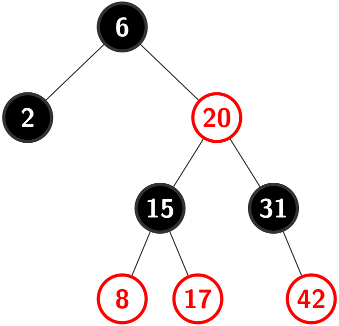
RB-INSERT-FIXUP(T, z)

```
while z.p.color == RED
    if z.p == z.p.p.left
        y = z.p.p.right
        if y.color == RED
            z.p.color = BLACK        // case 1
            y.color = BLACK          // case 1
            z.p.p.color = RED        // case 1
            z = z.p.p
        else
            if z == z.p.right
                z = z.p              // case 2a
                LEFT-ROTATE(T,z)     // case 2a
            z.p.color = BLACK        // case 3a
            z.p.p.color = RED        // case 3a
            RIGHT-ROTATE(T,z.p.p)    // case 3a
    else
        /* Simétrico ao código acima */
T.root.color = BLACK
```

Exercício

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Insira na árvore abaixo as seguintes chaves: 1, 5 e 19.



Remoção

Remoções em árvores rubro-negras

Universidade Federal do Ceará
Campus Quixadá

- Assim como no caso da inserção, nós utilizaremos rotações e recolorações para manter as propriedades da árvore rubro-negra.

Remoções em árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Assim como no caso da inserção, nós utilizaremos rotações e recolorações para manter as propriedades da árvore rubro-negra.
  - Contudo, a remoção de nós de uma árvore rubro-negra exige um pouco mais de trabalho.

Remoções em árvores rubro-negras

- Assim como no caso da inserção, nós utilizaremos rotações e recolorações para manter as propriedades da árvore rubro-negra.
  - Contudo, a remoção de nós de uma árvore rubro-negra exige um pouco mais de trabalho.

- Durante a inserção, baseamos as nossas operações de correção na cor do tio.

Remoções em árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Assim como no caso da inserção, nós utilizaremos rotações e recolorações para manter as propriedades da árvore rubro-negra.
  - Contudo, a remoção de nós de uma árvore rubro-negra exige um pouco mais de trabalho.

- Durante a inserção, baseamos as nossas operações de correção na cor do tio.
  - Já durante a remoção nós nos baseamos na cor do irmão do nó para decidir qual caso aplicar.

Remoções em árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Assim como no caso da inserção, nós utilizaremos rotações e recolorações para manter as propriedades da árvore rubro-negra.
  - Contudo, a remoção de nós de uma árvore rubro-negra exige um pouco mais de trabalho.
- Durante a inserção, baseamos as nossas operações de correção na cor do tio.
  - Já durante a remoção nós nos baseamos na cor do irmão do nó para decidir qual caso aplicar.
- Durante a inserção o principal problema que enfrentamos era ter um pai e um filho ambos vermelhos.

Remoções em árvores rubro-negras

- Assim como no caso da inserção, nós utilizaremos rotações e recolorações para manter as propriedades da árvore rubro-negra.
  - Contudo, a remoção de nós de uma árvore rubro-negra exige um pouco mais de trabalho.
- Durante a inserção, baseamos as nossas operações de correção na cor do tio.
  - Já durante a remoção nós nos baseamos na cor do irmão do nó para decidir qual caso aplicar.
- Durante a inserção o principal problema que enfrentamos era ter um pai e um filho ambos vermelhos.
  - Durante a remoção, se retirarmos um nó preto, estamos estragando a propriedade de altura negra da árvore.

Revisão — Remoção em árvore binária de busca

Universidade Federal do Ceará
Campus Quixadá

Problema: dada uma árvore binária de busca T e uma chave k, remover o nó com chave k de T (se existir) de modo que árvore binária resultante continue sendo uma árvore binária de busca.

- Reduzimos este problema ao problema de remover a raiz de uma árvore binária de busca.

Revisão — Remoção em árvore binária de busca

Problema: dada uma árvore binária de busca T e uma chave k, remover o nó com chave k de T (se existir) de modo que árvore binária resultante continue sendo uma árvore binária de busca.

- Reduzimos este problema ao problema de remover a raiz de uma árvore binária de busca.
- Caso o nó x a ser removido exista, dividimos o problema em 3 casos:

Revisão — Remoção em árvore binária de busca

Problema: dada uma árvore binária de busca T e uma chave k, remover o nó com chave k de T (se existir) de modo que árvore binária resultante continue sendo uma árvore binária de busca.

- Reduzimos este problema ao problema de remover a raiz de uma árvore binária de busca.
- Caso o nó x a ser removido exista, dividimos o problema em 3 casos:
  - Caso 1: a raiz não tem filhos. A árvore torna-se vazia.

Revisão — Remoção em árvore binária de busca

Problema: dada uma árvore binária de busca T e uma chave k, remover o nó com chave k de T (se existir) de modo que árvore binária resultante continue sendo uma árvore binária de busca.

- Reduzimos este problema ao problema de remover a raiz de uma árvore binária de busca.

- Caso o nó x a ser removido exista, dividimos o problema em 3 casos:
  - Caso 1: a raiz não tem filhos. A árvore torna-se vazia.
  - Caso 2: a raiz tem um único filho. Seu filho torna-se a nova raiz.

Revisão — Remoção em árvore binária de busca

Problema: dada uma árvore binária de busca T e uma chave k, remover o nó com chave k de T (se existir) de modo que árvore binária resultante continue sendo uma árvore binária de busca.

- Reduzimos este problema ao problema de remover a raiz de uma árvore binária de busca.

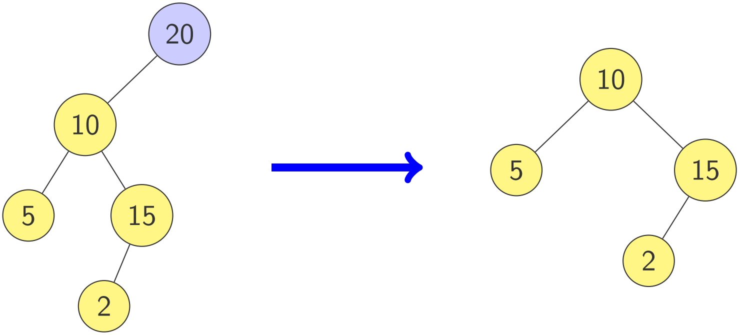
- Caso o nó x a ser removido exista, dividimos o problema em 3 casos:
  - Caso 1: a raiz não tem filhos. A árvore torna-se vazia.
  - Caso 2: a raiz tem um único filho. Seu filho torna-se a nova raiz.
  - Caso 3: a raiz tem dois filhos. Neste caso, tomamos o sucessor da antiga raiz como a nova raiz.

Revisão — Remoção em árvore binária de busca

- Caso 1: a raiz não tem filhos. A árvore torna-se vazia.

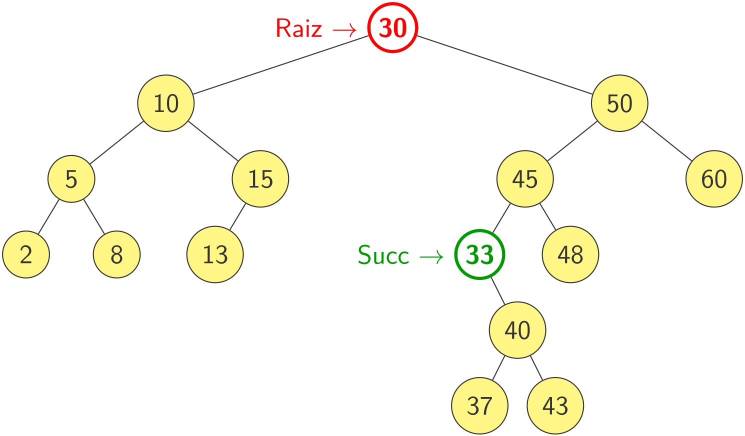
Revisão — Remoção em árvore binária de busca

- Caso 2: a raiz tem um único filho. Seu filho torna-se a nova raiz.



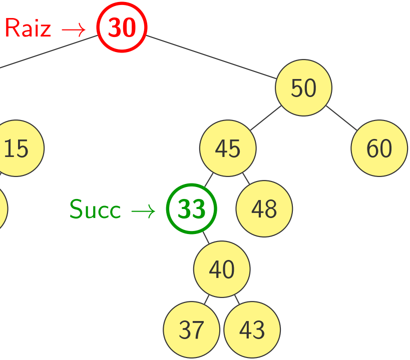
Revisão — Remoção em árvore binária de busca

- Caso 3: a raiz tem ambos os filhos não-nulos. Neste caso, tomamos o sucessor da antiga raiz como a nova raiz.



Revisão — Remoção em árvore binária de busca

- Como implementar o Caso 3: copiamos a informação do nó sucessor para a raiz e removemos o sucessor (recaindo no Caso 1 ou 2).



Função-membro `Remove`

Esta função recebe como argumentos a árvore T e a chave k a ser removida. Ela faz uma busca pelo nó p contendo esta chave. Se o nó p existir, é chamada a função `RB-Delete` para removê-lo.

```
REMOVE(T, k)
p = T.root
while p ≠ T.NIL and p.key ≠ k do
    if k < p.key then
        p = p.left
    else
        p = p.right
    end if
end while
if p ≠ T.NIL then
    RB-Delete(T, p)
end if
```

Função-membro `RB-Delete`

RB-DELETE(T, z)

```
if z.left == T.NIL or z.right == T.NIL
    y = z
else
    y = Minimum(z.right)
if y.left != T.NIL
    x = y.left
else
    x = y.right
x.p = y.p
if y.p == T.NIL
    T.root = x
else
    if y == y.p.left
        y.p.left = x
    else
        y.p.right = x
if y != z
    z.key = y.key
if y.color == BLACK
    RB-Delete-Fixup(T,x)
delete y
```

Remoção em árvores rubro-negras — Problemas

Universidade Federal do Ceará
Campus Quixadá

Que problemas podemos criar ao removermos fisicamente um nó?

- Remoção de um nó vermelho: Não causa problemas no balanceamento.

Remoção em árvores rubro-negras — Problemas

Que problemas podemos criar ao removermos fisicamente um nó?

- Remoção de um nó vermelho: Não causa problemas no balanceamento.
  - Nenhuma altura negra mudou.

Remoção em árvores rubro-negras — Problemas

Que problemas podemos criar ao removermos fisicamente um nó?

- Remoção de um nó vermelho: Não causa problemas no balanceamento.
  - Nenhuma altura negra mudou.
  - Nenhum nó vermelho se tornou vizinho de outro vermelho.

Remoção em árvores rubro-negras — Problemas

Que problemas podemos criar ao removermos fisicamente um nó?

- Remoção de um nó vermelho: Não causa problemas no balanceamento.
  - Nenhuma altura negra mudou.
  - Nenhum nó vermelho se tornou vizinho de outro vermelho.
  - Como o nó é vermelho, ele não era raiz e portanto a raiz permanece preta.

Remoção em árvores rubro-negras — Problemas

Que problemas podemos criar ao removermos fisicamente um nó?

- Remoção de um nó vermelho: Não causa problemas no balanceamento.
  - Nenhuma altura negra mudou.
  - Nenhum nó vermelho se tornou vizinho de outro vermelho.
  - Como o nó é vermelho, ele não era raiz e portanto a raiz permanece preta.

- Remoção de um nó preto: Mais de uma propriedade da árvore rubro-negra será quebrada. Quais?

Remoção em árvores rubro-negras — Problemas

Considere que o nó y a ser fisicamente removido é **preto**:

(P1) Um nó é vermelho ou é **preto**.

(P2) A raiz é preta.

(P3) Toda folha (NULL) é preta.

(P4) Se um nó é vermelho então ambos os seus filhos são pretos.

(P5) Para cada nó p, todos os caminhos desde p até as folhas contêm o mesmo número de nós pretos

Remoção em árvores rubro-negras — Problemas

Considere que o nó y a ser fisicamente removido é **preto**:

- Se y era raiz da árvore e um filho vermelho de y se torna raiz quebramos a Propriedade 2. (easy!)

(P1) Um nó é vermelho ou é **preto**.

(P2) A raiz é preta.

(P3) Toda folha (NULL) é preta.

(P4) Se um nó é vermelho então ambos os seus filhos são pretos.

(P5) Para cada nó p, todos os caminhos desde p até as folhas contêm o mesmo número de nós pretos

Remoção em árvores rubro-negras — Problemas

Considere que o nó $\mathtt{y}$ a ser fisicamente removido é **preto**:

- Se $\mathtt{y}$ era raiz da árvore e um filho vermelho de $\mathtt{y}$ se torna raiz quebramos a Propriedade 2. (easy!)
- Se um filho vermelho de $\mathtt{y}$ toma seu lugar e o pai de $\mathtt{y}$ também é vermelho, então, agora violamos a Propriedade 4. (easy!)

(P1) Um nó é vermelho ou é **preto**.

(P2) A raiz é preta.

(P3) Toda folha (NULL) é preta.

(P4) Se um nó é vermelho então ambos os seus filhos são pretos.

(P5) Para cada nó p, todos os caminhos desde p até as folhas contêm o mesmo número de nós pretos

Remoção em árvores rubro-negras — Problemas

Considere que o nó y a ser fisicamente removido é **preto**:

- Se y era raiz da árvore e um filho vermelho de y se torna raiz quebramos a Propriedade 2. (easy!)
- Se um filho vermelho de y toma seu lugar e o pai de y também é vermelho, então, agora violamos a Propriedade 4. (easy!)
- A remoção de y faz com que qualquer caminho que continha y anteriormente tenha um nó preto a menos. Desse modo quebramos a Propriedade 5.

(P1) Um nó é vermelho ou é **preto**.
(P2) A raiz é preta.
(P3) Toda folha (NULL) é preta.
(P4) Se um nó é vermelho então ambos os seus filhos são pretos.
(P5) Para cada nó p, todos os caminhos desde p até as folhas contêm o mesmo número de nós pretos

Remoção — Primeiro passo do algoritmo

Passo 1: Execute a remoção padrão da Árvore Binária de Busca.

- Na remoção padrão, sempre excluímos um nó que é folha ou que tem apenas um filho (para um nó que tem dois filhos, copiamos a chave do seu sucessor nele e depois removemos o sucessor).

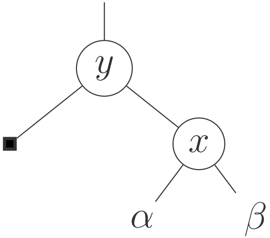
Remoção — Primeiro passo do algoritmo

Universidade Federal do Ceará
Campus Quixadá

Passo 1: Execute a remoção padrão da Árvore Binária de Busca.

- Na remoção padrão, sempre excluímos um nó que é folha ou que tem apenas um filho (para um nó que tem dois filhos, copiamos a chave do seu sucessor nele e depois removemos o sucessor).

- Seja `y` o nó a ser excluído e seja `x` o filho que substitui `y`.

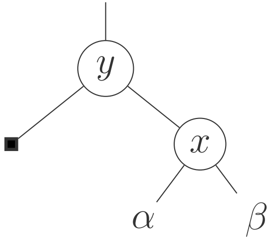


Remoção — Primeiro passo do algoritmo

Passo 1: Execute a remoção padrão da Árvore Binária de Busca.

- Na remoção padrão, sempre excluímos um nó que é folha ou que tem apenas um filho (para um nó que tem dois filhos, copiamos a chave do seu sucessor nele e depois removemos o sucessor).

- Seja `y` o nó a ser excluído e seja `x` o filho que substitui `y`.
    - Note que `x` é `nil` quando `y` é uma folha e a cor de `nil` é **BLACK**.

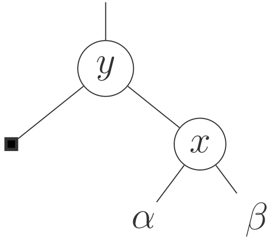


Remoção — Primeiro passo do algoritmo

Passo 1: Execute a remoção padrão da Árvore Binária de Busca.

- Na remoção padrão, sempre excluímos um nó que é folha ou que tem apenas um filho (para um nó que tem dois filhos, copiamos a chave do seu sucessor nele e depois removemos o sucessor).

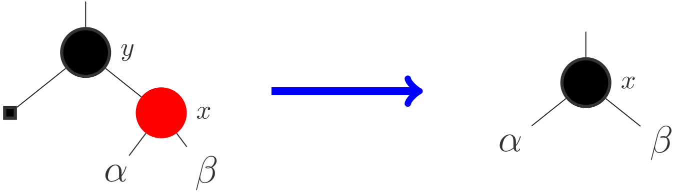
- Seja `y` o nó a ser excluído e seja `x` o filho que substitui `y`.
    - Note que `x` é `nil` quando `y` é uma folha e a cor de `nil` é **BLACK**.



- Se a cor de `y` for **BLACK** precisamos consertar a altura negra!

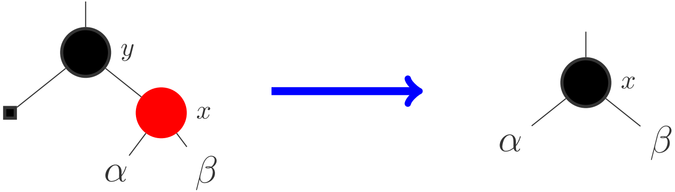
Análise das cores de `y` e `x` — Caso Fácil

- Sabemos que `y->color = BLACK` e `y` vai ser removido.



Análise das cores de y e x — Caso Fácil

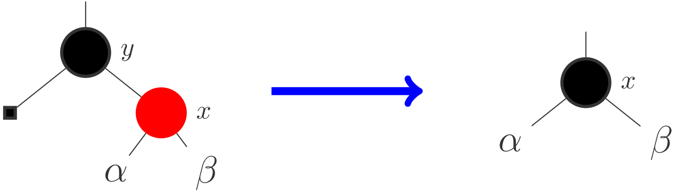
- Sabemos que `y->color = BLACK` e y vai ser removido.
- Se `x->color = RED`, então fazemos `x->color = BLACK` (não há alteração em nenhuma altura negra).



Análise das cores de `y` e `x` — Caso Fácil

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

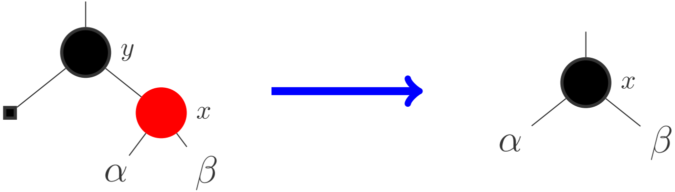
- Sabemos que `y->color = BLACK` e `y` vai ser removido.
- Se `x->color = RED`, então fazemos `x->color = BLACK` (não há alteração em nenhuma altura negra).



- Esse caso engloba os dois casos easy que vimos anteriormente:

Análise das cores de `y` e `x` — Caso Fácil

- Sabemos que `y->color = BLACK` e `y` vai ser removido.
- Se `x->color = RED`, então fazemos `x->color = BLACK` (não há alteração em nenhuma altura negra).

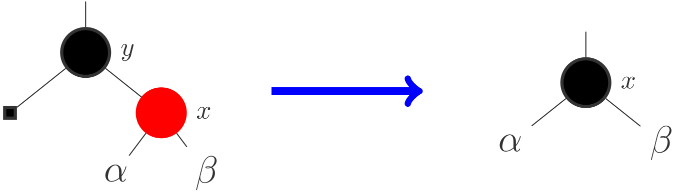


- Esse caso engloba os dois casos easy que vimos anteriormente:
  - se `y` era raiz da árvore;

Análise das cores de `y` e `x` — Caso Fácil

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

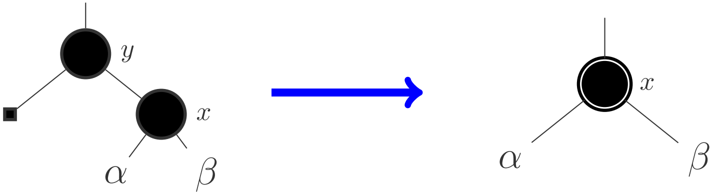
- Sabemos que `y->color = BLACK` e `y` vai ser removido.
- Se `x->color = RED`, então fazemos `x->color = BLACK` (não há alteração em nenhuma altura negra).



- Esse caso engloba os dois casos easy que vimos anteriormente:
  - se `y` era raiz da árvore;
  - e, caso contrário, se `y` tem pai != NIL e esse pai é vermelho.

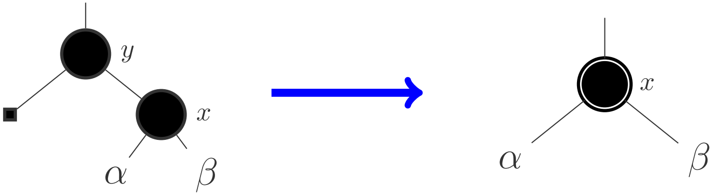
Análise das cores de y e x — Caso Ruim

- O verdadeiro problema na remoção ocorre quando ambos y e x são **pretos**.



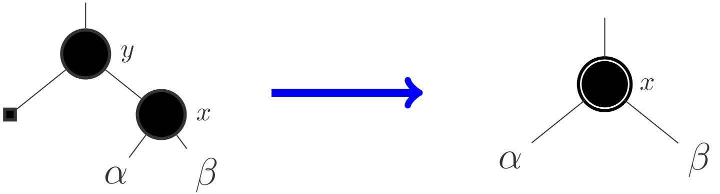
Análise das cores de y e x — Caso Ruim

- O verdadeiro problema na remoção ocorre quando ambos y e x são **pretos**.
- Neste caso, ao remover o nó y, as alturas negras de todos os seus ancestrais ficam "corrompidas".



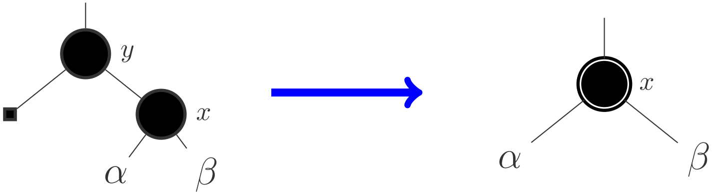
Análise das cores de `y` e `x` — Caso Ruim

- O verdadeiro problema na remoção ocorre quando ambos `y` e `x` são **pretos**.
- Neste caso, ao remover o nó `y`, as alturas negras de todos os seus ancestrais ficam "corrompidas".
- A fim de consertar este caso problemático, usamos a noção de **preto duplo**.



Análise das cores de `y` e `x` — Caso Ruim

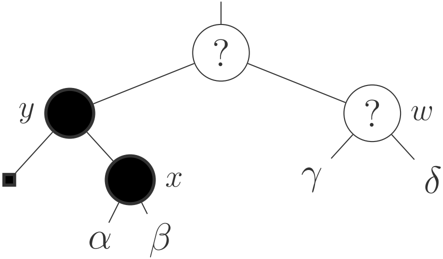
- O verdadeiro problema na remoção ocorre quando ambos `y` e `x` são **pretos**.
- Neste caso, ao remover o nó `y`, as alturas negras de todos os seus ancestrais ficam “corrompidas”.
- A fim de consertar este caso problemático, usamos a noção de **preto duplo**.
  - Quando um nó preto é excluído e substituído por um filho preto, o filho é marcado como **preto duplo**. A tarefa principal agora é converter esse **preto duplo** em **preto**.



Segundo passo: Eliminação do preto duplo

Usamos a seguinte nomenclatura para os nós envolvidos no processo de remoção:

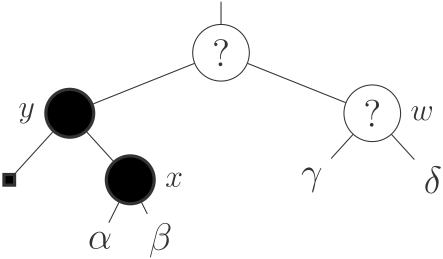
- `z` — O nó cuja chave será removida.



Segundo passo: Eliminação do preto duplo

Usamos a seguinte nomenclatura para os nós envolvidos no processo de remoção:

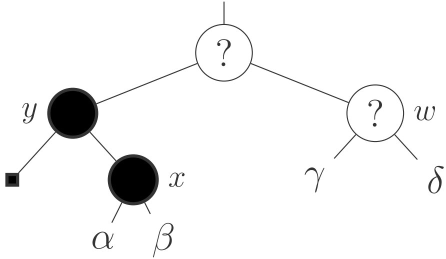
- `z` — O nó cuja chave será removida.
- `y` — O nó que será fisicamente removido. Onde `y = z` se `z` possui 0 ou 1 filho. Caso contrário, `y = sucessor(z)`.



Segundo passo: Eliminação do preto duplo

Usamos a seguinte nomenclatura para os nós envolvidos no processo de remoção:

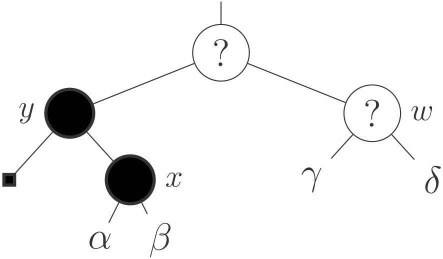
- `z` — O nó cuja chave será removida.
- `y` — O nó que será fisicamente removido. Onde `y = z` se `z` possui 0 ou 1 filho. Caso contrário, `y = sucessor(z)`.
- `x` — O único filho de `y` antes de qualquer modificação, ou `NIL` caso `y` não tenha filhos.



Segundo passo: Eliminação do preto duplo

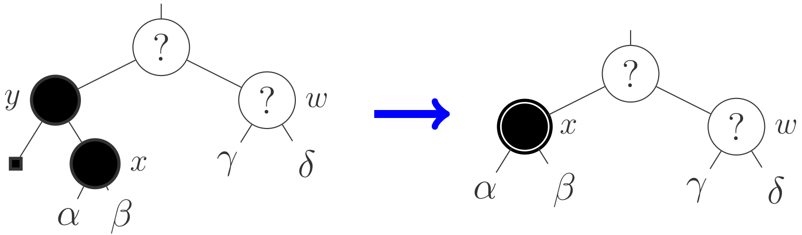
Usamos a seguinte nomenclatura para os nós envolvidos no processo de remoção:

- `z` — O nó cuja chave será removida.
- `y` — O nó que será fisicamente removido. Onde `y = z` se `z` possui 0 ou 1 filho. Caso contrário, `y = sucessor(z)`.
- `x` — O único filho de `y` antes de qualquer modificação, ou `NIL` caso `y` não tenha filhos.
- `w` — O tio de `x` (irmão de `y`) antes da remoção de `y`.



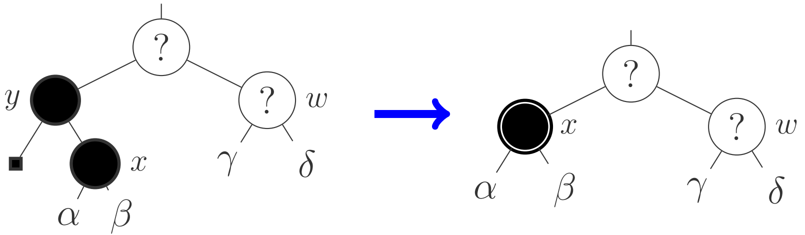
Segundo passo: Eliminação do duplo preto

- Durante o conserto da árvore, quando considerarmos o nó `x` vamos contá-lo como preto duas vezes (**duplo preto**), a fim de manter a Propriedade 5.



Segundo passo: Eliminação do duplo preto

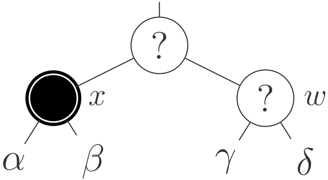
- Durante o conserto da árvore, quando considerarmos o nó `x` vamos contá-lo como preto duas vezes (**duplo preto**), a fim de manter a Propriedade 5.
- `x` aponta para um **duplo preto**, o que nos diz que o nó `w` existe (Por quê?)



Eliminação do duplo preto — 4 Casos

Considere que o nó `y` foi removido. Agora, `w` é o irmão de `x`. Existem 4 casos a considerar:

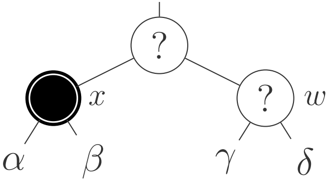
- Caso 1: `w` é vermelho.



Eliminação do duplo preto — 4 Casos

Considere que o nó `y` foi removido. Agora, `w` é o irmão de `x`.
Existem 4 casos a considerar:

- Caso 1: `w` é vermelho.
- Caso 2: `w` é **preto** e seus dois filhos são **pretos**.

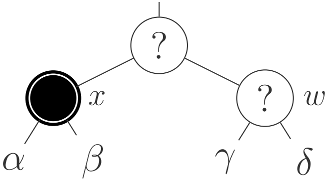


Eliminação do duplo preto — 4 Casos

Universidade Federal do Ceará
Campus Quixadá

Considere que o nó `y` foi removido. Agora, `w` é o irmão de `x`.
Existem 4 casos a considerar:

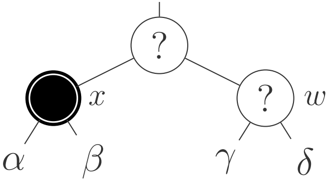
- Caso 1: `w` é vermelho.
- Caso 2: `w` é **preto** e seus dois filhos são **pretos**.
- Caso 3: `w` é **preto**, seu filho esquerdo é vermelho e seu filho direito é **preto**.



Eliminação do duplo preto — 4 Casos

Considere que o nó `y` foi removido. Agora, `w` é o irmão de `x`.
Existem 4 casos a considerar:

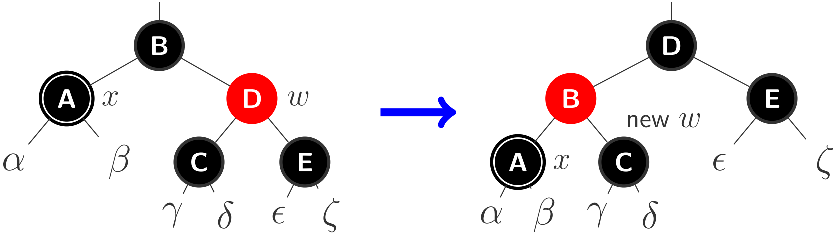
- Caso 1: `w` é vermelho.
- Caso 2: `w` é **preto** e seus dois filhos são **pretos**.
- Caso 3: `w` é **preto**, seu filho esquerdo é vermelho e seu filho direito é **preto**.
- Caso 4: `w` é **preto** e seu filho direito é vermelho.



Caso 1: o nó w é vermelho

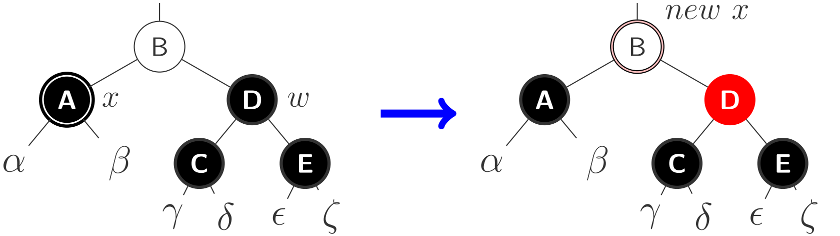
Como w é vermelho, ambos os filhos são **pretos**. Logo:

- Trocamos as cores de `w` e `x.parent`.
- Efetuamos uma rotação à esquerda tendo como pivô `x.parent`.
- Essas alterações não violam nenhuma propriedade da árvore rubro-negra.
- Mas transformam o Caso 1 no Caso 2, Caso 3 ou Caso 4.



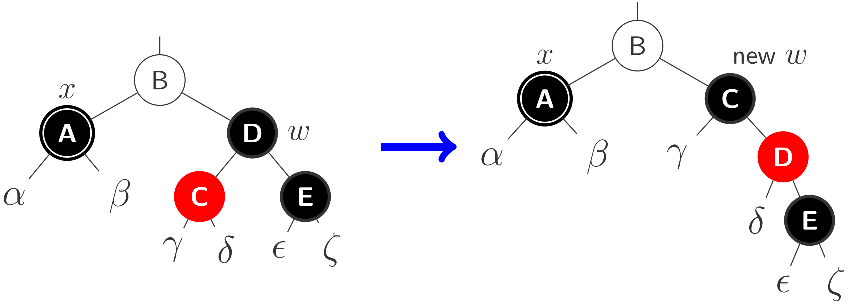
Caso 2: o nó w e seus dois filhos são pretos

- Retiramos um preto de x e um de w deixando x com apenas um **preto** e deixando w vermelho.
- Para compensar a remoção de um preto de `x` e de `w`, adicionamos um **preto extra** no `x.parent` que era originalmente **preto** ou vermelho.
- Note que se `x.parent` era vermelho, então terminou. Pintamo-lo de preto e acabou. Caso contrário, jogamos a bomba para cima, tratando o `x.parent` como sendo o novo `x`.



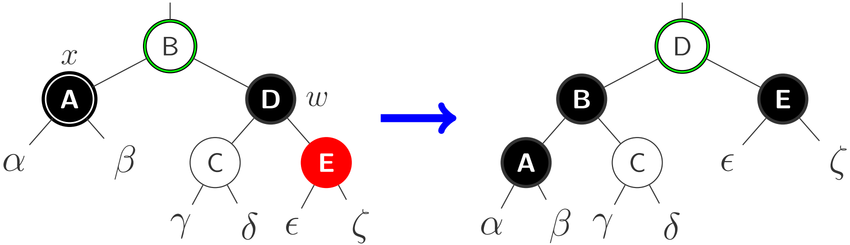
Caso 3: w e w.right são pretos, w.left é vermelho

- Trocamos as cores de `w` e de seu filho esquerdo.
- Rotaciona árvore à direita usando como pivô `w`.
- Essas operações não introduzem violações
- Neste ponto, o novo irmão `w` de `x` é um nó **preto** com um filho da direita vermelho. Estamos, portanto, no Caso 4.



Caso 4: w é preto e w.right é vermelho

- `w` é pintado com a cor de `x.parent`
- Pinta `x.parent` de **preto**
- Pinta o filho direito de `w` de **preto**
- Rotaciona árvore à esquerda usando como pivô `x.parent`



- Após a execução desse caso, nenhuma regulagem adicional será necessária.

Pseudocódigo do conserto da Remoção

Função-membro RB-Delete-FixUp

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
RB-DELETE-FIXUP(T, x)
1  while x ≠ T.root and x.color == BLACK
2      if x == x.p.left                  // is x a left child?
3          w = x.p.right                 // w is x's sibling
4          if w.color == RED
5              w.color = BLACK
6              x.p.color = RED                        case 1
7              LEFT-ROTATE(T, x.p)
8              w = x.p.right
9          if w.left.color == BLACK and w.right.color == BLACK
10             w.color = RED                          case 2
11             x = x.p
12         else
13             if w.right.color == BLACK
14                 w.left.color = BLACK
15                 w.color = RED                      case 3
16                 RIGHT-ROTATE(T, w)
17                 w = x.p.right
18             w.color = x.p.color
19             x.p.color = BLACK
20             w.right.color = BLACK                  case 4
21             LEFT-ROTATE(T, x.p)
22             x = T.root
23     else // same as lines 3–22, but with "right" and "left" exchanged
44 x.color = BLACK
```

AVL versus Rubro-Negra

AVL × Rubro-negra

- Na teoria, possuem a mesma complexidade computacional

AVL $\times$ Rubro-negra

- Na teoria, possuem a mesma complexidade computacional
  - Inserção, remoção e busca: $O(\lg(n))$.

AVL $\times$ Rubro-negra

- Na teoria, possuem a mesma complexidade computacional
  - Inserção, remoção e busca: $O(\lg(n))$.
- Na prática, a árvore AVL é mais rápida na operação de busca, e mais lenta nas operações de inserção e remoção.

AVL $\times$ Rubro-negra

- Na teoria, possuem a mesma complexidade computacional
  - Inserção, remoção e busca: $O(\lg(n))$.
- Na prática, a árvore AVL é mais rápida na operação de busca, e mais lenta nas operações de inserção e remoção.
  - A árvore AVL é mais balanceada do que a árvore rubro-negra, o que acelera a operação de busca.

AVL × Rubro-negra

Universidade Federal do Ceará
Campus Quixadá

- Na teoria, possuem a mesma complexidade computacional
  - Inserção, remoção e busca: $O(\lg(n))$.
- Na prática, a árvore AVL é mais rápida na operação de busca, e mais lenta nas operações de inserção e remoção.
  - A árvore AVL é mais balanceada do que a árvore rubro-negra, o que acelera a operação de busca.
  - A árvore rubro-negra realiza menor quantidade de rotações ao inserir ou remover nós.

AVL $\times$ Rubro-negra

- AVL: balanceamento mais rígido.
  - No pior caso, uma operação de remoção pode exigir $O(\lg(n))$ rotações na árvore AVL, mas apenas 3 rotações na árvore rubro-negra.

AVL $\times$ Rubro-negra

- AVL: balanceamento mais rígido.
  - No pior caso, uma operação de remoção pode exigir $O(\lg(n))$ rotações na árvore AVL, mas apenas 3 rotações na árvore rubro-negra.
- Qual usar?
  - Operação de busca é a mais usada? Melhor usar uma árvore AVL.
  - Inserção ou remoção são mais usadas? Melhor usar uma árvore Rubro-negra.

Exercícios

Exercícios

Universidade Federal do Ceará
Campus Quixadá

- É possível ter uma árvore rubro-negra em que todos os seus nós sejam pretos?
- Prove ou dê um contraexemplo: toda árvore rubro-negra é uma árvore AVL.
- Prove ou dê um contraexemplo: toda árvore AVL é rubro-negra.

FIM