Árvores Rubro-Negras

Estrutura de Dados Avançada — QXD0015

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

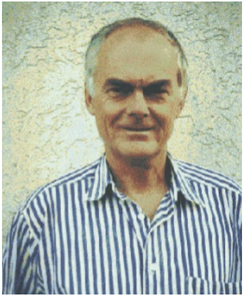
Universidade Federal do Ceará

1º semestre/2023

Árvores rubro-negras (Red-Black trees)

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- 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

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Uma árvore rubro-negra é uma árvore binária de busca.

- 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.
    - value – Valor associado à chave. Pode não ser único.
    - right – Subárvore direita.
    - left – Subárvore esquerda.
    - parent – Ponteiro para o pai do nó.

Árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Uma árvore rubro-negra é uma árvore binária de busca.

- 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.
  - value – Valor associado à chave. Pode não ser único.
  - right – Subárvore direita.
  - left – Subárvore esquerda.
  - parent – Ponteiro para o pai do nó.

- 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

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Uma árvore rubro-negra é uma árvore binária de busca.

- 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.
  - value – Valor associado à chave. Pode não ser único.
  - right – Subárvore direita.
  - left – Subárvore esquerda.
  - parent – Ponteiro para o pai do nó.

- 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

Uma árvore rubro-negra também satisfaz as seguintes propriedades:

Árvores rubro-negras

Uma árvore rubro-negra também satisfaz as seguintes propriedades:

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

Árvores rubro-negras

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

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

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**.

(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

Uma árvore rubro-negra também satisfaz as seguintes propriedades:

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

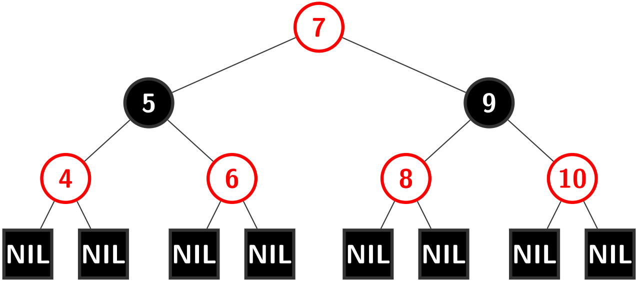
(P2) A raiz é **preta**.

(P3) Toda folha (NIL) é **preta**.

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

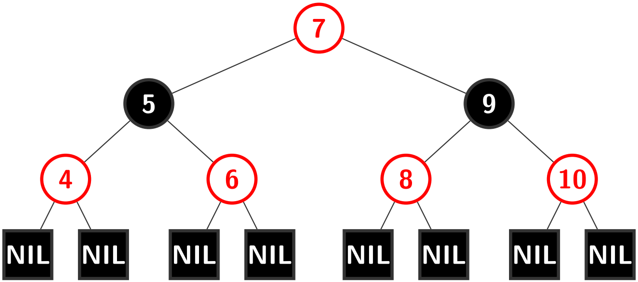
(P5) Para todo nó, todos os caminhos do nó até as folhas descendentes contêm o mesmo número de nós **pretos**.

Árvores rubro-negras – Reconhecendo 1



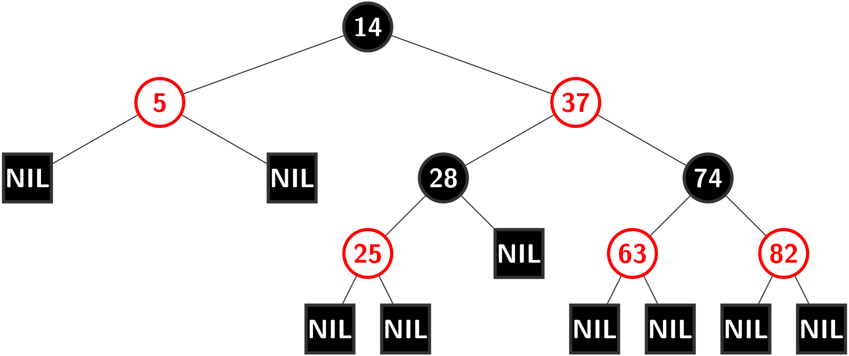
- É uma árvore rubro-negra?

Árvores rubro-negras – Reconhecendo 1



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

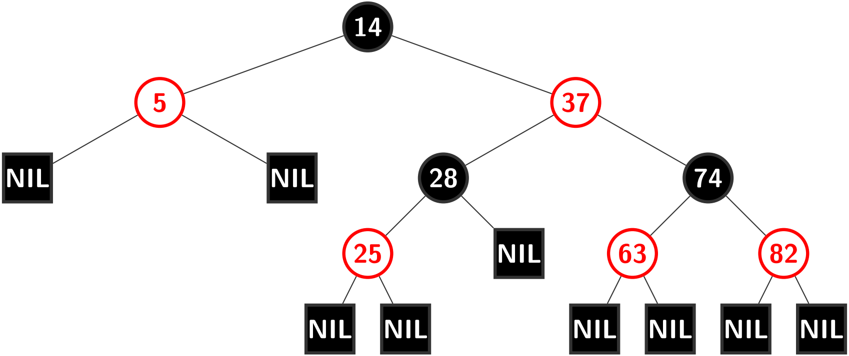
Árvores rubro-negras – Reconhecendo 2



- É uma árvore rubro-negra?

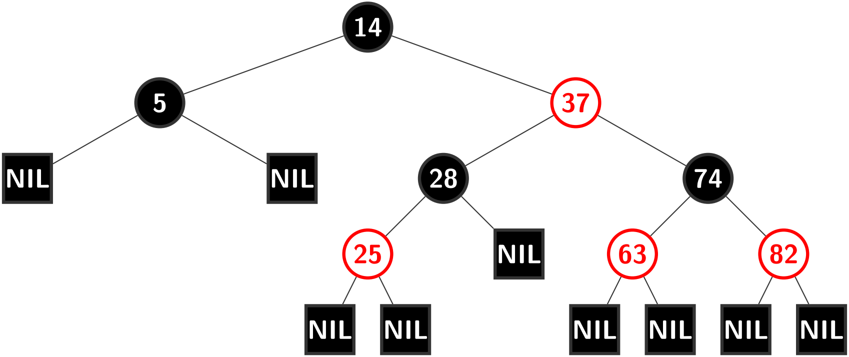
Árvores rubro-negras – Reconhecendo 2

Universidade Federal do Ceará
Campus Quixadá



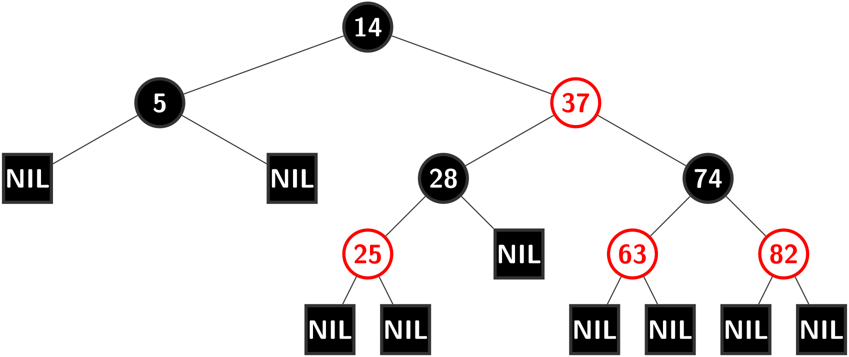
- É uma árvore rubro-negra?
- Não. Viola a Propriedade (5).

Árvores rubro-negras – Reconhecendo 3



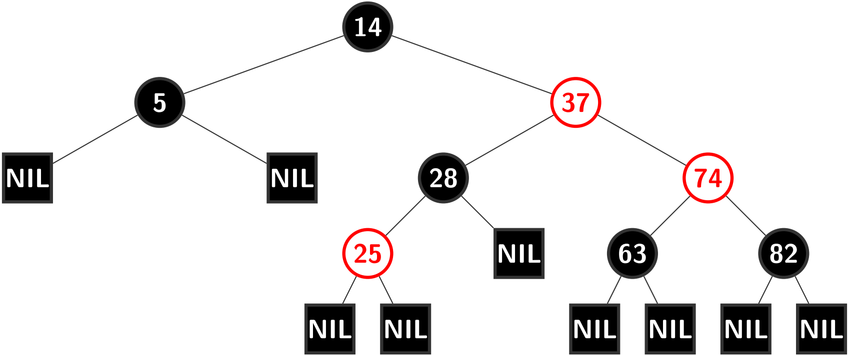
- É uma árvore rubro-negra?

Árvores rubro-negras – Reconhecendo 3



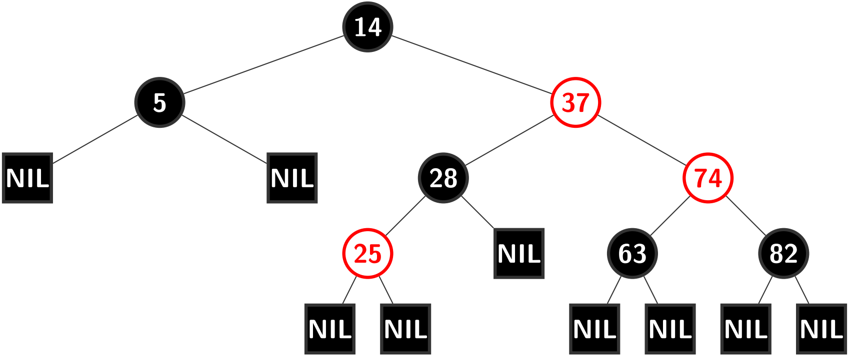
- É uma árvore rubro-negra?
- SIM!!!

Árvores rubro-negras – Reconhecendo 4



- É uma árvore rubro-negra?

Árvores rubro-negras – Reconhecendo 4



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

Ideia por trás da definição

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

Ideia por trás da definição

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

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

Ideia por trás da definição

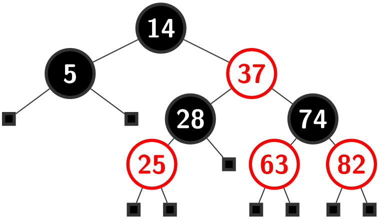
Qual a intuição por trás das propriedades da árvore rubro-negra? Como elas levam ao balanceamento?

- Restringindo a maneira que os nós podem ser coloridos do caminho da raíz 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.

Ideia por trás da definição

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

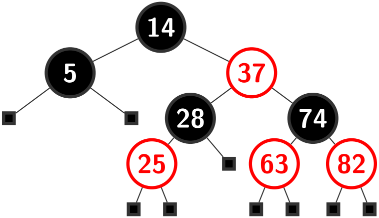
- Restringindo a maneira que os nós podem ser coloridos do caminho da raíz até qualquer uma das suas folhas, as árvores rubro-negras:
  - Garantem que nenhum dos caminhos será maior que 2 vezes o comprimento de qualquer outro.
  - Garantem que a árvore é aproximadamente balanceada.



Prova do balanceamento

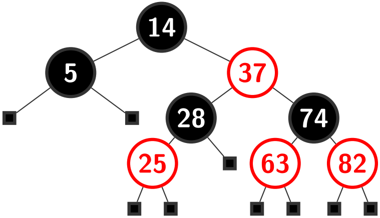
Definição de Altura negra

- A altura negra de um nó p é definida como o número de nós pretos (sem incluir o próprio p) visitados em qualquer caminho de p até as folhas.



Definição de Altura negra

- A altura negra de um nó p é definida como o número de nós pretos (sem incluir o próprio p) visitados em qualquer caminho de p até as folhas.
- A altura negra do nó p é denotada por $bh(n)$.



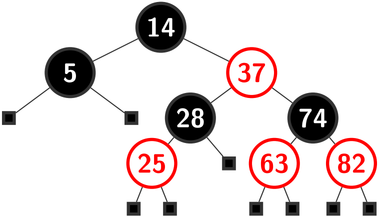
Definição de Altura negra

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- A altura negra de um nó p é definida como o número de nós pretos (sem incluir o próprio p) visitados em qualquer caminho de p até as folhas.

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

- Pela Propriedade (5), $bh(n)$ é bem definida para todos os nós da árvore. A altura negra da árvore rubro-negra é definida como sendo $bh(root)$.



Propriedades da árvore rubro-negra

Propriedade 1

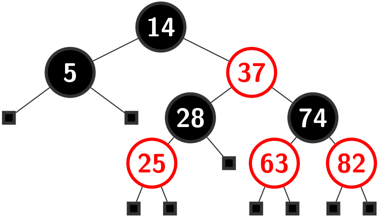
Seja v um nó com altura h. Então, $bh(v) \geq h/2$. $\square$

Propriedades da árvore rubro-negra

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Propriedade 2

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



Propriedades da árvore rubro-negra

Propriedade 2

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

Demonstração:

Propriedades da árvore rubro-negra

Propriedade 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.

Propriedades da árvore rubro-negra

Universidade Federal do Ceará
Campus Quixadá

Propriedade 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.

Propriedades da árvore rubro-negra

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Propriedade 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

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.

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.

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.

- 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. ■

Rubro-negras são balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Lema

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

Demonstração:

Rubro-negras são balanceadas

Lema

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

Demonstração:

- Seja h a altura da árvore rubro-negra com n nós.

Rubro-negras são balanceadas

Lema

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

Demonstração:

- Seja h a altura da árvore rubro-negra com n nós.

Rubro-negras são balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Lema

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

Demonstração:

- Seja h a altura da árvore rubro-negra com n nós.
- Pela **Propriedade 1**, pelo menos metade dos nós em qualquer caminho da raiz até uma folha (não incluindo a raiz) obrigatoriamente é preto.

Rubro-negras são balanceadas

Universidade Federal do Ceará
Campus Quixadá

Lema

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

Demonstração:

- Seja h a altura da árvore rubro-negra com n nós.
- Pela **Propriedade 1**, pelo menos metade dos nós em qualquer caminho da raiz até uma folha (não incluindo a raiz) obrigatoriamente é preto.

Rubro-negras são balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Lema

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

Demonstração:

- Seja h a altura da árvore rubro-negra com n nós.
- Pela **Propriedade 1**, pelo menos metade dos nós em qualquer caminho da raiz até uma folha (não incluindo a raiz) obrigatoriamente é preto.
- Consequentemente, a altura negra da raiz é pelo menos $h/2$. Assim, pela **Propriedade 2**, temos que $n \geq 2^{h/2} - 1$.

Rubro-negras são balanceadas

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Lema

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

Demonstração:

- Seja h a altura da árvore rubro-negra com n nós.
- Pela **Propriedade 1**, pelo menos metade dos nós em qualquer caminho da raiz até uma folha (não incluindo a raiz) obrigatoriamente é preto.
- Consequentemente, a altura negra da raiz é pelo menos $h/2$. Assim, pela **Propriedade 2**, temos que $n \geq 2^{h/2} - 1$.

Rubro-negras são balanceadas

Universidade Federal do Ceará
Campus Quixadá

Lema

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

Demonstração:

- Seja h a altura da árvore rubro-negra com n nós.
- Pela **Propriedade 1**, pelo menos metade dos nós em qualquer caminho da raiz até uma folha (não incluindo a raiz) obrigatoriamente é preto.
- Consequentemente, a altura negra da raiz é pelo menos $h/2$. Assim, pela **Propriedade 2**, temos que $n \geq 2^{h/2} - 1$.
- $n \geq 2^{h/2} - 1 \Longrightarrow n+1 \geq 2^{h/2} \Longrightarrow \lg(n+1) \geq \lg 2^{h/2}$
$\Longrightarrow \lg(n+1) \geq h/2 \Longrightarrow h \leq 2\lg(n+1)$, e o resultado segue. ■

Corolário

Corolário

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

Rotações

Rotações

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

Rotações

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

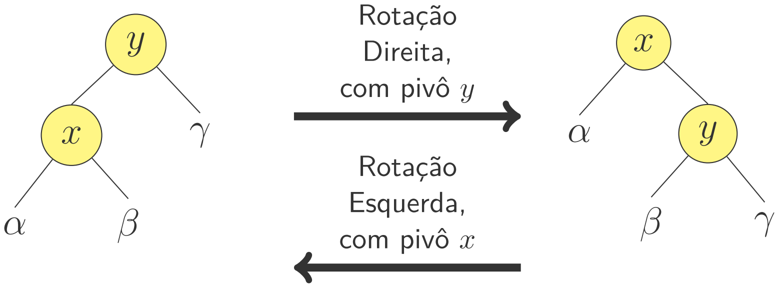
- Antes de vermos como fazer inserções em uma árvore rubro-negra, é preciso apresentar o conceito de rotações.
- Usamos as rotações para consertar parte do estrago feito pelas operações já conhecidas de inserção e remoção nas propriedades rubro-negras.
- O resto do estrago é consertado utilizando recoloração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

Rotações

Universidade Federal do Ceará
Campus Quixadá

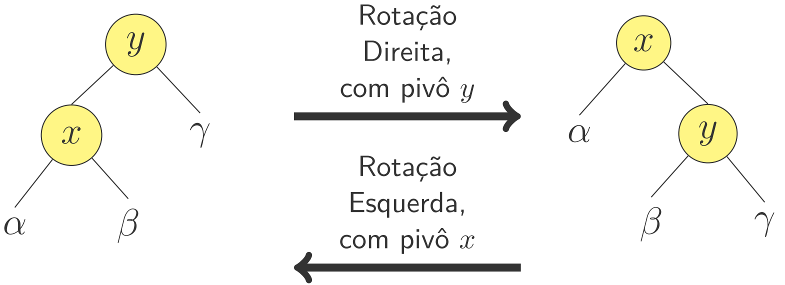
- Antes de vermos como fazer inserções em uma árvore rubro-negra, é preciso apresentar o conceito de rotações.
- Usamos as rotações para consertar parte do estrago feito pelas operações já conhecidas de inserção e remoção nas propriedades rubro-negras.
- O resto do estrago é consertado utilizando recoloração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

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
```

Supomos que `x->dir` não é nulo e que o pai da raiz é nulo (`NIL`).

Rotação à esquerda — Pseudocódigo

LEFT-ROTATE(T, x)

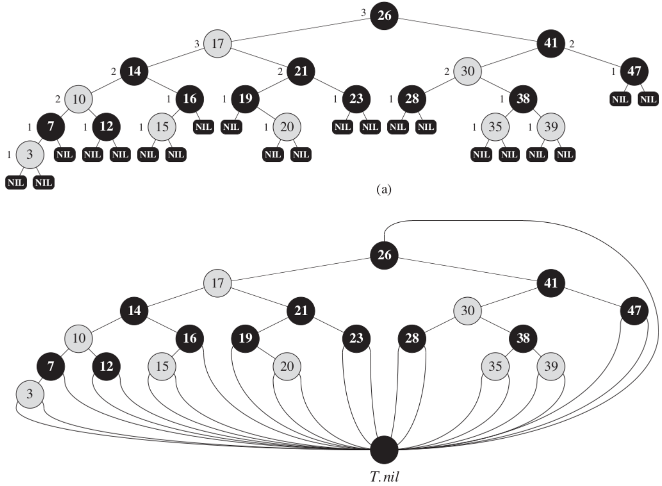
```
y = x.right
x.right = y.left
if y.left ≠ T.NIL
        y.left.p = x
y.p = x.p
if x.p == T.NIL
        T.root = y
elseif x == x.p.left
        x.p.left = y
else
        x.p.right = y
y.left = x
x.p = y
```

Supomos que `x->dir` não é nulo e que o pai da raiz é nulo (`NIL`).

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

Inserção

O nó T.NIL



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`

Visão geral da inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

Visão geral da inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

Visão geral da inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

Visão geral da inserção

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

Visão geral da inserção

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

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

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

- 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

Universidade Federal do Ceará
Campus Quixadá

- Um nó que satisfaz as regras de uma árvore rubro-negra é denominado equilibrado, caso contrário é dito desequilibrado.
- Em uma árvore rubro-negra todos os nós estão equilibrados.
- 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 quebrada, realizamos rotações e ajustamos as cores dos nós para que todas as propriedades continuem válidas.

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.

Inserções em árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

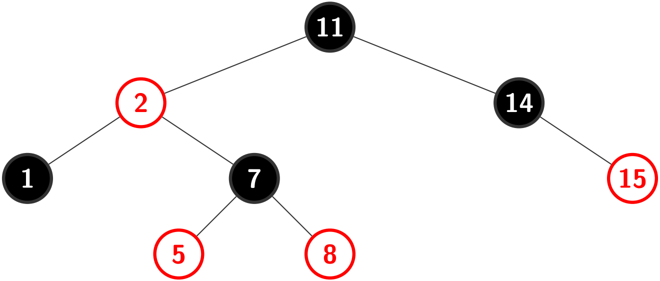
(P1) Um nó é vermelho ou é **preto**.
(P2) A raiz é **preta**.
(P3) Toda folha (NIL) é **preta**.
(P4) Se um nó é vermelho então ambos os seus filhos são **pretos**.
(P5) Para cada nó p, todos os caminhos desde p até as folhas contêm o mesmo número de nós pretos.

- 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

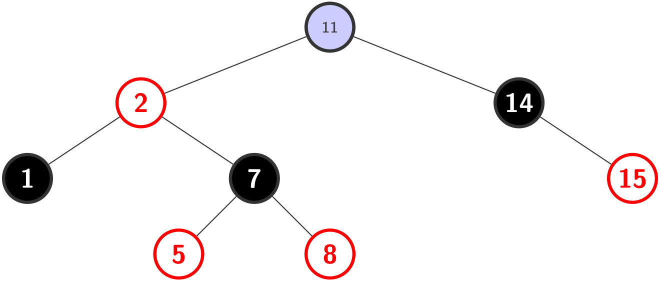
Revisando a inserção em árvore binária de busca

Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

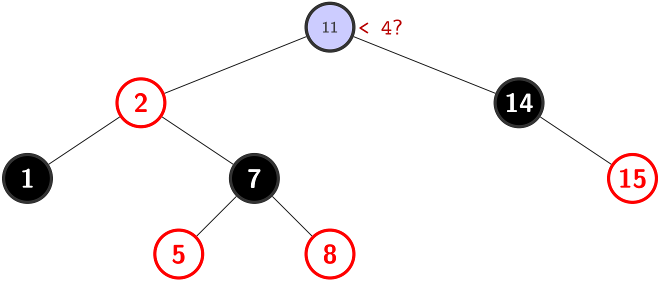
Revisando a inserção em árvore binária de busca
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

Revisando a inserção em árvore binária de busca

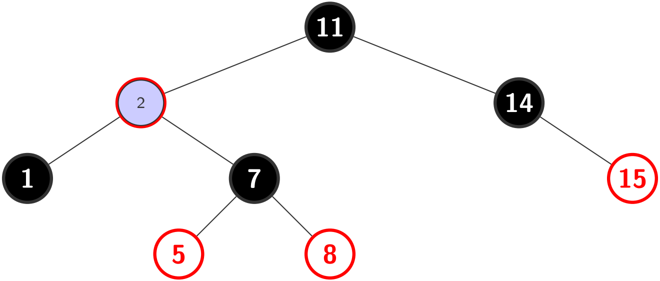
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

Revisando a inserção em árvore binária de busca

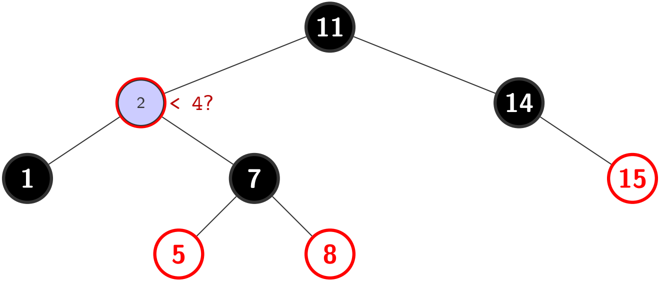
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

Revisando a inserção em árvore binária de busca

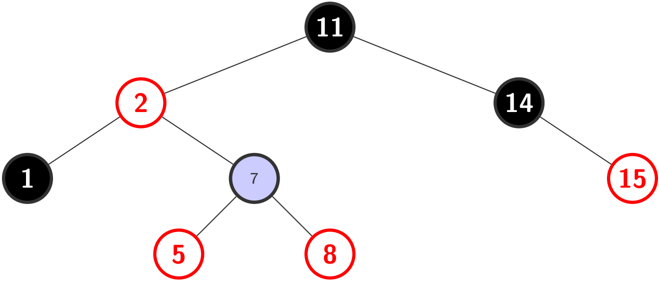
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

Revisando a inserção em árvore binária de busca

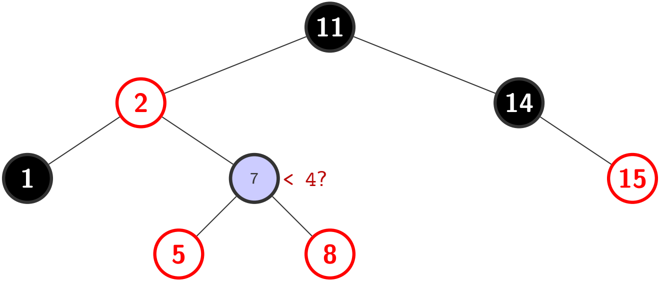
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

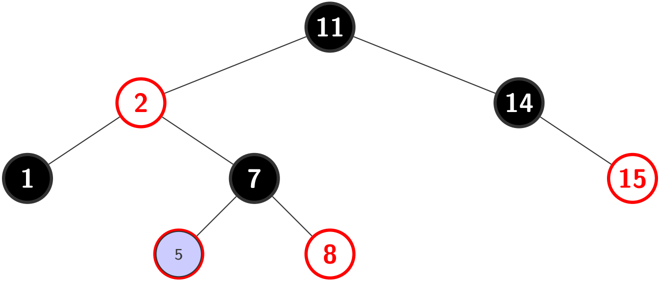
Revisando a inserção em árvore binária de busca
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

Revisando a inserção em árvore binária de busca

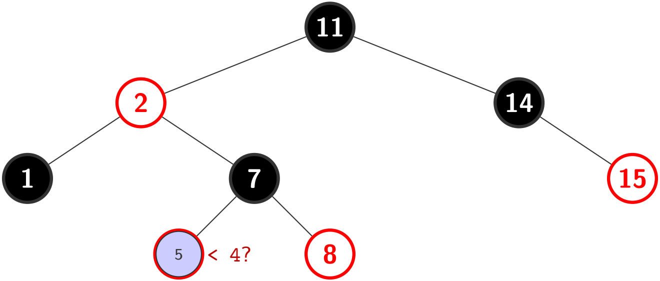
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

Revisando a inserção em árvore binária de busca

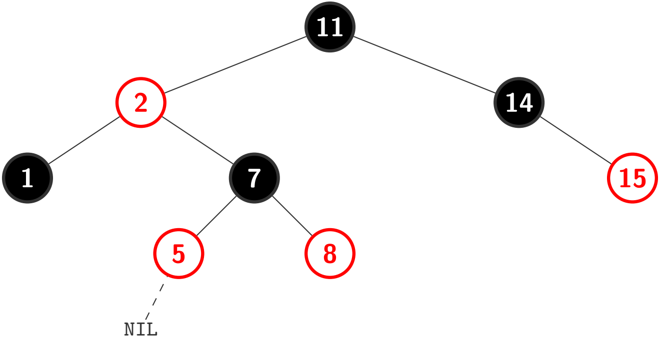
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

Revisando a inserção em árvore binária de busca

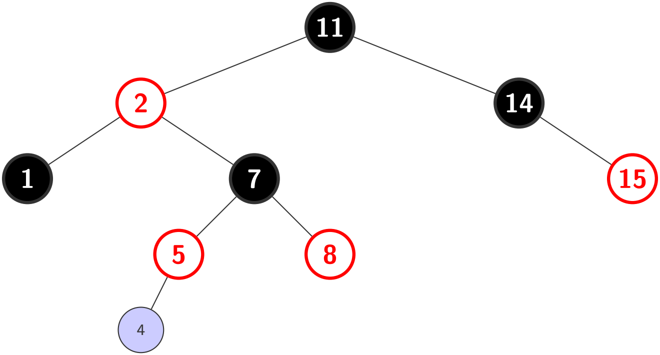
Exemplo: Inserindo a chave 4



Inserções em árvores rubro-negras

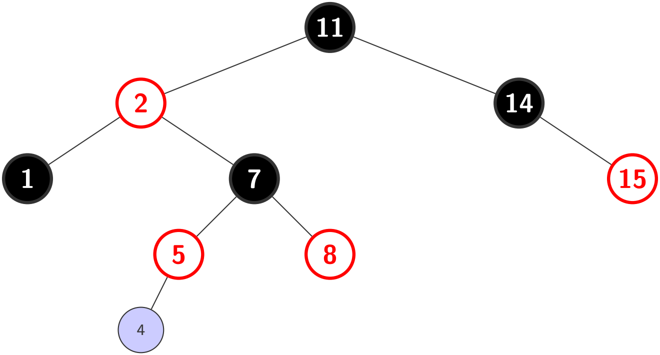
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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



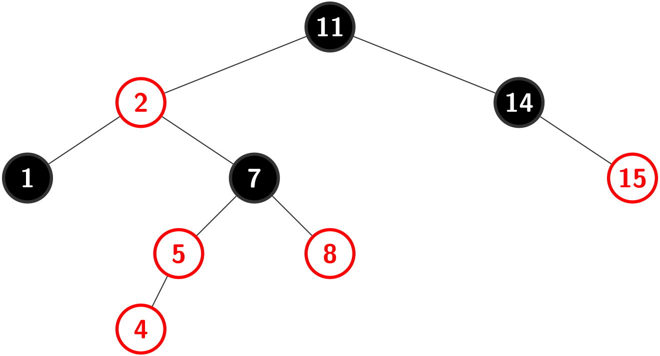
- Quebra P1 — Todo nó deve ser vermelho ou **preto**.

Inserções em árvores rubro-negras



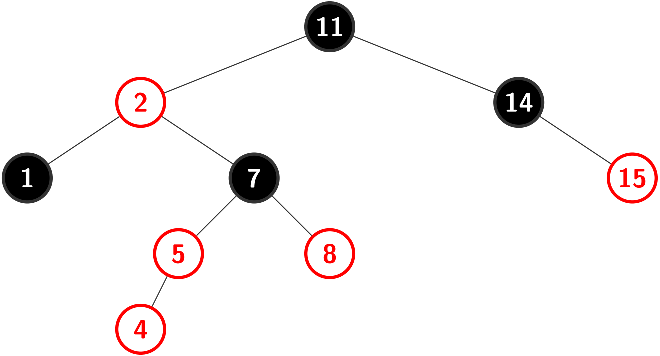
- É preciso decidir, qual faz menos mal, colocar um nó vermelho ou um **preto**?

Inserções em árvores rubro-negras



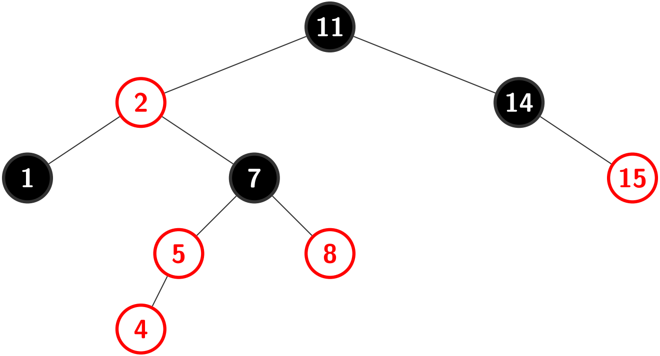
- É preciso decidir, qual faz menos mal, colocar um nó vermelho ou um **preto**?
  - Em algumas situações, o vermelho pode não quebrar nada.
  - O preto sempre vai desequilibrar a altura negra da raiz.

Inserções em árvores rubro-negras



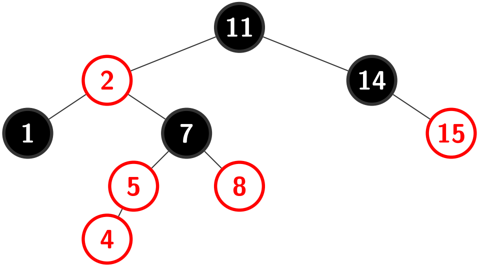
- Propriedade (1) satisfeita, sempre insiro um nó com a cor vermelha.

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**.

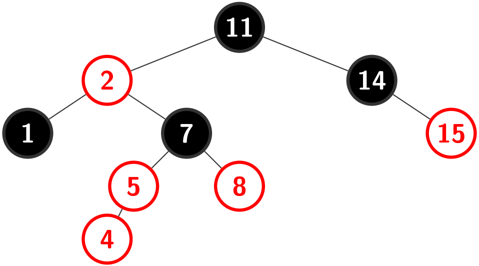
(P2) A raiz é **preta**.

(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



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

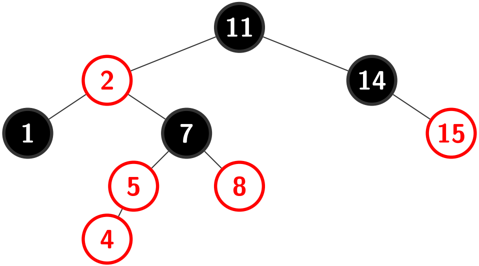
(P2) A raiz é **preta**.

(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**.~~

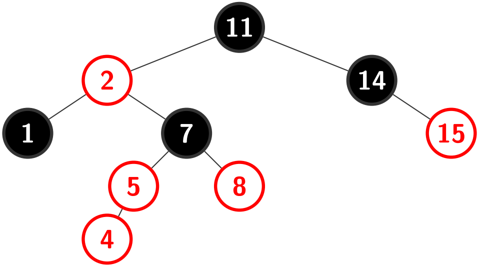
(P2) A raiz é **preta**.

(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



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

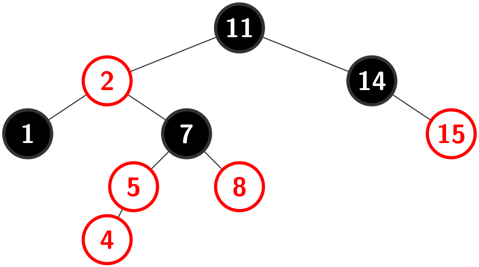
(P2) A raiz é **preta**.

(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



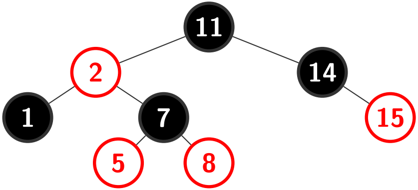
- Sabemos que podemos quebrar apenas as propriedades (P2) e (P4).
- Precisaremos de uma estratégia para recolorir os nós e rebalancear a árvore.
- Porém, já sabemos inserir o nó. Vamos ver o pseudocódigo!

Inserindo um nó – Pseudocódigo

RB-INSERT(T, z)

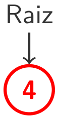
```
y = T.NIL
x = T.root
while x ≠ T.NIL
    y = x
    if z.key < x.key
        x = x.left
    else
        x = x.right
z.p = y
if y == T.NIL
    T.root = z
elseif z.key < y.key
    y.left = z
else
    y.right = z
z.left = T.NIL
z.right = T.NIL
z.color = RED
RB-INSERT-FIXUP(T,z)
```

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.



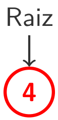
Resolvendo a quebra na Propriedade 2

Propriedade 2: A raiz é **preta**.

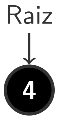


Resolvendo a quebra na Propriedade 2

Propriedade 2: A raiz é **preta**.



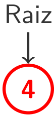
- Quando a raiz for vermelha, basta pintá-la de preto.



- Isto não quebra nenhuma outra propriedade. Por quê?

Resolvendo a quebra na Propriedade 2

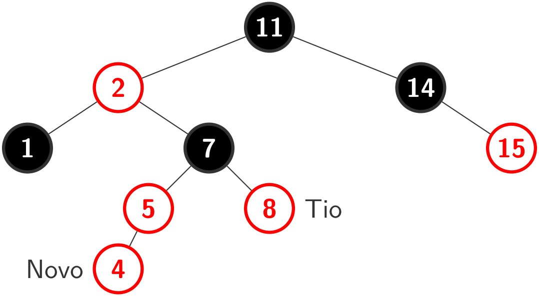
Propriedade 2: A raiz é **preta**.



- 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

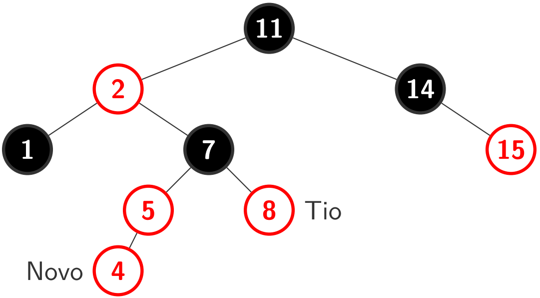
Resolvendo quebra da Propriedade 4



Existem 3 casos que precisamos tratar:

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

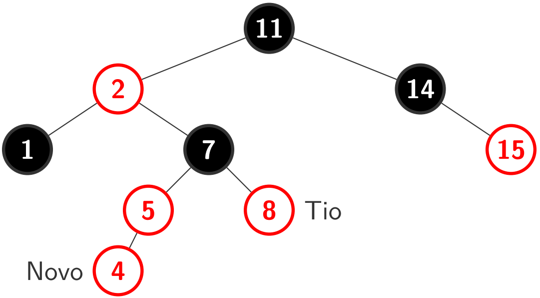
Resolvendo quebra da Propriedade 4



Existem 3 casos que precisamos tratar:

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

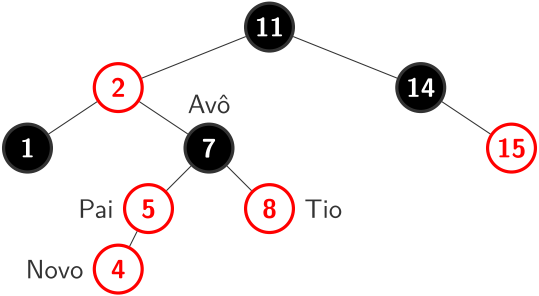
Resolvendo quebra da Propriedade 4



Existem 3 casos que precisamos tratar:

- 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.

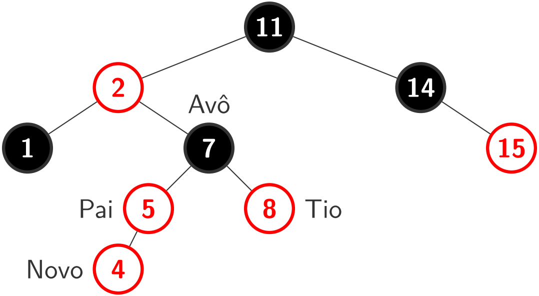
Inserção — Caso 1



Caso 1: O tio do nó Novo é vermelho.

- Se o tio é vermelho o avô obrigatoriamente é preto. Por quê?

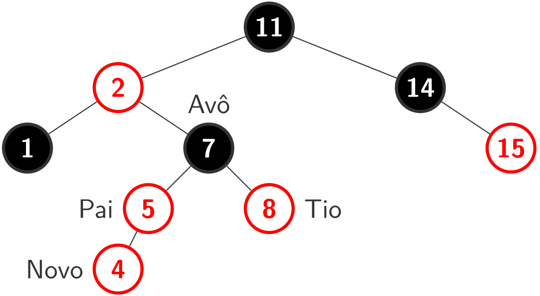
Inserção — Caso 1



Caso 1: O tio do nó Novo é vermelho.

- Se o tio é vermelho o avô obrigatoriamente é preto. Por quê?
- Troque a cor do pai e do tio para **preto**.

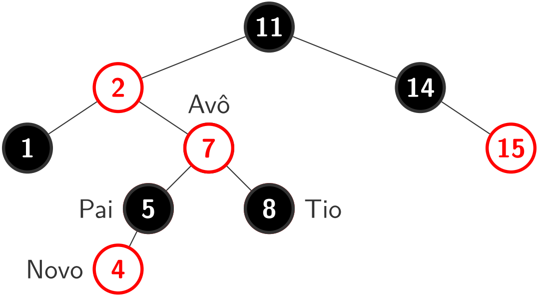
Inserção — Caso 1



Caso 1: O tio do nó Novo é vermelho.

- Se o tio é vermelho o avô obrigatoriamente é preto. Por quê?
- Troque a cor do pai e do tio para **preto**.
- Troque a cor do avô para vermelho.

Inserção — Caso 1

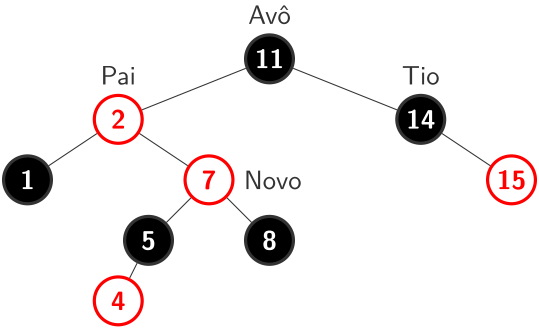


Caso 1: O tio do nó Novo é vermelho.

- Se o tio é vermelho o avô obrigatoriamente é preto. Por quê?
- 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

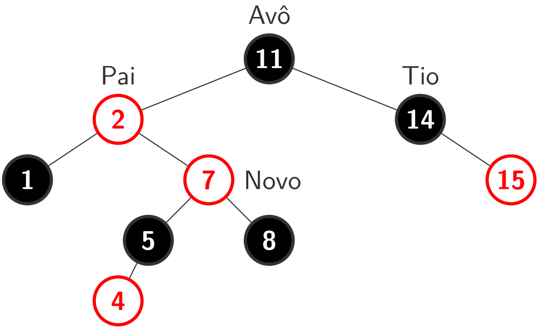
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



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

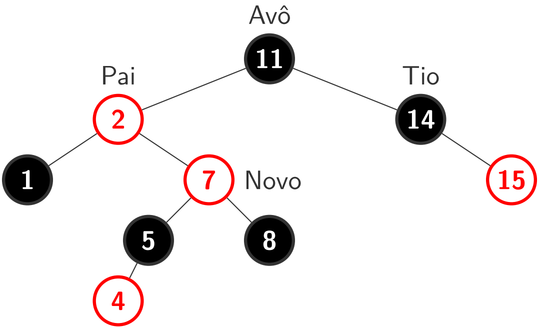


Continuamos o conserto, agora tomando como referência o antigo avô, o nó com o valor 7.

- ~~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

Universidade Federal do Ceará
Campus Quixadá

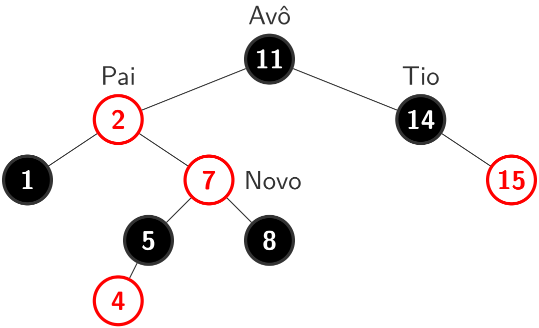


Continuamos o conserto, agora tomando como referência o antigo avô, o nó com o valor 7.

- ~~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

Universidade Federal do Ceará
Campus Quixadá

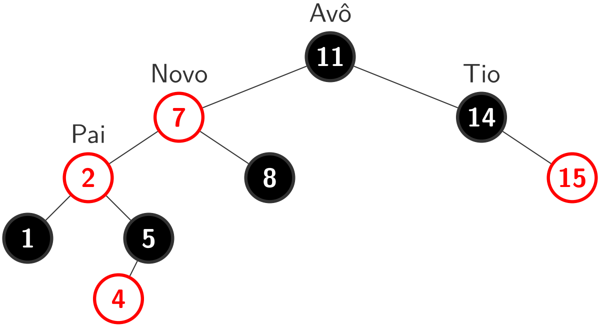


Caso 2: O tio de Novo é **preto** e Novo é filho da direita.

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

Inserção — Caso 2

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

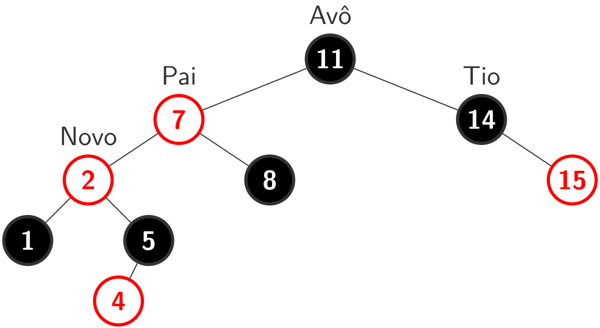


Caso 2: O tio de Novo é **preto** e Novo é filho da direita.

- 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

Universidade Federal do Ceará
Campus Quixadá

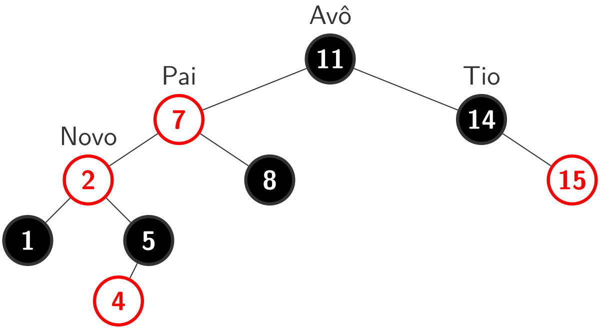


Continuamos o conserto, agora tomando como referência o antigo pai, o nó com a chave 2 na figura acima.

- 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

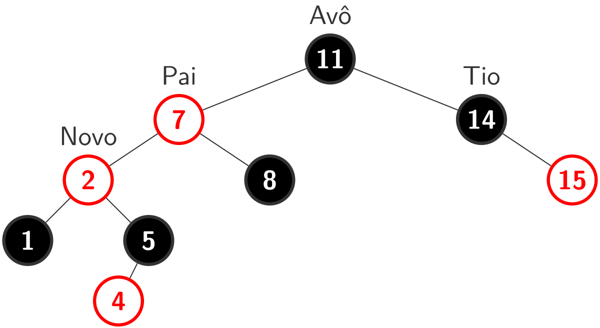
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



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.

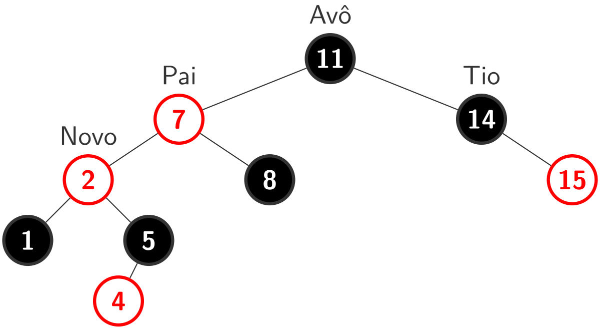
Inserções em árvores rubro-negras



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.

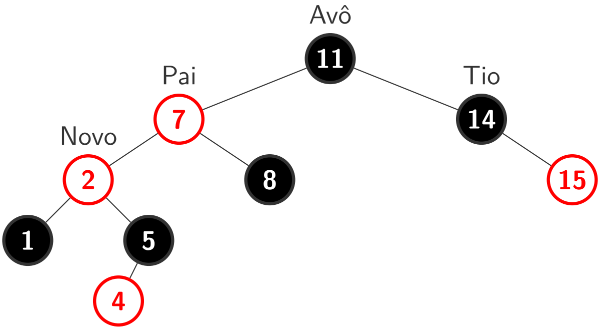
Inserção — Caso 3



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

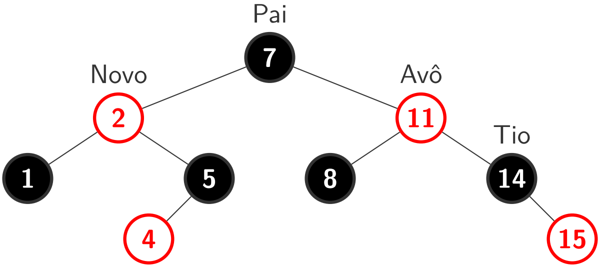


Caso 3: O tio de Novo é **preto** e Novo é filho da esquerda.

- 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

Universidade Federal do Ceará
Campus Quixadá



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.

Casos 2 e 3 — O tio do nó inserido é preto

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

Casos 2 e 3 — O tio do nó inserido é preto

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- 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)

Casos 2 e 3 — O tio do nó inserido é preto

Universidade Federal do Ceará
Campus Quixadá

- 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)

Casos 2 e 3 — O tio do nó inserido é preto

- 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)

Casos 2 e 3 — O tio do nó inserido é preto

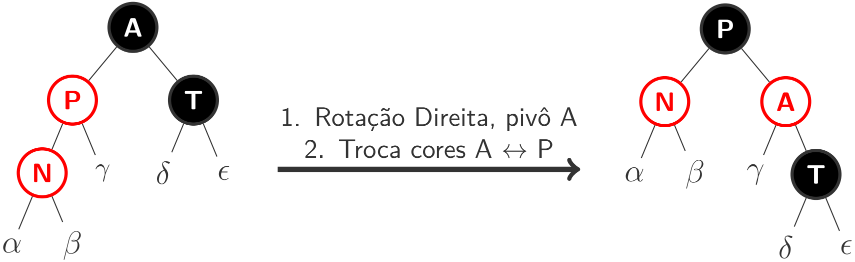
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- 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)

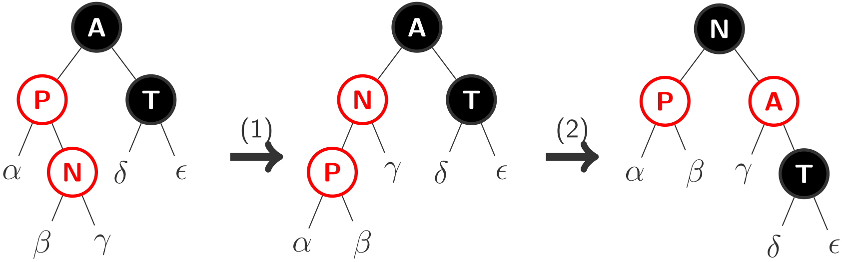
Universidade Federal do Ceará
Campus Quixadá



- A = Avô
- P = Pai
- T = Tio
- N = Novo
- α, β, γ, δ, ϵ são subárvores.

Configuração Esq-Dir (Caso 2a)

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



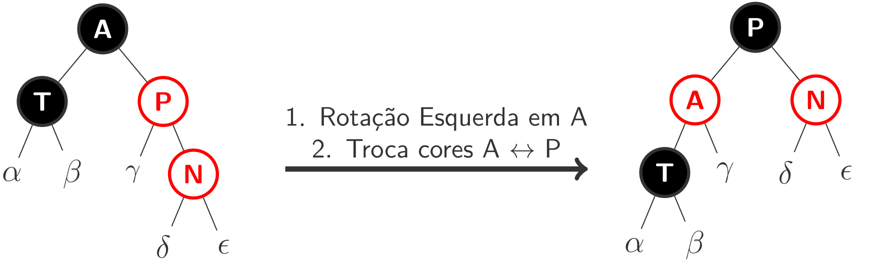
- A = Avô; P = Pai; T = Tio; N = Novo;
- α, β, γ, δ, ϵ são subárvores.

(1) Efetua rotação à esquerda em torno do pai.

(2) Aplica a configuração Esq-Esq.

Configuração Dir-Dir (Caso 3b)

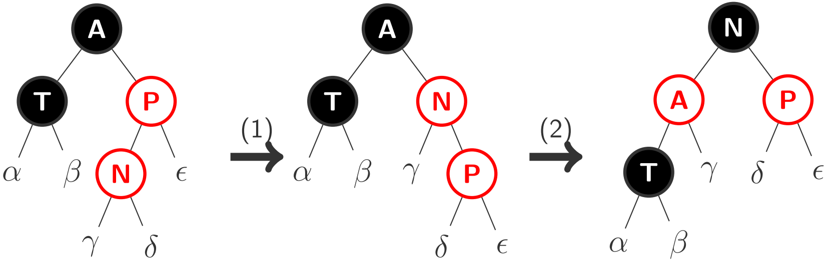
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



- A = Avô
- P = Pai
- T = Tio
- N = Novo
- α, β, γ, δ, ϵ são subárvores.

Configuração Dir-Esq (Caso 2b)

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ



- A = Avô; P = Pai; T = Tio; N = Novo;
- α, β, γ, δ, ϵ são subárvores.

(1) Efetua rotação à direita em torno do pai.

(2) Aplica a configuração Dir-Dir.

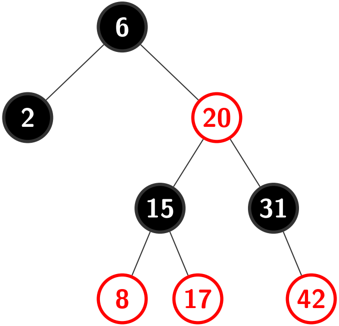
Código (incompleto) do conserto pós-inserção

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

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



Remoção

Remoções em árvores rubro-negras

- 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

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.

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.

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 um duplo nó vermelho.

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 um duplo nó vermelho.
  - Durante a remoção, se retirarmos um nó preto, estamos estragando a propriedade de altura negra da árvore.

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 (se existir) de T de modo que árvore binária resultante continue sendo uma árvore binária de busca.

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 (se existir) de T de modo que árvore binária resultante continue sendo uma árvore binária de busca.
- Isto é mais difícil do que a inserção. Para resolver este problema tratamos a princípio a remoção de uma raiz e depois partimos para a resolução de um problema mais geral.

Revisão — Remoção em árvore binária de busca

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Temos 3 casos:

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

Revisão — Remoção em árvore binária de busca

Universidade Federal do Ceará
Campus Quixadá

Temos 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

Temos 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 nó que sucede a raiz no percurso-em-ordem-simétrica como a nova raiz.

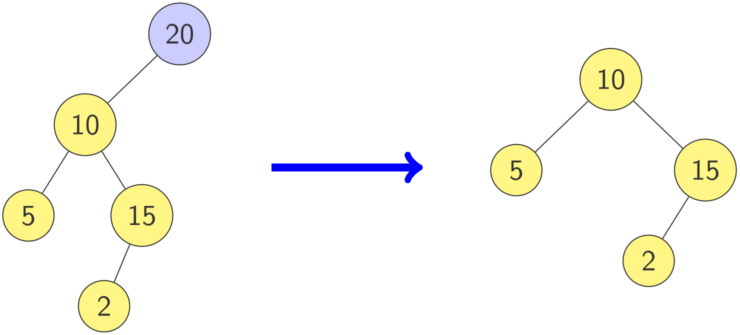
Revisão — Remoção em árvore binária de busca

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

Revisão — Remoção em árvore binária de busca

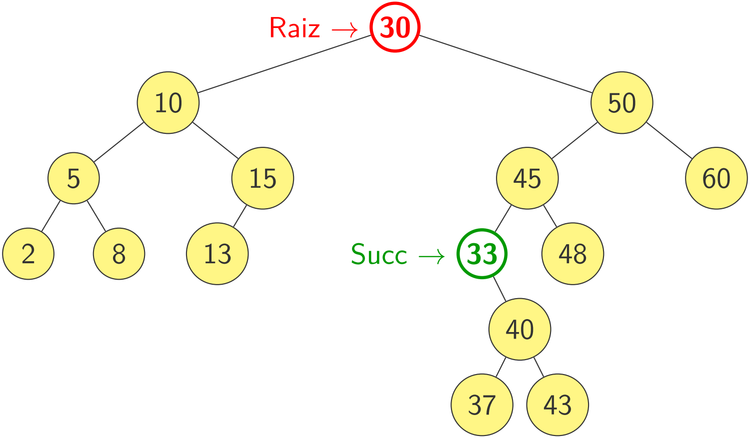
- Caso 2: a raiz tem um único filho. Seu filho torna-se a nova raiz.



Revisão — Remoção em árvore binária de busca

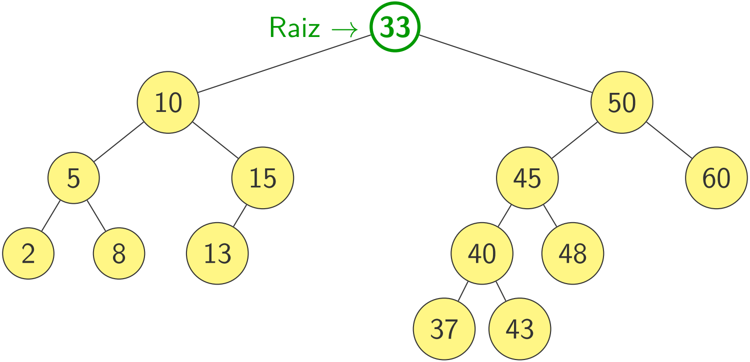
Universidade Federal do Ceará
Campus Quixadá

- Caso 3: a raiz tem dois filhos. Neste caso, tomamos o nó que sucede a raiz no percurso inordem como a nova raiz.



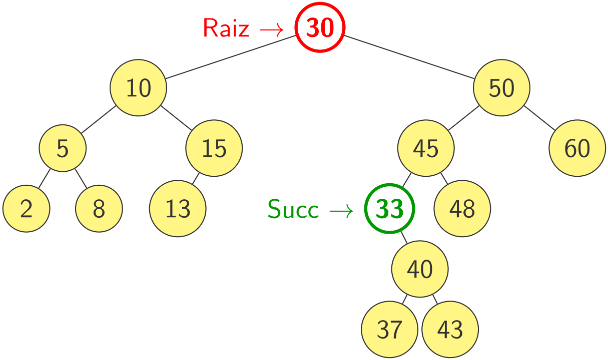
Revisão — Remoção em árvore binária de busca

- Caso 3: a raiz tem dois filhos. Neste caso, tomamos o nó que sucede a raiz no percurso inordem como a nova raiz.



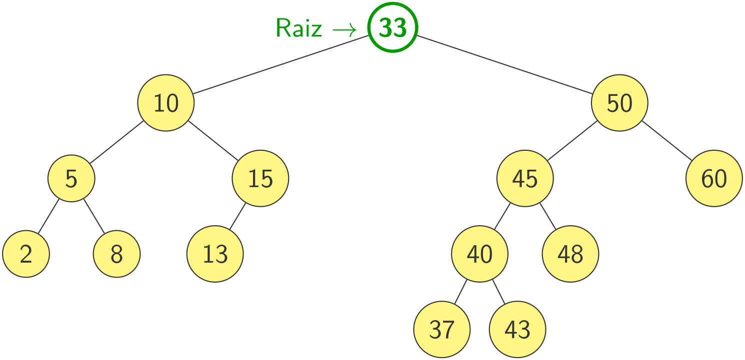
Revisão — Remoção em árvore binária de busca

- Outra maneira de interpretar o Caso 3 é a seguinte: copiamos a informação do nó sucessor para a raiz e removemos o nó copiado (ou seja, reduzimos ao Caso 1 ou Caso 2).



Revisão — Remoção em árvore binária de busca

- Outra maneira de interpretar o Caso 3 é a seguinte: copiamos a informação do nó sucessor para a raiz e removemos o nó copiado (ou seja, reduzimos ao Caso 1 ou Caso 2).



Remoção em árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Passos:

Remoção em árvores rubro-negras

- Passos:
  - Encontre o nó v a ser removido

Remoção em árvores rubro-negras

Universidade Federal do Ceará
Campus Quixadá

- Passos:
  - Encontre o nó v a ser removido
  - Remova o nó v da árvore (use o algoritmo que acabamos de rever)

Remoção em árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Passos:
  - Encontre o nó v a ser removido
  - Remova o nó v da árvore (use o algoritmo que acabamos de rever)
  - Conserte as propriedades da árvore rubro-negra.

Remoção em árvores rubro-negras

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Passos:
  - Encontre o nó v a ser removido
  - Remova o nó v da árvore (use o algoritmo que acabamos de rever)
  - Conserte as propriedades da árvore rubro-negra.

- Assim como as outras operações básicas em uma árvore rubro-negra com n nós, a remoção também tem complexidade de $O(\lg n)$.

Remoção em árvores rubro-negras — Problemas

Que problemas podemos criar quando copiamos o valor do sucessor para a posição do nó a ser removido e removemos o sucessor (original/copiado)?

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

Remoção em árvores rubro-negras — Problemas

Que problemas podemos criar quando copiamos o valor do sucessor para a posição do nó a ser removido e removemos o sucessor (original/copiado)?

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

Remoção em árvores rubro-negras — Problemas

Que problemas podemos criar quando copiamos o valor do sucessor para a posição do nó a ser removido e removemos o sucessor (original/copiado)?

- 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 quando copiamos o valor do sucessor para a posição do nó a ser removido e removemos o sucessor (original/copiado)?

- 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

Universidade Federal do Ceará
Campus Quixadá

Que problemas podemos criar quando copiamos o valor do sucessor para a posição do nó a ser removido e removemos o sucessor (original/copiado)?

- 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

Se o nó y a ser removido for **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

Se o nó y a ser removido for **preto**:

- Se y era raiz e um filho vermelho de y se torna raiz quebramos a Propriedade 2.

(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

Universidade Federal do Ceará
Campus Quixadá

Se o nó `y` a ser removido for **preto**:

- Se `y` era raiz e um filho vermelho de `y` se torna raiz quebramos a Propriedade 2.
- Se tanto `suc(y)->right` quanto `suc(y)->parent` eram vermelhos então, agora, violamos a Propriedade 4.

(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

Se o nó `y` a ser removido for **preto**:

- Se `y` era raiz e um filho vermelho de `y` se torna raiz quebramos a Propriedade 2.
- Se tanto `suc(y)->right` quanto `suc(y)->parent` eram vermelhos então, agora, violamos a Propriedade 4.
- A remoção de `suc(y)` faz com que qualquer caminho que continha `suc(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

Universidade Federal do Ceará
Campus Quixadá

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

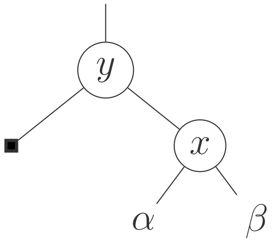
- Na remoção padrão, sempre excluímos um nó que é folha ou que tem apenas um filho (para um nó que tem dois filhos, copiamos a chave do seu sucessor nele e depois removemos o sucessor).

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`.

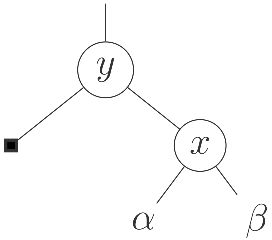


Remoção — Primeiro passo do algoritmo

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

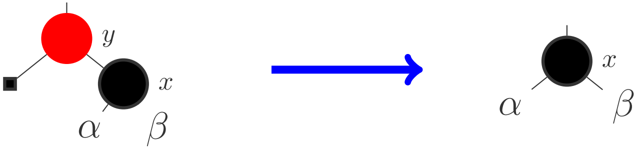
- Na remoção padrão, sempre excluímos um nó que é folha ou que tem apenas um filho (para um nó que tem dois filhos, copiamos a chave do seu sucessor nele e depois removemos o sucessor).

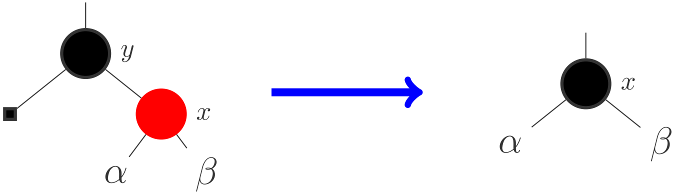
- Seja `y` o nó a ser excluído e seja `x` o filho que substitui `y`.
  - Note que `x` é `nil` quando `y` é uma folha e a cor de `nil` é **preta**.



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

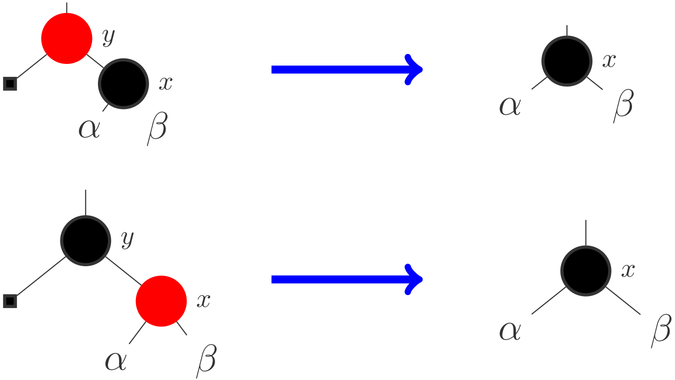
- Se `y` ou `x` forem vermelhos, fazemos `x->color = BLACK` (não há alteração em nenhuma altura negra).





Análise das cores de y e x — Caso Fácil

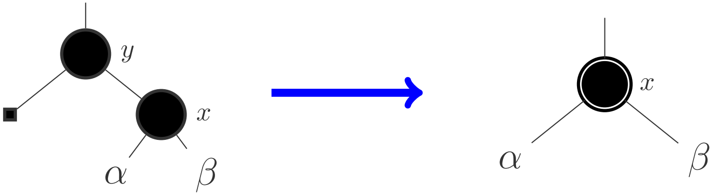
- Se y ou x forem vermelhos, fazemos `x->color = BLACK` (não há alteração em nenhuma altura negra).



- y e x não podem ser ambos vermelhos, pois pai e filho vermelhos não são permitidos na árvore rubro-negra.

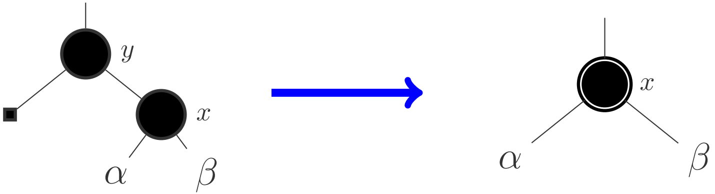
Análise das cores de y e x — Caso Ruim

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



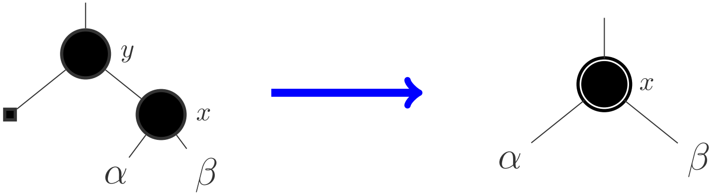
Análise das cores de `y` e `x` — Caso Ruim

- 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".



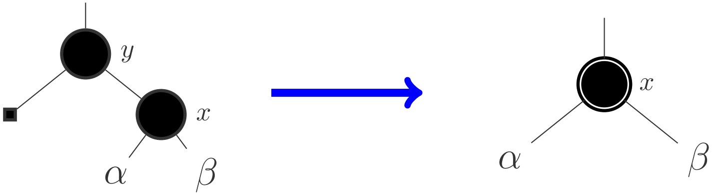
Análise das cores de y e x — Caso Ruim

- 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

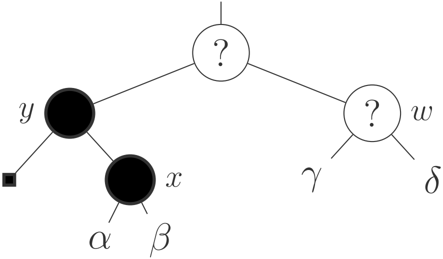
- O verdadeiro problema na remoção ocorre quando ambos `y` e `x` são **pretos**.
- Neste caso, ao remover o nó `y`, as alturas negras de todos os seus ancestrais ficam "corrompidas".
- A fim de consertar este caso problemático, usamos a noção de **preto duplo**.
  - Quando um nó preto é excluído e substituído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

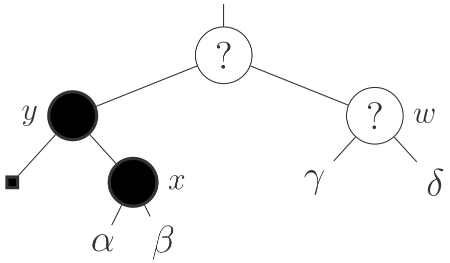
- `z` — O nó cuja chave será removida.



Segundo passo: Eliminação do preto duplo

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

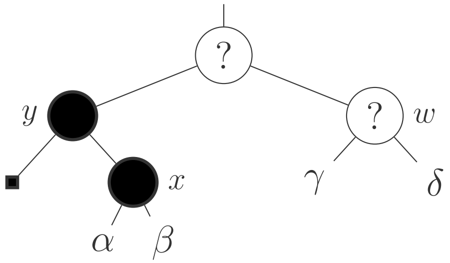
- `z` — O nó cuja chave será removida.
- `y` — O nó que será fisicamente removido. Onde `y = z` se `z` possui 0 ou 1 filho. Caso contrário, `y = sucessor(z)`.



Segundo passo: Eliminação do preto duplo

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

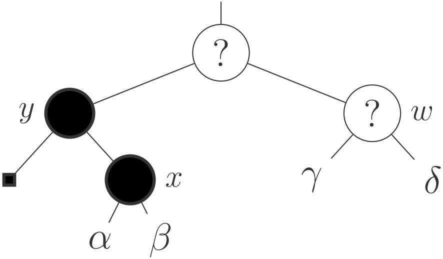
- `z` — O nó cuja chave será removida.
- `y` — O nó que será fisicamente removido. Onde `y = z` se `z` possui 0 ou 1 filho. Caso contrário, `y = sucessor(z)`.
- `x` — O único filho de `y` antes de qualquer modificação, ou `NIL` caso `y` não tenha filhos.



Segundo passo: Eliminação do preto duplo

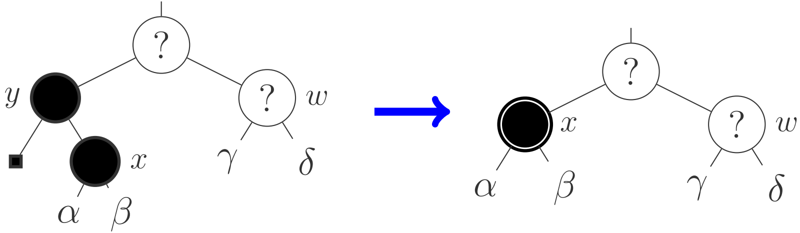
Usamos a seguinte nomenclatura para os nós envolvidos no processo de remoção:

- `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`.



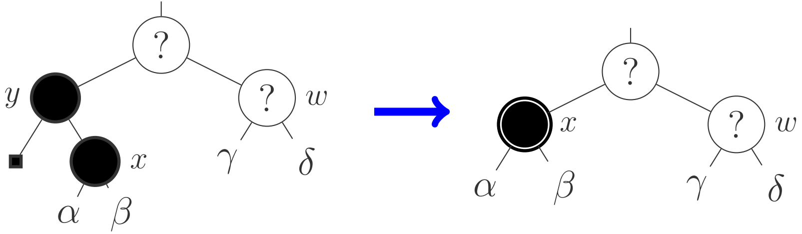
Segundo passo: Eliminação do preto duplo

- Durante o conserto da árvore, quando considerarmos o nó `x` vamos contá-lo como preto duas vezes (**duplo preto**)



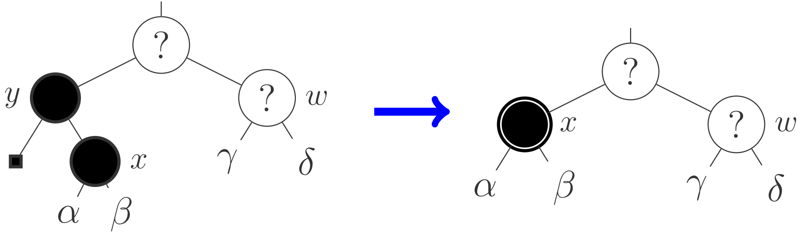
Segundo passo: Eliminação do preto duplo

- Durante o conserto da árvore, quando considerarmos o nó `x` vamos contá-lo como preto duas vezes (**duplo preto**)
- A ideia é que o `x` possa receber sempre a contagem de preto do pai para manter a Propriedade 5.



Segundo passo: Eliminação do preto duplo

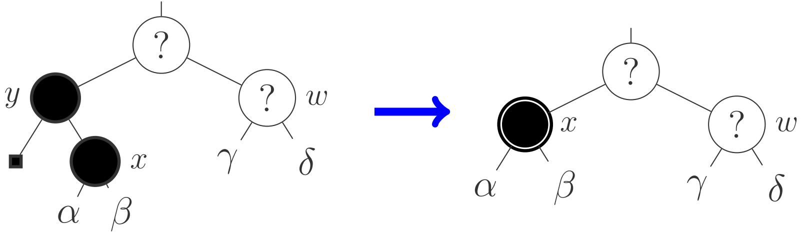
- Durante o conserto da árvore, quando considerarmos o nó `x` vamos contá-lo como preto duas vezes (**duplo preto**)
- A ideia é que o `x` possa receber sempre a contagem de preto do pai para manter a Propriedade 5.
- `x` aponta para um **duplo preto**, o que nos diz que o nó `w` existe (Por quê?)



Segundo passo: Eliminação do preto duplo

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

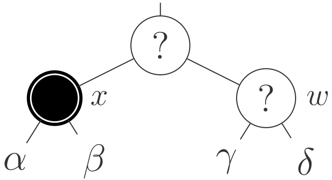
- Durante o conserto da árvore, quando considerarmos o nó `x` vamos contá-lo como preto duas vezes (**duplo preto**)
- A ideia é que o `x` possa receber sempre a contagem de preto do pai para manter a Propriedade 5.
- `x` aponta para um **duplo preto**, o que nos diz que o nó `w` existe (Por quê?)
- No caso em que `x` era inicialmente vermelho (ainda assim o contamos como preto no cálculo da altura negra), basta pintar `x` de **preto** para finalizar o conserto da árvore.



Eliminação do preto duplo — 4 Casos

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

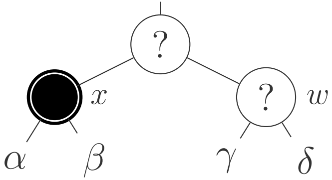
- Caso 1: `w` é vermelho.



Eliminação do preto duplo — 4 Casos

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

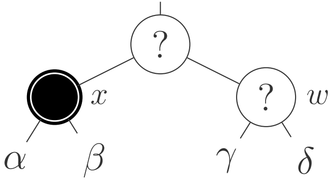
- Caso 1: `w` é vermelho.
- Caso 2: `w` é **preto** e seus dois filhos são **pretos**.



Eliminação do preto duplo — 4 Casos

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

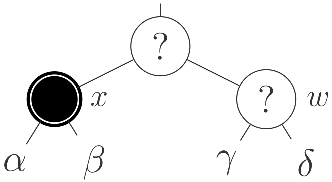
- Caso 1: `w` é vermelho.
- Caso 2: `w` é **preto** e seus dois filhos são **pretos**.
- Caso 3: `w` é **preto**, seu filho esquerdo é vermelho e seu filho direito é **preto**.



Eliminação do preto duplo — 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.

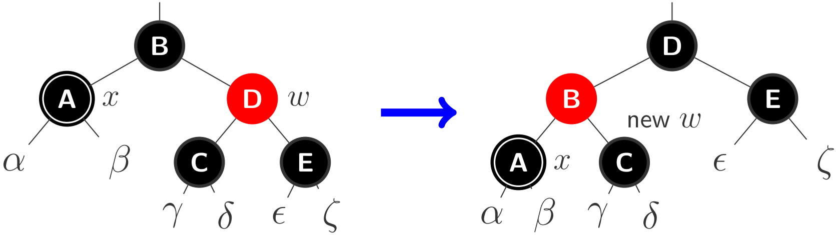


Caso 1: o nó w é vermelho

Universidade Federal do Ceará
Campus Quixadá

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

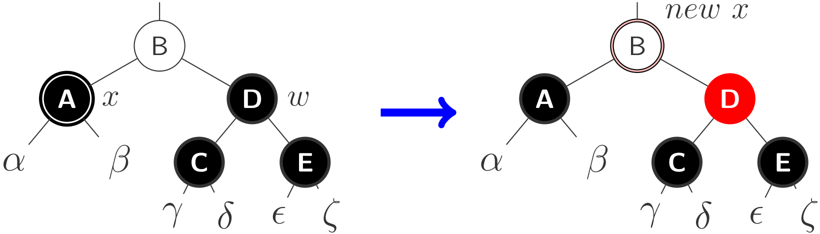
- Trocamos as cores de `w` e `x.parent`.
- Efetuamos uma rotação à esquerda tendo como pivô `x.parent`.
- Essas alterações não violam nenhuma propriedade da árvore rubro-negra.
- Mas transformam o Caso 1 no Caso 2, Caso 3 ou Caso 4.



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

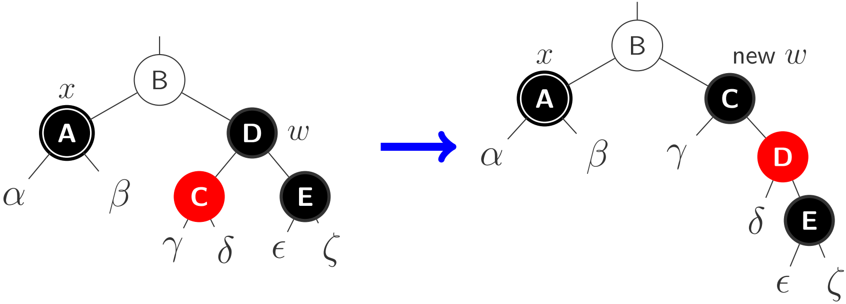
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- 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.
- Jogamos a bomba para cima, tratando o `x.parent` como sendo o novo `x`.



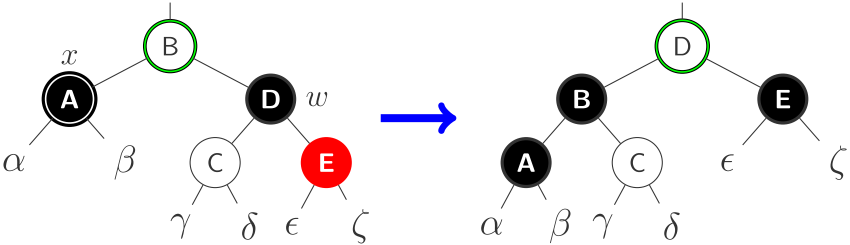
Caso 3: w e w.right são pretos, w.left é vermelho

- 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 da Remoção

Função-membro pública `Remove`

Universidade Federal do Ceará
Campus Quixadá

Esta função recebe como argumentos a árvore T e a chave k a ser removida.

```
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 privada `RB-delete`

RB-DELETE(T,z)

```
if z.left == T.NIL or z.right == T.NIL then
    y = z
else
    y = MINIMUM(z.right)        // pega o sucessor de z
end if
if y.left != T.NIL then x = y.left
else x = y.right
x.p = y.p        // ajusta o pai do x
if y.p == T.NIL then
    T.root = x
else
    if y == y.p.left then y.p.left = x
    else y.p.right = x
end if
if y != z then
    z.key = y.key
    z.value = y.value
end if
if y.color == BLACK then
    RB-Delete-Fixup(T, x)
end if
delete y
```

Função-membro privada `RB-Delete-FixUp`

Universidade Federal do Ceará
Campus Quixadá

RB-DELETE-FIXUP(T, x)

```
while x ≠ T.root and x.color == BLACK
    if x == x.p.left
        w = x.p.right
        if w.color == RED
            w.color = BLACK                                   // case 1
            x.p.color = RED                                   // case 1
            LEFT-ROTATE(T, x.p)                               // case 1
            w = x.p.right                                     // case 1
        if w.left.color == BLACK and w.right.color == BLACK
            w.color = RED                                     // case 2
            x = x.p                                           // case 2
        else if w.right.color == BLACK
                w.left.color = BLACK                          // case 3
                w.color = RED                                 // case 3
                RIGHT-ROTATE(T, w)                            // case 3
                w = x.p.right                                 // case 3
            w.color = x.p.color                               // case 4
            x.p.color = BLACK                                 // case 4
            w.right.color = BLACK                             // case 4
            LEFT-ROTATE(T, x.p)                               // case 4
            x = T.root                                        // case 4
    else (same as then clause with "right" and "left" exchanged)
x.color = BLACK
```

AVL versus Rubro-Negra

AVL $\times$ Rubro-negra

Universidade Federal do Ceará
Campus Quixadá

- 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

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.

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 $\times$ 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

Universidade Federal do Ceará
Campus Quixadá

- 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 × 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.

AVL $\times$ Rubro-negra

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Árvores Rubro-Negras são de uso mais geral do que as árvores AVL.
- São utilizadas em diversas aplicações e bibliotecas de linguagens de programação.
- Exemplos:
  - **Java:** java.util.TreeMap, java.util.TreeSet
  - **C++ STL:** map, multiset
  - **Linux kernel:** completely fair scheduler, linux/rbtree.h

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