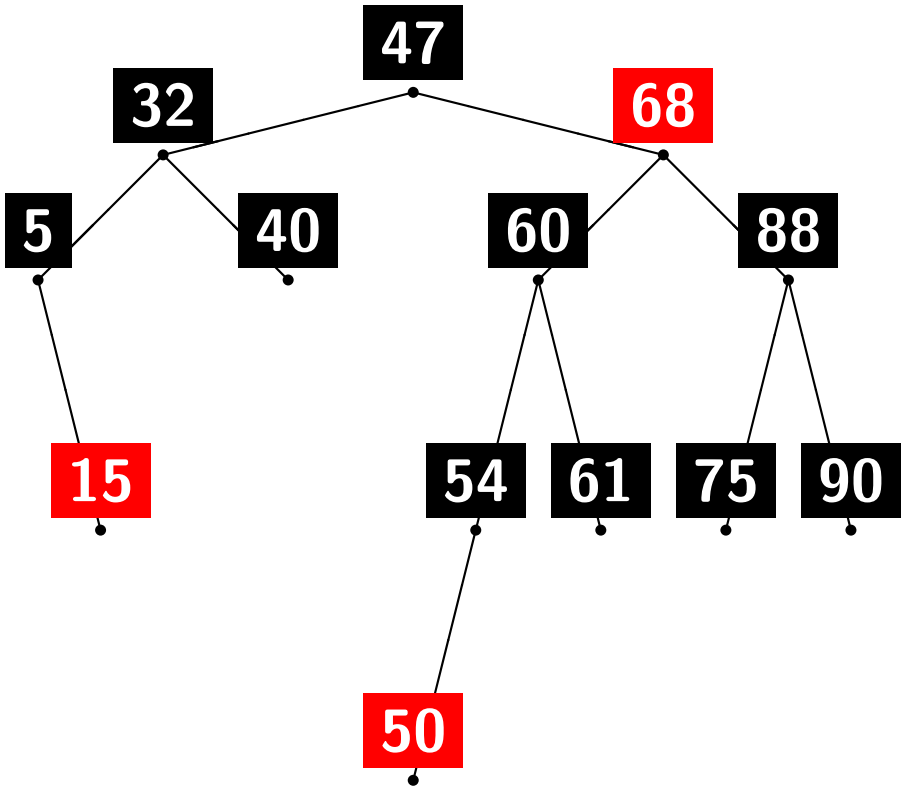
Aula 18: Árvores rubro-negras

David Déharbe
Programa de Pós-graduação em Sistemas e Computação
Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada

Download me from `http://DavidDeharbe.github.io`.

PPgSC

Plano da aula



Introdução

Propriedades

Operações

PPgSC

Árvores rubro-negra

Introdução

- Em 1972, Rudolf Bayer projetou a estrutura de dados árvores B binárias simétricas.
- Popularizada sob o nome árvores rubro-negras (Guibas & Sedgewick 1978)
- Complexidade
  - busca em $O(\lg n)$,
  - remoção em $\Theta(\lg n)$ e
  - inserção em $\Theta(\lg n)$
- Menos restritivas que árvores AVL, com alturas tendendo a ser maiores
  - busca: menos eficiente
  - inserção e remoção: mais eficiente
- bastante usadas na prática (escalonador Linux, STL)

PPgSC

Árvores rubro-negra

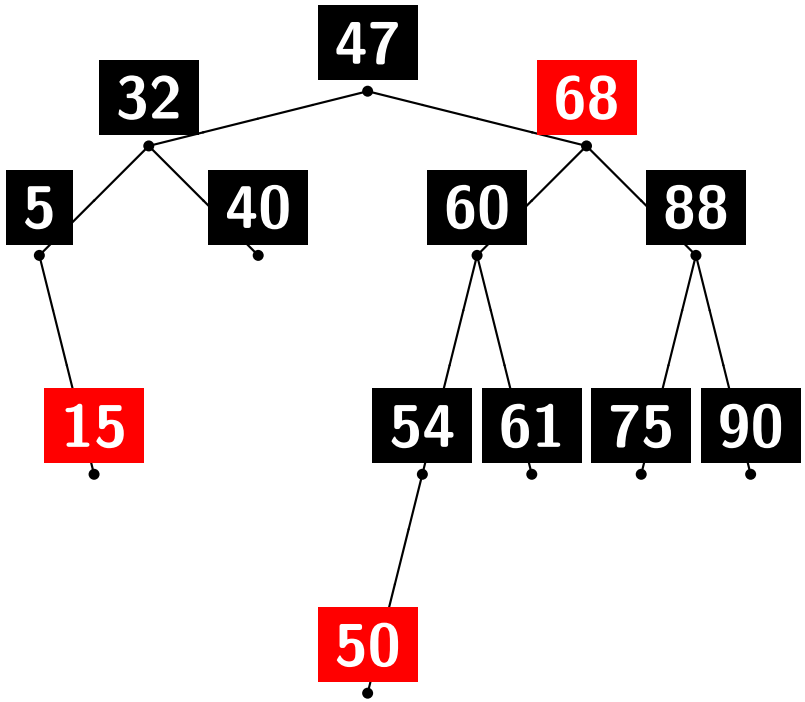
Especificação

1. árvore binária de busca
2. os nós tem um atributo $color \in \{\textsc{Red}, \textsc{Black}\}$.
   Convenção: $\textsc{Nil}.color = \textsc{Black}$
3. A raiz é negra $root.color = \textsc{Black}$
4. Qualquer caminho da raiz até uma sub-árvore vazia tem o mesmo número de nós negros.

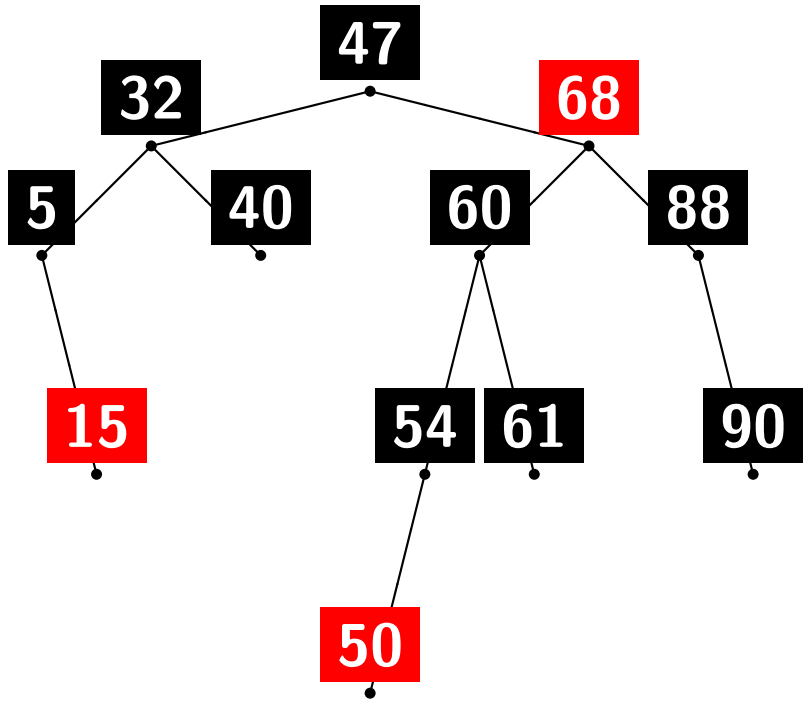
$$\begin{aligned} bh(\textsc{Nil}) &= 0 \\ bh(x) &= 1 + bh(x.left) \text{ se } x.left.color = \textsc{Black} \\ bh(x) &= bh(x.left) \text{ se } x.left.color = \textsc{Red} \end{aligned}$$

5. Os descendentes de um nó rubro são negros: $x.color = \textsc{Red} \Rightarrow x.left.color = \textsc{Black} \wedge x.right.color = \textsc{Black}$.

Ilustração



rubro-negra

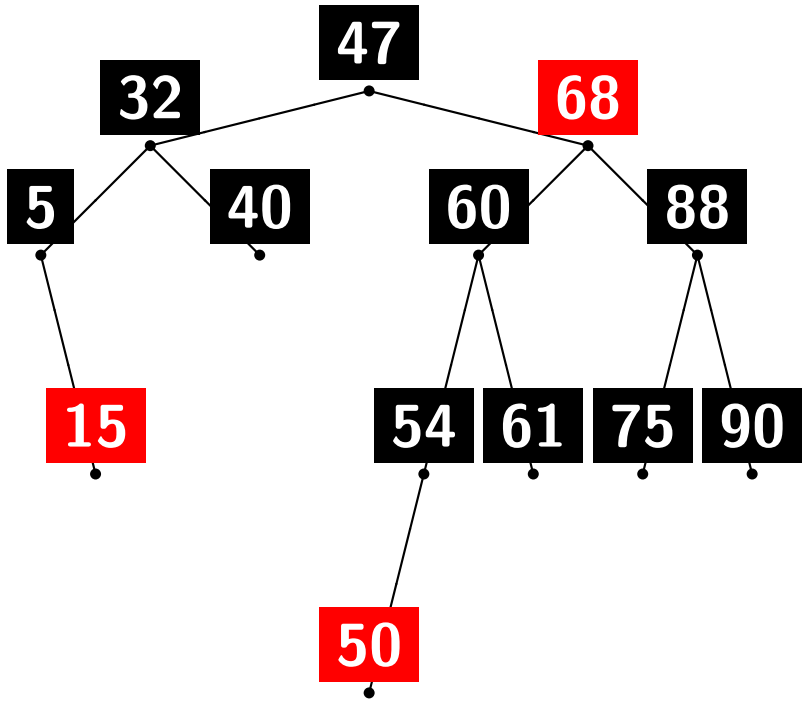


não rubro-negra

PPgSC

Exercício

Existe outras maneiras de colorir os nós da seguinte árvore de tal forma que respeite as propriedades de árvores rubro-negras?



Altura de árvores rubro-negras

Teorema

Seja T uma árvore rubro-negra, com n nós, então $\alpha(T.root) \in \Theta(\lg n)$.

Demonstração.

- seja $p = bh(T)$ (a altura negra de T)
- há pelo menos $2^p - 1$ nós negros, logo $n \geq 2^p - 1$
- a altura máxima α_m de T é $2p$ (alternando nós rubros e negros)
- $\alpha_m \leq 2p$ e $n \geq 2^p - 1$, logo $\alpha_m \leq 2\log_2(n+1)$.
- em uma árvore binária $\alpha_m \in \Omega(\log n)$
- logo $\alpha_m \in \Theta(\log n)$.

PPgSC

Implementação

- $x.color \in \{\text{RED}, \text{BLACK}\}$

Operação de busca

- A busca não modifica a árvore.
- O algoritmo de busca em árvores rubro-negra é o mesmo da busca em árvores binárias de busca qualquer.
- a complexidade é $O(\log n)$

PPgSC

Operação de inserção

1. inserção em árvore binária de busca
2. se necessário, re-balanceamento

Propriedade 2: qual cor escolher para o novo nó?

- se for negro, e a árvore inicialmente não vazia, quebra a propriedade 4
- se for rubro, e a árvore inicialmente vazia, quebra a propriedade 3

Propriedade 2: qual cor escolher para o novo nó?

- se for negro, e a árvore inicialmente não vazia, quebra a propriedade 4
- se for rubro, e a árvore inicialmente vazia, quebra a propriedade 3

solução

- é criado um nó, q
- a cor defaut é rubro: $q.color = \textsc{Red}$
- após a inserção, a raiz é atribuída a cor negro $root.color = \textsc{Black}$

Re-balanceamento

Inserção

Propriedade 5: nó rubro não pode ter descendente direto rubro

- seja v o nó ascendente de q
- se v for negro, não ha desequilibro
- se v for rubro, a propriedade 5 é violada

PPgSC

Propriedade 5: nó rubro não pode ter descendente direto rubro

- seja v o nó ascendente de q
- se v for negro, não ha desequilibro
- se v for rubro, a propriedade 5 é violada

solução

- como v é rubro, tem um nó ascendente, w, tal que $w.color == \text{BLACK}$
- seja t a outra sub-árvore de w
  1. $t.color == \text{RED}$
  2. $t.color == \text{BLACK}$

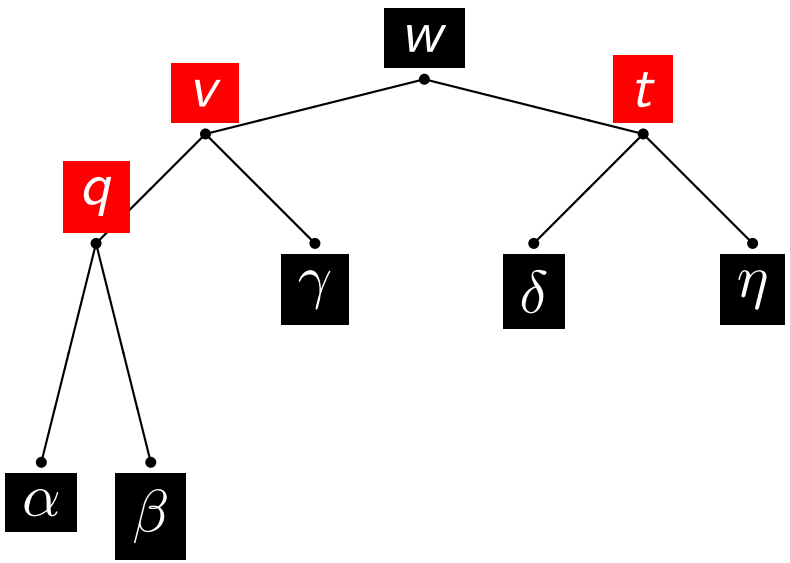
```
INSERT(T, k)
1   if T.root == NIL
2         T.root = MAKE-NODE(k)
3   else v = NIL
4         w = NIL
5         INSERT-AUX(T.root, v, w, k)
6   T.root.color = BLACK
```

O caso $t.color == \text{RED}$

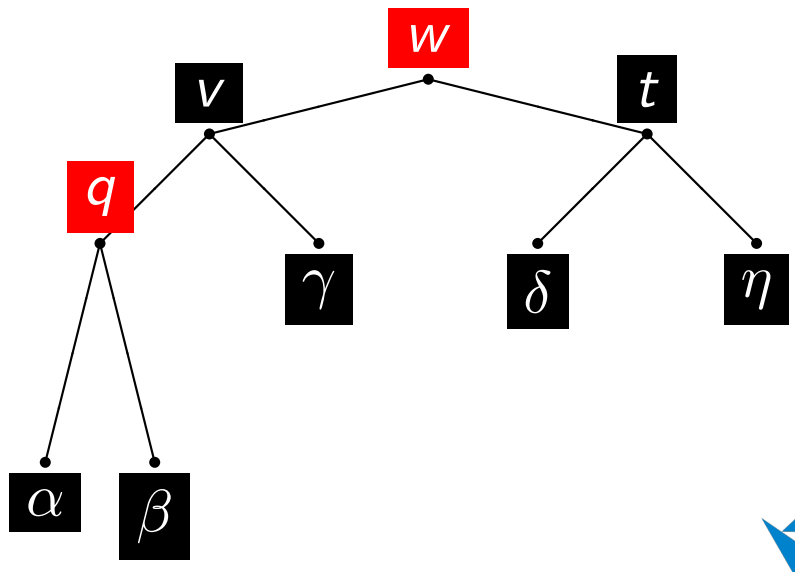
$t.color == \text{RED}$

- logo $t \neq \text{NIL}$
- inversão das cores de v, w e t.

(Antes)



(Depois)

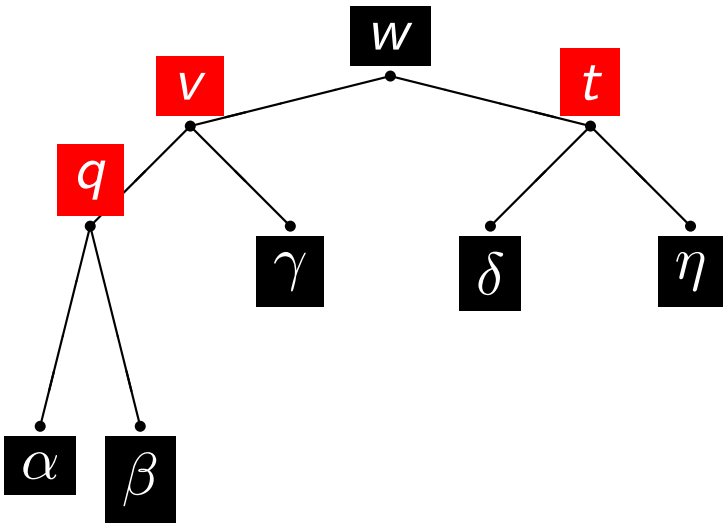


PPgSC

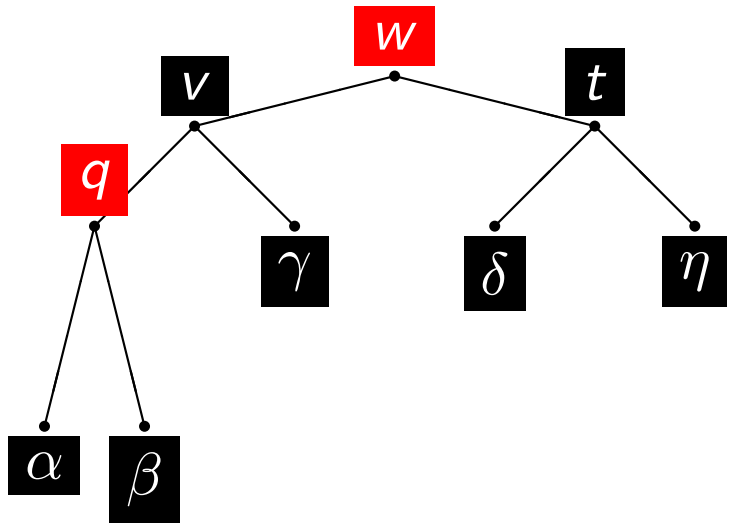
O caso $t.color == \text{RED}$

$t.color == \text{RED}$

(Antes)



(Depois)



1. se w era a raiz, é atribuído a cor negro
2. se w não era a raiz, opera-se eventual rebalanceamento em nível superior
3. a operação não alterou $bh(w)$

PPgSC

O caso $t.color == \text{BLACK}$

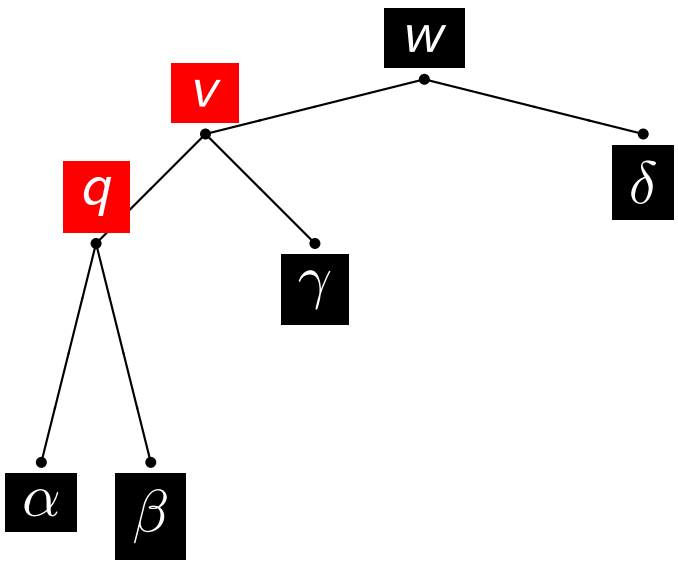
$t.color == \text{BLACK}$

1. $w.left == v$ e $v.left == q$
2. $w.left == v$ e $v.right == q$
3. $w.right == v$ e $v.right == q$ (simétrico de 1)
4. $w.right == v$ e $v.left == q$ (simétrico de 2)

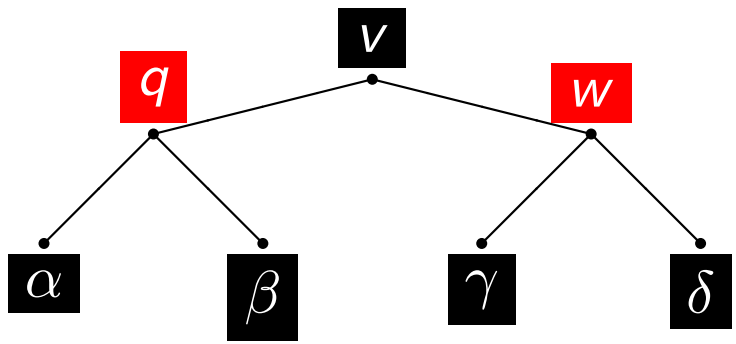
O caso $t.color == \text{BLACK}$ e $w.left == v$ e $v.left == q$

$t.color == \text{BLACK}$ e $w.left == v$ e $v.left == q$

(Antes)



(Depois)



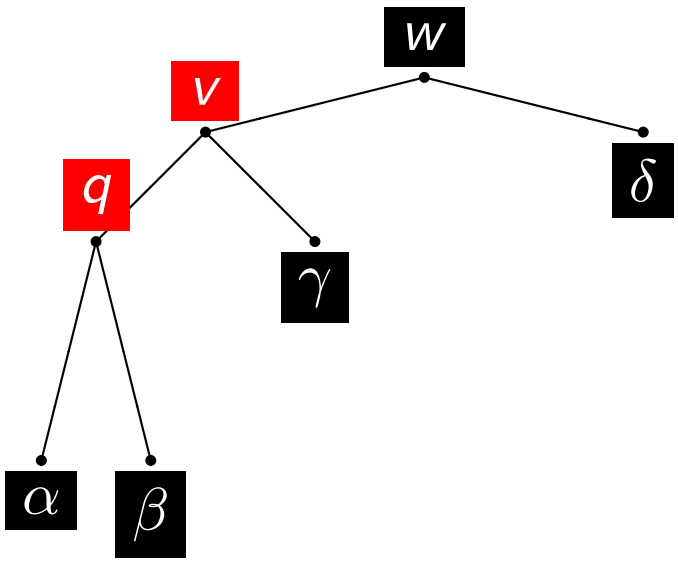
rotação simples a direita + inversão cores de v e w

PPgSC

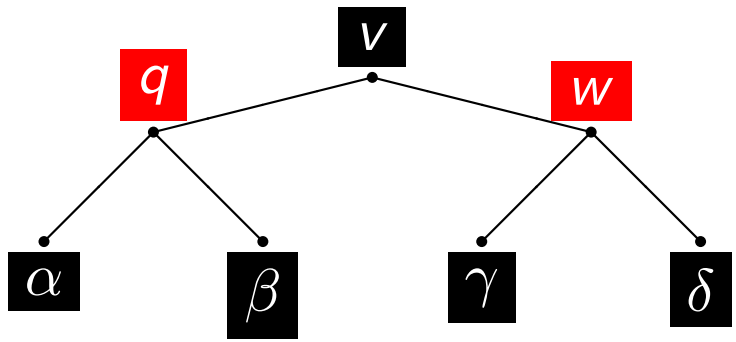
O caso $t.color == \text{BLACK}$ e $w.left == v$ e $v.left == q$

$t.color == \text{BLACK}$ e $w.left == v$ e $v.left == q$

(Antes)



(Depois)



rotação simples a direita + inversão cores de v e w

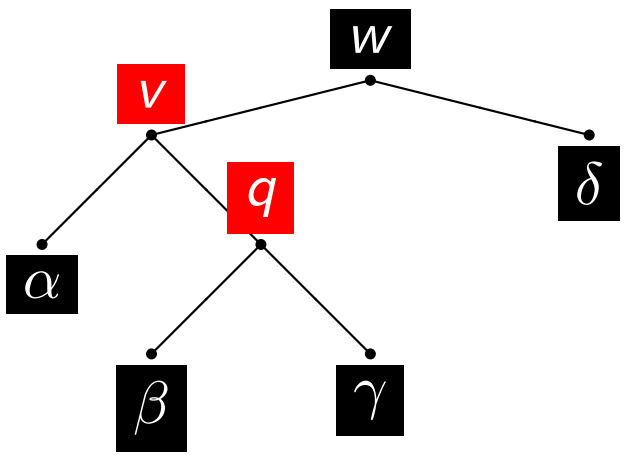
O resultado é uma árvore rubro-negra? Justifique.

PPgSC

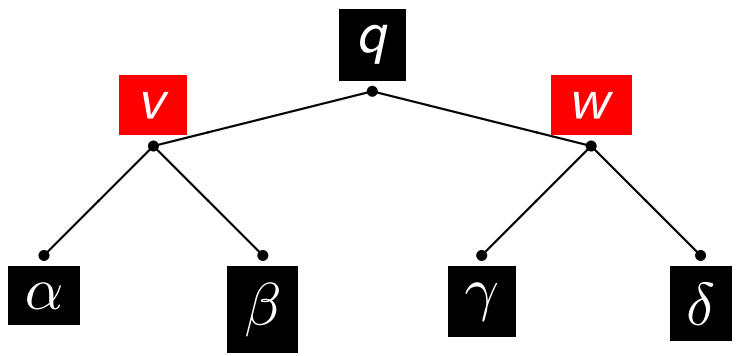
O caso $t.color == \text{BLACK}$ e $w.left == v$ e $v.right == q$

$t.color == \text{BLACK}$ e $w.left == v$ e $v.right == q$

(Antes)



(Depois)



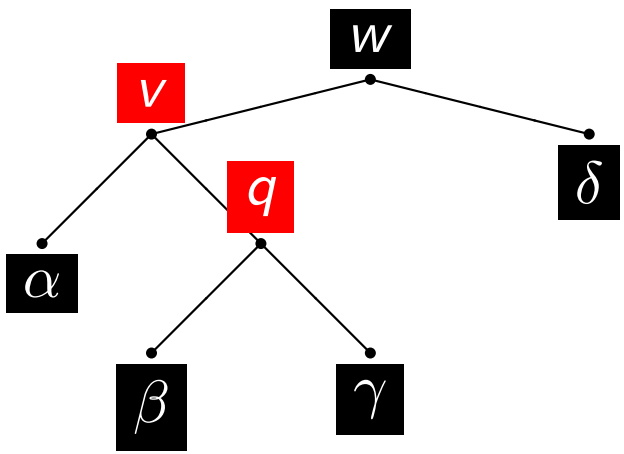
rotação dupla a direita + inversão cores de q e w

PPgSC

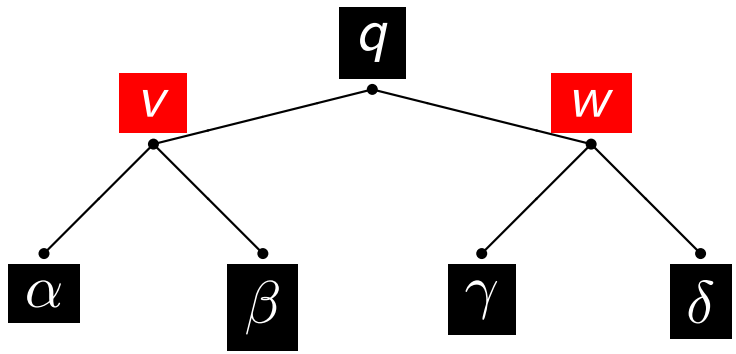
O caso $t.color == \text{BLACK}$ e $w.left == v$ e $v.right == q$

$t.color == \text{BLACK}$ e $w.left == v$ e $v.right == q$

(Antes)



(Depois)



rotação dupla a direita + inversão cores de q e w
O resultado é uma árvore rubro-negra? Justifique.

PPgSC

Algoritmo de inserção

```
INSERT-AUX(k, q, v, w)
1  if q == NIL
2        q = MAKE-NODE(k)
3  elseif k < q.key
4        INSERT-AUX(l, q.left, q, v)
5        if q.color == RED and q.left.color == RED
6              BALANCE(q.left, q, v)
7  elseif k > q.key
8        ...
```

Algoritmo de inserção

```
BALANCE(q, v, w)
 1  if v == w.esq
 2        t = w.right
 3  else t = w.left
 4  if t.color == RED
 5        t.color = v.color = BLACK
 6        w.color = RED
 7  else if q == v.left and v == w.left
 8              v.color = BLACK, w.color = RED
 9              ROTATE-SIMPLE-RIGHT(w)
10        elseif q == v.right and v == w.left
11              q.color = BLACK, w.color = RED
12              ROTATE-DOUBLE-RIGHT(w)
13        elseif q == v.right and v == w.right
14        ...
15  else
16        ...
```

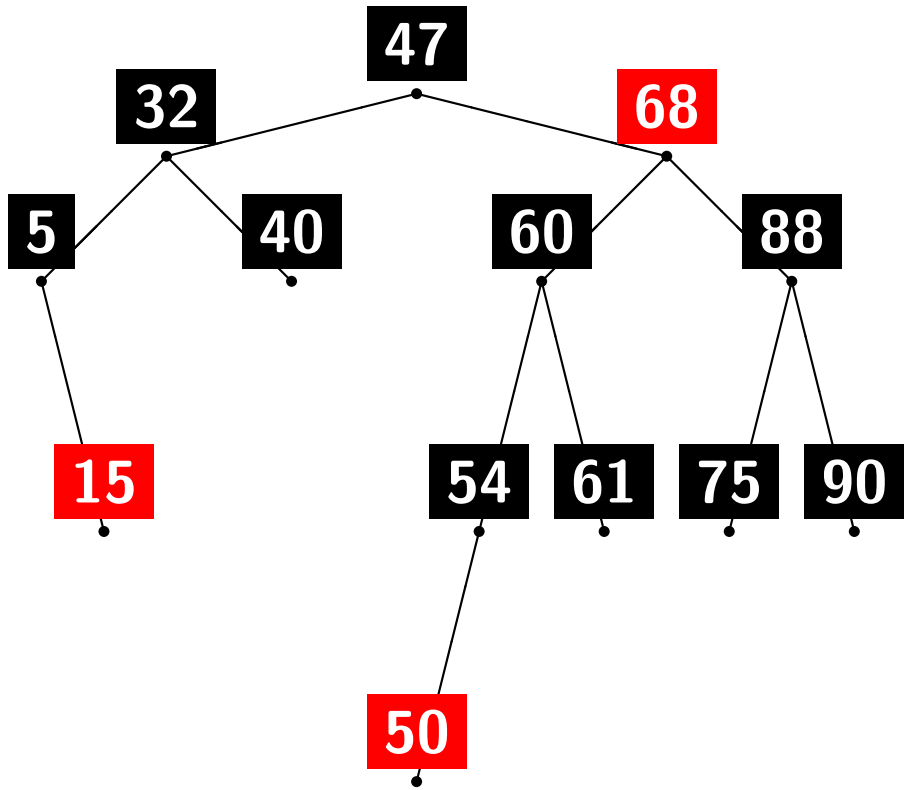
A operação de remoção

1. remoção em árvore binária de busca
2. se necessário, re-balanceamento
3. o nó removido tem pelo menos uma sub-árvore vazia

Análise da remoção

O que fazer quando o nó a remover tem a cor

rubro ?



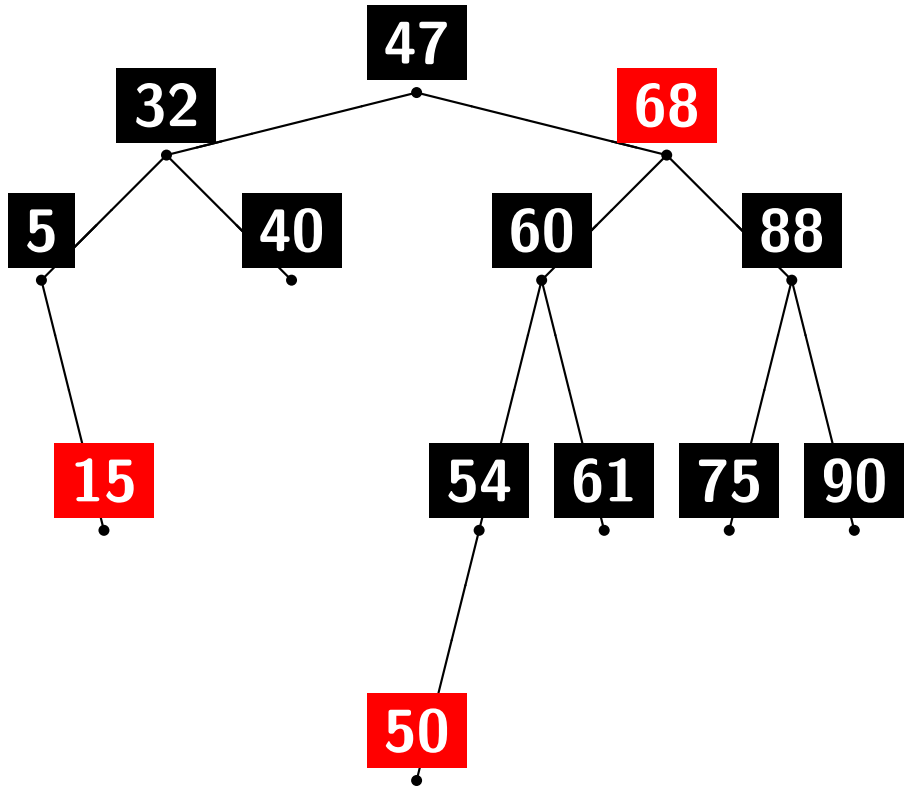
negro ?

PPgSC

Análise da remoção

O que fazer quando o nó a remover tem a cor

rubro ? Nada, pois nenhuma propriedade pode ser quebrada.



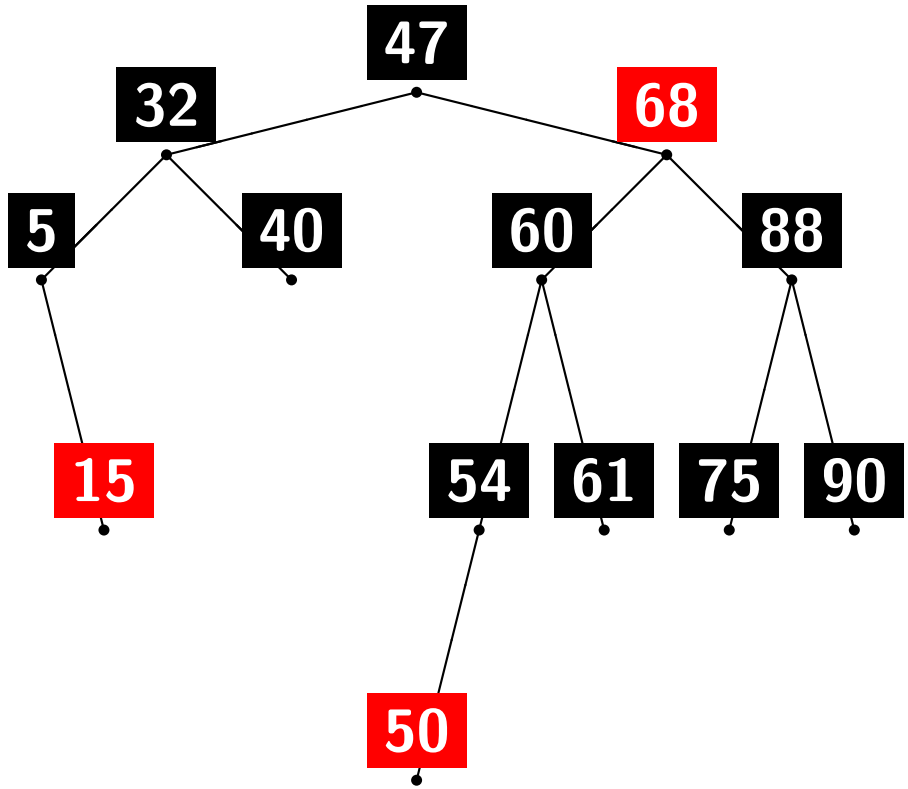
negro ?

Análise da remoção

O que fazer quando o nó a remover tem a cor

rubro ? Nada, pois nenhuma propriedade pode ser quebrada.

Exercício: remover os nós com valores 15 e 50 na árvore seguinte:

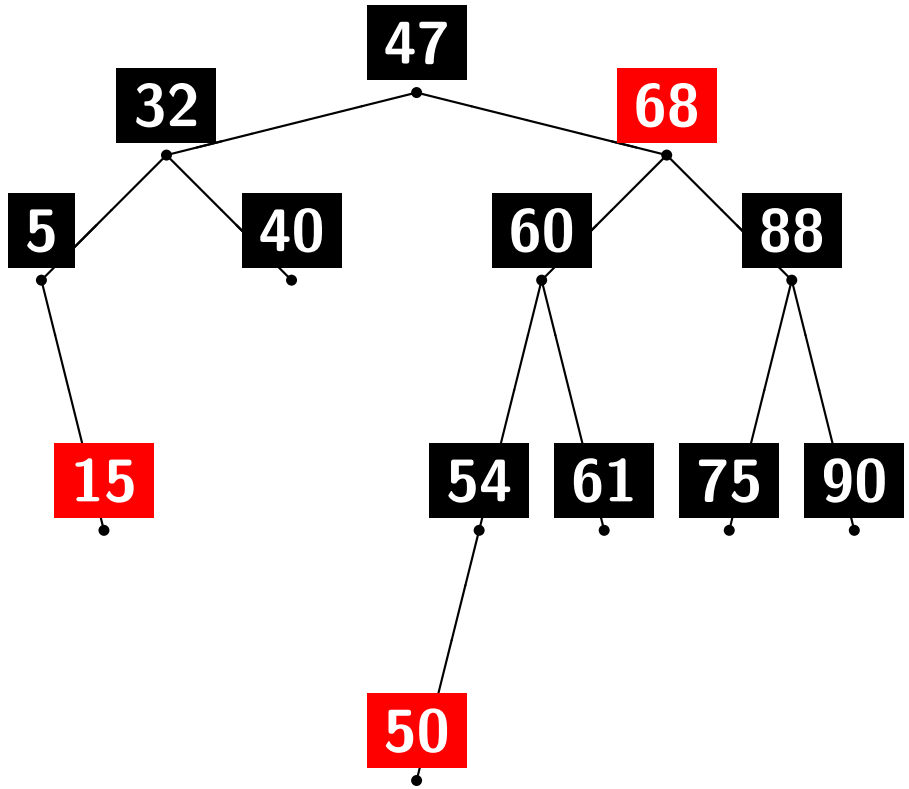


negro ?

Análise da remoção

O que fazer quando o nó a remover tem a cor

rubro ? Nada, pois nenhuma propriedade pode ser quebrada.



negro ? Rebalancear

Remoção de um nó negro

Seja y o nó a remover.

- todos os caminhos da raiz até uma folha passando por y tem um nó negro a menos
- seja x a sub-árvore após a remoção de y
  - x é a sub-árvore vazia (y era uma folha), ou
  - x era a sub-árvore não vazia de y
- solução: incrementar a altura negra de x.
  - se a raiz de x era de cor rubro, passa a ser de cor negro
  - se a raiz de x era de cor negra, a cor é duplamente negro
- se x é a raiz da árvore (global): atribuir negro à cor de x
- se x não é a raiz da árvore:
  - seja $v : x.up$, o ascendente direto de x
  - seja w a outra sub-árvore de v
    w não pode ser a árvore vazia

PPgSC

Remoção de um nó negro

Seja y o nó a remover.

- todos os caminhos da raiz até uma folha passando por y tem um nó negro a menos
- seja x a sub-árvore após a remoção de y
  - x é a sub-árvore vazia (y era uma folha), ou
  - x era a sub-árvore não vazia de y
- solução: incrementar a altura negra de x.
  - se a raiz de x era de cor rubro, passa a ser de cor negro
  - se a raiz de x era de cor negra, a cor é duplamente negro
- se x é a raiz da árvore (global): atribuir negro à cor de x
- se x não é a raiz da árvore:
  - seja $v : x.up$, o ascendente direto de x
  - seja w a outra sub-árvore de v
    w não pode ser a árvore vazia explique o motivo.

PPgSC

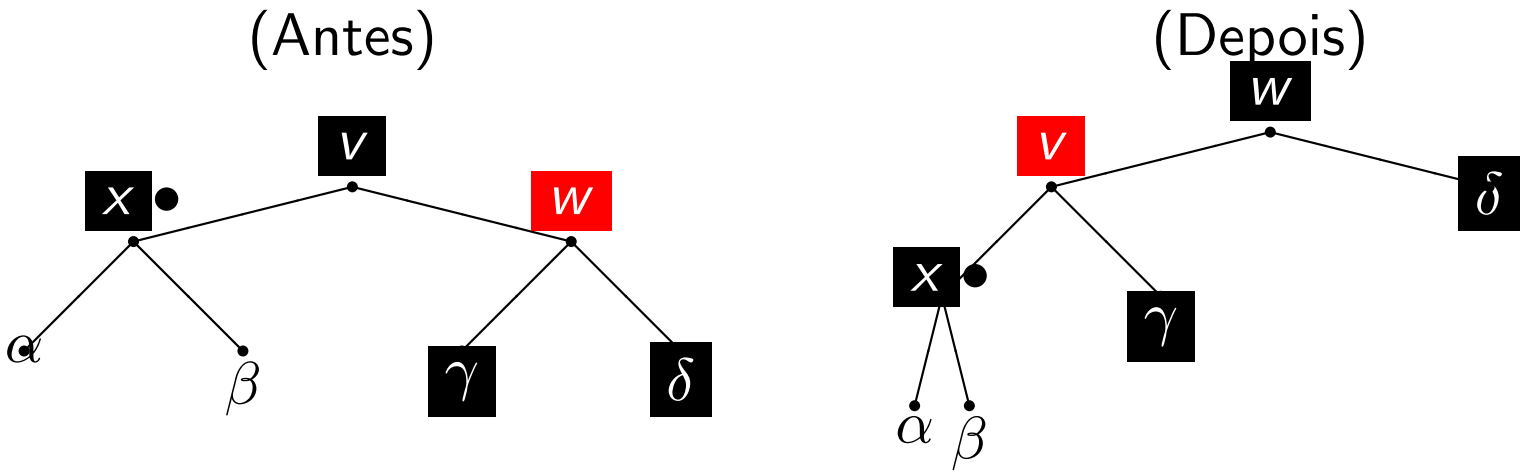
Os diferentes casos

$x \neq root, x.color == \text{BLACK}, x.up == v, w.up == v, w \neq x$

1. $w.color == \text{RED}$
2. $w.color == \text{BLACK}$, as sub-árvores de w são de cor negro.
3. $w.color == \text{BLACK}$, $w.left.color == \text{RED}$, $w.right.color == \text{BLACK}$.
4. $w.color == \text{BLACK}$, $w.right.color == \text{RED}$.

Caso 1

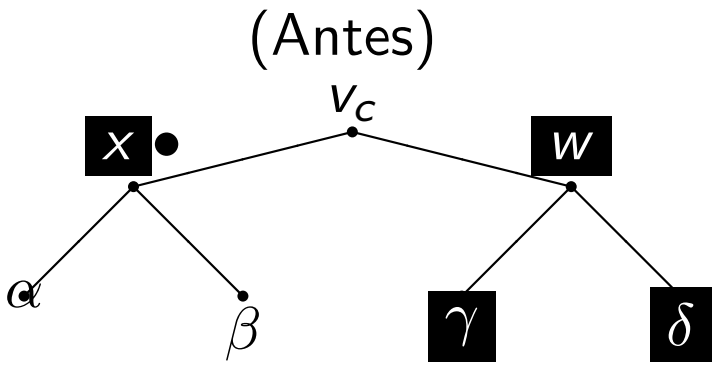
$x \neq root, x.color == \text{BLACK}, x.up == v, w.up == v, w \neq x, w.color == \text{RED}$

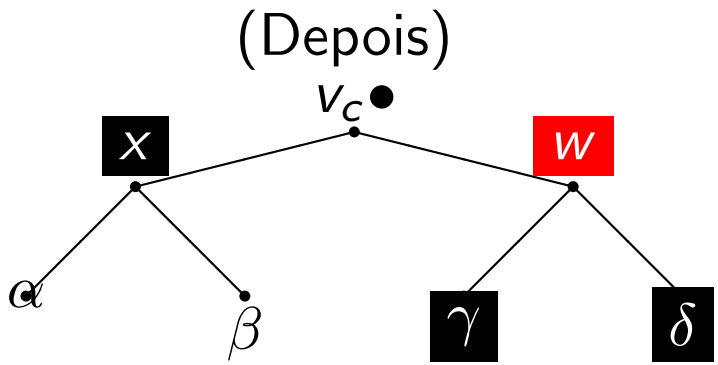


O nó irmão de x tem a cor negra: caso 2, 3 ou 4.

Caso 2

$x \neq root, x.color == \text{BLACK}, x.up == v, w.up == v, w \neq x, w.color == \text{BLACK}, w.left.color == \text{BLACK}, w.right.color == \text{BLACK}$

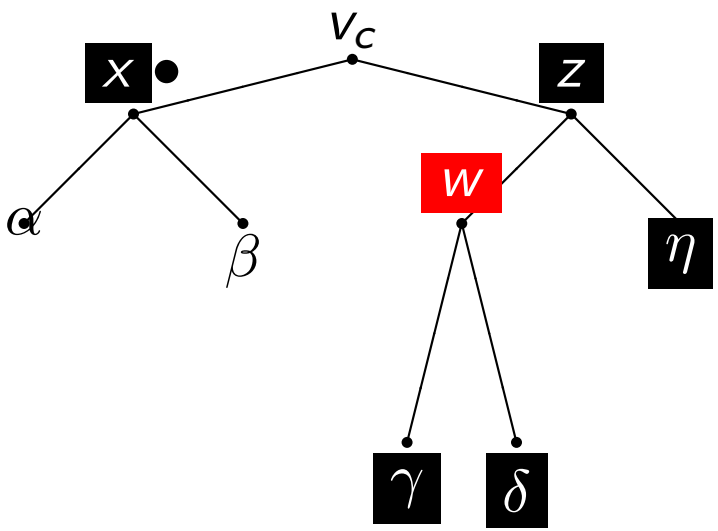




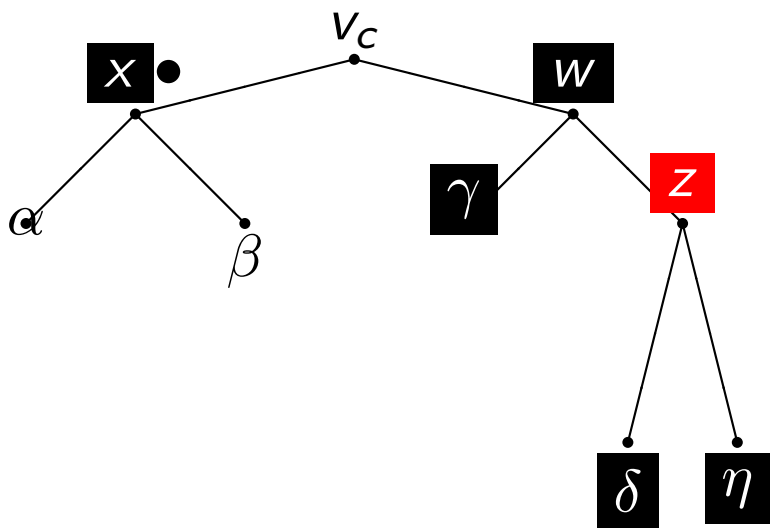
Caso 3

$x \neq root, x.color == \text{BLACK}, x.up == v, w.up == v, w \neq x, w.color == \text{BLACK}, w.left.color == \text{RED}, w.right.color == \text{BLACK}$

(Antes)



(Depois)



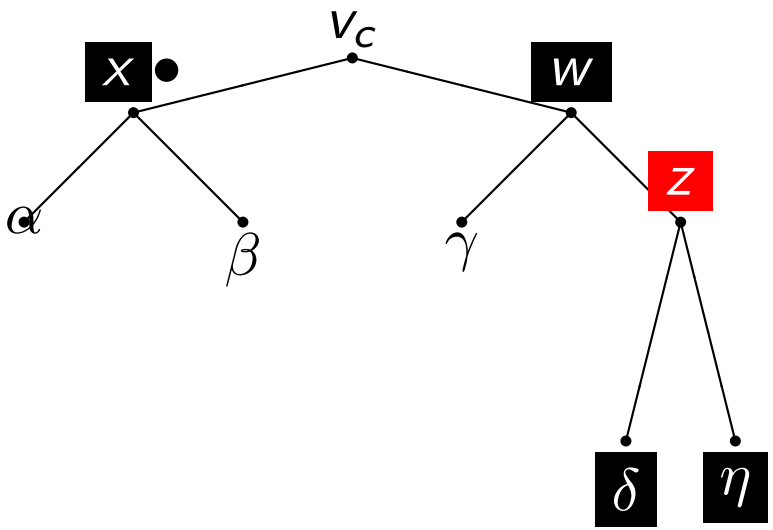
PPgSC

Caso 4

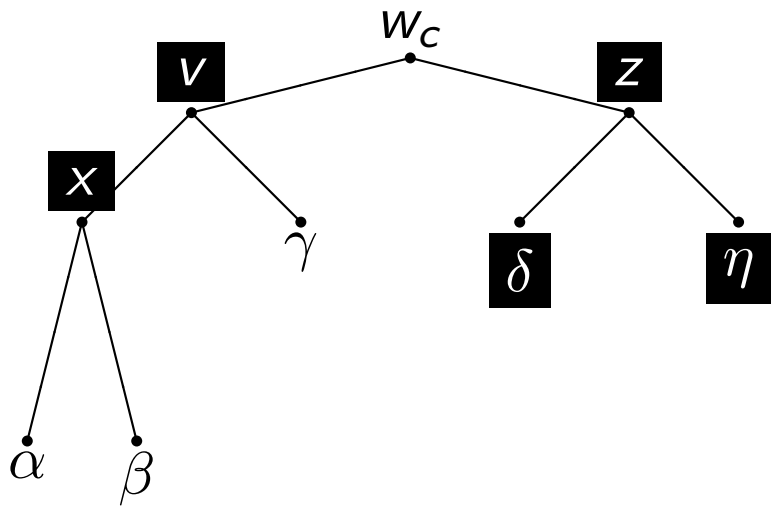
$x \neq root, x.color == \text{BLACK}, x.up == v, w.up == v, w \neq x, w.color == \text{BLACK}, w.right.color == \text{RED}$

.

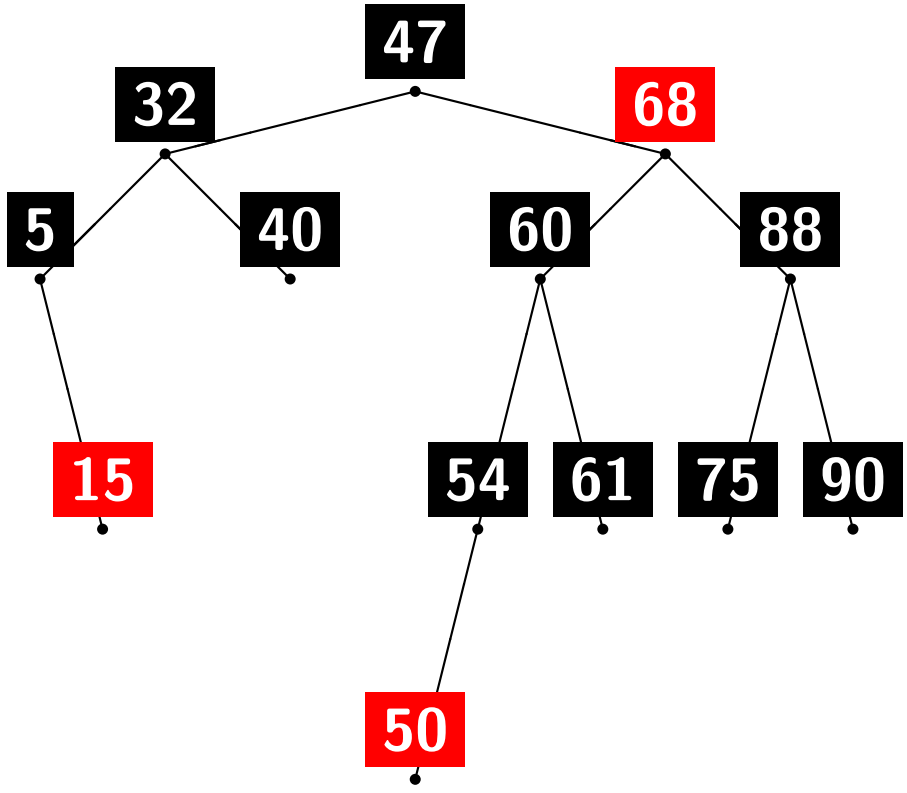
(Antes)



(Depois)



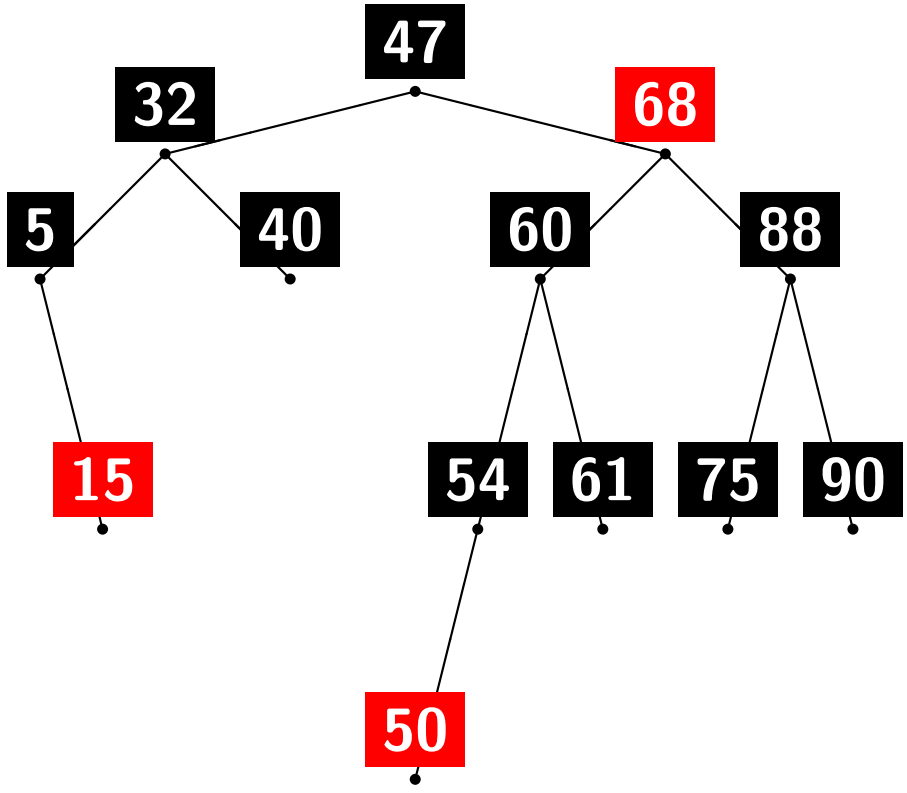
Exercícios



1. Repetidamente remove o valor na raiz até a árvore se tornar vazia
2. Repetidamente remove o menor valor até a árvore se tornar vazia
3. Repetidamente remove o maior valor até a árvore se tornar vazia

PPgSC

Exercícios



1. Repetidamente remove o valor na raiz até a árvore se tornar vazia
2. Repetidamente remove o menor valor até a árvore se tornar vazia
3. Repetidamente remove o maior valor até a árvore se tornar vazia

PPgSC