

# GGI026 - Árvore rubro-negra - Remoção

Marcelo K. Albertini

11 de Setembro de 2013

# Aula de hoje

Nesta aula veremos

- Remoção em Árvores rubro-negras

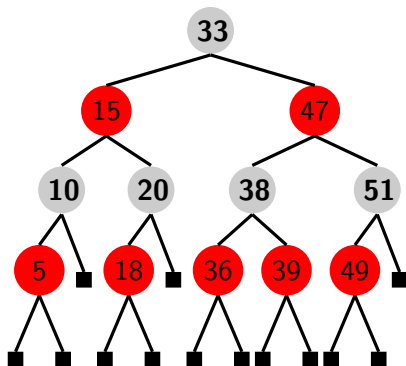
# Remoção em árvores rubro-negras

## Metodologia

- Possibilidade 1: nó com nenhum filho (ou dois filhos-folha vazios)
- Possibilidade 2: nó com um filho não-folha vazio
- Possibilidade 3: dois filhos, usar remoção do sucessor ou antecedente em árvores binárias
  - troca o valor a ser removido com o sucessor ou predecessor
  - então usa remoção de 1 ou 2
- Garantir que todos os casos respeitem e mantenham as propriedades da árvore rubro-negra

Copiar id de sucessor/predecessor não viola nenhuma propriedade rubro-negra.

# Árvore rubro-negra – Propriedades



## Propriedades:

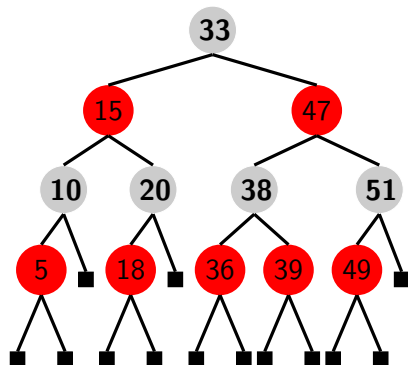
- 1 - Todo nó é vermelho ou preto
- 2 - A raiz é preta
- 3 - Todas as folhas são pretas, considerando inclusive as folhas vazias
- 4 - Se um nó é vermelho então seus filhos são pretos
- 5 - Todo caminho da raiz até qualquer folha tem sempre o mesmo número de nós pretos

# Estrutura de dados

Cada nó da árvore tem uma cor, preto ou vermelho e sabe quem é seu nó-pai e os dois filhos: direita e esquerda.

```
1 class Node {  
2     int cor; // 0 é BLACK preto, 1 é RED vermelho  
3     int id;  
4  
5     Node esq, dir;  
6     Node pai;  
7 }
```

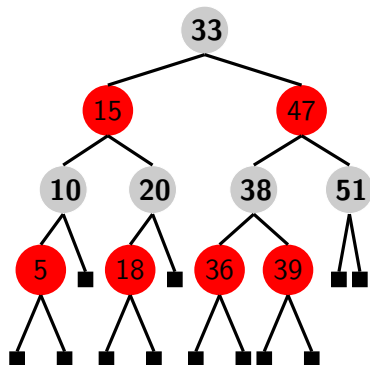
# Árvore rubro-negra – remoção exemplo de possibilidade 1



Se nó para remover for rubro:  
remoção de 49 .

- somente remoção, nenhum ajuste é necessário

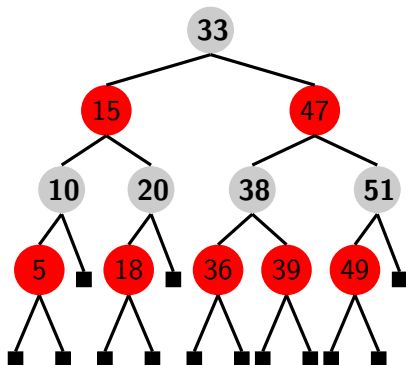
# Árvore rubro-negra – remoção possibilidade 1



Se nó para remover for rubro:  
remoção de 49.

- somente remoção, nenhum ajuste é necessário

## Árvore rubro-negra – possibilidade 2



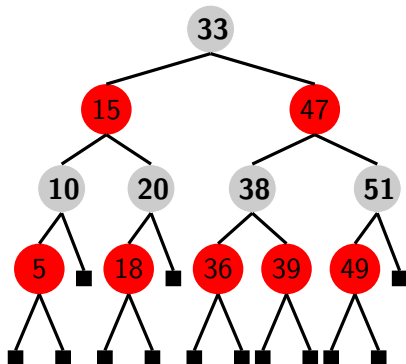
Se nó para remover for rubro e sucessor também: remoção de

47 :

- trocar id com nó sucessor
- coloração não é alterada
- remover então da subárvore do sucessor



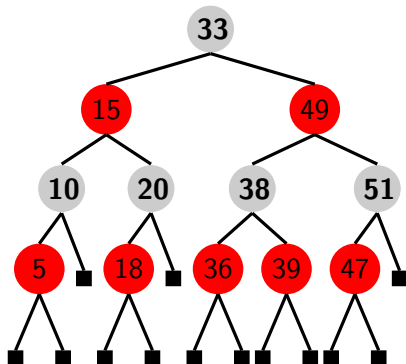
## Árvore rubro-negra – remoção possibilidade 2



Se nó para remover for rubro e filho também: remoção de 47 :

- **trocar id com nó sucessor**
- nó sucessor é 49
  - coloração não é alterada
- remover então da subárvore do sucessor

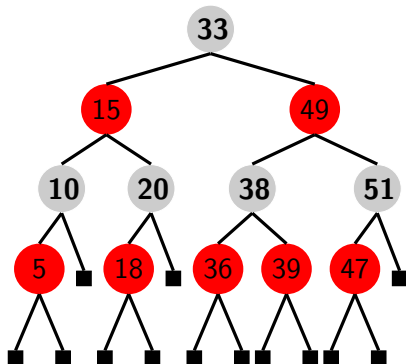
## Árvore rubro-negra: possibilidade 2



Se nó para remover for rubro e filho também: remoção de 47 :

- **trocar id com nó sucessor**
- nó sucessor é 49
  - coloração não é alterada
- remover então da subárvore do sucessor

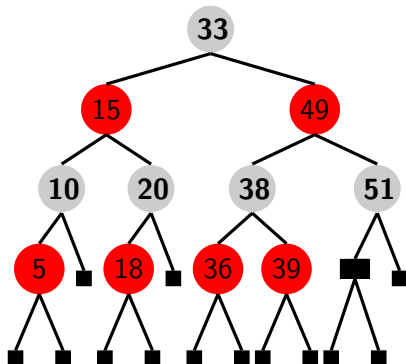
## Árvore rubro-negra: remoção caso 2



Se nó para remover for rubro e filho também: remoção de 47 :

- trocar id com nó sucessor
- nó sucessor é 49
  - coloração não é alterada
- **remover 47 então da subárvore do sucessor**

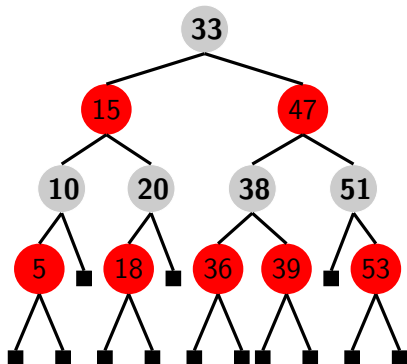
## Árvore rubro-negra – possibilidade 2



Se nó para remover for rubro e filho também: remoção de 47 :

- trocar id com nó sucessor
- nó sucessor é 49
  - coloração não é alterada
- **remover 47 então da subárvore do sucessor**

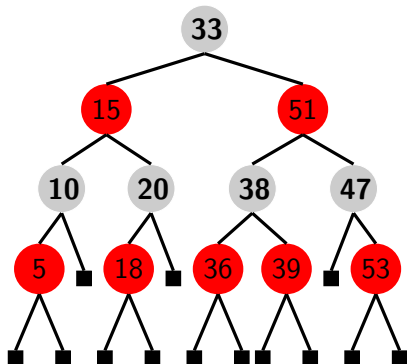
## Árvore rubro-negra – possibilidade 2



Se nó para remover for negro e filho for rubro: remoção de 47 :

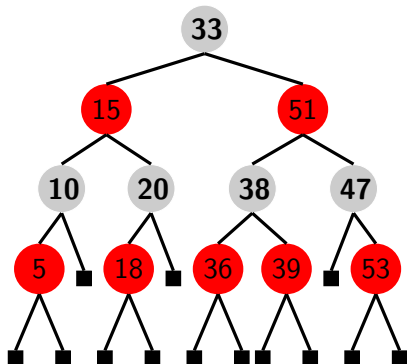
- **trocar id com nó sucessor**
- nó sucessor é 51
  - coloração não é alterada

## Árvore rubro-negra – possibilidade 2



Se nó para remover for negro e filho for rubro: remoção de 47 .

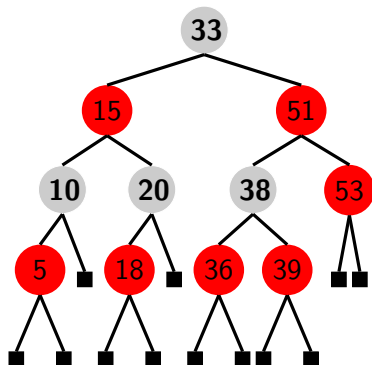
## Árvore rubro-negra – possibilidade 2



Se nó para remover for negro e filho for rubro: **Remover** 47 :

- remover 47
- pintar filho de preto

## Árvore rubro-negra – possibilidade 2

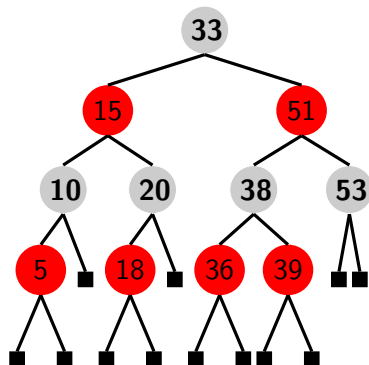


Se nó para remover for negro e filho for rubro: **Remover** 47 :

- remover 47
- pintar filho de preto



## Árvore rubro-negra – possibilidade 2

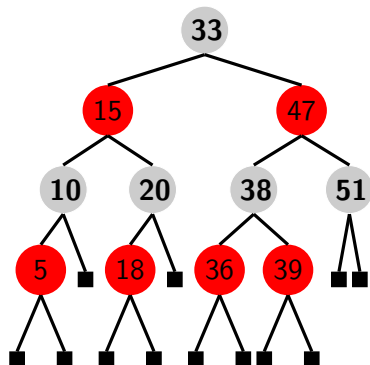


Se nó para remover for negro e filho for rubro: **remover** 47 :

- remover 47
- **pintar filho de preto**

Dessa forma mantém os números de nós pretos de todos os caminhos iguais.

# Árvore rubro-negra – remoção



Se nó para remover for negro:  
remoção de 51 .

• ...

## Função irmão(Node n)


```
1 Node irmão(Node n) {  
2     if (n == n.pai.esq)  
3         return n.pai.dir;  
4     else  
5         return n.pai.esq;  
6 }
```

Condição: **(n)** tem no máximo um filho não nulo

```
1 void excluir_um_filho(Node n) {
2     Node filho;
3
4     if (n.dir != null)
5         filho = n.dir;
6     else
7         filho = n.esq;
8
9     substituir_no(n, filho);
10
11     if (n.cor == BLACK) {
12         if (filho.cor == RED)
13             filho.cor = BLACK;
14         else
15             exclusao_caso1(filho);
16     }
17 }
```

substituir\_no(n, filho) substitui filho no lugar de **(n)** na árvore

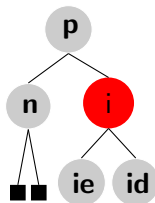
## Exclusão: caso 1

Caso 1:  é a nova raiz. Removemos nó negro de cada caminho e a nova raiz é negra, então propriedades são preservadas.

```
1 void exclusao_caso1(Node n) {  
2     if (n.pai != null) // nao estamos na raiz  
3         exclusao_caso2(n);  
4 }
```

## Exclusão: caso 2



```
1 void exclusao_caso2(Node n) {  
2     Node i = irmao(n);  
3  
4     if (i.cor == RED) {  
5         n.pai.cor = RED;  
6         i.cor = BLACK;  
7  
8         if (n == n.pai.esq)  
9             rotacao_esq(n.pai);  
10        else  
11            rotacao_dir(n.pai);  
12    }  
13    exclusao_caso3(n);  
14 }
```



### Passos

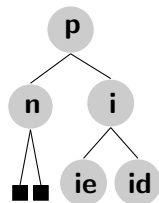
- Rotacionar à esquerda de



- Trocar cores de  e 

## Exclusão: caso 3

```
1 void exclusao_caso3(Node n) {  
2     Node i = irmao(n);  
3  
4     if ((n.pai.cor == BLACK) &&  
5         (i.cor == BLACK) &&  
6         (i.esq.cor == BLACK) &&  
7         (i.dir.cor == BLACK)) {  
8         i.cor = RED;  
9         exclusao_caso1(n.pai);  
10    } else  
11  
12    exclusao_caso4(n);  
13 }
```

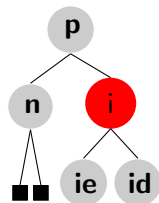


Passos: se pai **p**, irmão **i** e filhos de irmão **i** são pretos.

- Repintamos irmão **i** de vermelho

## Exclusão: caso 3

```
1 void exclusao_caso3(Node n) {
2   Node i = irmao(n);
3
4   if ((n.pai.cor == BLACK) &&
5       (i.cor == BLACK) &&
6       (i.esq.cor == BLACK) &&
7       (i.dir.cor == BLACK)) {
8     i.cor = RED;
9     exclusao_caso1(n.pai);
10  } else
11
12  exclusao_caso4(n);
13 }
```



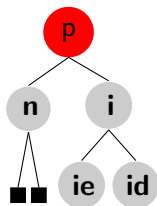
Passos: se pai **p**, irmão **i** e filhos de irmão **i** são pretos.

- Repintamos irmão **i** de vermelho



## Exclusão: caso 4

```
1 exclusao_caso4(Node n) {  
2     Node i = irmao(n);  
3  
4     if ((n.pai.cor == RED) &&  
5         (i.cor == BLACK) &&  
6         (i.esq.cor == BLACK) &&  
7         (i.dir.cor == BLACK)) {  
8         i.cor = RED;  
9         n.pai.cor = BLACK;  
10    } else  
11        exclusao_caso5(n);  
12 }
```

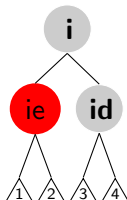


Passos: se pai **p** é rubro, mas irmão **i** e filhos são pretos

- Repintamos irmão **i** de rubro.
- Repintamos pai **p** de negro

## Exclusão: caso 5

```
1 void exclusao_caso5(Node n) {
2   Node i = irmao(n);
3   if (i.cor == BLACK) {
4     if ((n == n.pai.esq) &&
5         (i.dir.cor == BLACK) &&
6         (i.esq.cor == RED)) {
7       i.cor = RED;
8       i.esq.cor = BLACK;
9       rotacao_dir(i);
10    } else if ((n == n.pai.dir) &&
11               (i.esq.cor == BLACK) &&
12               (i.dir.cor == RED)) {
13       i.cor = RED;
14       i.dir.cor = BLACK;
15       rotacao_esq(i);
16    }
17  }
18  exclusao_caso6(n);
19 }
```

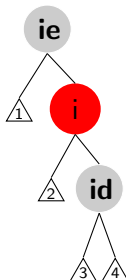


Passos:

- rotação de **i**
- troca cor de **i** com seu novo pai

## Exclusão: caso 5

```
1 void exclusao_caso5(Node n) {
2   Node i = irmao(n);
3   if (i.cor == BLACK) {
4     if ((n == n.pai.esq) &&
5         (i.dir.cor == BLACK) &&
6         (i.esq.cor == RED)) {
7       i.cor = RED;
8       i.esq.cor = BLACK;
9       rotacao_dir(i);
10    } else if ((n == n.pai.dir) &&
11               (i.esq.cor == BLACK) &&
12               (i.dir.cor == RED)) {
13       i.cor = RED;
14       i.dir.cor = BLACK;
15       rotacao_esq(i);
16    }
17  }
18  exclusao_caso6(n);
19 }
```



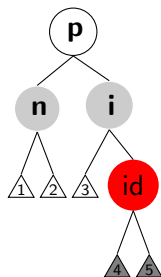
Passos:

- rotação de **i**
- troca cor de **i** com seu novo pai

## Exclusão: caso 6

```
1 void exclusao_caso6(Node n) {  
2     Node i = irmao(n);  
3  
4     i.cor = n.pai.cor;  
5     n.pai.cor = BLACK;  
6  
7     if (n == n.pai.esq) {  
8         i.dir.cor = BLACK;  
9         rotacao_esq(n.pai);  
10    } else {  
11        i.esq.cor = BLACK;  
12        rotacao_dir(n.pai);  
13    }  
14 }
```

Nó branco  $\textcircled{p}$  representa qualquer cor.



Passos:

- irmão copia cor do pai
- pai torna-se preto