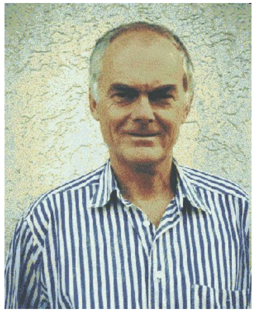
INF 1010
Estruturas de Dados Avançadas

Árvores rubro-negras

Árvores rubro-negras ou vermelho e preto (red-black tree)



Rudolf Bayer (1939-...)

Rudolf Bayer (1972). "Symmetric binary B-Trees: Data structure and maintenance algorithms". *Acta Informatica* **1** (4): 290--306.

Árvores Rubronegras

Uma árvore rubronegra é uma árvore binária de busca com os ponteiros nulos sendo substituídos por nós externos pretos e satisfazendo as seguintes restrições:

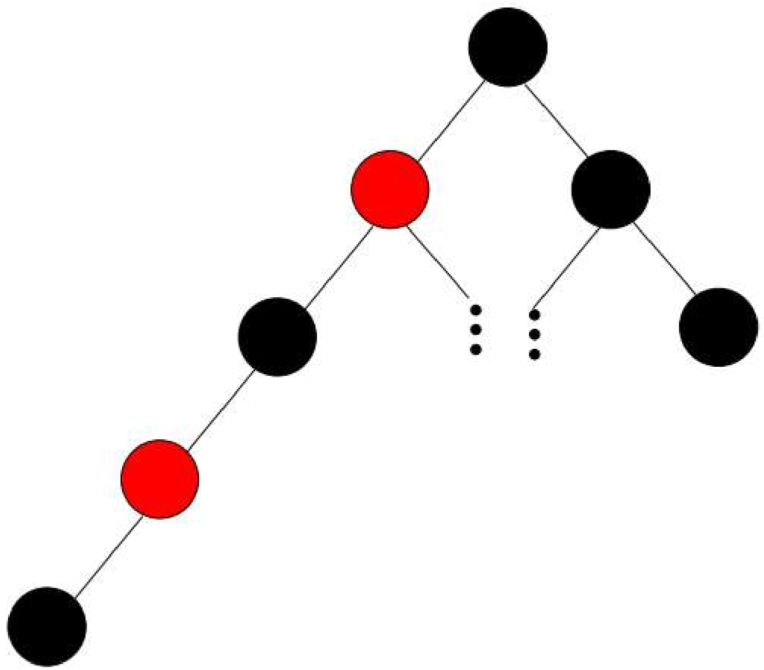
1. A raiz é preta (RB1)
2. Nós vermelhos só tem filhos pretos (RB2)

   (ou: nenhum caminho da raiz até uma folha pode conter 2 nós vermelhos consecutivos)

3. Todos os caminhos a partir da raiz da árvore até suas folhas passa pelo mesmo número de nós pretos (RB3)

Árvores Rubronegras

- Estas restrições garantem que o maior caminho entre a raiz e as folhas é no máximo o dobro do menor caminho.
- Elas garantem um balanceamento razoável!



Outra definição

Propriedades dos nós:

RB1: Raiz e nós externos são pretos

RB2: Nenhum caminho da raiz até um nó externo pode possuir dois **nós vermelhos consecutivos**

RB3: Todos os caminhos da raiz até um nó externo devem possuir o mesmo número de **nós pretos**.

Propriedades das arestas:

Uma aresta de um pai para um **filho preto** é **preta**.

Uma aresta de um pai para um **filho vermelho** é **vermelha**.

RB1': Arestas de um nó interno para um nó externo são **pretas**

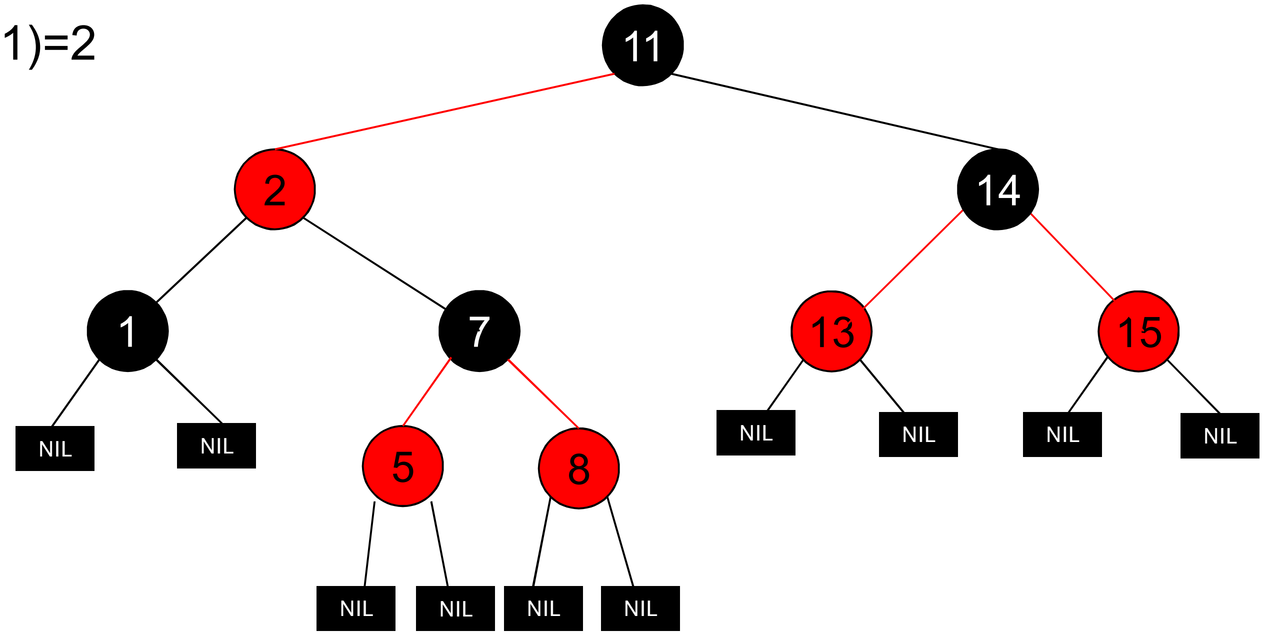
RB2': Nenhum caminho da raiz até um nó externo pode ter duas **arestas vermelhas consecutivas**

RB3': Todos os caminhos da raiz até um nó externo têm o mesmo número de **arestas pretas**.

Formas de representação

h(11)=4

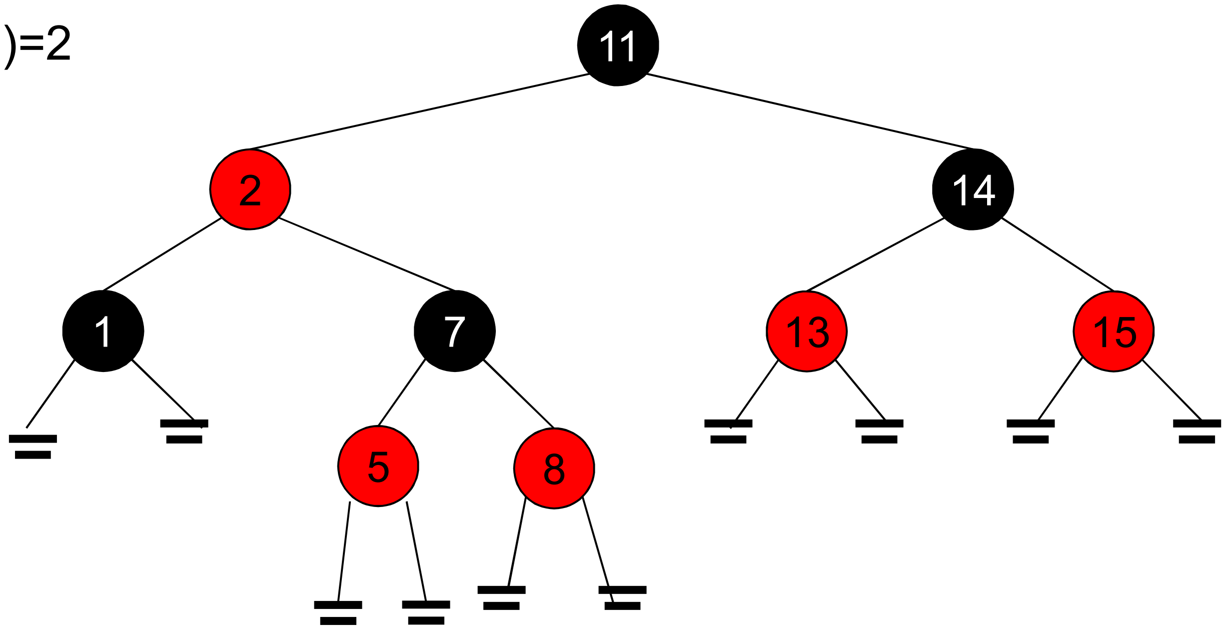
bh(11)=2



Formas de representação

h(11)=4

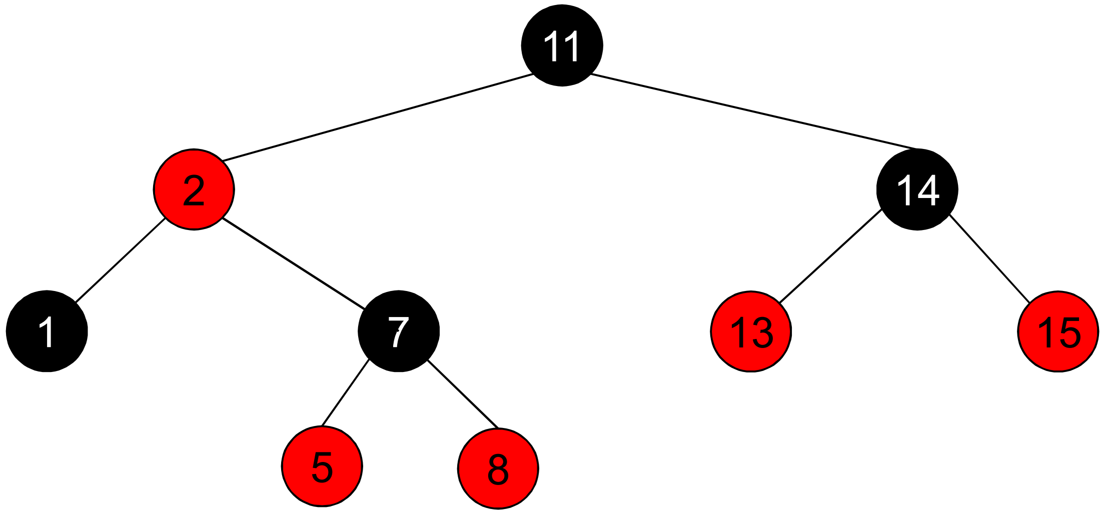
bh(11)=2



Formas de representação

h(11)=4

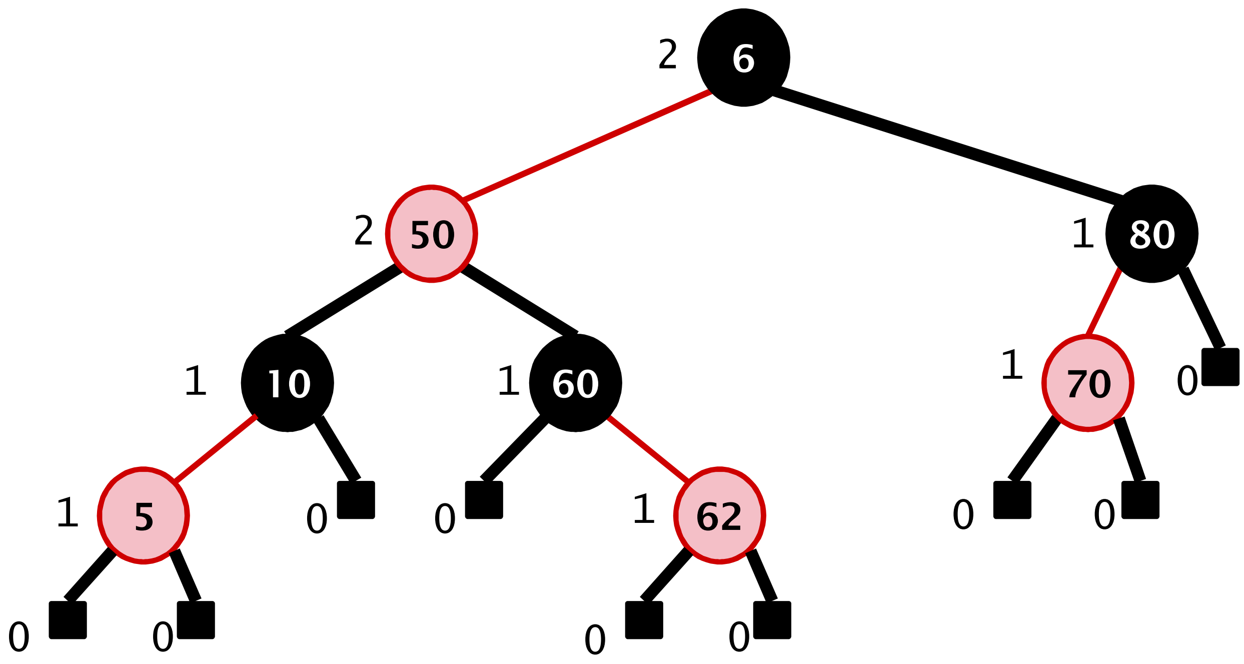
bh(11)=2



Árvore Red-Black - definição

Altura negra (rank) de um nó:

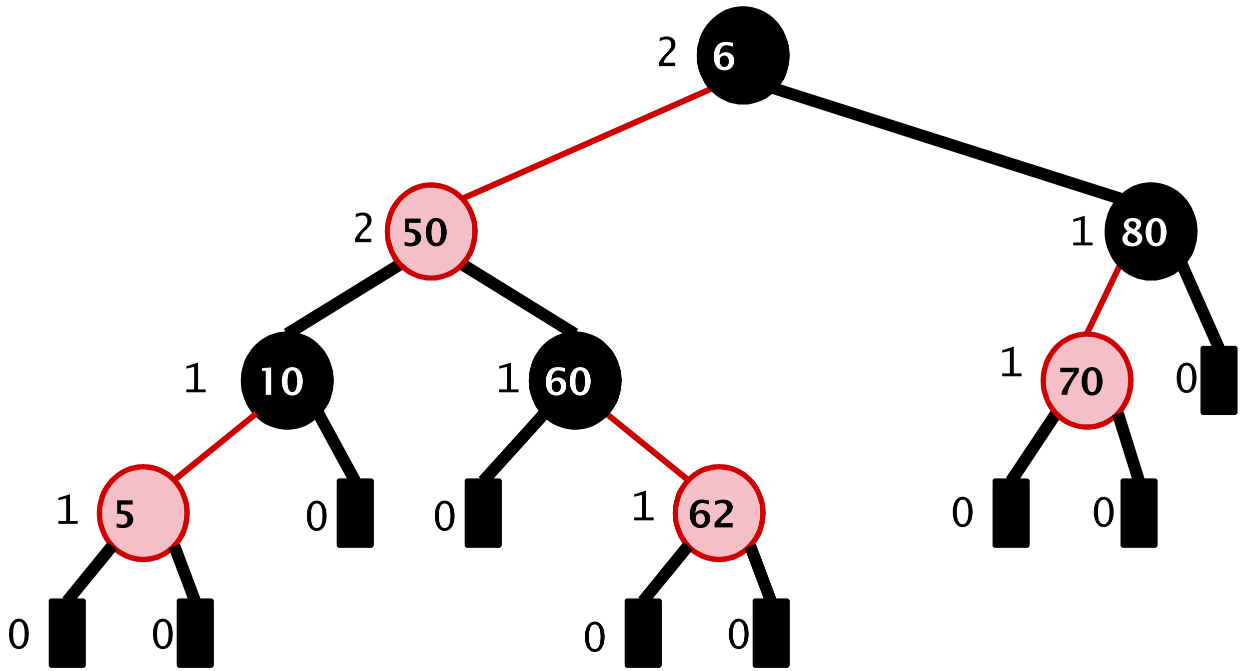
número de **arestas pretas** de qualquer caminho desde o nó até um nó externo



Árvore Red-Black - conceitos

Sejam P e Q dois caminhos da raiz até nós externos:

comprimento(P) $\leq$ 2$\times$comprimento(Q)



$h <= 2*bh$

até profundidade bh, árvore cheia:

$n >= 2^{bh}-1$ (n = número nós na árvore)

$\log(n+1) >= bh$ => $h <= 2*\log(n+1)$

Árvore Red-Black - busca

igual à busca numa árvore binária de busca

Árvore Red-Black - inserção

árvore vazia → novo nó preto

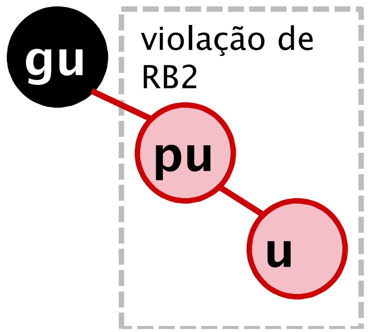
árvore não vazia → novo nó vermelho

se houver violação de RB2, a árvore está desequilibrada

u e pu vermelhos → pu não pode ser raiz (RB1)

gu existe e é preto (RB2)

(u = novo nó; pu = pai de u; gu = avô de u)



Árvore Red-Black - inserção

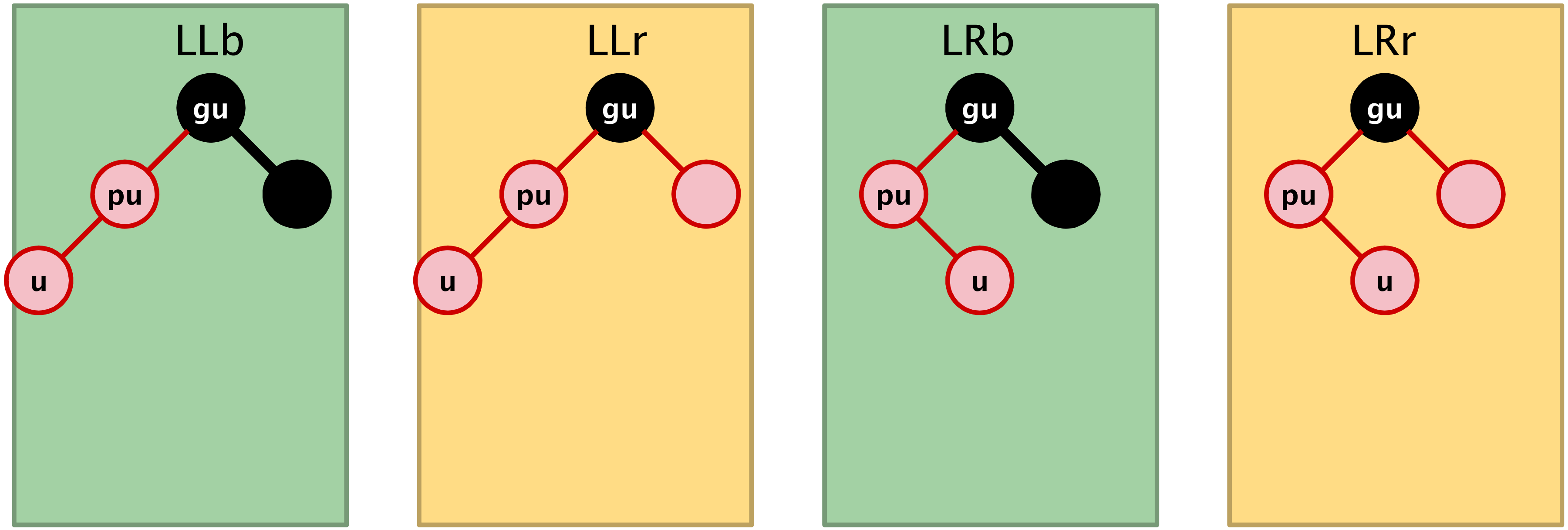
árvore vazia $\rightarrow$ novo nó preto

mais fácil pensarmos que sempre inserimos um nó novo como vermelho

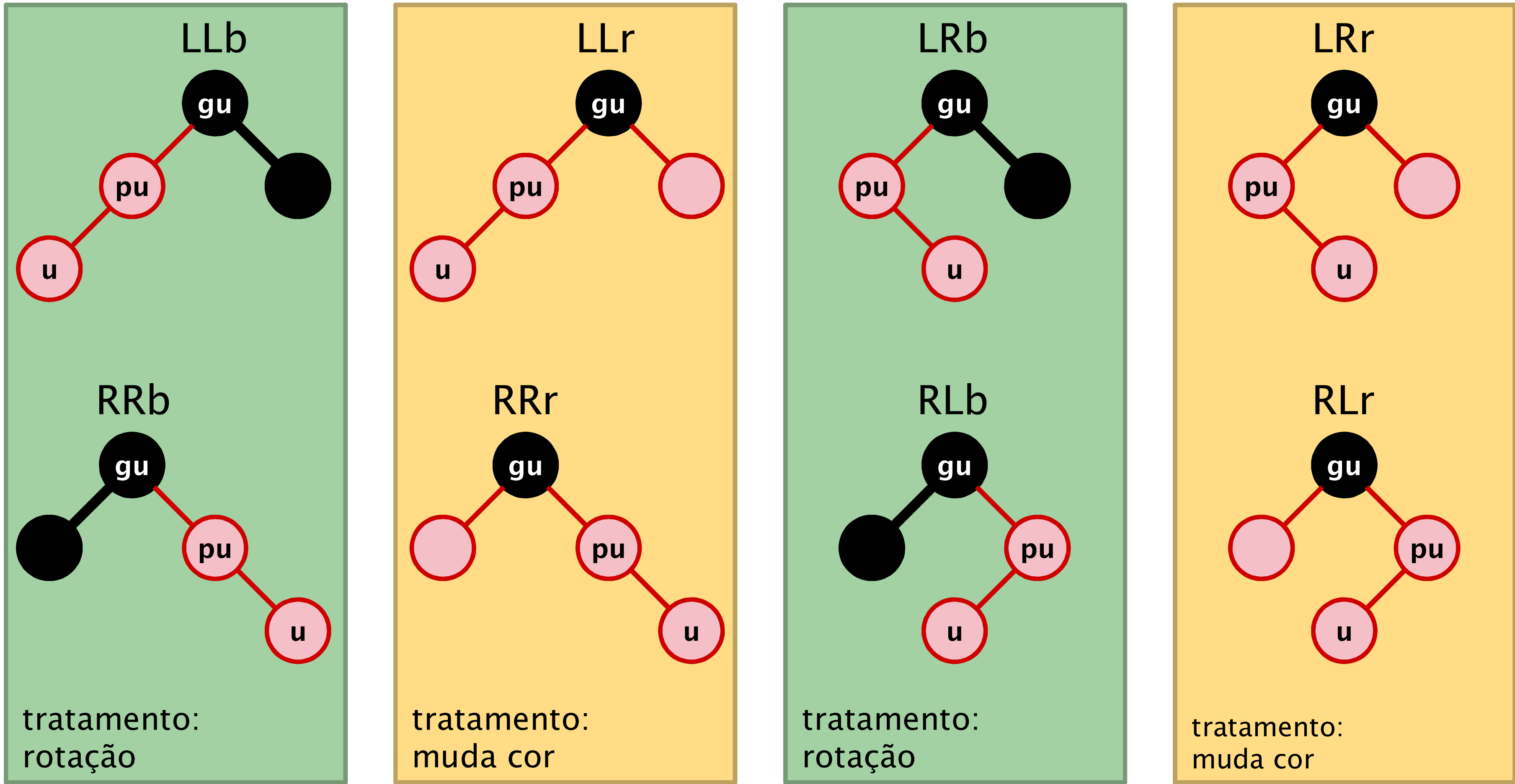
raiz pode sempre ser transformada em nó preto

Tipos de desbalanceamento

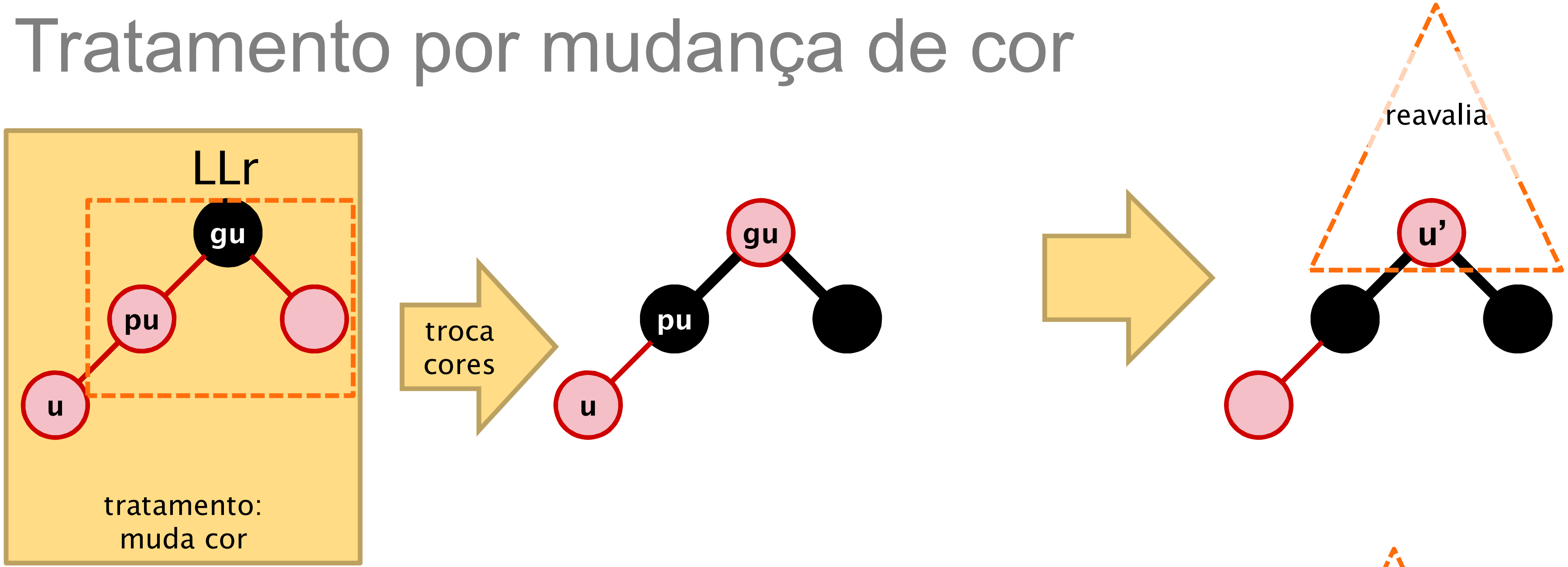
notação XYc: pu é filho X {L,R} de gu
u é filho Y {L,R} de pu
outro filho de gu é de cor c {b,r}

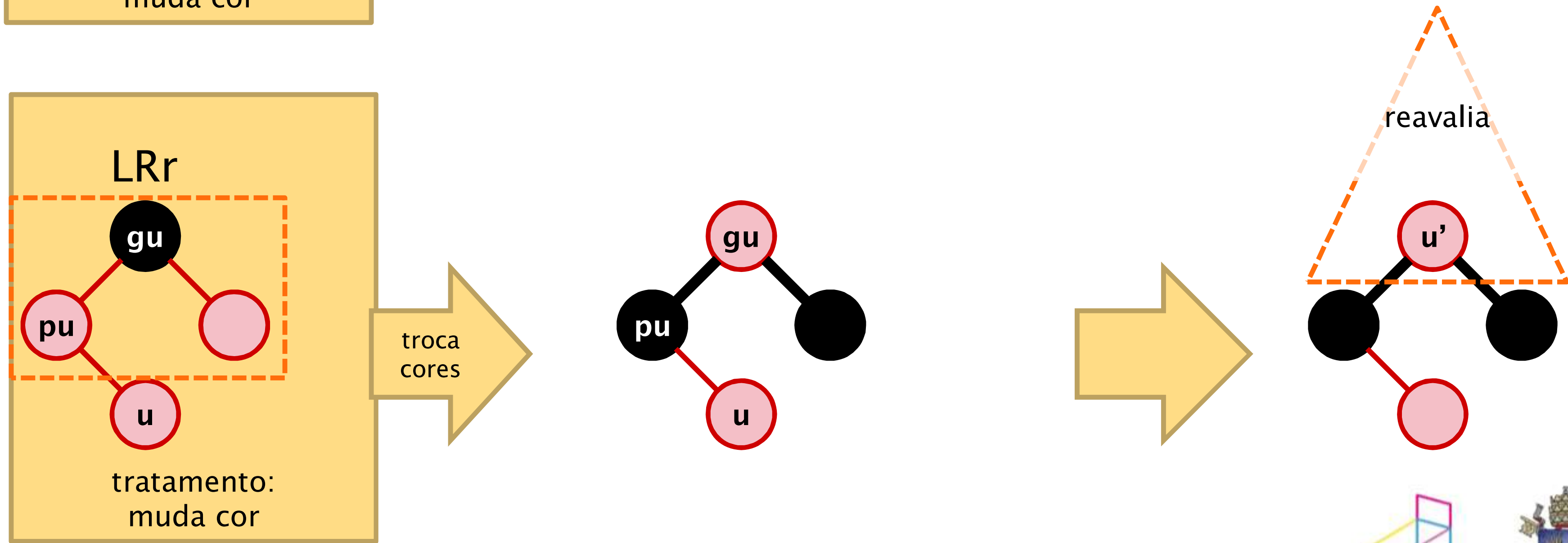


Tipos de desbalanceamento

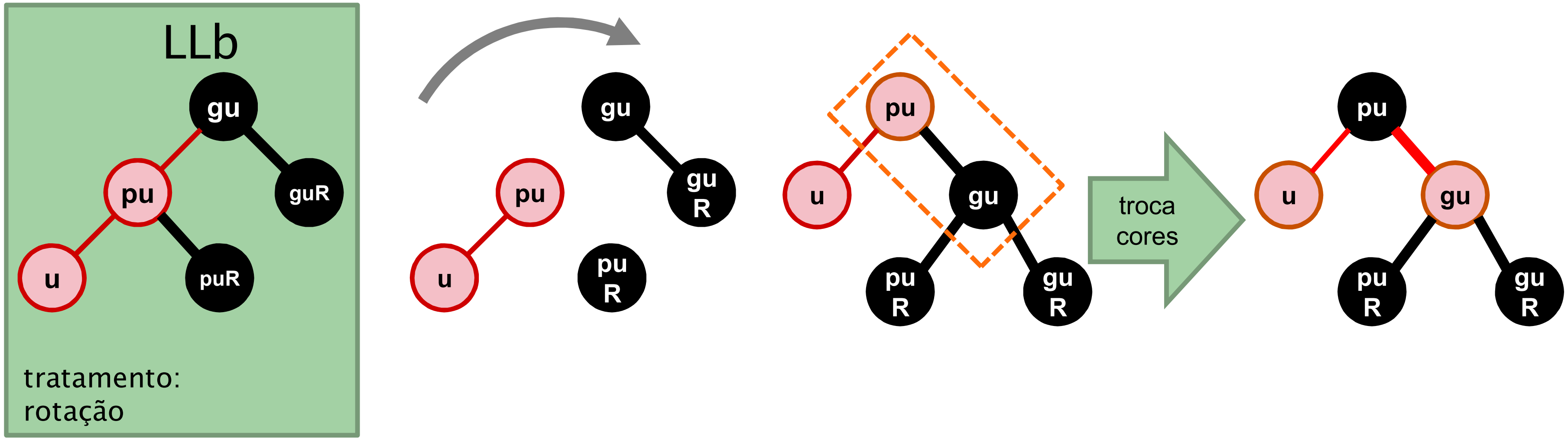


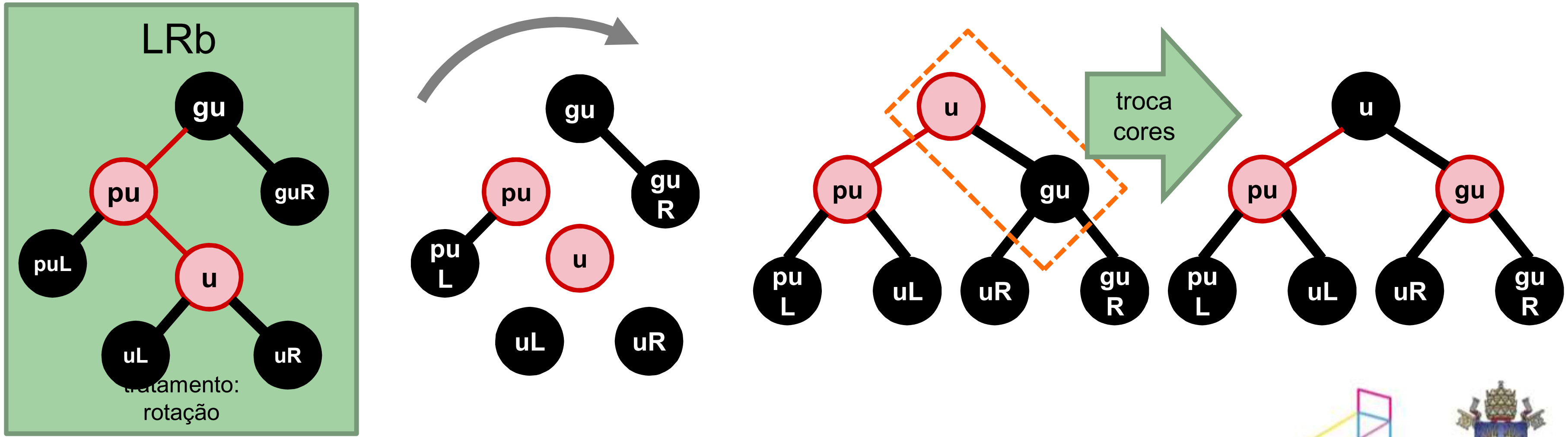
Tratamento por mudança de cor



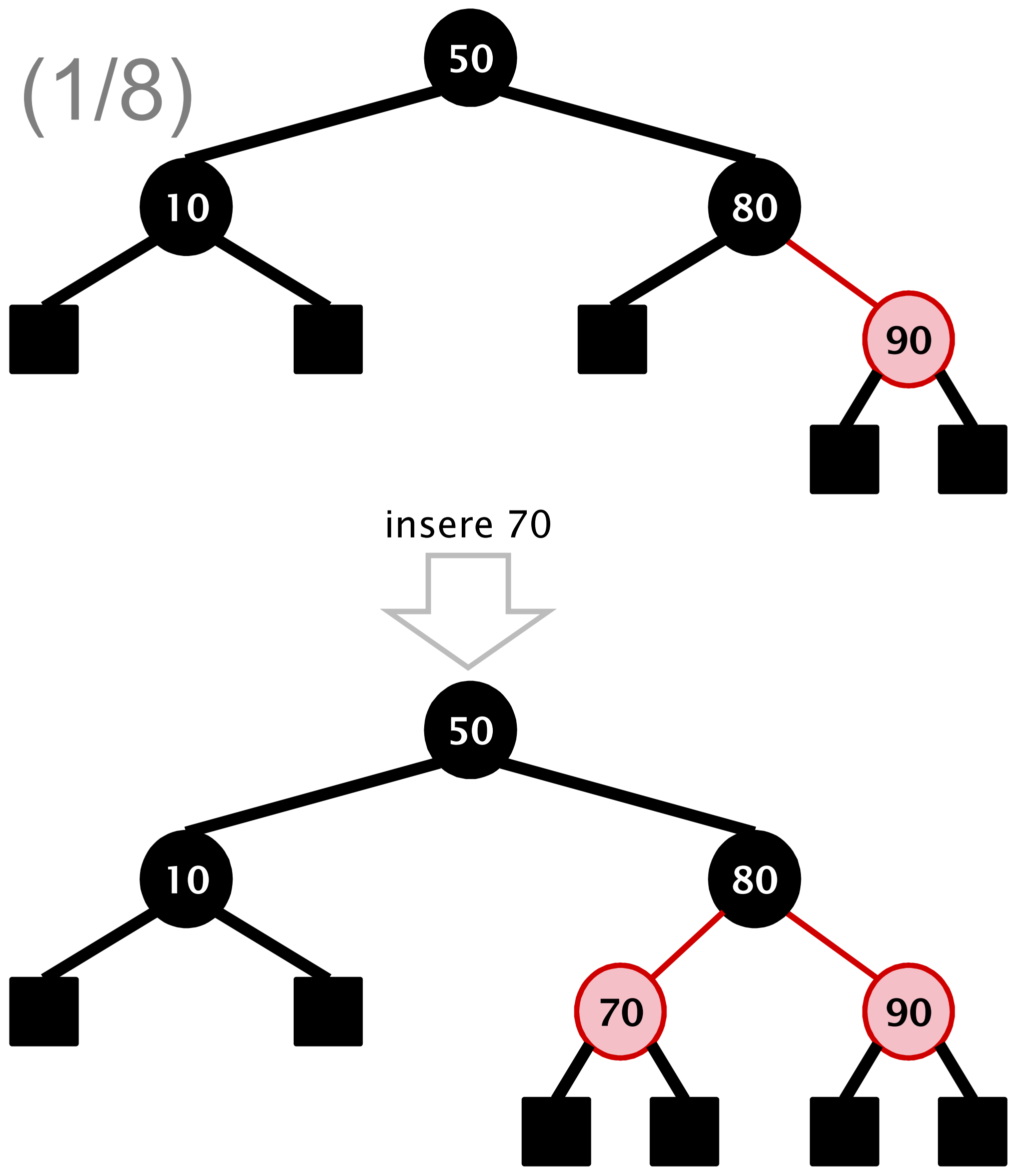


Tratamento por rotação

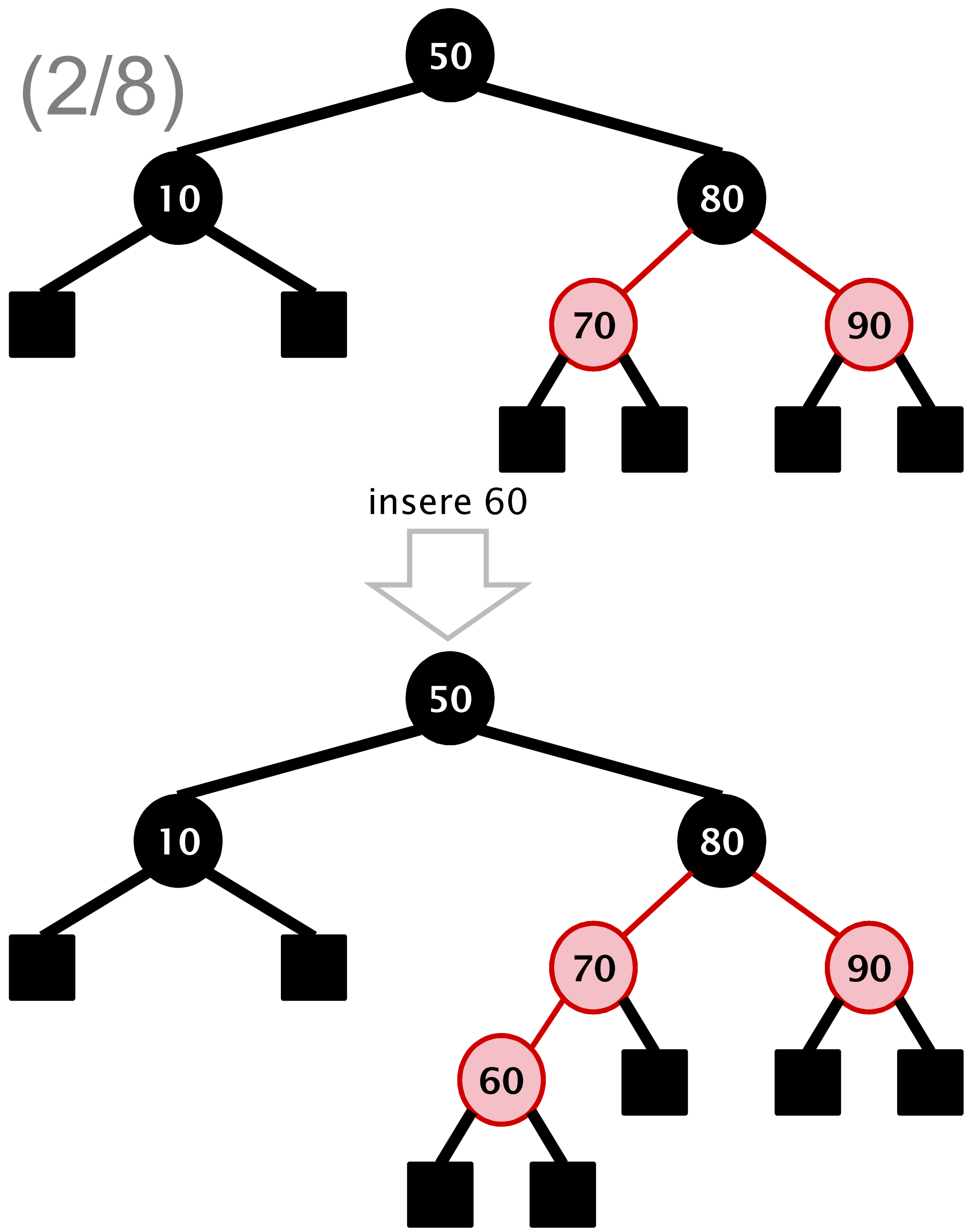




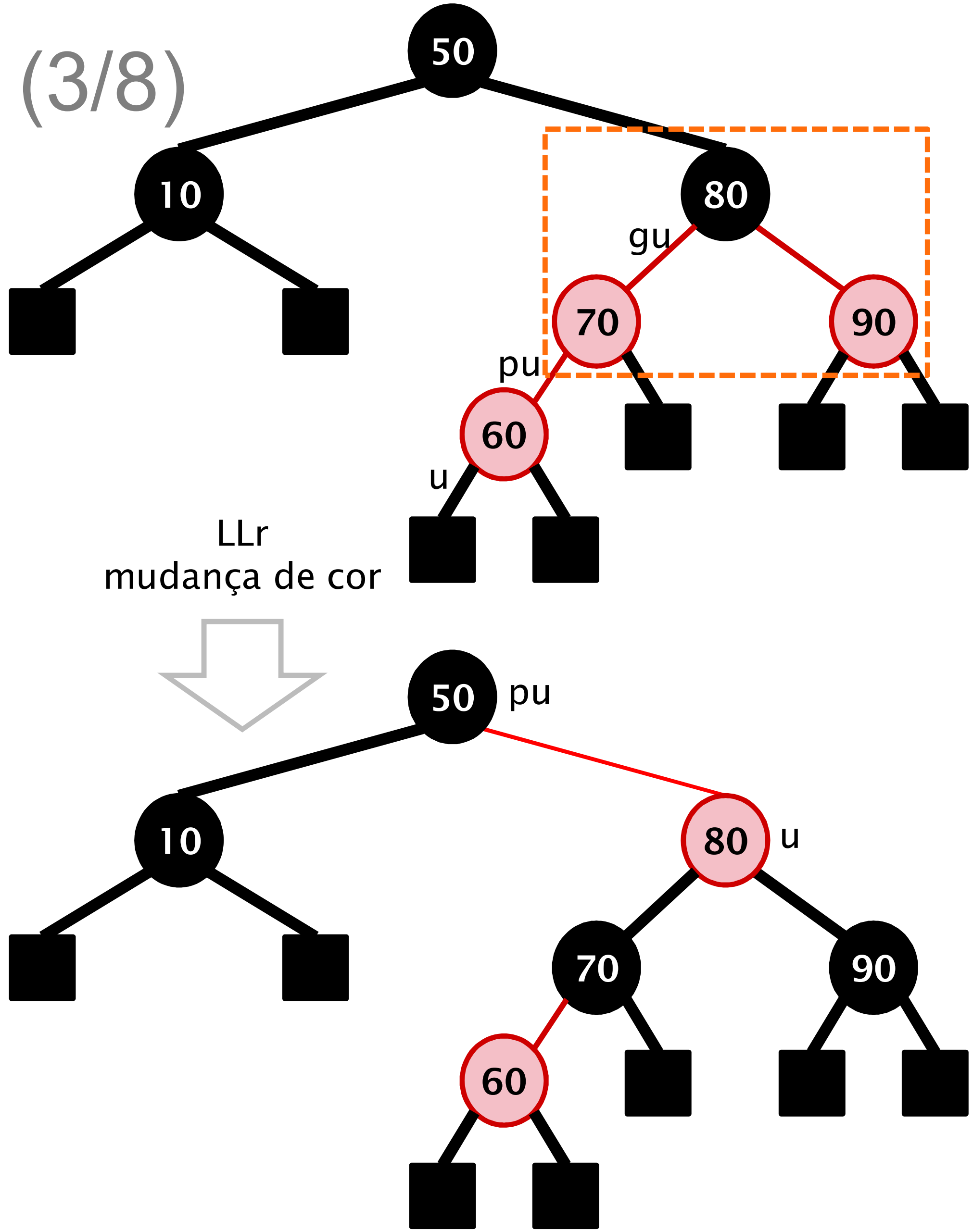
Exemplo (1/8)



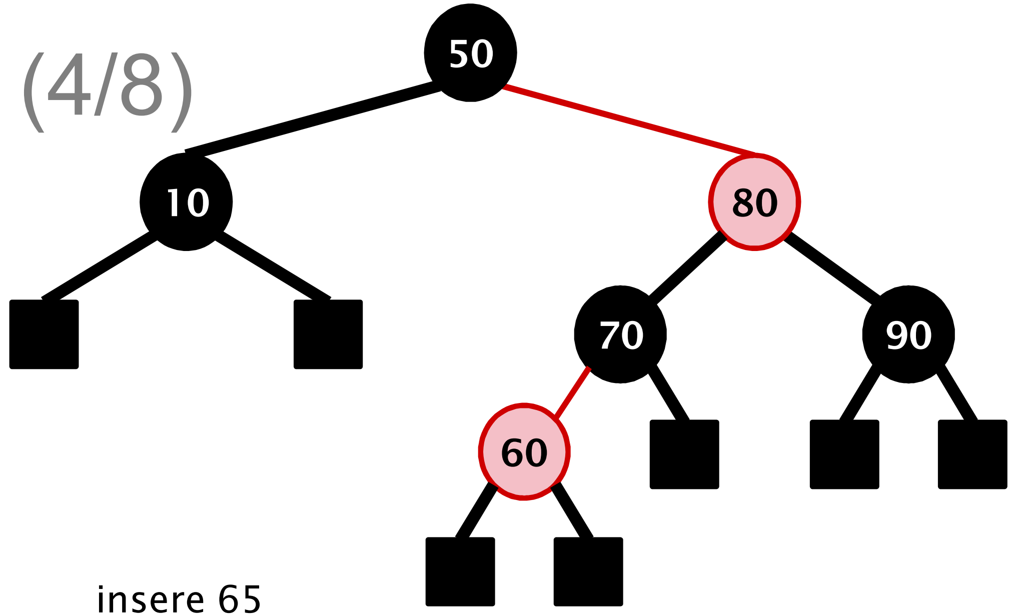
Exemplo (2/8)



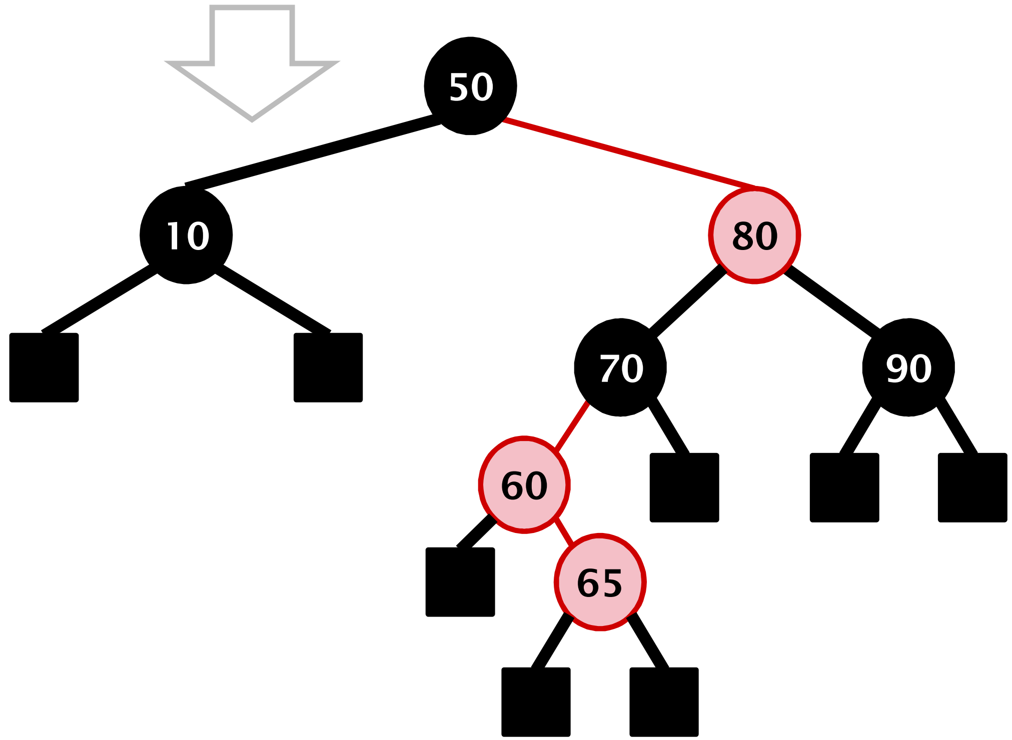
Exemplo (3/8)



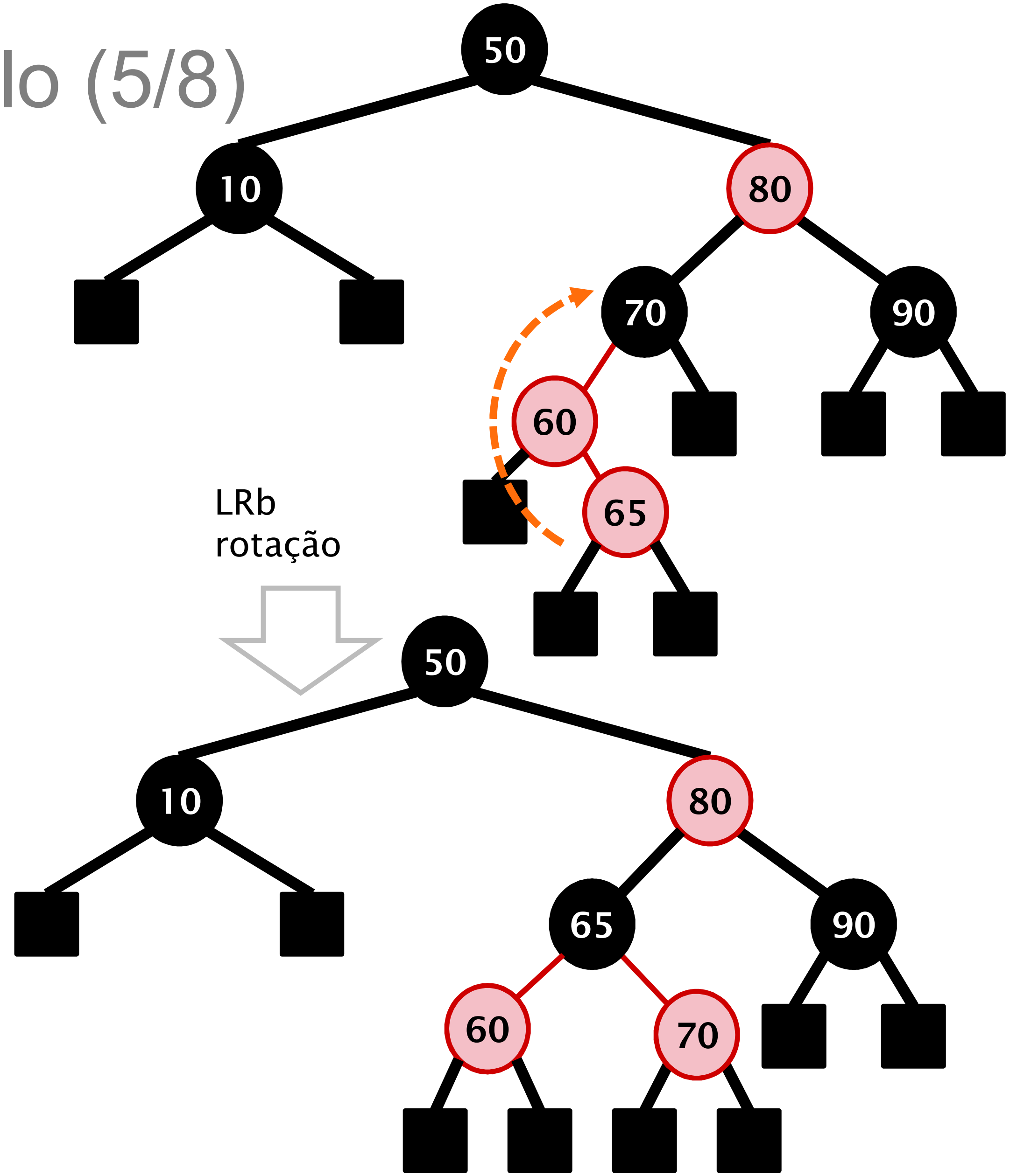
Exemplo (4/8)



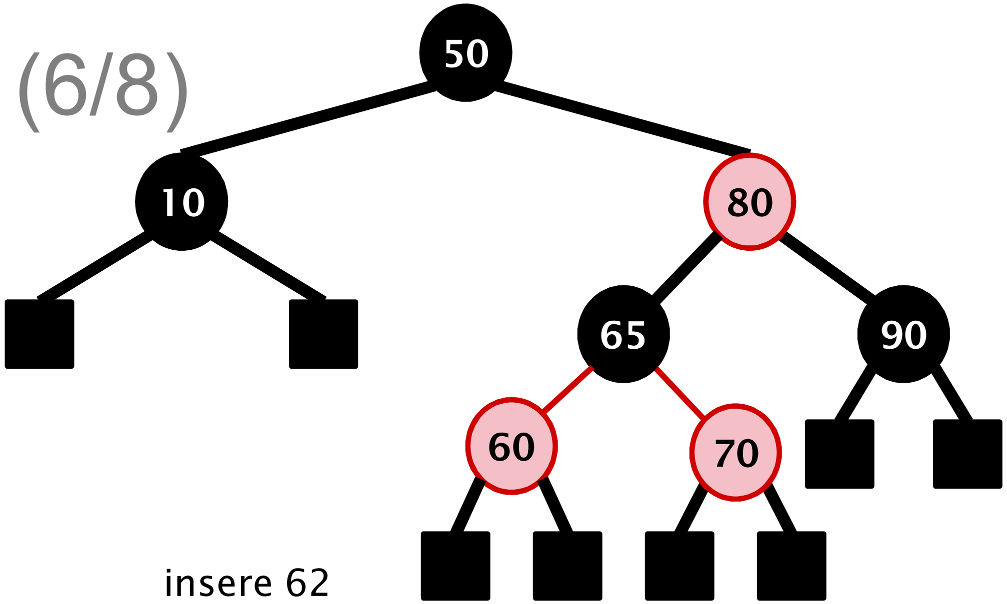
insere 65



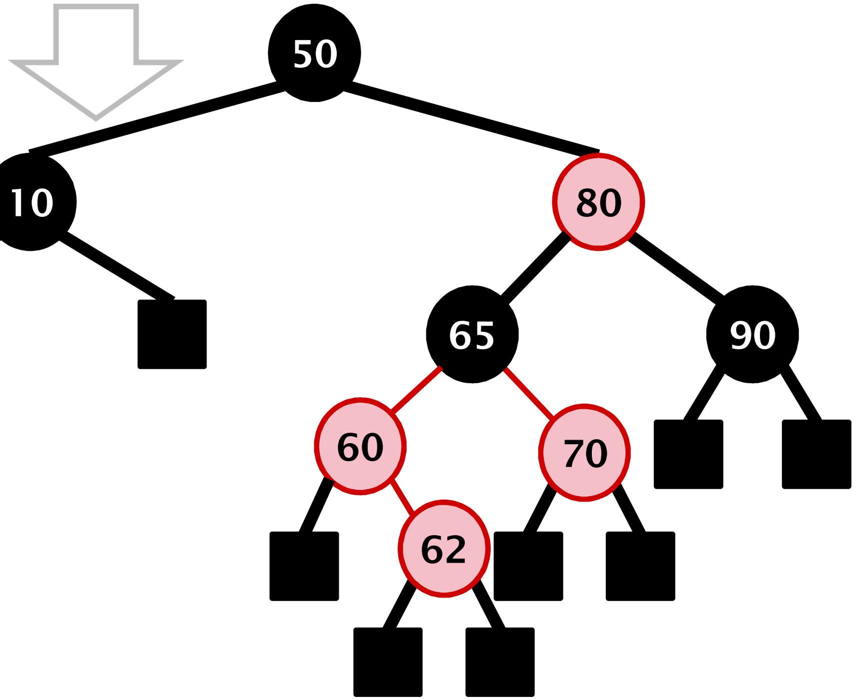
Exemplo (5/8)



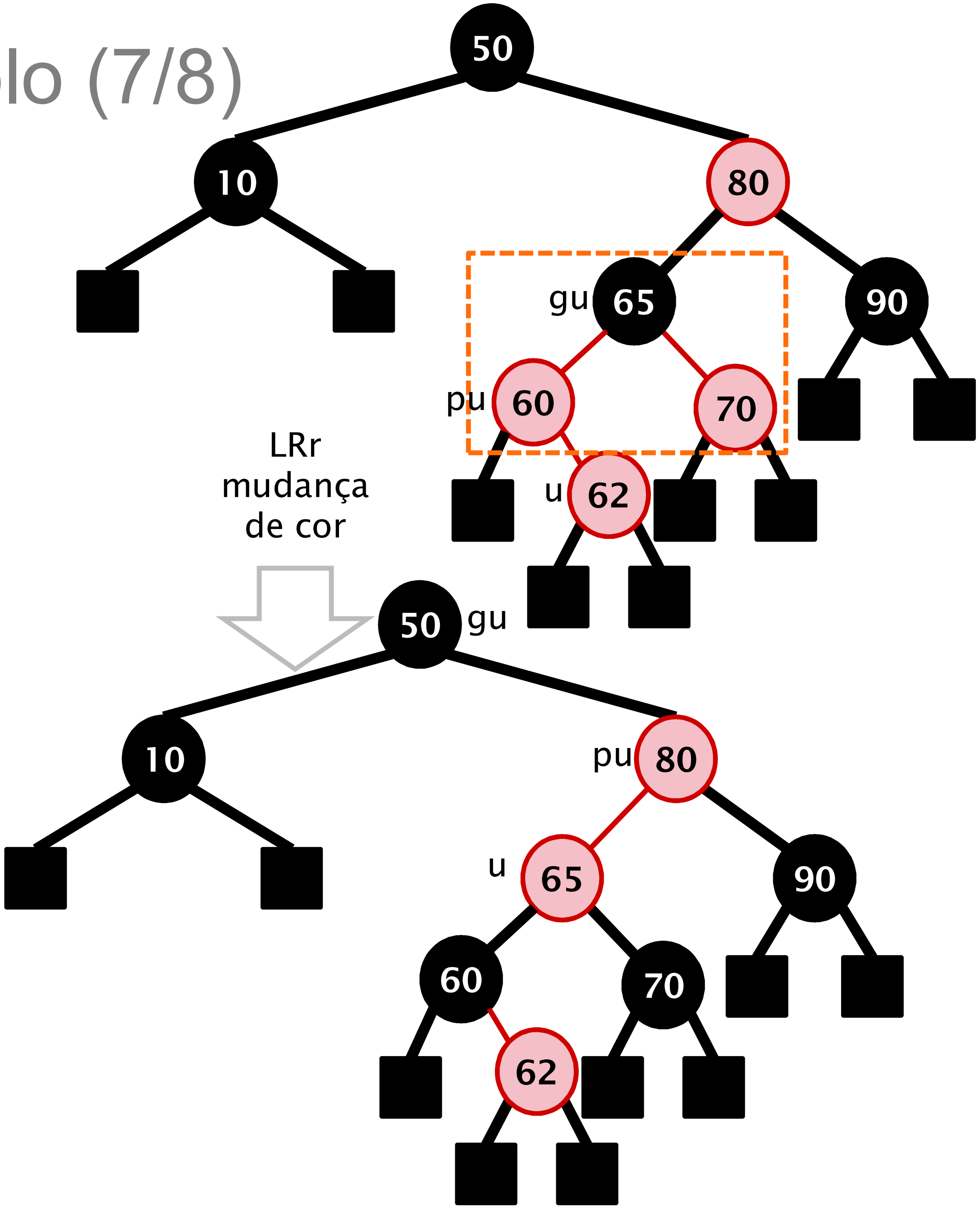
Exemplo (6/8)



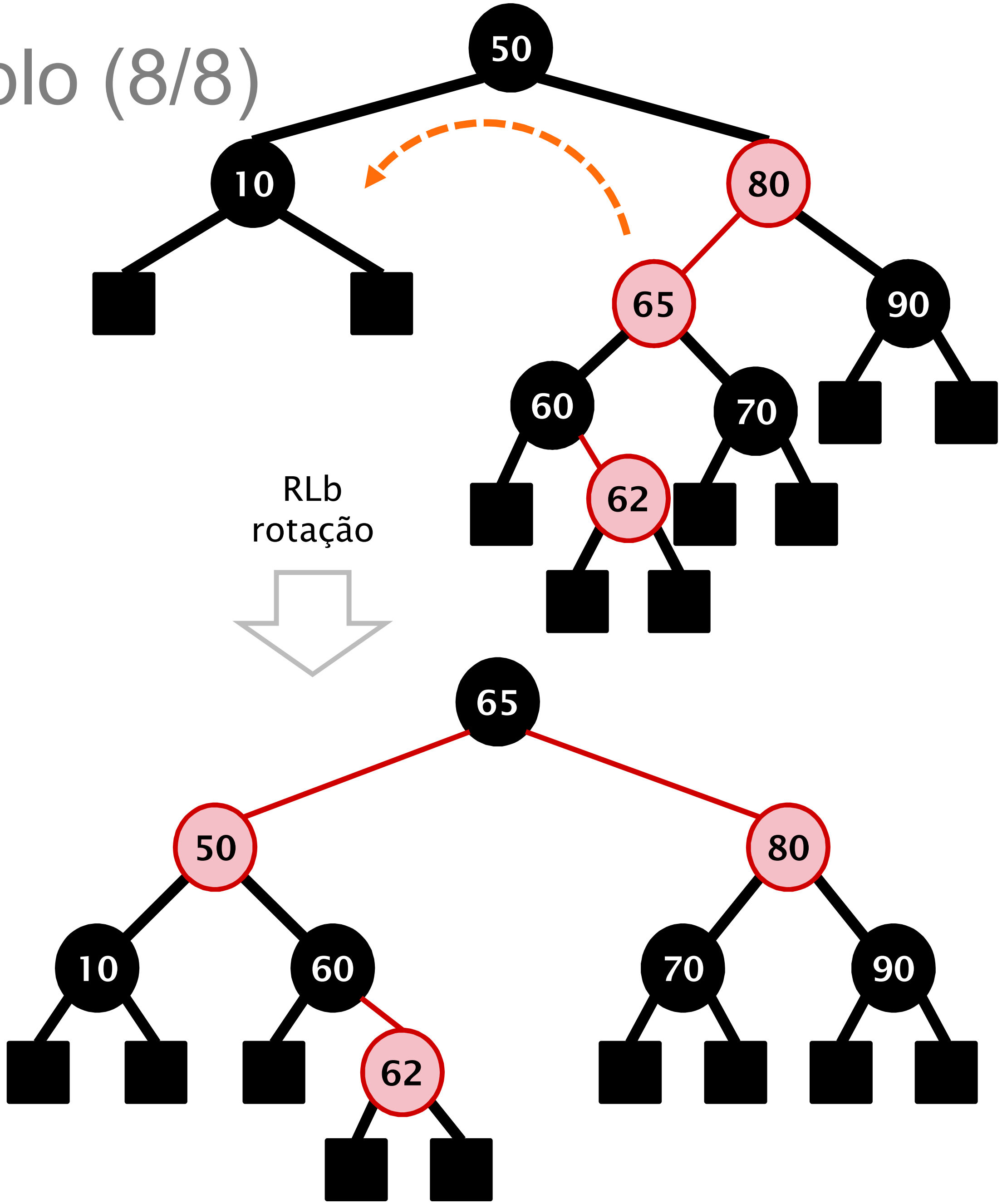
insere 62



Exemplo (7/8)


50
10
80
gu
65
90
pu
60
70
LRr
mudança
de cor
u
62
50
gu
10
pu
80
u
65
90
60
70
62

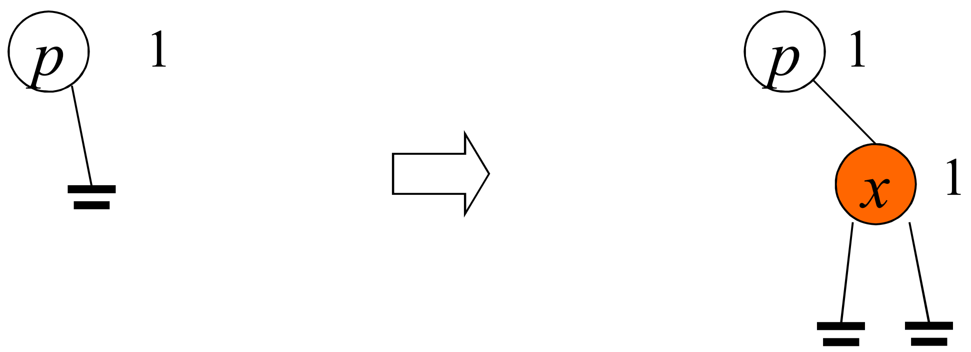
Exemplo (8/8)



Inserção em Árvore Rubro-Negra

Ao contrário da árvore AVL, agora temos agora vários critérios para ajustar simultaneamente

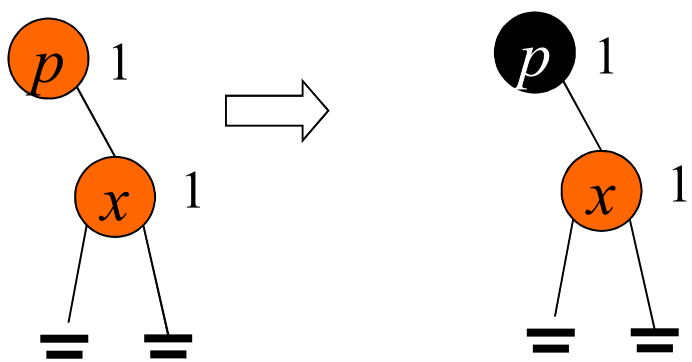
Ao inserir um nó *x* numa posição vazia da árvore (isto é, no lugar de um nó nulo) este é pintado de vermelho. Isto garante a manutenção do critério (2), já que um nó vermelho não contribui para a altura negra



Inserção em Árvore Rubro-Negra

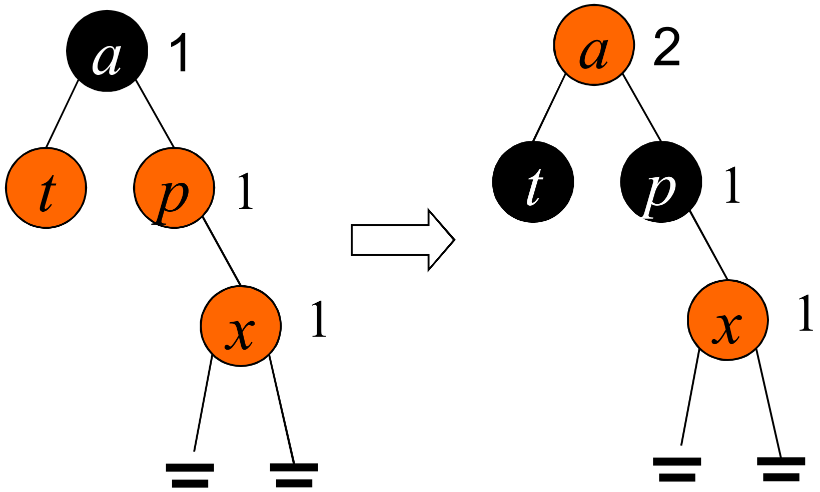
[Caso 0] Se x não tem pai ou se p, o pai de x, é negro, nada mais precisa ser feito já que o critério (3) também foi mantido

[Caso 1] Suponha agora que p é vermelho. Então, se p não tem pai, então p é a raiz da árvore e basta trocar a cor de p para negro



Inserção em Árvore Rubro-Negra

[Caso 2] Suponha agora que *p* é vermelho e *a*, o pai de *p* (e avô de *x*) é preto. Se *t*, o irmão de *p* (tio de *x*) é vermelho, ainda é possível manter o critério (3) apenas fazendo a recoloração de *a*, *t* e *p*

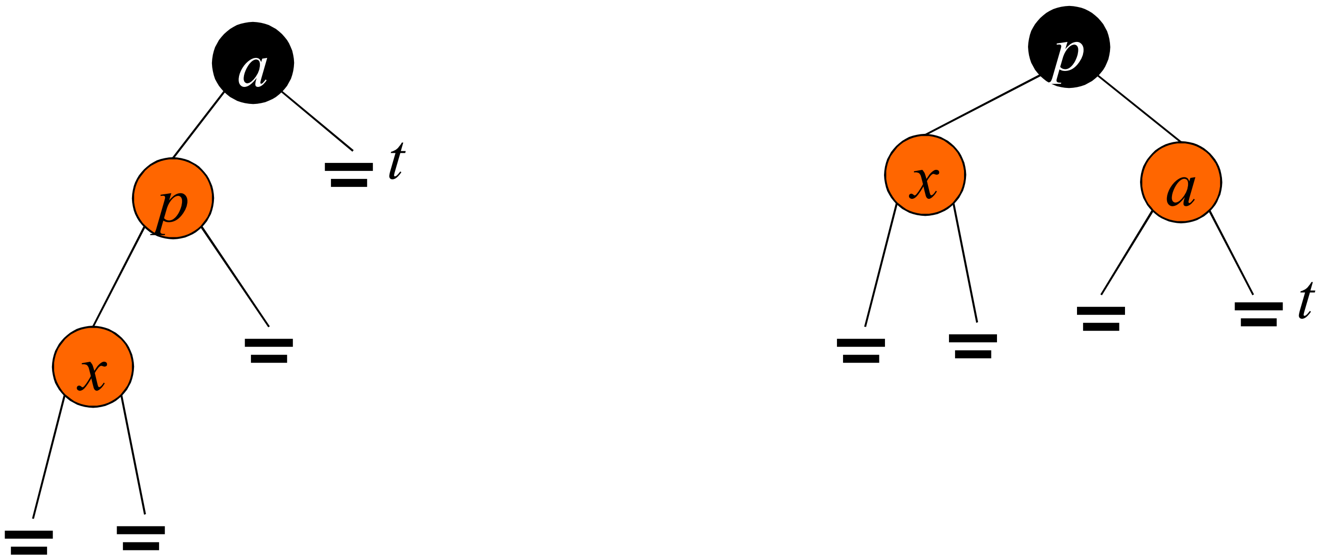


Obs.: Se o pai de *a* é vermelho, o rebalanceamento tem que ser feito novamente

Inserção em Árvore Rubro-Negra

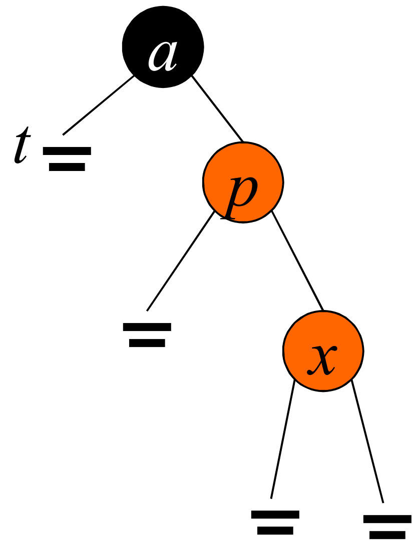
[Caso 3] Finalmente, suponha que *p* é vermelho, seu pai *a* é preto e seu irmão *t* é preto. Neste caso, para manter o critério (3) é preciso fazer rotações envolvendo *a*, *t*, *p* e *x*. Há 4 subcasos que correspondem às 4 rotações possíveis:

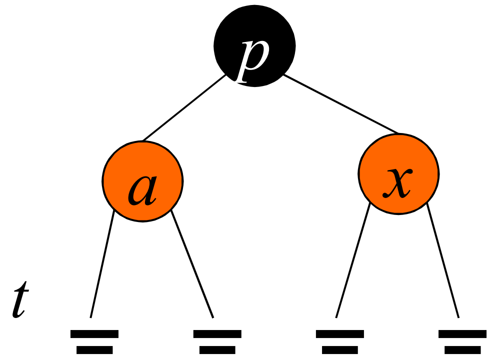
[Caso 3a] Rotação Direita



Inserção em Árvore Rubro-Negra

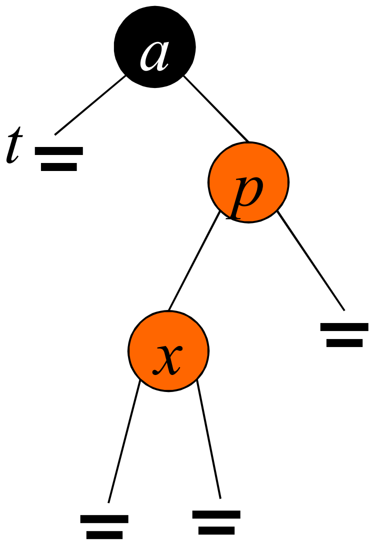
[Caso 3b] Rotação Esquerda

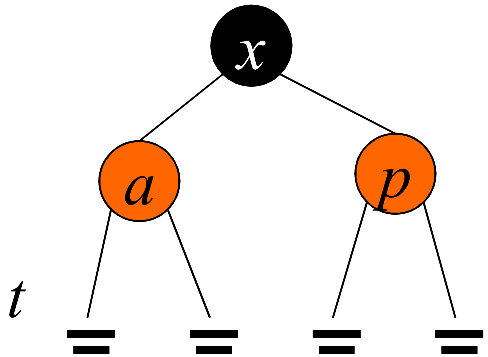




Inserção em Árvore Rubro-Negra

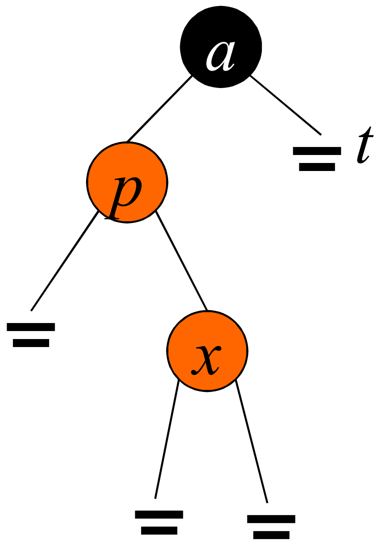
[Caso 3c] Rotação Dupla Esquerda

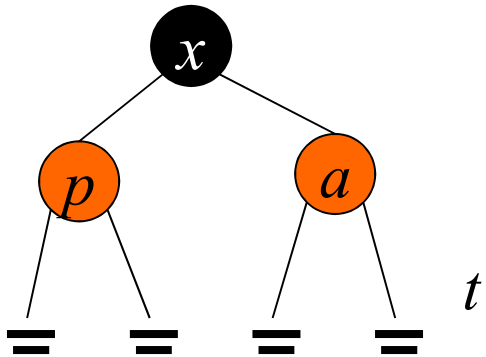




Inserção em Árvore Rubro-Negra

[Caso 3d] Rotação Dupla Direita





Complexidade da Inserção em Árvore Rubro-Negra

Rebalancear tem custo $O(1)$

Rotação têm custo $O(1)$

Inserir tem custo $O(\log n)$

dúvidas?