

# INF 1010

## Estruturas de Dados Avançadas

Árvores AVL



# voltando às árvores binárias de busca

- problemas com desbalanceamento!

diversas propostas de estruturas que se mantêm balanceadas



# Balanceamento de Árvores Binárias de Busca

**motivação:**

busca em  $O(\log(n))$

**estratégia:**

diminuir a diferença de altura entre  
a sub-árvore à esquerda e a sub-árvore à  
direita de cada nó



# Balanceamento de Árvores Binárias de Busca

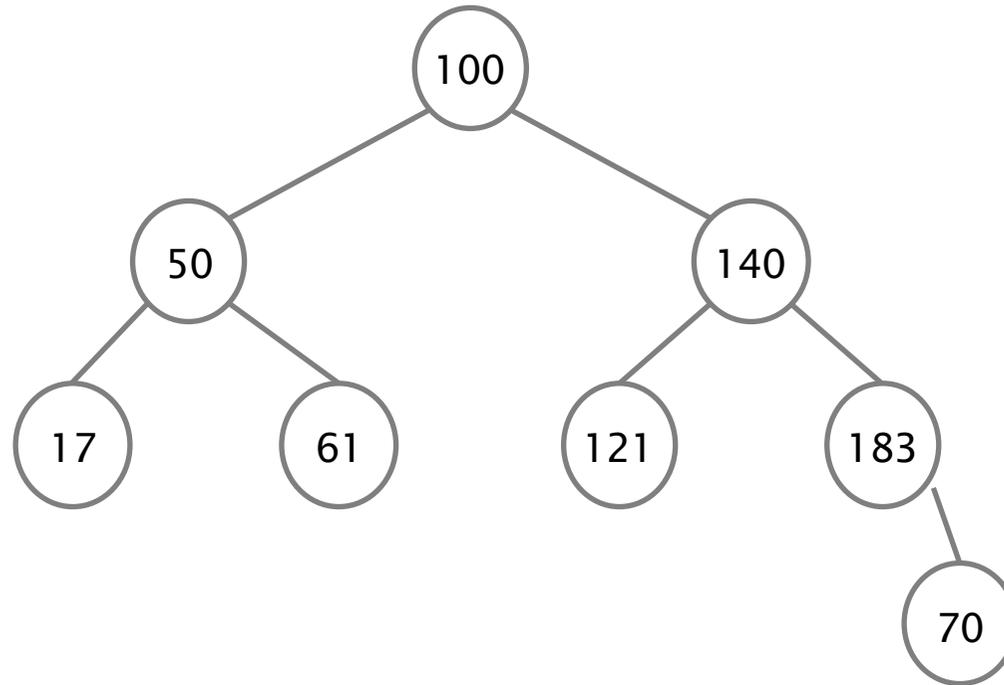


1962: AVLs  
Adelson-Velsky e Landis

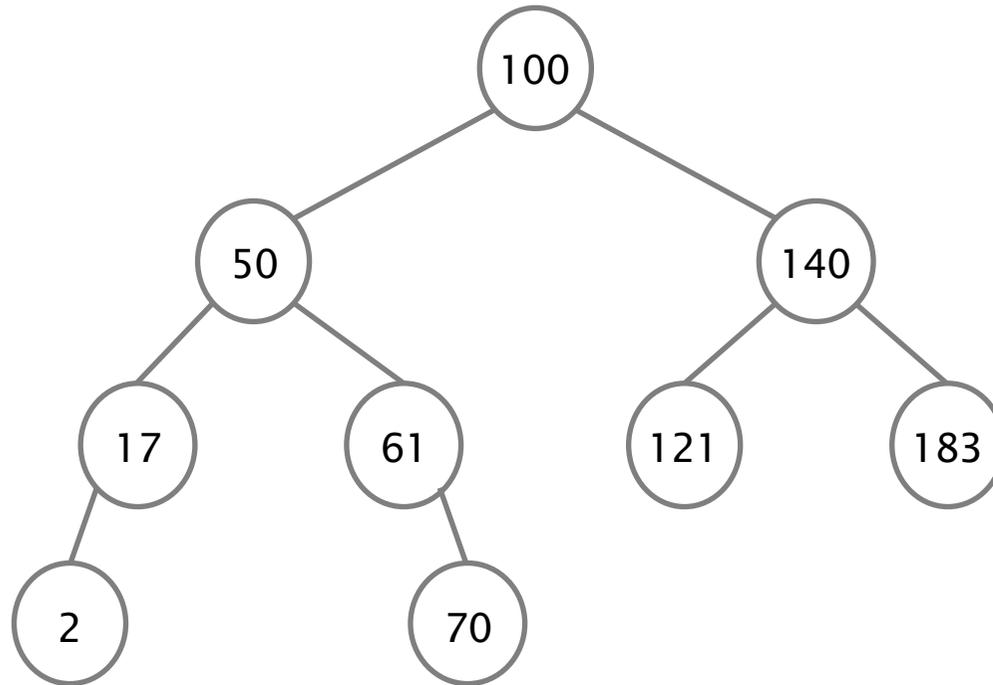
árvore binária de busca onde, em cada nó, a altura de sua sub-árvore à direita difere da altura da sub-árvore à esquerda de no máximo 1 unidade



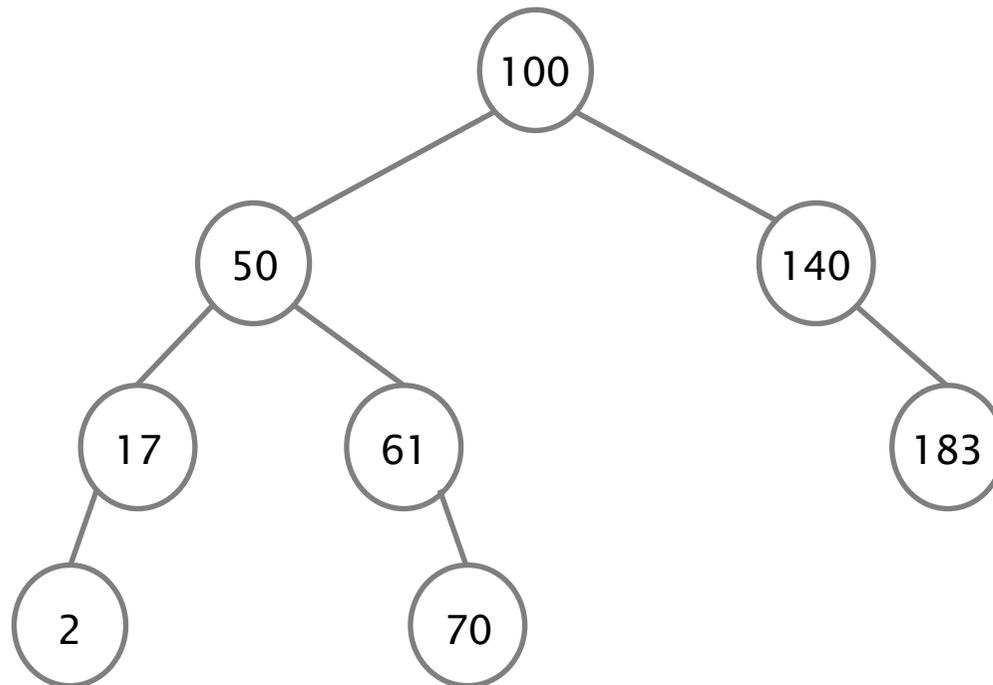
# exemplos: árvore AVL



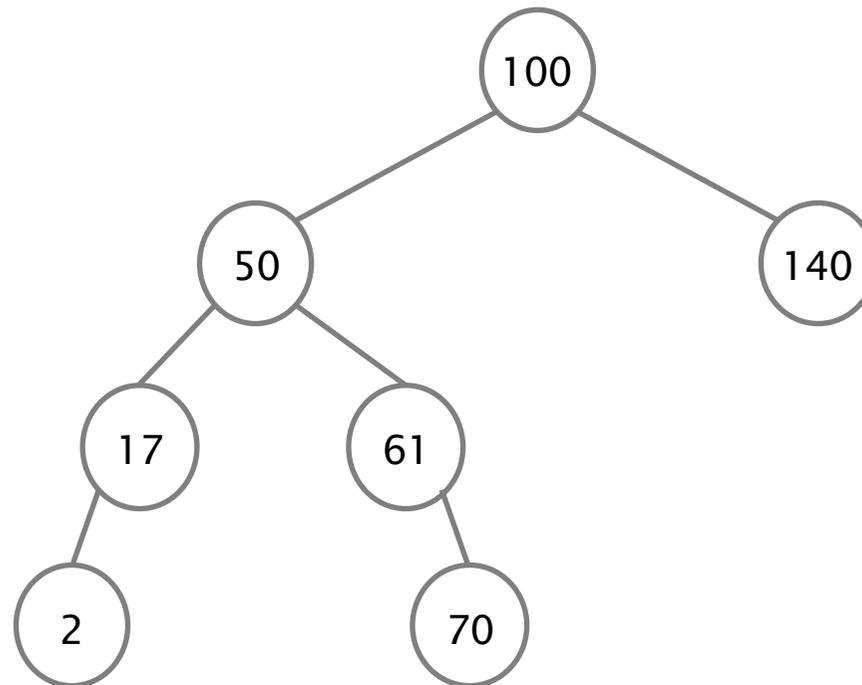
# exemplos: árvore AVL



# exemplos: árvore AVL



exemplos: árvore **não** AVL



# Fator de balanceamento

$$fb = hd - he$$

onde he = altura da sub-árvore à esquerda

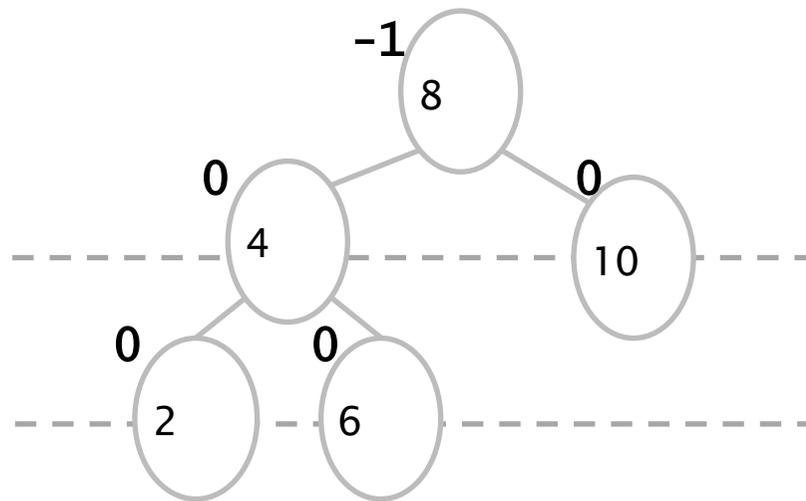
hd = altura da sub-árvore à direita

```
struct _avl {  
    int chave;  
    int fb; /*fator de balanceamento*/  
    struct _avl *esq;  
    struct _avl *dir;  
    tdados *dados;  
};
```



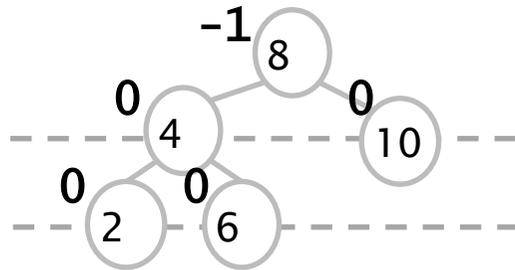
# Árvore AVL – desequilíbrio (caso 1a)

árvore balanceada (equilibrada)

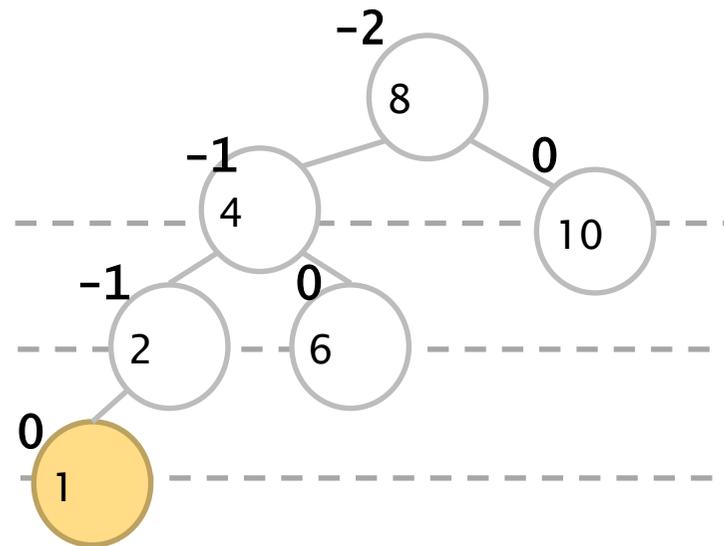


# Árvore AVL – desequilíbrio (caso 1a)

árvore balanceada (equilibrada)



desequilíbrio após inserir 1



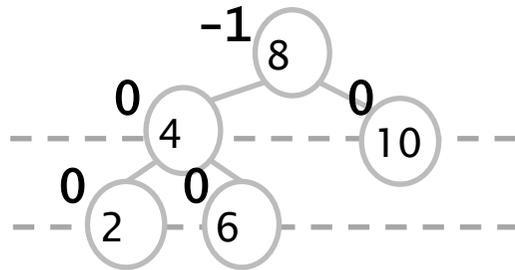
# Árvore AVL – esboço inserção

```
Avl* avl_inserere (Avl* r,int* cresceu) {
    if (r==NULL) {
        r = (Avl*) malloc(sizeof(Avl));
        r->esq = r->dir = NULL;
        r->chave = chave;
        r->fb = 0;
    }
    else if (r->chave > chave) {
        r->esq = avl_inserere (r->esq, chave, cresceu);
        verifica se precisa consertar
    }
    else if (r->chave < chave) {
        r->dir = avl_inserere (r->dir, chave, cresceu);
        verifica se precisa consertar
    }
    return r;
}
```

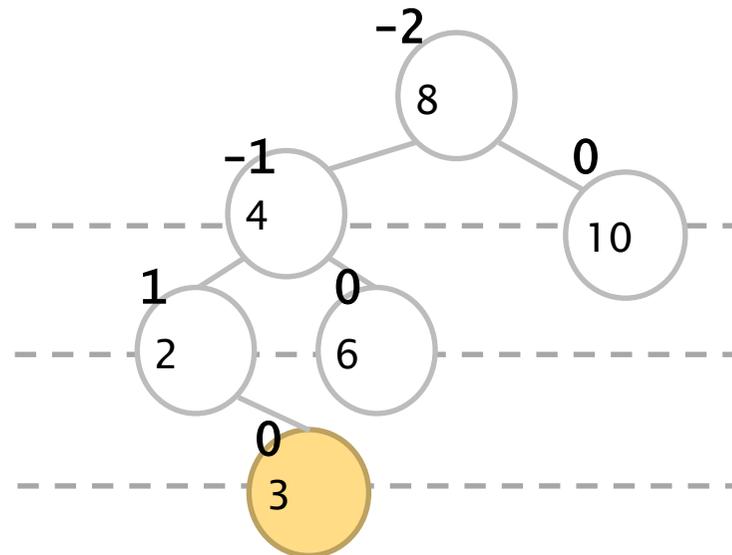


# Árvore AVL – desequilíbrio (caso 1a)

árvore balanceada (equilibrada)

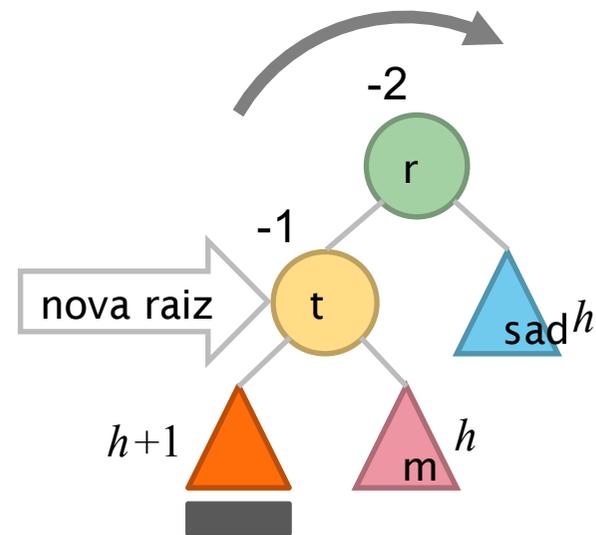
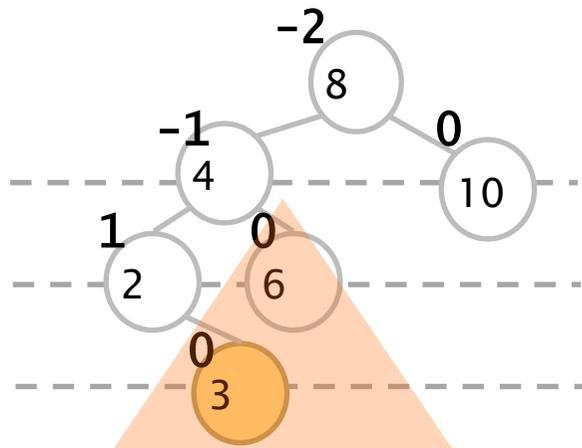


desequilíbrio após inserir 3

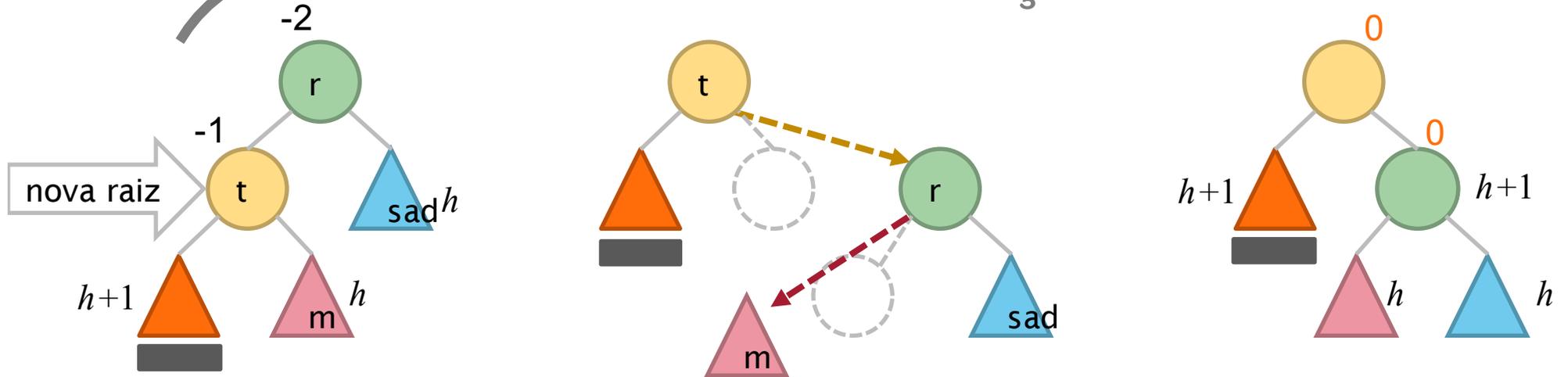


# Árvore AVL – Caso 1a: rotação à direita

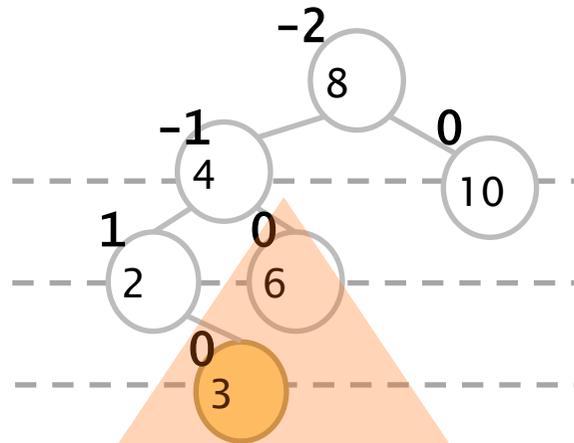
desequilíbrio após inserir 3



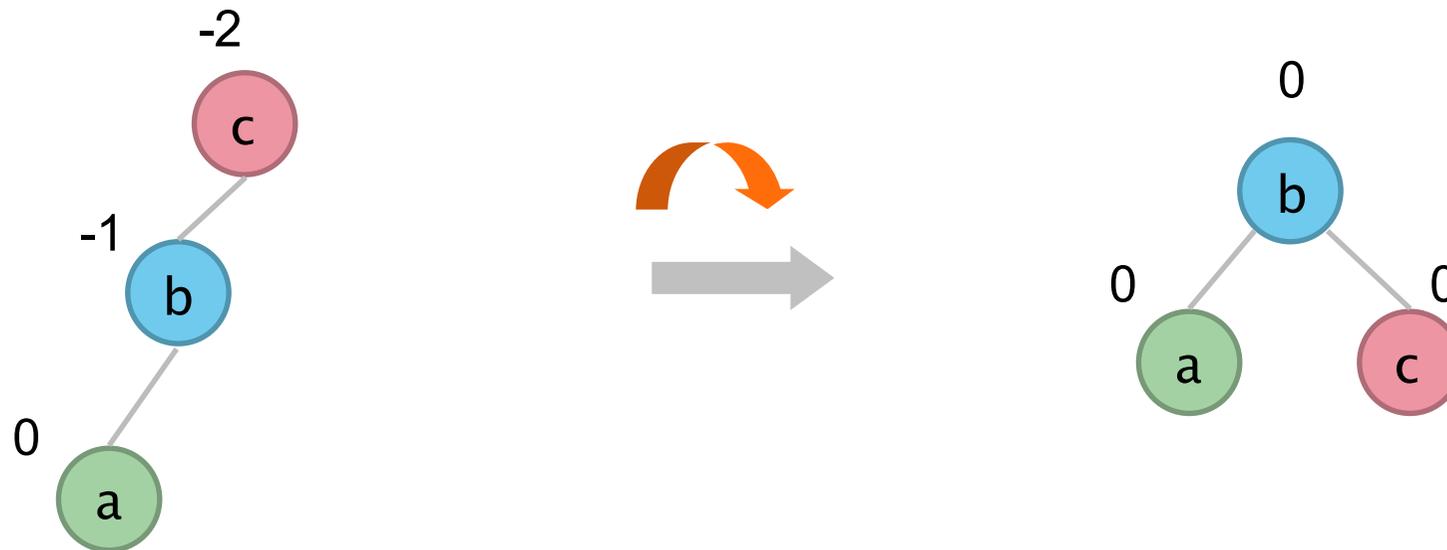
# Árvore AVL – Caso 1a: rotação à direita



desequilíbrio após inserir 3



# Caso 1a - Exemplo de rotação à direita

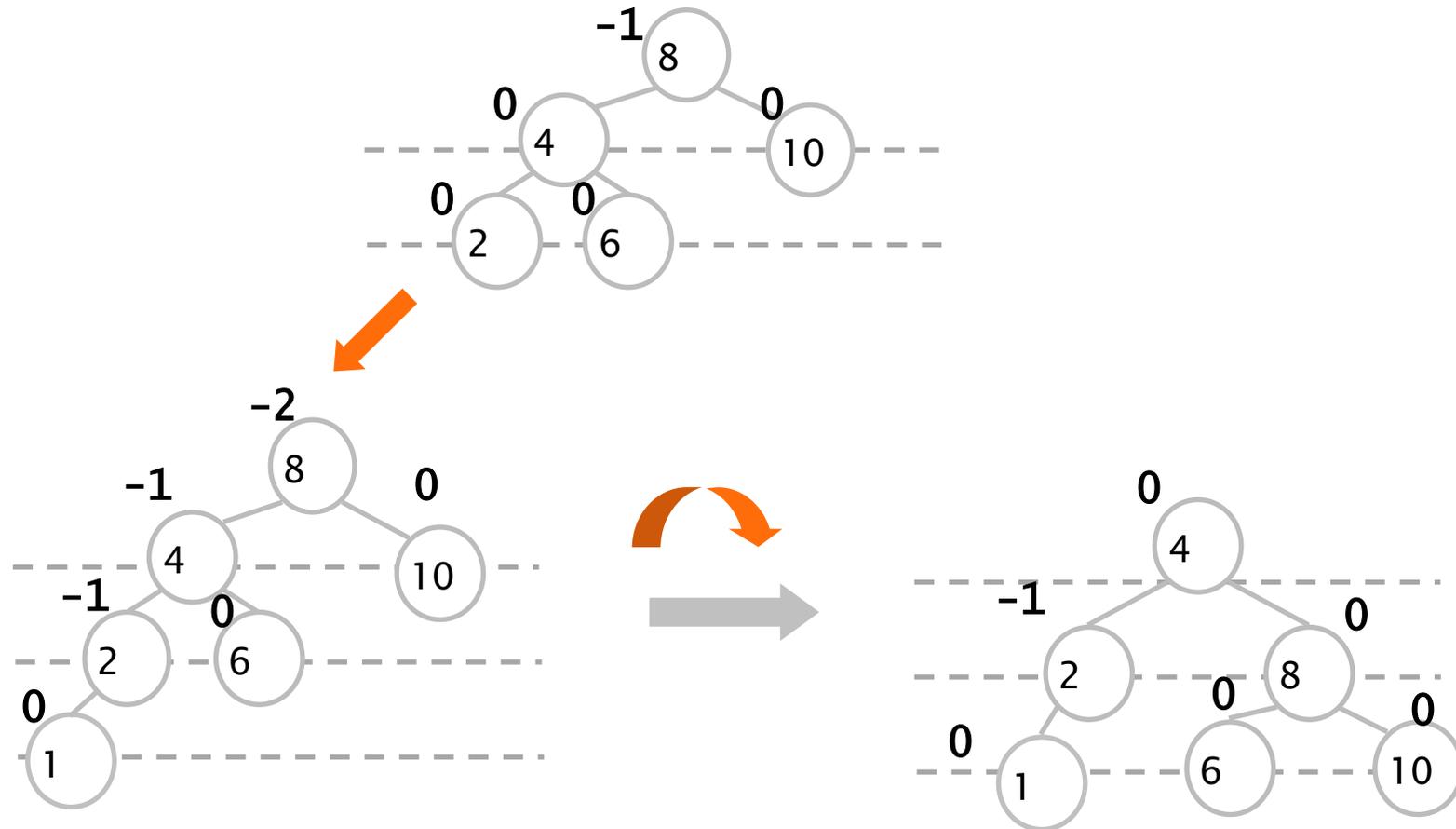


*Uma rotação à direita* ✓



# Caso 1a - Exemplo de rotação à direita

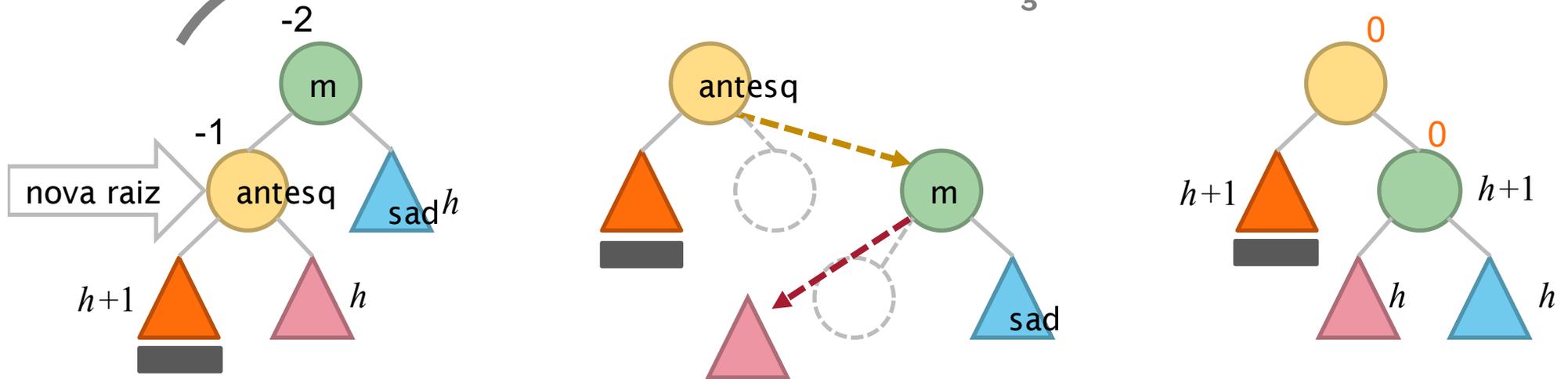
árvore balanceada (equilibrada)



*Uma rotação à direita* ✓



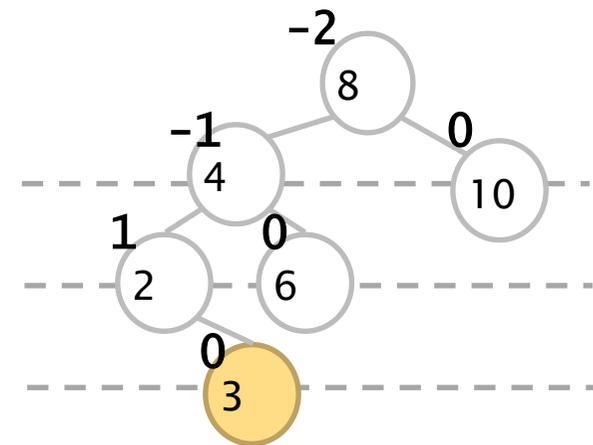
# Árvore AVL – Caso 1a: rotação à direita



```

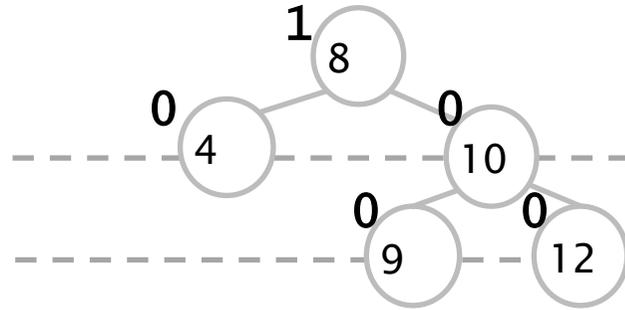
Avl* rotacao_direita(Avl *m) {
    Avl *antesq = m->esq;
    m->esq = antesq->dir;
    antesq->dir = m;
    return antesq;
}
    
```

desequilíbrio após inserir 3

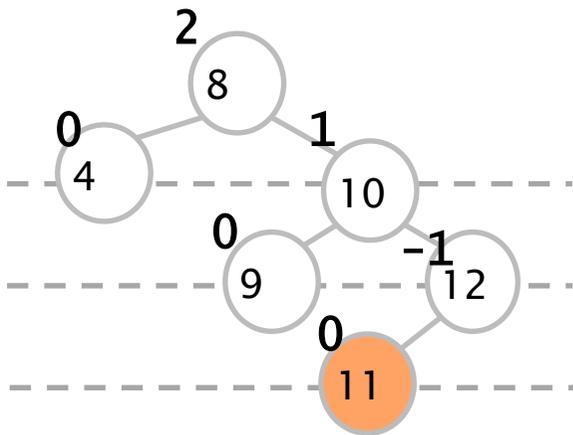


# Árvore AVL – desequilíbrio (caso 1b)

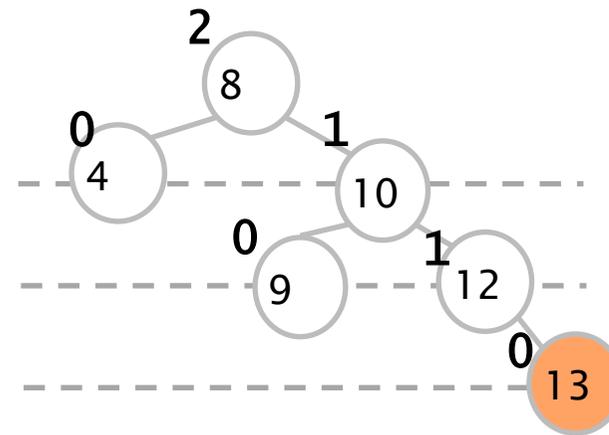
árvore balanceada (equilibrada)



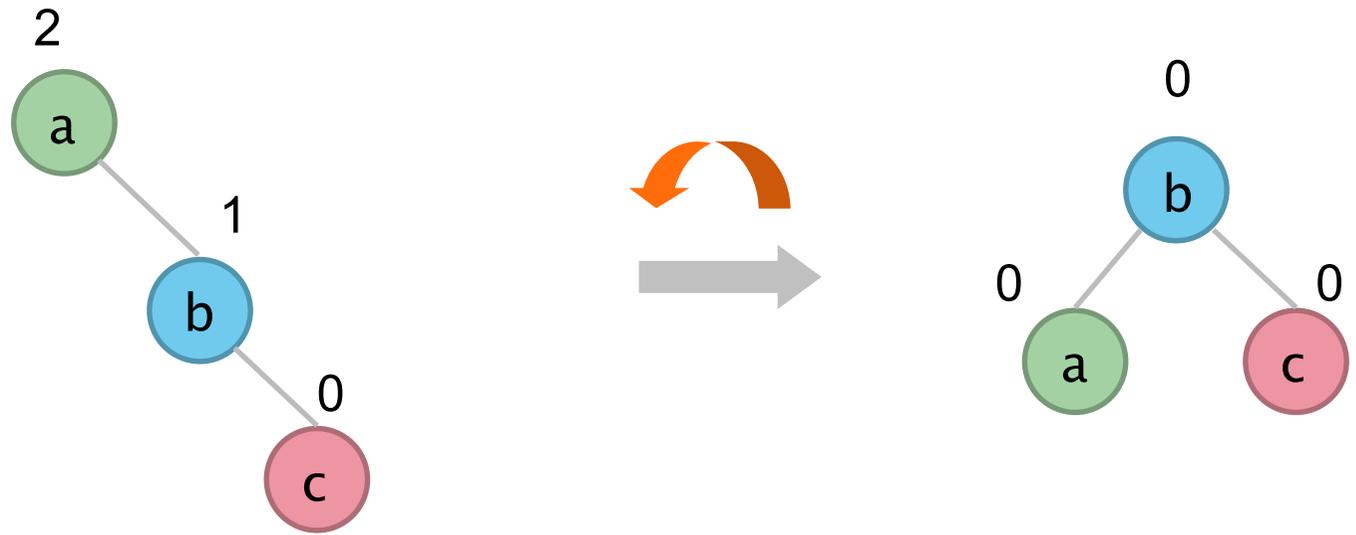
desequilíbrio após inserir 11



desequilíbrio após inserir 13



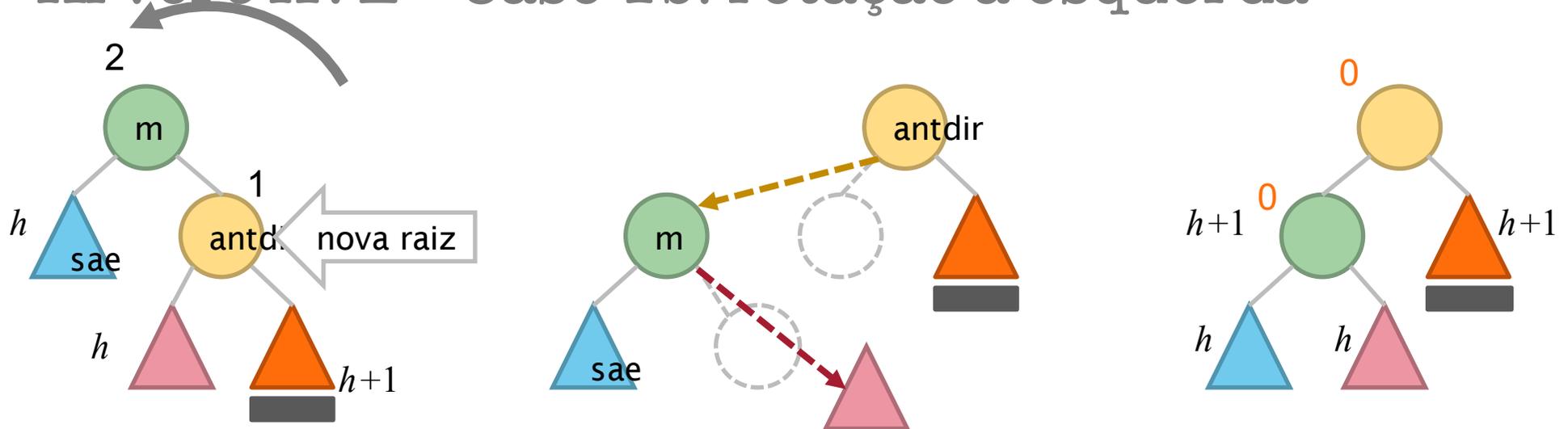
# Caso 1b - Exemplo de rotação à esquerda



uma rotação à esquerda ✓



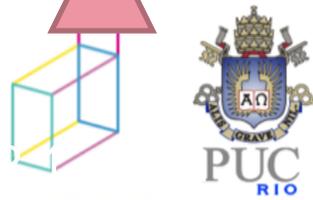
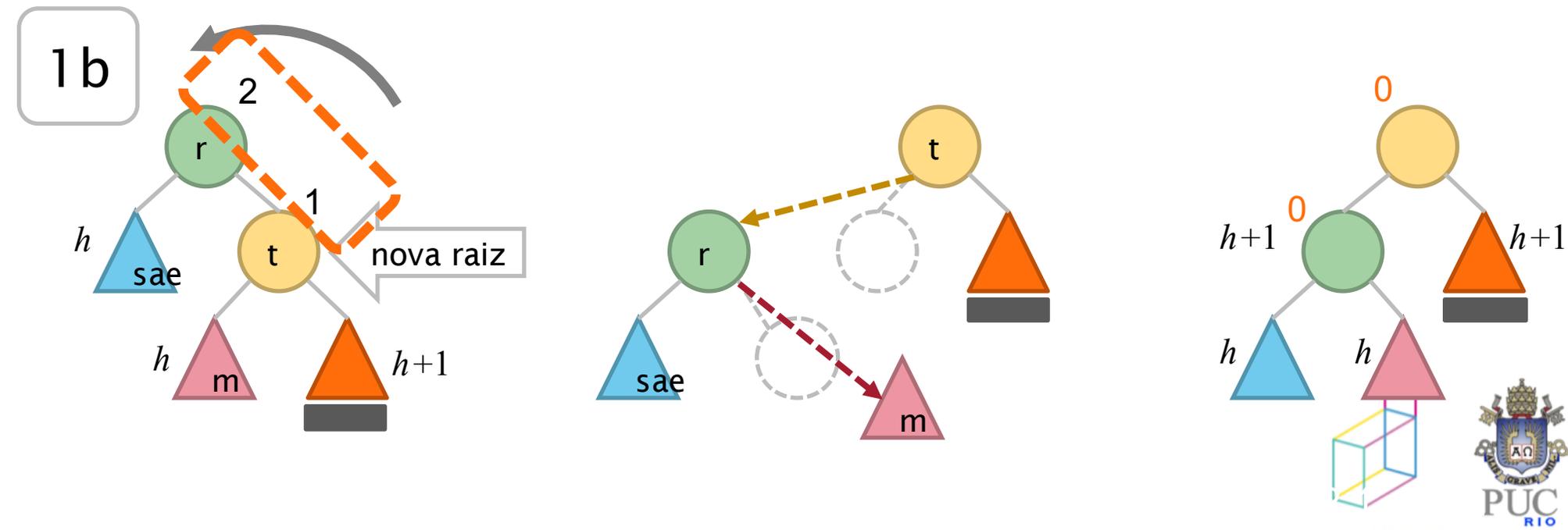
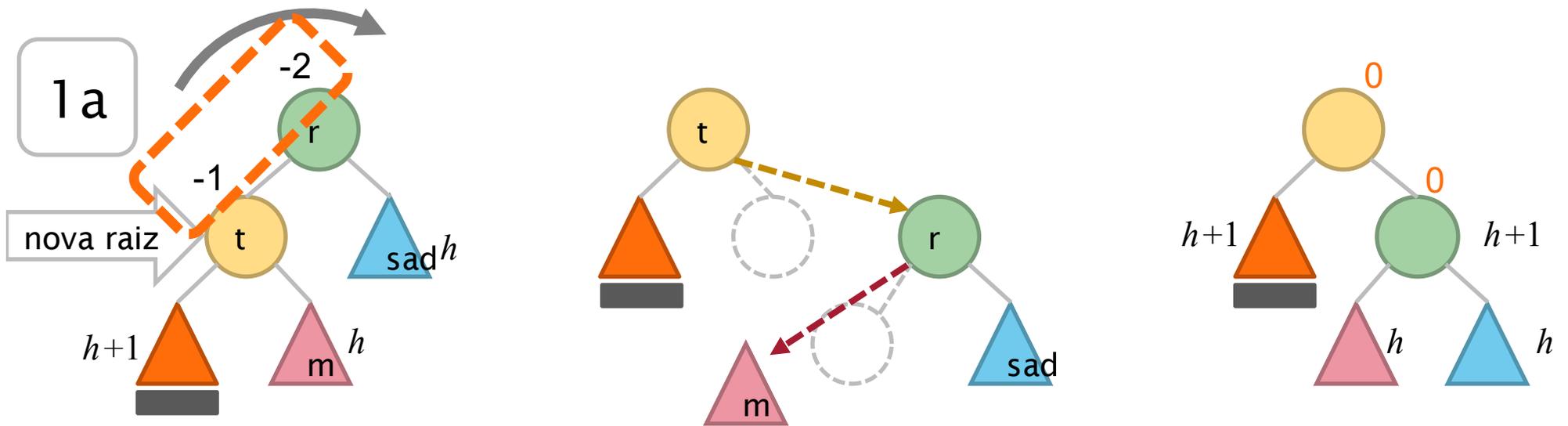
# Árvore AVL – Caso 1b: rotação à esquerda



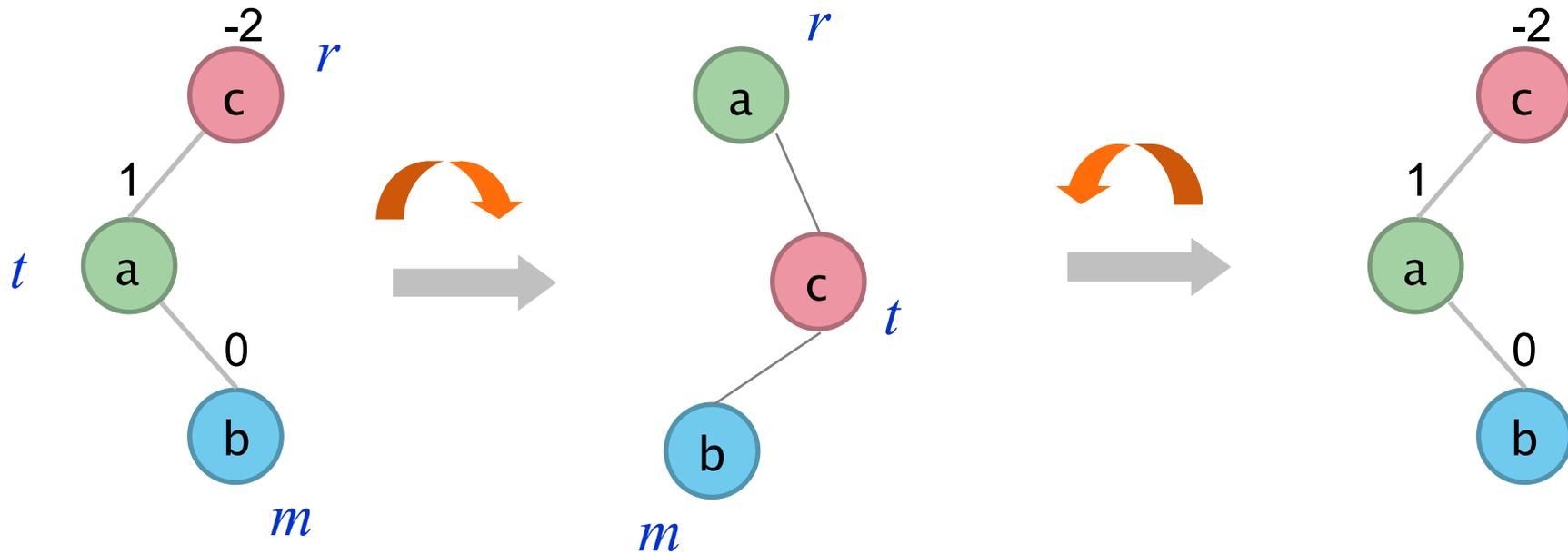
```
static Avl* rotacao_esquerda(Avl *m) {  
    /* falta testar se esses ponteiros existem */  
    Avl *antdir=m->dir;  
    antdir->esq = m;  
    m->dir = antdir->esq;  
    return antdir;  
}
```



# Árvore AVL - Casos simples: fb com **mesmo sinal**



Exemplo: caso em que rotação simples não resolve

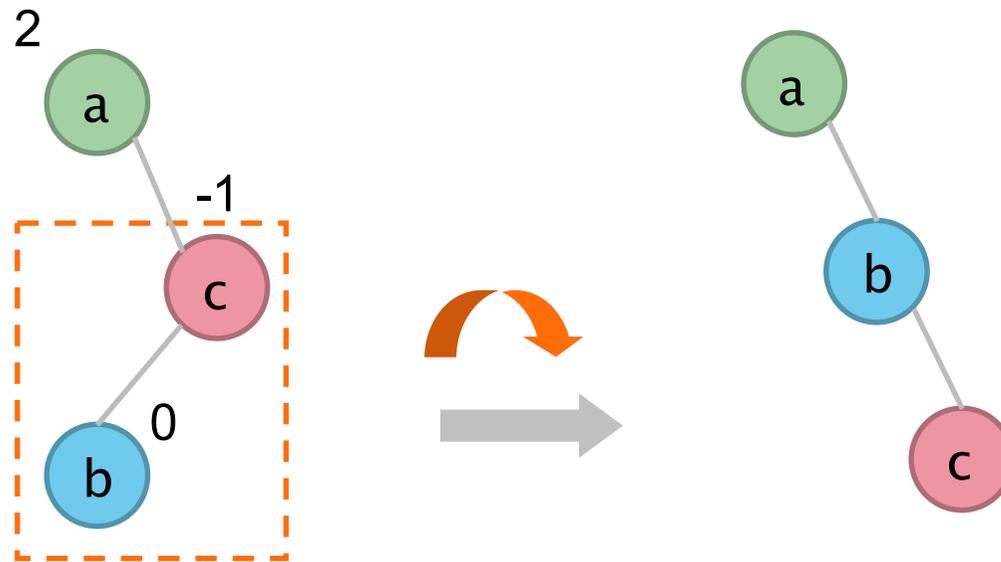


*Não resolve!*

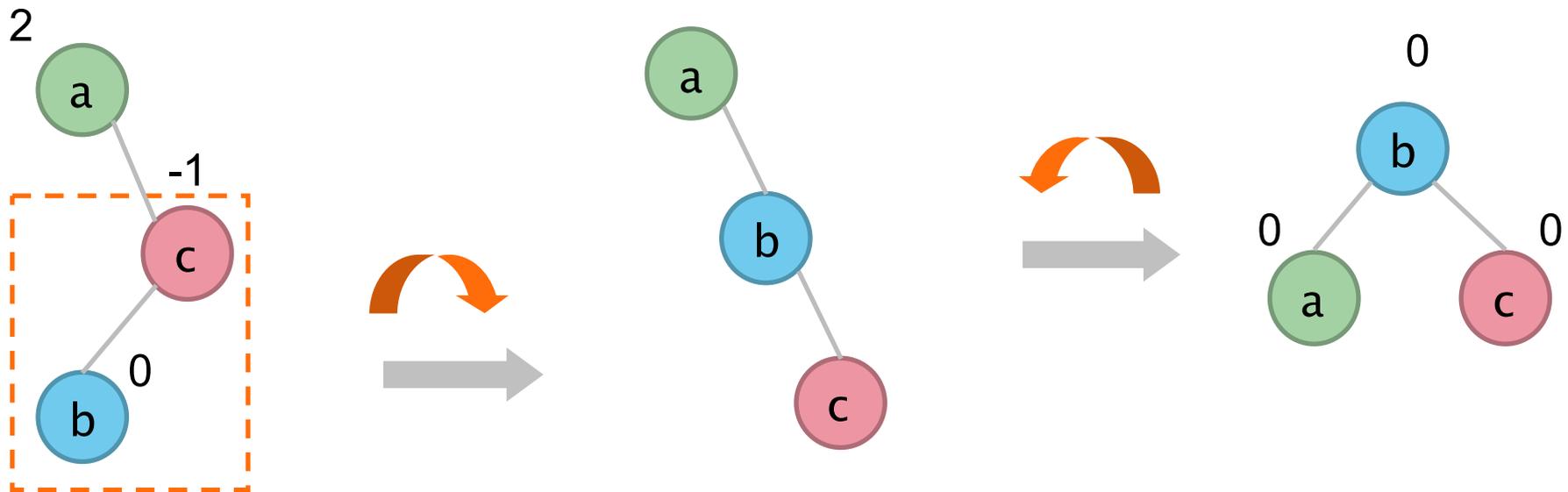


## Caso 2a - Balanceamento com rotação dupla

fazemos uma rotação à **direita** na sub-árvore à **direita**...



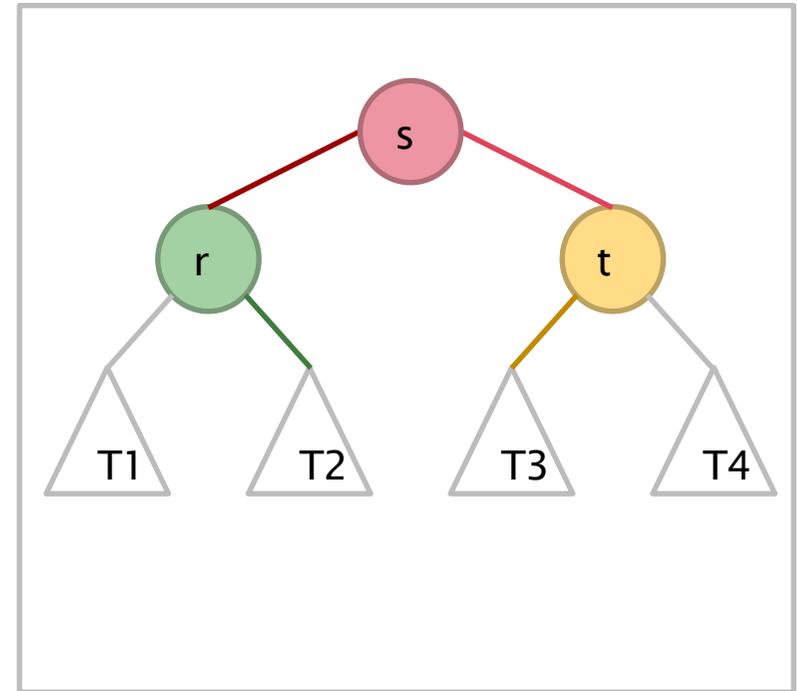
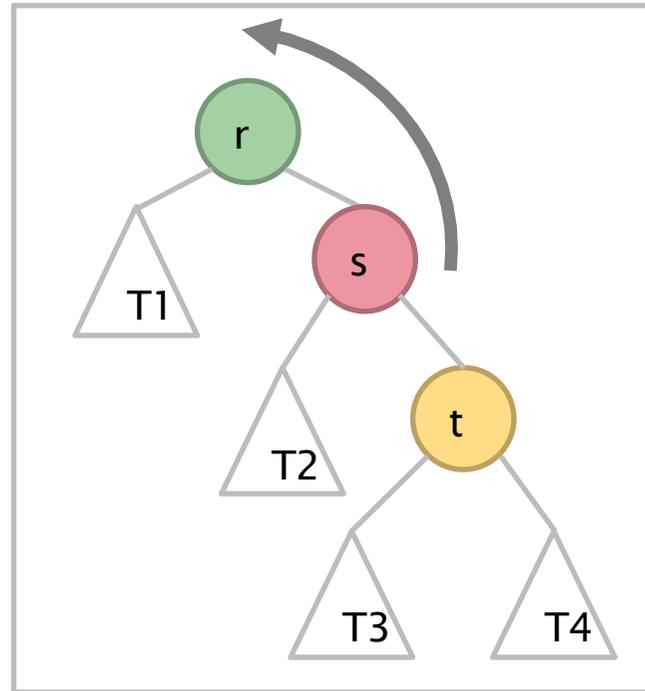
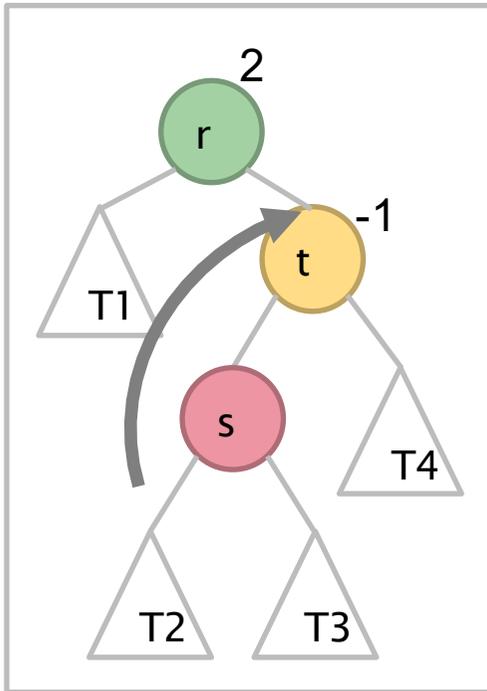
## Caso 2a - Balanceamento com rotação dupla



...seguida de uma rotação à esquerda

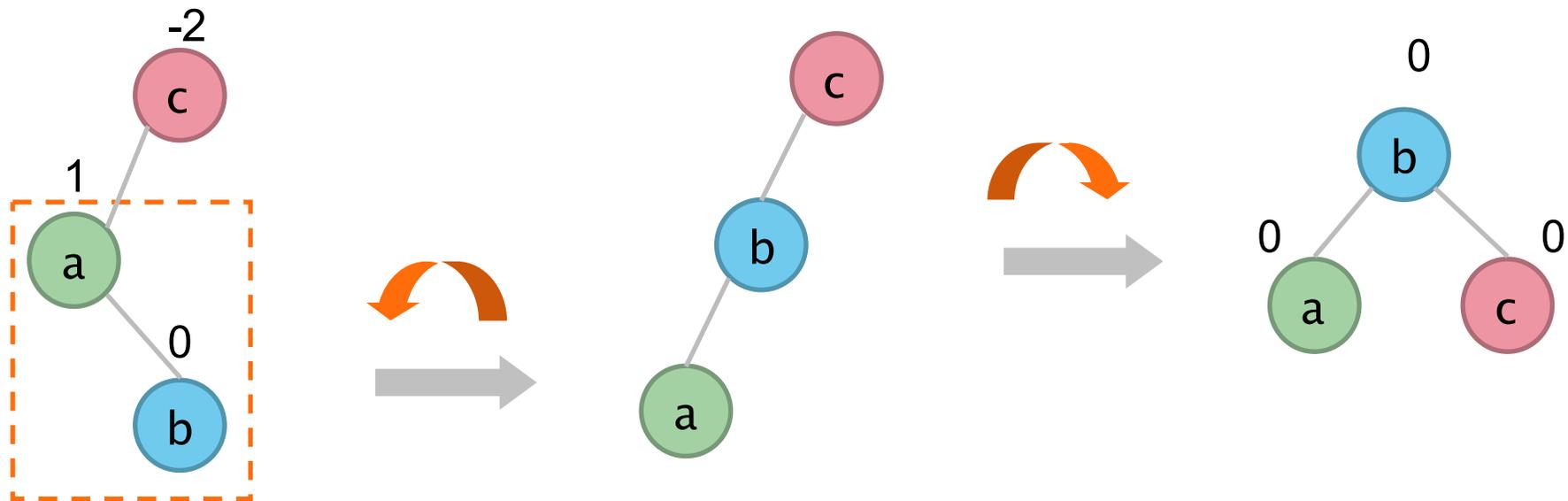


# Caso 2a – Rotação direita-esquerda



## Caso 2b - Balanceamento com rotação dupla

Uma rotação à **esquerda** na sub-árvore à **esquerda**...



...seguida de uma rotação à **direita**

