SCC0502 – Algoritmos e Estruturas de Dados I

Árvore (AVL)

Prof. Dr. Renato Moraes Silva

renatoms@icmc.usp.br

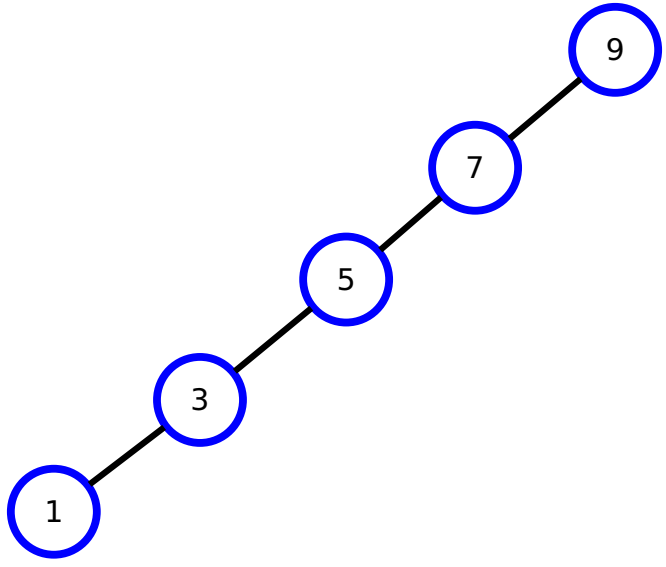
ICMC USP SÃO CARLOS

Objetivos de aprendizagem

- Entender o problema da árvore binária de busca
- Entender como implementar uma árvore AVL

Problema da inserção em árvores binárias de busca

- No melhor caso, a complexidade da busca, inserção e remoção de um elemento em uma árvore de busca binária é $O(\log n)$
- Problema:
  - Suponha que os elementos serão inseridos na seguinte ordem: 9, 7, 5, 3, 1. A árvore gerada será a mostrada abaixo.



Problema da inserção em árvores binárias de busca

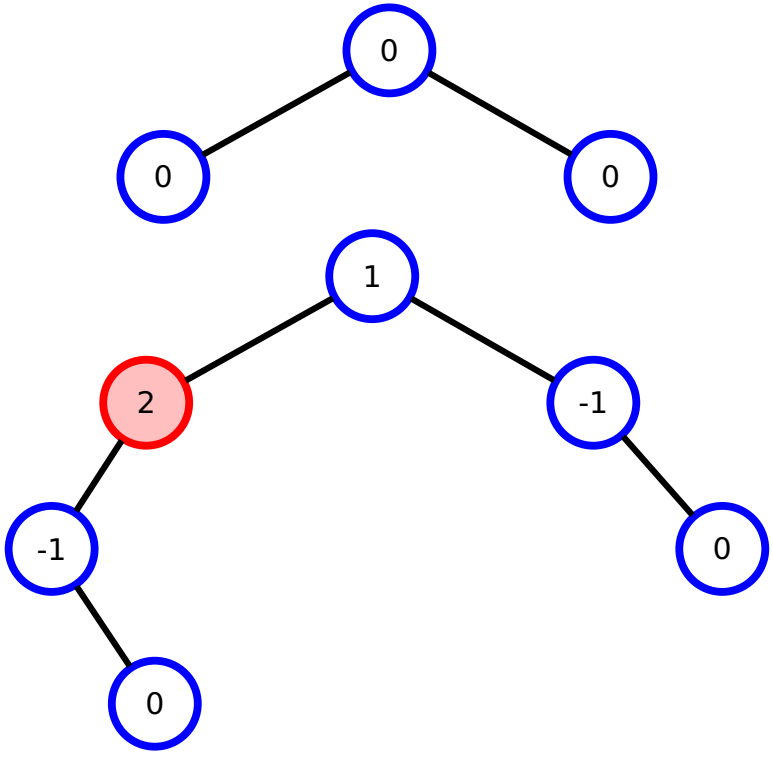
- A busca nesse caso (pior caso) é tão ineficiente quanto em uma lista ligada: O(n).

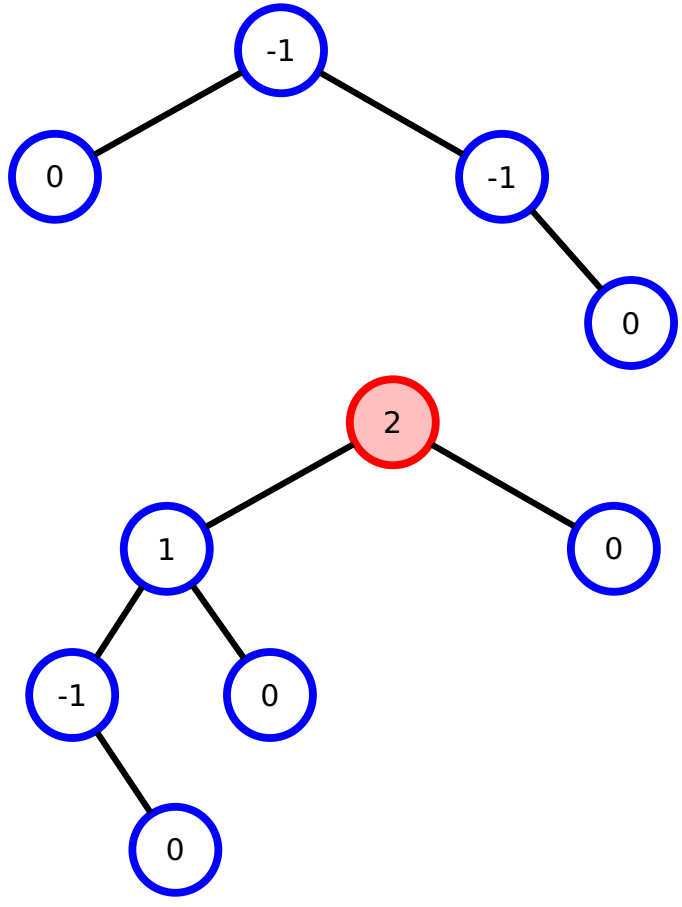
Árvore AVL

- AVL: proposta em 1962 por Adelson-Velskii e Landis (matemáticos russos)
- Balanceamento é importante
- AVL é uma árvore de busca binária balanceada em relação à altura de suas subárvores
- Fator de Balanceamento: $h_{esq} - h_{dir} \in \{0, 1, -1\}$
  - h_{esq} é a altura da subárvore esquerda
  - h_{dir} é a altura da subárvore direita
  - O fator de balanceamento deve ser calculado para cada nó

Árvore AVL

Nos exemplos abaixo, o valor em cada nó apresenta o fator de balanceamento dele. Quais nós são AVL?





Balanceamento de árvores AVL

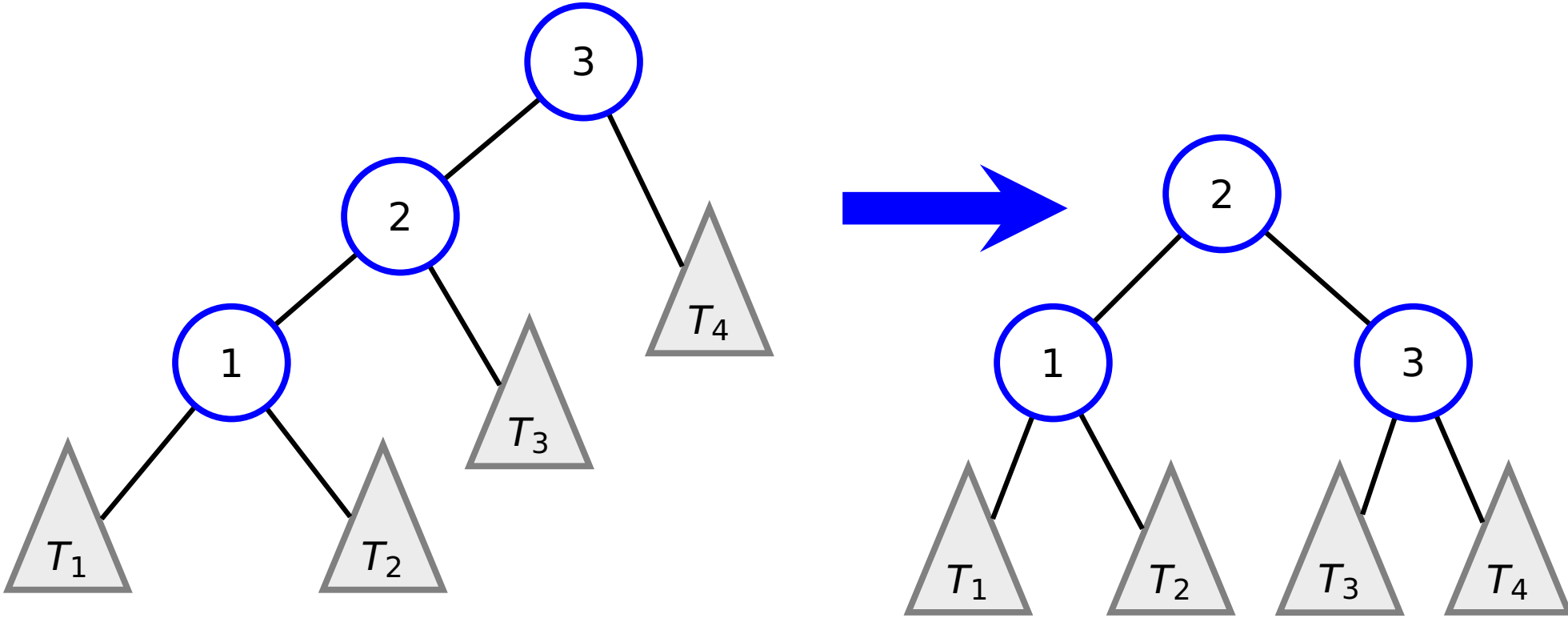
- Se uma **inserção** ou **remoção** de um nó em uma árvore AVL deixa ela desbalanceada, deve ser aplicada uma transformação, tal que:
  - A árvore continue sendo uma árvore binária de busca
  - A árvore transformada fique balanceada
- A transformação aplicada em uma árvore AVL para balanceá-la é chamada de rotação

Rotação em AVL

- Na inserção ou remoção utiliza-se um processo de balanceamento que pode ser de 2 tipos gerais:
  - Rotação simples
  - Rotação dupla
- Esses dois tipos gerais, podem ser detalhados em 4 tipos específicos:
  - Rotação à esquerda
  - Rotação à direita
  - Rotação esquerda-direita
  - Rotação direita-esquerda

Rotação à direita

- Usada se a subárvore esquerda do nó for pesada à esquerda (fator de balanceamento maior que 0)
- Todos os nós envolvidos se movem para a esquerda de sua posição atual. Portanto, o nó pai se torna o filho direito.



Rotação à direita

Algoritmo Rotação à direita

```
função rotacao_direita(raiz)

    novaRaiz = raiz.esquerda
    raiz.esquerda = novaRaiz.direita
    novaRaiz.direita = raiz
    raiz.altura = 1 + max(raiz.esquerda, raiz.direita)
    novaRaiz.altura = 1 + max(novaRaiz.esquerda, novaRaiz.direita)

    retorna novaRaiz
fim função
```

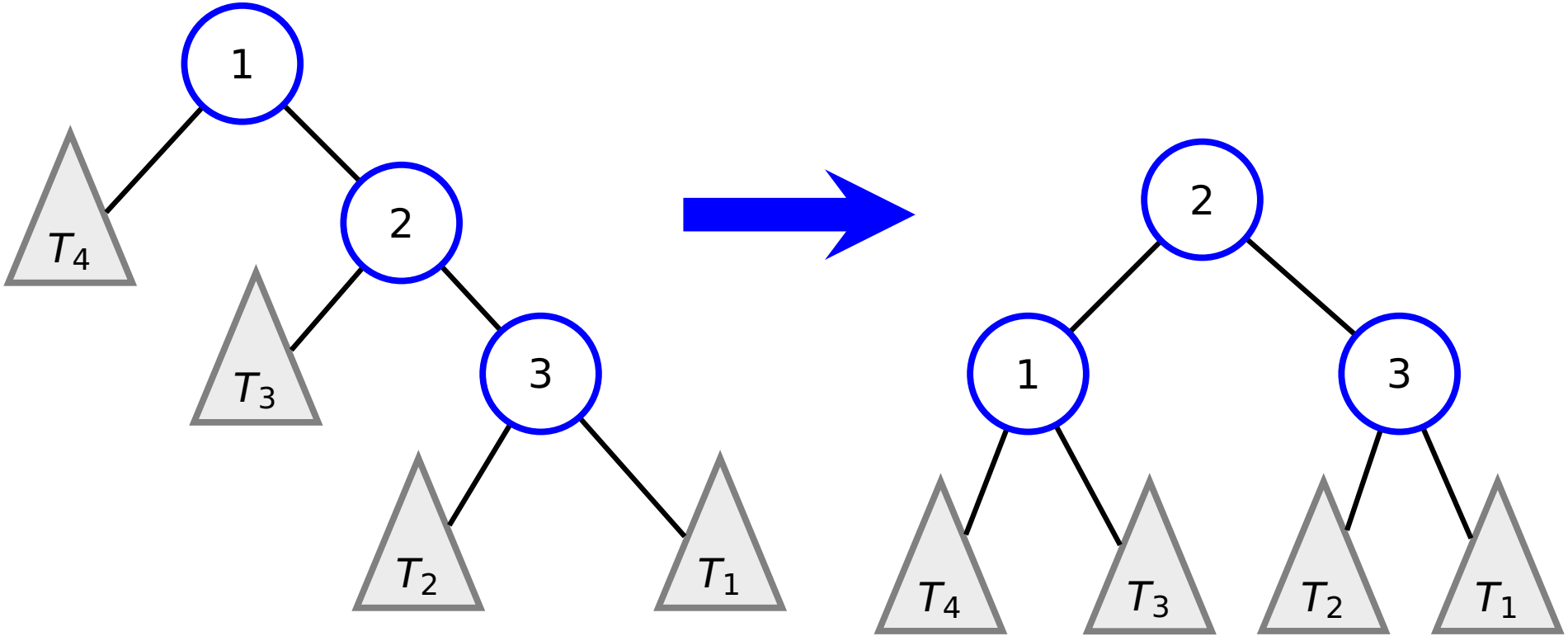
Separamos o nó à esquerda da raiz em uma nova variável

Alocamos qualquer valor que estava contido no lado direito da subárvore à esquerda da raiz, passando a ser o valor esquerdo da raiz

Passamos a raiz para o lado direito da nova raiz (a que era a subárvore esquerda da raiz)

Rotação à esquerda

- Usada se a subárvore direita do nó for pesada à direita (fator de balanceamento menor que 0)
- Todos os nós envolvidos se movem para a direita de sua posição atual. Portanto, o nó pai se torna o filho esquerdo.



Rotação à esquerda

Algoritmo Rotação à esquerda

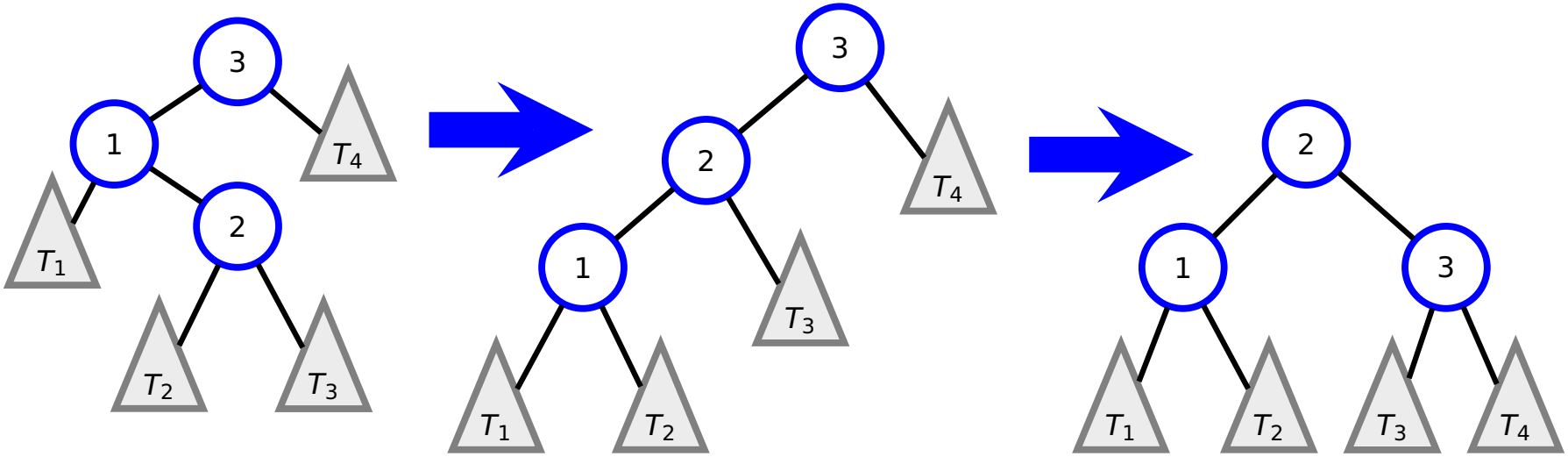
```
função rotacao_esquerda(raiz)

    novaRaiz = raiz.direita
    raiz.direita = novaRaiz.esquerda
    novaRaiz.esquerda = raiz
    raiz.altura = 1 + max(raiz.esquerda, raiz.direita)
    novaRaiz.altura = 1 + max(novaRaiz.esquerda, novaRaiz.direita)

    retorna novaRaiz
fim função
```

O valor a esquerda da subávore direita da raiz passa a ser o valor direito da raiz

A antiqa raiz passa a ser a subarvore esquerda da nova raiz

Rotação esquerda-direita

- Usada se o nó for pesado à esquerda e a subárvore esquerda for pesada à direita.
- Faz uma rotação à esquerda, seguida de uma rotação à direita. Todos os nós envolvidos se movem para a esquerda de sua posição atual e, em seguida, se movem uma posição para a direita.



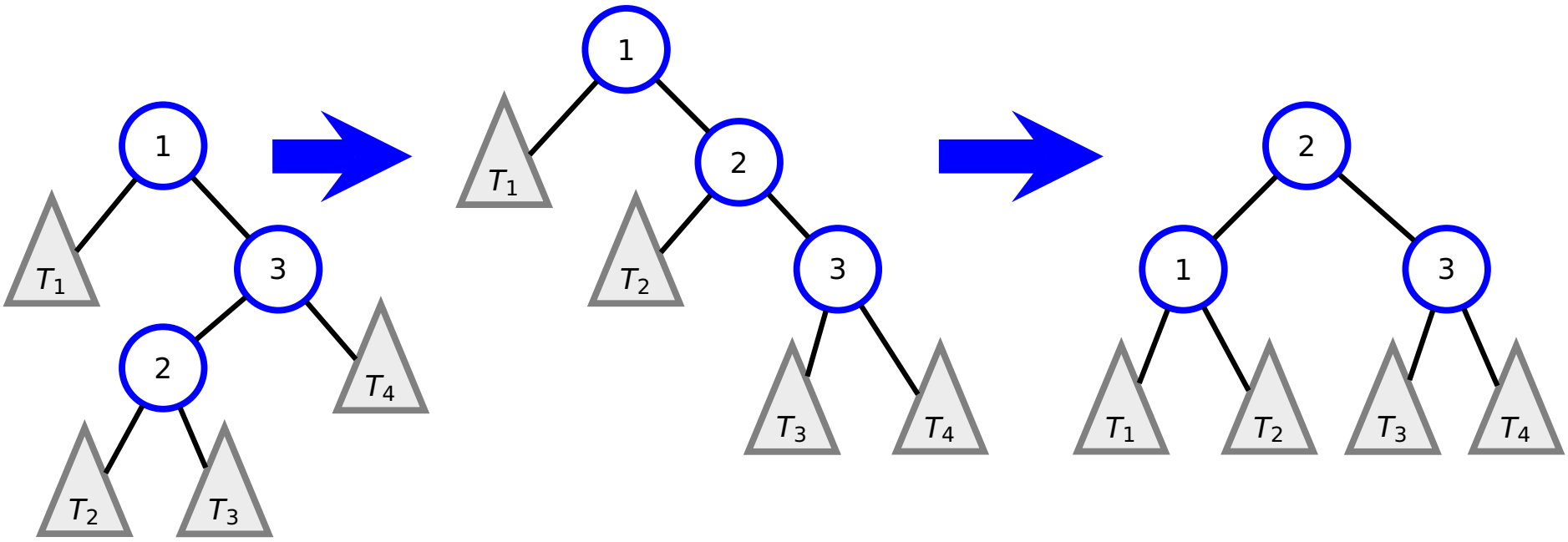
Rotação esquerda-direita

Algoritmo Rotação esquerda-direita

```
função rotacao_esquerdaDireita(raiz)

    raiz.esquerda = rotacao_esquerda(raiz.esquerda)
    novaRaiz = rotacao_direita(raiz)

    retorna novaRaiz
fim função
```

Rotação direita-esquerda

- Usada se o nó for pesado à direita e a subárvore direita for pesada à esquerda.
- Faz uma rotação à direita, seguida de uma rotação à esquerda. Todos os nós envolvidos se movem para a direita de sua posição atual e, em seguida, se movem uma posição para a esquerda.



Rotação direita-esquerda

Algoritmo Rotação direita-esquerda

```
função rotacao_direitaEsquerda(raiz)

    raiz.direita = rotacao_direita(raiz.direita)
    novaRaiz = rotacao_esquerda(raiz)

    retorna novaRaiz
fim função
```

Inserção

- Faça a inserção do novo nó da mesma maneira que em uma árvore de busca binária de busca comum
- Atualize as alturas dos nós
- Seja, b o fator de desbalanceamento da raiz da subárvore onde o novo nó foi inserido

Chave = valor do nó

  - Se $b > 1$ e chave < raiz.esquerda.chave
    - Use a rotação à direita
  - Se $b < -1$ e chave > raiz.direita.chave
    - Use a rotação à esquerda
  - Se $b > 1$ e chave > raiz.esquerda.chave
    - Use a rotação esquerda-direita
  - Se $b < -1$ e chave < raiz.direita.chave
    - Use a rotação direita-esquerda

Remoção

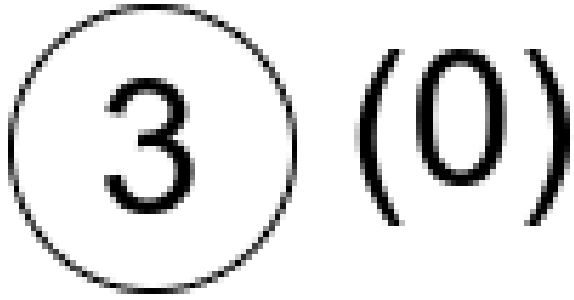
- Faça a remoção do nó desejado da mesma maneira que em uma árvore binária de busca comum
- Atualize as alturas dos nós atual e calcule seu fator de balanceamento
- Seja, *bAtual* o fator de desbalanceamento do nó atual, *bEsq* o balanceamento da subárvore da esquerda e $b - dir$ o balanceamento da subárvore da direita
  - Se $bAtual > 1$ e $bEsq >= 0$
    - Use a rotação à direita
  - Se $bAtual > 1$ e $bEsq < 0$
    - Use a rotação à esquerda-direita
  - Se $bAtual < -1$ e $bDir <= 0$
    - Use a rotação à esquerda
  - Se $bAtual < -1$ e $bDir > 0$
    - Use a rotação direita-esquerda

Ex. 01

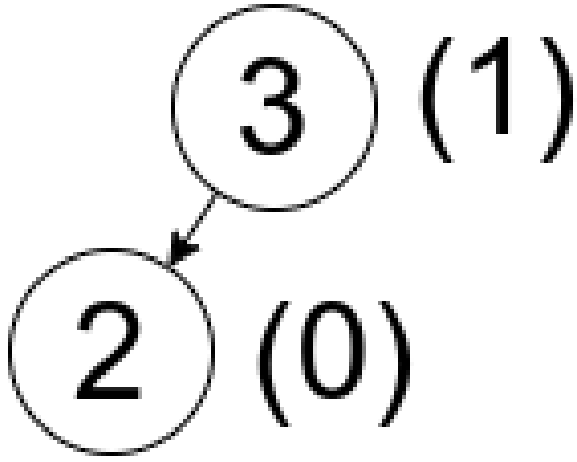
Mostrar as rotações necessárias para a construção da seguinte árvore AVL: 3, 2, 1, 4, 5, 6, 7, 16, 15.

Inserir 3

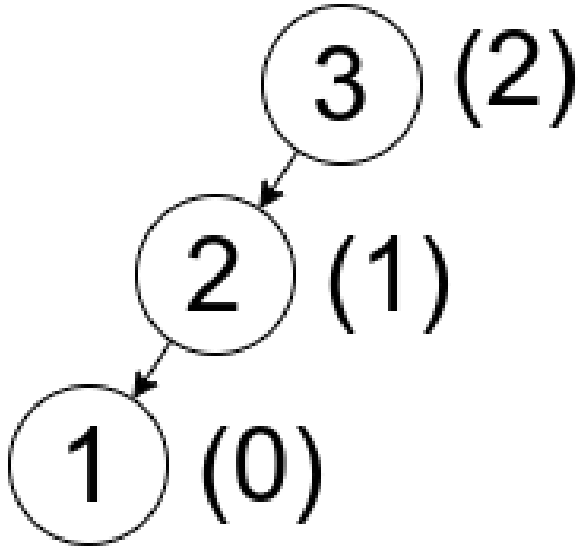
- Na figura, ao lado do nó, está sendo apresentado o balanceamento



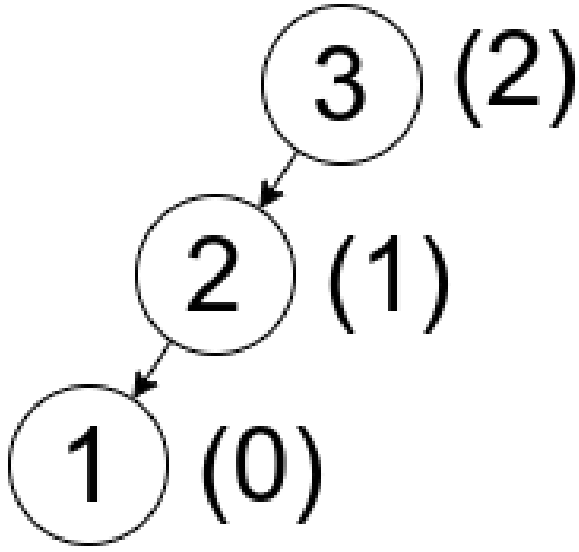
Inserir 2



Inserir 1



Inserir 1



- Desequilibrou a árvore.
- Como o balanceamento>1 e a chave a ser inserida é menor do que a chave do filho esquerdo do nó desbalanceado, vamos usar a **rotação à direita**.

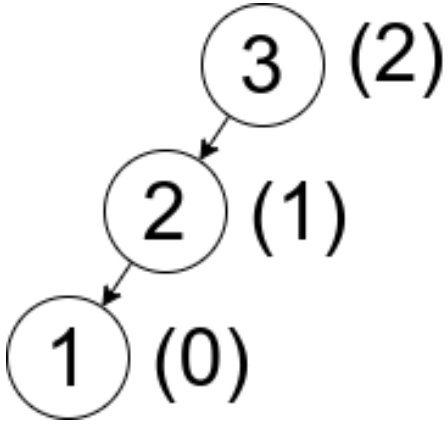
---

função rotacao_direita(raiz)
    novaRaiz = raiz.esquerda
    raiz.esquerda = novaRaiz.direita ▷ Na Figura, a subárvore direita da nova raiz é nula.
    novaRaiz.direita = raiz

    retorna novaRaiz
fim função

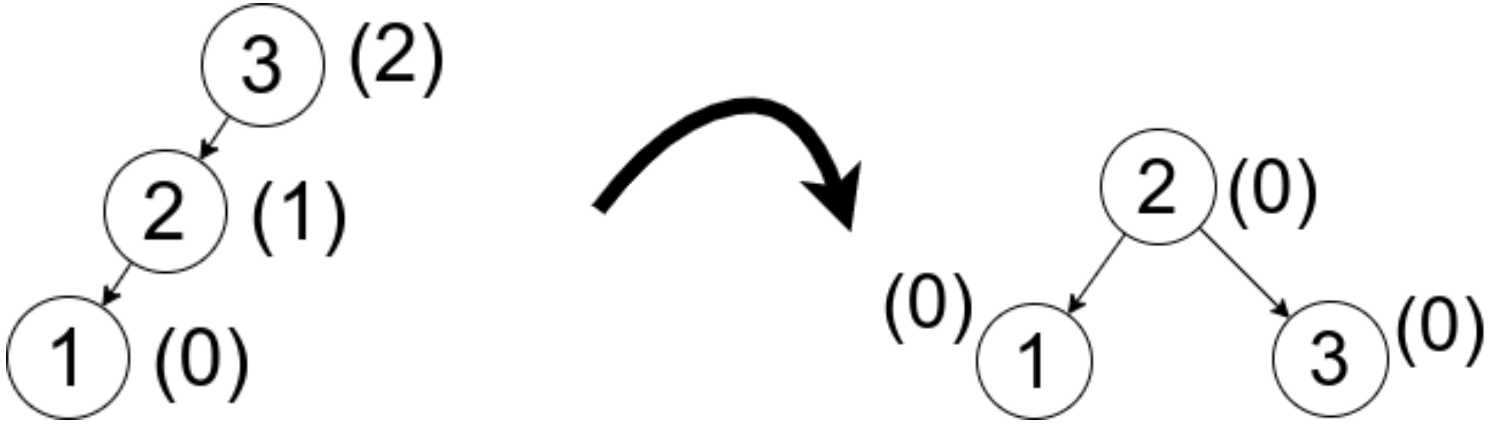
Rotação à direita

```
função rotacao_direita(raiz)
    novaRaiz = raiz.esquerda
    raiz.esquerda = novaRaiz.direita ▷ Na Figura, a subárvore direita da nova raiz é nula.
    novaRaiz.direita = raiz

    retorna novaRaiz
fim função
```

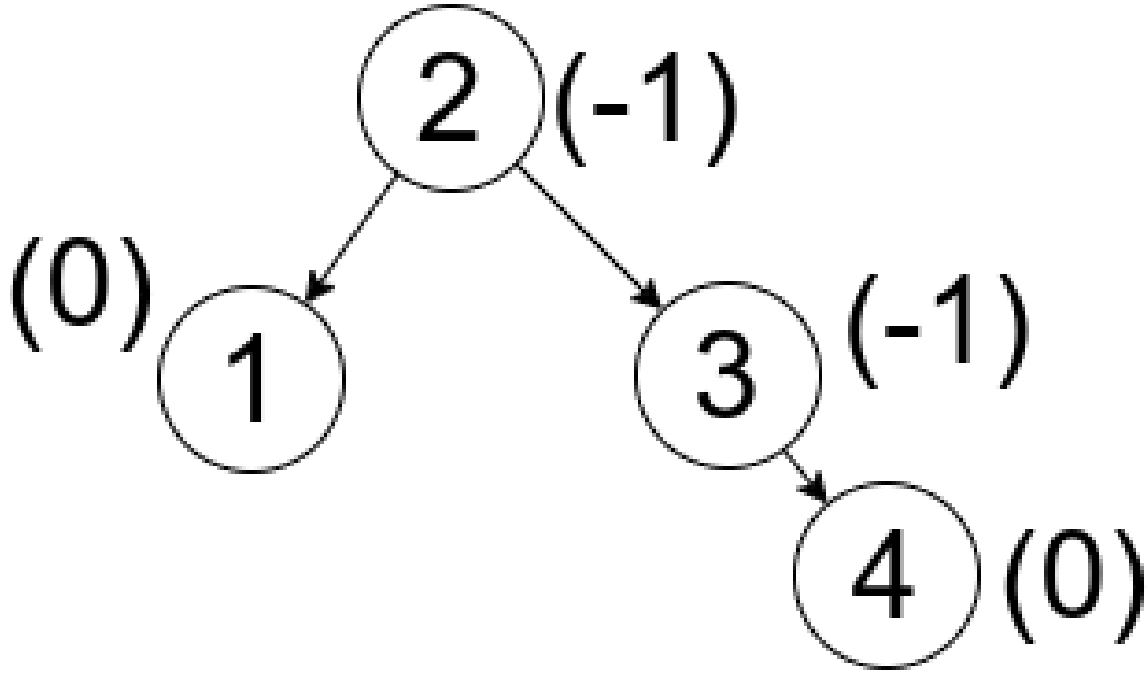


Rotação à direita

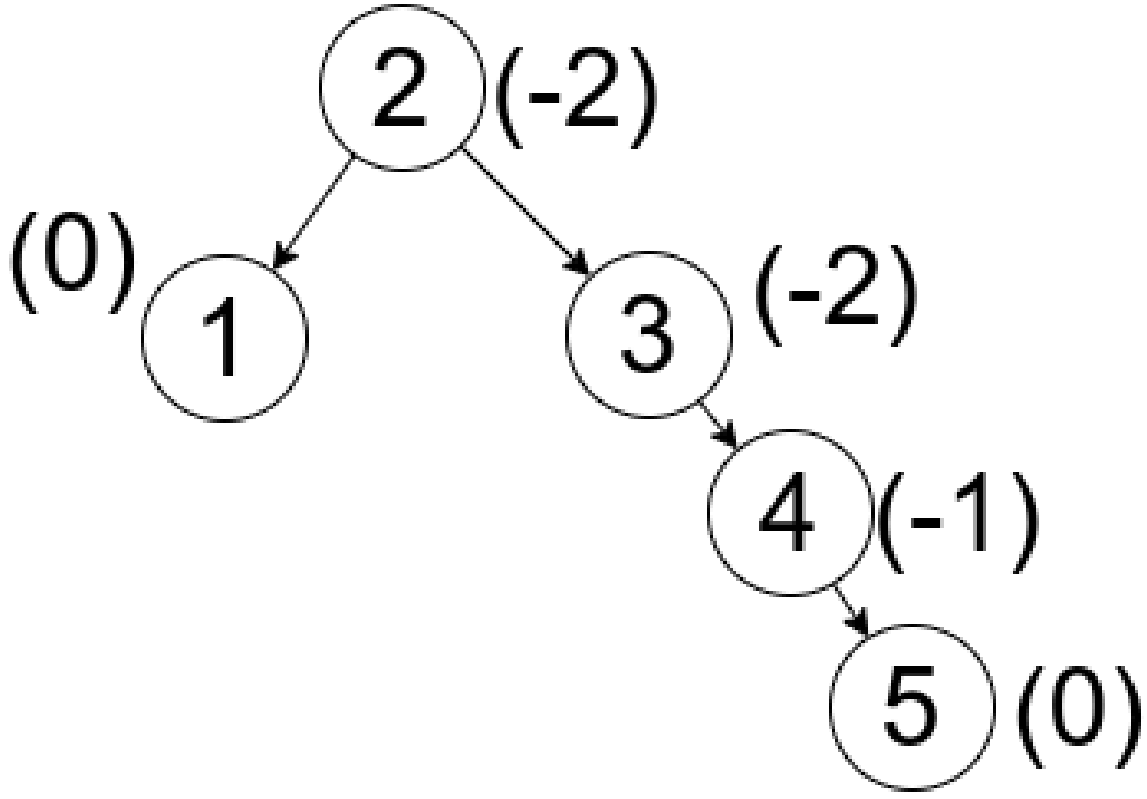
```
função rotacao_direita(raiz)
    novaRaiz = raiz.esquerda
    raiz.esquerda = novaRaiz.direita ▷ Na Figura, a subárvore direita da nova raiz é nula.
    novaRaiz.direita = raiz

    retorna novaRaiz
fim função
```



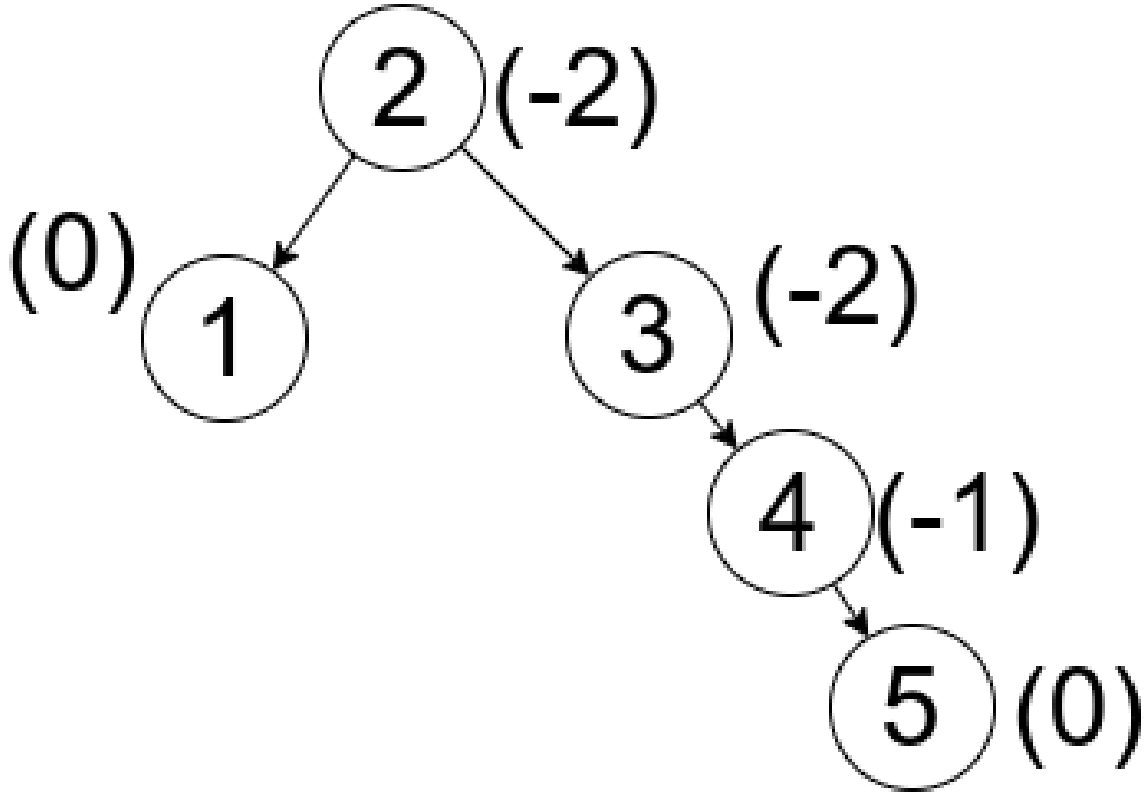
Inserir 4



Inserir 5



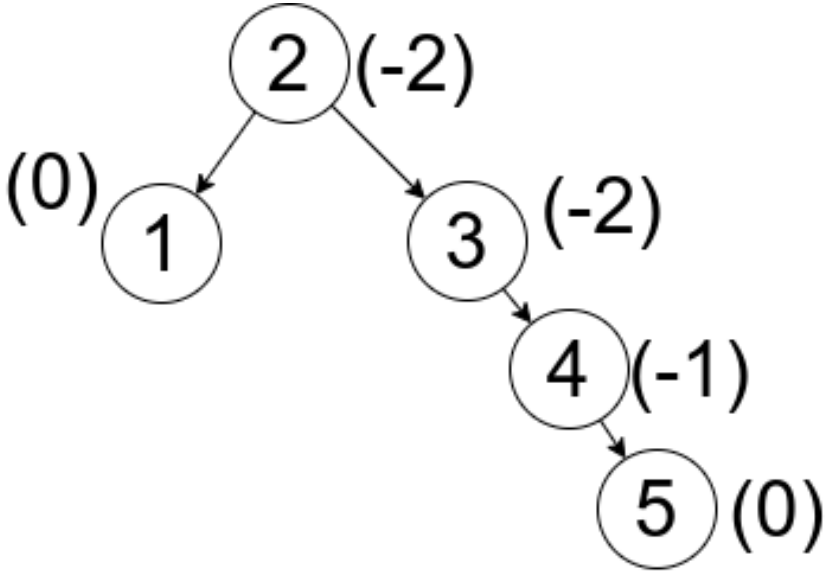
Inserir 5



- Desequilibrou a árvore.
- Como o balanceamento<-1 e a chave a ser inserida é maior do que a chave do filho direito do último nó desbalanceado, vamos usar a **rotação à esquerda**.

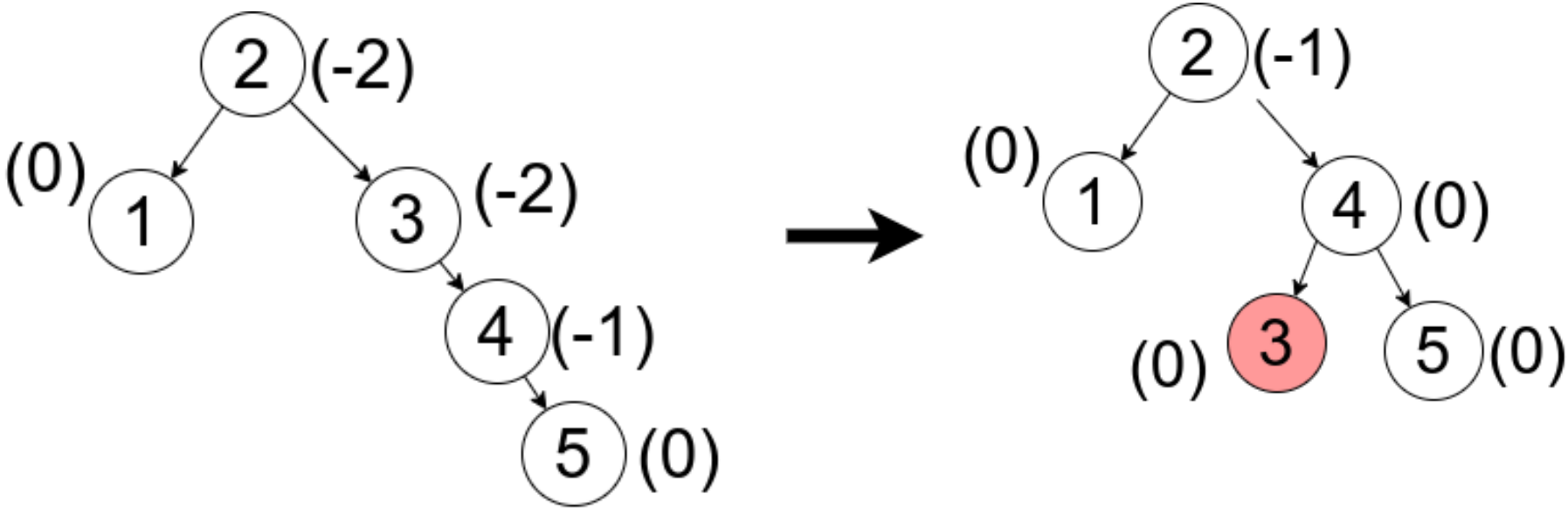
Rotação à esquerda

```
função rotacao_esquerda(raiz)

    novaRaiz = raiz.direita
    raiz.direita = novaRaiz.esquerda      ▷ Na Figura, a subárvore esquerda da nova raiz é nula.
    novaRaiz.esquerda = raiz
    retorna novaRaiz
fim função
```

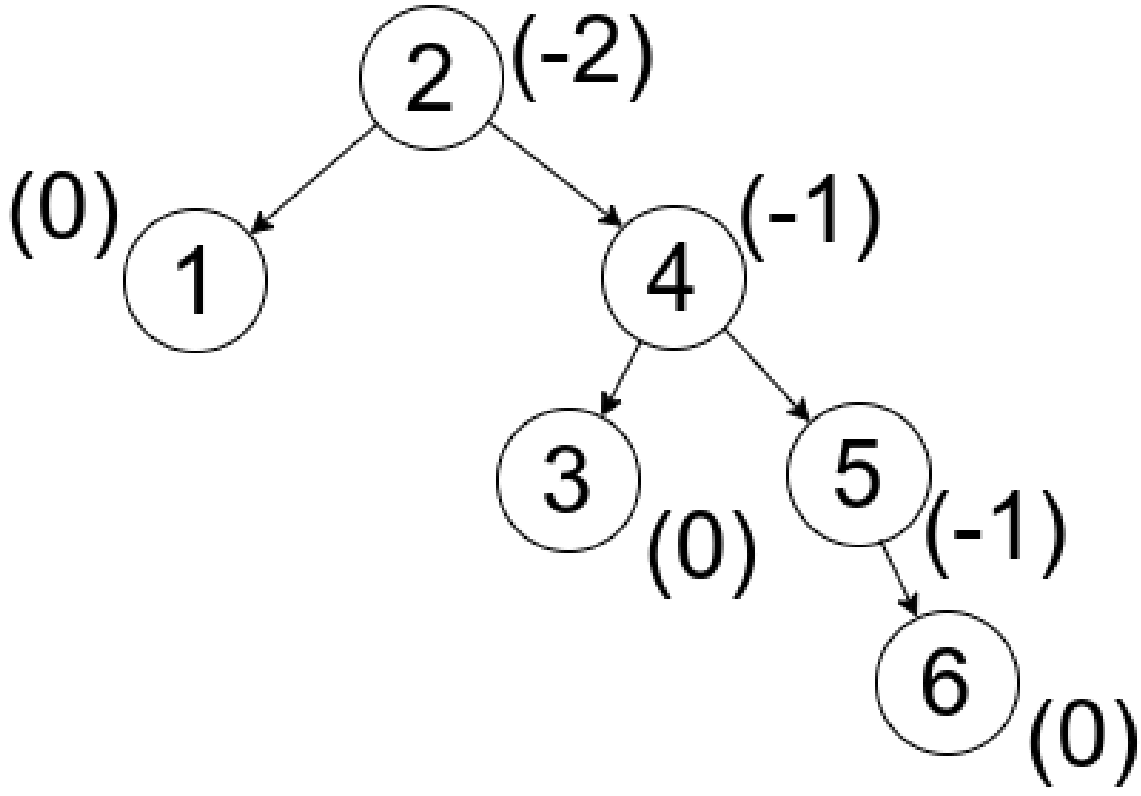


Rotação à esquerda

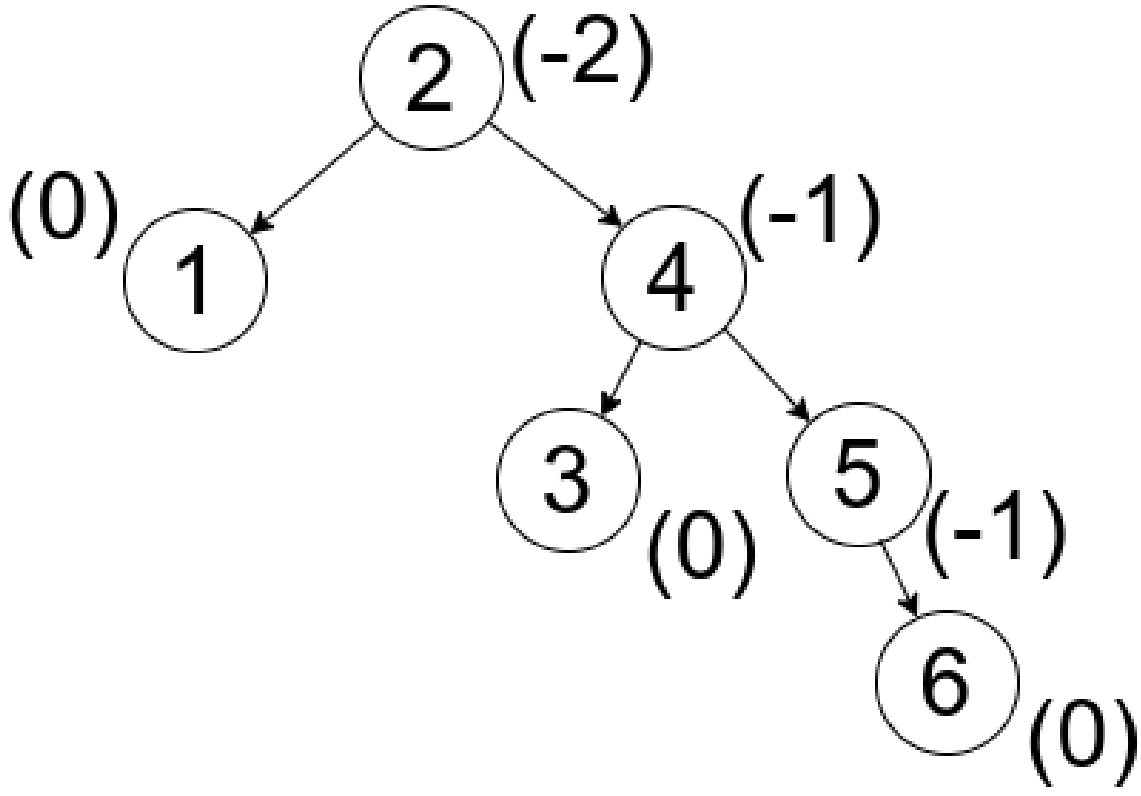
```
função rotacao_esquerda(raiz)

    novaRaiz = raiz.direita
    raiz.direita = novaRaiz.esquerda      ▷ Na Figura, a subárvore esquerda da nova raiz é nula.
    novaRaiz.esquerda = raiz
    retorna novaRaiz
fim função
```



Inserir 6



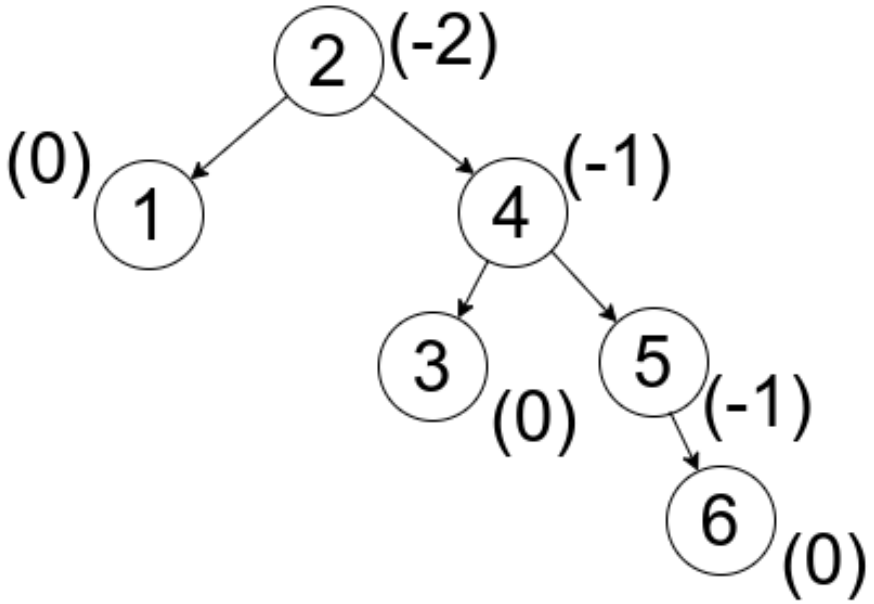
Inserir 6



- Desequilibrou a árvore.
- Como o balanceamento<-1 e a chave a ser inserida é maior do que a chave do filho direito do último nó desbalanceado, vamos usar a **rotação à esquerda**.

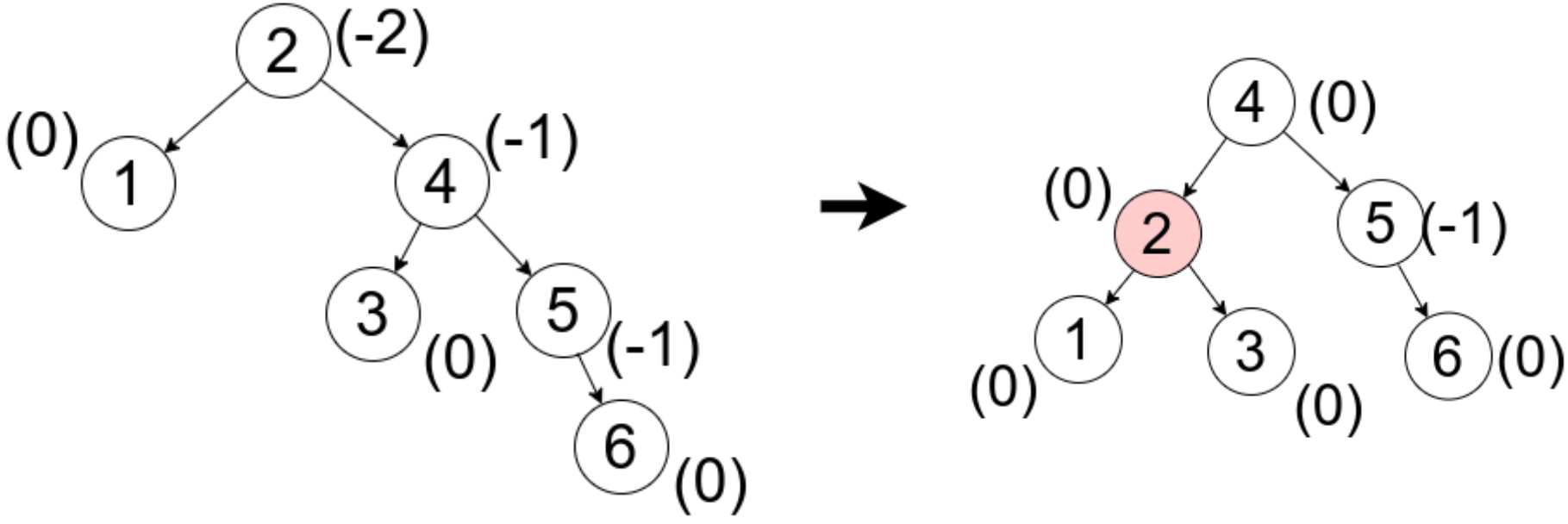
Rotação à esquerda

```
função rotacao_esquerda(raiz)

    novaRaiz = raiz.direita
    raiz.direita = novaRaiz.esquerda
    novaRaiz.esquerda = raiz
    retorna novaRaiz
fim função
```

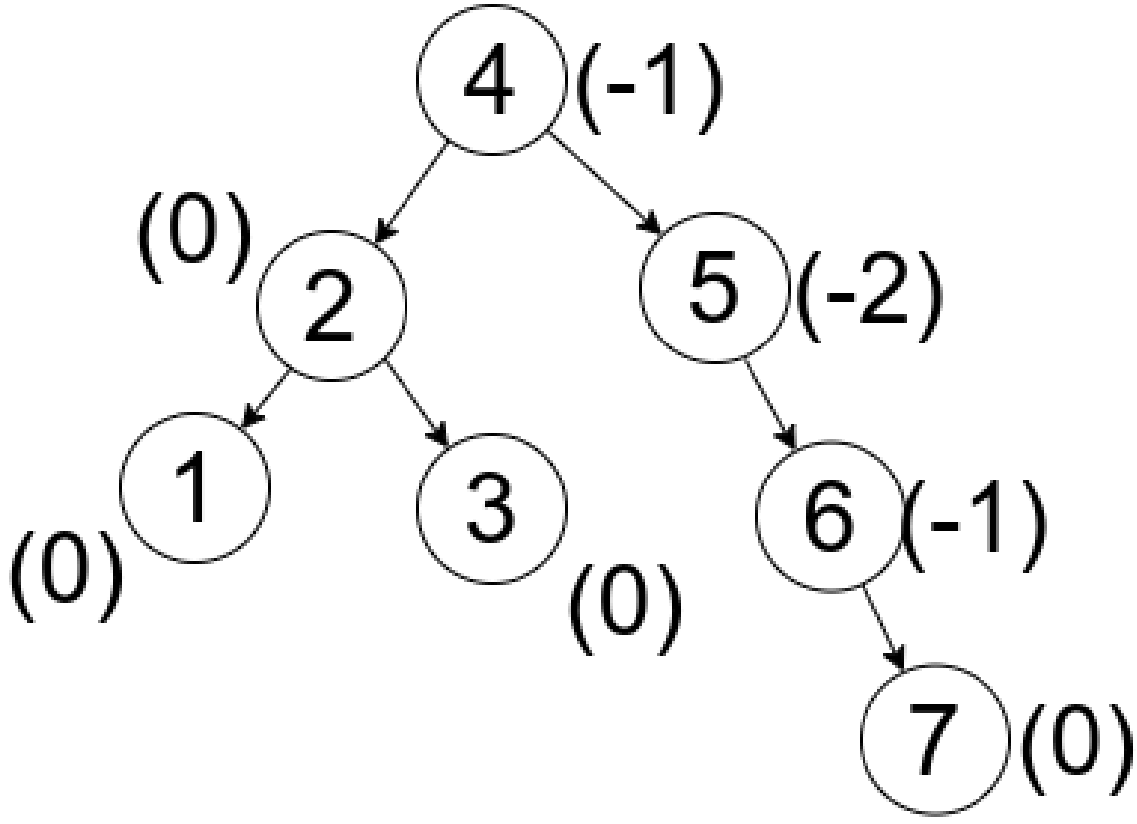


Rotação à esquerda

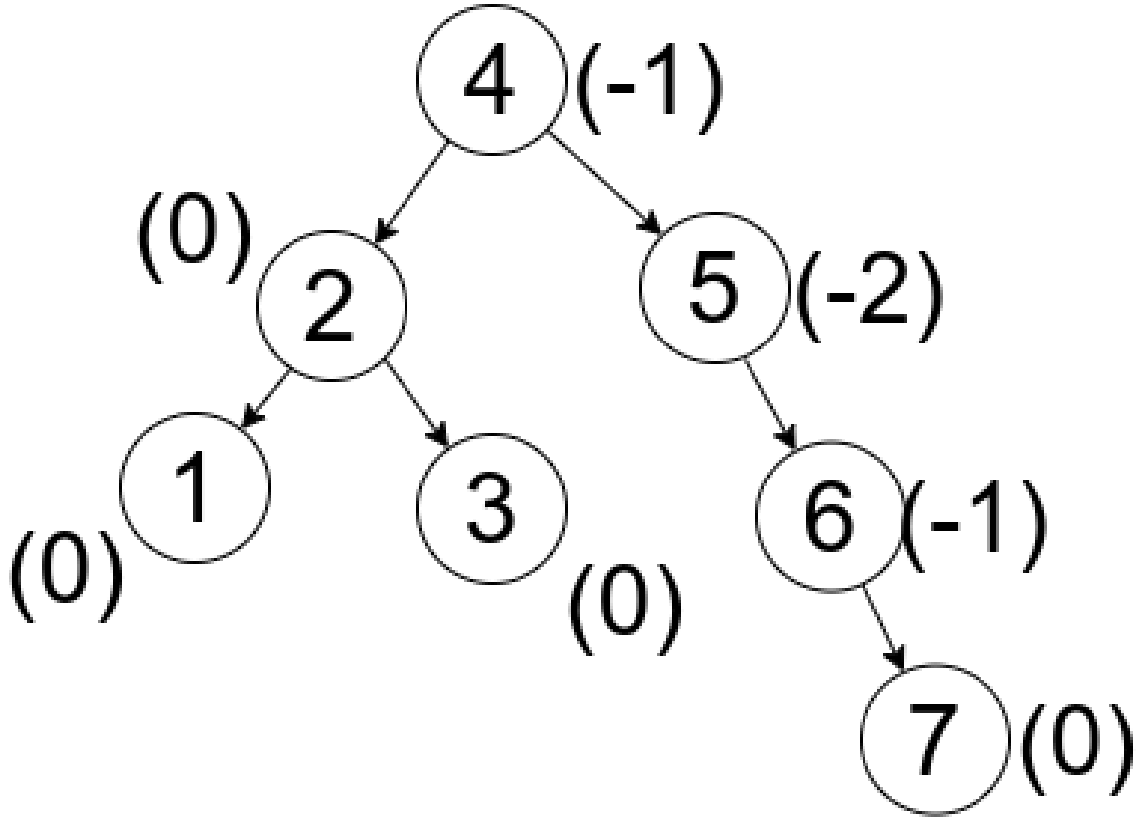
```
função rotacao_esquerda(raiz)

    novaRaiz = raiz.direita
    raiz.direita = novaRaiz.esquerda
    novaRaiz.esquerda = raiz
    retorna novaRaiz
fim função
```



Inserir 7



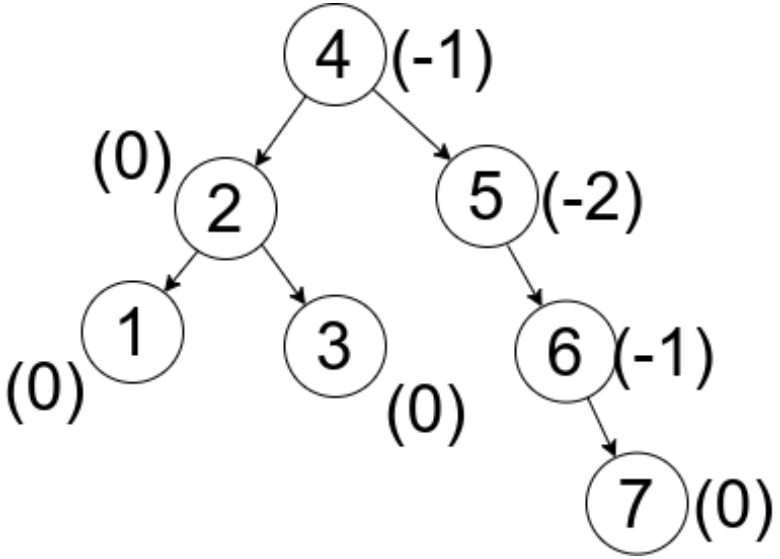
Inserir 7



- Desequilibrou a árvore.
- Como o balanceamento<-1 e a chave a ser inserida é maior do que a chave do filho direito do último nó desbalanceado, vamos usar a **rotação à esquerda**.

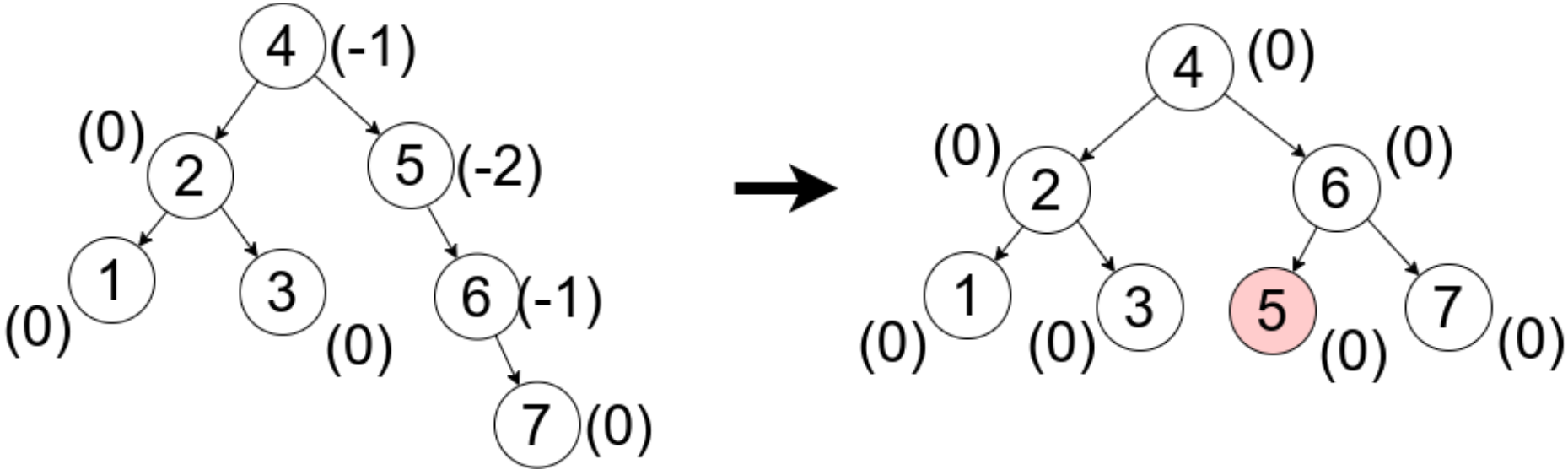
Rotação à esquerda

```
função rotacao_esquerda(raiz)

    novaRaiz = raiz.direita
    raiz.direita = novaRaiz.esquerda
    novaRaiz.esquerda = raiz
    retorna novaRaiz
fim função
```

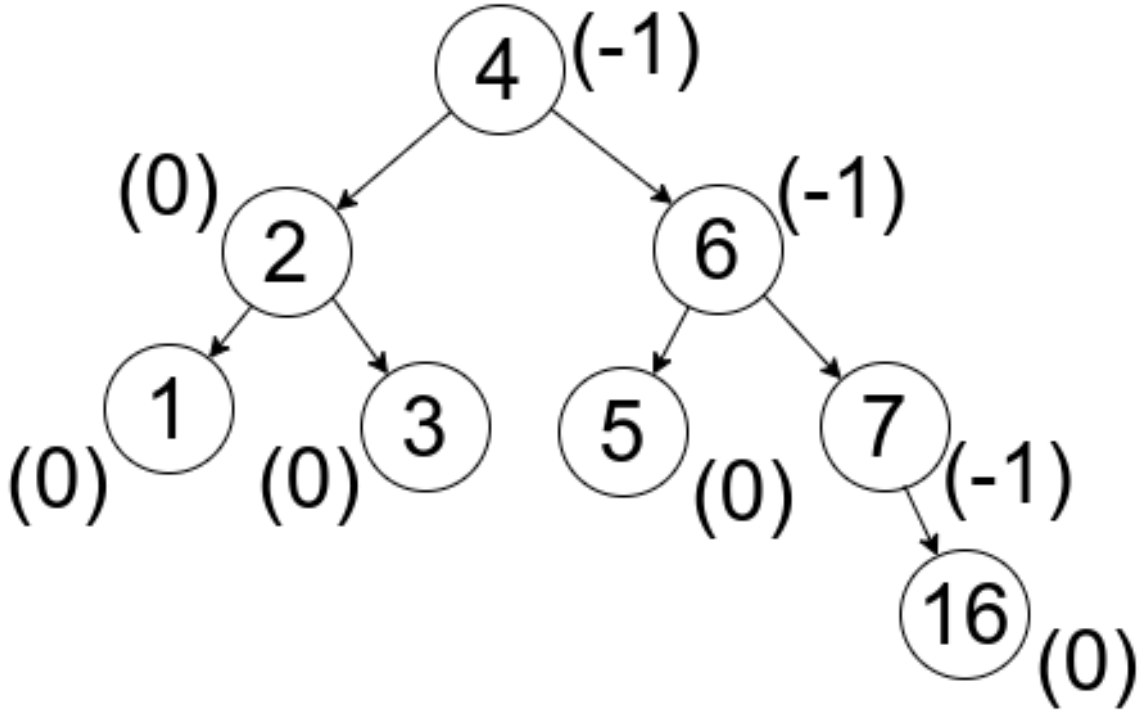


Rotação à esquerda

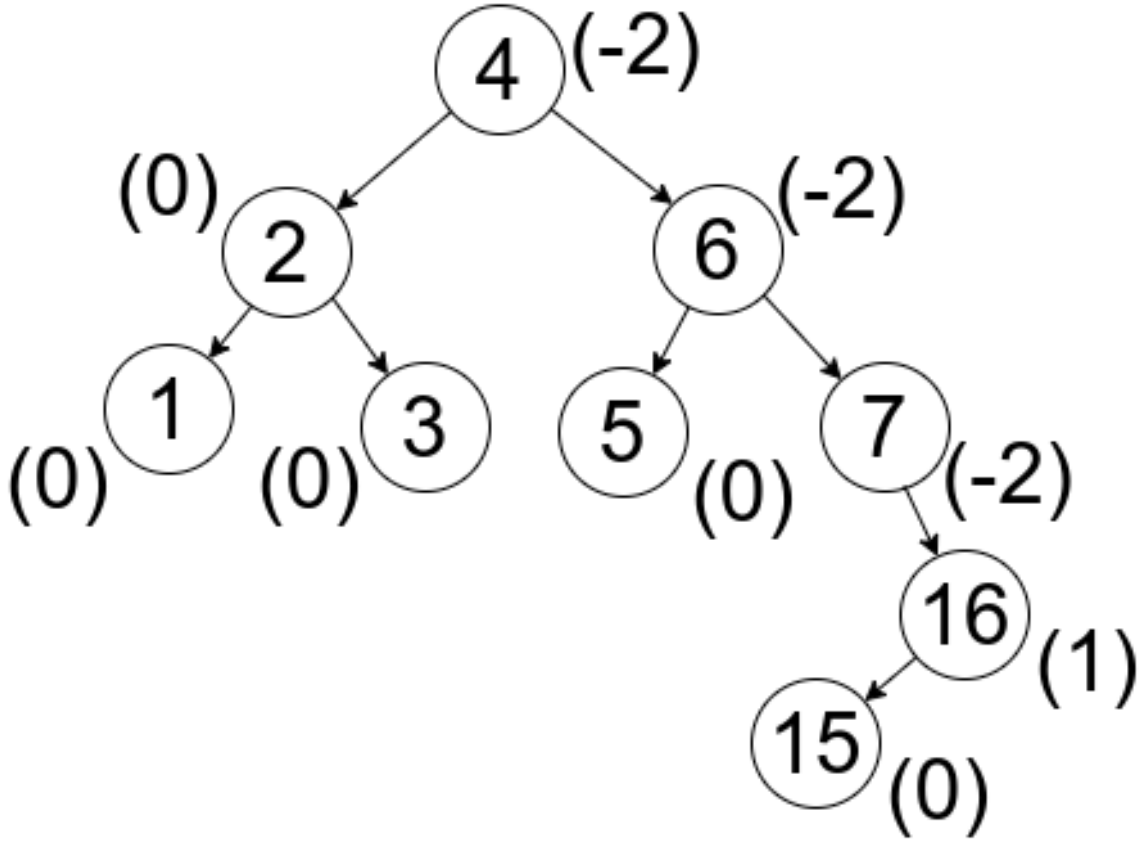
```
função rotacao_esquerda(raiz)

    novaRaiz = raiz.direita
    raiz.direita = novaRaiz.esquerda
    novaRaiz.esquerda = raiz
    retorna novaRaiz
fim função
```



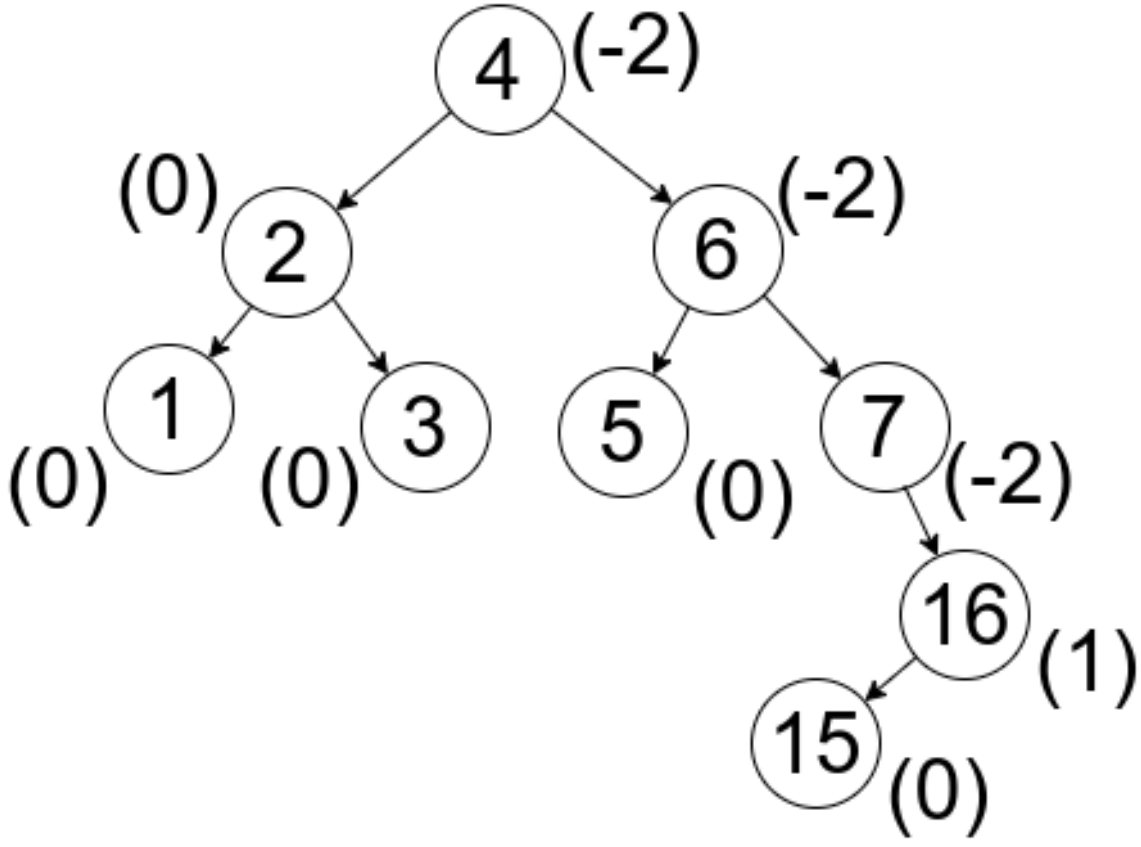
Inserir 16



Inserir 15



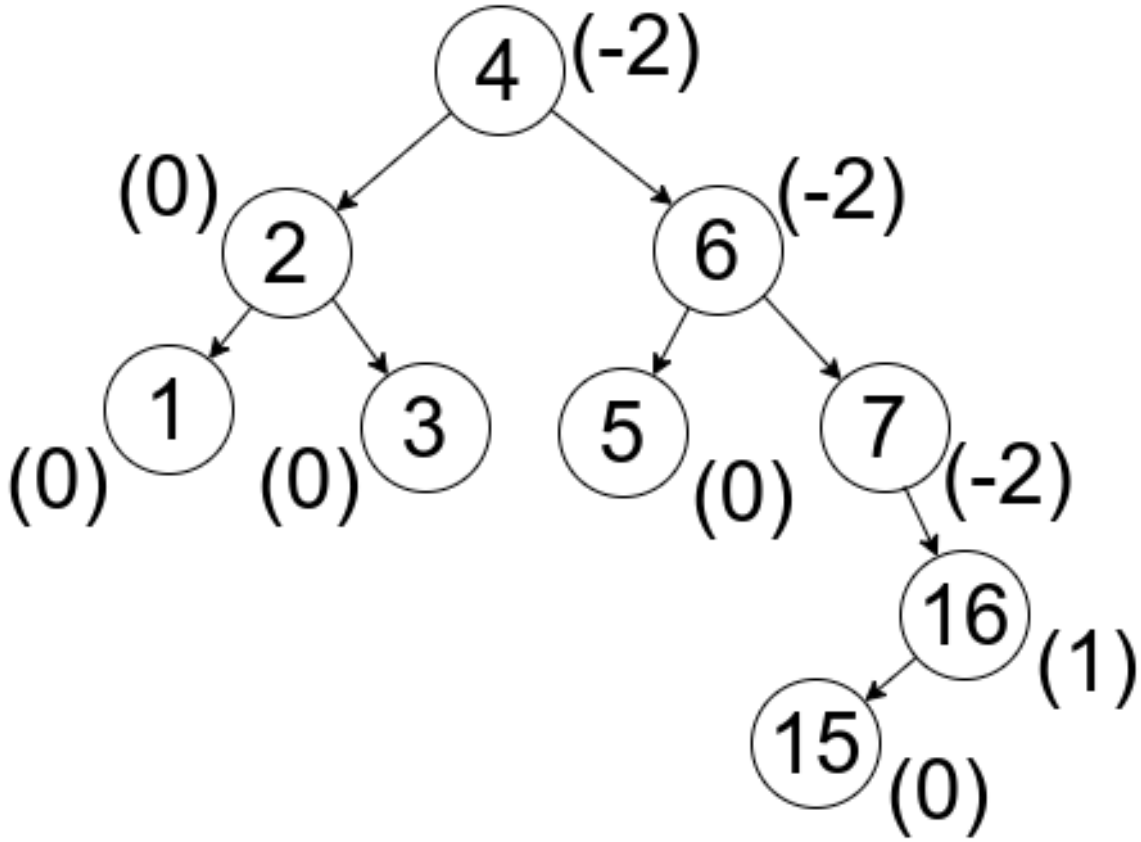
Inserir 15



- Desequilibrou a árvore.
- Como o balanceamento<-1 e a chave a ser inserida é menor do que a chave do filho direito do último nó desbalanceado, vamos usar a rotação **direita-esquerda**.

Rotação direita-esquerda

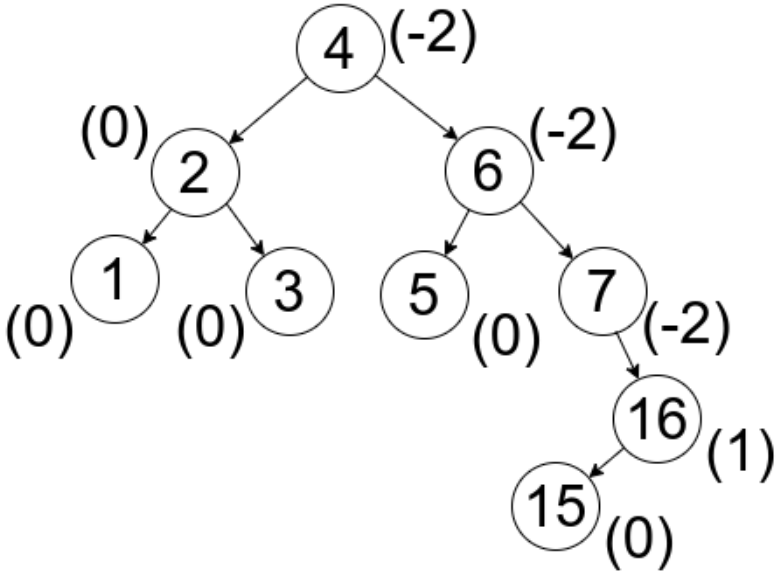
```
função rotacao_direitaEsquerda(raiz)
    raiz.direita = rotacao_direita(raiz.direita)
    novaRaiz = rotacao_esquerda(raiz)

    retorna novaRaiz
fim função
```



Rotação direita-esquerda

```
função rotacao_direitaEsquerda(raiz)
    raiz.direita = rotacao_direita(raiz.direita)
    novaRaiz = rotacao_esquerda(raiz)

    retorna novaRaiz
fim função
```

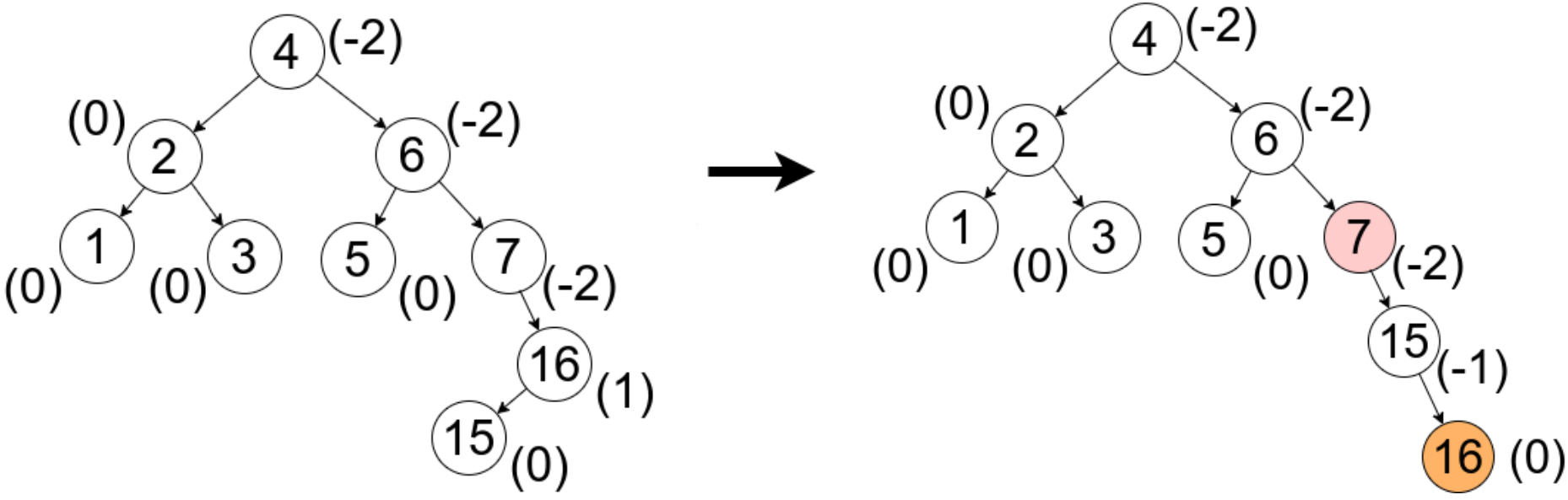
```
função rotacao_direita(raiz)
    novaRaiz = raiz.esquerda
    raiz.esquerda = novaRaiz.direita ▷ Na Figura, a subárvore direita da nova raiz é nula.
    novaRaiz.direita = raiz

    retorna novaRaiz
fim função
```



Rotação direita-esquerda

```
função rotacao_direitaEsquerda(raiz)
    raiz.direita = rotacao_direita(raiz.direita)
    novaRaiz = rotacao_esquerda(raiz)

    retorna novaRaiz
fim função
```

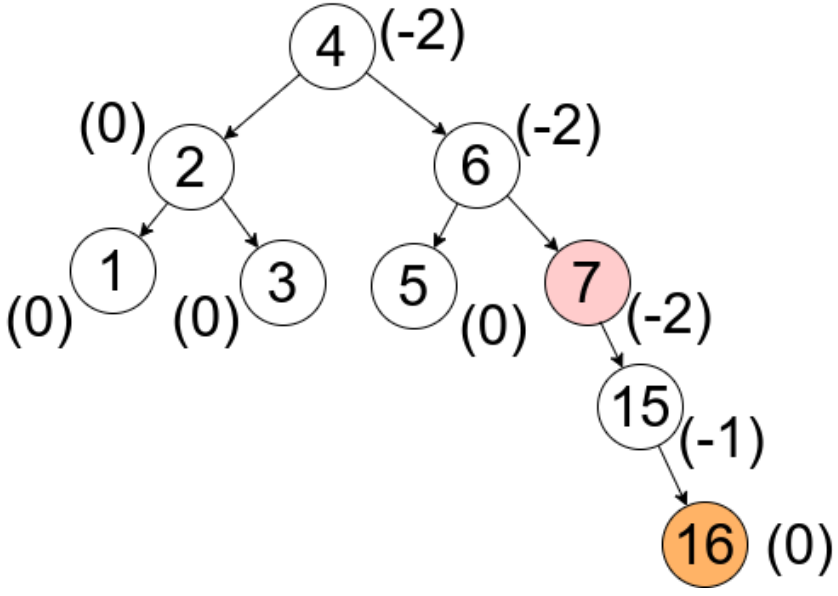
```
função rotacao_direita(raiz)
    novaRaiz = raiz.esquerda
    raiz.esquerda = novaRaiz.direita ▷ Na Figura, a subárvore direita da nova raiz é nula.
    novaRaiz.direita = raiz

    retorna novaRaiz
fim função
```



Rotação direita-esquerda

```
função rotacao_direitaEsquerda(raiz)
    raiz.direita = rotacao_direita(raiz.direita)
    novaRaiz = rotacao_esquerda(raiz)

    retorna novaRaiz
fim função
```

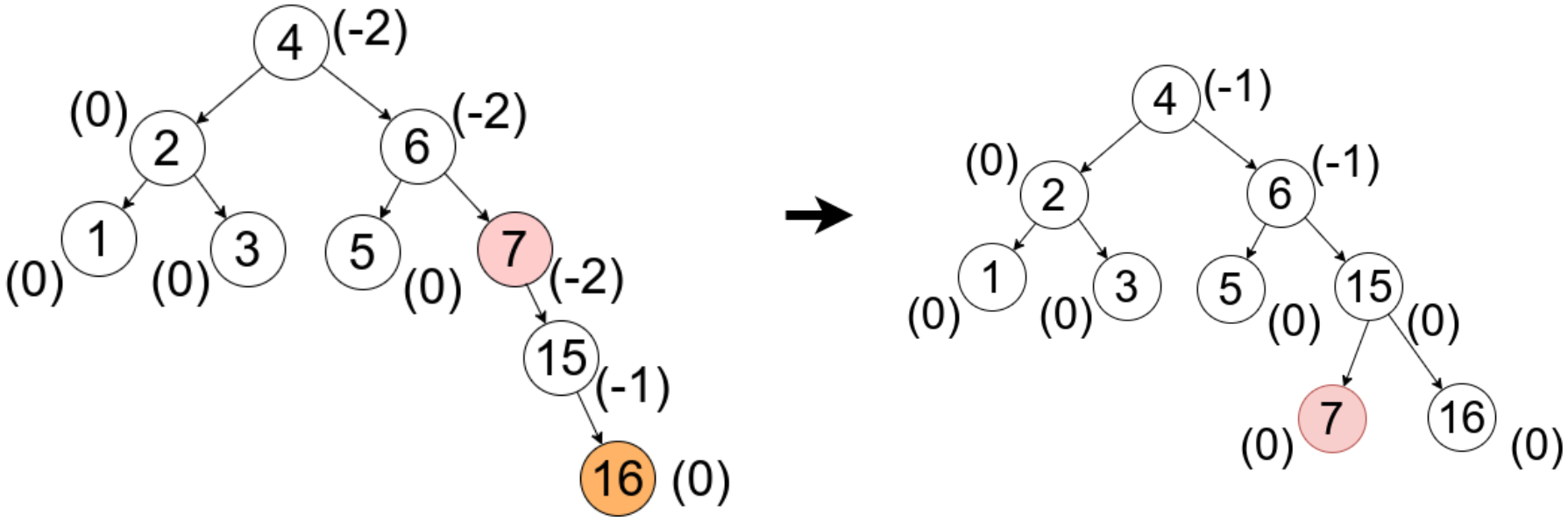
```
função rotacao_esquerda(raiz)

    novaRaiz = raiz.direita
    raiz.direita = novaRaiz.esquerda
    novaRaiz.esquerda = raiz
    retorna novaRaiz
fim função
```



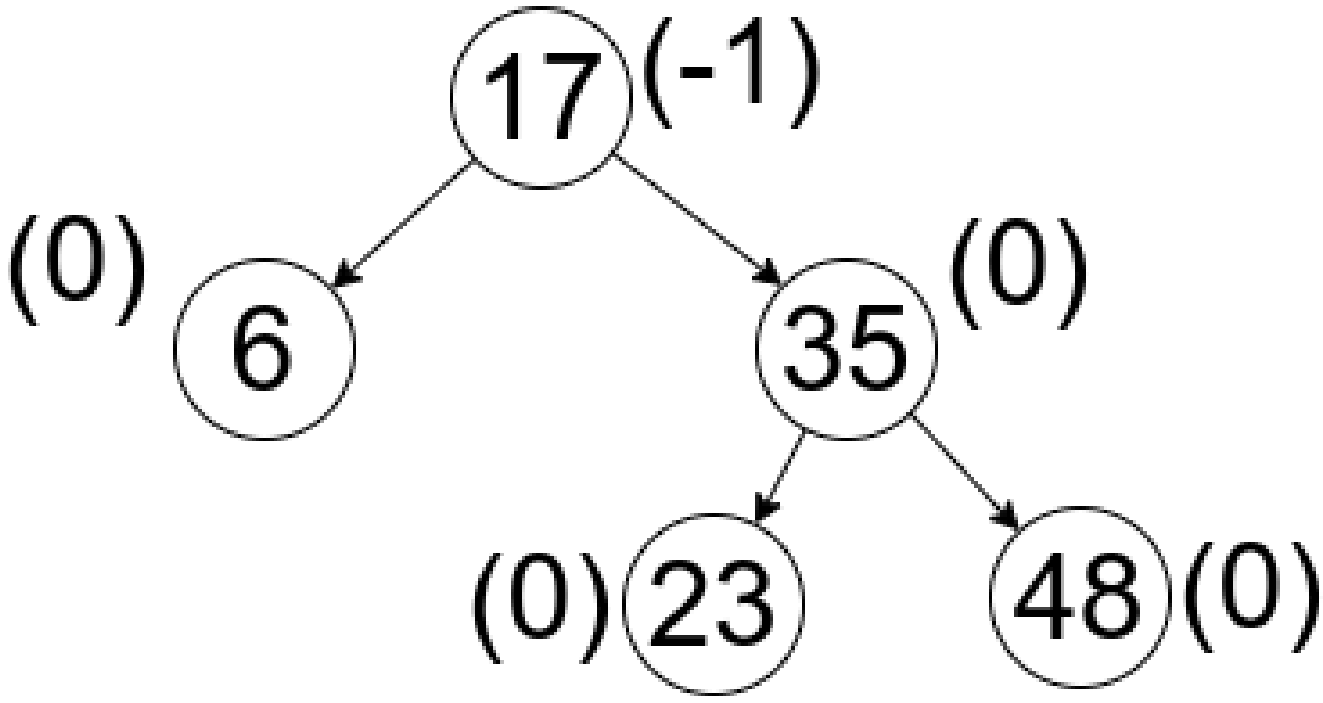
Rotação direita-esquerda

```
função rotacao_direitaEsquerda(raiz)
    raiz.direita = rotacao_direita(raiz.direita)
    novaRaiz = rotacao_esquerda(raiz)

    retorna novaRaiz
fim função
```

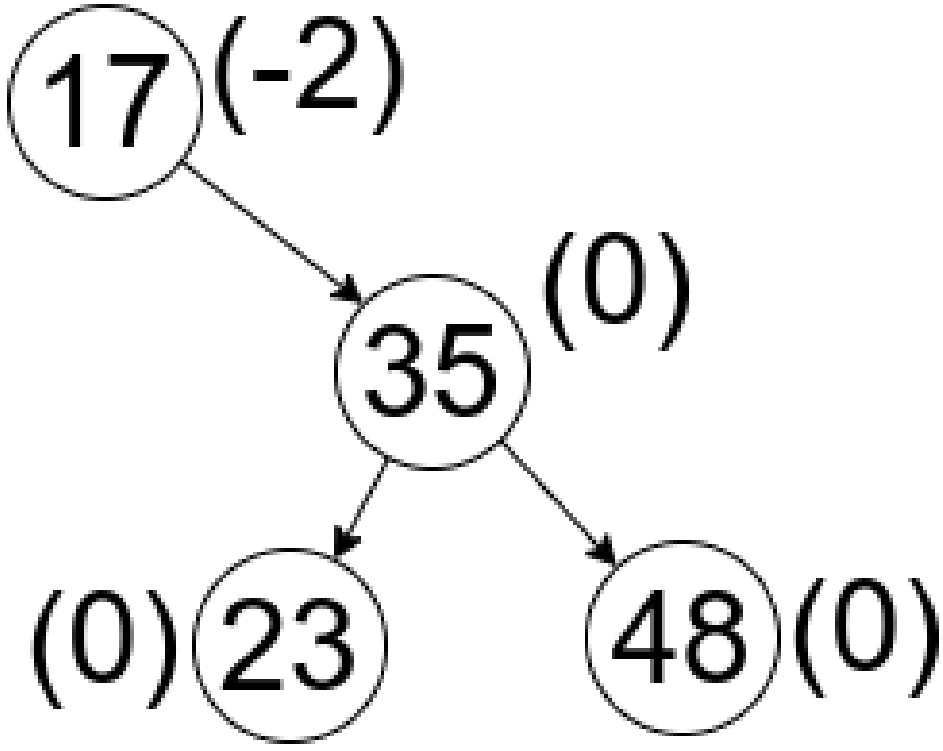
```
função rotacao_esquerda(raiz)

    novaRaiz = raiz.direita
    raiz.direita = novaRaiz.esquerda
    novaRaiz.esquerda = raiz
    retorna novaRaiz
fim função
```



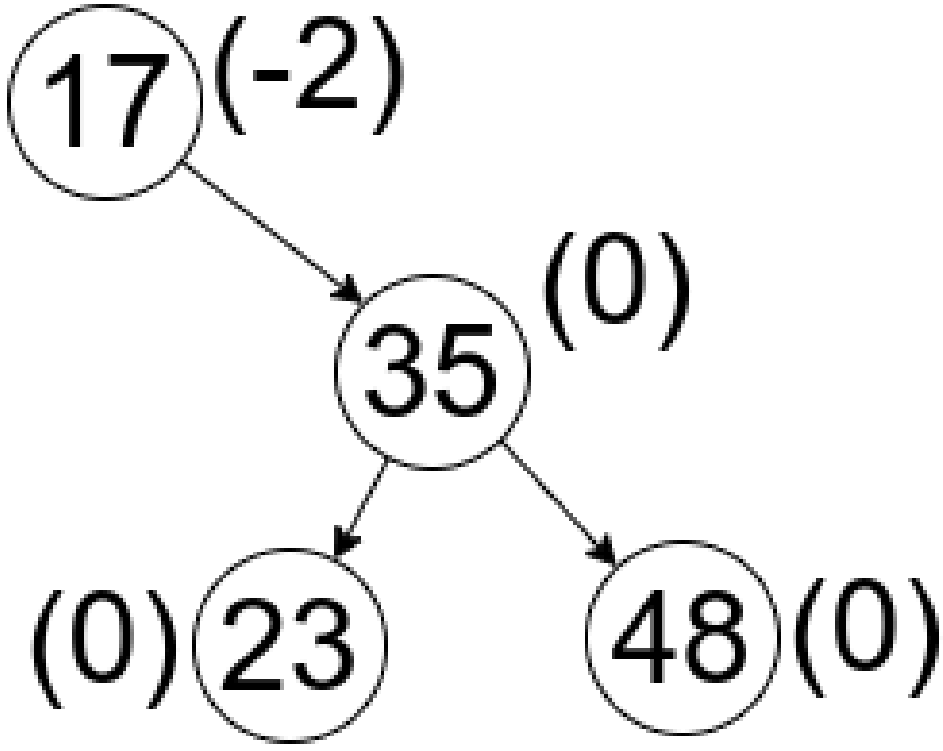
Ex. 08: Mostre todos os passos para remover os nós 6 e 35 **árvore AVL** mostrada abaixo:



Remover 6



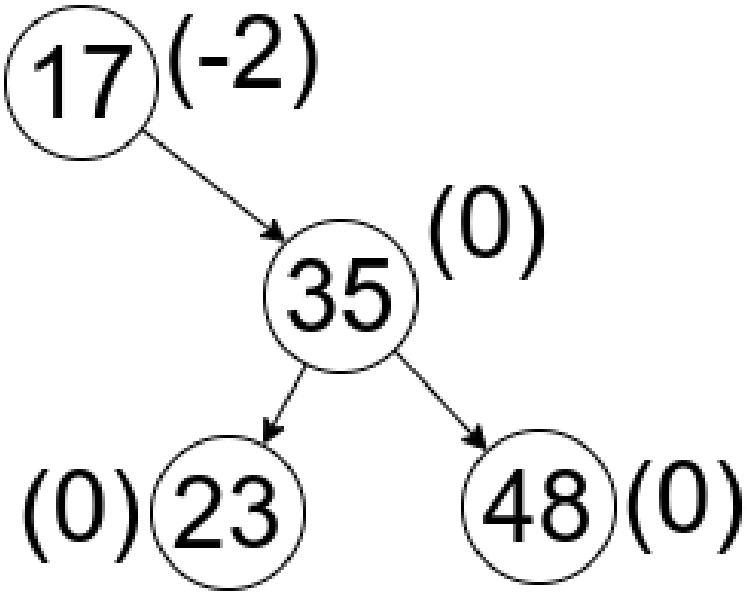
Remover 6



- Desequilibrou a árvore.
- Como o balanceamento do nó desbalanceado é menor que -1 e o balanceamento do seu filho à direita é menor ou igual a 0, vamos usar a rotação **esquerda**.

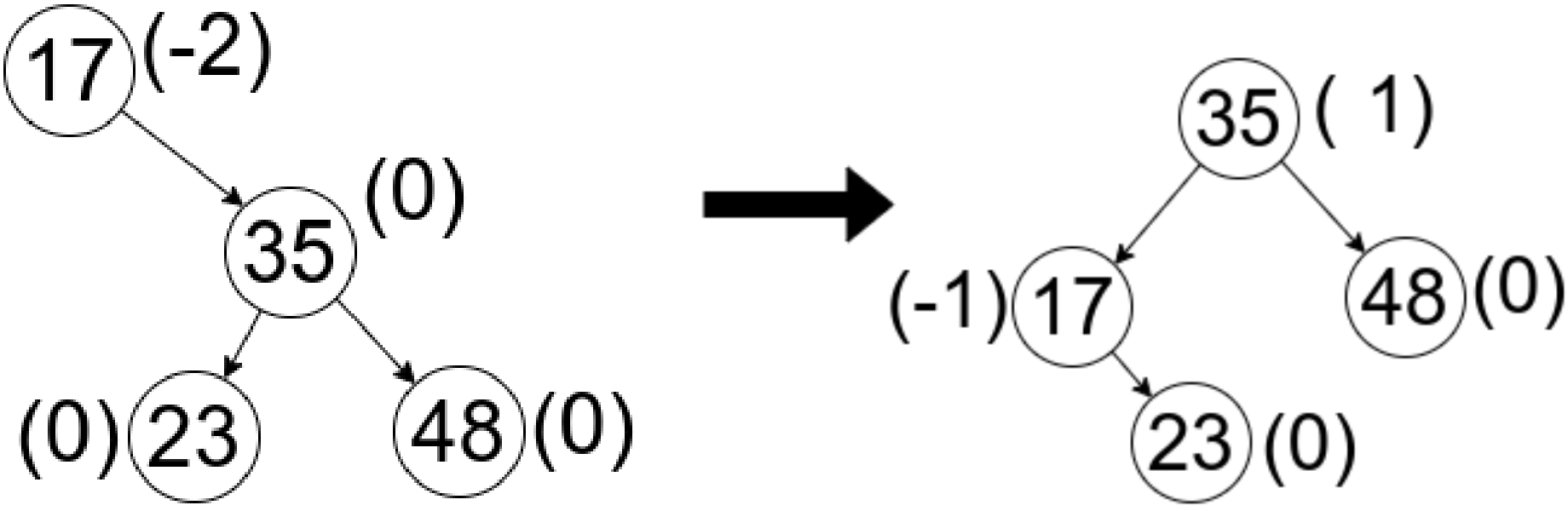
Rotação à esquerda

```
função rotacao_esquerda(raiz)

    novaRaiz = raiz.direita
    raiz.direita = novaRaiz.esquerda
    novaRaiz.esquerda = raiz
    retorna novaRaiz
fim função
```



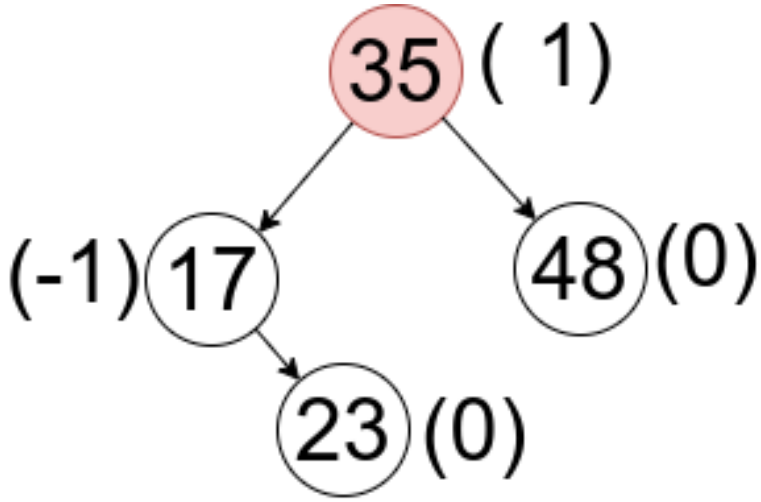
Rotação à esquerda

```
função rotacao_esquerda(raiz)

    novaRaiz = raiz.direita
    raiz.direita = novaRaiz.esquerda
    novaRaiz.esquerda = raiz
    retorna novaRaiz
fim função
```



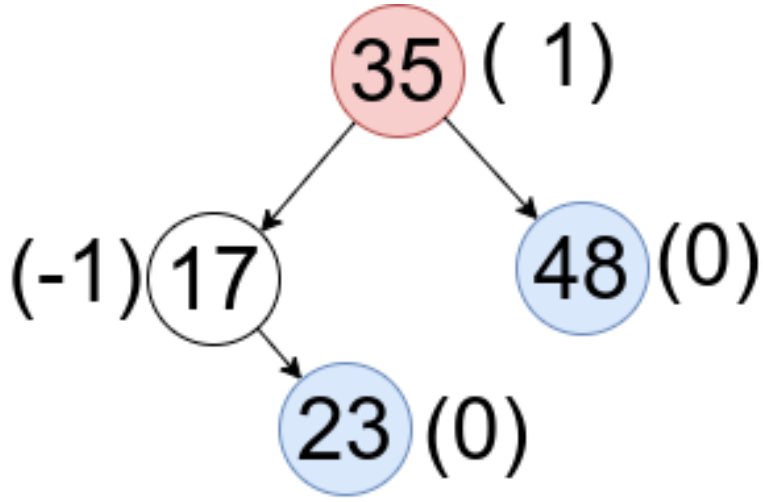
Removendo o nó 35.

- O nó possui filho?
- Como possui dois filhos, devemos:
  - Substituir o valor do nó a ser retirado pelo valor sucessor (o nó mais à esquerda da subárvore direita) ou
  - substituir pelo valor antecessor (o nó mais à direita da subárvore esquerda)



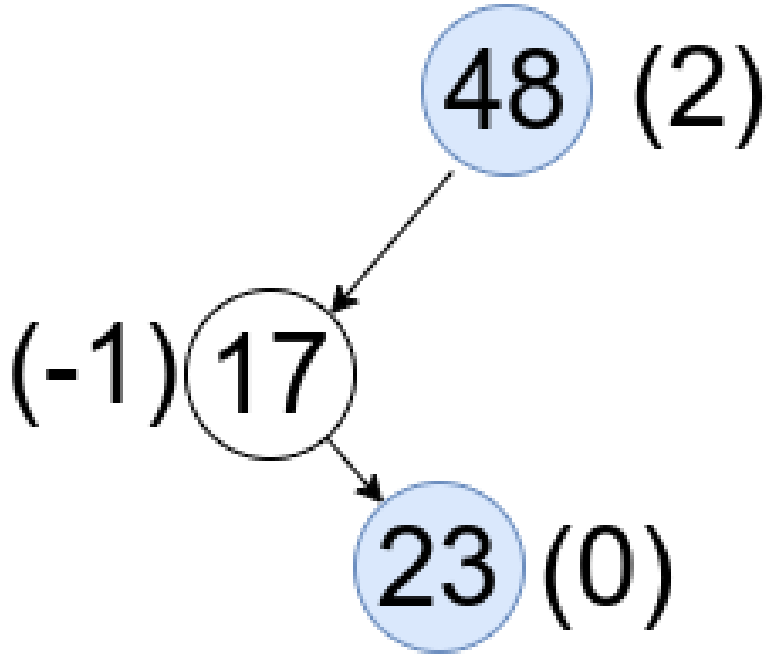
Removendo o nó 35.

- O nó possui filho?
- Como possui dois filhos, devemos:
  - Substituir o valor do nó a ser retirado pelo valor sucessor (o nó mais à esquerda da subárvore direita) ou
  - substituir pelo valor antecessor (o nó mais à direita da subárvore esquerda)
  - Os dois valores candidatos a substituírem o nó que será excluído estão destacados em azul
    - Vamos escolher o nó 48.



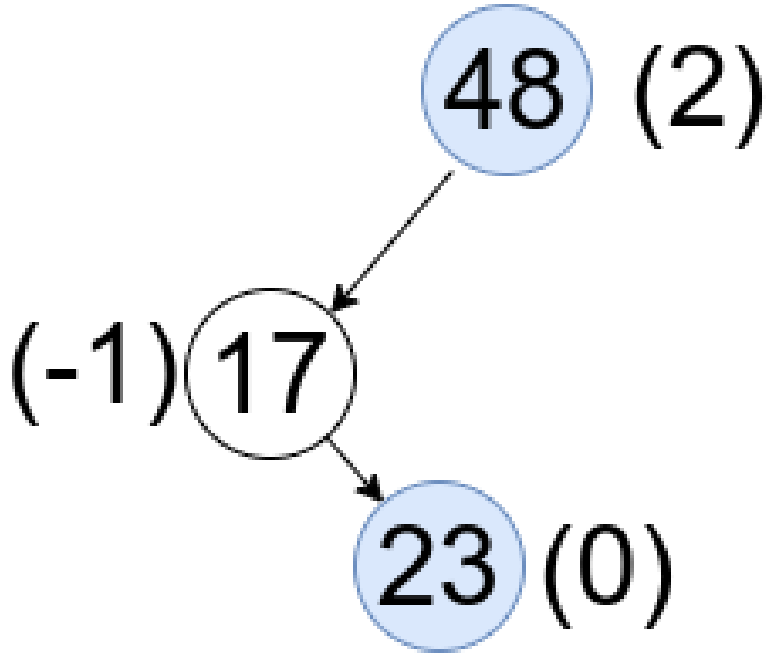
Remover 35.

- O nó possui filho?
- Como possui dois filhos, devemos:
  - Substituir o valor do nó a ser retirado pelo valor sucessor (o nó mais à esquerda da subárvore direita) ou
  - substituir pelo valor antecessor (o nó mais à direita da subárvore esquerda)



Remover 35

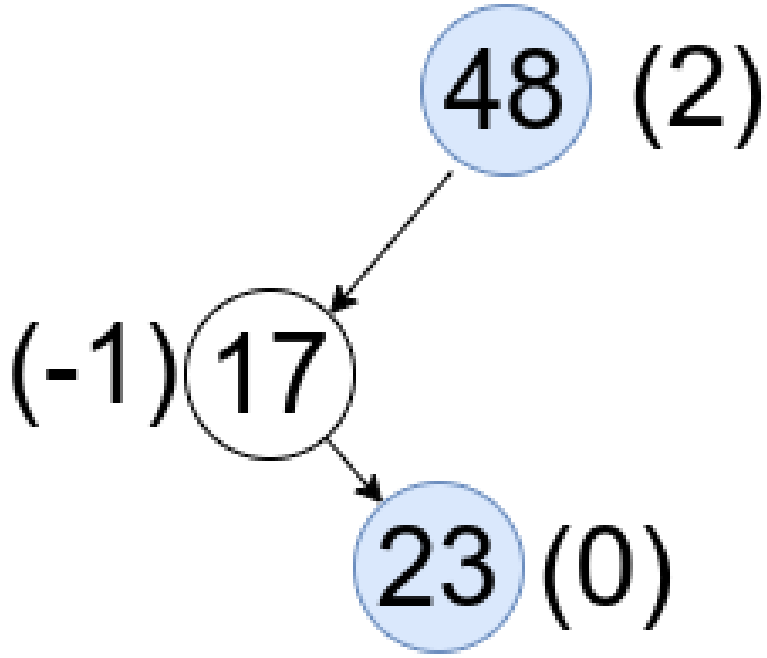
- Desequilibrou a árvore.
- Como o balanceamento do nó desbalanceado é maior que 1 e o balanceamento do seu filho à esquerda é menor que 0, vamos usar a rotação **esquerda-direita**.



Rotação esquerda-direita

Algoritmo Rotação esquerda-direita

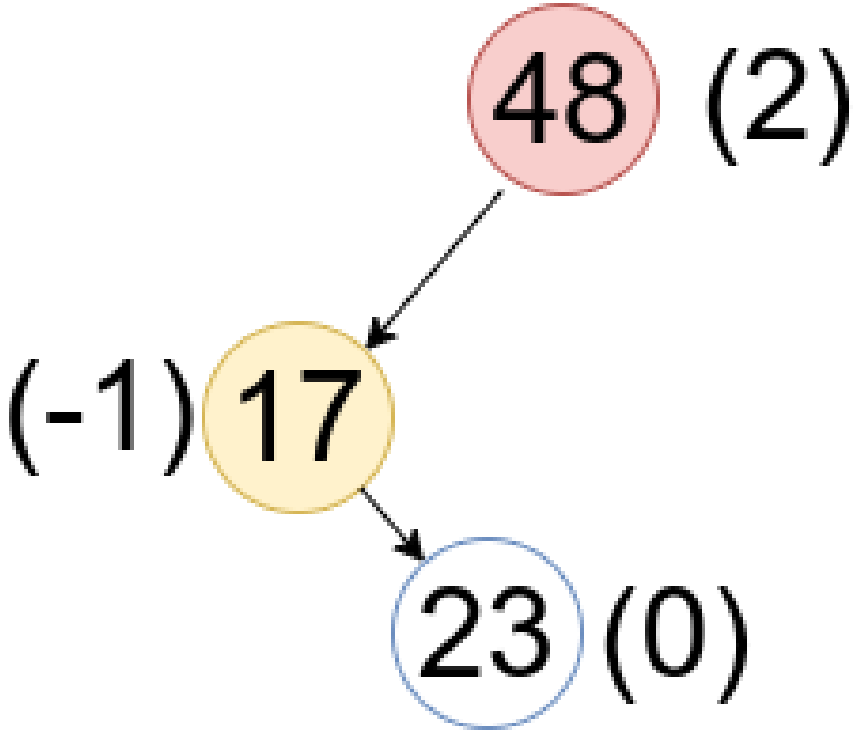
```
função rotacao_esquerdaDireita(raiz)

    raiz.esquerda = rotacao_esquerda(raiz.esquerda)
    novaRaiz = rotacao_direita(raiz)

    retorna novaRaiz
fim função
```



Rotação esquerda-direita

```
função rotacao_esquerdaDireita(raiz)

    raiz.esquerda = rotacao_esquerda(raiz.esquerda)
    novaRaiz = rotacao_direita(raiz)

    retorna novaRaiz
fim função
```

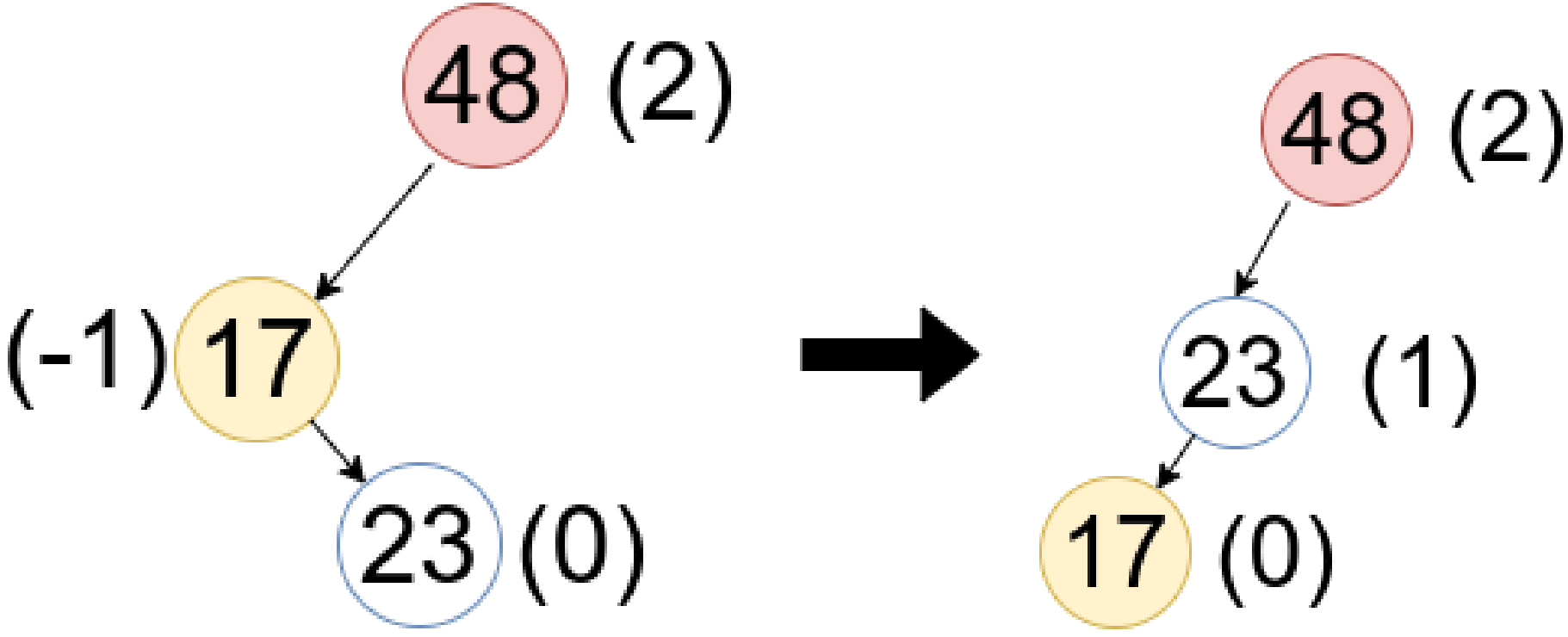
```
função rotacao_esquerda(raiz)

    novaRaiz = raiz.direita
    raiz.direita = novaRaiz.esquerda
    novaRaiz.esquerda = raiz
    retorna novaRaiz
fim função
```



Rotação esquerda-direita

```
função rotacao_esquerdaDireita(raiz)

    raiz.esquerda = rotacao_esquerda(raiz.esquerda)
    novaRaiz = rotacao_direita(raiz)

    retorna novaRaiz
fim função
```

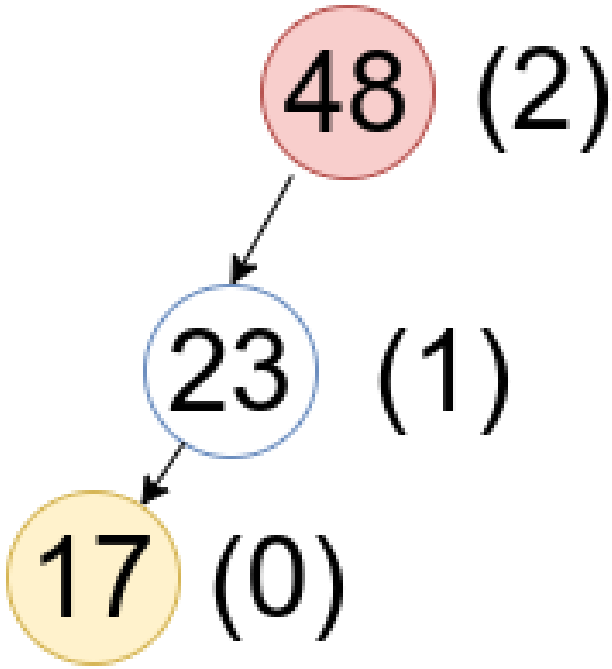
```
função rotacao_esquerda(raiz)

    novaRaiz = raiz.direita
    raiz.direita = novaRaiz.esquerda
    novaRaiz.esquerda = raiz
    retorna novaRaiz
fim função
```



Rotação esquerda-direita

```
função rotacao_esquerdaDireita(raiz)

    raiz.esquerda = rotacao_esquerda(raiz.esquerda)
    novaRaiz = rotacao_direita(raiz)

    retorna novaRaiz
fim função
```

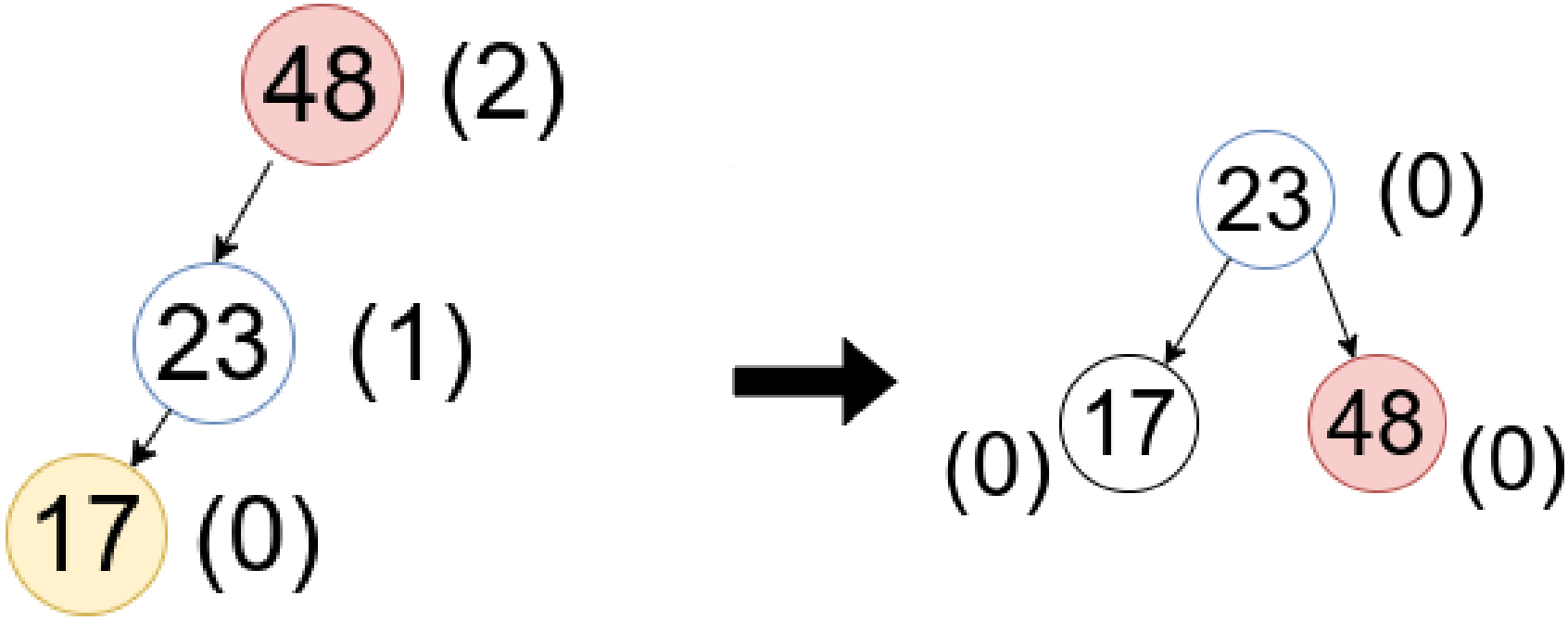
```
função rotacao_direita(raiz)
    novaRaiz = raiz.esquerda
    raiz.esquerda = novaRaiz.direita    ▷ Na Figura, a subárvore direita da nova raiz é nula.
    novaRaiz.direita = raiz

    retorna novaRaiz
fim função
```



Rotação esquerda-direita

```
função rotacao_esquerdaDireita(raiz)

    raiz.esquerda = rotacao_esquerda(raiz.esquerda)
    novaRaiz = rotacao_direita(raiz)

    retorna novaRaiz
fim função
```

```
função rotacao_direita(raiz)
    novaRaiz = raiz.esquerda
    raiz.esquerda = novaRaiz.direita    ▷ Na Figura, a subárvore direita da nova raiz é nula.
    novaRaiz.direita = raiz

    retorna novaRaiz
fim função
```



Referências

- Drozdek, Adam. Estrutura de Dados e Algoritmos em C++ – Tradução da 4ª edição norte-americana. Disponível em: Minha Biblioteca, (2nd edição). Cengage Learning Brasil, 2018.