Árvores AVL

Prof. Carlos Gustavo Resque dos Santos

gustavoresqueufpa@gmail.com

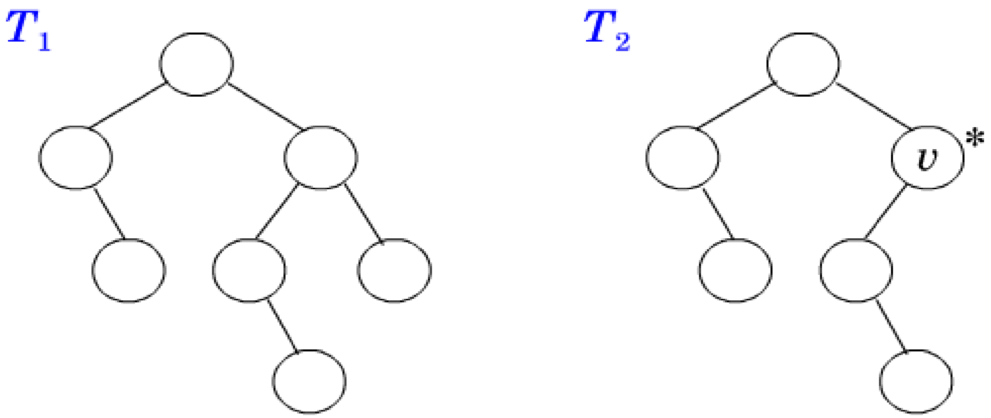
Autor dos Slides: Nelson Neto

Árvores AVL

- **Definição:** É uma árvore de busca binária altamente balanceada. Em tal árvore, **as alturas das duas sub-árvores a partir de cada nó diferem no máximo em uma unidade**.

- Também chamada de árvore balanceada pela altura.

- Se uma dada árvore é dito AVL, então todas as suas sub-árvores também são AVL.



T_1 é AVL

T_2 não é AVL
(possui nó v que está desregulado)

Árvores AVL

- As operações feitas em uma árvore AVL geralmente envolvem os mesmos algoritmos de uma árvore de busca binária.

- A altura de uma árvore AVL com *n* nós é *O*(*log n*). Assim, suas operações levam um tempo *O*(*log n*), **inclusive a de arrumação**.

- Para definir o balanceamento é utilizado um fator específico:

  - $FB(v) = h_e(v) - h_d(v)$

  - $FB(v)$: fator de balanceamento do nó *v*
  - $h_{\mathrm{e}}(v)$: altura da sub-árvore da esquerda
  - $h_d(v)$: altura da sub-árvore da direita

Árvores AVL

- Nós balanceados (ou **regulados**) são aqueles onde os valores de FB são $-1, 0$ ou $+1$.

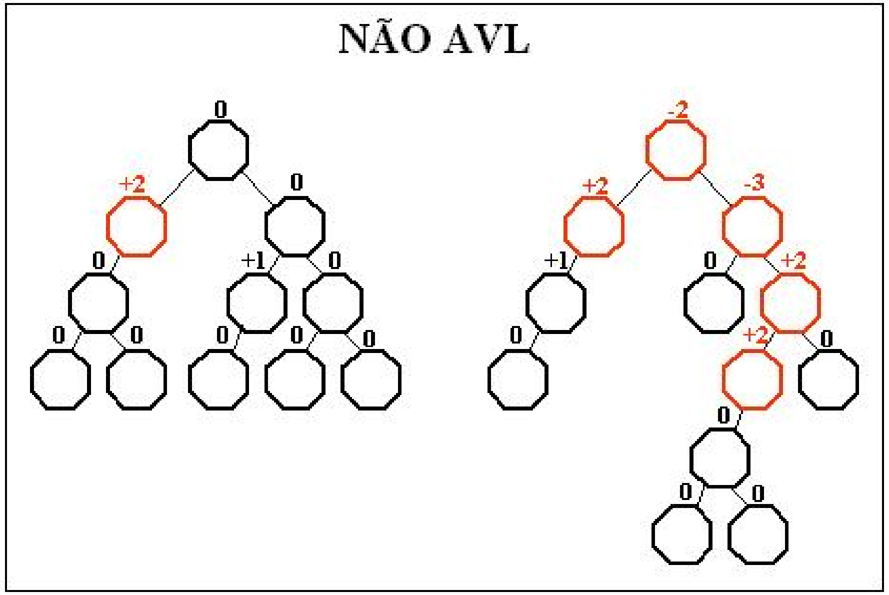
  - -1 : sub-árvore direita mais alta que a esquerda
  - 0 : sub-árvore esquerda igual a direita
  - $+1$: sub-árvore esquerda mais alta que a direita

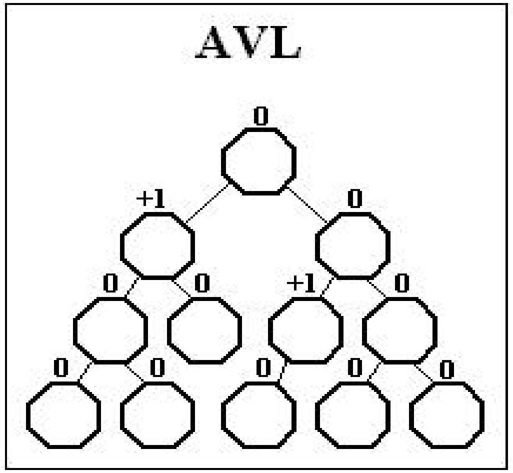
- Qualquer nó com FB diferente desses valores é dito **desregulado**.

  - > 1 : sub-árvore esquerda está desregulando o nó
  - < -1 : sub-árvore direita está desregulando o nó

- Se todos os nós da árvore são regulados, então a árvore é AVL.

Exemplo



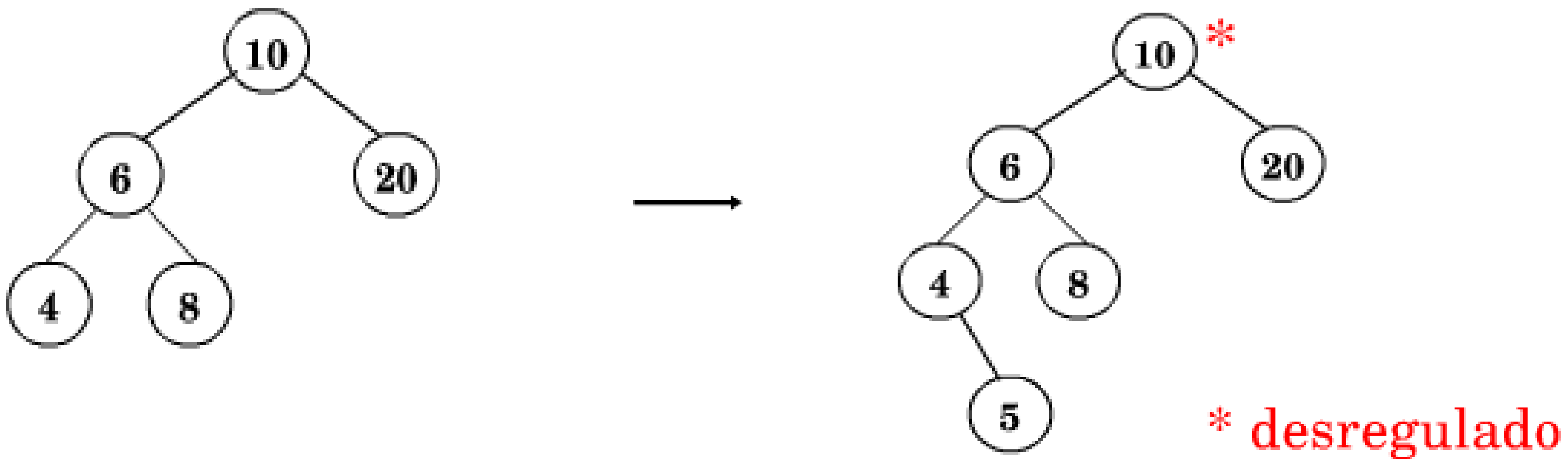


- Nos dois exemplos de árvore não AVL acima, percebe-se em vermelho os nós desregulados com $FB < -1$ ou $FB > +1$.

- Curiosidade: AVL = **A**delson-**V**elskii, G. e **L**andis, E. M.

Árvores AVL

- A inserção de um nó com chave 5 na árvore abaixo, faz com que ela deixe de ser AVL, com $FB(10) = +2$.
- E agora?



Árvores AVL

- Caso operações de inserção e remoção alterem o balanceamento da árvore, é necessário efetuar uma arrumação (ou **rotação**) para manter as propriedades da árvore AVL, tal que:

- a) O percurso em ordem fique inalterado em relação a árvore desbalanceada. Em outras palavras, a árvore continua a ser uma árvore de busca binária.

- b) A árvore modificada saiu de um estado de desbalanceamento para um estado de balanceamento.

Árvores AVL

- **Definição:** A operação de rotação altera o balanceamento de uma árvore *T*, garantindo a propriedade AVL e a sequência de percurso em ordem.

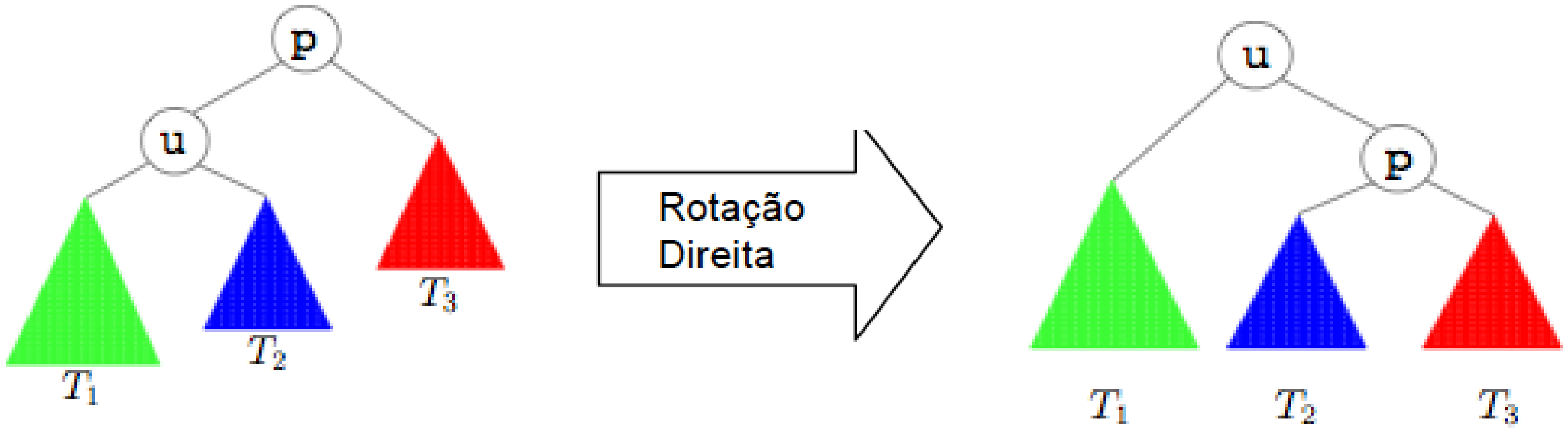
- Podemos definir 4 tipos diferentes de rotação:

  - Rotação à direita
  - Rotação à esquerda
  - Rotação dupla à direita
  - Rotação dupla à esquerda

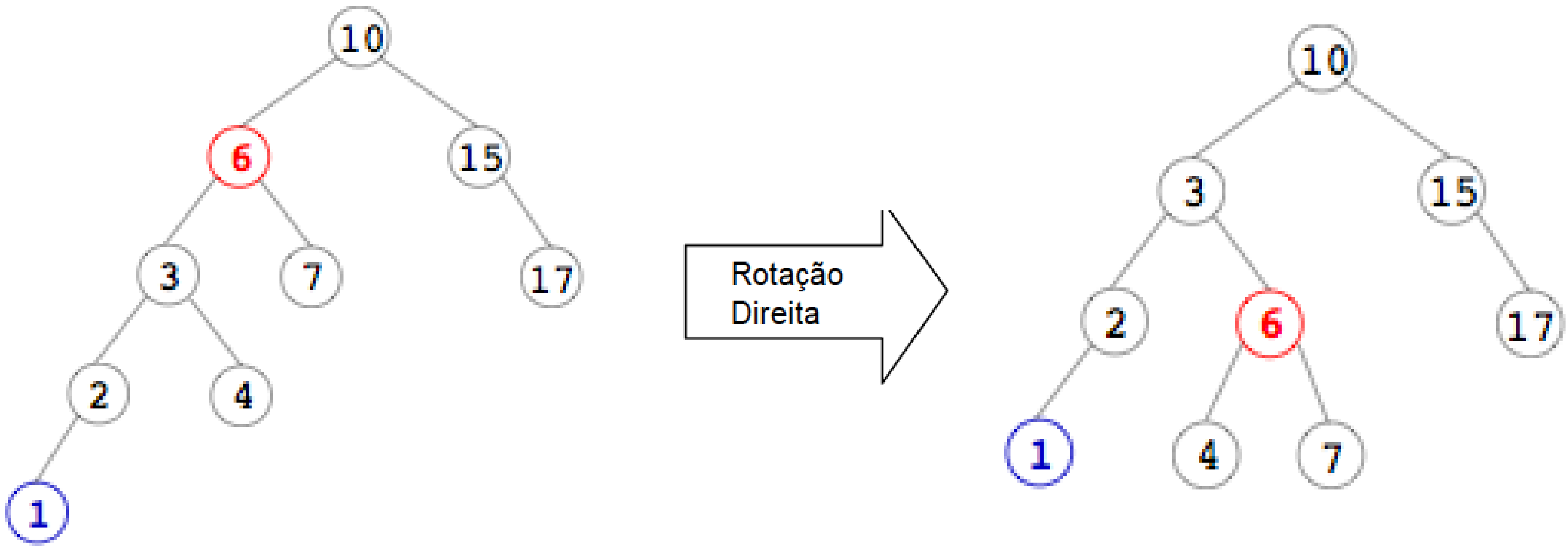
Árvores AVL

- Quando aplicar a rotação à direita?

$$\boxed{\begin{array}{l} h_E(p) > h_D(p) \\ h_E(u) > h_D(u) \end{array}}$$



Exemplo



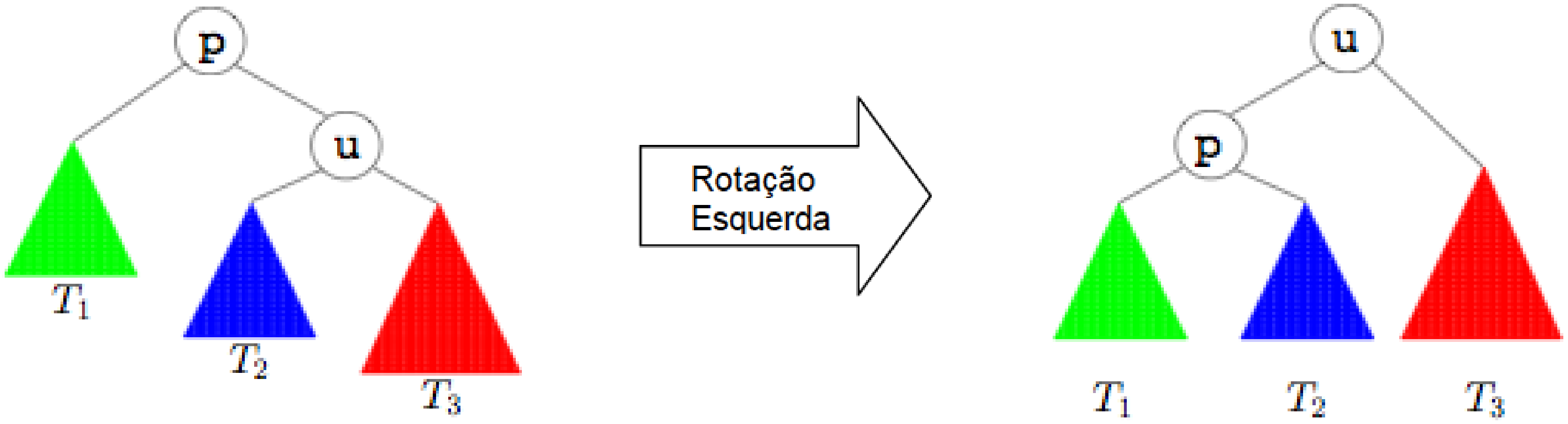
O nó 1 foi inserido na árvore, causando seu desbalanceamento

Árvore AVL

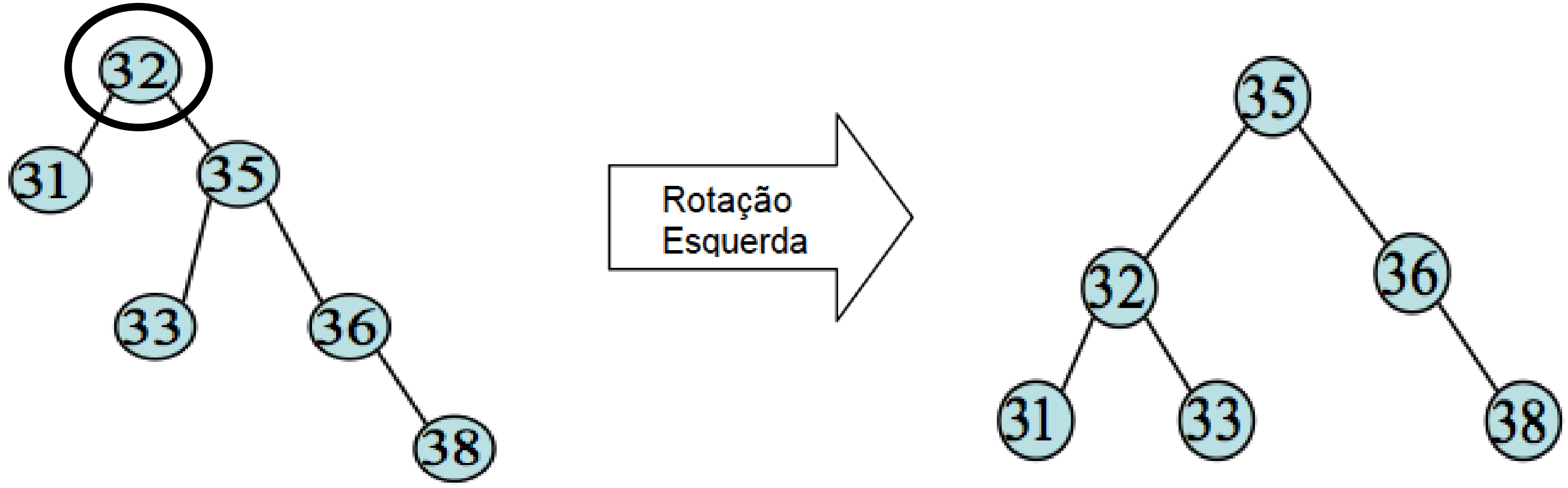
Árvores AVL

- Quando aplicar a rotação à esquerda?

$$h_D(p) > h_E(p)$$
$$h_D(u) > h_E(u)$$



Exemplo



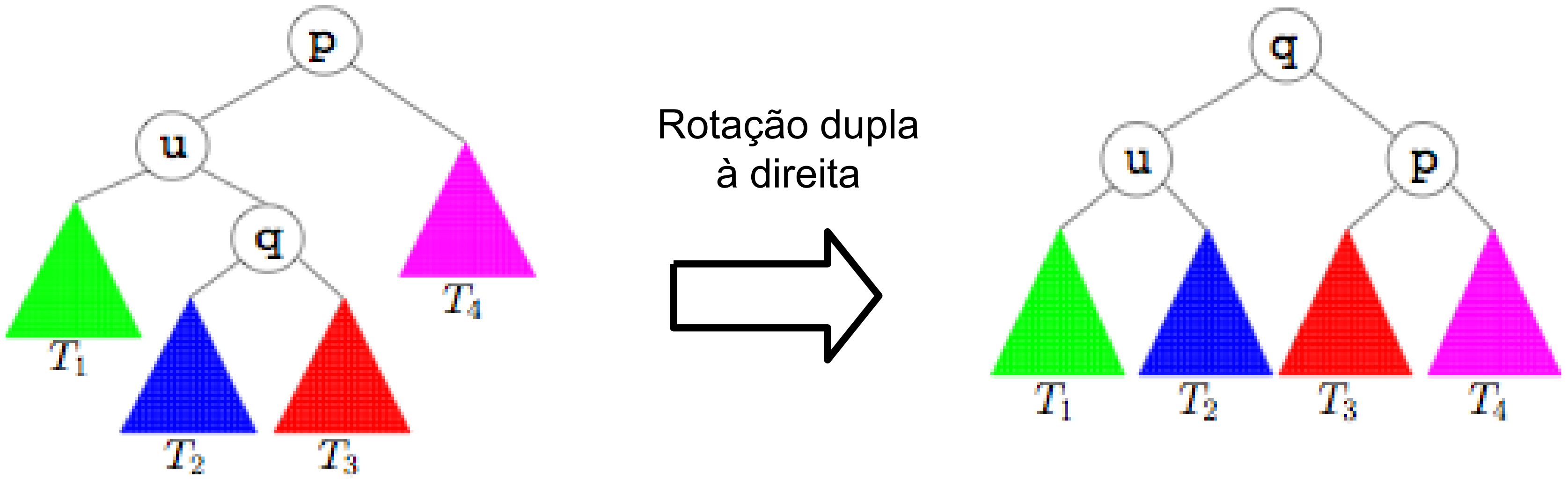
O nó 38 foi inserido na árvore, causando desbalanceamento no nó 32

Árvore AVL

Árvores AVL

- Quando aplicar a rotação dupla à direita?

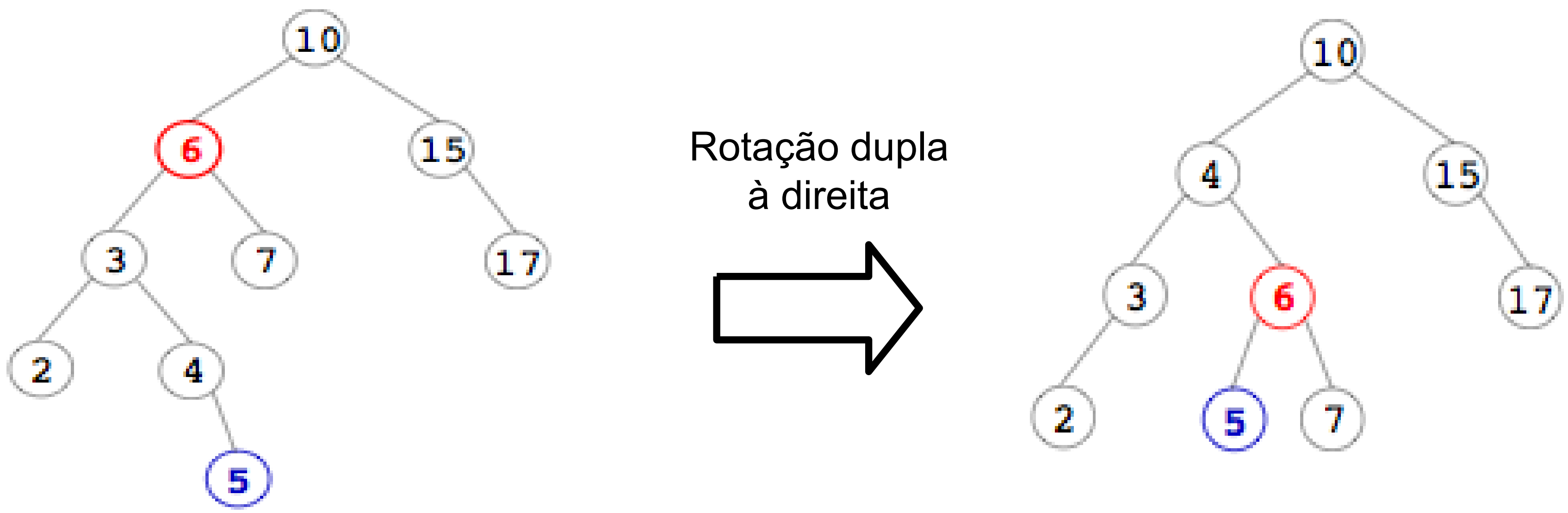
$$h_E(p) > h_D(p)$$
$$h_D(u) > h_E(u)$$



Exemplo

- Ao inserirmos o nó 5 na árvore, o nó 6 ficou desregulado.

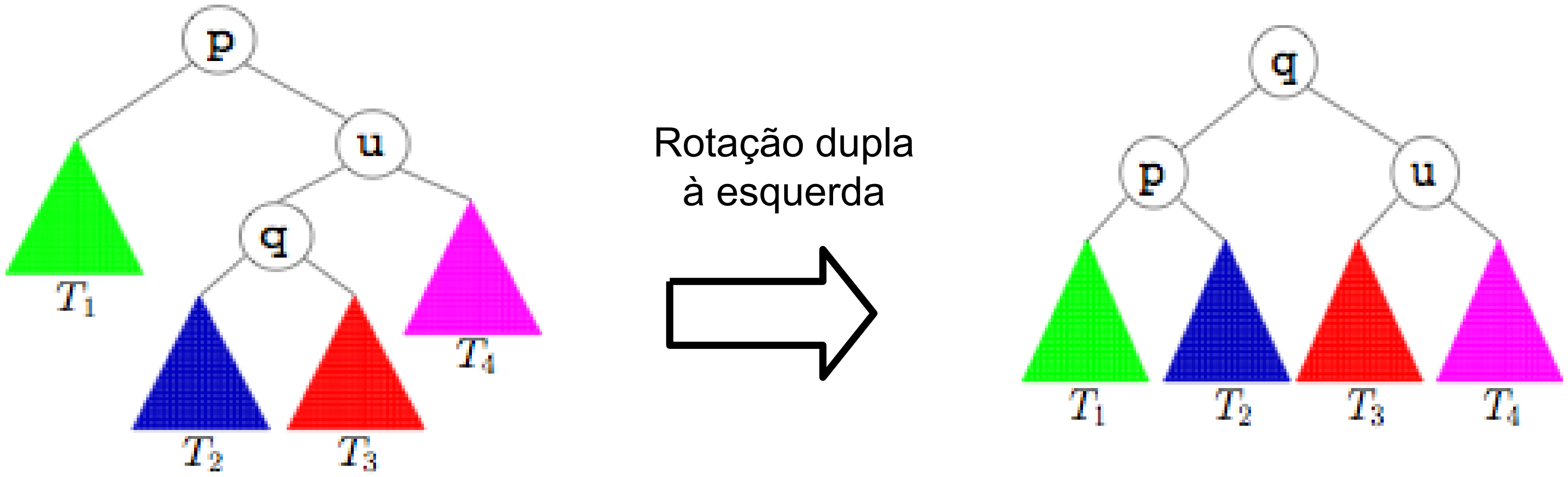
- Então, primeiro se faz uma rotação à esquerda e depois uma rotação à
- direita tendo o nó 4 como pivô.



Árvores AVL

- Quando aplicar a rotação dupla à esquerda?

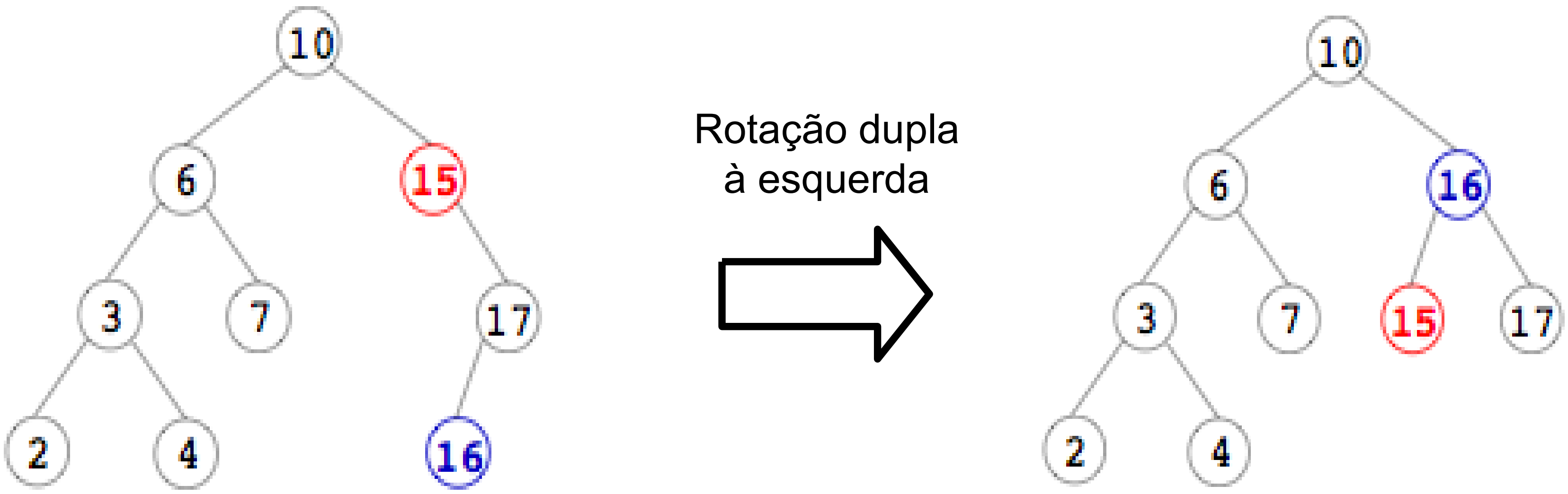
$$\boxed{\begin{array}{c} h_D(p) > h_E(p) \\ h_E(u) > h_D(u) \end{array}}$$



Exemplo

- Ao inserirmos o nó 16 na árvore, o nó 15 ficou desregulado.

- Então, primeiro se faz uma rotação à direita e depois uma rotação à
- esquerda tendo o nó 16 como pivô.



Árvores AVL

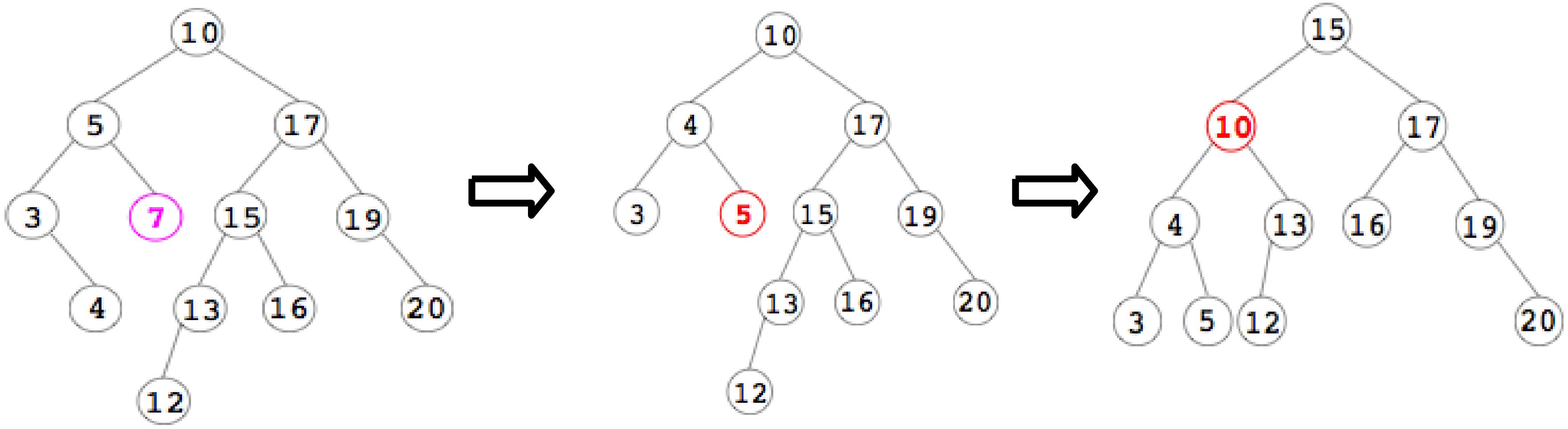
- Detalhes de implementação para inserção:
- 1. Realizar a operação de inserção como em uma árvore de busca binária comum: *O(log n)*
- 2. Verificar se existem nós desregulados. Para isso, deve-se atualizar o FB dos ancestrais do nó inserido: *O(log n)*
- 3. Se existir um nó desregulado, torná-lo regulado, aplicando a rotação necessária: *O(1)* (um simples ajuste de ponteiros)
  - Complexidade temporal total no pior caso: *O(log n)*

Árvores AVL

- Detalhes de implementação para remoção:
- 1. Realizar a operação de remoção como em uma árvore de busca binária comum: *O(log n)*
- 2. Verificar se existem nós desregulados. Para isso, deve-se atualizar o FB dos ancestrais do nó removido: *O(log n)*
- 3. Percorrer o caminho desde o pai do nó removido até a raiz, fazendo as operações de rotação apropriadas (pode ser mais de uma): *O(log n)*
- Obs: Se outro nó ocupar o lugar do nó removido, a análise deve começar no (antigo) pai desse nó.
  - Complexidade temporal total no pior caso: *O(log n)*

Exemplo

- Remova o nó com chave 7 da árvore AVL abaixo. Em seguida, aplique as rotações necessárias para garantir que a árvore resultante mantenha sua propriedade AVL.



Exercício

- Construir uma AVL com as chaves:
- (10, 20, 30, 5, 3, 50, 40, 70, 60, 90)

- Depois de construída a árvore AVL acima, remova as chaves:
- (20, 60, 90)
- e mantenha o balanceamento da árvore.