Estruturas de Dados

Árvores AVL

Prof. Nilton Luiz Queiroz Jr.

Árvores Binárias de Busca

- Nem todas árvores binárias de busca vão sempre estar estruturadas da melhor maneira possível;
- A maneira que os elementos são inseridos altera o formato da árvore;
- Em alguns casos, as árvores podem ter desempenho semelhante a uma lista;
- Como evitar que as ABBs tenham desempenho semelhante a uma lista?

Árvores Binárias de Busca

- Nem todas árvores binárias de busca vão sempre estar estruturadas da melhor maneira possível;
- A maneira que os elementos são inseridos altera o formato da árvore;
- Em alguns casos, as árvores podem ter desempenho semelhante a uma lista;
- Como evitar que as ABBs tenham desempenho semelhante a uma lista?
    - Usar árvores balanceadas!

Árvores binárias de busca

- No caso ideal, uma árvore binária de busca está balanceada;
    - Para todo nó na árvore, a diferença de altura entre seu filho direito e filho esquerdo é no máximo 1;
- Para a diferença de altura entre os filhos direito e esquerdo damos o nome de fator de balanceamento;
    - Fator de balanceamento(nó)=altura(subárvore direita) - altura(subárvore esquerda);

Árvores binárias de busca

- Uma árvore balanceada tem o fator de balanceamento 1, 0 ou -1 para **todo nó;**
- Em uma árvore balanceada se reduz significativamente a quantidade de comparações para encontrar um elemento;

Árvores binárias de busca

- Existem diversas maneiras de se construir uma árvore balanceada, dentre elas é reestruturar a árvore conforme os elementos são inseridos ou removidos;
  - Ou seja, se balanceia a árvore assim que ela é modificada, e se faz necessário um balanceamento;

Árvores AVL

- Um método de se balancear a árvore foi proposto por Adel’son-Vel’skii e Landis, o que fez com que as árvores que são balanceadas por esse método se chamassem árvores AVL;
- As árvores AVL são árvores de busca auto-balanceadas e tem como objetivo manter a árvore sempre balanceada;
    - Sempre que um nó tiver seu fator de balanceamento diferente dos valores 1, 0 e -1 são feitas operações nesse nó;

Árvores AVL

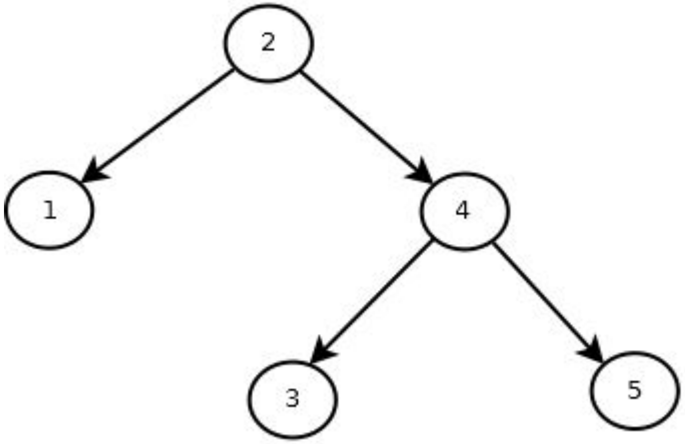
- A idéia de uma árvore AVL é a seguinte:
    - Após cada inserção ou remoção verificar se algum nó se tornou desbalanceado;
    - Caso tenha, aplica-se alguma rotação para balancear;
        - A rotação irá depender do novo estado da árvore;
- Existem 4 rotações possíveis:
    - Rotação simples à Esquerda;
    - Rotação simples à Direita;
    - Rotação esquerda-direita;
    - Rotação direita-esquerda;

Rotação simples à esquerda

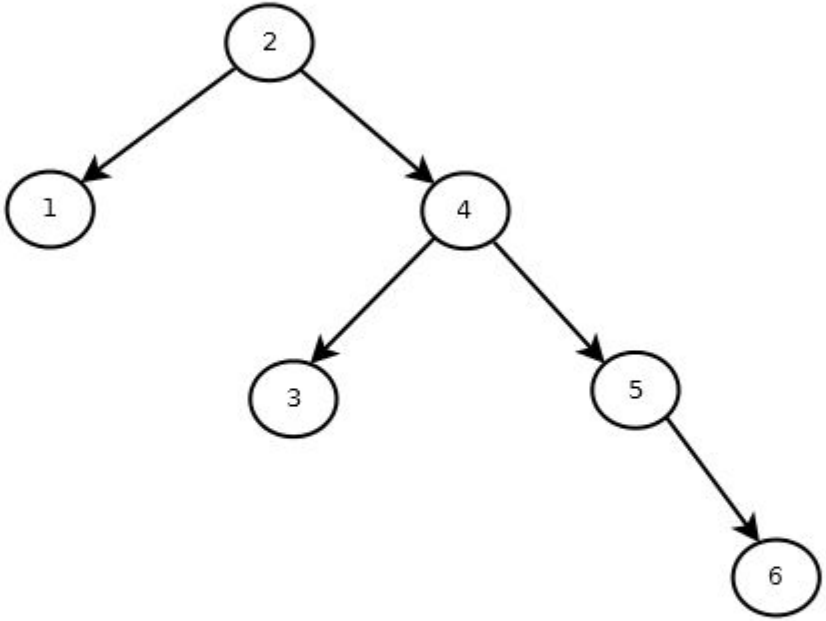
- A rotação simples à esquerda ocorre quando se insere um novo nó que causa o desbalanceamento do nó **p** e o filho direito de **p** tem a subárvore direita maior que a subárvore esquerda;
  - Ou seja, após a inserção temos:
    - FB(n) = +2;
    - FB(filho direito(n)) = +1;

Rotação simples à esquerda

- Dada a árvore à seguir, se inserirmos o valor 6, deverá ser feita uma rotação simples à esquerda para que ela se mantenha balanceada.



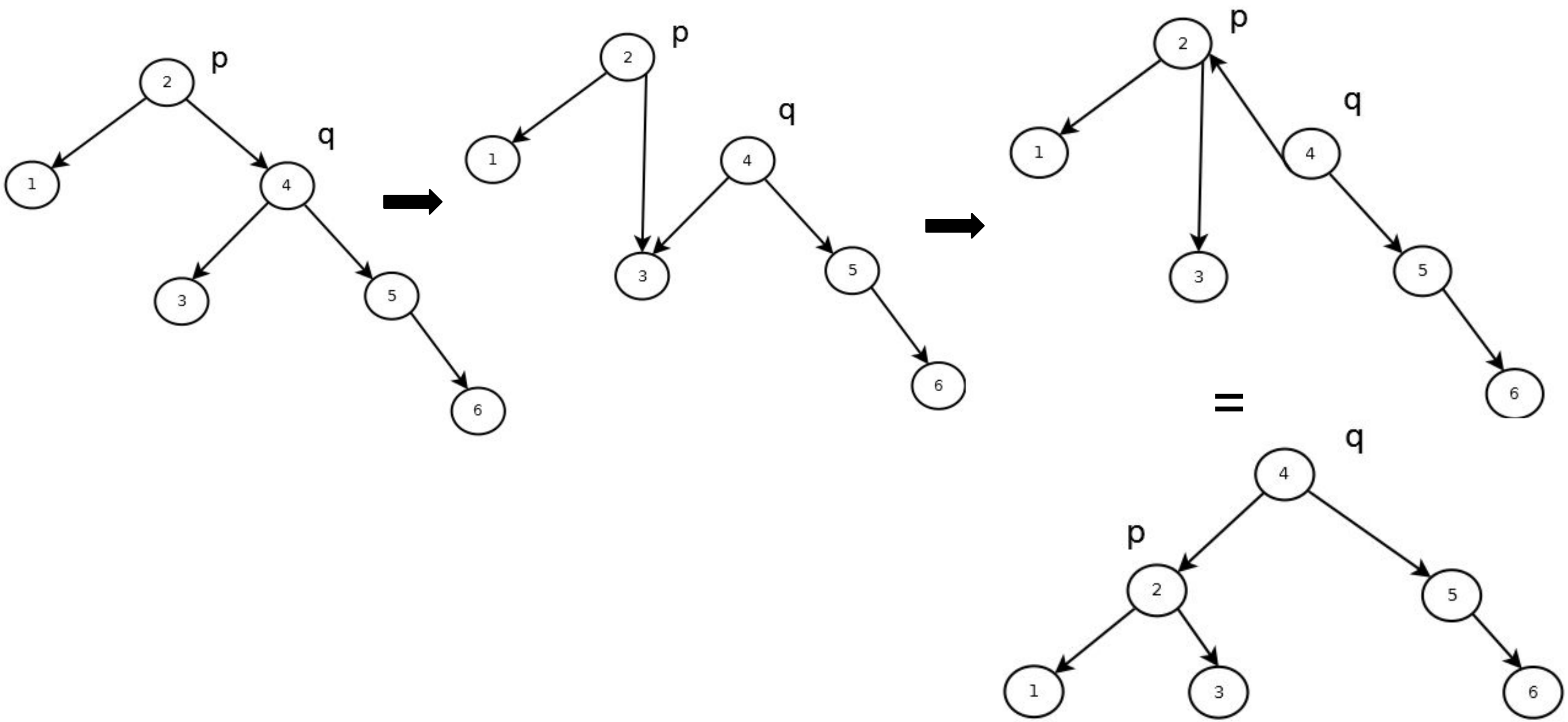
Rotação simples à esquerda



Rotação simples à esquerda

- Para aplicar a rotação simples à esquerda em um nó **p** devemos :
    - Localizar seu filho direito **q**;
    - Fazer com que o filho esquerdo de **q** se torne filho direito de **p**;
    - **p** se torne filho esquerdo de **q**;
    - O pai de **p** passa a ser o pai de **q;**

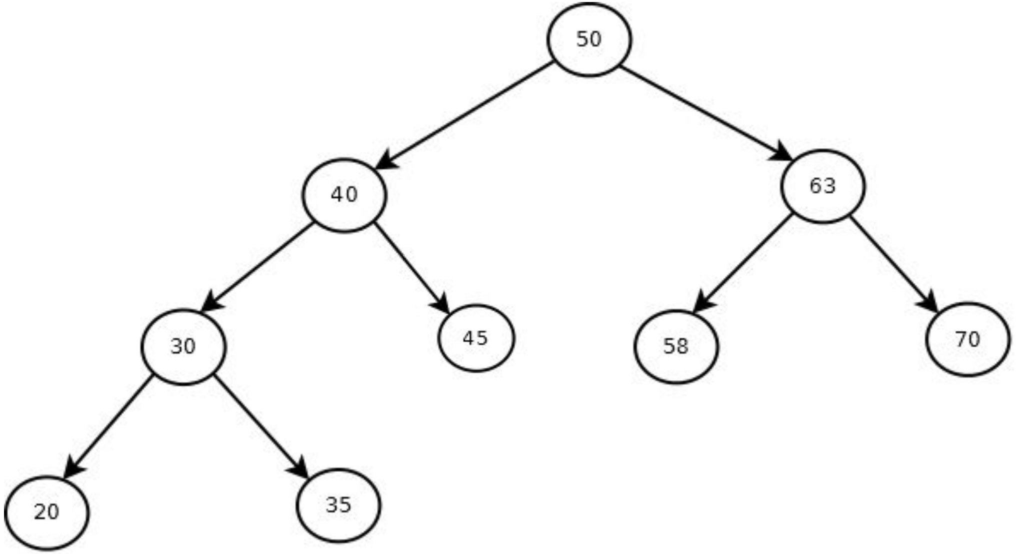
Rotação simples à esquerda



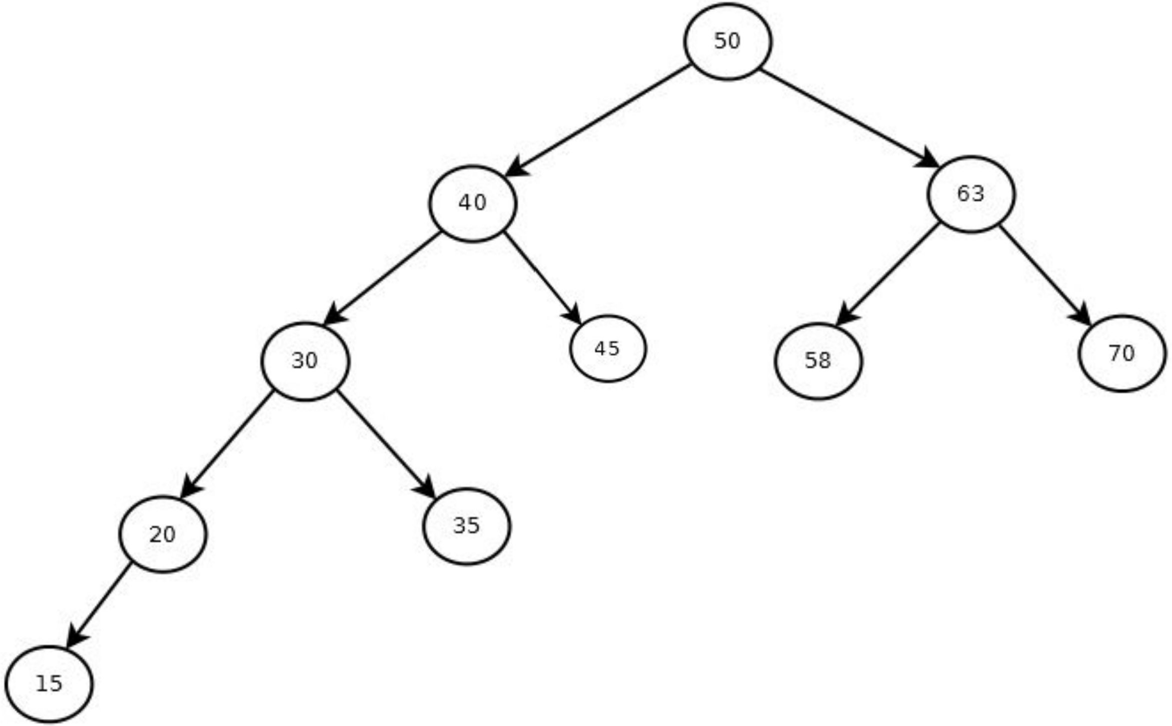
Rotação simples à direita

- A rotação simples à direita é o caso simétrico à rotação simples à esquerda;
- Ocorre quando se insere um novo nó que causa o desbalanceamento do nó **p** e o filho esquerdo de **p** tem a subárvore esquerda maior que a subárvore direita;
  - FB(p) = -2;
  - FB(filho esquerdo(p)) = -1;

Rotação simples à direita



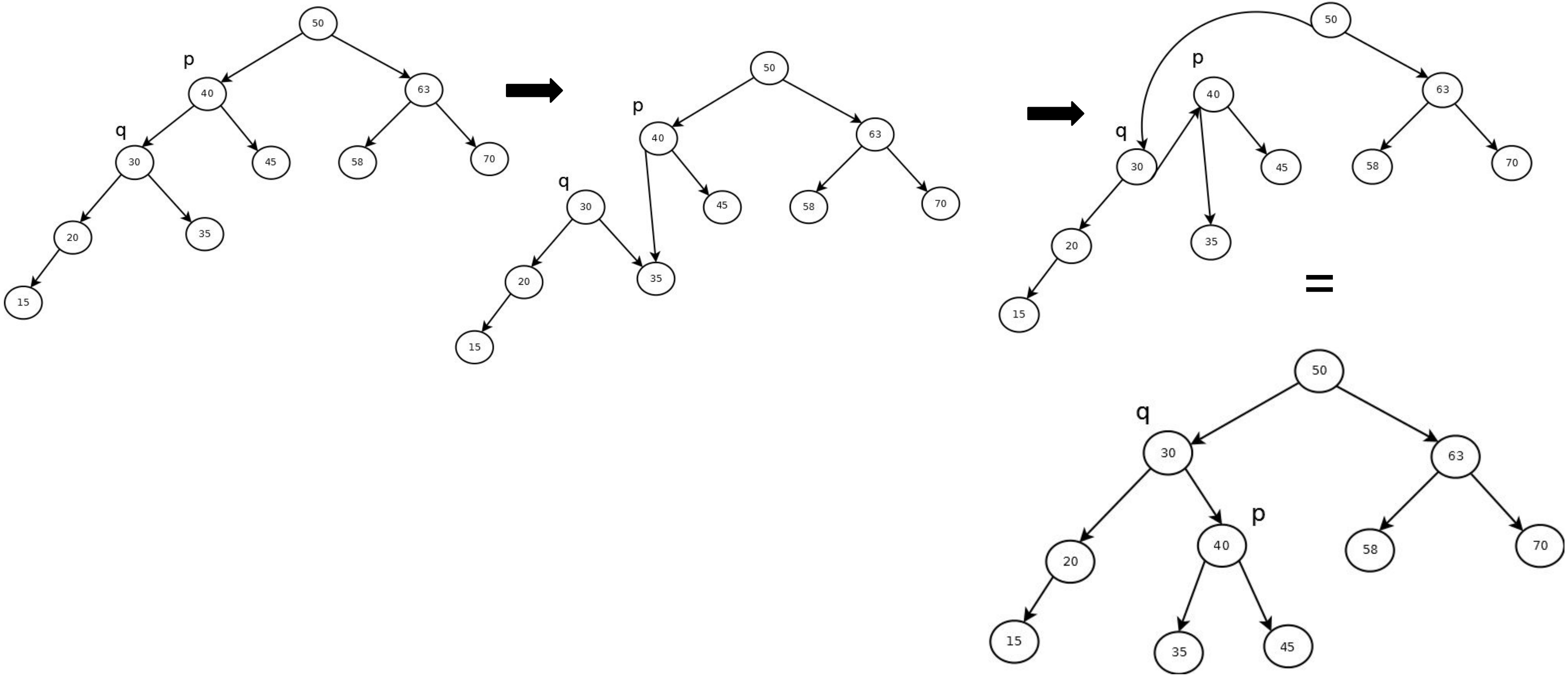
Rotação simples à direita



Rotação simples à direita

- Para aplicar a rotação simples à direita em um nó **p** devemos :
  - Localizar seu filho esquerdo **q**;
  - Fazer com que o filho direito de **q** se torne filho esquerdo de **p**;
  - **p** se torne filho direito de **q**;
  - O pai de **p** passa a ser o pai de **q;**

Rotação simples à direita



Exercício

1. Construa uma árvore AVL e uma binária de busca inserindo os valores na seguinte ordem:

   1, 2, 3, 4, 5, 6, 7

2. Construa uma árvore AVL e uma binária de busca inserindo os valores na seguinte ordem:

   7, 6, 5, 4, 3, 2, 1

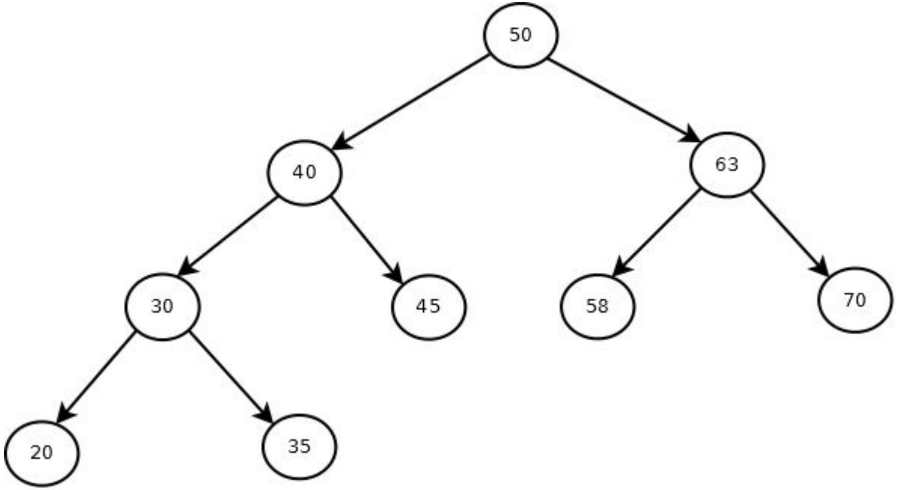
3. Qual a diferença entre as árvores AVL e binárias de busca dos exercícios 1 e 2?

Rotação esquerda-direita

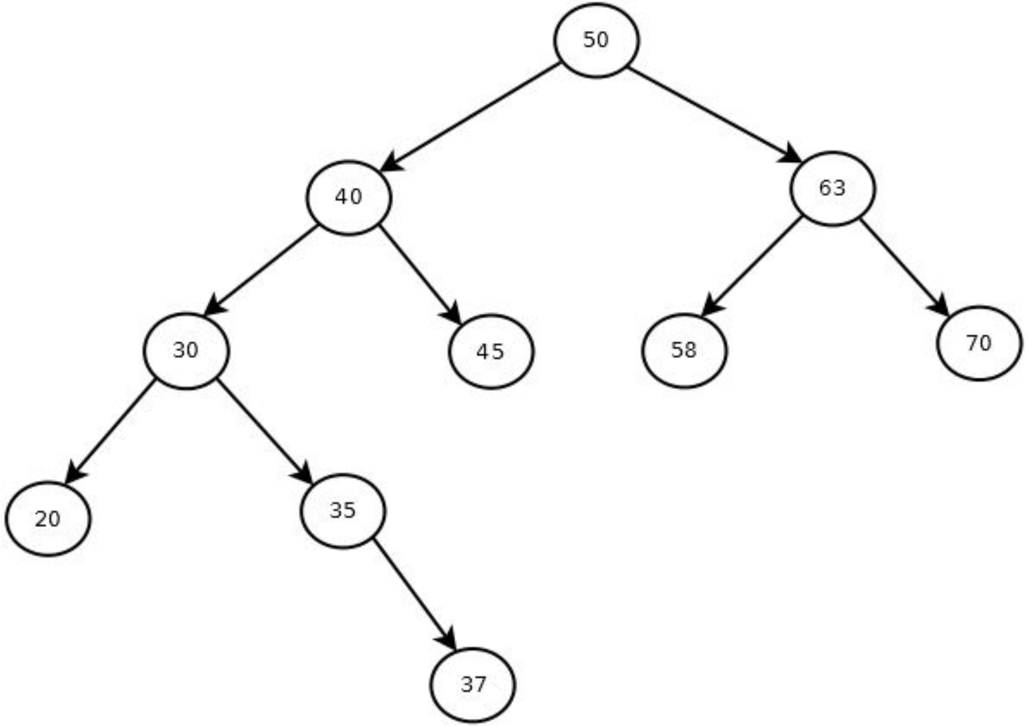
- A rotação esquerda-direita ocorre quando se insere um novo nó que causa o desbalanceamento de **p** e o filho esquerdo de **p** tem uma subárvore direita maior que a subárvore esquerda;
  - Ou seja, temos:
    - FB(p) = -2;
    - FB(filho esquerdo(p)) = +1;

Rotação esquerda-direita

- Inserir o nó 37 na árvore



Rotação esquerda-direita

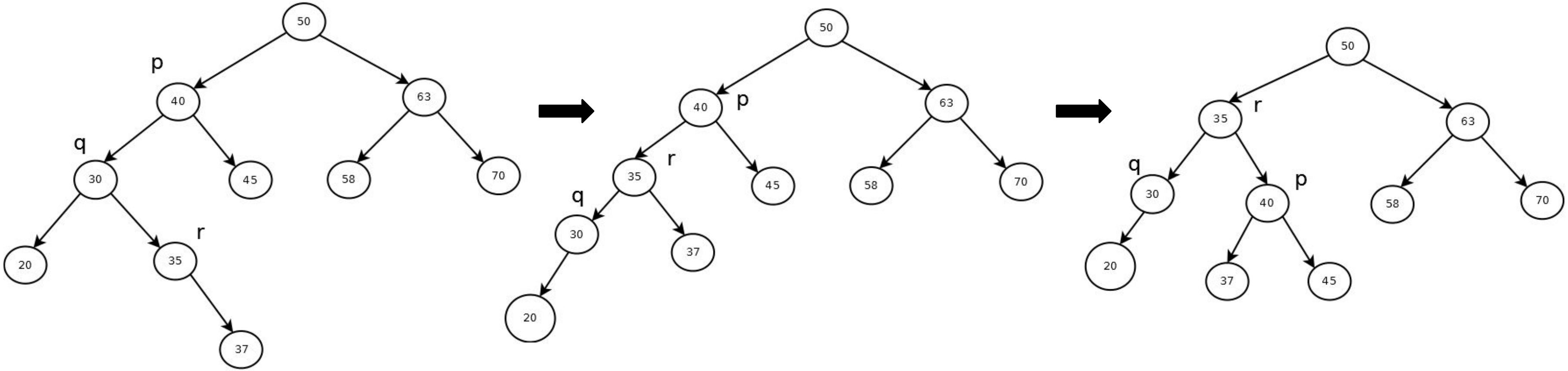


Rotação esquerda-direita

- Para aplicar a rotação esquerda-direita no nó **p** deve-se:
  - Localizar seu filho esquerdo **q**;
  - Aplicar uma rotação para a esquerda em **q**;
  - Aplicar uma rotação para a direita em **p**;

Obs: **q** terá um filho direito **r**, que se tornará o pai de **p** e **q**, ou seja, após uma rotação esquerda-direita **p** e **q** irão se tornar irmãos;

Rotação esquerda-direita

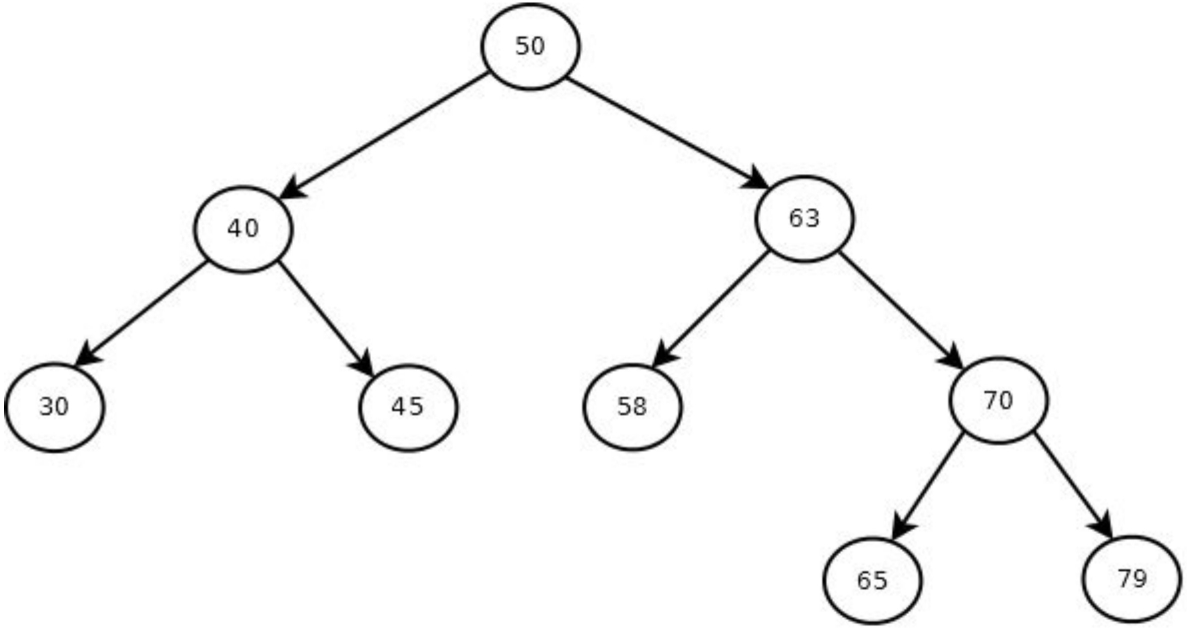


Rotação direita-esquerda

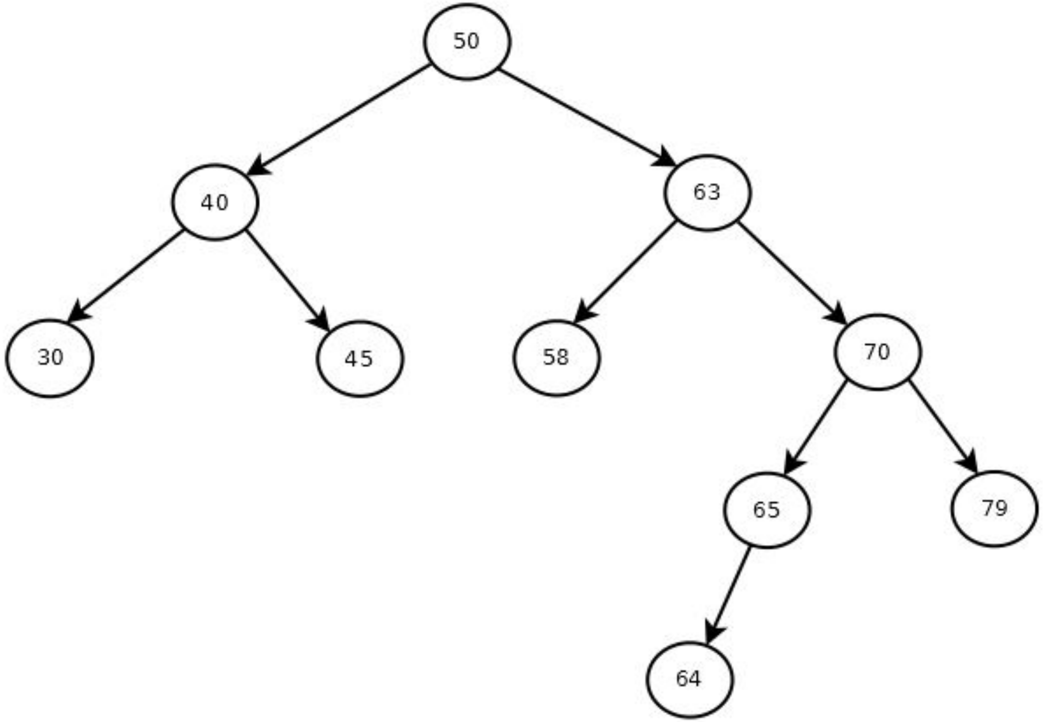
- A rotação direita-esquerda é o caso simétrico da rotação esquerda-direita;
- Ocorre quando se insere um novo nó que causa o desbalanceamento de **p** e o filho direito de **p** tem uma subárvore esquerda maior que a subárvore direita;
    - Ou seja, temos:
        - FB(p) = +2;
        - FB(filho esquerdo(p)) = -1;

Rotação direita-esquerda

- Inserir o elemento 64 na árvore:



Rotação direita-esquerda

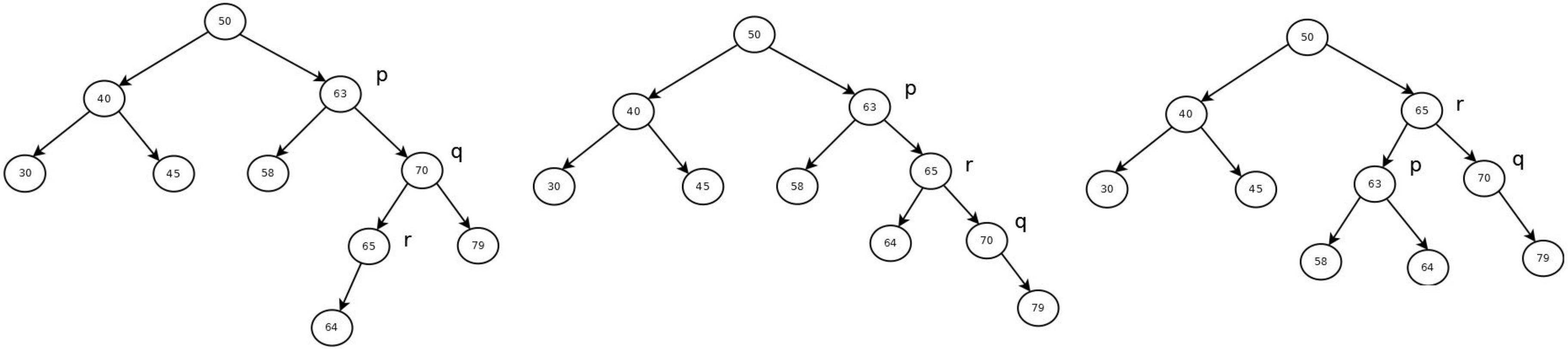


Rotação direita-esquerda

- Para aplicar a rotação esquerda-direita no nó **p** deve-se:
  - Localizar seu filho direito **q**;
  - Aplicar uma rotação para a direita em **q**;
  - Aplicar uma rotação para a esquerda em **p**;

Obs: **q** terá um filho esquerdo **r**, que se tornará o pai de **p** e **q**, ou seja, após uma rotação esquerda-direita **p** e **q** irão se tornar irmãos;

Rotação direita-esquerda



Árvores AVL

- Como as árvores estão sempre balanceadas temos as inserções em tempos:
  - Pior caso: $O(\log_2 n)$;
  - Rotação: $O(1)$;
    - Ambas as rotações, simples ou dupla, tem tempo constante;
- Apenas uma rotação precisa ser feita na inserção;

Exercícios

1. Construa as árvores AVL com a inserção dos seguintes elementos, e aponte antes de cada rotação, qual o tipo de rotação e em qual nó ela está sendo aplicada.

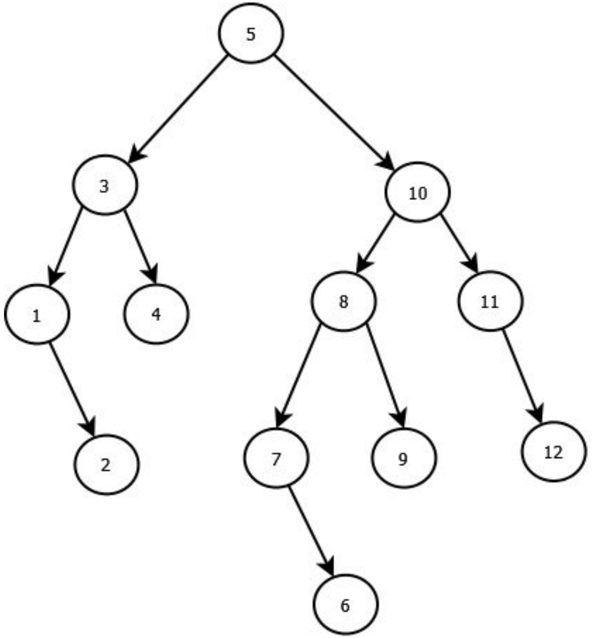
    10, 20, 30, 5, 7, 25, 28, 30, 15, 12

Remoção em árvore AVL

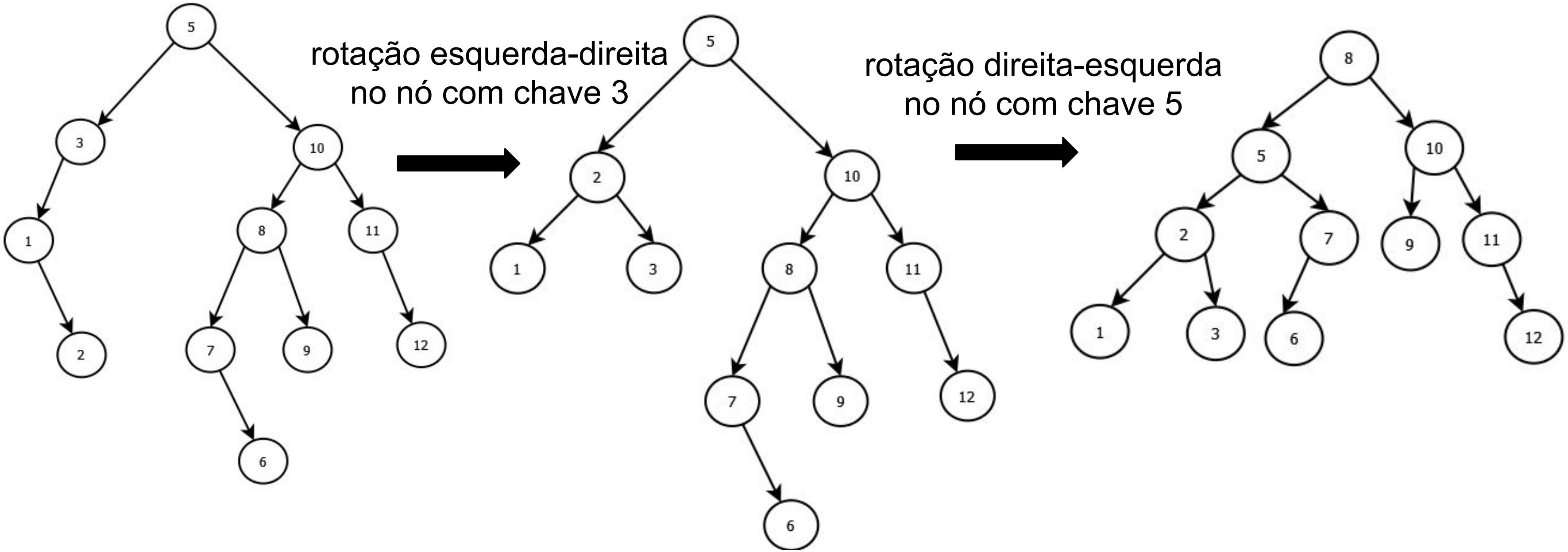
- Além da inserção, quando se remove elementos em uma árvore AVL deve-se mantê-la balanceada;
- A remoção em árvore AVL é feita da seguinte maneira:
    - Remove-se igual em uma ABB;
        - Porém, no caso do nó com dois filhos a remoção deve ser por substituição;
    - Para cada nó que é ancestral do nó removido faz-se o balanceamento, caso necessário;
        - Após voltar da chamada recursiva faz-se o balanceamento;

Remoção em árvore AVL

- Remova o elemento 4 da árvore a seguir



Remoção em Árvores AVL



Remoção em Árvores AVL

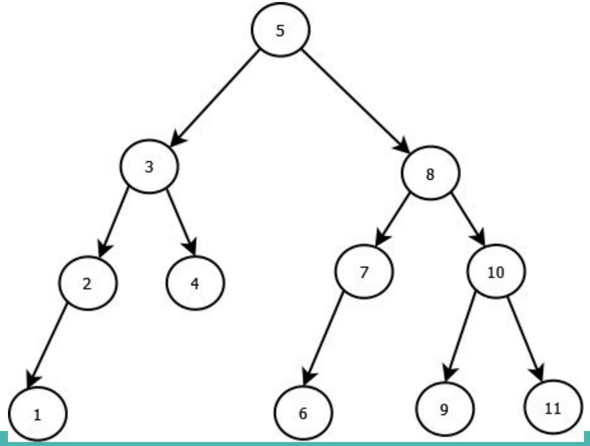
- As remoções em uma árvore AVL levam:
  - Pior caso: $O(\log_2 n)$;
  - Rotação: $O(\log_2 n)$;
    - Como a rotação não irá ocorrer somente em um nó, e sim em todos so ancestrais do nó que foi removido, podemos teremos $O(\log_2 n)$ rotações. Como as rotações tem tempo constante, o tempo gasto é $O(\log_2 n)$;

Exercícios

1. Construa uma árvore AVL inserindo as chaves na seguinte ordem, quando houver rotação informe qual rotação e em qual nó ela foi aplicada:

    4, 5, 7, 2, 1, 3, 6

2. Considere a seguinte árvore:



Remova os nós com as chaves a seguir

4, 8, 6, 5, 2, 1 e 7