Estrutura de Dados II
Ciência da Computação

Prof. André Kishimoto
2023

Árvore AVL

- Artigo de 1962: Один алгоритм организации информации (“Um algoritmo para organização de informações”)
- Escrito por dois matemáticos soviéticos: Georgii Maximovich **A**del’son-**V**el’skii e Evgenii Mikhailovich **L**andis.
- Um algoritmo para manter uma BST balanceada.
  - Há autores que chamam a árvore AVL de *height-balanced tree*, ou árvore com altura balanceada.
- Uma árvore AVL é uma BST, então podemos aproveitar os conceitos da BST (e implementação, com os devidos ajustes).

Árvore AVL

- Uma BST é dita **perfeitamente balanceada** quando:
  - As alturas das subárvores esquerda e direita da raiz são iguais
  - E as subárvores esquerda e direita da raiz também são BST perfeitamente balanceadas.
- Uma **árvore AVL** pode ser definida como uma BST cuja:
  - Diferença de altura entre as subárvores esquerda e direita da raiz é de no máximo 1
  - E que as subárvores esquerda e direita da raiz também são árvores AVL.
- Qualquer BST que satisfaça a propriedade descrita no item anterior é uma árvore AVL.

Árvore AVL

- A diferença de altura entre as subárvores esquerda e direita define o **balanceamento (ou balanço) da árvore**.
- O algoritmo da árvore AVL mantém um registro da diferença de altura entre cada subárvore (esquerda e direita).
- Conforme os nós são inseridos ou removidos da árvore, o balanço de cada subárvore, do local em que houve a alteração na árvore até a raiz, é atualizado.
- Caso a árvore fique desbalanceada, é necessário ajustar os nós da árvore de tal forma que ela volte a ficar balanceada.

Fator de balanceamento

- ***Balance factor*** (**BF**) é calculado para cada nó e usado para sabermos se uma árvore está balanceada ou não.
- Cálculo do fator de balanceamento de um nó:

$$bf(n) = hr - hl$$

Sendo:

- ***n***: Nó cujo fator de balanceamento (***bf***) estamos calculando.
- ***hr***: Altura da subárvore direita *ou* -1 se não há subárvore direita.
- ***hl***: Altura da subárvore esquerda *ou* -1 se não há subárvore esquerda.

Fator de balanceamento

- Árvore desbalanceada: se ***bf(n) > 1*** ou ***bf(n) < -1***.
- ***bf(n) = 0*** → subárvores esquerda e direita possuem a mesma altura.
- ***bf(n) = -1*** → subárvore esquerda é mais alta que a direita (“lado esquerdo mais pesado do que o lado direito”).
- ***bf(n) = +1*** → subárvore direita é mais alta que a esquerda (“lado direito mais pesado do que o lado esquerdo”).

- Atenção! Valores acima são válidos para ***bf(n) = hr – hl***.
  - Algumas referências usam ***bf(n) = hl – hr***.

Operações de rotação

- Operações de rotação são usadas para balancear uma árvore.
- Devem ser realizadas após uma alteração na árvore (inserção/remoção) que a deixe desbalanceada.
- Existem **quatro tipos de rotação para o balanceamento de árvore**, conforme o tipo de desbalanceamento da árvore:
  - Rotação para esquerda (Rotação LL, Árvore RR).
  - Rotação para direita (Rotação RR, Árvore LL).
  - Rotação esquerda-direita (Rotação LR, Árvore LR).
  - Rotação direita-esquerda (Rotação RL, Árvore RL).

Operações de rotação

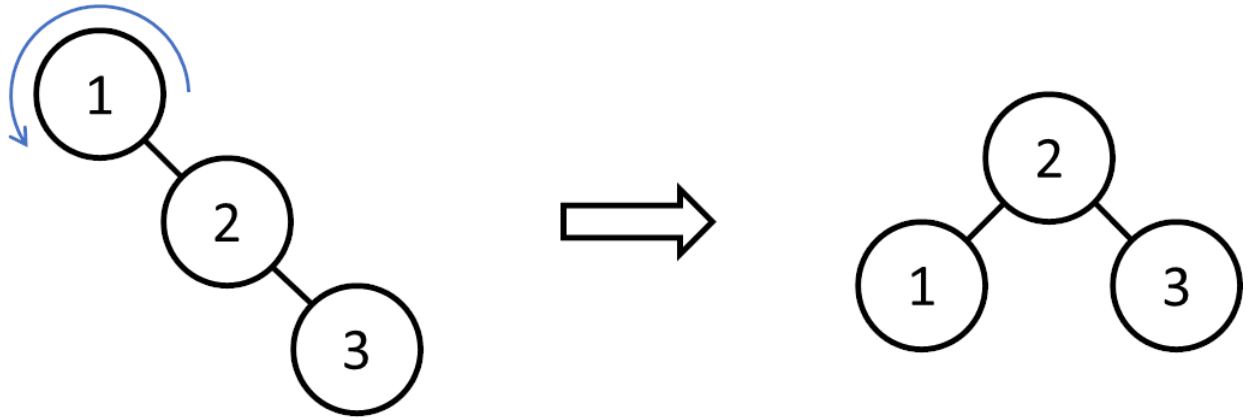
Atenção!
Rotação != Árvore

Exemplo: Rotação para esquerda (Rotação LL, Árvore RR).
“Desbalanceamento ocorre na subárvore direita da subárvore direita (filho direito) de um nó.”

- É uma árvore RR: árvore direita-direita (*right-right tree*, *RR tree*).
- Nesse desbalanceamento, aplicamos uma rotação para esquerda para balancear a árvore.
  - Rotação LL: *left-left rotation*, *LL rotation*.

Rotação para esquerda

- Rotação LL, Árvore RR.
- Aplicada quando o desbalanceamento ocorre na subárvore **direita** da subárvore **direita** (filho direito) de um nó.

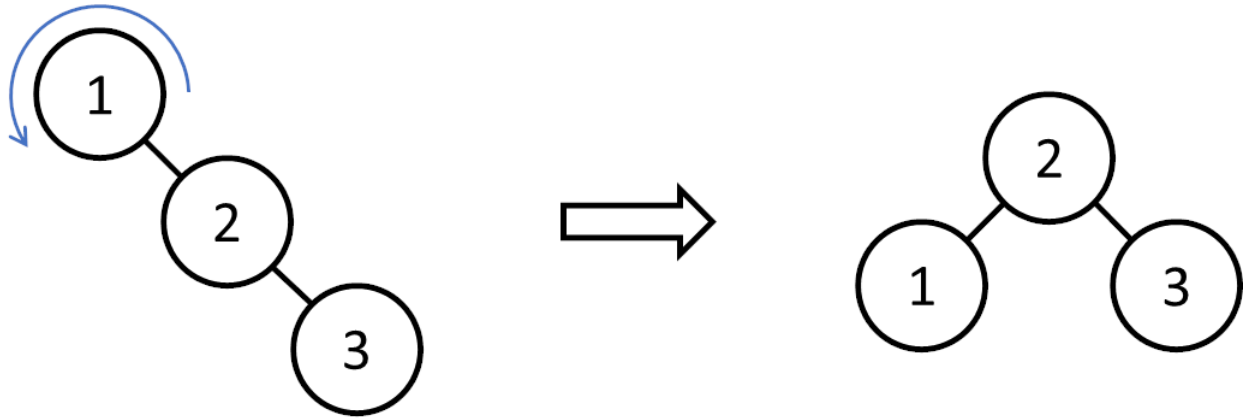


BF(1) = +2 e BF(2) = +1

Rotação para esquerda

Rotação à esquerda em torno do pai:

- Nó 2 se torna raiz.
- Nó 2 se torna pai do nó 1.
- Filho direito do nó 1 recebe filho esquerdo do nó 2.
- Nó 1 se torna pai do filho esquerdo do nó 2.
- Nó 1 se torna filho esquerdo do nó 2.



Rotação para direita

- Rotação RR, Árvore LL.
- Aplicada quando o desbalanceamento ocorre na subárvore **esquerda** da subárvore **esquerda** (filho esquerdo) de um nó.

BF(3) = -2 e BF(2) = -1

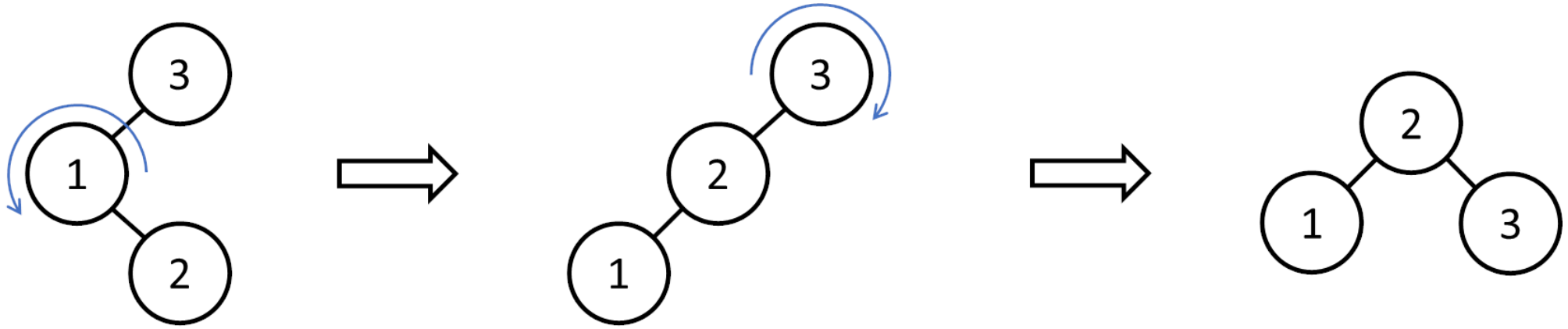
Rotação para direita

Rotação à direita em torno do pai:

- Nó 2 se torna raiz.
- Nó 2 se torna pai do nó 3.
- Filho esquerdo do nó 3 recebe filho direito do nó 2.
- Nó 3 se torna pai do filho direito do nó 2.
- Nó 3 se torna filho direito do nó 2.

Rotação esquerda-direita

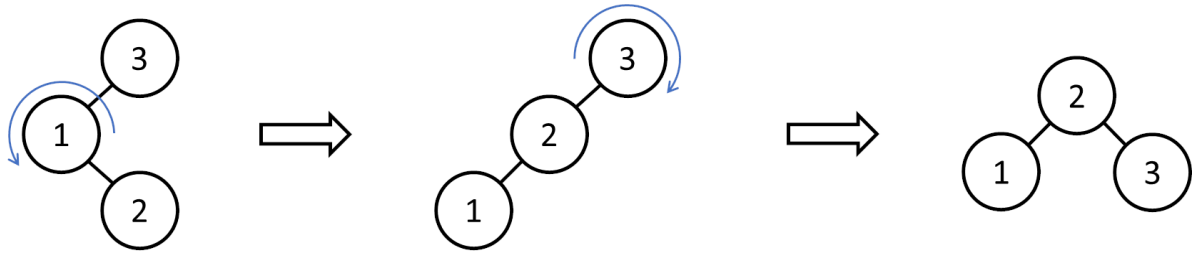
- Rotação LR, Árvore LR, rotação dupla à direita.
- Aplicada quando o desbalanceamento ocorre na subárvore **direita** da subárvore **esquerda** (filho esquerdo) de um nó.



BF(3) = -2 e BF(1) = +1

Rotação esquerda-direita

Rotação à esquerda em torno do filho esquerdo e, em seguida, rotação à direita em torno do pai/raiz:



Rotação à esquerda do nó 1:

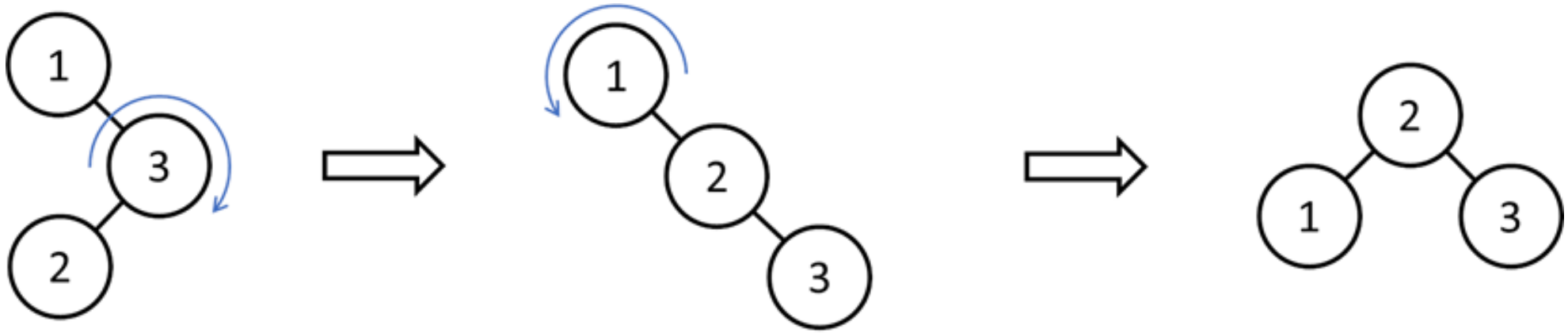
- Nó 2 se torna filho esquerdo do nó 3.
- Nó 2 se torna pai do nó 1.
- Filho direito do nó 1 recebe filho esquerdo do nó 2.
- Nó 1 se torna pai do filho esquerdo do nó 2.
- Nó 1 se torna filho esquerdo do nó 2.

Rotação à direita do nó 3:

- Nó 2 se torna raiz.
- Nó 2 se torna pai do nó 3.
- Filho esquerdo do nó 3 recebe filho direito do nó 2.
- Nó 3 se torna pai do filho direito do nó 2.
- Nó 3 se torna filho direito do nó 2.

Rotação direita-esquerda

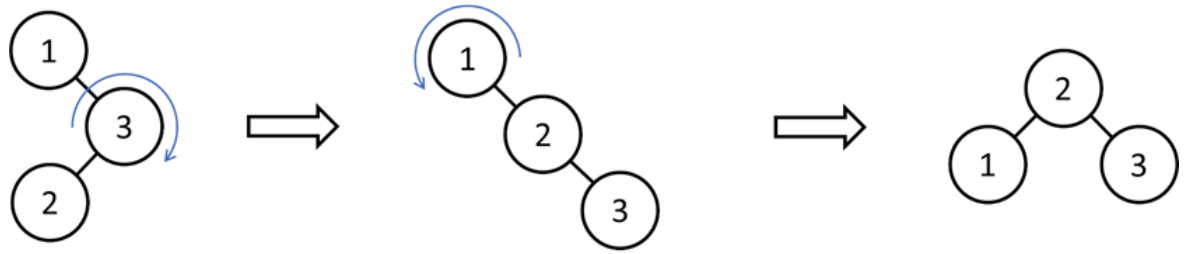
- Rotação RL, Árvore RL, rotação dupla à esquerda.
- Aplicada quando o desbalanceamento ocorre na subárvore **esquerda** da subárvore **direita** (filho direito) de um nó.



BF(1) = +2 e BF(3) = -1

Rotação direita-esquerda

Rotação à direita em torno do filho e, em seguida, rotação à esquerda em torno do pai/raiz:



Rotação à direita do nó 3:

- Nó 2 se torna filho direito do nó 1.
- Nó 2 se torna pai do nó 3.
- Filho esquerdo do nó 3 recebe filho direito do nó 2.
- Nó 3 se torna pai do filho direito do nó 2.
- Nó 3 se torna filho direito do nó 2.

Rotação à esquerda do nó 1:

- Nó 2 se torna raiz.
- Nó 2 se torna pai do nó 1.
- Filho direito do nó 1 recebe filho esquerdo do nó 2.
- Nó 1 se torna pai do filho esquerdo do nó 2.
- Nó 1 se torna filho esquerdo do nó 2.