

# **de Dados - Aula 05 - Árvores Balanceadas (AVL)**

Prof. André Gustavo Hochuli

[gustavo.hochuli@pucpr.br](mailto:gustavo.hochuli@pucpr.br)

[aghochuli@ppgia.pucpr.br](mailto:aghochuli@ppgia.pucpr.br)

# Plano de Aula

- **Discussão Árvores Binárias / Código Morse**
- **Árvores Balanceadas (AVL)**

# Árvores Binárias Balanceadas (AVL)

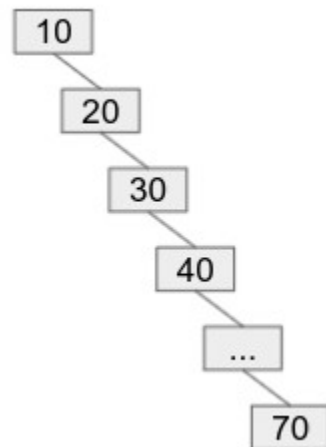
- **Adelson Velsy e Landis (AVL)**



Adelson-Velsky, Georgy; Landis, Evgenii (1962). "An algorithm for the organization of information". Proceedings of the USSR Academy of Sciences (in Russian). 146: 263–266

- **Inclusão sequencial em Binary Search Tree (BST)**

- **10,20,30,40,50,60,70**



Desbalanceada

# Árvores Binárias Balanceadas (AVL)

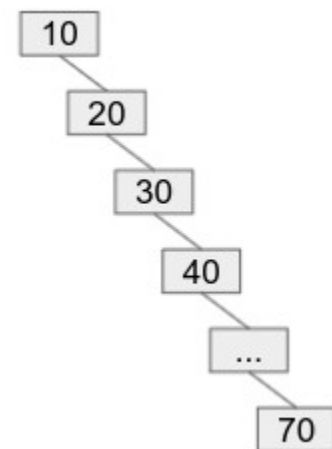
- **Adelson Velsy e Landis (AVL)**



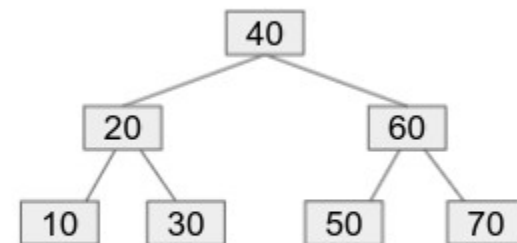
Adelson-Velsky, Georgy; Landis, Evgenii (1962). "An algorithm for the organization of information". Proceedings of the USSR Academy of Sciences (in Russian). 146: 263–266

- **Inclusão sequencial em Binary Search Tree (BST)**

- **10,20,30,40,50,60,70**



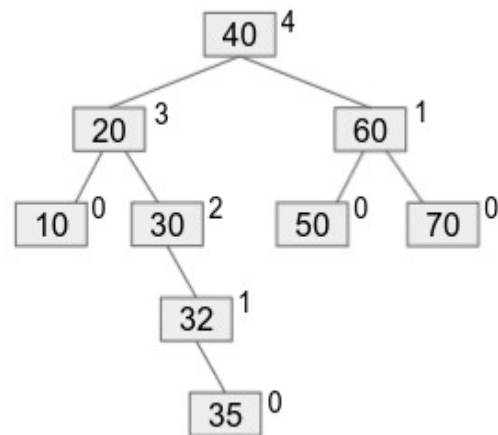
Desbalanceada



Balanceada

# Árvores Binárias Balanceadas (AVL)

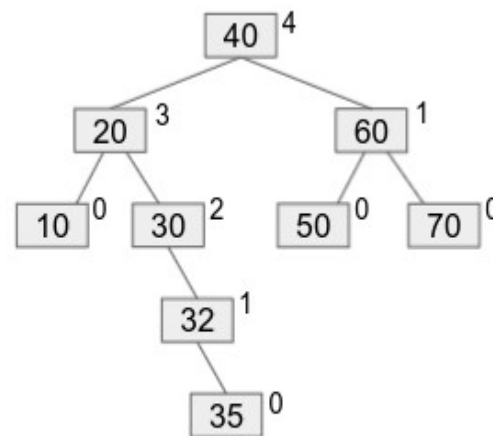
- **Árvore balanceada:** altura do lado esquerdo da árvore não difere mais de  $\pm 1$  do lado direito.
- **Ou seja, Fator de Balanceamento é dado por:**
  - $FB( ) = he - hd \rightarrow |FB| \leq 1 == \text{Nodo balanceado}$ 
    - $FB( ) == 0 \rightarrow he == hd$
    - $FB( ) > 0 \rightarrow he > hd$
    - $FB( ) < 0 \rightarrow he < hd$



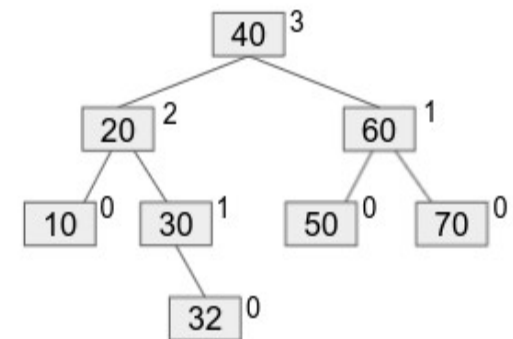
Desbalanceada

# Árvores Binárias Balanceadas (AVL)

- **Árvore balanceada:** altura do lado esquerdo da árvore não difere mais de  $\pm 1$  do lado direito.
- **Ou seja, Fator de Balanceamento é dado por:**
  - $FB( ) = he - hd \rightarrow |FB| \leq 1 == \text{Nodo balanceado}$
  - $FB( ) == 0 \rightarrow he == hd$
  - $FB( ) > 0 \rightarrow he > hd$
  - $FB( ) < 0 \rightarrow he < hd$

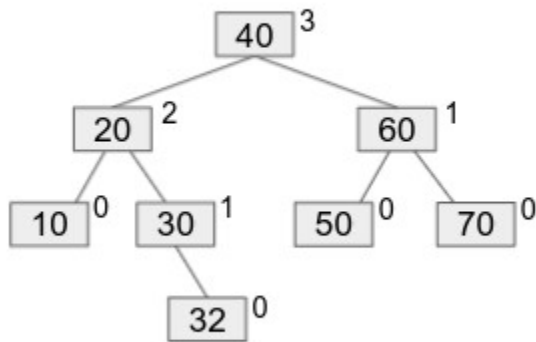


Desbalanceada



Balanceada

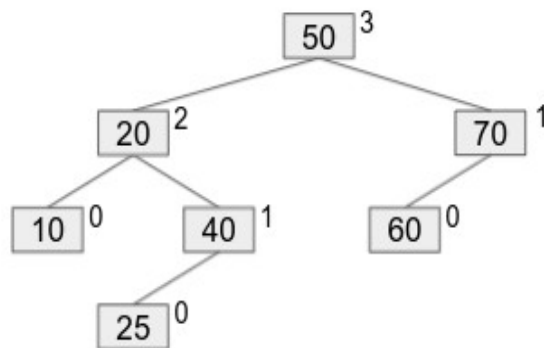
# Árvores Binárias Balanceadas (AVL)



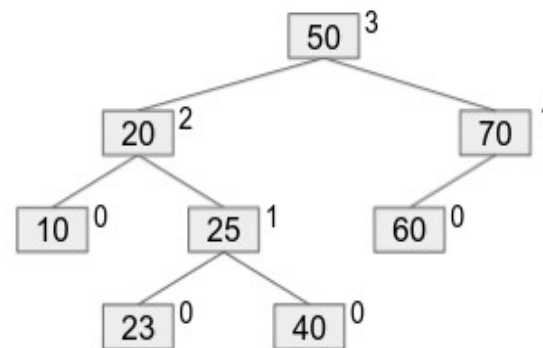
```
class Node:
    def __init__(self, data):
        self.data = data # Assign data
        self.left = None # Initialize as None
        self.right = None # Initialize as None
        self.height = 1
```

# Árvores Binárias Balanceadas (AVL)

- **Inclusões:**
  - Inserir 23 e 65
  - Balancear se  $|FB| > 1$



Após inclusão e  
balanceamento

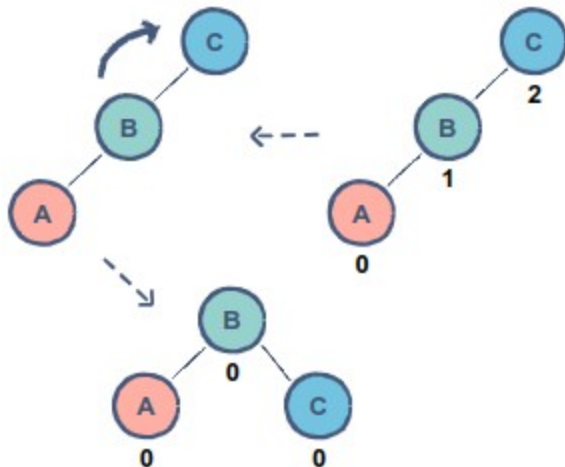




# Árvores Binárias Balanceadas (AVL)

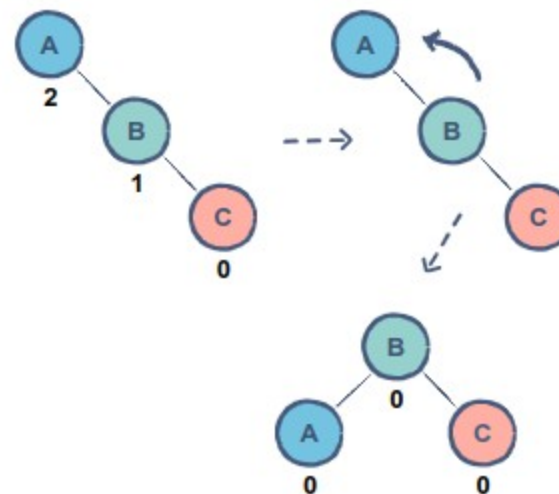
- Rotação a Direita

- $FB > 1$  e valor inserido a esquerda



- Rotação a Esquerda

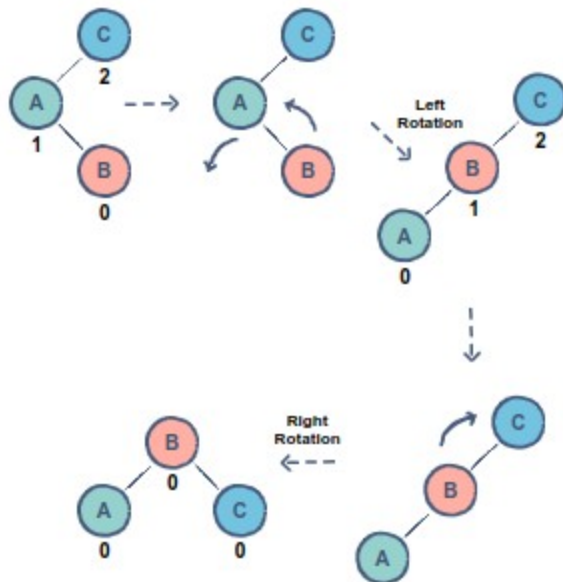
- $FB < -1$  e valor inserido a direita



# Árvores Binárias Balanceadas (AVL)

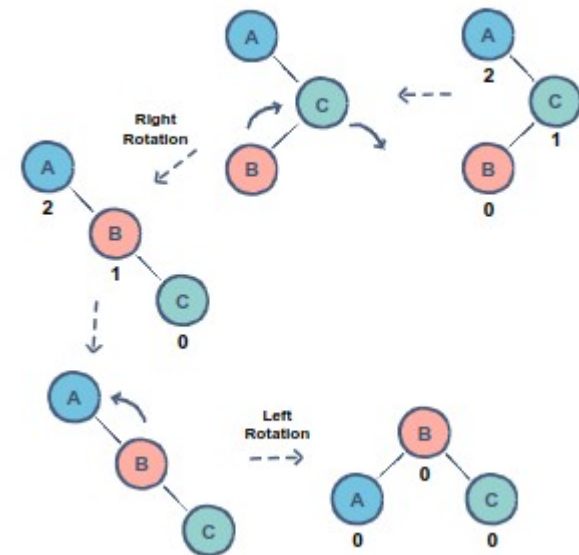
- Rotação Dupla Esq-Dir

- $FB > 1$  e valor inserido a direita



- Rotação Dupla Dir-Esq

- $FB < -1$  e valor inserido a esquerda

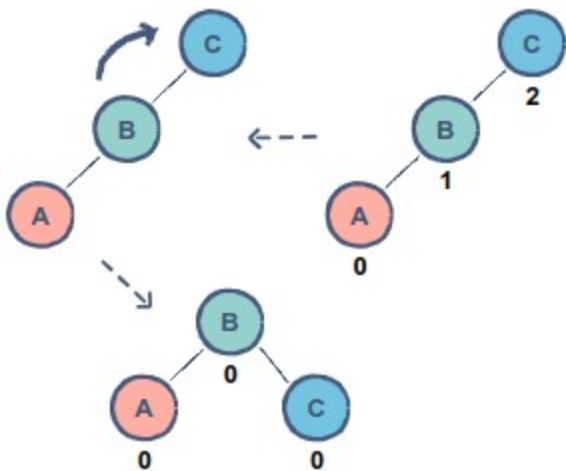


# Árvores Binárias Balanceadas (AVL)

- **Exercício prático**
  - **Inserir as sequências abaixo, utilizando arvores AVL e não-AVL**
    - 10, 3, 2, 5, 7 e 6
    - A,B,C ..... J
  - **Apresente os caminhos em in-ordem em ambas as arvores**
  - **Apresente o número de buscas necessárias para encontrar o elemento 7 e H em ambas as árvores**
- **Simulador AVL:**
  - <https://cmps-people.ok.ubc.ca/ylucet/DS/AVLtree.html>
  - <https://visualgo.net/en>

# Árvores Binárias Balanceadas (AVL)

- Pseudocódigos
- Rotação a Direita
  - $FB > 1$  e valor inserido a esquerda



```
if (balance > 1 && key < node->left->key)
    node = rightRotate(node);
```

```
Node *rightRotate(Node *y)
{
    Node *x = y->left;
    Node *T2 = x->right;

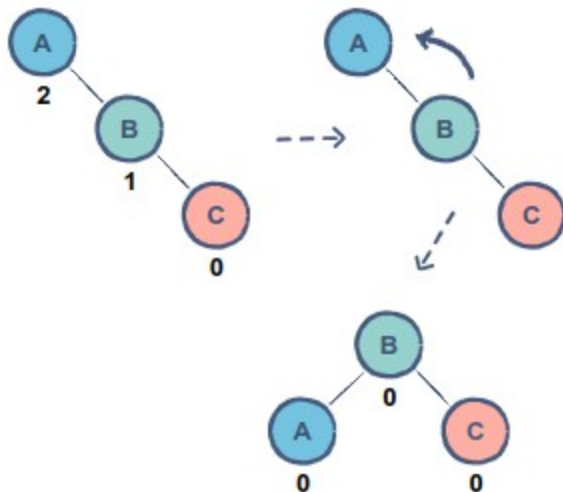
    // Perform rotation
    x->right = y;
    y->left = T2;

    // Update heights
    y->height = max(height(y->left),
                    height(y->right)) + 1;
    x->height = max(height(x->left),
                    height(x->right)) + 1;

    // Return new root
    return x;
}
```

# Árvores Binárias Balanceadas (AVL)

- Rotação a Esquerda
  - $FB < -1$  e valor inserido a direita



```
if (balance < -1 && key > node->right->key)
    node = leftRotate(node);
```

```
Node *leftRotate(Node *x)
{
    Node *y = x->right;
    Node *T2 = y->left;

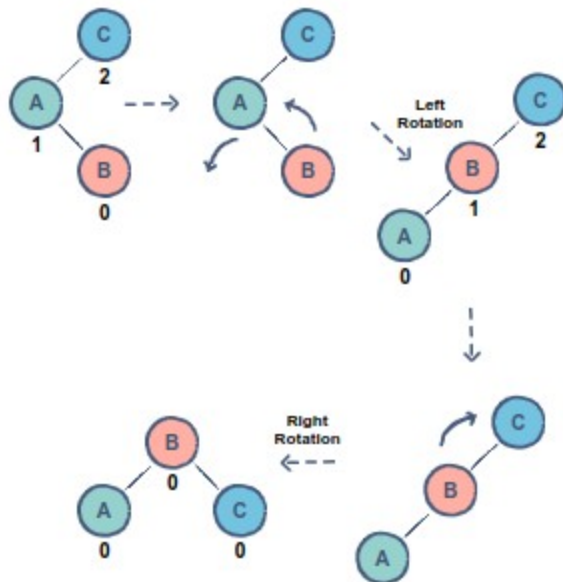
    // Perform rotation
    y->left = x;
    x->right = T2;

    // Update heights
    x->height = max(height(x->left),
                    height(x->right)) + 1;
    y->height = max(height(y->left),
                    height(y->right)) + 1;

    // Return new root
    return y;
}
```

# Árvores Binárias Balanceadas (AVL)

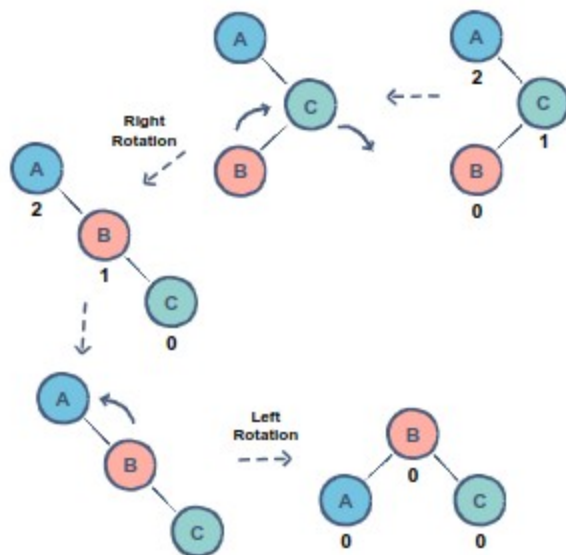
- Rotação Dupla Esq-Dir
- $FB > 1$  e valor inserido a direita



```
if (balance > 1 && key > node->left->key)
{
    node->left = leftRotate(node->left);
    node = rightRotate(node);
}
```

# Árvores Binárias Balanceadas (AVL)

- Rotação Dupla Dir-Esq
- $FB < -1$  e valor inserido a esquerda



```
if (balance < -1 && key < node->right->key)
{
    node->right = rightRotate(node->right);
    return leftRotate(node);
}
```

# Trabalhos

- **Implementação AVL Python com codificação dialogada (10 pt)**
  - **Inserção e Remoção com re-balanceamento**
  - **Vídeo-Apresentação (8 a 10 min)**
  - **12/09 - 23:59**
- **Artigo sobre AVL x Red-Black (10 pts)**
  - **Explicar a diferença entre as abordagens**
  - **Inserção, Remoção, Desempenho e Casos de Uso**
  - **Artigo em formato de artigo SBC**
  - **12/09**
- **Verificar os critérios de avaliação no ambiente virtual (AVA)**