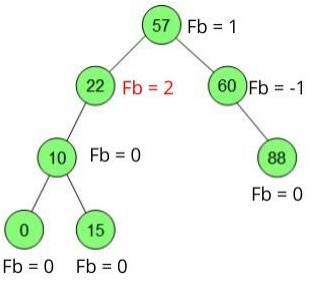
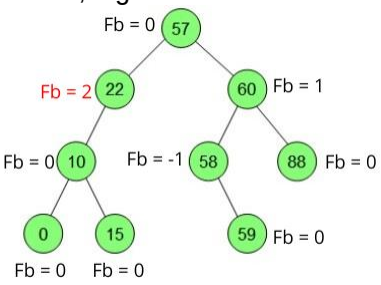
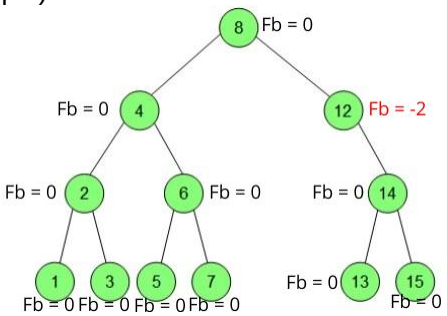
UNIVERSIDADE FEDERAL DE RORAIMA
CENTRO DE CIÊNCIA E TECNOLOGIA – CCT
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO – DCC
DCC405 – ESTRUTURA DE DADOS II
PROF. ACAUAN C. RIBEIRO
KELVIN ARAÚJO FERREIRA – 2019037653

Exercício – Aula 06 - Árvores AVL

1. Para cada árvore binária abaixo, diga se é uma árvore AVL (justifique).



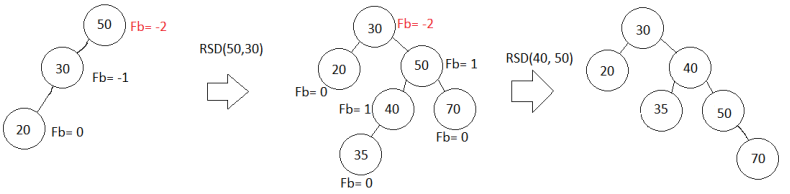


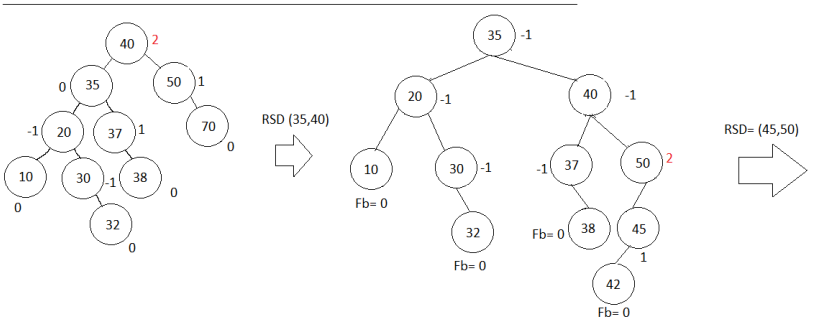


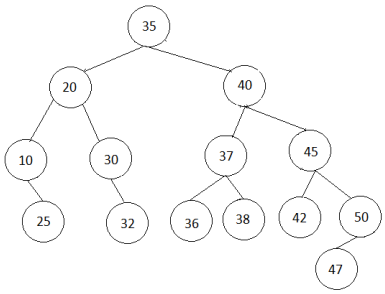
1- Arvore Binária Incompleta Degenerada
2- Arvore Binária Incompleta Degenerada
3- Arvore Binária Balanceada
4- Arvore Binária Incompleta Degenerada

2. Monte a árvore AVL (passo-a-passo) para as seguintes inserções de chaves, indicando a cada passo qual elemento foi inserido ou qual rotação foi realizada:

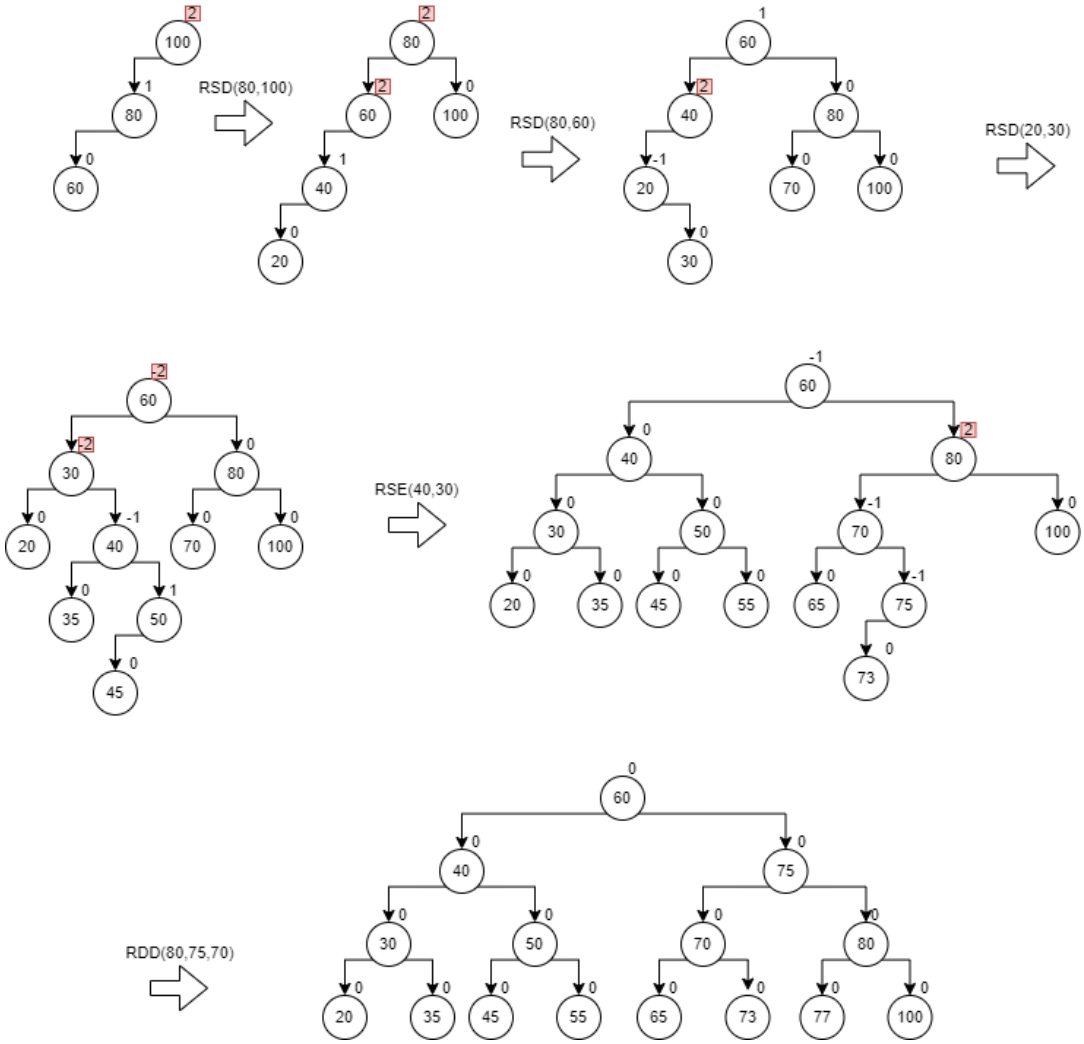
a) 50, 30, 20, 70, 40, 35, 37, 38, 10, 32, 45, 42, 25, 47, 36.



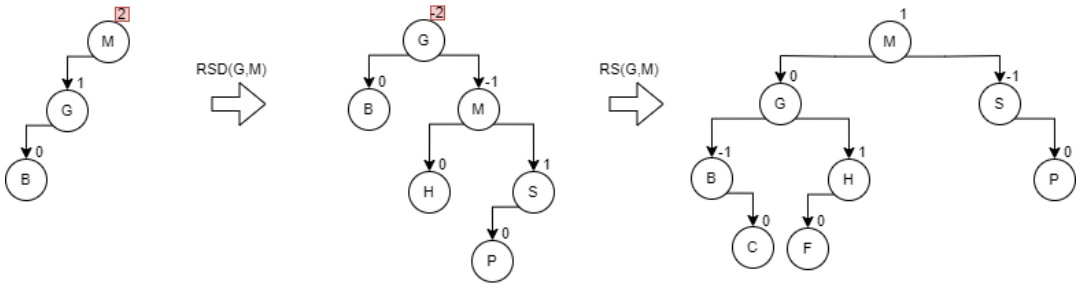




b) 100, 80, 60, 40, 20, 70, 30, 50, 35, 45, 55, 75, 65, 73, 77

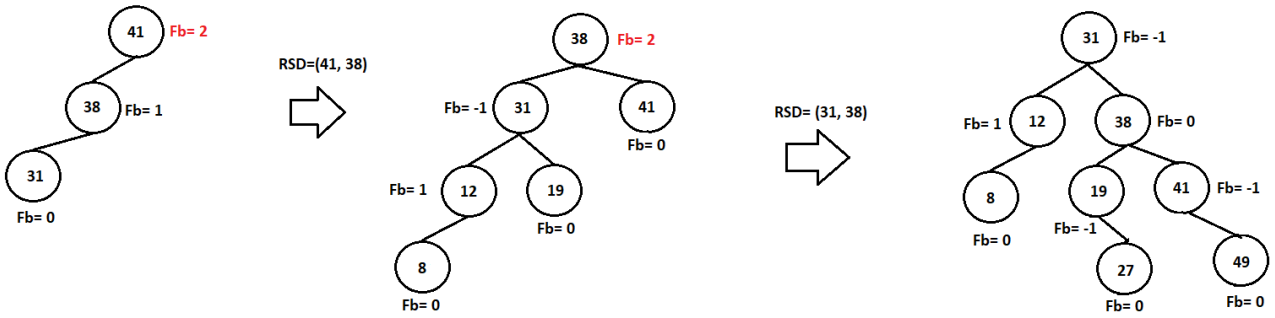


3. Dadas as seguintes chaves M, G, B, H, S, P, F, C como entrada (nesta ordem), desenhe a respectiva árvore AVL (balanceando-a quando for necessário).



4. Monte a árvore AVL (passo-a-passo) para as seguintes inserções de chaves:

41, 38, 31, 12, 19, 8, 27, 49 (nesta ordem), indicando a cada passo qual elemento foi inserido, o grau de balanceamento de cada nó e qual rotação foi realizada.



5. Implementea arvore AVL. Para um bom entendimento tente utilizar o arquivo bst.py (como fizemos em sala) e implemente as funções:

```
# Funções Auxiliares
def getHeight(root):

# FB = H(T.left) - H(T.right)
def getBalance(root):

# ROTAÇÕES
def leftRotate(z):

def rightRotate(z):


# Função de inserção
def insertNode(root, value):
    # Faz a inserção de maneira recursiva
    # Atualiza altura (pois um filho foi adicionado)
    # Verificar o Fator de Balanceamento
    # Chama as rotações para os casos específicos


# Função de remoção
def deleteNode(root, value):
```