Parte 1

1) Rotate left y right

```python
class AVLTree:
    root = None


class AVLNode:
    parent = None
    leftnode = None
    rightnode = None
    key = None
    value = None
    bf = None
```

```python
def rotateLeft(Tree, avlnode):

    raizVieja = avlnode
    Tree.root = avlnode.rightnode
    raizNueva = Tree.root

    raizNueva.parent = avlnode.parent
    raizNueva.parent.rightnode = raizNueva
    raizVieja.parent = raizNueva
    # no desconecte la raiz vieja de la nueva raiz
    raizVieja.rightnode = None

    aux = raizNueva.leftnode
    raizNueva.leftnode = raizVieja

    if aux is not None:
        raizVieja.rightnode = aux

    return raizNueva
```

```
34  def rotateRight(Tree, avlnode):
35
36      raizVieja = avlnode
37      Tree.root = avlnode.leftnode
38      raizNueva = Tree.root
39
40      raizNueva.parent = avlnode.parent
41      raizNueva.parent.leftnode = raizNueva
42      raizVieja.parent = raizNueva
43      # no desconecte la raiz vieja de la nueva raiz
44      raizVieja.leftnode = None
45
46      aux = raizNueva.rightnode
47      raizNueva.rightnode = raizVieja
48
49      if aux is not None:
50          raizVieja.leftnode = aux
51
52      return raizNueva
53
```

2) Calculate Balance

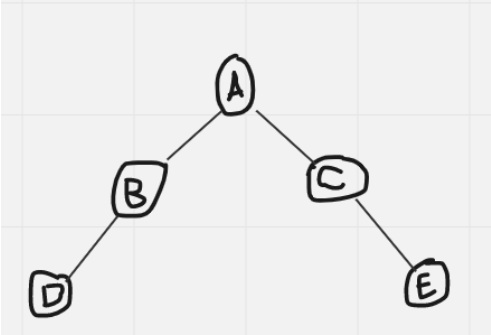
```
55  def calculateBalance(ALVTree):
56
57          if ALVTree is None:
58              return
59          node = AVLNode()
60          node = ALVTree.root
61          #Queremos actualizar el node.bf
62          height_left = balanceRecursive(node.leftnode)
63          height_right = balanceRecursive(node.rightnode)
64          bf = height_left - height_right
65          node.bf = bf
66
67          return ALVTree
68
```

```python
def balanceRecursive(node):

    if node is None:
        return 0

    height_left = balanceRecursive(node.leftnode)
    height_right = balanceRecursive(node.rightnode)

    bf = height_left - height_right
    node.bf = bf

    altura = max(height_left , height_right)

    return 1 + altura
```

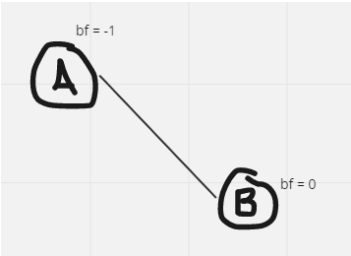
Parte 2

Ej 6:

a) Falso.

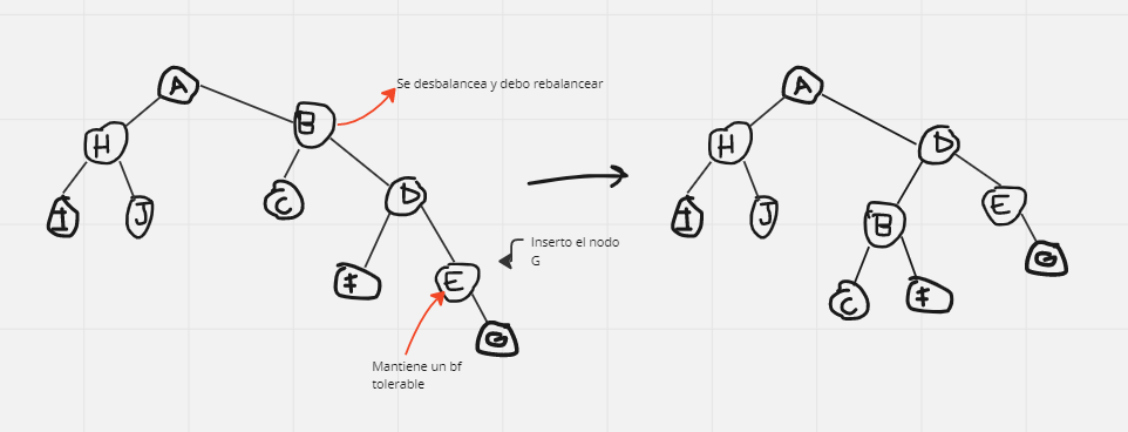


b) Verdadero. Por contraejemplo, vamos a suponer un Árbol AVL que no esté completo cuyos nodos tienen un bf = 0. Como no es completo existe al menos un nodo con 1 solo hijo



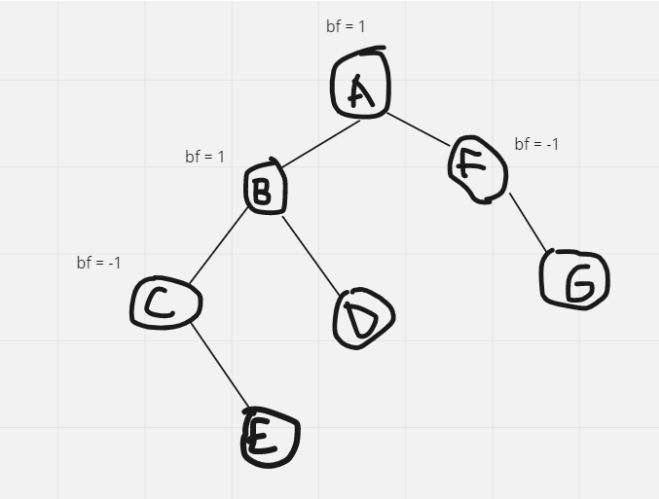
Por lo tanto, para que todos sus nodos tengan bf = 0, el árbol debe estar completo.

c) Falso. Planteo contraejemplo



Al insertar un nuevo nodo G. el padre de este no se desbalancea, pero debo seguir verificando hacia arriba ya que B si se ha desbalanceado

d) Falso. Planteo contraejemplo



Si no contamos las hojas, presento un contraejemplo que cumple ser AVL del cual ninguno de sus nodos posee bf = 0

Ej 7:

1º Calculo y comparo las alturas de los arboles A y B

2º inserto “x” en el árbol con mayor altura (B) a la altura del árbol menor (A)

3º El subárbol izquierdo de “x” va a ser el árbol con menor altura (A)

4º inserto el subárbol izquierdo de B en el lado derecho de “x”

5º El subárbol derecho de B queda igual

6º Se rebalancea desde “x” hacia la raíz verificando que no hayan desbalances

Costo de las operaciones:

Op1. O(log n) Altura Árbol A

Op2. O(log m) Altura Árbol B

Op3. O(log n) Inserción de “x“ en Altura de A

Op4. O(1) Insertar subárbol izq. B en lado derecho de “x”

Op5. O(1) Insertar árbol A en lado izq. de “x”

Op6. O(log n) Rebalancear desde “x” hacia la raíz

3 log(n) + log(m)

O(log(n) + log(m))