Trabajo practico de AVL

(tp1)

Nombre: Palau Enzo

Ejercicio 1

```python
import binarytree
import linkedlist
from myqueue import *
class AVLTree:
    root = None

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

"""Ejercicio 1"""

def rotateLeft(Tree,avlnode):
  #realiza una rotacion hacia la izquierda en un AVl
  Nroot=avlnode.rightnode
  avlnode.rightnode=Nroot.leftnode
  if Nroot.leftnode is not None:
     Nroot.leftnode.parent=avlnode
  Nroot.parent=avlnode.parent
  if avlnode.parent is None:
     Tree.root=Nroot
  else:
     if avlnode.parent.leftnode is avlnode:
        avlnode.parent.leftnode=Nroot
     else:
        avlnode.parent.rightnode=Nroot
  Nroot.leftnode=avlnode
  avlnode.parent=Nroot
  return Nroot
```

```python
def rotateRight(Tree,avlnode):
  #realiza una rotacion hacia la derecha en un AVl
  Nroot=avlnode.leftnode
  avlnode.leftnode=Nroot.rightnode
  if Nroot.rightnode is not None:
     Nroot.rightnode.parent=avlnode
  Nroot.parent=avlnode.parent
  if avlnode.parent is None:
     Tree.root=Nroot
  else:
     if avlnode.parent.rightnode is avlnode:
        avlnode.parent.rightnode=Nroot
     else:
        avlnode.parent.leftnode=Nroot
  Nroot.rightnode=avlnode
  avlnode.parent=Nroot
  return Nroot
```

Ejercicio 2

```
def calculateBalance(AVLTree):
    #Calcula el balance para cada nodo.
    #calculo la altura.
    Current=AVLTree.root
    CalculateHeight(AVLTree,Current)
    #calculo el balance factor
    Current=AVLTree.root
    CalculateBalanceR(AVLTree,Current)
    return AVLTree
```

```
def CalculateBalanceR(AVLTree,Current):
    if Current is not None:
        if Current.leftnode is not None and Current.rightnode is not None:
            #calculo el balance que es hoja con dos hijos
            Current.bf=((Current.leftnode.height)-(Current.rightnode.height))
        elif Current.leftnode is not None and Current.rightnode is None:
            #Nodo con un hijo caso #1
            Current.bf=((Current.height)-0)
        elif Current.rightnode is not None and Current.leftnode is None:
            #Nodo con un hijo caso #2
            Current.bf=(0-(Current.height))
        else:
            #balance de una  hoja es 0
            Current.bf=0
    if Current is not None:
        CalculateBalanceR(AVLTree,Current.leftnode)
        CalculateBalanceR(AVLTree,Current.rightnode)
```

```
    def CalculateHeight(AVLTree,Current):
        #calcula la altura de cada nodo.
        if Current is not None:
            Current.height=maxalt(Current)-1
        if Current is not None:
            CalculateHeight(AVLTree,Current.leftnode)
            CalculateHeight(AVLTree,Current.rightnode)

    def maxalt(current):
        #ve la altura maxima
        if current == None:
            return 0
        else:
            lalt = maxalt(current.leftnode)
            ralt = maxalt(current.rightnode)
            if (lalt > ralt):
                return lalt+1
            else:
                return ralt+1
```

Ejercicio 3

```python
"""Ejercicio 3"""

def reBalance(AVLTree):
    #Balancea un arbol para que este resulte un AVL
    #calculo los Bf de el arbol
    calculateBalance(AVLTree)
    Current=AVLTree.root
    reBalanceRecursive(AVLTree,Current)
    return AVLTree
```

```python
def reBalanceRecursive(AVLTree,Current):
    #recorre los nodos en busca de hojas y busca los padres para ir balanceando
    if Current is not None:
        if Current.rightnode is None and Current.leftnode is None:
            #hoja
            CurrentAux=Current#aux para padres
            while CurrentAux is not None:
                if CurrentAux.bf > 1 or CurrentAux.bf < -1:
                    #ahora necesitamos que este no sea una hoja
                    if CurrentAux.rightnode is not None or CurrentAux.leftnode is not None:
                        BalanceNode(AVLTree,CurrentAux)
                CurrentAux=CurrentAux.parent
    if Current is not None:
        reBalanceRecursive(AVLTree,Current.leftnode)
        reBalanceRecursive(AVLTree,Current.rightnode)
```

```python
def BalanceNode(AVLTree,Current):
    #realiza el balance de ese subarbol que tiene a current como raiz
    if Current.bf < 0:
        if Current.rightnode.bf > 0:
            rotateRight(AVLTree,Current.rightnode)
            rotateLeft(AVLTree,Current)
        else:
            rotateLeft(AVLTree,Current)
    elif Current.bf > 0:
        if Current.leftnode.bf < 0:
            rotateLeft(AVLTree,Current.leftnode)
            rotateRight(AVLTree,Current)
        else:
            rotateRight(AVLTree,Current)
    #actualizo los Bf
    calculateBalance(AVLTree)
```

Ejercicio 4 y Ejercicio 5

```
"""Ejercicio 4 y 5"""

def insert(AVLTree, element, key):
    # Inserta un elemento con una clave determinada del árbol Avl y luego rebalancea si corresponde
    newNode = AVLNode()
    newNode.key = key
    newNode.value = element
    currentNode = AVLTree.root
    insertR(AVLTree, newNode, currentNode)
    #rebalanceo
    reBalance(AVLTree)
    """no calculo de nuevo el factor de balanceo (actualizar) ya que (en mi caso)
    la funcion reBalance() calcula el Bf por si misma """
```

```
165   def insertR(B, newNode, currentNode):
166       #funcion que inserta en un arbol binario recursivamnete tambien sirve para un AVl
167       aux = False
168       if B.root == None:
169           B.root = newNode
170           aux = True
171           return newNode.key
172       if aux == False:
173           if currentNode.key < newNode.key:
174               if currentNode.rightnode == None:
175                   newNode.parent = currentNode
176                   currentNode.rightnode = newNode
177                   return newNode.key
178               else:
179                   insertR(B, newNode, currentNode.rightnode)
180           else:
181               if currentNode.key == newNode.key:
182                   return None
183               if currentNode.leftnode == None:
184                   newNode.parent = currentNode
185                   currentNode.leftnode = newNode
186                   return newNode.key
187               else:
188                   insertR(B, newNode, currentNode.leftnode)
```

```
190    def delete(AVLTree, element):
191        # Elimina un elemento del TAD árbol AVL
192        # Recorre un árbol binario en post-orden
193        current = AVLTree.root
194        Busca(current, AVLTree, element)
195
196
197    def Busca(current, B, element):
198        if current != None:
199            Busca(current.leftnode, B, element)
200            Busca(current.rightnode, B, element)
201            if current.value == element:
202                return deleteKey(B, current.key)
203
```

```
204    def deleteKey(B, key):
205        # Elimina una clave del TAD árbol AVL
206        # situaciones con la raiz
207        aux = 0
208        if B.root.leftnode == None and B.root.rightnode == None and B.root.key == key:
209            B.root = None
210            return None
211        elif B.root.leftnode == None and B.root.rightnode == None and B.root.key != key:
212            return None
213        else:
214            # otras
215            current = B.root
216            aux = deleteKeyR(current, key, B)
217            reBalance(AVLTree)
218            """no calculo de nuevo el factor de balanceo (actualizar) ya que (en mi caso)
219        la funcion reBalance() calcula el Bf por si misma """
220            return aux
221
```

```
def deleteKeyR(current, key, B):
    # busca el nodo
    aux = 0
    if current.key < key:
        if current.rightnode != None:
            aux = deleteKeyR(current.rightnode, key, B)
    elif current.key > key:
        if current.leftnode != None:
            aux = deleteKeyR(current.leftnode, key, B)
    elif current.key == key:
        # bloque de borrado
```

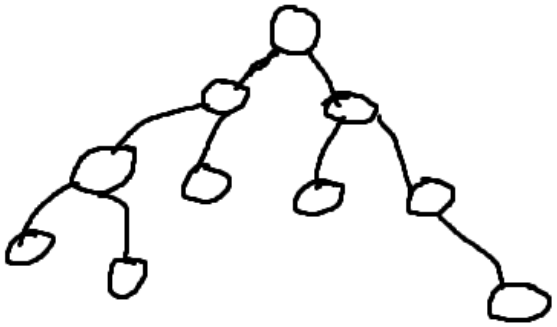
```
        # bloque de borrado
        # si encontro el nodo con la key
        if current.leftnode == None and current.rightnode == None:
            # encontro una hoja
            aux = current.key
            if current.key == current.parent.leftnode.key:
                current.parent.leftnode = None
            else:
                current.parent.rightnode = None
        else:  # nodo rama
            aux = current.key
            nodoActu = AVLNode()
            # dos hijos
            if current.leftnode != None and current.rightnode != None:
                buscarNodo(current.leftnode, B, nodoActu)
                current.key = nodoActu.key
                current.value = nodoActu.value
            else:
                # solo hijo der
                if current.leftnode == None:
                    if current.parent.leftnode.key == current.key:
                        current.parent.leftnode = current.rightnode
                    elif current.parent.rightnode.key == current.key:
                        current.parent.rightnode = current.rightnode
                else:
                    # solo hijo izq
                    if current.parent.leftnode.key == current.key:
                        current.parent.leftnode = current.leftnode
                    elif current.parent.rightnode.key == current.key:
                        current.parent.rightnode = current.leftnode
    else:
        # no lo encuentra
        return None
    return aux
```

```
def buscarNodo(current, B, nodoActu):
    # CRITERIO: EL MAYOR DE LOS MENORES
    # dos hijos
    if current.rightnode != None:
        buscarNodo(current.rightnode, B, nodoActu)
    else:
        # Borra la hoja correspondiente
        if current.leftnode == None and current.rightnode == None:
            nodoActu.key = current.key
            nodoActu.value = current.value
            if current.parent.leftnode != None:
                if current.key == current.parent.leftnode.key:
                    current.parent.leftnode = None
            else:
                if current.key == current.parent.rightnode.key:
                    current.parent.rightnode = None
        return nodoActu
```

Ejercicio 6

___ En un AVL el penúltimo nivel tiene que estar completo.

Falso
Contra ejemplo**:**



Es un AVL que no cumple la consigna.

___Un AVL donde todos los nodos tengan factor de balance 0 es completo

Verdadero

Suponiendo que existe un Avl con todos los bf==0 pero no es completo, entonces un nodo del árbol tiene un hijo y su bf entonces no seria 0

___ En la inserción en un AVL, si al actualizarle el factor de balance al padre del nodo insertado éste no se desbalanceó, entonces no hay que seguir verificando hacia arriba porque no hay cambios en los factores de balance.

Falso

Algún nodo podría haberse desbalanceado por que si o si hay que verificar hacia arriba

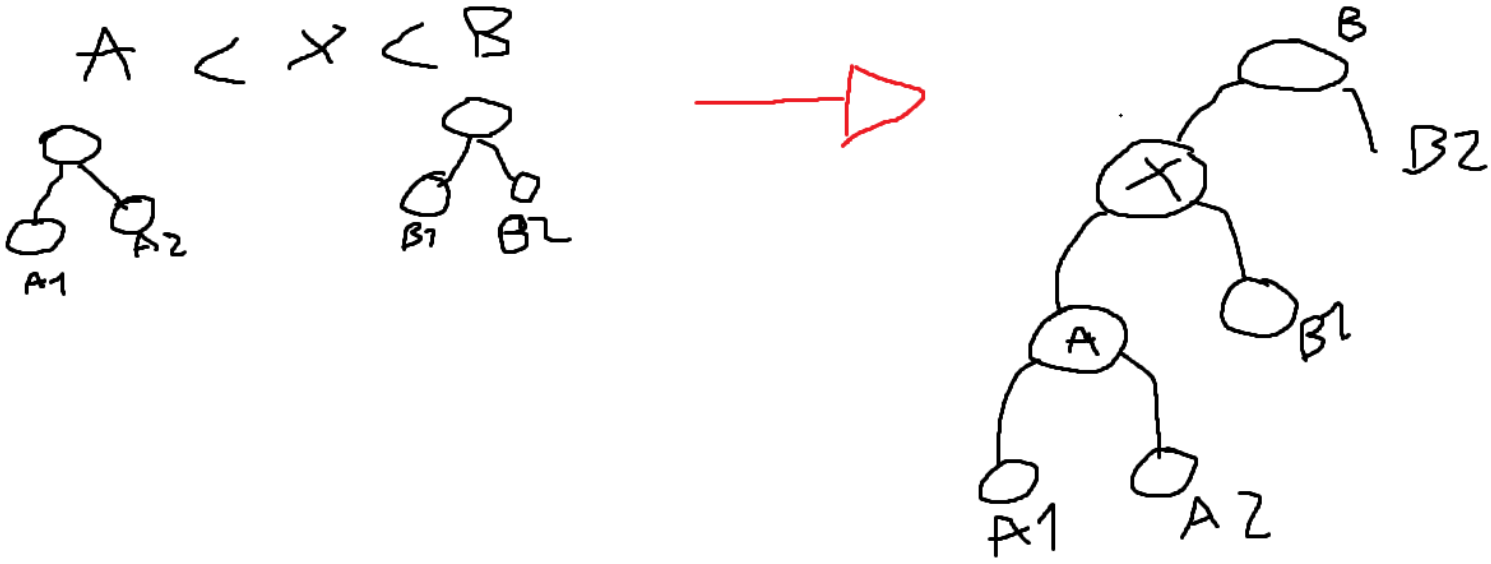
___ En todo *AVL* existe al menos un nodo con factor de balance 0.

Verdadero

Si contamos las hojas esto resulta cierto pero ni no las contamos vemos que es falso por ejemplo:

Ejercicio 7

Deberíamos insertarlos de esta manera



de este modo la altura varia en solo una unidad

Ejercicio 8

Es H/2 ya que al ser un árbol AVL, los sub-árboles siempre tendra a lo sumo un nivel menos que el siguiente sub-árbol

Link de Replit:

https://replit.com/@EnzoPalau/AVl#main.py