AVL树

class TreeNode(object):  
 def \_\_init\_\_(self):  
 self.data=0  
 self.left=None  
 self.right=None  
 self.height=0  
class BTree(object):  
 def \_\_init\_\_(self):  
 self.root=None  
 def \_\_Max(self,h1,h2):  
 if h1>h2:  
 return h1  
 elif h1<=h2:  
 return h2  
 def \_\_LL(self,r):  
 node=r.left  
 r.left=node.right  
 node.right=r  
 r.height=self.\_\_Max(self.getHeight(r.right),self.getHeight(r.left))+1  
 node.height=self.\_\_Max(self.getHeight(node.right),self.getHeight(node.left))+1  
 return node  
 def \_\_RR(self,r):  
 node = r.right  
 r.right = node.left  
 node.left = r  
 r.height = self.\_\_Max(self.getHeight(r.right), self.getHeight(r.left)) + 1  
 node.height = self.\_\_Max(self.getHeight(node.right), self.getHeight(node.left)) + 1  
 return node  
 def \_\_LR(self,r):  
 r.left=self.\_\_RR(r.left)  
 return self.\_\_LL(r)  
 def \_\_RL(self,r):  
 r.right=self.\_\_LL(r.right)  
 return self.\_\_RR(r)  
 def \_\_insert(self,data,r):  
 if r==None:  
 node=TreeNode()  
 node.data=data  
 return node  
 elif data==r.data:  
 return r  
 elif data<r.data:  
 r.left=self.\_\_insert(data,r.left)  
 if self.getHeight(r.left)-self.getHeight(r.right)>=2:  
 if data<r.left.data:  
 r=self.\_\_LL(r)  
 else:  
 r=self.\_\_LR(r)  
 else:  
 r.right=self.\_\_insert(data,r.right)  
 if self.getHeight(r.right)-self.getHeight(r.left)>=2:  
 if data>r.right.data:  
 r=self.\_\_RR(r)  
 else:  
 r=self.\_\_RL(r)  
 r.height=self.\_\_Max(self.getHeight(r.left),self.getHeight(r.right))+1  
 return r  
   
 def \_\_delete(self,data,r):  
 if r==None:  
 print ("don't have %d"%data)  
 return r  
 elif r.data==data:  
 if r.left==None:  
 return r.right  
 elif r.right==None:  
 return r.left  
 else:  
 if self.getHeight(r.left)>self.getHeight(r.right):  
   
 node=r.left  
 while(node.right!=None):  
 node=node.right  
 r=self.\_\_delete(node.data,r)  
 r.data=node.data  
 return r  
 else:  
 node=r.right  
 while node.left!=None:  
 node=node.left  
 r=self.\_\_delete(node.data,r)  
 r.data=node.data  
 return r  
 elif data<r.data:  
 r.left=self.\_\_delete(data,r.left)  
 if self.getHeight(r.right)-self.getHeight(r.left)>=2:  
 if self.getHeight(r.right.left)>self.getHeight(r.right.right):  
 r=self.\_\_RL(r)  
 else:  
 r=self.\_\_RR(r)  
 elif data>r.data:  
 r.right=self.\_\_delete(data,r.right)  
 if self.getHeight(r.left)-self.getHeight(r.right)>=2:  
 if self.getHeight(r.left.right)>self.getHeight(r.left.left):  
 r=self.\_\_LR(r)  
 else:  
 r=self.\_\_LL(r)  
 r.height=self.\_\_Max(self.getHeight(r.left),self.getHeight(r.right))+1  
 return r  
   
 def \_\_show(self,root):  
 if root!=None:  
 print (root.data)  
 self.\_\_show(root.left)  
 self.\_\_show(root.right)  
 else:  
 return 0  
 def Insert(self,data):  
 self.root=self.\_\_insert(data,self.root)  
 return self.root  
 def Delete(self,data):  
 self.root=self.\_\_delete(data,self.root)  
   
 def getHeight(self,node):  
 if node==None:  
 return -1  
 #print node  
 return node.height  
 def Show(self):  
 self.\_\_show(self.root)  
if \_\_name\_\_=='\_\_main\_\_':  
 bi=BTree()  
 array=[5,1,2,3,4]  
 for i in array:  
 bi.Insert(i)  
 bi.Delete(2)  
 bi.Insert(2)  
 bi.Show()

