平衡二叉树(AVL树)

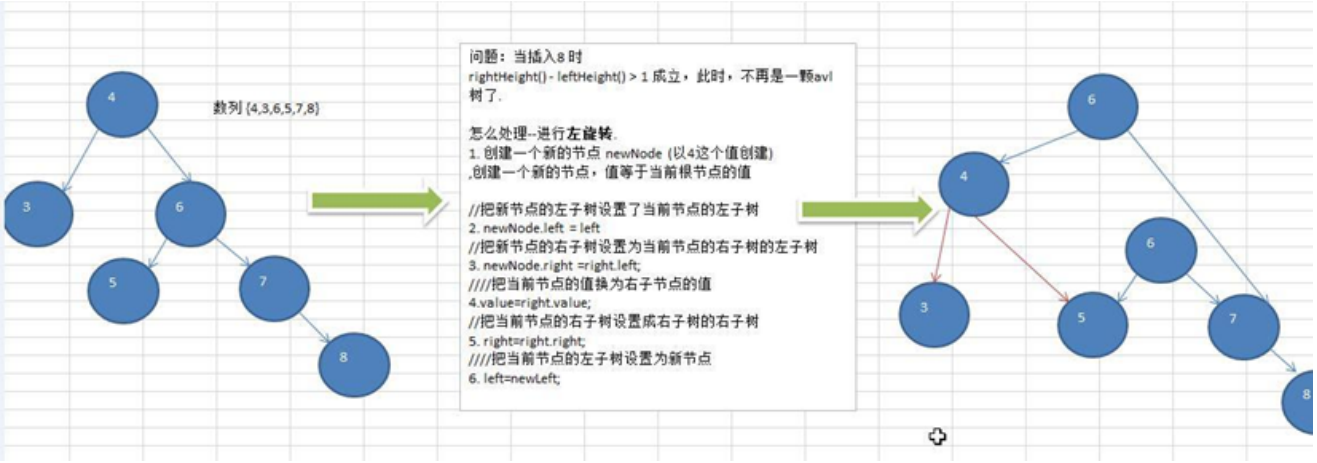
一、特点：

它是一 棵空树或它的左右两个子树的高度差的绝对值不超过 1，并且左右两个子树都是一棵平衡二叉树。

平衡二叉树的常用实现方法有红黑树、AVL、替罪羊树、Treap、伸展树等。

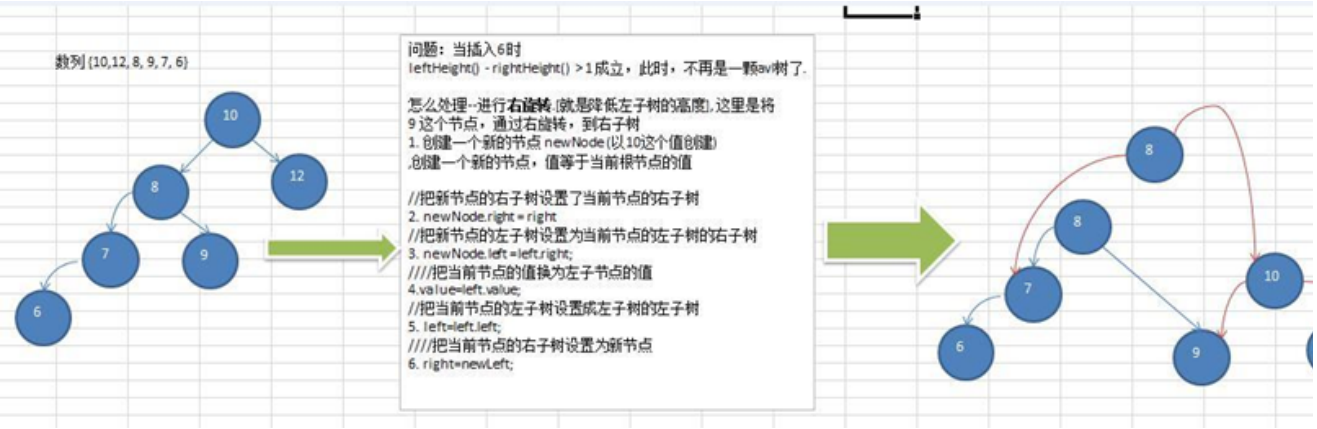
二、单旋转

1、左旋转




```
private void leftRotate() {
//创建新的结点，以当前根结点的值
Node newNode = new Node(value);
//把新的结点的左子树设置成当前结点的左子树
newNode.left = left;
//把新的结点的右子树设置成带你过去结点的右子树的左子树
newNode.right = right.left;
//把当前结点的值替换成右子结点的值
value = right.value;
//把当前结点的右子树设置成当前结点右子树的右子树
right = right.right;
}
```

2、右旋转




```
private void rightRotate() {
Node newNode = new Node(value);
newNode.right = right;
newNode.left = left.right;
value = left.value;
```

```
left = left.left;
}
```

三、存在问题

一次旋转不能平衡：


```
//当添加完一个结点后，如果: (右子树的高度-左子树的高度) > 1 , 左旋转
if(rightHeight() - leftHeight() > 1) {
    //如果它的右子树的左子树的高度大于它的右子树的右子树的高度
    if(right != null && right.leftHeight() > right.rightHeight()) {
        //先对右子结点进行右旋转
        right.rightRotate();
        //然后在对当前结点进行左旋转
        leftRotate(); //左旋转..
    } else {
        //直接进行左旋转即可
        leftRotate();
    }
    return ; //必须要!!!
}

//当添加完一个结点后，如果 (左子树的高度 - 右子树的高度) > 1, 右旋转
if(leftHeight() - rightHeight() > 1) {
    //如果它的左子树的右子树高度大于它的左子树的高度
    if(left != null && left.rightHeight() > left.leftHeight()) {
        //先对当前结点的左结点(左子树)->左旋转
        left.leftRotate();
        //再对当前结点进行右旋转
        rightRotate();
    } else {
        //直接进行右旋转即可
        rightRotate();
    }
}
```