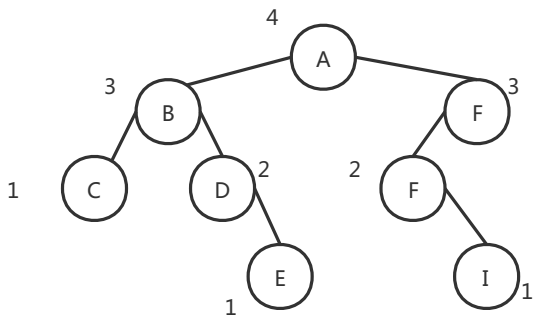
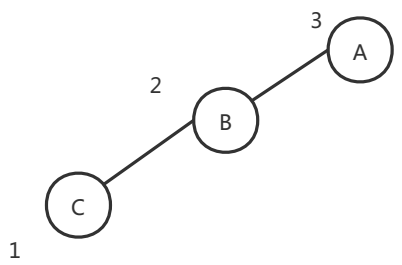
该文学习于https://blog.csdn.net/weixin_30363263/article/details/85702725#comments_13527831



平衡二叉树特点：
1. AVL是一颗二叉搜索树
2.AVL的左右子节点也是一个AVL树
3. AVL树有所有二叉搜索树的特点
4. 每个节点的左右子节点的高度差绝对值不超过1，即平衡因子范围为[-1,1]
以左侧树为例
1. B的子节点 C，D 一个是2，一个是1，相差为1；
2. A的子节点 B，F 两个都是3，相差为0；

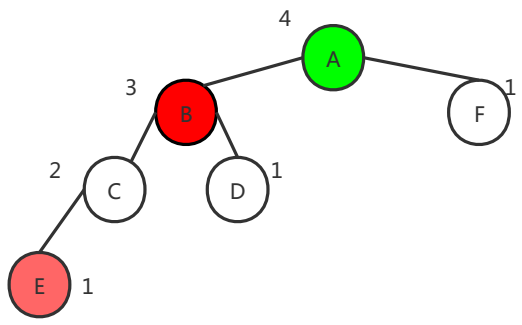


左侧树之所以不是AVL树，
可以根据根据AVL树特点进行判定

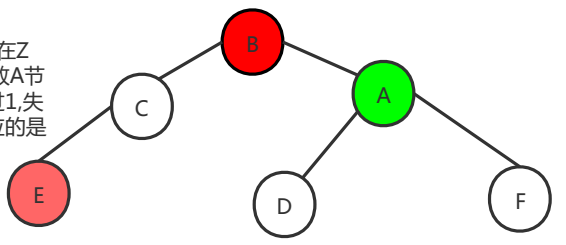
A节点的左侧子节点高度为2；右侧没有节点，所以高度为0；高度绝对差超过1，所以该树不是AVL树

平衡二叉树的过程中怎么判定以那个为轴??

AVL树一共有四种插入方式，LL插入，RR插入，LR插入；RL插入
设受插入节点影响而失去平衡的节点的父节点为Z

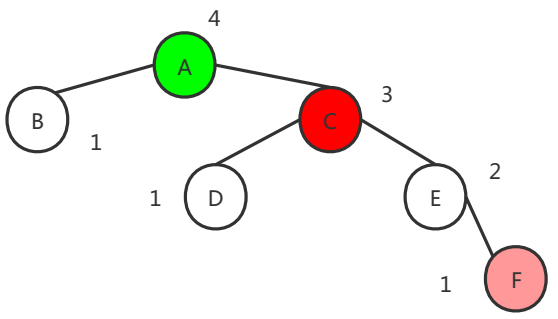


LL插入
插入的节点为E，插入的节点在Z节点的左子树的左子树上.导致A节点的左右子节点的高度差超过1,失去平衡的节点Z 在本例中对应的是B节点

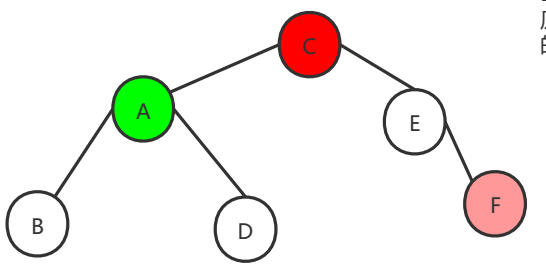


右单旋

1. 失去平衡节点B变成原父节点A的父节点
2. 原父节点A的右子节点F不变
3. 失去平衡节点B的右子节点D变成其原父节点A的左子节点

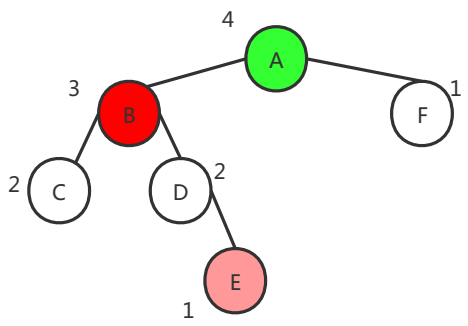


RR插入
插入的节点为F，插入的节点在Z节点的右子树的右子树上.导致A节点的左右子节点的高度差超过1,失去平衡的节点Z 在本例中对应的是C节点

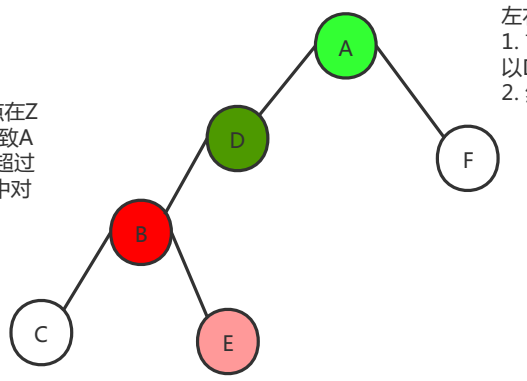


左单旋

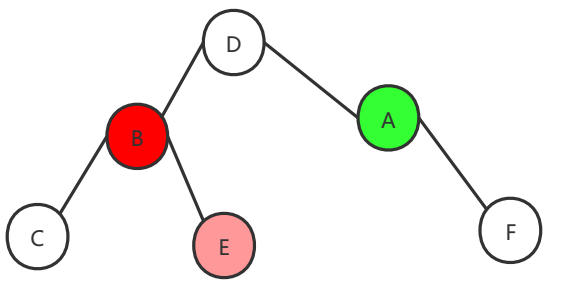
1. 失去平衡节点C变成原父节点A的父节点2. 原父节点A的左子节点F不变3. 失去平衡节点B的左子节点D变成其原父节点A的右子节点

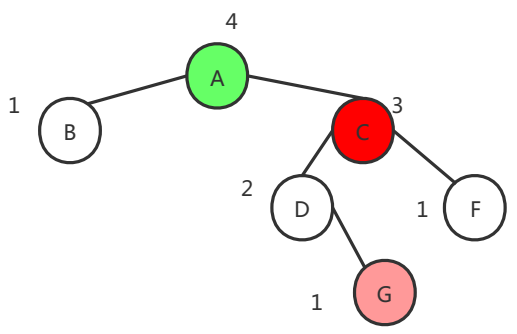


LR插入
插入的节点为E，插入的节点在Z节点的左子树的右子树上.导致A节点的左右子节点的高度差超过1,失去平衡的节点Z 在本例中对应的是B节点

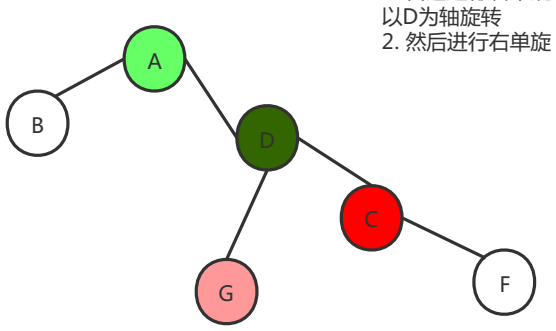


左右双旋
1. 首选进行左单旋以D为轴旋转
2. 然后进行右单旋

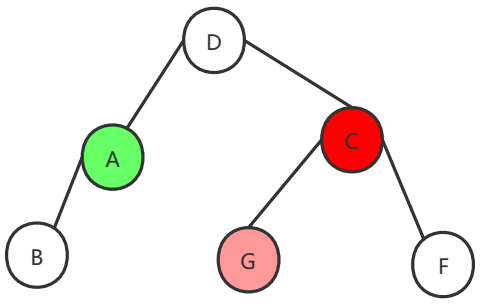




RL插入
插入的节点为F，插入的节点在Z节点的右子树的右子树上.导致A节点的左右子节点的高度差超过1,失去平衡的节点Z 在本例中对应的是C节点



右左双旋
1. 首选进行右单旋以D为轴旋转
2. 然后进行右单旋



插入插入时可能会遇到四种不同的插入方式，分别是：LL插入方式、RR插入方式、LR和RL插入方式。根据不同的插入方式对应做旋转操作即能使树达到平衡状态。注意的一点是确定最底层首先出现不平衡的节点去判断其属于哪种方式进行调整。

查找AVL树因为属于二叉搜索树，所以查找时与BST树完全一样

删除删除操作主要分两种情况，一种是删除后不会影响平衡，那么直接按照BST树规则删除。另外一种是删除后会影响树的平衡，那么则需要再做旋转处理。