AVL Tree

AVL树

描述：

AVL树是最早发明的一种自平衡二叉查找树，树中的任何节点的左右两个子树的高度最大差别为1，因此也称为高度平衡树。AVL树的查找、插入、删除操作的平均时间复杂度都是，AVL树高度为。

为了保持树的左右子树的平衡，避免一侧过长或过短，AVL树会对LL（左左）、RR（右右）、LR（左右）、RL（右左）四种情况进行调整：









上面四种情况包含了所有从不平衡转化为平衡的步骤，其中单向右旋平衡处理LL，单向左旋平衡处理RR，双向旋转（先左后右）平衡处理LR，双向旋转（先右后左）平衡处理RL。

这四种操作既能够平衡左右子树的高度，还能够保持树的有序性。即平衡后的树的左子树中所有节点仍然小于（或大于）树的根节点，而右子树中所有节点仍然大于（或小于）树的根节点。

AVL树的每个节点都有一个高度值，树的平衡因子为，即左右子树的深度之差。当一个节点的时该子树平衡；当时该子树不平衡。

将空节点的高度值视作，一个节点的高度值为。上面LL、RR、LR和RL四种操作，都会将其节点1的高度值减2，其余节点的高度值都不变。

对于下面这个AVL树，每个节点中上面的数字是节点下标号，下面的数字是该节点的高度值。将节点18插入下面的AVL树：



1. 从根节点开始，将节点18与节点10比较，有，因此把节点18插入节点10的右子树；
2. 将节点18与节点15比较，有，因此把节点18插入节点15的右子树；
3. 将节点18与节点19比较，有，因此把节点18插入节点19的左子树；
4. 将节点18与节点16比较，有，因此把节点18插入节点16的右子树；
5. 将节点18与节点17比较，有，因此把节点18插入节点17的右子树，节点17的右孩子节点为空，因此节点18成为节点17的右孩子节点；

然后从节点18开始，向上依次更新所有节点的高度值，若新的高度值不满足AVL树的平衡性，则进行旋转操作：

1. 节点18为叶子节点，因此高度值为0；

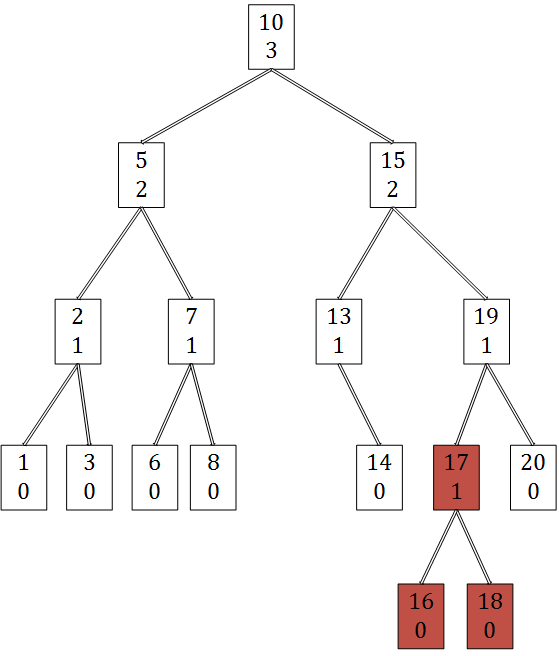


1. 节点17的平衡因子为，不需要旋转，高度值更新为；



1. 节点16的平衡因子为，高度值更新为，由于节点16的平衡因子超过1，需要进行RR操作，旋转后节点16的高度值减2；





1. 节点19的平衡因子为，高度值更新为；



1. 节点15的平衡因子为，高度值更新为；



1. 节点10的平衡因子为，高度值更新为；

