当确定搜索树的高度总是O(logn)时，能够保证每个搜索树操作所占用的时间为O(logn)。高度为O(logn)的树称为平衡树。平衡二叉树主要是为了解决二叉搜索树插入的元素不随机，导致平衡性能破坏，不能在O(nlogn)复杂度下完成搜索的情况出现的。

**定义**：

空二叉树是AVL树；如果T是一棵非空的二叉树，TL和TR分别是其左子树和右子树，那么当T满足以下条件时，T是一棵AVL树：

1）TL和TR是AVL树；

2）|hl-hr|<=1, hl和hr分别是左子树和右子树的高度；

一般用链表方式来描述AVL树，为每个节点增加一个平衡因子bf。节点x的平衡因子bf(x)定义为：x的左子树的高度 -  x的右子树的高度。因为n元素的AVL树的高度是O(logn), 所以搜索所需的时间为O(logn)。当进行插入操作时，我们需要更新通向根节点路径上那些节点的所有平衡信息。更新平衡信息之后，我们必须对树进行调整，以使它成为新的AVL树。

维基百科：对AVL平衡二叉树描述非常好；重点是理解什么时候左旋（LL）、右旋(RR)、先右旋后左旋(LR)、先左旋后右旋(RL);左旋和右旋的基准是中间根节点的旋转方向。

以左旋为例，说明一下右旋的四个步骤：

1. K1的父节点指向K2;
2. 用temp暂存K1的右子节点；
3. K2的左指针指向K1;
4. K2的右指针指向K1右子节点即temp

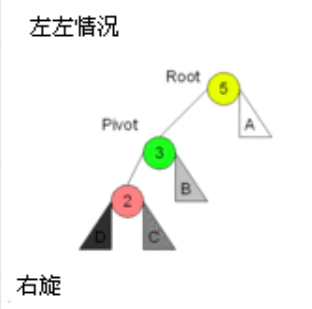


图1 5节点为K1，3节点为K2,2节点为K3

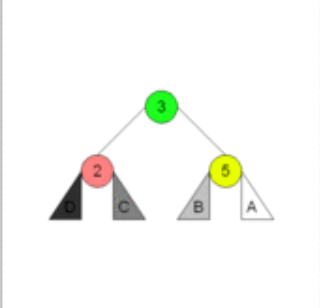


图2 右旋之后的结果

<https://zh.wikipedia.org/wiki/AVL%E6%A0%91>