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(nlogn)。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。

把必须重新平衡的节点叫做x。由于任意节点最多有两个儿子，因此出现高度不 平衡就需要x点的两棵子树的高度差2。这种不平衡可能出现在下面4中情况中：

1.对x的左儿子的左子树进行一次插入；

2.对x的左儿子的右子树进行一次插入；

3.对x的右儿子的左子树进行一次插入；

4.对x的右儿子的右子树进行一次插入；

对于1、4这两种情况通过对树的一次单旋转完成树的调整；对于2、3这两种情况可以通过稍微复杂些的双旋转来处理。

**单旋转和双旋转的示例图**

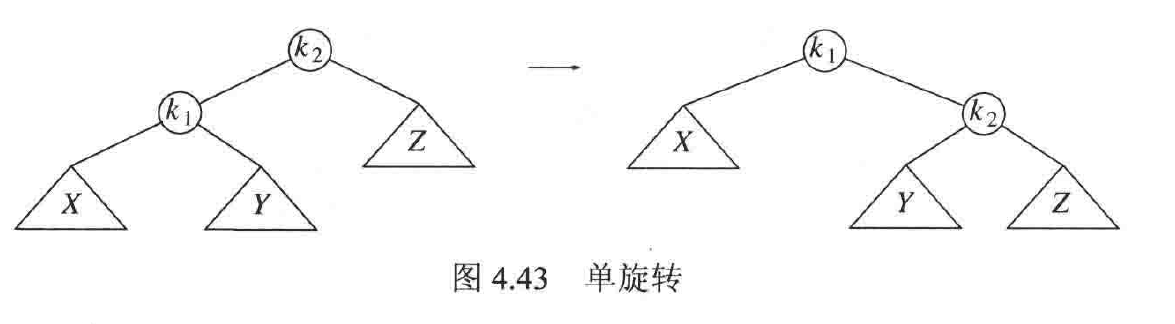


图1 单旋转的示例

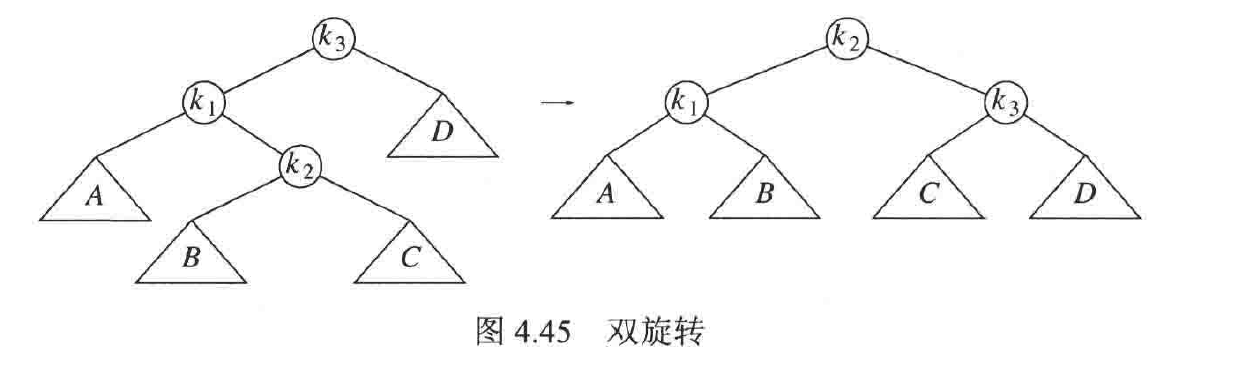


图2 双旋转的示例