一、基本概念

AVL树既是平衡二叉树。AVL树的定义首先要求该树是二叉查找树（满足排序规则），并在此基础上增加了每个节点的平衡因子的定义，一个节点的平衡因子是该节点的左子树树高减去右子树树高的值。

===========================================================================

1. 二叉查找树：又称二叉排序树/二叉搜索树，具有以下性质：

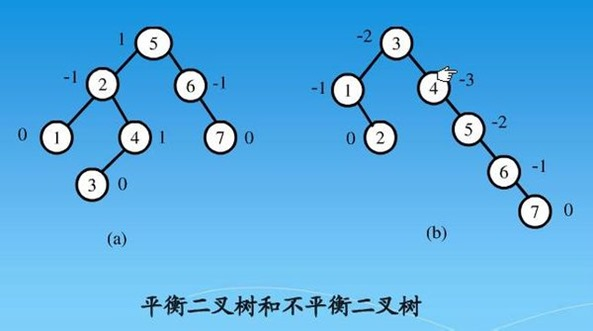
（1）若左子树不空，则左子树上所有结点的值均小于它的根结点的值；

（2）若右子树不空，则右子树上所有结点的值均大于或等于它的根结点的值；

（3）左、右子树也分别为二叉排序树；

（4）没有键值相等的节点。

2. 平衡因子BF：该节点的左子树的深度减去它的右子树深度。（平衡二叉树上所有节点的平衡因子可能是-1,0和1）以图示法讲解吧——



eaf81a4c510fd9f9d5da2b5a242dd42a2834a437

（1）平衡二叉树：此节点往下 左孩子 右孩子 结果

5的结点平衡因子就是 3 - 2 = 1；

2的结点平衡因子就是 1 - 2 = -1；

4的结点平衡因子就是 1 - 0 = 1；

6的结点平衡因子就是 0 - 1 = -1；

叶子结点都是为 0；

你在根据平衡二叉树的新得推测 不平衡二叉树的平衡因子；

（2）不平衡二叉树：此节点往下 左孩子 右孩子 结果

3 的结点平衡因子就是 2 - 4 = -2；

1 的结点平衡因子就是 0 - 1 = -1；

4 的结点平衡因子就是 0 - 3 = -3；

5 的结点平衡因子就是 0 - 2 = -2；

6 的结点平衡因子就是 0 - 1 = -1；

叶子结点都是为 0；

===========================================================================

二、为什么要用AVL树呢？

我们先来看看二叉搜索树吧（因为AVL树本质上是一棵二叉搜索树），假设有这么一种极端的情况：二叉搜索树的结点为1、2、3、4、5，也就是：

1

\

2

\

3

\

4

\

5

从这张图里就能很清晰的看出来二叉搜索树其实退化成了一个链表了，也就是说，它在查找上的优势已经全无了。这种情况下，查找一个节点的时间复杂度是O（n）。

那如果是一个AVL树呢？

2

/ \

1 4

/ \

3 5

可以看得出，AVL树的查找平均复杂度是O（log(n)）的，也就是说大量随机数据中AVL树表现的要好。（AVL树查找时间复杂度就是树高）

三、AVL树旋转

1. 造成AVL树不平衡的原因可能有以下四点：

(1) 对该结点的左儿子的左子树进行了一次插入。

(2) 对该结点的左儿子的右子树进行了一次插入。

(3) 对该结点的右儿子的左子树进行了一次插入。

(4) 对该结点的右儿子的右子树进行了一次插入。

2. 旋转

由于AVL树需要做到平衡，所以每次插入叶子节点，如果发现不平衡，都需要进行旋转以保持平衡。这里转载了一个图片，描述了AVL树旋转平衡的操作——

Center

