【数据结构】Day6

Class	Advanced Data Structures
Date	
Material	
Series Number	
Summary	

【Ch4】树

4.3 查找二叉树ADT—二叉查找树

二叉查找树的性质：对于树中的每个节点X，它的左子树中所有关键值小于X的关键字值，而它的右子树中所有关键字值大于X的关键字值。

4.4 AVL树

AVL树是其每个结点的左子树和右子树的高度最多差1的二叉查找树。

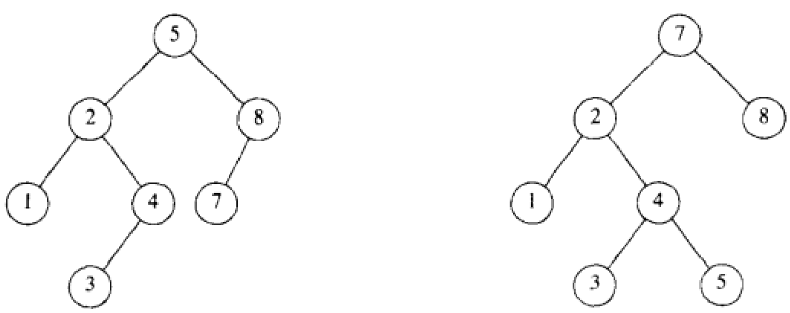

图 4-29　两棵二叉查找树，只有左边的树是 AVL 树

每一个结点保留高度信息，实际上的AVL树高度只比logN稍微多一点。

除去可能的插入外，所有的树操作都可以用时间O(logN)执行。

当进行插入时，我们需要更新通向根结点路径上那些结点的所有平衡信息。而插入操作隐含着困难的原因在于，插入一个结点可能破坏AVL树的特性

如果发生这种情况，那么就要把性质恢复以后才认为这一步插入完成。事实上，这总可以通过对树进行简单修正来做到，我们称其为旋转(rotation)

在插入以后，只有那些从插入点到根结点的路径上的结点的平衡可能被改变。因为只有这些结点的子树可能发生变化

当我们沿着这条路径上行到根并更新平衡信息时，我们可以找到一个结点，它的新平衡破坏了AVL条件。我们将址处如何在第一个这样的结点（即最深的结点）重新平衡这棵树，并证明，这一重新平衡保证整个树满足AVL特性

让我们把需要重新平衡的结点叫做a，由于任意结点最多有两个儿子，因此高度不平衡时，a点的两棵子树高度差为2。

可以看出，这种不平衡可能出现在下面4种情况中：

1. 对a的左儿子的左子树进行了一次插入
2. 对a的左儿子的右子树进行了一次插入
3. 对a的右儿子的左子树进行了一次插入
4. 对a的右儿子的右子树进行了一次插入

情形1和4是关于a点的镜像对称，而2和3是关于a点的镜像对称。

第一种情况是插入发生在外边的情况（即左-左或右-右的情况），该情况通过对树的一次单旋转(single rotation)而完成调整

第二种情况是插入发生在内部的情况（即左-右或右-左的情况），该情况通过稍微复杂些的双旋转（double rotation）来处理。

4.4.1 单旋转

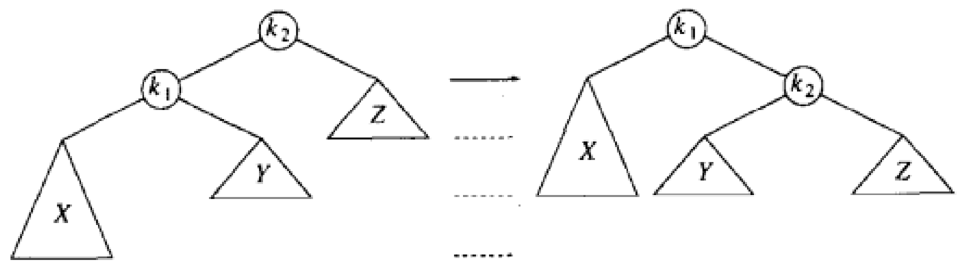

图 4-31 调整情形 1 的单旋转

上图显示单旋转（左旋转）如何调整情形1。旋转前的图在左边，而旋转后的图在右边

结点k_2不满足AVL平衡特性，因为它的左子树比右子树深2层。该图所描述的情况只是情形1的一种可能情况，在插入之前k_2满足AVL特性，但在插入之后这种特性被破坏了。

子树X已经长出一层，这使得它比子树Z深处2层

为使树恢复平衡，我们把X上移一层，并把Z下移一层。我们让k_1变成新的根。

在原树中$k_2 > k_1$，因此k_2变成了k_1的右子树，X和Z仍然分别是k_1的左儿子和k_2的右儿子。

子树Y包含在k_1和k_2之间的结点，可以将子树Y变为k_2的左子树。这样，所有对顺序的要求都能得到满足

同理，对于情形4，我们对树进行左旋得到平衡树