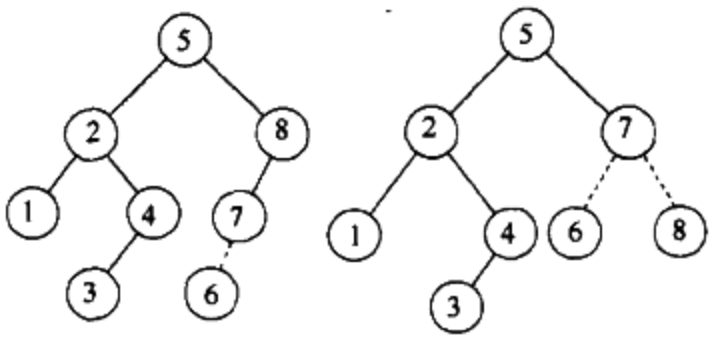


图 4-32　插入 6 破坏了 AVL 性质，而后经过单旋转又将性质恢复

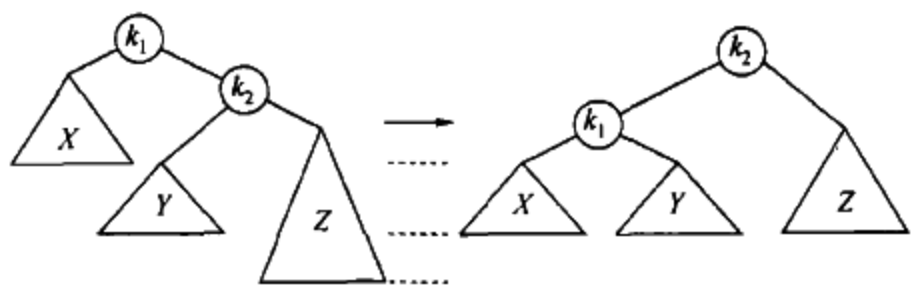
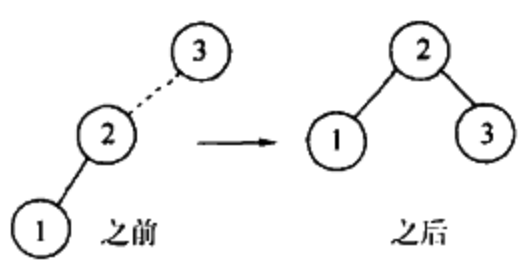
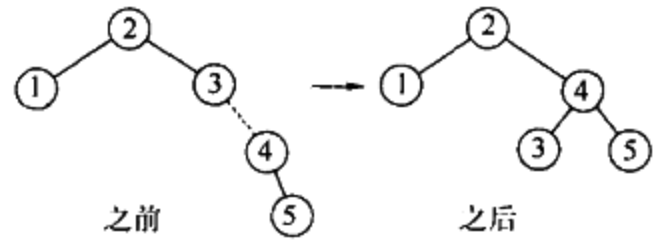


图 4-33　单旋转修复情形 4

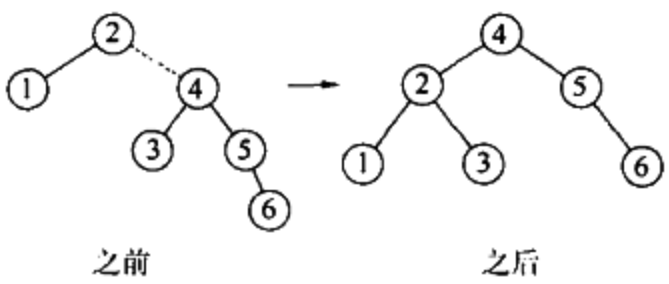
7。在插入关键字 1 时第一个问题出现了，AVL 性质在根处被破坏。我们在根与其左儿子之间施行单旋转修正这个问题。下面是旋转之前和之后的两棵树：



图中虚线连接两个节点，它们是旋转的主体。下面我们插入关键字为 4 的节点，这没有问题，但插入 5 就破坏了在节点 3 处的 AVL 性质，而通过单旋转又将其修正。除旋转引起的局部变化外，编程人员必须记住：树的其余部分必须被告知该变化。如本例中节点 2 的右儿子必须重新设置以链接到 4 来代替 3。这一点很容易忘记，从而导致树被破坏(4 就会是不可访问的)。



下面我们插入 6。这在根节点产生一个平衡问题，因为它的左子树高度是 0 而右子树高度为 2。因此我们在根处在 2 和 4 之间实施一次单旋转。



旋转的结果使得 2 是 4 的一个儿子，而 4 原来的左子树变成节点 2 的新的右子树。在该子树上的每一个关键字均在 2 和 4 之间，因此这个变换是成立的。我们插入的下一个关键字是 7，它导致另外的旋转：