2) + 1 给出。对于 $h=0$, $S(h)=1$; $h=1$, $S(h)=2$。函数 $S(h)$与斐波那契数密切相关，由此推出上面提到的关于 AVL 树的高度的界。

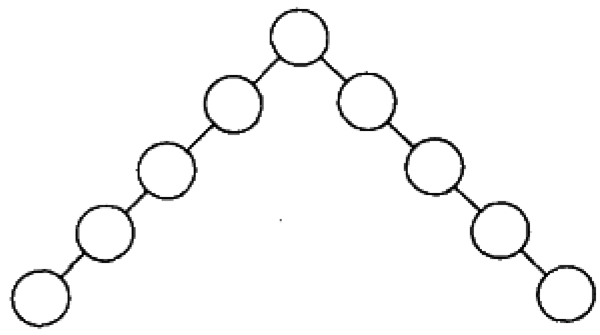

图 4-28　一棵坏的二叉树。只要求在根节点平衡是不够的

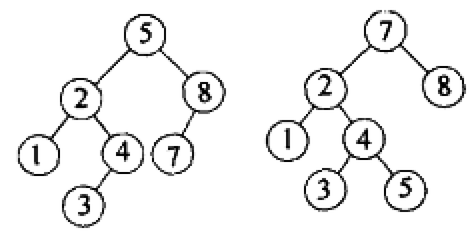

图 4-29　两棵二叉查找树，只有左边的树是 AVL 树

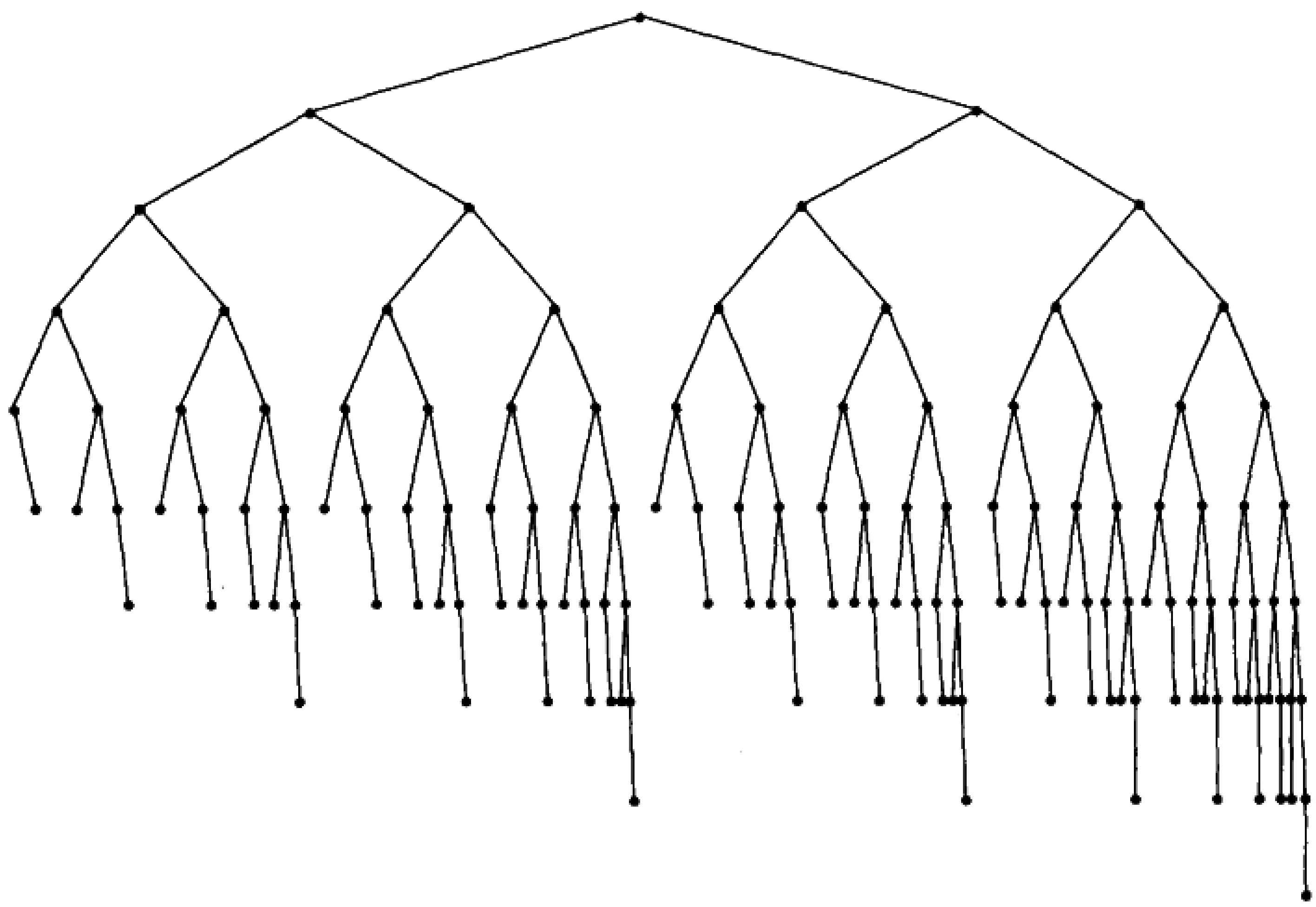

图 4-30　高度为 9 的最小的 AVL 树

因此，除去可能的插入外(我们将假设懒惰删除)，所有的树操作都可以以时间 $O(\log N)$执行。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，而插入操作隐含着困难的原因在于，插入一个节点可能破坏 AVL 树的特性(例如，将 6 插入到图 4-29 中的 AVL 树中将会破坏关键字为 8 的节点处的平衡条件)。如果发生这种情况，那么就要在考虑这一步插入完成之前恢复平衡的性质。事实上，这总可以通过对树进行简单的修正来做到，我们称其为**旋转**(rotation)。

在插入以后，只有那些从插入点到根节点的路径上的节点的平衡可能被改变，因为只有这些节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，可以发现一个节点，它的新平衡破坏了 AVL 条件。我们将指出如何在第一个这样的节点(即最深的节点)重新平衡这棵树，并证明这一重新平衡保证整个树满足 AVL 性质。

我们把必须重新平衡的节点叫做 α。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 α 点的两棵子树的高度差 2。容易看出，这种不平衡可能出现在下面四种情况中：