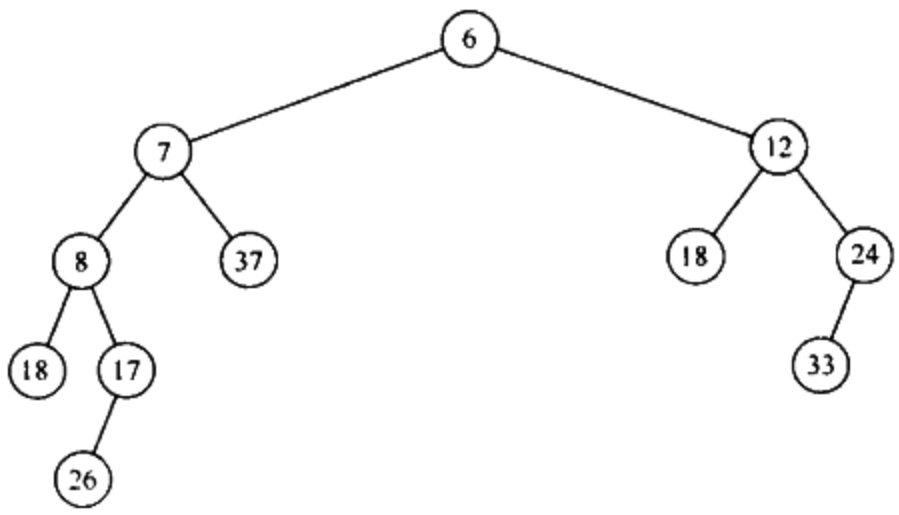

图 6-32 将 H_2 与 H_1 的右子堆合并的结果

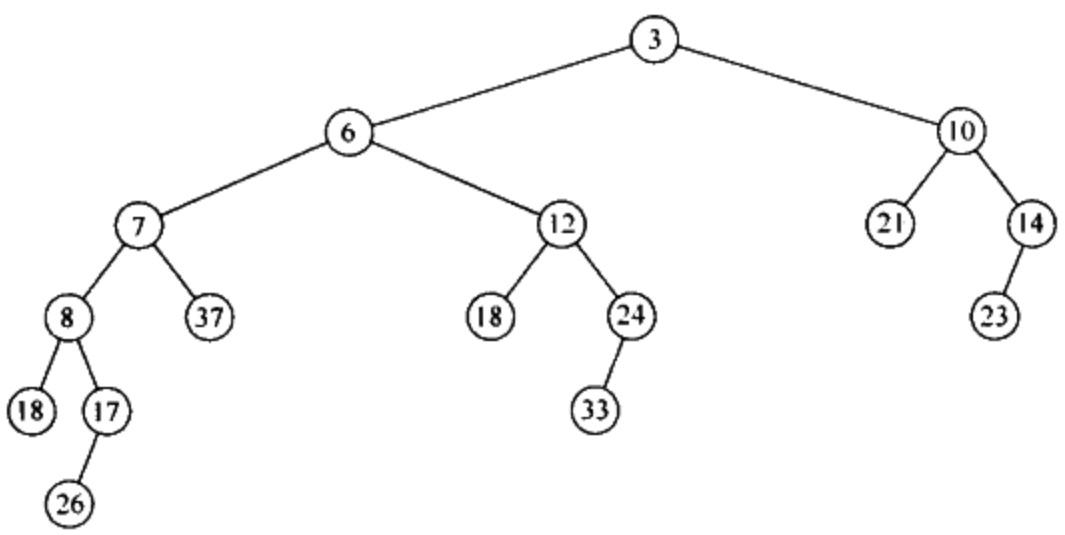

图 6-33 合并斜堆 H_1 和 H_2 的结果

整个树是左式的，但是容易看到这并不总是成立的：将 15 插入到新堆中将破坏左式性质。

我们也可像左式堆那样非递归地进行所有操作：合并右路径，除最后的节点外交换右路径上每个节点的左儿子和右儿子。经过几个例子之后，事情变得很清楚，由于除去右路径上最后的节点外的所有节点都将它们的儿子交换，因此最终效果是它变成了新的左路径(参见前面的例子以便使你自己确信)。这使得合并两个斜堆非常容易[⊖]。

斜堆的实现留作(平凡的)练习。注意，因为右路径可能很长，所以递归实现可能由于缺乏栈空间而失败，尽管在其他方面性能是可接受的。斜堆有一个优点，即不需要附加的空间保留路径长以及不需要测试以确定何时交换儿子。精确确定左式堆和斜堆的右路径长的期望值是一个尚未解决的问题(后者无疑更为困难)。这样的比较将更容易确定平衡信息的轻微遗失是否可由缺乏测试来补偿。

6.8 二项队列

虽然左式堆和斜堆都在每次操作以 $O(\log N)$时间有效地支持合并、插入和 deleteMin，但还是有改进的余地，因为我们知道，二叉堆以每次操作花费常数平均时间支持插入。二项队列支持所有这三种操作，每次操作的最坏情形运行时间为 $O(\log N)$，而插入操作平均花费常数时间。

⊖ 这与递归实现不完全相同(但服从相同的时间界)。如果一个堆的右路径用完而导致右路径合并终止，而我们只交换终止的那一点上面的右路径上那些节点的儿子，那么将得到与递归做法相同的结果。