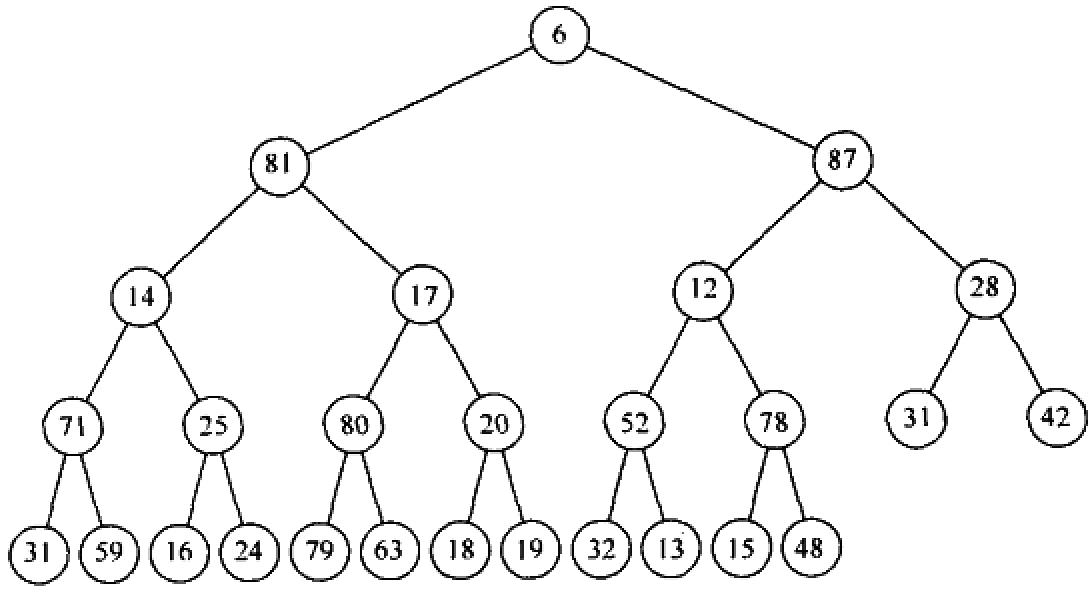

图 6-57 最小-最大堆

**e. 设我们想要支持操作 deleteMin、deleteMax 以及 merge。提出一种数据结构以时间 $O(\log N)$支持所有的操作。

6.19 合并图 6-58 中的两个左式堆。

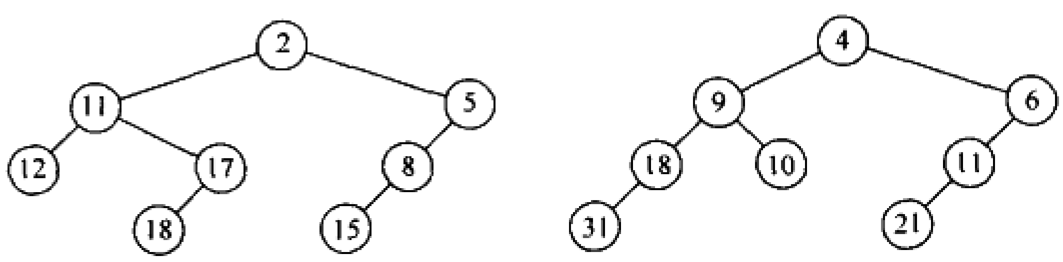

图 6-58 练习 6.19 和 6.26 的输入

6.20 写出依序将关键字 1 到 15 插入到一个初始为空的左式堆中的结果。

6.21 证明下述结论成立或证明其不成立：如果将关键字 1 到 2^k-1 依序插入到一个初始为空的左式堆中，那么结果形成一棵理想平衡树(perfectly balanced tree)。

6.22 给出生成最佳左式堆的输入的例子。

6.23 a. 左式堆能否有效地支持 decreaseKey?

b. 完成该功能需要哪些改变(如果可能的话)?

6.24 从左式堆中一个已知位置删除节点的一种方法是使用懒惰策略。为了删除一个节点，只要将其标记为被删除即可。当执行一个 findMin 或 deleteMin 时，若标记根节点被删除则存在一个潜在的问题，因为此时该节点必须被实际删除且需要找到实际的最小元，这可能涉及到删除其他一些已做标记的节点。在该策略中，这些 delete 花费一个单位，但一次 deleteMin 或 findMin 的开销却依赖于被作删除标记的节点的个数。设在一次 deleteMin 或 findMin 后作标记的节点比操作前减少 k 个。

*a. 说明如何以 $O(k \log N)$时间执行 deleteMin。

**b. 提出一种实现方法，通过分析，证明执行 deleteMin 的时间为 $O(k \log (2N/k))$。

6.25 我们可以以线性时间对一些左式堆执行 buildHeap 操作：把每个元素当作是单节点左式堆，把所有这些堆放到一个队列中，之后，让两个堆出队，合并它们，再将合并结果入队，直到队列中只有一个堆为止。

a. 证明该算法在最坏情形下为 $O(N)$。

b. 为什么该算法优于课文中描述的算法?