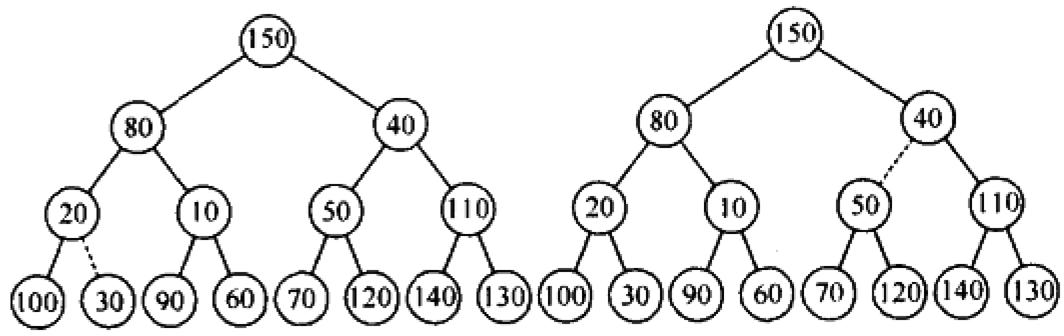

图 6-17 左：在 percolateDown(4)后；右：在 percolateDown(3)后

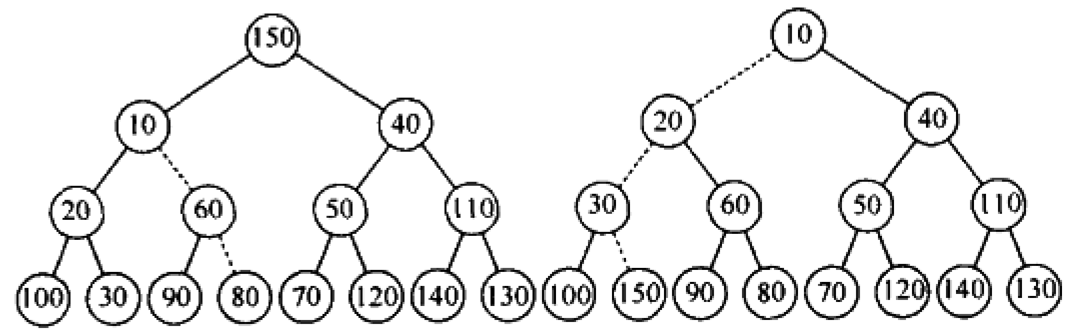

图 6-18 左：在 percolateDown(2)后；右：在 percolateDown(1)后

$$S = \sum_{i=0}^{h} 2^i(h-i)$$
$$= h + 2(h-1) + 4(h-2) + 8(h-3) + 16(h-4) + \cdots + 2^{h-1}(1) \tag{6.1}$$

两边乘以 2 得到方程

$$2S = 2h + 4(h-1) + 8(h-2) + 16(h-3) + \cdots + 2^h(1) \tag{6.2}$$

将这两个方程相减得到方程(6.3)。我们发现，非常数项差不多都消去了，例如，$2h - 2(h-1) = 2, 4(h-1) - 4(h-2) = 4$，等等。方程(6.2)的最后一项 2^h 在方程(6.1)中不出现；因此，它出现在方程(6.3)中。方程(6.1)中的第一项 h 在方程(6.2)中不出现；因此，$-h$ 出现在方程(6.3)中。我们得到

$$S = -h + 2 + 4 + 8 + \cdots + 2^{h-1} + 2^h = (2^{h+1} - 1) - (h+1) \tag{6.3}$$

该定理得证。■

一棵完全树不是理想二叉树，但我们得到的结果却是一棵完全树的节点高度的和的上界。由于一棵完全树节点数在 2^h 和 2^{h+1}之间，因此该定理意味着这个和是 $O(N)$，其中 N 是节点的个数。

虽然我们得到的结果对证明 buildHeap 是线性的而言是充分的，但是高度的和的界却不是尽可能的强。对于具有 $N = 2^h$ 个节点的完全树，我们得到的界大致是 $2N$。由归纳法可以证明，高度的和是 $N - b(N)$，其中 $b(N)$是在 N 的二进制表示法中 1 的个数。

6.4 优先队列的应用

我们已经提到优先队列如何在操作系统的设计中应用。在第 9 章，我们将看到优先队列如何在有效地实现几个图论算法中应用。此处，我们将介绍如何应用优先队列来得到两个问题的解答。

6.4.1 选择问题

我们将要考察的第一个问题是来自第 1 章的选择问题(selection problem)。回忆当时的输入是 N 个元素以及一个整数 k，这 N 个元素的集可以是全序集。该选择问题是要找出第 k 个最大的元素。