**d. 这些方法有通过将重物按重量预先排序而功能得到增强的吗?

6.38 设我们想要将操作 decreaseAllKeys(Δ)添加到堆的指令系统中。该操作的结果是堆中所有的关键字都将它们的值减少量 Δ。对于你所选择的堆的实现方法,解释所作的必要的修改使得所有其他操作都保持它们的运行时间而 decreaseAllKeys 以 $O(1)$运行。

6.39 两个选择算法中哪个具有更好的时间界?

参考文献

二叉堆首先在[28]中描述。构造它的线性时间算法来自[14]。

d-堆最初的描述见于[19]。最近的结果提出,4 叉堆在某些情形下可以改进二叉堆[22]。左式堆由 Crane[11]发现并在 Knuth[21]中描述。斜堆由 Sleator 和 Tarjan[24]开发。二项队列由 Vuillemin[27]发明; Brown 提供了详细的分析和经验性的研究,指出若能仔细地实现它们则在实践中性能很好[4]。

练习 6.7(b~c)取自[17]。练习 6.10(c)源于[6]。平均使用大约 $1.52N$ 次比较构造二叉堆的方法在[23]中描述。左式堆中的懒惰删除(练习 6.24)出自[10]。练习 6.36 的一种解法可在[8]中查到。

最小-最大堆(练习 6.18)原始描述见于[1]。这些操作的更有效的实现在[18]和[25]中给出。双端优先队列(double-ended priority queues)的另外一些表示形式是 deap 双端堆和菱形双端队列(diamond deque),细节可见于[5]、[7]和[9]。练习 6.18(e)的解法在[12]和[20]中给出。

理论上有趣的优先队列表示法是斐波那契堆(Fibonacci heap)[16],我们将在第 11 章中描述它。斐波那契堆使得所有的操作均以 $O(1)$摊还时间执行,但要除去删除操作,它是 $O(\log N)$。松堆(relaxed heaps)[13]得到最坏情形下完全相同的界(但 merge 除外)。[3]的过程对所有操作均得到最佳的最坏情形界。另外一种有趣的实现方法是配对堆(pairing heap)[15],将在第 12 章描述。最后,当数据由一些小的整数组成时能够正常工作的优先队列在[2]和[26]中描述。

1. M. D. Atkinson, J. R. Sack, N. Santoro, and T. Strothotte, "Min-Max Heaps and Generalized Priority Queues," *Communications of the ACM*, 29 (1986), 996–1000.
2. J. D. Bright, "Range Restricted Mergeable Priority Queues," *Information Processing Letters*, 47 (1993), 159–164.
3. G. S. Brodal, "Worst-Case Efficient Priority Queues," *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms* (1996), 52–58.
4. M. R. Brown, "Implementation and Analysis of Binomial Queue Algorithms," *SIAM Journal on Computing*, 7 (1978), 298–319.
5. S. Carlsson, "The Deap—A Double-Ended Heap to Implement Double-Ended Priority Queues," *Information Processing Letters*, 26 (1987), 33–36.
6. S. Carlsson and J. Chen, "The Complexity of Heaps," *Proceedings of the Third Symposium on Discrete Algorithms* (1992), 393–402.
7. S. Carlsson, J. Chen, and T. Strothotte, "A Note on the Construction of the Data Structure 'Deap'," *Information Processing Letters*, 31 (1989), 315–317.
8. S. Carlsson, J. I. Munro, and P. V. Poblete, "An Implicit Binomial Queue with Constant Insertion Time," *Proceedings of First Scandinavian Workshop on Algorithm Theory* (1988), 1–13.
9. S. C. Chang and M. W. Due, "Diamond Deque: A Simple Data Structure for Priority Deques," *Information Processing Letters*, 46 (1993), 231–237.
10. D. Cheriton and R. E. Tarjan, "Finding Minimum Spanning Trees," *SIAM Journal on Computing*, 5 (1976), 724–742.
11. C. A. Crane, "Linear Lists and Priority Queues as Balanced Binary Trees," *Technical Report*