WEAK-HEAP SORT

RONALD D. DUTTON

Dept. of Computer Science, University of Central Florida, Orlando, FL 32816, USA

Abstract.

A data structure called a weak-heap is defined by relaxing the requirements for a heap. The structure can be implemented on a 1-dimensional array with one extra bit per data item and can be initialized with \( n \) items using exactly \( n - 1 \) data element compares. Theoretical analysis and empirical results indicate that it is a competitive structure for sorting. The worst case number of data element comparisons is strictly less than \((n - 1)\log n + 0.086013n\) and the expected number is conjectured to be approximately \((n - 0.5)\log n - 0.413n\).

CR Categories: E.1, F2.2.

Keywords: Algorithms, data structures, heap, priority queue.

1. Introduction.

This paper introduces a sorting algorithm based upon the weak-heap data structure defined initially in [3]. A weak-heap uses a relaxed, or weakened, version of the standard heap property first described by Williams in [11] for the sorting algorithm heap sort. A tree in which the value in each node, other than the root, is less than or equal to that in its parent node has the heap property. We assume the maximum heap property throughout. A binary tree with the heap property (and with supporting algorithms) is often simply called a heap. In an implicit implementation, nodes with less than two children appear on the bottom two levels of the tree with those on the bottom level being as far to the left as possible.

In terms of the number of comparisons, the information theoretic lower bound on the maximum number of data compares for comparison based sequential sorting algorithms is \( \log(n!) \), approximately \( n \log n - 1.442695n \). Logarithms are assumed to be taken base two. The literature abounds with algorithms approaching this limit in one sense or the other. Most variants of Quicksort [5] have a worst case bound of order \( n^2 \), but have a much more efficient expected case. The best-of-three version called Clever Quicksort has been analyzed (see [9]) to have an expected number of compares of approximately \( 1.188n \log(n - 1) - 2.255n + 1.188 \log(n - 1) + 2.507 \).

The original versions of heapsort \([4, 11]\) have a worst case complexity of \(2n \log n\). A variant introduced by McDiarmid and Reed, called Bottom-Up Heapsort \([6, 9]\), was shown to be bounded by \(1.5n \log n - 0.4n\). Recently, Wegner \([10]\) introduced a clever implementation of Bottom-Up Heapsort, using \(n\) additional bits, which he called MDR-Heapsort and showed it had a worst case bound of \((n + 1) \log n + 1.086072n\).

No exact average case results exist for any of the Heapsort variants because the intermediate heaps formed during the execution of the algorithms quickly become nonrandom and it is not clear how to analyze this phenomenon. Simplifying assumptions, supported by simulations, though have given results indicating that the average is \(n \log n + f(n)n\) where \(f(n)\in[0.355, 0.39]\) for Bottom-Up Heapsort and \(f(n)\in[-0.05, 0.10]\) for MDR-Heapsort (see \([6, 9]\) and \([10]\), respectively).

The Weak-Heap sorting algorithm described here, also using \(n\) additional bits, is shown to have a worst case number of compares that is less than \((n - 1) \log n + 0.086013n\) and we conjecture the average number required is approximately \((n - 0.5) \log n - 0.413n\).

A weak-heap is formally defined as a binary tree where:
1. the value in any node is at least the value of any node in its right subtree,
2. the root has no left subtree, and
3. nodes having fewer than two children, except the root, appear on the bottom two levels of the tree.

Weak-heaps do not require leaf nodes on the bottom level to be as far to the left as possible, nor that a relationship exists between a node's value and the values in the left subtree of that node. Condition (2) guarantees the root contains the largest value. Because weak-heaps are a variant of heaps, they bear a striking similarity to other variants of heaps, e.g., general heaps and binomial heaps (see \([2, 7, 8]\)). One can often incorporate the features of one variant within those of another by suitable restrictions and/or relaxations. Nevertheless, different variants can provide subtle insights into algorithm design, implementation and analysis.

Figure 1 gives two views of a weak-heap: the second being a more detailed version of the first. The root contains the largest value and, for \(1 \leq i \leq k\), all values in subtree \(T_i\) are less than or equal to the value in the corresponding node \(i\), that is, the subtree induced by node \(i\) and \(T_i\) is a sub-weak-heap. In fact, any node and its right subtree is a weak-heap. Notice, because of (3) in the definition, the subtree \(T_i\) has between \(2^{k-i} - 1\) and \(2^{k-i+1} - 1\) nodes.

For any node \(x\) in a weak-heap, let

\[S_x = \{\text{Rson}(x)\} \cup \{y \mid y \text{ is reachable from } \text{Rson}(x) \text{ by left branches only}\}\]

with \(x\) being the unique "Grandparent" of the nodes in \(S_x\); i.e.,

\[x = \text{Gparent}(y) \text{ if and only if } y \in S_x.\]

In Figure 1, \(S_h = \{1, 2, \ldots, k\}\) and \(h\) is the grandparent of all nodes \(i\), \(1 \leq i \leq k\).