The 4-ary AM-heap

Haejae Jung
Department of Information Communication Edgineering
Andong National University, Andong, South Korea
hjjung@anu.ac.kr

Abstract. The 4-ary AM-heap is a heap to implement a priority queue. It has regularity in indexing scheme in which the first child of every node has an index of a multiple of 4 and the leftmost node at each level has an index of a power of 4. According to our experimental results, the 4-ary AM-heap is up to 13 percent faster than AM-heap in the hold test model and the post-order heap shows the worst performance.

Keywords: heap, priority queue, data structures, computer algorithm.

1 Introduction

Heaps have a wide range of applications such as scheduling, event-driven simulation, sorting, shortest paths computation, and extracting elements with the highest priority [9],[10],[11]. Some implicit heaps with constant amortized insertion and O(logn) deletion time complexities have been published and representative heaps are post-order heap by N. Harvey and K. Zatloukal and AM-heap by H. Jung [1],[2]. The post-order heap was published by N. Harvey and K. Zatloukal [1]. This heap is simpler than the implicit binomial heap[4]. The AM-heap by H. Jung is much simpler than the post-order heap since it uses the simple indexing scheme of the conventional binary heap by J. Williams [3]. For the post-order heap [1], each node is indexed in the postorder that is different from the conventional heap, as can be seen in Figure 1. Figure 1 shows a post-order heap with four component heaps rooted at node 7, 10, 11, 12 and node 12 is the last root. The post-order heap uses a full binary tree. In this data structure, insertion is performed in the post-order and deletion in the reverse postorder. The right child of node \( n \) is simply node \( n-1 \). The left child of node \( n \) can be calculated by subtracting the size of the right subtree from \( n-1 \). They maintained a bit array \( d[i] \) to find the root of the left neighbor component heap from the root of a component heap where \( d[i] \) is set to 0/1 if node \( i \) is left/right child of its parent.
AM-heap uses a complete binary tree, instead of a full binary tree. Also, the AM-heap utilizes the simple indexing scheme of the conventional heap by J. Williams, setting the root index to 1. Elements are inserted in the post-order, starting from the leftmost leaf node. The indexes of the parent and child of a node can be calculated by the expressions used in the conventional heap. Elements are removed in the reverse post-order, as with the post-order heap. To find the root index $r_l$ of the left neighbor component heap from the root $r$ of a component heap, the AM-heap simply utilizes the node index: starting from node $r$, we repeatedly move $r$ up until $r$ becomes odd. Then, $r_l$ is simply $r-1$.

This paper proposes 4-ary AM-heap with constant amortized insertion time complexity to implement priority queues. This data structure uses a variant of 4-ary tree in which only the root has three children and the rest have 4 children, to utilize the regularity of index numbers. The proposed heap is based on a 4-ary complete tree, not on a 4-ary full tree. In this paper, we define 4-ary full tree as a tree in which every leaf node is at the same level and every internal node has four children. We also define 4-ary complete tree as a 4-ary tree in which there may be some missing leaf nodes on the right from a 4-ary full tree.

In the next section, we describe the 4-ary AM-heap for implementing priority queues. In section 3, we show the experimental results conducted by us. And then, the conclusion of this paper follows.

2 4-ary AM-heap

This section presents the 4-ary AM-heap in which each node has 4 children except for the root. As can be seen in Fig. 2, the 4-ary AM-heap adopts the 4-ary tree structure of [7] to utilize the regularity of its indexes: that is, the leftmost child of every node has an index of a multiple of 4.
In the 4-ary AM-heap, the parent index of node \( i \) is calculated by a simple expression \( \lceil i/4 \rceil \) and the 4 children indexes of node \( i \) are calculated by \( 4i, 4i+1, 4i+2, 4i+3 \) each.

Using the structure of the 4-ary AM-heap, we can describe insert and deleteMin operations: for these operations, we maintain two variables \( \text{lastRoot} \) and \( \text{lastLeaf} \) that indicate the root and the rightmost leaf node of the rightmost component heap, respectively.

[insert operation: input element \( x \)]
1. If node \( \text{lastRoot} \) is the rightmost child of its parent, put \( x \) in the parent of node \( \text{lastRoot} \).
2. Otherwise, a new component with a single node is created. Put \( x \) in the node of the new component heap.
3. Update variables \( \text{lastRoot} \) and \( \text{lastLeaf} \).

[deleteMin operation]
1. Find the root node \( \text{minRoot} \) with the minimum element. This is done by traversing the roots of all component heaps, starting from node \( \text{lastRoot} \).
2. Move the element of node \( \text{lastRoot} \) to node \( \text{minRoot} \). And heapify the component heap rooted at node \( \text{minRoot} \).
3. Adjust \( \text{lastRoot} \) and \( \text{lastLeaf} \).

3 Experimental results

We experimented with hold test model: starting with an initially empty heap, \( n \) elements are inserted. And then an intermixed sequence of \( 2n \) insert and delete operations are performed. The time taken to perform this intermixed sequence of \( 2n \) operations is measured.

Figure 3 shows the measured run times and speedup relative to the AM-heap in the hold model. We can see that the 4-ary AM-heap is about 13 percent faster than AM-heap and the post-order heap shows the worst performance.
Fig. 3. run time and speedup relative to AM-heap in hold model

References

3. J. Williams, Algorithm 232 Heapsort, Communications of the ACM, 7, 1 (1964)