PERGAMON Computers and Mathematics with Applications 43 (2002) 911–916

An International Journal
computers & mathematics with applications

www.elsevier.com/locate/camwa

The Bounds of Min-Max Pair Heap Construction

R. A. Chowdhury, M. Z. Rahman and M. Kaykobad
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka-1000, Bangladesh
shaikat2@yahoo.com, zia@bangla.net, kaykobad@cse.buet.edu

(Received July 1999; revised and accepted March 2000)

Abstract—In this paper, lower and upper bounds for min-max pair heap construction has been presented. It has been shown that the construction of a min-max pair heap with n elements requires at least $2.07n$ element comparisons. A new algorithm for creating min-max pair heap has been devised that lowers the upper bound to $2.43n$.

Keywords—Algorithm, Heap, Min-max pair heap, Min-max heap, Double-ended priority queue.

1. INTRODUCTION

The MinMax Heap structure, introduced by Strothotte [1], is a structure for the implementation of double-ended priority queue. It is based on heap structure under the notion of min-max ordering: values stored at nodes on even (odd) levels are smaller than or equal to (respectively, greater than) values stored at their descendants. This structure can be constructed in linear time. *FindMin, FindMax* operations are performed in constant time and *Insert*(x), *DeleteMin*, and *DeleteMax* in logarithmic time using this structure. A sublinear merging algorithm for this structure is given with the relaxation of strict ordering [2].

The min-max pair heap, introduced by Olariu *et al.* [3], has the advantage that his double-ended priority queue supports merging in sublinear time. Recently, Rahman *et al.* [4] have improved the algorithms for min-max pair heap. These improved algorithms for min-max pair heap outperform the original algorithms of Strothotte in all aspects excepting creation, for which the latter requires $2.15n$ comparisons [1], whereas the improved min-max pair heap creation requires $2.566n$ comparisons [4]. This paper will investigate further to obtain the bounds of min-max pair heap construction.

2. MIN-MAX PAIR HEAPS

Definition. *A min-max pair heap is a binary tree H featuring the heap-shape property, such that every node in $H[i]$ has two fields, called the min field and the max field, and such that H*

We would like to thank M. Manzur Murshed of the Australian National University for his generous support.

0898-1221/02/$ - see front matter Typeset by $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-TEX
PII: S0898-1221(01)00331-5

has min-max ordering: for every i ($1 \le i \le n$), the value stored in the min field of $H[i]$ is the smallest of all values in the subtree of H rooted at $H[i]$; similarly the value stored in the max field of $H[i]$ is the largest key stored in the subtree of H rooted at $H[i]$.

However, we can consider those two heaps separately by taking min(max) elements. We name the min heap as A and max heap as B. Then, we can show their relationship by a Hasse diagram. For example, a min-max pair heap is shown in Figure 1. Its corresponding Hasse diagram is shown in Figure 2.

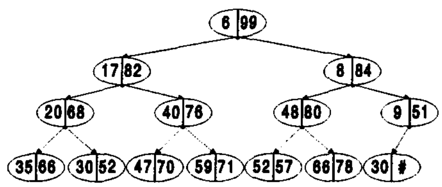


Figure 1. A sample min-max pair heap of height 3.

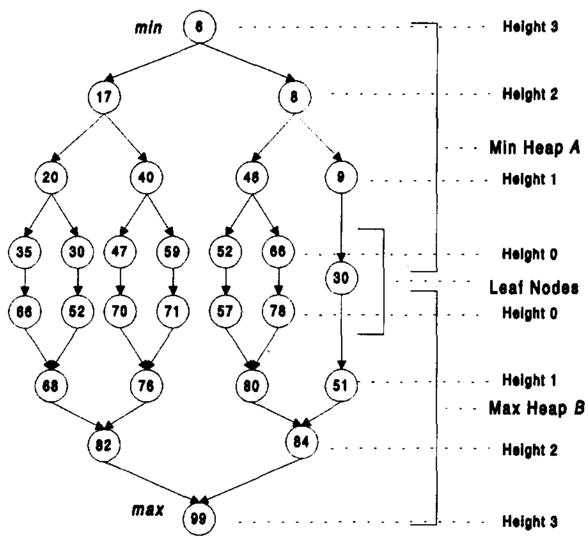


Figure 2. The Hasse diagram for min-max pair heap of Figure 1.

3. WORST CASE COMPLEXITIES OF MIN, MIN-MAX, AND MIN-MAX PAIR HEAPS

The known complexity for min heaps, min-max heaps, and min-max pair heaps is shown in Table 1 [1,4]. The results of [4] can be easily obtained by analyzing the Hasse diagram for min-max pair heap. In Table 1, the function $g(x)$ is defined as follows: $g(x) = 0$ for $x \le 1$ and $g(n) = g(\lceil \lg(n) \rceil) + 1$.

Table 1. Worst-case complexities for min-heaps, min-max heaps, and min-max pair heaps.

	Min-Heaps	Min-Max Heaps	Min-Max Pair Heaps
Create	$1.625n$	$2.15\ldots n$	$2.566\ldots n$
Insert	$\lg(\lg(n+1))$	$\lg(\lg(n+1))$	$\lg(\lg(n+2))$
DeleteMin	$\lg(n) + g(n)$	$1.5\lg(n) + \lg(\lg(n))$	$\lg(n+2) + \lg(\lg(n+2))$
DeleteMax	$0.5n + \lg(\lg(n))$	$1.5\lg(n) + \lg(\lg(n))$	$\lg(n+2) + \lg(\lg(n+2))$

4. LOWER BOUND ANALYSIS

First, we will calculate the information theoretic lower bound for the construction of min-max pair heaps.

THEOREM 1. *The number of comparisons necessary to construct a complete min-max pair heap of height h is at least $2.07286n$, where $n(=2^{h+2}-2)$ is the number of elements in the heap.*

PROOF. Let $N(h)$ be the number of distinct min-max pair heaps that can be constructed from $2^{h+2}-2$ elements. This min-max pair heap can be viewed as the minimum and maximum element connecting to two min-max pair heaps of height $h-1$ each. Hence,

$$N(h) = \binom{2^{h+2}-4}{2^{h+1}-2}[N(h-1)]^2 = \frac{(2^{h+2}-4)!}{(2^{h+1}-2)!\,(2^{h+1}-2)!}[N(h-1)]^2,$$

$$\Rightarrow \frac{N(h)}{(2^{h+2}-2)!} = \frac{1}{(2^{h+2}-2)\,(2^{h+2}-3)}\left\{\frac{N(h-1)}{(2^{h+1}-2)!}\right\}^2,$$

$$\Rightarrow \frac{(2^{h+2}-2)!}{N(h)} = \left(2^{h+2}-2\right)\left(2^{h+2}-3\right)\left\{\frac{(2^{h+1}-2)!}{N(h-1)}\right\}^2.$$

By the information-theoretic lower bound, we know that the minimum number of comparisons, on the average, needed to build a min-max pair heap of n elements is at least

$$\lg\left(\frac{(2^{h+2}-2)!}{N(h)}\right) = \lg\left(2^{h+2}-2\right) + \lg\left(2^{h+2}-3\right) + 2\lg\left(\frac{(2^{h+1}-2)!}{N(h-1)}\right).$$

Assuming

$$C(h) = \lg\left(\frac{(2^{h+2}-2)!}{N(h)}\right),$$

we get

$$C(h) = \lg\left(2^{h+2}-2\right) + \lg\left(2^{h+2}-3\right) + 2C(h-1) = \sum_{i=0}^{h} 2^i\left[\lg\left(2^{h+2-i}-2\right) + \lg\left(2^{h+2-i}-3\right)\right]$$

$$= \sum_{i=0}^{h-l} 2^i\left[\lg\left(2^{h+2-i}-2\right) + \lg\left(2^{h+2-i}-3\right)\right]$$

$$+ \sum_{i=h-l+1}^{h} 2^i\left[\lg\left(2^{h+2-i}-2\right) + \lg\left(2^{h+2-i}-3\right)\right], \qquad [h \geq l \geq 0],$$

$$\leq \sum_{i=0}^{h-l} 2^{i+1}(h+2-i) + \sum_{j=2}^{l+1}\frac{2^{h+2}}{2^j}\lg\left\{\left(2^j-2\right)\left(2^j-3\right)\right\}$$

$$= 2^{h+2}\left[\frac{l+3}{2^l} - \frac{h-4}{2^{h+1}} + \sum_{j=2}^{l+1}\frac{1}{2^j}\lg\left\{\left(2^j-2\right)\left(2^j-3\right)\right\}\right].$$

Hence,

$$\lim_{h\to\infty} C(h) \leq \lim_{h\to\infty}\frac{2^{h+2}}{2^{h+2}-2}\left[\frac{l+3}{2^l} - \frac{h-4}{2^{h+1}} + \sum_{j=2}^{l+1}\frac{1}{2^j}\lg\left\{\left(2^j-2\right)\left(2^j-3\right)\right\}\right]n$$

$$= \left[\frac{l+3}{2^l} + \sum_{j=2}^{l+1}\frac{1}{2^j}\lg\left\{\left(2^j-2\right)\left(2^j-3\right)\right\}\right]n.$$

Taking $l=10$, $\lim_{h\to\infty} C(h) \leq 2.07286n$. ∎

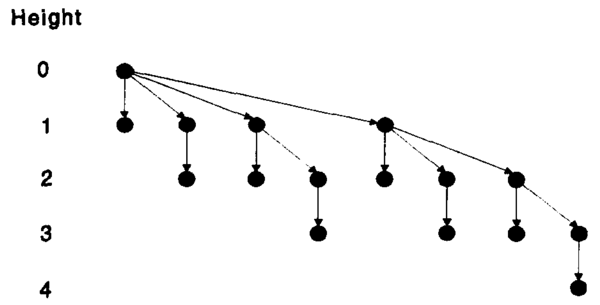


Figure 3. Binomial tree of height 4.

5. UPPER BOUND FOR MIN-MAX PAIR HEAP

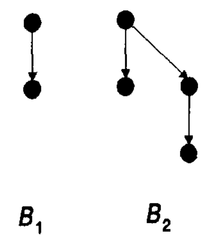
THEOREM 2. *A min-max pair heap of* $n(= 2^{h+2})$ *nodes can be constructed in at most* $2.4311n$ *comparisons in the worst case.*

PROOF. We describe an algorithm for the creation of min-max pair heap first and then analyze it to show that in the worst case it requires no more than $2.4311n$ comparisons.

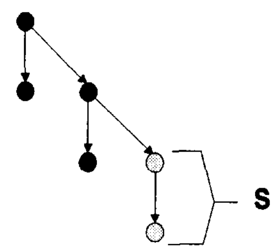
We start by constructing a binomial tree structure B_k of height k from n nodes. By R_k we denote the root of the binomial tree B_k. (See Figure 3.)

(1) Then, the smallest element is identified. It is the root of the whole structure.

(2) If the child of the root is labeled from $0 \ldots k-1$ from left to right, then the child i is the root of the subtree B_i.



(a) B_1 and B_2.



(b) The case when $R_1 < R_2$.

Figure 4.

(3) Figure 4a shows B_1 and B_2. Now, we compare R_1 with R_2 to get a $B_1 + B_2$ structure shown in Figure 5. If $R_1 \geq R_2$, then we make R_1 as a child of R_2 and easily get the $B_1 + B_2$ structure. If $R_1 < R_2$, then we obtain a structure shown in Figure 4b. Moving the subtree S to the root, we obtain the $B_1 + B_2$ structure.

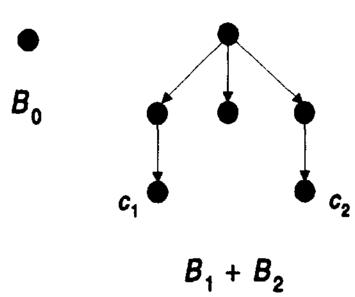


Figure 5.

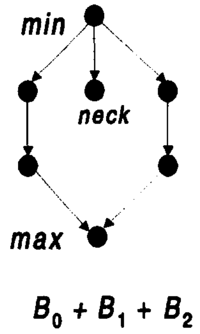


Figure 6.

(4) We next compare C_1 and C_2 in Figure 5 to obtain the chain of maximum sons. Then, we insert R_0 into that chain using binary insertion. Thus, we have now a min-max pair structure as is shown in Figure 6, with the exception of one node hanging from the root. We call this node the *neck* of the structure. Steps 3 and 4 require four comparisons.

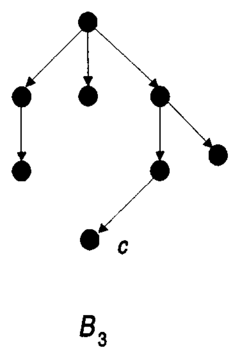


Figure 7. Redrawn B_3.

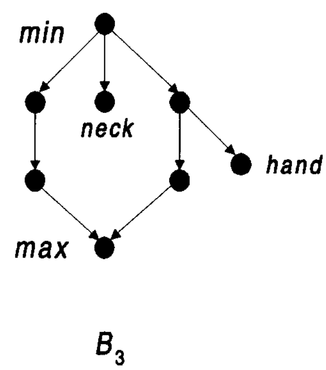


Figure 8. Min-max pair heap with *neck* and *hand*.

(5) For the subtree B_3, we have a structure that is redrawn in Figure 7. Then, we insert C into the leftmost three-element chain from root to produce a structure as shown in Figure 8. Here, we also have a *neck* and another node branching from the min-max pair heap. We call this node the *hand*. This step requires two comparisons.

(6) Now, we have two structures, one that is shown in Figure 6 and the other shown in Figure 8. In general, we need to construct a min-max pair heap with neck of height $h + 1$ from a min-max pair heap with neck of height h and a min-max pair heap with both neck and hand.

We compare the two *min* elements to find the element that will become the *min* element of the structure of height $h + 1$. Then, we find out the chain of minimum sons from the same subheap whose *min* element has been chosen. Thus, $h + 1$ comparisons are made. We insert *hand* into that chain excluding minimum since already we know that it is smaller than *hand*. This requires $\lceil \lg(2h + 2) \rceil$ comparisons.

Next, we compare the *max* elements, and the maximum becomes the *max* element of the structure of height $h + 1$. Then, we find out the chain of maximum sons in the chosen subheap. This requires $h + 1$ comparisons. Then, we will insert the *neck* of the chosen subheap into the chain excluding minimum. This requires $\lceil \lg(2h + 2) \rceil$ comparisons. The neck that has not been inserted is surely greater than the min element, and it becomes the neck of current structure. Thus, we obtain another min-max pair heap of height $h + 1$ having *neck*.

Thus, this step is performed recursively until we reach the root of the binomial tree and requires $2(h + 1) + 2\lceil \lg(2h + 2) \rceil$ comparisons at height h.

(7) Thus, we continue merging and reach to merge at the root of the binomial tree. Since the minimum is already there, no *min* element adjustment is required. We just perform *max* element adjustment. Then, the unused *neck* and *hand* become the *neck* and *hand* of the whole structure. Thus, at the root we obtain a min-max pair heap having *neck* and *hand*. This requires a total of $h + \lceil \lg(2h) \rceil$ comparisons.

(8) At last, we need to insert the *neck* and *hand* of the structure to the min-max pair heap. This step is performed only at the root and requires $2(h + 1) + 2\lceil \lg(2h + 2) \rceil$ comparisons. Thus, this step will not affect the order of the construction.

Now, to construct a min-max pair heap of height h, a binomial tree of height $h+2$ is constructed first. This step requires $2^{h+2} - 1$ comparisons. Let $C(h)$ be the additional cost to make a min-max pair heap from the B_{h+2}. We can easily construct a recurrence relation for $C(h)$ from the above description as

$$C(h) = 4 + \sum_{i=1}^{h-1} C(i) + \sum_{i=1}^{h-2} 2\{i + 1 + \lceil \lg(2i+2) \rceil\} + h + \lceil \lg(2h) \rceil.$$

Subtracting $C(h)$ from $C(h+1)$,

$$C(h+1) - C(h) = C(h) + 2h + 1 + 2\lceil \lg(2h) \rceil + \lceil \lg(2h+2) \rceil - \lceil \lg(2h) \rceil,$$
$$\Rightarrow C(h+1) = 2C(h) + 2h + 1 + \lceil \lg(2h) \rceil + \lceil \lg(2h+2) \rceil.$$

With boundary condition $C(1) = 2$, we have

$$C(h) = 2^h + \sum_{i=1}^{h-1} \left[2^{i-1}\{2(h-i) + 1 + \lceil \lg(2h-2i) \rceil + \lceil \lg(2h-2i+2) \rceil\} \right],$$
$$\Rightarrow C(h) = \frac{7}{2}2^h - 2h - 3 + \sum_{i=1}^{h-1} \left[2^{i-1}\{\lceil \lg(2h-2i) \rceil + \lceil \lg(2h-2i+2) \rceil\} \right],$$
$$\therefore \lim_{h \to \infty} C(h) = 2.4311n.$$

Thus, the theorem is proved. ∎

6. CONCLUDING REMARKS

We have analyzed the lower and upper bounds for min-max pair heap creation. We have given a constructive upper bound by presenting a new algorithm for its construction. This appears to be the first attempt to give bounds for the creation of min-max pair heaps. A gap between the information theoretic lower bound and the attained upper bound indicates the possibility of further improvement of the results.

REFERENCES

1. M.D. Atkinson, J.R. Sack, N. Santoro and Th. Strothotte, Min-max heaps and generalized priority queues, programming techniques and data structures, *Comm. ACM* **29** (10), 996–1000, (October 1986).
2. Y. Ding and M.A. Weiss, The relaxed min-max heap—A mergeable double-ended priority queue, *Acta Informatica* **30**, 215–231, (1993).
3. S. Olariu, C.M. Overstreet and Z. Wen, A mergeable double-ended priority queue, *Computer Journal* **34**, 423–427, (1991).
4. M.Z. Rahman, R.A. Chowdhury and M. Kaykobad, Improvements in double ended priority queues, In *Proceedings of International Conference on Computer and Information Technology*, pp. 1–5, Sylhet, Bangladesh, (1999).