Towards a Final Analysis of Pairing Heaps

Seth Pettie*
Max Planck Institut für Informatik

Abstract

Fredman, Sedgewick, Sleator, and Tarjan proposed the pairing heap *as a self-adjusting, streamlined version of the Fibonacci heap. It provably supports all priority queue operations in logarithmic time and is known to be extremely efficient in practice. However, despite its simplicity and empirical superiority, the pairing heap is one of the few popular data structures whose basic complexity remains open. In this paper we prove that pairing heaps support the deletemin operation in optimal logarithmic time and all other operations (insert, meld, and decreasekey) in time* $O(2^{2\sqrt{\log\log n}})$. *This result gives the* first *sub-logarithmic time bound for decreasekey and comes close to the lower bound of* $\Omega(\log\log n)$ *established by Fredman.*

Pairing heaps have a well known but poorly understood relationship to splay trees and, to date, the transfer of ideas has flowed in one direction: from splaying to pairing. One contribution of this paper is a new analysis that reasons explicitly with information-theoretic measures. Whether these ideas could contribute to the analysis of splay trees is an open question.

1 Introduction

Twenty years ago Fredman et al. [5] proposed the *pairing heap*, a simple priority queue data structure that they conjectured to be as efficient as Fibonacci heaps [6]. In their original analysis Fredman et al. proved that all priority queue operations (insert, deletemin, meld, and decreasekey) were supported in $O(\log n)$ amortized time, where n is the size of the heap. Although logarithmic time is optimal for deletemin, theoretically efficient priority queues like the Fibonacci heap enjoy $O(1)$ amortized time for insert, meld, and decreasekey.

In the intervening years pairing heaps have become the priority queue of choice in applications requiring the decreasekey operation. Stasko & Vitter [8] demonstrated experimentally that the cost of *decreasekey* is virtually, if not asymptotically, constant, and Moret & Shapiro [10] concluded that in Prim's minimum spanning tree algorithm pairing heaps are faster than the alternatives. These included binary heaps, splay heaps, Fibonacci heaps, and others. The pairing heap is now included in implementations of the GNU C++ library and the LEDA library [9].

Fredman [4] proved the remarkable result that on a specific distribution of operation sequences, no (generalized) pairing heap can perform optimally. Specifically, there are sequences of n insertions, n deletions, and $n\log n$ decreasekeys that take $\Omega(n\log n\log\log n)$ time to execute. He concluded that the complexity of the decreasekey operation lies somewhere between $\Omega(\log\log n)$ and $O(\log n)$.

The difference between optimality and suboptimality is not as stark as one might think. Because the user of a priority queue has already resigned himself to paying $\Theta(n\log n)$ time for n deletemins, a non-constant cost for decreasekey will only be detectable for *sufficiently dense* input sequences. (*Density* is the ratio of decreasekeys to deletemins.) For example, if Fredman's lower bound turns out to be tight, this would imply that in the standard graph applications (shortest paths, weighted matchings, minimum spanning trees, etc.) pairing heaps are no worse than Fibonacci heaps when the number of edges is $O(n\log n/\log\log n)$. Fredman's lower bound only establishes that the pairing heap is suboptimal in the density range $\omega(\log n/\log\log n)$ to $o(\log n\log\log n)$; for other ranges its complexity remains open.

In this paper we provide the first proof that pairing heaps support decreasekeys in sub-logarithmic time.

Theorem 1 *The pairing heap executes a mixed sequence of* m *priority queue operations, including* n *deletemins, in* $O(n\log n + m2^{2\sqrt{\log\log n}})$ *time.*

The $2^{2\sqrt{\log\log n}}$ function lies in the asymptotic spectrum between $\omega(\text{poly}(\log\log n))$ and $o(\log^{\epsilon} n)$. It is an ugly bound and undoubtedly not tight. Nonetheless, we can now claim with certainty that the pairing heap is asymptotically optimal for any density $O(\log n/2^{2\sqrt{\log\log n}})$.

*Supported by an Alexander von Humboldt Postdoctoral Fellowship.
Email: pettie@mpi-inf.mpg.de.

The Pairing Heap. The structure consists of a single rooted tree where the children of a node are assigned some left-to-right ordering. Each node is identified with a *key* and the key of a parent is no larger than the key of any child. (It is *heap ordered*.) During the execution of an operation there may be multiple rooted trees. If x_1, x_2 are two roots then *linking* x_1 and x_2 either makes x_1 the leftmost child of x_2 or vice versa, depending on which node's key is smaller. The operations are implemented as follows. An insert creates a new node with a given key and links it with the existing root. A meld of two given heaps simply links their roots. A decreasekey operation is given both a node and the heap to which it belongs. It detaches the subtree rooted at the given node, reduces the node's key appropriately, then links the given node with the existing heap root.

By the heap-order property the minimum node is necessarily at the root. A deletemin proceeds by detaching every child of the root then linking children in a prescribed fashion until one tree remains. Several linking strategies have been proposed [5, 8, 3]. In this paper we consider the standard 2-pass version given in [5]. Let $x_1, x_2, \ldots, x_k$ be the children of the root in left-to-right order. In the *coupling* pass we link the pairs $(x_1, x_2), (x_3, x_4), (x_5, x_6), \ldots$. Let $y_1, y_2, \ldots, y_\ell$ be the winners of these linkings, where $y_\ell = x_k$ if k is odd. In the *accumulation* pass we link y_ℓ with $y_{\ell-1}$, the winner of this comparison with $y_{\ell-2}$, and so on; in general, y_i is linked with the winner of $y_{i+1}, \ldots, y_\ell$.

Previous Work. Fredman et al. [5] proved that the 2-pass version of deletemin takes $O(\log n)$ amortized time. They also considered *arbitrary coupling*, *multipass*, and *backward 2-pass*.[1] It was shown that arbitrary coupling takes $\Theta(\sqrt{n})$ time and that multipass takes $O(\log n \log\log n / \log\log\log n)$ time. No $o(\sqrt{n})$ time bound is known for backward 2-pass. Stasko and Vitter [8] and Elmasry [2] proposed several variants of these linking strategies. In [8] it was observed that with a small insertion buffer the amortized cost of insertion can be reduced to $O(1)$. Iacono [7] proved that even the standard 2-pass version supports insert and meld in $O(1)$ time. Bringing the insertion cost to $O(1)$ is significant in applications where the heap is likely to end in a non-empty state, for instance, when computing single-source, single-destination shortest paths with Dijkstra's algorithm. Iacono [7] and Elmasry [2] studied the distribution-sensitive properties of pairing heaps. Before the pairing heap was introduced Sleator and Tarjan [12] invented the skew-heap, a self-adjusting data structure that supported insert, deletemin, and meld in logarithmic time; however, it did not support decreasekeys directly.

[1] Arbitrary coupling partitions the roots into couplets arbitrarily and accumulates the couplet winners in no specific order. Multipass repeatedly executes the coupling pass of 2-pass and backward 2-pass accumulates the $y_1, \ldots, y_\ell$ roots from left-to-right rather than right-to-left.

Overview. In Section 2 we discuss the original analysis of pairing heaps in order to provide a reference point for our approach. In Section 3 we introduce some key ideas and, as a warmup, sketch a proof that decreasekeys take $O(\sqrt{\log n})$ time. The full proof of Theorem 1 appears in Section 4. In Section 5 we discuss some open problems and make a conjecture on the actual complexity of pairing heaps.

2 Pairing Heaps, Entropy, and Potentials

The original analysis of pairing heaps [5] was a stroke of elegance. However, it was achieved by drawing a slightly unnatural correspondence between pairing heaps and splay trees. In particular, Fredman et al. [5] map a general rooted tree to a binary tree: a node's left child in the binary representation corresponds to its leftmost child in the general tree and its right child corresponds to its right sibling in the general tree. They observe that in the binary representation a deletemin operation looks very similar to a splay up the right shoulder of the tree. By borrowing the same potential function and analysis from [11] they derive amortized logarithmic bounds on deletemins and all other operations.

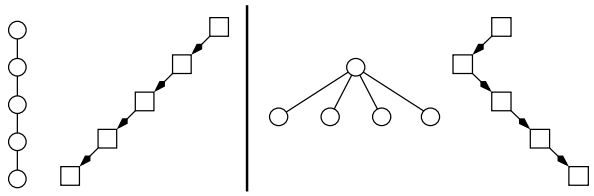

Figure 1. The general trees are marked with circular nodes, the binary representation with square nodes. On the left is a tree with entropy zero and on the right is a tree with entropy $\log(4!)$.

The analysis of [5] uses the potential function $\Phi = \sum_x \log s(x)$ where $s(x)$ is the size of the subtree rooted at x in the *binary* representation. One property of this function is that general trees with completely different characteristics can get assigned the same potential. In Figure 1 we have, among rooted 5-node trees, the two with minimum and maximum entropy, respectively. Here entropy is the logarithm of the number of linear extentions consistent with the heap-order. Since both trees get assigned a potential of $\log(5!)$ one wonders if there is an interpretation of this potential that makes sense in the context of a priority queue.

Let us examine the effect of one comparison on Φ. Suppose that a tree rooted at z is linked as the leftmost child of y, with y and z having A and B nodes in their trees, respectively. See Figure 2.

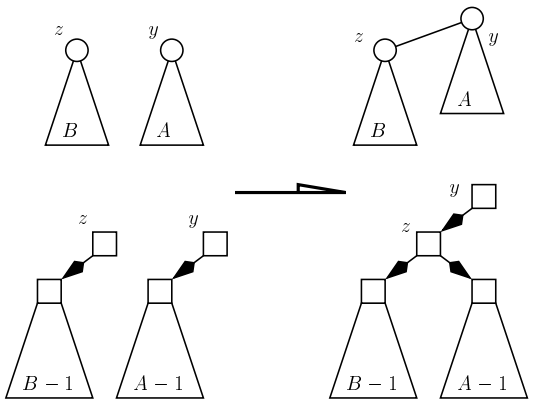

Figure 2. The effect of one linking. Circular nodes correspond to the general representation and square ones to the binary representation.

A quick calculation shows that regardless of the outcome of the comparison the increase in potential Φ is $\log\left(\frac{(A+B)(A+B-1)}{AB}\right)$. Ignoring the pesky "$-1$", this increase can be decomposed into $h(z) + \hbar(z)$:

$$h(z) = \log\left(\tfrac{A+B}{A}\right) \qquad \hbar(z) = \log\left(\tfrac{A+B}{B}\right)$$

where $h(z)$ represents the entropy reduction (bits of information) of the link (z, y) and $\hbar(z)$ is, for lack of a better term, the anti-information of link (z, y). That is, anti-information measures the hypothetical entropy reduction if the comparison between y and z had the opposite outcome. The roles of information and anti-information are not symmetric. As we will see in Sections 3 and 4, anti-information is a critical measure. Although not stated explicitly, the role of information in the original analysis [5] is to keep the proofs simple: with a symmetric potential function one can afford to be oblivious to the outcomes of individual comparisons. It is possible to reprove the results of [5] using the alternative potential $\Phi' = \sum_x \hbar(x)$. Lemma 2 is an interesting exercise.

Lemma 2 *An insert, decreasekey, or meld increases Φ' by $O(\log n)$. If the root has k children then performing a deletemin reduces Φ' by $k - O(\log n)$.*

There is an intuitive argument for why Φ' is a sensible potential function. Because the pairing heap behaves in a somewhat unintelligent fashion a deletemin operation can fail to reduce entropy by more than a negligible amount. In order to bound the amortized complexity of such an operation we must show that it nevertheless made *some* kind of measurable progress. Consider a singleton node x with no children. If, before the deletemin, its parent's subtree had size p and afterward it has size $p/2$, the difference in $h(x)$ (information) is miniscule for p large. On the other hand, if this trend continues then x will only have to endure another $O(\log p)$ inefficient linkings before an efficient one. In short, by the measure of anti-information, x has made one "anti-bit" of progress.

The intuition above can be recast in several ways. One way, advocated by Fredman [4] in his lower bound proof, is to view a pairing heap as *searching* for the *rank* of every node, where rank is the logarithm of the size of its subtree. An *efficient* link is one between nodes with ranks differing by at most a constant. (A high-rank node becoming the child of a low-rank node is also efficient but this should happen infrequently.) Although ranks change over time it is useful to think of them as essentially fixed. The kernel of Fredman's lower bound is the observation that after a decreasekey on x, nothing is "known" about the rank of x except that it lies in the range $[0, \log n]$. The pairing heap, incapable of revealing more than one bit of information about a node's rank in every comparison, must engage x in at least $\Omega(\log\log n)$ comparisons on the average before its first efficient linking. That is, no linking policy can do better than a binary search. On the upper bound side, the analysis of [5] says that pairing heaps do at least as well as a linear search. (The term *search* here is only meant to evoke a familiar concept; pairing heaps do not search for anything.) In this language, Section 3 gives a proof that pairing heaps do at least as well as a two-stage linear search. This leads to a $O(\sqrt{\log n})$ upper bound on decreasekeys. In Section 4 we generalize this idea to a kind of multi-stage search.

3 Introduction to the Analysis

At any moment the links of a pairing heap can be partitioned into three categories: those destined to be cut by a decreasekey operation, those destined to be cut by a deletemin, and those not cut at all. We call the first type *illusory* and the rest normal. The size of the subtree rooted at x is an important statistic, but an unstable one since nodes that lie below an illusory edge will, at some point in the future, be moved out of x's subtree. Let $|x|$ be the size of the normal subtree rooted at x. Clearly, until x is deleted $|x|$ is non-decreasing over time. Let $p(x)$ be the parent of x and $|p(x)|$ be w.r.t. the moment just before establishing the link $(x, p(x))$. We redefine $\hbar(x)$ w.r.t. the $||$ measure:

$$\hbar(x) = \log\left(\frac{|x| + |p(x)|}{|x|}\right)$$

and if x is a root, $\hbar(x) = 0$ by definition.

Recall that the deletemin operation consists of two passes: a *coupling* pass and an *accumulation* pass. Two nodes form a *couplet* if, in their next linking, they are destined to be linked in the coupling pass of a deletemin. Every node is either *open* or *closed*. A closed-couplet is one in which both nodes are closed. The overall potential of the heap is:

$$\begin{aligned}\Phi'' &= -(\#\text{ of closed-couplets}) \\ &\quad + \sum_x \begin{cases} \hbar(x) & \text{if } x \text{ is closed} \\ \frac{\hbar(x)}{\sqrt{\log n}} + \sqrt{\log n} & \text{if } x \text{ is open} \end{cases}\end{aligned}$$

As a rule, every node begins life in an open state. Since $\hbar(x) \leq \log n$ the potential contributed by x is no more than $2\sqrt{\log n}$. If x is open and $\hbar(x)$ drops below $\sqrt{\log n}$ we *close* it, which, one sees by inspection, can only cause a reduction in potential.

Decreasekey. Let x be the node in question. The link $(x, p(x))$ must be cut, and is by definition illusory. Since the nodes in x's subtree do not contribute to the size (by the $||$ measure) of ancestors above $p(x)$, cutting this link does not affect the $\hbar$ function or the overall potential of the tree. We link x with the root; let x' be the *loser* of this comparison. If $\hbar(x') \leq \sqrt{\log n}$ we designate x' closed, increasing the potential by $\hbar(x')$. Otherwise x' is open. Since $\hbar(x')$ is bounded by $\log n$ the increase in potential is at most $2\sqrt{\log n}$.

Insert, Meld. See decreasekey.

Deletemin. Our analysis focuses on one couplet x_i, x_{i+1} (two children of the root). Let x' be the tree derived by coupling and accumulating all the right siblings of x_{i+1}. Let $A = |p(x_{i+1})|, B = |x_{i+1}|, C = |x_i|$, and $A' = |x'|$. Due to the introduction of illusory links A' may be smaller than $A - 1$. As far as this couplet is concerned the deletemin performs two linkings: x_i with x_{i+1} and the winner with x'. There are numerous cases to analyze, depending on the outcomes of the comparisons, and the open/closed states of x_i, x_{i+1}, and x'. The cases boil down to three scenarios:

- If x_i, x_{i+1} are both closed then this leads to fruitful comparisons. (Fruitful = reduction in potential.)
- If x_i, x_{i+1} are both open this *also* leads to fruitful comparisons.
- If x_i is closed and x_{i+1} open (or the opposite) then this couplet may not produce fruitful comparisons. However, the number of these unfruitful couplets is dominated by the fruitful ones *plus* the cost of all deletemins.

We illustrate just two sub-cases of this analysis, in order to exhibit the important features of our approach. An exhaustive case analysis is pointless since these results are superceded by those in Section 4.

Case 1. Suppose that x_i, x_{i+1} form a closed couplet, and that x_{i+1} loses to x_i, which loses to x'. We charge $2\log((A+B+C)/A)$ units of potential to the deletemin operation, which we account for later. Assuming that neither of the new edges is illusory, the reduction in potential is:

$$\begin{aligned}&\log\tfrac{A+B}{B} + \log\tfrac{A+B+C}{C} - \log\tfrac{B+C}{B} \\ &- \log\tfrac{A'+B+C}{B+C} + 2\log\tfrac{A+B+C}{A} - 1 \\ \geq\ &\log\left(\tfrac{(A+B)(A+B+C)^2}{2CA^2}\right) > \log\left(\tfrac{(A+C)^2}{2AC}\right) \geq 1\end{aligned}$$

where the "-1" in the second line accounts for the loss of a closed couplet. The potential reduction is used to pay for $O(1)$ comparisons, including those involving x_i, x_{i+1} but possibly some others.

Case 2. Suppose that x_i, x_{i+1} are both open, and again, that x_{i+1} loses to x_i, which loses to x'. We borrow nothing from the deletemin operation; the reduction in potential is:

$$\begin{aligned}&\frac{1}{\sqrt{\log n}}\left(\log\tfrac{A+B}{B} + \log\tfrac{A+B+C}{C} - \log\tfrac{B+C}{B}\right. \\ &\left. - \log\tfrac{A'+B+C}{B+C}\right) \\ \geq\ &\left(\log\tfrac{A+B}{C}\right)/\sqrt{\log n} \\ \geq\ &1\end{aligned}$$

Where the last line follows from $\log C \leq \log(A+B) - \sqrt{\log n}$; if this inequality did not hold then x_i would have already been closed. This one unit is used to pay for $O(1)$ comparisons involving this couplet and possibly some others. After the links are established one or both of x_i, x_{i+1} might be closed, which can only reduce the potential further.

The cases involving a mixed couplet (with one open and one closed node) are paid for by (a) the creation of new closed-couplets, or (b) letting a non-mixed couplet elsewhere pay for the comparisons of the couplet in question. The way the closed-couplet potential works is fairly straightforward. Suppose z_1, y_1 and z_2, y_2 are adjacent mixed couplets, with y_1, y_2 open. If y_i defeats z_i then a lot of potential is released, which can pay for the comparisons in both couplets. On the other hand, if z_1 and z_2 emerge as winners (and both lose to the tree derived by accumulating y_2's right siblings) then z_1, z_2 can form a closed-couplet,

reducing the overall potential by at least 1. This is just a sketch of the arguments to come in Section 4.

In Case 1, and in many cases in Section 4, we "borrow" potential from deletemin operations. Lemma 3 is used to show that at most $O(\log n)$ units of potential are borrowed, implying an amortized cost of $O(\log n)$ per deletemin. This is the same argument used in [5], though phrased differently.

Lemma 3 *If every couplet x_i, x_{i+1} charges the deletemin operation at most*

$$c \log \left(\frac{|x_i| + |x_{i+1}| + |p(x_{i+1}|)}{|p(x_{i+1})|} \right)$$

units, the total amount charged is at most $c \log(n-1)$.

Proof: Let $x_1, \ldots, x_k$ be the children of the root to be deleted. Notice that the following sum telescopes,

$$\sum_{i=1}^{\lfloor k/2 \rfloor} c \log \left(\frac{|x_{2i-1}| + |x_{2i}| + |p(x_{2i})|}{|p(x_{2i})|} \right)$$

totaling $c \log \sum_i x_i$ if k is even and $c \log((\sum_i x_i)/(x_k+1))$ otherwise. □

4 A Better Analysis

One suspects that our analysis from the previous section could be generalized to show that decreasekeys take $O(\log^{1/3} n)$ time. Rather than consider two orders (open and closed) we could define three orders, where the potential of a node x would be proportional to $\hbar(x)$ at order 0, $\hbar(x)/\log^{1/3} n + \log^{1/3} n$ at order 1, or $\hbar(x)/\log^{2/3} n + 2\log^{1/3} n$ at order 2. Whenever two nodes of the same order end up in the same couplet we are assured of a fruitful comparison. However, managing three orders is significantly more complicated because there are new ways in which a deletemin operation can fail to release any potential. The comparisons must be paid for by showing the configuration of order 0, 1, and 2 nodes after the deletemin is measurably "better" than before. This requires a potential function on configurations. A huge case analysis ensues.

In this section we attempt a generalized analysis. Following the idea sketched above we assign each node x an *order* $o(x) \in \{0, 1, \ldots, \kappa - 1\}$, where κ is a parameter to be fixed later. It is still useful to make an open/closed distinction; order-0 nodes are closed and all other orders are open. The parent of x is $p(x)$ and $\widehat{o}(x)$ refers to $o(p(x))$ *immediately before establishing the link* $(x, p(x))$. The overall potential function consists of five components: $\tau, \alpha, \theta, \rho$, and μ. The α, θ, and τ parts are generalized forms of those introduced in Section 3 and all have some intuitive meaning. It is more difficult to explain the ρ and μ potentials because they are tailored to the *operational* properties of the pairing heap rather than structural features of the tree. We define the five potentials then sketch their relationships and individual roles within the analysis.

$$\begin{aligned}
\theta(x) &= c_2 2^{o(x)} (\log n)^{1/\kappa} \\
\alpha(x) &= \begin{cases} \hbar(x) & \text{if } o(x) = 0 \\ \frac{\hbar(x) \cdot c_1 2^{o(x)}}{(\log n)^{o(x)/\kappa}} & \text{otherwise} \end{cases} \\
\rho(x) &= \begin{cases} 1 & \text{if } o(x) = \widehat{o}(x) \geq 1 \\ 2^{\widehat{o}(x)+1} & \text{if } o(x) < \widehat{o}(x) \\ 0 & \text{otherwise} \end{cases} \\
\mu(x) & \begin{cases} \in [0, 2^{\widehat{o}(x)} - 1] & \text{if } o(x) > \widehat{o}(x) \\ = 0 & \text{otherwise} \end{cases} \\
\tau &= \text{number of closed couplets}
\end{aligned}$$

$$\text{Total Pot.} = -\tau + \sum_x (\theta(x) + \alpha(x) + \rho(x) - \mu(x))$$

In a deletemin operation each node has the possibility of making progress in two ways, which correspond to α and θ potentials. Let $o(y), \widehat{o}(y), \hbar(y)$ and $o'(y), \widehat{o}'(y), \hbar'(y)$ be with respect to y before and after the deletemin operation. If, after the operation, $o(y) \geq 1$ and $\hbar'(y)$ is less than $(\log n)^{o(y)/\kappa}$, then y undergoes an *order reduction*. We set $o'(y) = \min\{i \geq 0 : \hbar'(y)/(\log n)^{i/\kappa} < (\log n)^{1/\kappa}\}$. This releases at least $(c_2/2)2^{o(y)}(\log n)^{1/\kappa}$ units of θ-potential, which, for c_2 sufficiently large compared to c_1, pays for some linkings and the increases in ρ, α, and $-\mu$ potentials. (More on this later.) If y fails to undergo an order reduction it can still make progress. If $o(y) > 0$ and $\hbar'(y) \leq \hbar(y) - (\log n)^{o(y)/\kappa}$ then at least $c_1 2^{o(y)}$ units of α-potential are released. This pays for some comparisons and possible increases in ρ and $-\mu$ potentials. The other three potentials (ρ, μ, τ) represent an accounting scheme to pay for fruitless comparisons. The most interesting of these is the μ-potential, which represents the bottleneck in our whole approach. Suppose $o(y) \geq \widehat{o}'(y) > \widehat{o}(y)$, that is, the order of y's parent has become strictly closer to y's own order. This is a kind of progress since, if the trend continues, y will be linked with a node of the same order after less than κ linkings. (As we will see, such a linking is always fruitful in the sense that it coincides with a release of α-potential or an order reduction.) However, in the time between two order reductions on y it is not necessarily the case that $\widehat{o}(y)$ is increasing, or even non-decreasing. That is, this type of progress is very easily destroyed. Using the μ potentials we can show that, on the average, y participates in less than

$2^{o(y)}$ unfruitful linkings before its next fruitful one. Consider how the μ potentials of y and z might behave in the situation depicted in Figure 3.

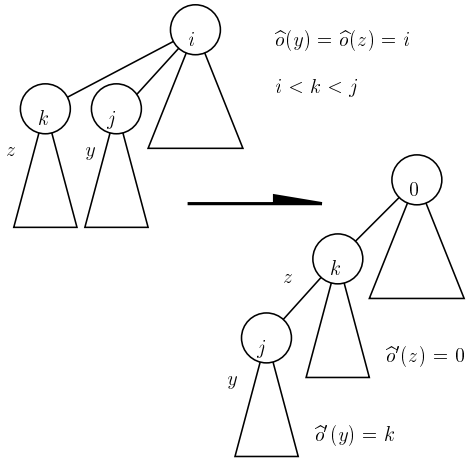

Figure 3. Nodes are labeled with their order.

One can see that y has made a kind of progress ($\widehat{o}(y) = i < k = \widehat{o}'(y)$) and that z has lost any progress made so far ($\widehat{o}(z) = i \geq 0 = \widehat{o}'(z)$). Since y made progress it only makes sense that *it* should be responsible for paying for the links (y, z) and $(z, p(z))$. Since no α, θ, ρ, or τ potential has been released y just borrows one unit of potential from the void. In the past both y and z may have borrowed potential to pay for these types of linkings. Suppose the debt of y and z are $\mu(y)$ and $\mu(z)$. Because y made progress we force it to pay for the two links *and* take on z's debt. That is, we set $\mu'(y) = \mu(y) + \mu(z) + 1$ and $\mu'(z) = 0$. One can prove by induction that $\mu(y) \leq 2^{\widehat{o}(y)} - 1$ at all times.[2] Since $o(y) < \kappa$ the number of links made but not paid for is $O(n2^{\kappa})$, where n is the size of the tree. These linkings can be attributed to earlier insert operations.

It may be the case that both y and z make no progress, in the sense that $\widehat{o}'(y) \leq \widehat{o}(y)$ and $\widehat{o}'(z) \leq \widehat{o}'(z)$. The τ and ρ potentials come into play in this situation.

In the remainder of this section we show that the amortized cost of deletemin is logarithmic while the cost of decreasekey, insert, and meld is on the order of $2^{\kappa}(\log n)^{1/\kappa}$. We set $\kappa = \sqrt{\log \log n} - O(1)$. Theorem 1 follows.

The proof focuses on one couplet z, y in a deletemin operation, where z is to the left of y. Let $o(z) = k$, $o(y) = j$ and assume wlog that $j \geq k$; the analysis for $j < k$ is symmetric. Let $x = p(y)$ be the root to be deleted. We assume that $\widehat{o}(y) = \widehat{o}(z) = i$; the case when $\widehat{o}(y) \neq \widehat{o}(z)$, i.e., x underwent an order reduction between linking with y and z, can be handled separately. Let x' be the tree root obtained by coupling and accumulating the right siblings of y and let $o(x') = \ell$. We assume z, y are *not* the rightmost children of x and ignore the special case when x' does not exist. Let $C = |z|$, $B = |y|$, $A = |x|$ (where $|x|$ is the size of x before linking with y and z) and $A' = |x'|$. Due to the introduction of illusory links we can only guarantee that $1 \leq A' < A$. Generally speaking, if g is an evolving function then g is w.r.t. the moment before the deletemin and g' w.r.t. a moment during or after the deletemin. The following inequalities hold:

$$\begin{array}{rcll} \log A' & \leq & \log A - (\log n)^{\ell/\kappa} & \text{if } \ell > 0 \\ \log B & \leq & \log A - (\log n)^{j/\kappa} & \text{if } j > 0 \\ \log C & \leq & \log(A + B) - (\log n)^{k/\kappa} & \text{if } k > 0 \end{array}$$

During the accumulation linking we assign x' (the accumulation root) an α-potential of:

$$\alpha(x') = 2\ell + f(\ell) \cdot \log\left(\frac{A}{A'}\right)$$

$$\text{where } f(\ell) = \begin{cases} 1 & \{\ell = 0\} \\ c_1 2^{\ell}/(\log n)^{\ell/\kappa} & \{0 < \ell < \kappa\} \end{cases}$$

Two properties of f to keep in mind are $f(\ell) < 1$ for $\ell > 0$ and $f(\ell + 1) < f(\ell)$, for n sufficiently large.

The situation is simple: we link z and y and the winner with x'. However, due to the number of parameters involved there are several cases to analyze. Factors include the outcomes of the comparisons, the orders of the nodes involved, whether the two new links become illusory or not, and whether any nodes take an order reduction. Our analysis also must take into account features of the couplets containing y and z *after* the deletemin operation. We make repeated reference to cases (a)–(d) depicted in Figure 4. (Most of the key ideas in the proof can be absorbed by considering only case (a) through Situations 1–4. The other cases are of lesser importance.)

Lemma 4 *After linking z and y and the winner with x', the adjusted reduction in α-potential (after borrowing potential from the current deletemin operation and future and past decreasekey operations) is at least 2 if $j = k = 0$ and at least $c_1 2^j$ if $j = k > 0$. In case (a) it is at least 0 and if $k \leq \ell$, $\ell > 0$ the reduction is at least $c_1 2^{\ell}$. In case (b) it is at least 2 and if $\ell \leq k$, $k > 0$ the reduction is at least $c_1 2^k$. In cases (c) and (d) the reduction is at least $c_1 2^j$.*

Proof: Regardless of the final winner among $\{y, z, x'\}$ we set its order to be $\min\{\ell, k, j\} = \min\{\ell, k\}$. After addressing cases (a)–(d) we consider the effect on our potential function when one or both of the new links become illusory.

[2] This induction hypothesis is actually part of the definition of μ. There is a small anomaly when $\widehat{o}(y) \neq \widehat{o}(z)$, that is, when $p(y)$ had undergone an order reduction between its linkings with y and z. This case is easy to handle separately.

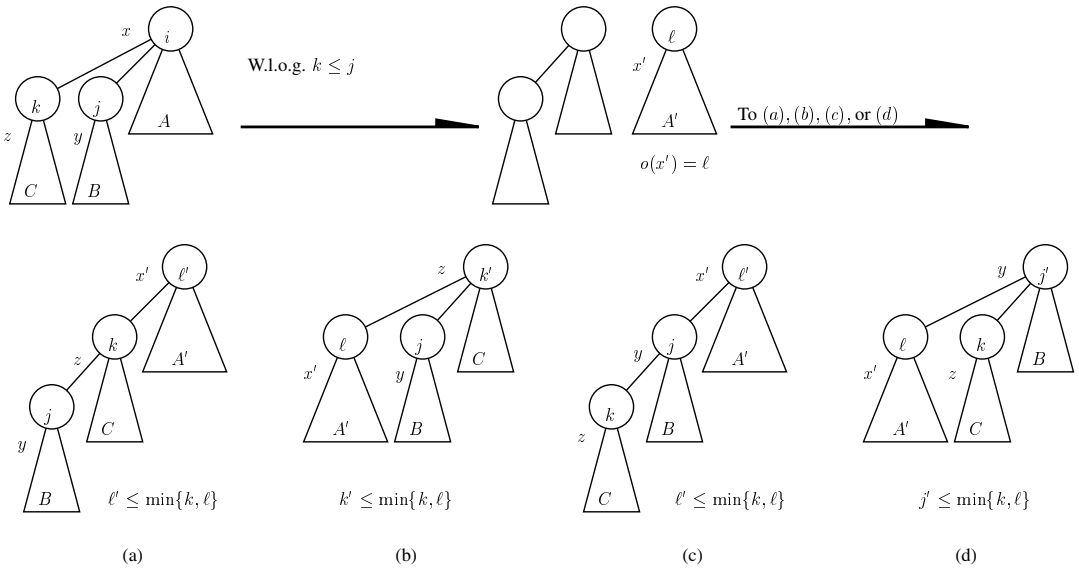

Figure 4. Top left: x is the root to be deleted and z, y is the couplet under investigation. We assume $\widehat{o}(z) = \widehat{o}(y) = i$. Top right: x' is the root of the tree derived from coupling and accumulating all of y's right siblings. If no illusory links were introduced A' (the size of x') would be $A - 1$; in general it may be smaller. Cases (a)–(d) depict the four possible outcomes of the two linkings. Regardless of which node becomes the new root (z, y, or x'), its order is reduced to $\min\{k, \ell\}$.

Case (a). Here y loses to z, which loses to x'. We set $\ell' = o(x') = \min\{k, \ell\}$. The adjusted reduction in α potential, after borrowing $2\log((A+B+C)/A)$ units from the deletemin operation, is the *logarithm of*:

$$\left(\frac{A+B}{B}\frac{B}{B+C}\right)^{f(j)}\left(\frac{A+B+C}{C}\frac{B+C}{A'+B+C}\right)^{f(k)}$$

$$\cdot\left(\frac{A}{A'}\right)^{f(\ell)}\left(\frac{A'+B+C}{A+B+C}\right)^{f(\ell')}\left(\frac{A+B+C}{A}\right)^{2}4^{\ell-\ell'}$$

$$\ge \left(\frac{A+B}{B+C}\right)^{f(j)}\left(\frac{B+C}{C}\right)^{f(k)}\left(\frac{A+B+C}{A}\right)^{2}$$

$$\begin{aligned}
\{j=k=0\} &> \frac{(A+C)^2}{AC} \ge 4\\
\{j=k>0\} &> \left(\frac{A+B}{C}\right)^{f(j)} \ge (2^{\log^{j/\kappa} n})^{f(j)}\\
&= (2^{\log^{j/\kappa} n})^{c_1 2^j/\log^{j/\kappa} n} = 2^{c_1 2^j}\\
\{k \le \ell\} &> \left(\frac{A}{A'}\right)^{f(\ell)} \ge (2^{\log^{\ell/\kappa}})^{f(\ell)} = 2^{c_1 2^\ell}\\
\{\text{otherwise}\} &> 1
\end{aligned}$$

Case (b). Here y and x' lose to z. We set $k' = o'(z) = \min\{k, \ell\}$. After borrowing $3\log((A+B+C)/A)$ units from deletemin, the reduction in α-potential is log of:

$$\left(\frac{A+B}{B+C}\right)^{f(j)}\left(\frac{A+B+C}{C}\right)^{f(k)}\left(\frac{A}{A'}\frac{A'}{A'+B+C}\right)^{f(\ell)}$$

$$\cdot\left(\frac{A'+B+C}{A+B+C}\right)^{f(k')}\left(\frac{A+B+C}{A}\right)^{3}4^{\ell-k'}$$

$$\begin{aligned}
\{j=k=0\} &\ge \frac{(A+B)(A'+B+C)(A+B+C)^2}{(B+C)CA^2}\\
&> \frac{(A+C)^2}{AC} \ge 4\\
\{j=k>0\} &\ge \left(\frac{A+B}{C}\right)^{f(j)} > \left(2^{\log^{j/\kappa} n}\right)^{f(j)}\\
&= 2^{c_1 2^j}\\
\{\ell \le k, k>0\} &\ge \left(\frac{A+B+C}{C}\right)^{f(k)} \ge 2^{c_1 2^k}\\
\{\ell=k=0\} &\ge \left(\frac{A+B}{B}\right) \ge 2^{c_1 2^j}\\
\{\text{otherwise}\} &> 4^{\ell-k'} \ge 4
\end{aligned}$$

Case (c). Here z loses to y, which loses to x'. Set $\ell' = o'(x') = \min\{k, \ell\}$. The reduction in α-potential, after borrowing $\log((A+B+C)/A)$ from the deletemin, is the log of:

$$\left(\frac{A+B}{B}\frac{B+C}{A'+B+C}\right)^{f(j)}\left(\frac{A+B+C}{C}\frac{C}{B+C}\right)^{f(k)}$$

$$\cdot\left(\frac{A}{A'}\right)^{f(\ell)}\left(\frac{A'+B+C}{A+B+C}\right)^{f(\ell')}\frac{A+B+C}{A}\,4^{\ell-\ell'}$$

$$> \left(\frac{A+B}{B}\right)^{f(j)}\frac{A+B+C}{A}$$

(Consider $\ell' = \ell$ and $\ell' = k$ separately)

$$\{j=k=0\} \geq \frac{(A+B)^2}{AB} \geq 4$$

$$\{\text{otherwise}\} \geq \left(\frac{A+B}{B}\right)^{f(j)} > \left(2^{\log^{j/\kappa} n}\right)^{f(j)} = 2^{c_1 2^j}$$

Case (d). In this situation y defeats both z and x'. Set $j' = o'(y) = \min\{k, \ell\}$. We borrow $2\log((A+B+C)/A)$ units from the deletemin operation, for a reduction in α-potential of:

$$\left(\frac{A+B}{B}\right)^{f(j)}\left(\frac{A+B+C}{B+C}\right)^{f(k)}\left(\frac{A}{A'}\frac{A'}{A'+B+C}\right)^{f(\ell)}$$

$$\cdot\left(\frac{A'+B+C}{A+B+C}\right)^{f(j')}\left(\frac{A+B+C}{A}\right)^{2}4^{\ell-j'}$$

$$\geq \left(\frac{A+B}{B}\frac{A+B+C}{B+C}\right)^{f(j)}\left(\frac{A}{A'+B+C}\right)^{f(\ell)}$$

$$\cdot\left(\frac{A'+B+C}{A+B+C}\right)^{f(j')}\left(\frac{A+B+C}{A}\right)^{2}$$

$$\{j=k=0\} > \frac{(A+B)(A+B+C)}{AB} > 4$$

$$\{\text{otherwise}\} > \left(\frac{A+B}{B}\right)^{f(j)} > 2^{c_1 2^j}$$

If either of the new links becomes illusory (in any case (a)–(d)) then we may need to increase the orders of z, y, or x'. The maximum increase in all potentials (including θ, ρ, μ, τ) is $O(2^\kappa(\log n)^{1/\kappa})$, which we charge to the future decreasekey operation that cuts the new illusory link.

If any of z, y, x' undergoes an order reduction, say y, then at least $(c_2/2)2^{o(y)}(\log n)^{1/\kappa}$ units of θ-potential are released. This pays for an increase in $\alpha(y)$ by at most $c_1 2^{o'(y)}(\log n)^{1/\kappa}$, where $o'(y) < o(y)$ is the new order, as well an increases in ρ and μ potentials. □

Lemma 4 gives us a bound on the reduction in α-potential. We analyze the effects on τ, ρ, μ, and θ-potential with six case distinctions, which depend only on the relative order of i, j, k.

Situation 1: $j = k = 0$**.** ***(Closed couplet)*** By Lemma 4 the reduction in α-potential is at least 2. We use 1 unit to pay for the increase in $-\tau$ (since this closed couplet is being destroyed) and 1 unit to pay for $O(1)$ comparisons; see Situation 2. The creation of new closed couplets can only reduce the potential further.

Situation 2: $j > k = i = 0$**.** ***(Mixed couplet)*** If y defeats z in the first comparison (that is, we are in case (c) or (d) of Lemma 4) then the reduction in α-potential at least $c_1 2^j$. We use one unit to pay for the comparisons, 2^{j+1} units for the increase in $\rho(z)$ and perhaps 2^{j+1} for an increase in $\rho(x')$ if we are in case (d). Now suppose that y loses the first comparison. The adjusted reduction in α potential is positive in all cases, though in case (a) perhaps too small to pay for any comparisons.

After z defeats y it next forms a couplet with some z'. If z' was the winner of its last coupling (as z is in this coupling) then there are two cases. If z' is open then, after inspecting Situations 3–6, one sees that at least 1 extra unit of potential was released in the last coupling of z', which we use to pay for the comparisons involving z. On the other hand, if z' is closed then z, z' is a closed couplet, implying a reduction of 1 unit of $-\tau$ potential, which pays for our comparisons.

The tricky case is if z' was the *loser* of its last coupling. Let us stand back a bit and consider the ways in which a winner/loser coupling like z, z' could be created. Let q be a child of the root just before a deletemin and and let $q_0, q_1, \ldots, q_w$ be the *new* children of q after the deletemin operation, in *right to left* order. Clearly q_0 was coupled with q before the deletemin and is therefore a loser. The edge (q_1, q) was created in Case (b) or (d), where the winner of the couplet, q, defeated the accumulator root, q_1 in this case. (This type of edge is a *spine edge*. Spine edges are marked with dashed lines in Figure 5.) The edges $(q_2, q), \ldots, (q_w, q)$ were all created in Case (a) or (c), where the winner of the couplet, $q_2, \ldots, q_w$ in this case, was defeated by the accumulator root, q in this case. Thus the *only* loser in this new batch of children is q_0. Among $q_1, \ldots, q_w$ at most 2 can be next coupled with a loser: one can be coupled with q_0 and another can be coupled with a *future* child of q. See Figure 5.

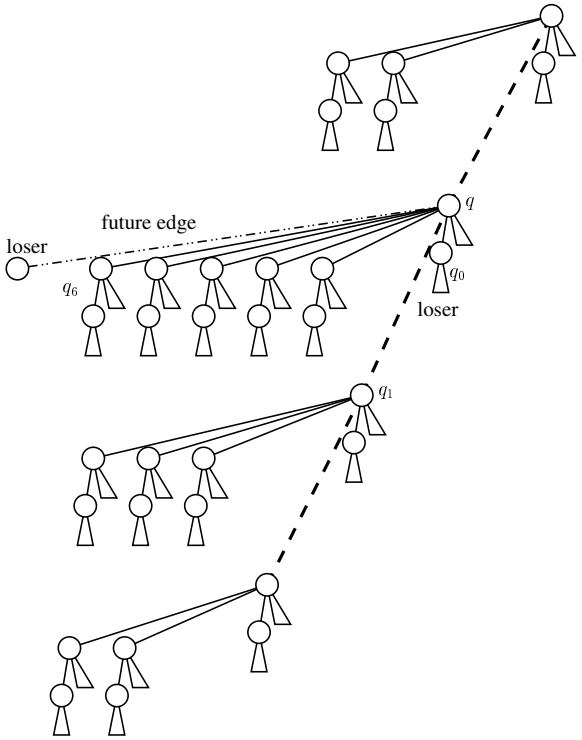

Figure 5. Dashed links form the *spine*; they always come from the second linking in cases (b) and (d). A *loser* node is one that, in the last coupling pass in which it participated, lost to its couplet partner.

Since couplet q, q_0 fell in either case (b) or (d), Lemma 4 states that at least 2 units of α potential were released. We use one unit to pay for the destruction of a closed couplet (if $o(q) = o(q_0) = 0$; see Situation 1) and 1 unit to pay for $O(1)$ comparisons including those of the 2 children of q that are winners but are unlucky enough to be coupled with losers.

Situation 3: $j = k > 0$**.** ***(Open couplet; same order)*** By Lemma 4 the reduction in α-potential is at least $c_1 2^j$, which we use to pay for increases in $-\mu$ and ρ potential. We set $\mu'(y) = \mu'(z) = 0$, always a valid assignment to μ, increasing the overall $-\mu$ potential by at most $2 \cdot (2^{j-1} - 1)$. Any of $\rho(z), \rho(y)$, or $\rho(x')$ may increase by 1. If any of y, z, x' undergoes an order reduction, the drop in θ-potential pays for the minor increases in α, ρ, and $-\mu$ potentials.

Situation 4: $j > k > i$**.** ***(Open couplet; different orders)*** If we are in case (c) or (d) then at least $c_1 2^j$ units of α-potential are released, which pays for resetting $\mu'(y) = \mu'(z) = 0$ and the increase in $\rho(z)$ by 2^{j+1} and $\rho(y), \rho(x')$ by at most 1. (Subsequent order reductions pay for increases in $-\mu, \rho$ with large drops in θ-potential.)

If we are in cases (a) or (b) then $\widehat{o}'(y) = k > i = \widehat{o}(y)$, that is, from y's point of view this comparison made a little progress since its parent's order is now closer to its own. We set $\mu'(y) = \mu(y) + \mu(x) + 1$ and $\mu'(z) = 0$. The overall reduction in $-\mu$ potential is 1. Since $\mu(y), \mu(z) \leq 2^i - 1$ we have $\mu'(y) \leq 2(2^i - 1) + 1 = 2^{i+1} - 1 \leq 2^k - 1 = 2^{\widehat{o}'(y)} - 1$. In other words, μ' is still a valid assignment. If we are in case (a) and $k \leq \ell$ then by Lemma 4, $c_1 2^\ell$ units are released, which pays for the increase in $\rho(z)$. If we are in case (b) and $\ell \leq k$ then $c_1 2^k$ units are released, which pays for increases at $\rho(x')$. Again, subsequent order reductions on y, z, x' pay for increases in $-\mu, \rho$ potentials with large drops in θ-potential. To sum up, the overall reduction is at least 1, which pays for $O(1)$ linkings: those involving y and z and possibly some others described in Situation 2.

Situation 5: $j > k = i > 0$**.** ***(Open couplet; different orders)*** Notice that $\rho(z) = 1$. We will use this unit to pay for the comparisons and show that any further fluctuation in potentials does no harm. If we are in case (a) or (b) then $\widehat{o}(y) = \widehat{o}'(y) = i$ is unchanged, meaning $\mu'(y) = \mu(y)$ is still a valid assignment. If we are in case (a) and $k \leq \ell$ then the $c_1 2^\ell$ reduction in α-potential can pay for any increase in $\rho(z)$. Similarly, reductions in α potential pay for increases in $\rho(x')$ in case (b) when $\ell \leq k$. Cases (c) and (d) are easily handled.

Situation 6: $j, i > k \geq 0$**.** ***(Open/mixed couplet)*** Up until now we have been worrying about sudden increases in ρ-potential. It is in this situation that ρ finally comes into play. Suppose that we are in case (a), that y lost to z which lost to x'. Assuming $\ell < k$, *neither* of these comparisons were fruitful in the sense that no α potential was released. Furthermore, neither made progress in the sense of Situation 4 since $\widehat{o}'(y) = k < i = \widehat{o}(y)$ and $\widehat{o}'(z) = \ell < i = \widehat{o}(z)$; in fact, we now need to *reduce* $\mu(y)$ to restore validity. We pay for increases in $-\mu$ with the reduction in $\rho(z)$, from $2^{\widehat{o}(z)+1} = 2^{i+1}$ to zero. (If $\rho(z)$ is not reduced to zero then $k \leq \ell$ and $\ell > 0$; by Lemma 4, at least $c_1 2^\ell$ units of α-potential are released. This is a good case.) Since $\mu(z) = 0$ and $\mu(y) \leq 2^i - 1$, setting $\mu'(y) = 0$ leaves us with $2^i + 1$ units leftover to pay for $O(1)$ comparisons. Cases (b), (c), and (d) are handled similarly. Again, increases in $\rho, -\mu$ due to order reductions on y, z, x' are paid for with reduction in θ-potential.

We have established that the cost of deletemin is $O(\log n)$. Analyzing inserts, melds, and decreasekeys is trivial. In these three operations there is exactly one link-

ing; we assign the *loser* x of this linking the minimum order o such that $\hbar(x)/(\log n)^{o/\kappa} < (\log n)^{1/\kappa}$. The increase in $\theta, \alpha, \rho, -\mu, -\tau$ potential is $O(2^{\kappa}(\log n)^{1/\kappa})$.

5 Discussion and Conclusion

We have given the first sub-logarithmic bound on the complexity of the pairing heap's decreasekey operation. We conjecture that our bound is not tight, and that Fredman's lower bound of $\Omega(\log\log n)$ is tight for sparse instances. For dense instances, where $m = n^{1+\Omega(1)}$, Fredman [4] claimed that pairing heaps run in linear time.

Conjecture 5 *The pairing heap executes any sequence of m operations, including n deletemins, in $O(m\log\log_{m/n} n + n\log n)$ time. In particular, the pairing heap is asymptotically optimal for $m = O(n\log n/\log\log n)$ and $m = n^{1+\Omega(1)}$.*

The primary open question in the area of binary search trees is whether the splay tree [11] (or any structure) is *dynamically optimal*; see [1] for some recent progress on this question. Despite the connection between splay trees and pairing heaps, there is no obvious analogue of dynamic optimality for priority queues. However, the corollaries of dynamic optimality do have natural analogues. For instance, if the pairing heap possessed versions of the working set and dynamic finger properties, this would lead to some very simple adaptive sorting algorithms. In [7] Iacono proved that pairing heaps satisfy a weaker form of the working set property.

Acknowledgment. I would like to thank Amr Elmasry and Irit Katriel for several valuable discussions.

References

[1] E. Demaine, D. Harmon, J. Iacono, and M. Pătraşcu. Dynamic optimality — almost. In *Proceedings 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 484–490, 2004.

[2] A. Elmasry. Adaptive properties of pairing heaps. Technical Report 2001-29, DIMACS, September 2001.

[3] A. Elmasry. Parameterized self-adjusting heaps. *J. Algor.*, 52(2):103–119, 2004.

[4] M. L. Fredman. On the efficiency of pairing heaps and related data structures. *J. ACM*, 46(4):473–501, 1999.

[5] M. L. Fredman, R. Sedgewick, D. D. Sleator, and R. E. Tarjan. The pairing heap: a new form of self-adjusting heap. *Algorithmica*, 1(1):111–129, 1986.

[6] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987.

[7] J. Iacono. Improved upper bounds for pairing heaps. In *Scandinavian Workshop on Algorithm Theory (SWAT, LNCS 1851*, pages 32–43, 2000.

[8] J. S. Vitter J. T. Stasko. Pairing heaps: Experiments and analysis. *Comm. ACM*, 30(3):234–249, 1987.

[9] K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 2000.

[10] B. M. E. Moret and H. D. Shapiro. An empirical assessment of algorithms for constructing a minimum spanning tree. In *DIMACS Series on Discrete Math. and Theoretical Computer Science*, 1994.

[11] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *J. ACM*, 32(3):652–686, 1985.

[12] D. D. Sleator and R. E. Tarjan. Self-adjusting heaps. *SIAM J. Comput.*, 15(1):52–69, 1986.