Information Processing Letters 31 (1989) 3–5
North-Holland

12 April 1989

# A TIGHT AMORTIZED BOUND FOR PATH REVERSAL

David GINAT

*Department of Computer Science, University of Maryland, College Park, MD 20742, U.S.A*

Daniel D. SLEATOR *

*Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213, U.S.A.*

Robert E. TARJAN **

*Department of Computer Science, Princeton University, Princeton, NJ 08544, U.S.A. and AT&T Bell Laboratories, Murray Hill, NJ 07974, U.S.A.*

Communicated by T. Lengauer
Received 28 July 1988
Revised 5 December 1988

Path reversal is a form of path compression used in a disjoint set union algorithm and a mutual exclusion algorithm. We derive a tight upper bound on the amortized cost of path reversal.

*Keywords*: Analysis of algorithms, path compression, disjoint set union, data structures, amortized efficiency

Let $T$ be a rooted tree. A *path reversal* at a node $x$ in $T$ is performed by traversing the path from $x$ to the tree root $r$ and making $x$ the parent of each node on the path other than $x$. Thus $x$ becomes the new tree root. (See Fig. 1). The *cost* of the reversal is the number of edges on the path reversed. Path reversal is a variant of the standard path compression algorithm for maintaining disjoint sets under union [5]. It has also been used in a novel mutual exclusion algorithm [2,6].

Suppose that a sequence of $m$ reversals is performed on an arbitrary initial $n$-node tree. What is the total cost of the sequence? Let $T(n, m)$ be the worst-case cost of such a sequence, and let $A(n, m) = T(n, m)/m$. We are most interested in the value of $A(n, m)$ for fixed $n$ as $m$ grows. As discussed by Tarjan and Van Leeuwen [5], binomial trees provide a class of examples showing that $A(n, m) \geqslant \lfloor \lim n \rfloor$ [1], and their rather com-

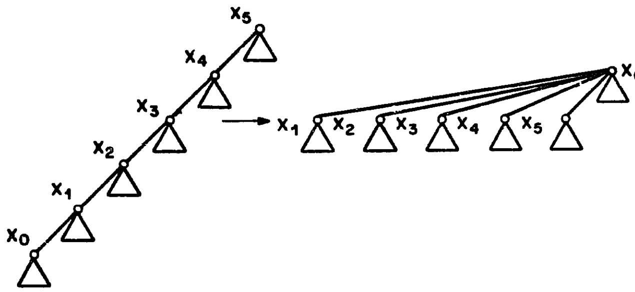


Fig. 1. Path reversal (triangles denote subtrees).

* Research partially supported by DARPA, ARPA order 4976, amendment 19, monitored by the Air Force Aeronautics Laboratory under Contract No. F33615-87-C-1499, by the National Science Foundation under Grant No. CCR 8658139, and by AT&T Bell Laboratories.

** Research partially supported by NSF Grant No. DCR-8605962 and ONR Contract No. N00014-87-K-0467.

[1] All logarithms in this paper are base 2.

0020-0190/89/$3.50 

plicated and their rather complicated analysis gives an upper bound of

$$A(n, m) = O\left(\log n + \frac{n \log n}{m}\right).$$

Ginat and Shankar [2] prove that

$$A(n, m) \leqslant 2 \log n + \frac{n \log n}{m}.$$

We shall prove that

$$A(n, m) \leqslant \log n + \frac{n \log n}{2m}.$$

In the special case that the initial tree consists of a root with $n-1$ children, which is the case in the mutual exclusion algorithm, the bound is

$$A(n, m) \leqslant \log n.$$

To obtain the bound, we apply the *potential function* method of amortized analysis (see [4]). Let the *size* $s(x)$ of a node $x$ in $T$ be the number of descendants of $x$, including $x$ itself. Let the *potential* of $T$ be

$$\Phi(T) = \frac{1}{2} \sum_{x \in T} \log s(x).$$

Define the *amortized cost* of a path reversal over a path of $k$ edges to be $k - \Phi(T) + \Phi(T')$, where $T$ and $T'$ are the trees before and after the reversal, respectively. For any sequence of $m$ reversals, we have

$$\sum_{i=1}^{m} a_i = \sum_{i=1}^{m} (t_i - \Phi_{i-1} + \Phi_i) = \sum_{i=1}^{m} t_i - \Phi_0 + \Phi_m,$$

where $a_i$, $t_i$, and $\Phi_i$ are the amortized cost of the $i$th reversal, the actual cost of the $i$th reversal, and the potential after the $i$th reversal respectively, and $\Phi_0$ is the potential of the initial tree. Since $\Phi_0 \leqslant \frac{1}{2} n \log n$ and $\Phi_m \geqslant \frac{1}{2} \log n$, this inequality yields

$$\sum_{i=1}^{m} t_i \leqslant \sum_{i=1}^{m} a_i + \tfrac{1}{2}(n-1) \log n,$$

which in turn implies

$$A(n, m) \leqslant \frac{1}{m} \sum_{i=1}^{m} a_i + \frac{n \log n}{2m}.$$

We shall prove that the amortized cost of any reversal is at most $\log n$, thereby showing that

$$A(n, m) \leqslant \log n + \frac{n \log n}{2m}.$$

When the initial tree consists of a root with $n-1$ children, the bound drops to $A(n, m) \leqslant \log n$, since then $\Phi_0 \leqslant \Phi_m$, and the extra additive term drops out.

Let $x_0, x_1, x_2, \ldots, x_k$ be a path that is reversed, and let $A$ be the amortized cost of the reversal. For $0 \leqslant i \leqslant k$, let $s_i$ be the size of $x_i$ before the reversal. The size of $x_0$ after the reversal is $s_k$ and the size of $x_i$ after the reversal, for $1 \leqslant i \leqslant k$, is $s_i - s_{i-1}$. We can thus write $A$ as

$$\begin{aligned} A &= k - \sum_{i=0}^{k} \tfrac{1}{2} \log s_i + \tfrac{1}{2} \log s_k \\ &\quad + \sum_{i=1}^{k} \tfrac{1}{2} \log(s_i - s_{i-1}) \\ &= k + \frac{1}{2} \sum_{i=0}^{k-1} (\log(s_{i+1} - s_i) - \log s_i) \\ &= k + \frac{1}{2} \sum_{i=0}^{k-1} \log((s_{i+1} - s_i)/s_i). \end{aligned}$$

For $0 \leqslant i \leqslant k-1$, let $\alpha_i = s_{i+1}/s_i$. Note that $(s_{i+1} - s_i)/s_i = \alpha_i - 1$. We have

$$\begin{aligned} A &= k + \tfrac{1}{2} \sum_{i=0}^{k-1} \log(\alpha_i - 1) \\ &= \sum_{i=0}^{k-1} (1 + \tfrac{1}{2}\log(\alpha_i - 1)). \end{aligned}$$

We now make use of the following inequality, which will be verified below: for all $\alpha > 1$, $1 + \frac{1}{2} \log(\alpha - 1) \leqslant \log \alpha$. From this inequality we obtain

$$\begin{aligned} A &\leqslant \sum_{i=0}^{k-1} \log \alpha_i \\ &= \sum_{i=0}^{k-1} \log(s_{i+1}/s_i) = \sum_{i=0}^{k-1} (\log s_{i+1} - \log s_i) \\ &= \log s_k - \log s_0 \\ &\leqslant \log n, \end{aligned}$$

since $s_k = n$ and $s_0 \geqslant 1$.

This completes the amortized analysis. We verify the needed inequality by the following chain of reasoning:

$$0 \leqslant (\alpha-2)^2$$
$$\Rightarrow 0 \leqslant \alpha^2 - 4\alpha + 4$$
$$\Rightarrow 4(\alpha-1) \leqslant \alpha^2$$
$$\Rightarrow \log(4(\alpha-1)) \leqslant \log(\alpha^2)$$
$$\Rightarrow 2 + \log(\alpha-1) \leqslant 2\log\alpha$$
$$\Rightarrow 1 + \tfrac{1}{2}\log(\alpha-1) \leqslant \log\alpha.$$

We conclude some remarks. The definition of the potential function used here has been borrowed from Sleator and Tarjan's analysis of splay trees [3]; it has also been used to analyze pairing heaps [1]. As in the case of splay trees, the upper bound can be generalized in the following way. Assign to each tree node $x$ a fixed but arbitrary positive weight $w(x)$. Define the *total weight* of $x$, $tw(x)$, to be the sum of the weights of all descendants of $x$, including $x$ itself. Define the potential of the tree $T$ to be

$$\Phi(T) = \frac{1}{2}\sum_{x \in T} \log tw(x).$$

A straightforward extension of the above analysis shows that the total cost of a sequence of $m$ reversals is at most

$$\sum_{i=1}^{m} \log(W/w_i) + \Phi_0 - \Phi_m,$$

where $w_i$ is the weight of the node $x_i$ at which the $i$th reversal starts and $W$ is the sum of all the node weights.

Choosing $w(x) = 1$ for all $x \in T$ gives our original result. Choosing $w(x) = f(x) + 1$, where $f(x)$ is the number of times a reversal begins at $x$, gives an upper bound for the total time of all reversals of

$$\sum_{i=1}^{m} \log\left(\frac{n+m}{f(x_i)}\right) + \frac{1}{2}\sum_{x \in T} \log\left(\frac{n+m}{f(x)}\right).$$

It is striking that the "sum of logarithms" potential function serves to analyze three different data structures. We are at a loss to explain this phenomenon; whereas there is a clear connection between splay trees and pairing heaps (see [1]), no such connection between trees with path reversal and the other two data structures is apparent. In the case of path reversal, the sum of logarithms potential function gives a bound that is exact to within an additive term depending only on the initial and final trees. It would be extremely interesting and useful to have a systematic method for deriving appropriate potential functions. The three examples of splaying, pairing, and reversal offer a setting in which to search for such a method.

## Acknowledgment

The first author thanks D. Mount and A.U. Shankar for valuable discussions and useful comments.

## References

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

[2] D. Ginat and A. Udaya Shankar, Correctness proof and amortization analysis of a log $N$ distributed mutual exclusion algorithm, Tech. Rept. CS-TR-2038, Department of Computer Science, University of Maryland, 1988.

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

[4] R.E. Tarjan, Amortized computational complexity, *SIAM J. Alg. Disc. Meth.* **6** (1985) 306–318.

[5] R.E. Tarjan and J. van Leeuwen, Worst-case analysis of set union algorithms, *J. Assoc. Comput. Mach.* **31** (1984) 245–281.

[6] M. Trehel and M. Naimi, A distributed algorithm for mutual exclusion based on data structures and fault tolerance, In: *Proc. Sixth Ann. Internat. Phoenix Conf. on Computers and Communication*, Scottsdale, AZ (February 1987) 35–39.