Chris Osborn
CS 573

1 Mergeable trees problem

Does there exist a data structure that maintains a heap-ordered forest while supporting the following operations:

- *parent*(v): Return v's parent, or *null* if v is a root.
- *nca*(v, w): Return the nearest common ancestor of v and w, or *null* if v and w are in different trees.
- *make*(v, x): Create a new, one-node tree consisting of node v with label x.
- *link*(v, w): Given a root v and a node w such that $label(v) \geq label(w)$ and w is not a descendent of v, combine the trees containing v and w by making w the parent of v.
- *delete*(v): Delete leaf v from the forest.
- *merge*(v, w): Where P is the path from v to the root, and Q is the path from w to the root, restructure the tree or trees containing v and w by merging P and Q in a way that preserves heap order.
- *cut*(v): Given non-root node v, make it a root by deleting the edge connecting v to its parent

Furthermore, this data structure must support each operation in $O(\log n)$ amortized time.

The difficultly is in supporting both **cut** and **merge**. When "merge" is not required and heap-order need not be maintained, the problem is solved using link-cut trees, introduced in [2].

When "merge" is supported (and heap-order is maintained), but support for "cut" is dropped, the paper that introduces this open problem [1] gives a viable algorithm.

2 Conjecture

It is conjectured in [1] that Sleator and Tarjan's link-cut trees [2], augmented to support "merge" as described in the next section, are a sufficient data structure to support all of the above operations in $O(\log n)$ amortized time, including "merge" and "cut", even under the assumption of heap-order.

Proving this would be sufficient to solve the above open problem.

3 Dynamic trees problem and link-cut trees

The *mergeable trees problem* (described above) is a variant of the *dynamic trees problem*, a more general problem where we desire the above operations except for "merge", and where we do not assume heap order.

One data structure that solves the *dynamic trees problem* is the **Link-Cut Tree**.

The key idea behind the link-cut tree is that of **solid paths** (also refererred to as "preferred paths") which are maintained in their own auxiliary binary trees.

The **expose** operation, applied to v, puts v in a preferred path with the root of the tree containing v. Also, v becomes the root of the auxiliary tree representing this path.

All operations on the link-cut tree rely on a constant number of calls to **expose** (also referred to as "access"), and their running times are dominated by the $O(\log n)$ amortized running time of **expose** (assuming the auxiliary trees are splay trees).

Note that the "link" and "cut" operations only require modifying only one edge (of the main tree). The "merge" operation, however, may required modifying many edges.

So how do we modify the link-cut tree to accomodate the merge operation and maintain heap order?

4 Augmenting link-cut trees to support merge

In [1], Sleator and Tarjan modify link-cut trees. They introduce the "merge" operation by modifying the existing "link" operation in a very simple way.

Specifically, to perform $merge(u, v)$, an expose operation is first performed on u, then on v. A third expose operation is then performed on $w = nca(u, v)$ (the nearest common ancestor). The result is that the paths from u to w and from v to w (both paths excluding w itself) are in their own auxilliary trees.

$merge(u, v)$ then combines the two paths.

$merge(u, v)$ can either do this naively, merging by insertion, or intelligently, using interleaved merges.

5 Merging by insertion

In the naive method, the shorter path is merged into the longer path one node at a time, using the pre-existing "link" operation.

The result is an inferior $\Omega(n^{3/2})$ time bound.

6 Interleaved merges

This method finds contiguous paths that can be merged as a whole, resulting in a more acceptable $O(\log^2 n)$ time bound.

However, the analysis is not tight assuming the auxiliary trees are splay trees. In this case, it is possible that the bound can be tightened to $O(\log n)$.

As mentioned above, if this is the case, then the open problem is solved.

7 Disallowing cuts

The authors of [1] also present an algorithm to solve merge in $O(\log n)$ time assuming that arbitrary cuts are not allowed.

8 Motivation

The original motivation for *mergeable trees problem* is an algorithm in [3] that computes the structure of 2-manifolds imbedded in R^3. Tree nodes correspond to critical points of the manifold.

9 Previous work

The *mergeable trees problem* is introduced in [1]. Link-cut trees are devloped in [2]. For similar open problems, see the "Final Remarks" section of [1].

By far the most important paper is [1].

References

[1] L. Georgiadis, R.E. Tarjan, and R.F. Werneck *Design of Data Structures for Mergeable Trees*, Symposium on Discrete Algorithms, pages 349-403, 2006.

[2] D.D. Sleator, R.E. Tarjan, *A Data Structure for Dynamic Trees*, Journal. Comput. Syst. Sci., 1983.

[3] P.K. Agarwal, H. Edelsbrunner, J. Harer, and Y. Wang. *Extreme Elevation on a 2-manifold.* Proc. 20th Symp. on Comp. Geometry, pages 357-365, 2004.