6.897 Advanced Data Structures (Spring'05)

Prof. Erik Demaine TA: Mihai Pătraşcu

Problem 4 – Solution

The were many wrong solutions to this problem. In particular, note that a simple application of B-trees will not suffice: splitting a node requires $O(b)$ time, and you have to split $O(\lg_b n)$ nodes in the worst-case (even though it's only $O(1)$ nodes amortized). There were also some very different good solutions. Here is one particular solution that works.

We represent each component by a tree, with no connection to the shape of the represented tree. The nodes of the represented tree are in the leaves, while internal nodes of the data structure tree are just for organization purposes. The root holds a pointer to the root of the represented tree, so we only need to search for the root in the data structure's tree.

Each node has a certain height $h(v)$. We maintain the following invariants:

1. heights strictly increase on any leaf-to-root path (but they need not be consecutive).
2. a root v has only children of height $h(v) - 1$.
3. a root has at most b children.
4. a child of a root having height h (the child, not the root) has at least $(b/2)^h$ leaves in its subtree.

A query simply walks up the tree to the root. By the last invariant, any height is at most $1 + \lg_{(b/2)} n = O(\lg_b n)$. By the first invariant, the running time is bounded by the maximum height.

For the sake of union, we maintain explicitly the children of a root, as a list of at most b elements. Assume that a union only happens between two roots u and v (otherwise, call find on the vertices). Assume by symmetry that $h(u) \le h(v)$.

If $h(u) = h(v)$, we merge the two lists of children, all of which are at depth $h(u) - 1$. If in total the lists have size $\le b$, we make all children point to u, and we destroy v. If the list exceeds b elements, we break it in two, and assign each half as children of u and v. We also create a node which becomes the new root, of height $h(u)+1$, and make u and v children of the new root. Because u and v have $\ge b/2$ children at height $h(u) - 1$, the size of their subtrees satisfy the last invariant.

If $h(u) < h(v)$, we consider all children of u, and add them as children of an arbitrary child of v. We then destroy u. Note that children of u have height $h(u) - 1 \le h(v) - 2$, so by adding them below children of v (at height $h(v) - 1$), we are not violating the first invariant. Also, we are not violating the last invariant, since adding things below a node can only increase the size of its subtree.