Linear Time Restricted Union/Find

Alon Itai

July 2, 2006

We consider the following restricted **Union/find** problem:

The universe is $[0..n-1]$ and the unions are restricted to sets of consecutive integers. Initially all the sets are singletons. To maintain the above condition, a union between sets A and B, $Union(A, B)$, is permitted only if

$$\max\{A\} = \min\{B\} - 1.$$

Gabow and Tarjan [1] considered a similar problem. However, they assumed that the Union-tree is known in advance. Here, the order of the unions is not known—only that the resultant sets are consecutive. We show that if the tree is known in advance, the nodes can be renumbered so as to fit into our framework. Since the preprocessing requires linear time, our procedure may be used to get an alternative, and we believe simpler, proof of their results.

Without loss of generality we identify each set with its maximal element. Let $b = \log\log n$. We partition $[0\,..\,n-1]$ into n/b *micro-universes*, such that the kth micro-universe consists of $[(k-1)b\,..\,kb-1]$.

The data structure consists of two tiers. The upper one is a regular **Union/find** data structure whose universe consists of the first and last element of each micro-universe (a total of $2n/b$ elements). The operations on this set are *macro-find* and *macro-union* they are carried out by any efficient **Union/find** algorithm that requires $m\alpha(m, N)$-time for m operations on a universe of size N. During the course of the unions, we keep track of the largest member of each set.

The lower data structure consists of operations within a micro-universe. Since there are n/b micro-universes we number them $0\,..\,n/b-1$. The micro-universe of an integer p is thus determined by its $\log n - \log b$ most significant bits, and its id within its micro-universe by the $\log b$ least significant bits.

Initially each micro-universe consists of b singletons. As a result of a series of unions each micro-universe is partitioned into subsets. There may be $\beta = 2^{b-1} = 2^{\log\log n - 1} = (\log n)/2$ such partitions. We maintain each micro-universe as a bit vector v of length b: $v[i] = 1$ iff i is the largest member of its set.

We also consider v as an integer in the range $0 \,..\, 2^b - 1$, and construct an array T of length 2^b, such that $T[v][i] = \min\{j \geq i \mid v[j] = 1\}$, if $v[i] = v[i+1] = \ldots = v[b-1] = 0$ we set $T[v][i] = ... = T[v][b] = b$. All the β tables $T[v]$ are prepared in advance at a preprocessing stage.(Time $\beta \log\log n = O(\log n \log\log n)$).

To perform $find(j)$ let j be the ith member of micro-universe s, i.e., $j = bs + i$. Let micro-universe s be represented by bit vector v. We first look-up $f = T[v][i]$. If $f < b - 1$ then return $f + s * b$. otherwise ($f = b$) let $x =$ *macro-find*(sb). Return $xb + T[w][0]$, where w is the bit vector of micro-universe x.

To execute $B = Union(A, B)$, assume that $A = [l \,..\, j]$ and $B = [j+1 \,..\, r]$. Let $j = kb + i$ (for some $0 \leq i < b$), and v the bit-vector representing micro-universe k. As result of the union i is no longer the maximum member of its set. Therefore, set the ith bit of v to 0. Call the new bit vector v'. We now perform a find $f =$ *micro-find*(v', i). If $f = b$ then we perform $B =$ *macro-union*(A, B). Otherwise B is a subset of micro-universe k, so the result of the union is A.

Each union requires a micro-union $O(1)$-time, and possibly a macro-union, also $O(1)$-time.

Each find requires a micro-find $O(1)$-time and possibly a macro-find. In a sequence of m finds there may be at most m macro-finds. Thus we need $O(m\alpha(m, n/\log\log n)) = O(m+n)$-time (the last equality is proved in [2, p. 221]).

1 Known trees

If the tree of unions is known in advance, we may do a (depth-first) search on the tree. In the depth first search, observe the order of the unions, i.e., scan the children of a node v, by the order by which they were **union**ed with a set containing v. Renumber the vertices according to the depth-first number.

Lemma 1.1 *All unions are between sets whose depth-first number consists of consecutive integers.*

Proof: Let S be the smallest non consecutive set obtained by the **Union/find** algorithm. Let v be the root of this set, i.e., S was created by **Union**(S_1, S_2), where v was the root of S_1 and u the root of S_2, and the **Union** was implemented by linking u to v. Because of the minimality of S, both S_1 and S_2 are consecutive. Also, since v belonged to S_1 before S_2 was added, the children of v which belong to S_1 were linked to v before u was. Thus, all vertices of S_1 have depth-first search number which is smaller than that of u.

If S were not consecutive, then there must be a vertex $w \notin S_1$ whose depth-first search number, $dfs(w)$, satisfies

$$\max_{v' \in S_1} dfs(v') < dfs(w) < dfs(u). \tag{1}$$

□

By the order by which the depth-first search algorithm scanned the children of v, w was linked to v before u. By the definition of S_1, $w \in S_1$, contradicting (1).

So if we renumber the elements of the universe by the depth-first numbers, the resultant **Union/find** problem fits into our framework. Therefore, we can apply our algorithm. The depth-first search requires linear time. The use of depth-first search numbers also adds $O(1)$ time to each operation. Thus, the resultant algorithm also requires $O(n)$ time, just like Gabow and Tarjan's.

References

[1] Gabow, H., and R. E. Tarjan, A linear time algorithm for a special case of disjoint set union, *J. Computer and System Science*, 30,(1985), 209-221.

[2] Tarjan, R. E., Efficiency of a good but not linear set union algorithm, *J. Assoc, Comput. Mach.*, 22 (1975), 215-225.