COS 423 Problem Set 5 Due Wed. 3/11/09
Spring 2009

Collaboration Allowed

1. The purpose of this problem is to compare the linked-list representation of disjoint sets (CLRS Section 21.2) with the disjoint-set forest representation (CLRS Section 21.3). Consider the linked-list representation, using the weighted union heuristic versus the disjoint-set forest representation using the same weighted union heuristic and path compression. We assume that both methods use the same tie-breaking rule when combining two sets of equal size. Specifically, to do union(x, y), the list method changes the representative of every element in the smaller set; if the two sets have the same size, it changes the representative of all elements in the set containing y. The tree method makes the root of the larger tree the parent of the root of the smaller tree, where size is measured by the number of elements. If the two sets have equal size, it makes the root of the set containing x the parent of the root of the set containing y.

As a measure of efficiency, we charge each method 1 time unit per operation plus one per representative changed in the list method, one per parent pointer changed in the tree method.

(a) Prove that on any sequence of set operations, the time spent by the tree method never exceeds the time spent by the list method.
(b) For every value of n, give a sequence of set operations including n make-set operations on which the list method spends $\Omega(n \log n)$ time but the tree method spends $O(n)$ time. The sequence should include at least one find operation on each element of the set.

2. (Maintenance of simple linear equalities). Give an efficient algorithm that can process a sequence of the following kinds of operations:

"Declare X = C" where X is a variable and C is a real number;
"Declare $X = Y$ + C" where X and Y are variables and C is a real number;
"Compute X" where X is a variable;
"Compute $X - Y$" where X and Y are variables.

In the case of a declaration, the algorithm should add the equation to its representation of the set of declarations, and return "incompatible" if the new equation together with the old ones has no solution. (Once the equations are incompatible, the algorithm can halt and refuse to accept new declarations.) In the case of a computation, if the variable or difference of variables has a unique possible value, the algorithm should return it; otherwise, it should return "multiple solutions" and give two.

You can assume that addition and subtraction of real numbers take O(1) time, and that the set of variables is $X_1, X_2, \ldots, X_n$, so that they can be represented by the integers 1 through n. (You do not need to deal with looking up the names of variables.) Your algorithm

should process a sequence of m declarations and intermixed computations in $O(m\alpha(n))$ time.

3. (Extra credit) Extend your solution to problem 2 to handle declarations of the form "Declare AX + BY = C", where X and Y are variables and A, B, and C are real numbers, possibly zero, and to handle computations of the form "Compute AX + BY", where X and Y are variables and A and B are real numbers, possibly zero.