Algorithms Core 15-750 PROBLEM SET # 3 Spring 1998

DUE: 10:30AM Wednesday 4 March 1998

1 Lazy Find

For the disjoint set problem we defined a collapsing find. Namely, **Find(**a**,**S**)** would set the parent of each ancestor of a, including a to the root. Note that the parent of the root is the root. Let's now define **L-Find**. In this case, **L-Find(**a**,**S**)** will set $parent(b) = parent(parent(b))$ for each b that is an ancestor of a including a. That is the grandparent of each ancestor b of a is now the parent of b. The implementation **L-Find(**a**)** is slightly easier since we no longer need a stack to up date the parent pointers. But it will still take the depth of a time.

Show that if we apply a sequence of m **L-Find**s to a tree T consisting of n nodes it will take time at most $O(n \log n + m)$ to perform the **L-Find**.

Hint: Use either a potential function argument or amortized analysis.

2 Parsing Uniquely Decipherable Codes

Give an $O(m \cdot l^2)$ time algorithm to parse a message of length m over a UD code C where l is the length of the longest word. You may pre-process C and not pay this pre-processing time.

3 Increment a Priority

In class we showed that Fibonacci heaps could implement the following operations in the following amortized times:

findmin	$O(1)$
insert	$O(1)$
deletemin	$O(\log n)$
decrement	$O(1)$
delete	$O(\log n)$

Consider the operation **increment**(h, i, Δ) which increases the value of i by Δ.

Give upper and lower bounds on the amortized cost for **increment** if we added it to the above list of operations.