COS 423 Problem Set 4 Problems 1 and 2 due Wednesday March 30 11 AM
Spring 2011 Problem 3 (extra credit) due on or before April 27
Tarjan
Collaboration allowed on problem 2, not on 1 and 3.

1. Let T be a tree with edge weights and let B be its Borůvka tree. Is it always true that the weights of the edges of B increase toward the root? If so, prove it. If not, give a counterexample. (For an example that is NOT a counterexample, see Lecture 11, slide 19.)

2. The goal of this problem is to extend the upper bounds on the amortized find time using path compression in two directions: to handle the case $m << n$, and to make the bound for *find*($x$) a function of the size of the set containing $x$ when the find occurs, rather than of the total number of elements in all sets. Consider an intermixed sequence of make-set, link, and find operations, with $n > 0$ the number of make-set operations, $m > 0$ the number of find operations, all finds done with path compression, and all links done by rank.

(a) Call a node "live" if it is returned by some find and "dead" otherwise. Then there are at most $m$ live nodes, one per find. (There may be fewer, because two different finds can return the same node.) Prove that the total number of occurrences of dead nodes on find paths is $O(n + m)$. Thus to bound the total number of occurrences of nodes on find paths it suffices to bound the number of occurrences of live nodes on find paths.

(b) Use the result in part (a) to prove that the total time for finds is $O(n + m\alpha(n, d))$. Hint: adapt the proof of the $O((n + m)\alpha(n, d))$ bound on parent changes given in Lecture 13. To do this you may find it useful to partition the live nodes into low-rank nodes and high-rank nodes as in the proof, but to change the upper bound on the rank of low-rank nodes. A bound you might try is $2n/m + d$.

(c) If $x$ is a node, let $\alpha(x) = \alpha(r(x), d)$. Call a non-root node $x$ "active" if $\alpha(x) = \alpha(a(x))$ and "passive" otherwise. Prove that an active node can become passive, but once passive it stays passive. Consider an operation *find*($x$). Let n' be the number of nodes in the tree containing $x$ during the find. Show that the number of passive nodes on the find path is at most $\alpha(n', d)$, as is the number of non-root nodes that are last in their level. Show that the total number of occurrences of active nodes on find paths that are not last in their level is $O(n + m)$.

(d) Use the results in (a), (b), and (c) to obtain the following amortized time bounds: O(1) for make-set and link, and $O(\alpha(n', d))$ for a find in a set containing $n'$ elements.

Note: the goal of this problem is to do part (d). Part(c) suffices for this; parts (a) and (b) are in fact not needed. The bulk of the available points will be given for a solution to (d). If you have another way to prove (d) that does not use (a), (b), and/or (c), feel free to do it your way.

3. (Extra credit) Slides 27-38 of Lecture 13 on the analysis of path compression (which we did not cover in class) derives the following recurrence bounding the number of parent pointer changes done by a sequence of finds with path compression:

$C(n, m, r) \leq C(n', m', r') + C(n - n', m - m', r - r') + n' + m - m'$

Here $n$ and $n'$ are the number of elements in the original problem and in the recursive subproblem on the low-rank nodes, respectively; $m$ and $m'$ are the number of finds in the original problem and in the low-rank subproblem, respectively; $r$ and $r'$ are the maximum rank in the original problem and in the low-rank subproblem, respectively. This recurrence applies whether linking is naïve or by rank, and holds for any split into subproblems.

(a) Use the recurrence to prove that if naïve linking is used, $C(n, m, r) = O(n + m\log_{d+1} n)$, where d is the density of finds, $d = \lceil m/n \rceil$, $n > 1$, and $m > 0$.

(b) Use the recurrence to show that if linking by rank is used, $C(n, m, r) = O(n + \alpha(m, d)m)$, where $n > 1$ and $m > 0$.

Bonus points will be awarded for the simplest, most elegant solution. You have until the last day of class to do this optional extra credit problem. Here comes the caveat: this problem is extra credit because I don't (yet) know how to do it. But I am confident it can be done. And you are the ones to do it!