COS 423 Problem Set 2 Due Wednesday March 12 in class
Spring 2014 No collaboration
Tarjan

1. (Fibonacci heap operations can take linear time in the worst case)
   (a) For every positive integer n, give a sequence of n operations ending in a *delete-min* such that the *delete-min* takes $\Omega(n)$ time (worst-case).
   (b) For every positive integer n, give a sequence of n operations ending in a *decrease-key* such that the *decrease-key* takes $\Omega(n)$ time(worst-case).
2. Consider Fibonacci heaps. Suppose we change the way we do links, as follows: when linking two trees with roots x and y and $k(x) < k(y)$ (so that y becomes a child of x), we call the link *fair* if $r(x) \leq r(y)$, and *unfair* otherwise. When doing a fair link, we increase r(x) by 1; when doing an unfair link, no ranks change. These rules apply to all links done in any operation. We still do *delete-min* by preferentially linking roots of equal rank, but now some of the links of unequal-rank nodes may be fair, and links done during *insert*, *meld*, and *decrease-key* may be fair. Prove that Theorems 3 and 4 in the lecture slides (the rank bound and the running time) remain true with this change.
3. (MTF beats both TR and FC) In both (a) and (b), consider a sequence of access operations on a list of n items 1 through n, initially in sorted order.
   (a) Prove that for *some* positive constant c, for every n, there is an access sequence on which the cost of TR(transpose) is at least cn times the cost of MTF(move-to-front). Recall that the cost to MTF of accessing the item in position k is $2k - 1$, including the cost of the $k - 1$ swaps to move it to the front of the list, and the cost to TR of accessing the item in position k is $k + 1$, including the cost of the single swap to move it one position toward the front of the list. Note: You pick the value of c. You then need to produce, for each $n \geq 1$, an access sequence on which the TR cost is at least cn times the MTF cost.
   (b) Prove that for *any* c, there is a sequence of accesses on which the cost of FC (frequency count) is at least c times the cost of MTF. Recall that FC maintains a count of the number of accesses to each item, initially 0. When an item is accessed, its count is increased by 1, and it is repeatedly swapped with the preceding item on the list as long as its frequency count is greater than that of the preceding item. The cost to FC of accessing the item in position k is thus between k and $2k - 1$ (inclusive), depending on how many swaps it does. (There is no cost to FC for maintaining the frequency counts.) Note: You do not get to choose c; given any c, you need to produce a sequence of accesses on which the FC cost is at least c times the MTF cost. Both the number of accesses m and the number of list items n can depend on c, however.
   (c) In part (b), the number of accesses m and the number of list items n in your bad example will depend on c. Inversely, c will depend on n and m. How large can you make c as a function of n (for arbitrarily large m)?

4. (Analysis of MH) The *move half-way to root* (*MH*) update rule for lists is like MTF, except that it moves the accessed item only half-way toward the front of the list. Specifically, if the item is in position $2k$ or $2k + 1$, it is moved to position $k + 1$ by swapping it past the items in positions $k + 2$,… Prove that for some constant c independent of the number of items n on the list and the number of accesses m, *MH* is c-competitive with the optimum off-line algorithm, on every access sequence. Make c as small as you can.