CS 361A - Advanced Data Structures and Algorithms

Autumn Quarter, 2005

Homework #3 (Due: 11/28/05)

1. (15 points) Prove the best lower bounds you can for the competitive ratio of randomized list update algorithms against adaptive online (ADON) and adaptive offline (ADOFF) adversaries.

2. (10 points) Show that the randomized paging algorithm we called MARK is H_k-competitive in the case when there are only $k+1$ distinct pages.

3. (10 points) Define a marking algorithm for the paging problem as follows: the algorithm proceeds in phases; at the beginning of each phase, all pages are unmarked; whenever a page is requested, it is marked; upon a page fault, only an unmarked page (chosen by a rule to be specified by the algorithm) can be evicted; and, a phase ends just before the first fault which occurs after every page in the fast memory is marked.

   Are LRU and FIFO expressible as marking algorithms? Justify your answer.

4. (15 points) Consider the following randomized paging algorithm: upon a page fault, evict a page chosen uniformly at random from among the k pages currently in the fast memory. Show a lower bound of k on the competitiveness of this algorithm against an oblivious adversary. What is the best upper bound you can prove for this algorithm?

5. (30 points)

   (a) Let T be a random treap for the set X, and let $x, y \in X$ be two elements whose ranks differ by r. Prove that the expected length of the (unique) path from x to y in T is $O(\log r)$.

   (b) Let us now analyze the number of random bits needed to implement the operations of a treap. Suppose we pick each priority p_i uniformly at random from the unit interval $[0, 1]$. Then, the binary representation of each p_i can be generated as a (potentially infinite) sequence of bits which are the outcome of unbiased coin flips. The idea is to generate only as many bits in this sequence as is necessary for resolving comparisons between different priorities. Suppose we have only generated some prefixes of the binary representation of the priorities of the elements in the treap T. Now, while inserting an item y, we will have to compare its priority p_y to others' priorities to determine how y should be rotated. While comparing p_y to some p_i, if their current partial binary representation can resolve the comparisons then we are done. Otherwise, they have the same partial binary representation and we keep generating more bits for each till they first differ. (Refer to the end of Section 3 in the paper.)

Compute a tight upper bound on the expected number of coin flips or random bits needed for each update operation.

6. (20 points) Consider the following version of Mulmuley's games. The pool consists of the sets P, B, T and S, where P is a set of p players, B is a set of b bystanders, T is a set of t triggers and S is a set of s stoppers. Assume that the players are totally ordered and that all sets are non-empty and pairwise disjoint. The game consists of picking random elements of the pool, without replacement, until the pool is empty. The value of the game, $G^p_{t,s}$, is defined as the expected value of the following quantity: after *all* triggers have been chosen, and before *any* stopper has been chosen, the number of chosen players who are larger than all previously chosen players. (Note that this is the same as Game E defined in class except for the requirement that we start counting only after *all* triggers have been picked.)

   Bound the expected value of $G^p_{t,s}$ as tightly as you can.