CS 141, Advanced Topics in Algorithms
Spring 2004
Homework 3b
Due Monday, February 9

1. **[10 Points] Professor I. Lai!** Professor I. Lai of the Pasadena Institute of Technology makes the following claim at a colloquium talk: "I have developed a new type of binary search tree called a Splat Tree. Using amortized analysis I have proved that the amortized cost of an insert, find, and delete operation is *asymptotically* better than the $O(\log n)$ amortized cost of these operations in Sleator and Tarjan's Splay Trees. That is, the amortized cost of these operations in a Splat Tree is $o(\log n)$." (If you're not familiar with "little oh" notation, take a quick look in an algorithms book. It's worth knowing about.) Explain precisely but succinctly why Professor Lai is lying (as usual!).

2. **[15 Points] Union-Find with Partial Path Compression.** In class we examined the union-find data structure using union-by-size and path compression. One disadvantage of path compression is that it is a two-phase process: First we must "climb" the path to find the root. Then, we go back and set the parent of each node to be the root.

   A simpler approach is to use *partial path compression* in which each node on the path has its parent pointer changed to point to its grandparent. This process can be done in just one pass as we climb up the path. Show that using partial path compression instead of full path compression still allows us to perform any sequence of n MAKESET, UNION, and FIND operations in time $O(n \log^* n)$ time.

3. **[15 Points] Union-Find Again.** Consider again the union-find data structure using union-by-size and path compression. Assume that we will perform a sequence of n MAKESET, UNION, and FIND operations where all of the MAKESETs are performed before the UNIONs and all of the UNIONs are done before the FINDs. Show that the total running time is now asymptotically even better than $\Theta(n \log^* n)$.

4. **[20 Points] ACN Programming Competition!** You are competing in the International ACN (Association of Computer Nerds) Programming Competition. In one of the programming problems, you need to implement a queue. Unfortunately, you are pressed for time. Fortunately, you already have good working code for a stack. You may assume that the stack is implemented efficiently and supports operations PUSH, POP, and SIZE (which returns the number of items on the stack) in time $O(1)$. Show how 2 stacks can be instantiated to implement a queue. Moreover, show that the amortized running time for ENQUEUE and DEQUEUE are $O(1)$ using this 2-stack implementation.