Problem Set 1—Due Friday, Oct. 13

Guidelines for writeups

When describing an algorithm first attempt to give a high level overview of your approach, then fill in details as need to justify your time bound and correctness. Your pseudo-code should always be commented (unless it is self commenting). You are also encouraged to describe your solutions in terms of known algorithms and data structures (e.g. sort the numbers, insert into a balanced binary search tree, delete the minimum of a heap, could all be described with little or no further elaboration).

For proofs, strive for clear, clean arguments. It will always be somewhat a matter of taste which steps can be skipped, but try to avoid proofs of obvious points, and be clear on anything tricky. Define your notation carefully. This will often allow you to give a much crisper argument.

Submission details

Homework is due at Noon on the due date (it can be turned in to the TA (hard copy or electronic) or put in the homework box in 2131). Everyone gets two free late days (M-F count) for the quarter. Once you have used up your late days there is a 20% penalty per day for turning in late homework. Also, no homework will be accepted more than four days late (weekends count here).

(30) Problem 1. Problem 1-3 in the text (page 18). Also

e. What is the expected number of inversions in a random permutation (Hint: use a recurrence relation)?

(10) Problem 2. Problem 12.3-1 in the text (page 231).

(15) Problem 3. Problem 12-5 in the text (page 242) parts a) and b).

(20) Problem 4. 12.4-5. Hint: start by finding an exact expression for p(n,m). Then use equation (2.7) to transform each term in your expression. Why is this result relevant to perfect hashing?

(25) Problem 5.

The following questions relate to the perfect hashing scheme we described.

(a) One drawback of the the perfect hashing scheme is the need to compute two hash functions. Argue that for many items we can expect to use a **very** easy to compute second hash function.

(b) Our perfect hashing scheme assumed the set of keys stored in the table is static. Suppose instead that we want to add a few new items to our table after the initial construction. Suggest a way to modify our intitial construction so that we can insert these new items using no new space and without making significant changes to our existing table (in particular, we don't want to change our initial hash function). Your scheme should still do lookups of all items in O(1) time, but you may use a bit more initial space.

(c) Suppose now that we want to delete some of our initial items. Describe a simple way to support deletions in our perfect hashing scheme.