Problem Set 2

This problem set is due **at the beginning of** class on *Thursday, February 27, 2003*.

Each problem is to be done on a separate sheet (or sheets) of paper. Mark the top of each sheet with your name, 6.046J/18.410J, the problem number, your recitation section, the date, and the names of any students with whom you collaborated. Problems marked **Optional** carry no credit and don't have to be handed in. They are only meant to challenge those who like being challenged!

Problem 2-1. Matching Nuts and Bolts

A little known fact about 6.046 TA Yoav Yerushalmi is his favorite hobby of assembling furniture while blindfolded. He is currently interested in building a table and set of chairs, so he purchases several boards along with a collection of n nuts and n bolts to connect them together. The nuts are all of different sizes as are the bolts, and for each nut there is a unique bolt that it fits.

Unfortunately, while thinking about what to present during his next recitation, Yoav slips and spills all of the nuts and bolts across the floor. Being blindfolded, he cannot compare two nuts or two bolts together — he can only compare a nut with a bolt and determine if the nut fits, if it is too large, or if it is too small. It is easy to match the nuts and bolts in $O(n^2)$ time by comparing every (nut, bolt) pair; however, we would like to match the nuts and bolts much more efficiently.

(a) Help Yoav in his predicament by designing a randomized algorithm for matching the nuts with the bolts. Your algorithm should have an expected running time of $O(n \log n)$.

(b) Prove that any algorithm that solves part (a) must make $\Omega(n \log n)$ comparisons in the worst case.

Problem 2-2. Counting Inversions

Please do problem 2-4 on page 39 of CLRS.

Problem 2-3. Matrix Vector Products

Let $\mathbf{A}$ be a $n \times n$ matrix, and $\mathbf{x}$ be a $n \times 1$ column vector. In many numerical eigenvector computations, we are required to compute $(\mathbf{A} \ldots \text{m times} \ldots (\mathbf{A}(\mathbf{A}\mathbf{x}))) \equiv \mathbf{A}^m\mathbf{x}$.

(a) What is the running time for computing the result by a series of matrix-vector products? Express your answer in terms of m and n.

(b) Design an algorithm that outperforms the above technique when m is large compared to n. Quantify how large m must be in relation to n for this algorithm to be asymptotically faster than the approach from part (a). You may assume that matrix multiplication can be done in $O(n^{\lg 7})$ time.

Problem 2-4. Almost Sorted!

The final for 6.046 has been graded and all the papers have been alpahabetically sorted. TA Yoav Yerushalmi picks up the pile in order to enter the grades but unfortunately he slips (yet again!) and the pile of papers ends up scattered on the floor. Yoav picks up the papers and hurriedly reassembles the pile but the professors are clearly annoyed. The pile will have to be sorted all over again.

By way of apology, Yoav argues that the resorting will not take too long since the pile is almost sorted. Due to the particular physics behind the scattering, there exists a small number k such that none of the papers moved more than k places away from their actual position. In other words, if $A[1..n]$ is the sorted pile, then paper $A[i]$ is guaranteed to lie somewhere between positions $i-k$ and $i+k$ in the new pile. Yoav claims that INSERTION-SORT will actually sort the scrambled pile in $O(kn)$ time.

(a) Prove that Yoav is right[1]

(b) Prof Indyk has always been very interested in lower bounds. After thinking a little, he claims that any algorithm that solves this problem must take $\Omega(n \log k)$ time. Prove his claim. (Hint: Try to come up with a lower bound on the number of possible permutations for the scrambled pile)

(c) (**Optional**) The gap between the lower and upper bounds for this problem arouses Prof Tidor's curiosity. The hard taskmaster that he is, he challenges the TAs to come up with an algorithm to sort the scrambled array in $O(n \log k)$ time. Help the TAs in rising to the challenge. (Hint: Assuming k is known beforehand, how quickly can the scrambled array's median be computed ?)

[1] Proofs based on the premise that "Yoav is always right" are not sufficiently rigorous for this class.