CS 573: Graduate Algorithms, Fall 2011
HW 2 (due in class on Tuesday, September 27th)

This homework contains five problems. **Read the instructions for submitting homework on the course webpage**. In particular, *make sure* that you write the solutions for the problems on separate sheets of paper. Write your name and netid on each sheet.

Collaboration Policy: For this home work students can work in groups of up to three students each. Only one copy of the homework is to be submitted for each group. Make sure to list all the names/netids clearly on each page.

Note on Proofs: Details are important in proofs but so is conciseness. Striking a good balance between them is a skill that is very useful to develop, especially at the graduate level.

1. (20 pts) In the selection problem we are given an array A of n numbers (not necessarily sorted) and an integer k and the goal is output the rank k element of A. We saw a linear time deterministic algorithm for this problem. Consider a randomized version where we pick a number x uniformly at random from A and use it as a pivot as in quick sort to partition A into numbers less than equal to x and numbers greater than x. The algorithm recurses on *one* of these arrays depending on k and the size of the two arrays. It can be shown that this algorithm runs in $O(n)$ expected time and has the advantage of being quite simple when compared to the median of median algorithm.

   - (10 pts) Write down a description of randomized quick selection in pseudocode. Show that the expected depth of the recursion of randomized quick selection is $O(\log n)$. *Hint:* Write a recurrence for the depth of the recursion.
   - (10 pts) Let $A_1, A_2, \ldots, A_h$ be h *sorted* arrays where A_i has n_i elements. Let $n = \sum_i n_i$. Assume that the arrays have distinct elements. Describe a randomized algorithm that given integer k finds the k'th smallest element in the combined set of arrays in $O(h \log^2 n)$ expected time. *Hint:* Adapt the randomized quick selection algorithm and the analysis from the first part.
   - Extra credit (20 pts): A deterministic algorithm for the second part above. In fact a time bound of $O(h \log n)$ is achievable.

2. (20 pts) The following problem is Problem 4 in Chapter 5 from the Kleinberg-Tardos book. We omit the detailed motivation. There are n charged particles at integer points $1, 2, \ldots, n$; at point j the particle has charge q_j (can be positive or negative number). These particles exert force on each other according to Coulomb's Law. The force F_j exerted on particle j by the other particles is given as

$$F_j = \sum_{i<j} \frac{C q_i q_j}{(j-i)^2} - \sum_{i>j} \frac{C q_i q_j}{(j-i)^2}.$$

   In the above C is an absolute constant that is known. The goal is to compute F_j for each j. A naive algorithm runs in $O(n^2)$ time. Show how one can compute all F_j in $O(n \log n)$ time.

3. (20 pts) Consider a special case of 3-SAT called Linear 3-SAT. In Linear 3-SAT the input is a 3-CNF formula φ with the following locality property: if φ is over n variables they are numbered 1 to n in such a way that each clause contains only variables whose indices differ by at most 10. Show that Linear 3-SAT can be solved in linear time; that is, given a formula satisfying the above property your algorithm should run in time linear in the size of the formula.

4. (20 pts) Given a graph $G = (V, E)$ a set of edges $E' \subseteq E$ is said to be an *edge-cover* if each node $v \in V$ is incident to some edge in E' (in other words the edges in E' cover all the nodes). Let T be a tree in which each edge e has a non-negative weight $w(e)$. Describe an efficient algorithm to find a minimum weight edge-cover in a given tree T.

5. (20 pts) Let x be a string over some finite and fixed alphabet (think English alphabet). Given an integer k we use x^k to denote the string obtained by concatenating k copies of x. If x is the string **HELLO** then x^3 is the string **HELLOHELLOHELLO**. A *repetition* of x is a prefix of x^k for some integer k. Thus **HELL** and **HELLOHELL** are both repetitions of **HELLO**.

   An *interleaving* of two strings x and y is any string that is obtained by shuffling a repetition of x with a repetition of y. For example **HELwoLOHELLrldwOH** is an interleaving of **HELLO** and **world**.

   Describe an algorithm that takes three strings x, y, z as input and decides whether z is an interleaving of x and y.

Questions to ponder:

1. Consider the deterministic linear time algorithm for selection. What is the recurrence you obtain if you choose the lists to be of size 3 or 7 instead of 5?

2. Recall the recurrence $T(n) = n + \frac{1}{n}\sum_{i=1}^{n}(T(i-1) + T(n-i))$ that arises in the analysis of randomized quick sort. Show by the guess and verify method that $T(n) = O(n \log n)$.

3. Can you analyize a linear time randomized selection algorithm via the same idea as randomized quick sort?

4. Read Problem 2.33 from Dasgupta etal book to see another example of the power of randomization. We will address related issues later in the course.

5. Can you solve the minimum weight vertex cover problem in a tree via dynamic programming? See Problem 6.21 in Dasgutal etal book.

6. Longest common subsequence is another prototypical dynamic programming problem. See Problem Problem 6.8 in Dasgupta etal book.