Homework #2: Sorting and Selection
Due Date: Tuesday, February 17, 1998 (9:00am)

Reading

Read Ch. 9.1, Ch. 10.1, and Ch. 10.3 of CLR.

A note about problem numbers in CLR

There is a somewhat confusing numbering system for problems from the textbook. The book has what it terms "exercises" which are at the end of each section, and "problems" which are at the very end of each chapter. For example, Exercise 9.1-1 is at the end of Section 9.1 on page 174, whereas Problem 9-1 is at the end of Chapter 9, on page 183.

Practice

Recall, these exercises are purely for your own practice. You need only turn in the official problems.

- Ex. 9.1-1, 9.1-3 of CLR
- Ex. 10.3-1 of CLR
- Ex. 10.3-7 of CLR

Collaboration Policy

You may collaborate with others on this entire homework. However, please reread the policy guidelines from Handout #1.

Problems

1. CLR 9-1(a-e), (40 points)
   In this problem, we prove an $\Omega(n \lg n)$ lower bound on the expected running time of any deterministic comparison sort on n inputs. We begin by examining a deterministic comparison sort A with decision tree T_A. We assume that every permutation of A's inputs is equally likely.

   a. Suppose that each leaf of T_A is labeled with the probability that it is reached given a random input. Prove that exactly $n!$ leaves are labeled $1/n!$ and that the rest are labeled 0.

b. Let $D(T)$ denote the external path length of a tree T; that is, $D(T)$ is the sum of the depths of all the leaves of T. Let T be a tree with $k > 1$ leaves, and let RT and LT be the right and left subtrees of T. Show that $D(T) = D(RT) + D(LT) + k$.

c. Let $d(m)$ be the minimum value of $D(T)$ over all trees T with m leaves. Show that $d(k) = min_{1 \leq i \leq k}\{d(i) + d(k-i) + k\}$. (*Hint:* Consider a tree T with k leaves that achieves the minimum. Let i be the number of leaves in RT and $k - i$ the number of leaves in LT.)

d. Prove that for a given value of k, the function $i \lg i + (k-i)\lg(k-i)$ is minimized at $i = k/2$. Conclude that $d(k) = \Omega(k \lg k)$.

e. Prove that $D(T_A) = \Omega(n! \lg(n!))$ for T_A, and conclude that the expected time to sort n elements is $\Omega(n \lg n)$.

2. CLR 9.1-5, (20 points)
Prove that $2n - 1$ comparisons are necessary in the worst case to merge two sorted lists containing n elements each. (*Note:* Make sure your proof is completely general. That is, you cannot simply argue that a particular algorithm you have in mind, might take $2n - 1$ comparisons.)

3. similar to CLR 10.1-1, (20 points)
Show that the second *largest* of n elements can be found with $n + \lceil \lg n \rceil - 2$ comparisons in the worst case. (*Hint:* if you instead find the largest, can you then identify a subset of elements which must then contain the second largest item? How large is this subset?)

4. CLR 10.3-8, (20 points)
Let $X[1..n]$ and $Y[1..n]$ be two arrays, each containing n numbers already in sorted order. Give an $O(\lg n)$-time algorithm to find the median of all $2n$ elements in arrays X and Y. (*Hint:* If $X[k]$ is the median of array X, how quickly can you determine whether it is also the median of the combined $2n$ elements? If it is not, did you learn any new information about the identity of the true median?)

Extra Credit

We want to show that the upper bound in Problem 3 was the best possible result. Give an adversarial argument that *any* algorithm will require $n + \lceil \lg n \rceil - 2$ comparisons in the worst case. (*Hint:* Argue that in the process of finding the second largest element, any algorithm must have also determined the largest element. Now consider the set of elements which lost a comparison to the largest element. How can an adversary try to make this set as large as possible? How does this translate into a lower bound for finding the second largest element?)