CS140: Algorithms
Fall 1999
Assignment 3, 9/22/99
Due Wed, 9/29/99

This assignment is due at the start of class on Wed., 9/29/99. No late homework will be accepted. Your solutions will be graded for clarity, conciseness, and correctness.

1. (20 pts.) Prove the best upper bound you can on the number of comparisons needed to find the second largest in a set of n distinct integers. (The bound should be expressed simply as a function of n; i.e. the answer $O(n)$ will not garner any points.)

2. (20 pts.) Let A be an array of n distinct integers sorted in ascending order. We want to determine if there is an index i such that $A[i] = i$.

   (a) Give an algorithm for this problem that makes at most $\Theta(\lg n)$ comparisons.

   (b) Prove that any comparison-based algorithm for this problem must make $\Omega(\lg n)$ comparison.

   (Assume an algorithm for the problem accesses the data through a comparator that accepts queries of the form "$i?A[i]$" and answers $i < A[i]$, $i = A[i]$, or $i > A[i]$.

3. (20 pts.) Let S be a multiset of n integers. We call k the *majority element* of S if k occurs more than $\lfloor \frac{n}{2} \rfloor$ times in S. For example, $\{1, 2, 2\}$ has a majority element 2. The set $\{1, 2, 3\}$ does not have a majority element. We want an algorithm that determines the majority element of a set of integers when one exists. The algorithm is to run on a computer with very limited IO capability. In fact, the only way to access the set S is through a built-in comparator which takes as input $i, j \in [1, n]$ and answers the question "Is $s_i = s_j$?" Design a linear time algorithm that takes as input n and determines the majority of S when one exists through calls to the comparator. Prove your algorithm is correct.

4. (20 pts.) Problem 10.3-5 on page 192 of Cormen, Leiserson, and Rivest.

5. (20 pts.) Problem 10.3-8 on page 192 of Cormen, Leiserson, and Rivest.

BONUS Prove the best lower bound you can for the problem of finding the second largest in a set of n distinct integers.