CS466/666, Fall 2009: Assignment 2

Out: October 5, Due: October 21, 5pm

1. **Median of two arrays:** Let $A[1..n]$ and $B[1..n]$ be two arrays, each containing n integers in sorted order. Show how to find the median of the $2n$ numbers in A and B in $O(\log n)$ worst-case time. You may assume that n is a power of 2 and that all integers are distinct.

2. **Finding the two smallest elements:** Given a set S of n integers, the task is to find both the smallest and the 2nd smallest element in S. You may assume that n is a power of 2 and that all integers are distinct.

   (a) Show how to find the two smallest elements with at most $n+\log n-2$ data comparisons in the worst-case. Your algorithm should be deterministic, but randomized algorithms, or slightly more comparisons, will give partial credit.

   (b) Show that any deterministic algorithm that uses only comparisons needs to use at least $n + \log n - 2$ data comparisons in the worst case to find the two smallest elements.

3. **Monte Carlo to Las Vegas:** Suppose we have a Monte Carlo algorithm A whose worst-case run time is $t_A(n)$ on an input of size n, and that produces a correct answer with probability $p(n)$. Suppose we have a deterministic algorithm T that can test in $t_T(n)$ time whether the answer given by A is correct. We can then create a Las Vegas algorithm B as follows:

   For $i = 1, 2, \ldots$
   Run algorithm A.
   Run algorithm T on the output of A.
   If T certifies that A gave a correct answer, break.

   (a) Show that the expected number of executions of the for-loop is $1/p(n)$.
   This question is essentially a review in basic probability; you may want to consult your STAT230 notes for expected values of some known distribution, but please write your proof up so that it can be read independently.

   (b) What is the expected run-time of algorithm B?

   (c) Recall that the Monte-Carlo algorithm for median-finding failed with probability $3n^{-1/4}$ (or less, but use the exact bound here), used $1.5n + o(n)$ comparisons, and returned with its answer whether it was correct. Use the above to argue that turning it into a LasVegas algorithm gives an algorithm with expected number of comparisons $1.5n + o(n)$.

4. **Minimum enclosing disk for other distance-measure** We studied in class the minimum encosing disk problem for the L_2-distance, i.e., a disk with center-point c and radius r was defined as $\{p \in \mathbb{R}^2 : ||p-c||_2 \leq r\}$, where $||p-c||_2 = \sqrt{(p_x-c_x)^2 + (p_y-c_y)^2}$.

   One can define the same problem using other distance-measures. Two common measures are the L_1-*distance* $||p-c||_1 = |p_x-c_x|+|p_y-c_y|$, and the L_∞-*distance* $||p-c||_\infty = \max\{|p_x - c_x|, |p_y - c_y|\}$. An L_1-*disk* is then the set of points $\{p \in \mathbb{R}^2 : ||p-c||_1 \leq r\}$, and an L_∞-*disk* is the set of points $\{p \in \mathbb{R}^2 : ||p-c||_\infty \leq r\}$.

   (a) Give an algorithm that, given a set of points $p_1, \ldots, p_n$, finds a minimum-radius L_1-disk that contains all points. Your algorithm should be deterministic and have $O(n)$ worst-case run-time, though as always partial credit may be given for slower or randomized algorithms.

   (b) Formulate the problem of finding the minimum-radius L_∞-disk of a set of n points $p_1, \ldots, p_n$ as a linear program with 3 variables and $O(n)$ constaints.