CMPT 705 | Design and Analysis of Algorithms

Exercises on Divide and Conquer. Due: Tuesday, October 4th (at the beginning of the class)

Reminder: the work you submit must be your own. Any collaboration and consulting outside resources must be explicitly mentioned on your submission.

1. You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains n numerical values | so there are $2n$ values total | and you may assume that no two values are the same. You would like to determine the median of this set of $2n$ values, which we will de ne here to be the n-th smallest value.

   However, the only way you can access these values is through queries to the databases. In a single query, you can specify a value k to one of the two databases, and the chosen database will return k-th smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

   Give an algorithm that nds the median using at most $O(\log n)$ queries.

2. Recall the problem of nding the number of inversions. As in the class, we are given a sequence of n numbers $a_1, \ldots, a_n$, which we assume are all distinct, and we de ne an inversion to be a pair $i < j$ such that $a_i > a_j$.

   We motivated the problem of counting inversions as a good measure of how di erent two orderings are. However, one might feel that this measure is too sensitive. Let us call a pair a *significant inversion* if $i < j$ and $a_i > 2a_j$. Give an $O(n \log n)$ algorithm to count the number of signi cant inversions between two orderings.

3. *Hidden surface removal* is a problem in computer graphics that scarcely needs an introduction: when Woody is standing in front of Buzz, you should be able to see Woody, but not Buzz; when Buzz is standing in front of Woody, ... well, you get the idea.

   You are given n nonvertical lines in the plane, labeled $L_1, \ldots, L_n$, with the i-th line speci ed by the equation $y = a_i x_i + b_i$. We will make the assumption that no three of the lines all meet at a single point. We say line L_i is *uppermost* at a given x-coordinate x_0 if its y-coordinate at x_0 is greater than the y-coordinates of all the other lines at x_0, i.e. $a_i x_0 + b_i > a_j x_0 + b_j$ for all $j \neq i$. We say line L_i is *visible* if there is some x-coordinate at which it is uppermost.

   Give an algorithm that takes n lines as input and in $O(n \log n)$ time returns all of the ones that are visible.