Problem Set 6: Divide-and-Conquer Algorithms

Handed out Friday, October 4. Due at the start of class Friday, October 11.

Problem 1. (15 points) Given a list $A =< a_1, ..., a_n >$ of integers, an *off-parity inversion* is a pair a_i and a_j such that $i < j$, $a_i > a_j$, and a_i and a_j are of different parities. (In other words, it is an inversion in which one number is even and the other is odd.) See the example below in Figure 1.

Design an $O(n \log n)$ algorithm that counts the number of off-parity inversions in a list A containing n elements. Justify your algorithm's correctness and derive its running time.

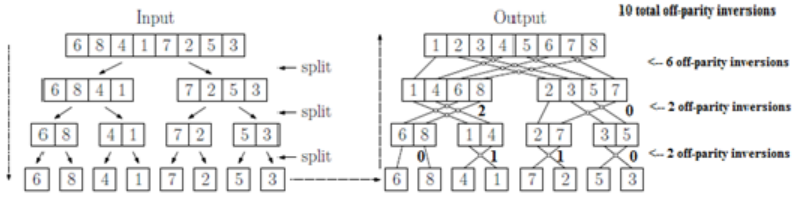

Figure 1: Problem 1.

Problem 2.(15 points) You are given a set of n lines in the plane. Each line is given by a pair of numbers (a_i, b_i), which represents the line $y = a_i x + b_i$. That is, a_i gives the slope of the line and b_i gives its y-intercept. You are told that $a_i < 0$ (all slopes are negative) and $b_i > 0$ (all intercepts are positive). You are asked to count the number of intersections that occur in the positive (x, y)-quadrant. (For example, in Figure 2 below, there are six intersections in the first quadrant, shown as black dots.)

Of course, this would be easy to do in $O(n^2)$ time, but I want you to develop an answer for this problem that runs in $O(n \log n)$ time.

For simplicity, you may assume that the a_i values are all distinct and the b_i values are also all distinct. You may also assume that no two lines intersect exactly on the x-axis or the y-axis.

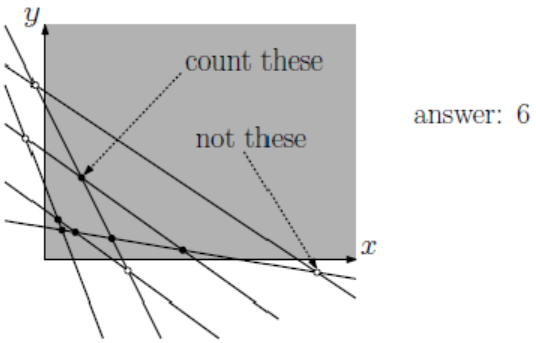

Figure 2: Problem 2.

Problem 3. (15 points) In computer graphics it is often desirable to compute the silhouette of a collection of objects. In this problem, we'll consider a simple example involving rectangles.

You are given a collection of (possibly overlapping) rectangles that extend upwards from the x-axis. Each rectangle is defined by a triple (a_i, b_i, h_i) where a_i and b_i denote the rectangles left and right x-coordinates and h_i denotes the height of the rectangle. The union of these rectangles defines an upper envelope, which consists of a sequence of non-overlapping intervals from left to right along the x-axis. Each interval is associated with a height value of the tallest rectangle that spans this interval.

For example, the input for the rectangles shown in Figure 3 might be as follows. (Note that the rectangles are not given in any particular order.)

(1, 4, 2), (11, 12, 4), (8, 10, 7), (7, 13, 5), (2, 6, 4), (9, 15, 3), (3, 5, 3).

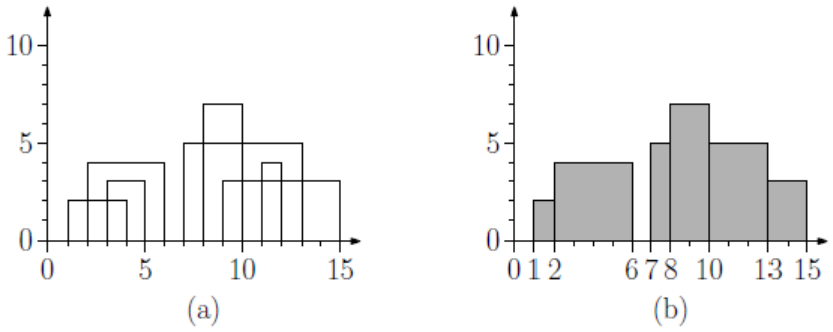

Figure 3: Problem 3.

The output consists of seven intervals (including one interval of height 0). Suppose that the output consists of m intervals. We represent this as an array $x[1..m+1]$ such that the ith interval spans $x[i]$ to $x[i+1]$, and an array $t[1..m]$ where $t[i]$ is the height of the tallest rectangle spanning the ith interval. The output for the above input would be:

$$x = \langle 1, 2, 6, 7, 8, 10, 13, 15 \rangle \text{ and } t = \langle 2, 4, 0, 5, 7, 5, 3 \rangle$$

Design an $O(n \log n)$ algorithm which, given a sequence of n such triples, computes the upper envelope of these union of the rectangles. Derive the running time of your algorithm. You may assume that the values a_i, b_i, and h_i are all distinct.