**CS381** **Homework 7** **Fall 2013**

**Question 1. (20 points)** Let $S$ be a set of $n$ two-dimensional points $p_i = (x_i, y_i)$, $1 \leq i \leq n$, where all the cordinate values are distinct (i.e., no two are equal). Let $M(S)$ denote the subset of $S$ that contains every point $p_i$ such that there is no point $p_j$ of $S$ that has both $x_i < x_j$ and $y_i < y_j$. For example, if $S$ consists of the five points $(36, 17)$, $(7, 20)$, $(5, 19)$, $(56, 18)$, $(1, 78)$ then $M(S)$ consists of the three points $(7, 20)$, $(56, 18)$, $(1, 78)$. Design an $O(n \log n)$ time algorithm for computing $M(S)$ when given $S$ as input.

**Question 2. (20 points)** This question is about the algorithm we covered in class for a longest increasing subsequence (if you missed that lecture or did not take notes, you can find much online material about this topic, e.g., http://en.wikipedia.org/wiki/Longest_increasing_subsequence). Run the algorithm on the following input and state the subsequence it outputs:

$$3, 2, 1, 14, 5, 7, 11, 10, 4, 15, 13, 12$$

Note that there are many increasing subsequences of same length: We are interested in only one of them, the one that is computed by the algorithm we gave in class.

**Question 3. (25 points)** This question is about the intersection-enumeration algorithm we covered in class, when the input to the algorithm consists of the line segments shown in Figure 1 on the next page.

1. List the eight intersection points in the order they are output by the intersection-enumeration algorithm.
2. It is possible for the algorithm to detect an intersection more than once (of course it would output it and insert it in the event list only once, ignoring subsequent re-discoveries of the same intersection).
   (a) List the intersections that are discovered more than once for the segments in Figure 1, and for each such intersection state how many times it is discovered by the algorithm.
   (b) Draw an example of $n$ segments and a single intersection between two of them, in which that intersection is discovered $n - 1$ times by the intersection-enumeration algorithm (your example must work for any $n$, not just for a small value of $n$).
   (c) Briefly explain why the existence of multiple discoveries of the same intersection does not invalidate the claim of $O(n \log n + t \log n)$ time performance for the algorithm, where $t$ is the number of intersections.

*Note.* The textbook contains only the intersection-detection version of the algorithm, so if you missed the lecture or did not take notes you can find the enumeration version of the algorithm at the online lecture notes at http://www.cs.wustl.edu/$\sim$pless/506/l4.html or by reading the following journal paper at the IEEE online library that is accessible from purdue.edu: J. L. Bentley and T. Ottmann., Algorithms for reporting and counting geometric intersections, IEEE Transactions on Computers C28 (1979), 643–647.

**Question 4. (35 points)** Let $S = x_1 x_2 \ldots x_n$ be a sequence of numbers (possibly large and non-integer), and let $p$ be an integer that is $\leq n$. Give an $O(n)$ time algorithm that, when given $S$ and $p$

as input, computes $s_1, s_2, \ldots, s_{n-p+1}$ where $s_i = \max\{x_i, x_{i+1}, \ldots, x_{i+p-1}\}$. You cannot assume that $p$ is constant.

*Note:* A brute-force solution takes $O(p(n-p))$ time, by using $O(p)$ time to compute each $s_i$ value. An $O(n \log p)$ time algorithm is easy to obtain by sliding left-to-right a window of size $p$ along $S$, maintaining the $p$ items of $S$ that overlap the sliding window in a dynamic data structure that supports, in logarithmic time, the operations of $Insert$, $Delete$, and $Max$ (sliding the window by 1 position involves deleting the element that exits the window, inserting the element that enters the window, and doing a $Max$ query). But such a solution is not acceptable, because $p$ is not constant: $p$ could be proportional to $n$ (e.g., $n/2$), in which case this approach would take $O(n \log n)$ time. We seek a solution that runs in $O(n)$ time, no matter what $p$ is.

**Date due: Thursday November 7, 2013**

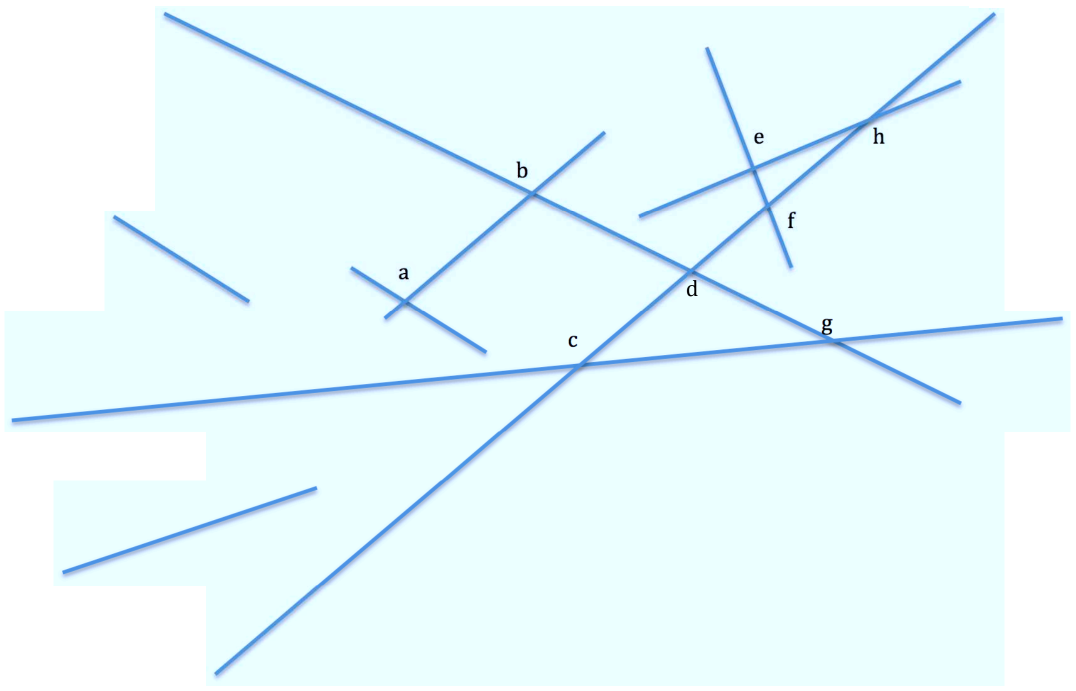

Figure 1: The input segments for question 3