ASSIGNMENT 2

DUE: Tuesday September 28, 11:59 PM. DO NOT COPY. ACKNOWLEDGE YOUR SOURCES.
Please read http://www.student.cs.uwaterloo.ca/~cs341 for general instructions and policies.

Note: As it says on the web page, throughout the course “giving” an algorithm means (i) describe the main idea first, (ii) present clearly written pseudocode (e.g., at a level of detail mimicking the style of the lectures, the model solutions, or the textbook), (iii) give a correctness proof/argument if it is not immediately obvious, and (iv) include an analysis of the running time.

1. [10 marks] Divide and Conquer I. Suppose you have two lists of numbers in main memory, $A[1..n]$ and $B[1..n]$ where $A$ is in non-decreasing order (i.e., $A[i] \leq A[i+1]$) and $B$ is in non-increasing order. Give an algorithm to find, if it exists, an index $i$ such that $A[i] = B[i]$. (An $O(n)$ time algorithm gets no marks.) Your analysis of runtime should include a recurrence relation and a big-O expression.

2. [15 marks] Divide and Conquer II.

In the lectures you saw an algorithm to find non-dominated (or “maximal”) points from a given set of points in the plane. See Figure 1a. That algorithm began by sorting the points by $x$ coordinate and then splitting the sorted list into the first half and the second half.

Here is an alternative algorithm that does not start by sorting. If the points are $P = \{p_1, \ldots, p_n\}$ (in unsorted order), split them into $P_1 = \{p_1, \ldots, p_{\lfloor n/2 \rfloor}\}$ and $P_2 = \{p_{\lfloor n/2 \rfloor+1}, \ldots, p_n\}$, recurse on the two parts, and then combine the answers. The output should be a list of the non-dominated points in $x$ order.

(a) [3 marks] Fill in the details of how to do the combine step of this alternative algorithm and analyze its runtime. The input consists of two lists $A$ and $B$ each consisting of non-dominated points in $x$ order. The output should be a list of the non-dominated points of $A \cup B$ in $x$ order. Analyze the runtime in terms of $n_A = |A|$ and $n_B = |B|$.

(b) [2 marks] Analyze the overall runtime of the whole alternative algorithm. Your answer should include a recurrence relation and a big-O expression.

A more general problem is to find non-dominated boxes. The input is a set of boxes $B_1, \ldots, B_n$, where each box $B_i$ is given as a triple $(l_i, r_i, y_i)$ and consists of all the points $(x, y)$ with $l_i \leq x \leq r_i$ and $y \leq y_i$. Note that these boxes extend downwards forever. A box $B_i$ is non-dominated if there is some point in $B_i$ that is not in any $B_j, j \neq i$, or equivalently, if there is some exposed point on the top boundary of $B_i$ that is not in any $B_j, j \neq i$. The output should be the list of the corner points along the exposed top boundary. See Figure 1b.

(c) [10 marks] Give an $O(n \log n)$ time divide and conquer algorithm for the non-dominated boxes problem.
Challenge Question. This is for fun and enrichment only. Do not hand it in.

1. Suppose you are given a stack of $n$ pancakes of different sizes. You want to sort the pancakes so that smaller pancakes are on top of larger pancakes. The only operation you can perform is a flip—insert a spatula under the top $k$ pancakes, for some integer $k$ between 1 and $n$, and flip those top $k$ pancakes over.

Describe an algorithm to sort an arbitrary stack of $n$ pancakes using $O(n)$ flips. Exactly how many flips does your algorithm perform in the worst case? Prove that any algorithm must use $\Omega(n)$ flips in the worst case. (Bill Gates wrote a paper on this!)