CS 350 Spring 2011 HW 3—Greed and Division

1. Prove or disprove: The greedy change making algorithm (largest coin first) will still make change optimally (in the cases where it succeeds at all) even if the supply of each coin is limited (for example, we have only four quarters, four dimes, four nickles and four pennies). HINT: A disproof would probably be by example—give a supply of coins and a change-making instance that fails to be optimal. One possible approach to a proof would be to show that for any supply of coins one can make "synthetic" coins that are equivalent to the standard set.

2. Consider the following generalization of the famous "Dutch Flag Problem".

   RAINBOW FLAG

   Given: An array a of length n, with elements of the array in the range $1 \ldots m$. (Think of the elements of the array as being "color numbers".)

   Do: Sort the array so that all the 1s come first, then all the 2s, and so forth.

   (a) Give pseudocode for an in-place algorithm for solving this problem, with performance $O(m \cdot n)$. HINT: You have time for m passes over the array, although this is not enough time to sort it. (The cool people solve this with one pass, but this still involves $O(m \cdot n)$ work in the worst case. The really cool people do it with a two-pass algorithm that runs in $O(n)$ time but takes $O(m)$ space.)

   (b) Give an implementation of this algorithm in your favorite programming language. Test it on the array

   1322225331

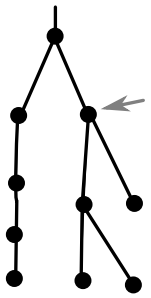
   and verify that it produces

   1122223335

3. Consider the following measure $I(t)$ of *rooted tree imbalance*

$$I(t) = \max_{t' \in \mathsf{subtrees}(t)} \begin{cases} \max\limits_{t_1, t_2 \in \mathsf{children}(t')} \mathsf{nodes}(t_1) - \mathsf{nodes}(t_2) & \text{when } |\mathsf{children}(t')| \geq 2 \\ 0 & \text{otherwise} \end{cases}$$

   That is, consider all possible subtrees t' of a tree rooted at t. The rooted tree imbalance $I(t)$ is the maximum imbalance of any t'. The imbalance of subtree t' is zero if t' has fewer than two children. Otherwise, the imbalance of subtree t' is the difference between the number of nodes in the tree rooted at a largest child t_1 (most nodes) of t' and the number of nodes in the tree rooted at a smallest child t_2 (least nodes) of t'.



   A tree with imbalance 2. The gray arrow indicates the subtree with maximal imbalance.

   (a) Give pseudocode for an algorithm for computing $I(t)$, with complexity $O(n)$ where n is the number of nodes in the tree. HINTS: You will want to divide-and-conquer here. Sorting is not a fast way to find the maximum or minimum of a set of numbers.

   (b) Give an implementation of your algorithm in your favorite programming language. Test it on some examples.

4. Consider the following problem, due to Prof. Jeff Erickson at University of Illinois Urbana-Champaign.

STABBING POINTS

Given: A set S of segments on the integral line. Each segment s has a left coordinate $l(s)$ and a right coordinate $r(s)$ with $l(s) < r(s)$.

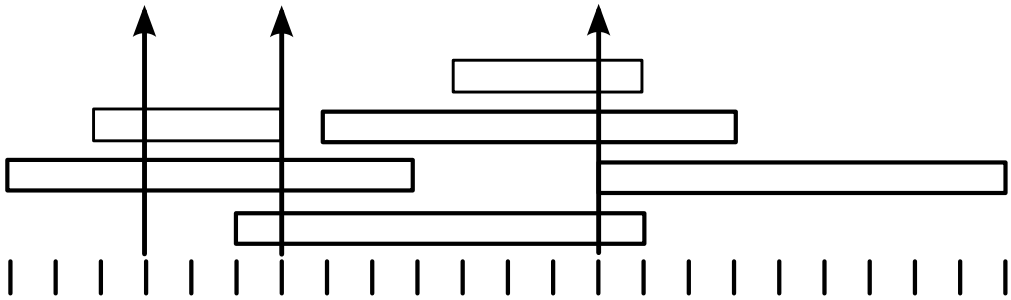
Find: A minimum-size set P of points on the real line such that every segment S is *stabbed* by some point in P. We say that a segment s is stabbed by a point p if p touches s, that is, if $l(s) \leq p \leq r(s)$.

For example, consider the set

$$S = \{[2, 10], [3, 7], [6, 15], [8, 17], [11, 15], [14, 23]\}$$

This set of segments can be stabbed by the points $p = 7$ and $p = 14$. So a minimum-size set of points stabbing S is

$$P = \{7, 14\}$$



Six segments stabbed by three points (not optimal).

Give pseudocode for a greedy algorithm for this problem that runs in time $O(|S| \log |S|)$. HINT: You might want to start by finding a point that will stab the "leftmost" segment, since that will have to be stabbed somehow. I guess that means sorting the segments somehow, which is probably where the time bound comes from. You might as well stab as many other segments as possible while stabbing this one. Once you've stabbed all of those, how can you proceed?