CSC265 Fall 2021 Homework Assignment 3

due Wednesday October, 2021

In Homework Assignment 1, you considered the problem of counting the number of (unordered) pairs of lines in $S \cup S'$ that intersect, where S and S' are two sets of disjoint line segments. Recall that a set of line segments is *disjoint* if no two line segments in the set intersect one another. You designed algorithms for solving two special cases of that problem.

In this homework, you will design an algorithm for solving the general case of this problem, given a nice representation for the line segments, but with the restriction that no line segments in $S \cup S'$ are vertical.

Let $X = \{s.\ell \mid s \in S \cup S'\} \cup \{s.r \mid s \in S \cup S'\}$ denote the set of x coordinates of the left and right end points of all the line segments in $S \cup S'$. Let $N = |X|$ and let $x_1 < x_2 < \cdots < x_N$ be the elements of X. Note that a line segment in S might have the same endpoint as a line segment in S'.

Consider a red-black tree T with $N - 1$ leaves. The i'th leaf v_i represents the interval $[x_i, x_{i+1}]$. An internal node v represents the union of the intervals represented by the leaves in the subtree rooted at v. Each node v of T has 10 fields:

- $v.colour$, a bit denoting its colour
- $v.left$, a pointer to its left child, which is nil if and only if v is a leaf,
- $v.right$, a pointer to its right child, which is nil if and only if v is a leaf,
- $v.p$, a pointer to its parent, which is nil if and only if v is the root of T,
- $v.\ell$, the left end point of the interval that v represents,
- $v.r$, the right end point of the interval that v represents,
- $v.S$, a pointer to a red-black tree whose nodes are a subset of the line segments $s \in S$ and whose keys are $s.b$,
- $v.S'$, a pointer to a red-black tree whose nodes are a subset of the line segments in $s' \in S'$ and whose keys are $s'.b$,
- $v.H$, (a pointer to) a doubly linked list whose nodes are the line segments in $u.S$, for all proper descendants u of v, and
- $v.H'$, (a pointer to) a doubly linked list whose nodes are the line segments in $u.S'$, for all proper descendants u of v.

Note that T is not exactly a binary search tree, since the nodes don't have keys.

A line segment $s \in S$ is in the tree pointed to by $v.S$ if and only if the projection of s onto the x axis contains the interval that v represents (i.e. $s.\ell \leq v.\ell$ and $v.r \leq s.r$) and, if v is not the root of T, does not contain the interval that $v.p$ represents. The set of line segments $s' \in S'$ in the tree pointed to by $v.S'$ is defined similarly.

1. Draw a picture of this data structure when $S = \{s_1, s_2\}$ and $S' = \{s'\}$, where
   $s_1.\ell = 1$, $s_1.r = 4$, $s_1.m = 0$, $s_1.b = 3$,
   $s_2.\ell = 5$, $s_2.r = 6$, $s_2.m = 0$, $s_2.b = 1$, and
   $s'.\ell = 2$, $s'.r = 6$, $s'.m = -1$, $s'.b = 6$.

2. Let $s \in S$. Give a matching upper and lower bound, to within a constant factor, on the maximum number of nodes v in T such that s is a node in $v.S$.
   Give a matching upper and lower bound, to within a constant factor, on the maximum number of nodes v in T such that s is a node in $v.H$.
   Give a good upper bound on the number of nodes the data structure, to within a constant factor. Justify your answers.

3. Describe an algorithm to compute the number of (unordered) pairs of lines in $S \cup S'$ that intersect in $O(N \log^2 N)$ time.

   You may use the two algorithms COUNT1 and COUNT2 from the solution to Homework 1 as subroutines. If you want to use a small modification of either of these algorithms, give the new specification and briefly describe (in at most a few lines) what modifications are needed.

   Explain why your algorithm is correct and why it runs in $O(N \log^2 N)$ time.

4. Give pseudocode for an algorithm that updates the data structure when a line segment s is added to the set S. You may assume, as preconditions, that $s \notin S$ and $S \cup \{s\}$ is a disjoint set of line segments. Explain why your algorithm is correct (i.e. the resulting data structure represents the sets $S \cup \{s\}$ and S' and it satisfies all the specified properties) and why it runs in $O(\log N)$ time.

5. Explain how to augment the data structure so that it is possible to delete a line segment s from S in $O(\log N)$ time. Explain, at a high level, how to delete a line segment s from S in $O(\log N)$ time in your augmented data structure. The time to compute the number of intersections should not increase and the time to perform insertion should remain $O(\log N)$.

   Assume, as a precondition, that $s \in S$. Explain why your algorithm is correct and why it runs in $O(\log N)$ time.