Intersection of a multiset of line segments with a quantified uncertainty

Let $\epsilon > 0$ be the user-defined criterion of small distance so that two points $p, q \in \mathbb{R}^2$ are considered to be equal:

$$p = q \iff \|p - q\|_2 \leq \epsilon.$$  \hspace{1cm} (1)

A line segment with endpoints $p_1$ and $p_2$, is the set of points

$$\overline{p_1p_2} = \{ p = rp_1 + (1 - r)p_2 : r \in [0, 1] \}.$$  \hspace{1cm} (2)

A point $q$ is said to be on a line segment $\overline{p_1p_2}$ iff $p = q$ holds for some point $p \in \overline{p_1p_2}$; in this case we write $q \in \overline{p_1p_2}$.

A set of $n > 3$ points are collinear iff any three out of these $n$ points are collinear. Two line segments $\overline{p_1p_2}$ and $\overline{q_1q_2}$ overlap iff the four endpoints are collinear and $\overline{p_1p_2} \cap \overline{q_1q_2}$ is also a line segment, which is called the overlie. In particular, two line segments $\overline{p_1p_2}$ and $\overline{q_1q_2}$ are identical iff there are only one pair of distinct endpoints out of the four.

In this homework we develop an algorithm as follows.

Input: (i) a multiset of $n$ line segments that may contain overlapping line segments;

(ii) the uncertainty quantifier $\epsilon > 0$ for two points to be equal in (1).

Precondition: The overlie of any two overlapping line segments does not overlap another line segment in the multiset.

Output: The set of intersection points and the associated intersecting segments (for each intersection point).

Postcondition: For any pair of overlapping segments, endpoints of the overlie are the only reported intersection points.

Questions

(a) Write an algorithm of which the signature is the same as above. Your algorithm should contain enough details so that step (c) is a straightforward translation of the description into C++.

(b) Prove that

(i) your algorithm finds all intersections and report only these intersections,

(ii) the running time of your algorithm is $O(n \log n + I \log n)$ where $I$ is the number of intersection points,

(iii) the space needed for your algorithm is at most $O(n + I)$.

(c) Implement your algorithm in C++.

(d) Test your implementation by a set of test cases. Your tests should at least contain those on Pages 21, 23, 26, 29. You should also test the cases where there are multiple overlapping, identical, and horizontal line segments.

Each of the four steps weigh 25 points, hence the total is 100 points.

Hints

• A good starting point is a brushup of the standard C++ library, after which you might find std::vector, std::set, and std::map very useful.

• As a key point in understanding Section 2.1, the order of the segments in the status tree $T$ does not change between two event points. Hence, the $y$-coordinate of the sweep line should be taken as the average of the two $y$-coordinates of the two event points.

• We do not require the optimization of the space from $O(n + I)$ to $O(n)$. But if your algorithm only takes $O(n)$ space, you will be given some extra credit.