Abstract

Problem Statement: Given $n$ line segments $s_i, i \leq n$, report all the pairwise intersections. Naive approach would be to take every pair of line and check its intersection. But its complexity would be $O(n^2)$. So we prefer to do it with line line sweep method which has complexity of $O((1 + 2n)\log n)$.

1 Line sweep method:

1. Take a vertical line left to the leftmost point.
2. Move it towards the segments and during this movement, we maintain the list of segments that intersects the vertical line.
3. Also the ordering of intersections is maintained. Ordering in the sense that a particular line segment is above / below a line segment.
4. The moment the vertical line crosses the intersection point the ordering is changed. The lines which are swapped must be consecutive as those the only one that intersects.
5. The moment any two lines become consecutive they can potentially intersect.

2 lemma

Given two segment $S_i$ and $s_j$, which interest in a single point $p$ (and assuming no other line segment passes through this point) there is a placement of the line sweep prior to this event, such that $s_i$ and $s_j$ are adjacent along the sweep line (and hence will be tested for intersection.)

2.1 Events we track

1. Intersections can flip ordering.
2. Some line segments can begin.
3. Some line segments can end.
2.2 Data structures for sweep line

To store the sweep line status, we maintain a balanced binary tree or a skip list whose entries are pointers to the line segment, store in decreasing order of y coordinate along the current sweep line. Since the sweep line varies, we need 'variable' key.

To do this, the key value in each node of the tree is a pointer to a line segment. To compute the y coordinate of some segment at the location of the current sweep line, we simply take the current x coordinate of the sweep line and plug it into the line equation for this line.

The operations that we need to support are those of deleting a line segment, inserting a line segment, swapping the position of two line segments and determining the immediate predecessor and successor of any item. Assuming any balanced binary tree or a skip list, these operations can be performed in $O(\log n)$ time each.

2.3 complexity

By ordering endpoints we can know events 2 and 3 in priority. We maintain an event queue ordered on x coordinate. Before intersection, 2 lines must become consecutive and this happens only when an event occurs. If two lines become consecutive, we check for an intersection and add the intersection if it exits to the event queue. So the overall complexity is $O((I + 2n)\log n)$ where $I$ is the number of intersection points, $2n$ is the ordering of end points of $n$ segments. And $\log n$ is there due to heap data structures for events ordering on the sweep line.
Space complexity is $O(n + I)$. But sometimes $I$ is as large as $O(n^*n)$. 