Solutions for homework #1: Arrangements, zones, straight and topological sweep

- **The Common Theory Problems**

Problem 1. [10 points]

In an arrangement $\mathcal{A}$ of n lines in the plane, a single face can have at most n sides. Prove that any m distinct faces can have at most $n + 4\binom{m}{2}$ sides altogether. [[This bound is best possible if $4\binom{m}{2} \leq n$ and is known as *Canham's Lemma*; it implies, for instance, that any $\sqrt{n}$ faces can have a total of only $O(n)$ sides altogether.]]

Solution

We first prove a lemma that bounds the number of times a line can be part of two different faces.

Claim 1. *Let F_1 and F_2 be any two faces in an arrangement of n lines. We claim F_1 and F_2 share at most four lines.*

Proof: If a line ℓ is shared by two faces, it is tangent to both faces (a line is said tangent to a convex face if it intersects the boundary but not the interior of the face). The above statement then follows from the well-known fact that any two interior disjoint convex objects can have at most four bi-tangent lines. □

Now let $\mathcal{F}$ be a set of m distinct faces, $F_1, F_2, \ldots, F_m$, of the arrangement. For a line ℓ, consider the edges on ℓ. Suppose that they are $e_{i_1}, \ldots, e_{i_s}$, where e_{i_j} is on face F_{i_j}. We now charge each edge as follows. For $j < s$, charge e_{i_j} to the pair $(F_{i_j}, F_{i_{j+1}})$; charge e_{i_s} to the line ℓ. It is clear that each edge gets charged to something. For each line, it can be charged at most once. For each pair of faces, it can be charged at most four times according to the above claim. Thus, there are at most $n + 4\binom{m}{2}$ edges.

Remark The above bound is not tight. The tight bound is $\Theta(m^{2/3}n^{2/3} + n)$ (refer to Handouts #11 in the course reader). □

Problem 2. [10 points]

We saw in class that, in an arrangement $\mathcal{A}$ of n lines in the plane, the horizon or zone of another line ℓ has combinatorial complexity $O(n)$. Given as input only the n lines of the arrangement and ℓ (say by their equations), show how to compute all the faces of $\mathcal{A}$ comprising the zone of ℓ, in linear space and $O(n \log n)$ time.

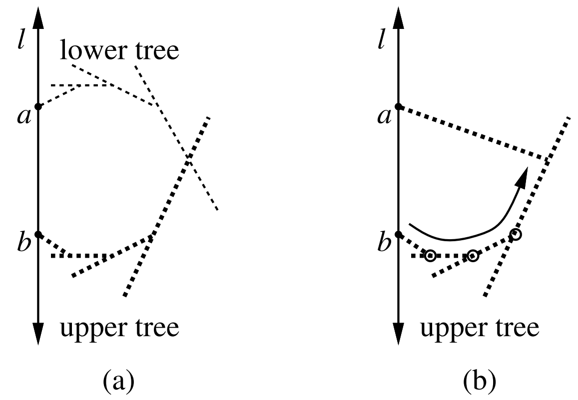

Figure 1: The face between a and b is the intersection of their bays (a); inserting the line from a can be charged to the circled points of the lower bay (b).

Solution

We suppose line ℓ is vertical, and we want to compute the zone of all faces on the right side of ℓ. We begin by computing the $n-1$ intersection points of the other lines with ℓ, and sort them by y-coordinate along ℓ in $O(n \lg n)$ time. The remainder of this algorithm will run in $O(n)$ time.

We will compute the upper and lower horizon trees of lines to the right of ℓ. Given the upper and lower horizon trees, we claim that each face adjacent to ℓ may be traversed in time linear in its size. If we can show this, then it will follow from the horizon theorem that we can walk all the faces in $O(n)$ time.

To prove the claim, consider any two consecutive points a and b along ℓ with a above b, as in figure 1(a). Then the face between a and b is the intersection of the lower horizon tree bay of a and the upper horizon tree bay of b. To construct the face, we need to walk up the bays from a and b, being careful not to walk too far in either bay. A sufficient rule is to visit the points of the two bays in order of increasing x-coordinate; on each step we need to check if we have crossed the line from the other bay.

Now it suffices to show how to construct the horizon trees in linear time; we will describe the method for the upper horizon tree. We construct the tree by inserting the lines in increasing order of their *intersections* with ℓ. Suppose we have just inserted the line from point b, and we now want to insert the line from the next higher point a (figure 1(b)). We simply walk up the bay from b, checking whether the line from a intersects each edge. Then the time to do insert this line is $O(1)$ plus the number of vertices passed over in the bay from b. Since such a vertex will thereafter be hidden from above by the line from a, it will never again be charged, and hence the total charges and running time are $O(n)$.

It is important that we insert the lines in order of increasing intersection with ℓ: if we try to insert the lines in increasing or decreasing slope order then there are example arrangements that will take time $\Theta(n^2)$. However, you can verify that those two different

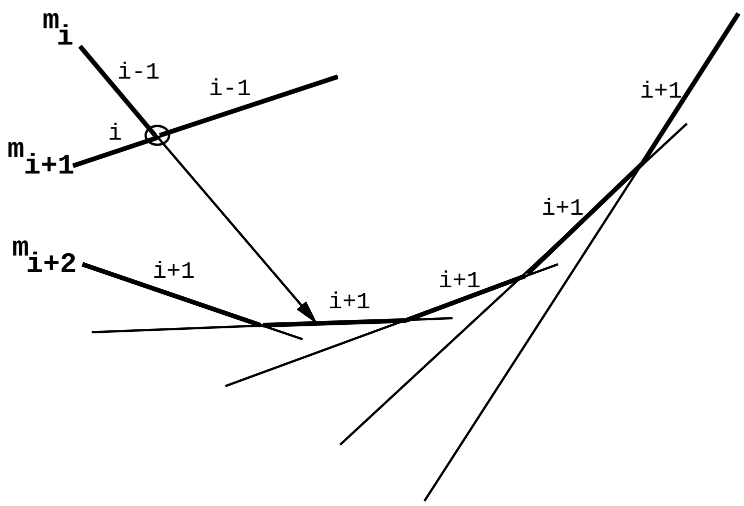

Figure 2: Searching for an intersection clockwise: the weights are specified for each edge of the UHT before ES is performed.

orderings yield the same tree.

Problem 3. [**10 points**]

Show that the topological sweep that computes the arrangement of n lines in the plane can be carried through within the same time and space bounds even if the search for an intersection when propagating a line into a bay of the upper horizon tree is done by traversing the bay *clockwise* instead, starting from the other (shortened) edge of the elementary step.

Solution

We follow the notation and the analysis outlined in the paper "Topologically Sweeping an Arrangement" by Edelsbrunner and Guibas (page 11).

Number the bays associated with a cut from 0 to $n-1$, where the ith bay is bounded above by the line m_i of the cut. The bay numbered 0 is not bounded above at all. We define the weight $w(e)$ of an edge e bounding a bay from below as the index of the bay immediately above it. Let $w(U)$ be the weight of the upper horizon tree U, defined as the sum of its edge weights. That is,

$$w(U) = \sum_{e \epsilon U} w(e).$$

The upper horizon tree associated with the leftmost cut has weight $O(n^2)$. (We proved that it can be built in linear time, so it must have $O(n)$ edges, and each edge can have a weight at most $n-1$.)

Let's consider what happens to the weight of the UHT when an elementary step is performed at the intersection of lines m_i and m_{i+1} of the cut.

1. The two edges of the processed elementary step are deleted. Their weights are $i-1$ and i, since they belong to lines m_i and m_{i+1} of the cut.
2. Each edge in the clockwise bay traversal whose intersection test fails becomes part of a lower numbered bay. In fact, its weight is reduced from $i+1$ to i.
3. Two new edges of weight i are created. One is the lower edge from the elementary step that is extended to meet the bay, and the other is created by splitting the edge of the bay whose intersection test succeeded.

Hence, the net change in $w(U)$ due to an elementary step is $1-k$, where k is the number of failed intersection tests. That is, each elementary step increases the weight by one, while each failed intersection decreases the weight by one.

This allows us to bound the total number of failed intersections. The initial UHT weight is $O(n^2)$, and the total increase due to elementary steps is $O(n^2)$ as well. Clearly the weight can never be negative, so total decrease due to failed intersections must also be $O(n^2)$. All the other work for an elementary step takes constant time, so the total running time is $O(n^2)$.

- **The Additional Theory Problems**

Problem 4. [20 points]

Suppose that we modify the straight-line sweep method for computing a line arrangement so that, when processing an event, the future events corresponding to intersections for the two newly created adjacencies are added to the priority queue, but the events corresponding to the two adjacencies just destroyed are *not* removed. This will still give a correct algorithm, but now the priority queue size may increase, as each event adds possibly two new adjacencies but removes only one. Prove that, given any four lines a, b, x, y in descending slope order, not all three intersections ax, ay, by can be events present in the priority queue at once. Use this fact to argue that maximum size of the priority queue now is at most $O(n \log n)$. Give an example showing that this bound is in fact attainable. (Partial credit will be given for any subquadratic upper bound.)

Solution

(a) Consider the first event that involves a pair of two of the lines among a, b, x, y. Since their vertical sequence (bottom to top) initially is a, b, x, y, there are three cases:

1. Lines a, b intersect first. The vertical sequence now is b, a, x, y. For by to be in the queue, b and y have to become adjacent, which means that y has to pass above a, as a and b cannot intersect twice. However, this means that ay, by cannot be in the queue at the same time.

2. Lines b, x intersect first, resulting in the sequence a, x, b, y. Then for ay to be in the queue, either a has to pass x, or b has to pass y. That is, either by or ax cannot remain in the queue.

3. Lines x, y intersect first. Similar analysis to the case 1 shows that ax, ay cannot be in the queue at the same time.

The above analysis covers all the cases. Thus, ax, ay, by cannot be in the priority queue at the same time.

(b) Sort the lines by decreasing slopes, and divide them into two groups at the median slope. Call the group with the highest slopes "red" and the other group "blue". At a given position of the sweep line, the intersections present in the queue are of three types: red-red, blue-blue, or red-blue. Denote by $T(n)$ the worst case number of intersections in the event queue. If we can prove that the number of bichromatic intersections in the queue is linear, then we can obtain that $T(n) \leq 2T(n/2) + O(n)$, which will give us the $O(n \log n)$ bound.

Let us name the red lines $a_1, a_2, \cdots, a_{n/2}$, and the blue lines $b_1, b_2, \cdots, b_{n/2}$, both in decreasing slope order. Consider the priority queue at position t, and build an $n/2 \times n/2$ matrix M by setting the entry m_{ij} in the matrix to 1 if the intersection $a_i b_j$ is in the queue, and 0 otherwise.

Charge all the non-zero elements as follows: if m_{ij} is the first, i.e. the one with the smallest column index, non-zero element in the i-th row, charge it to line a_i; otherwise charge it to line b_j. Obviously, each non-zero element gets charged somewhere, and each red line is charged at most once. We shall show that each blue line gets charged at most once as well.

Otherwise, suppose the blue line b_j is charged twice by m_{ij} and m_{kj}, where $i < k$. Since m_{ij} is charged to the blue line b_j, there must be a non-zero element, say $m_{ij'}$, having smaller column index than m_{ij}. Now, consider the lines $a_i, a_k, b_{j'}, b_j$. They are in descending slope order, and the intersections $a_i b_{j'}, a_k b_j, a_i b_j$ are in the queue at the same time, which contradicts (a). Thus, no blue line can be charged more than once. That is, the number of non-zero elements in the matrix is at most $2(n/2) = n$.

To give a construction for the lower bound, we use a recursive construction: we will build collections of lines $\{S_r\}_{r \geq 1}$, so that S_r has 2^r lines and gives rise to a priority queue of size $K_r \geq r2^{r-1}$.

S_1 is just two lines intersecting to the right of the sweep line s. For this case $K_1 = 1$, as required. Now given that we have already built S_r, we construct S_{r+1} as follows. We start by taking two copies of S_r, say S_r^1 and S_r^2, where S_r^1 is just S_r, and S_r^2 is S_r translated slightly upwards (we will destroy the parallel pairs in a moment). Now we "shear" upwards the lines in S_r^2 at left infinity so that all intersections between the lines in S_r^1 and those in S_r^2 that are to the left of s actually occur to the right of any intersections within S_r^1 (and therefore also S_r^2). See figure 3. This guarantees that any pairs of intersecting lines from within S_r^1 or S_r^2 that gave rise to an event in the priority

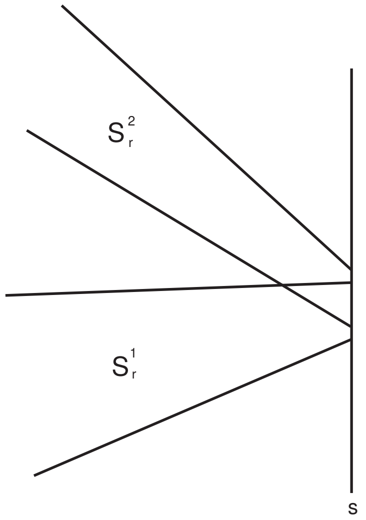

Figure 3: The $\Omega(n \log n)$ construction.

queue before will still do so in the combined configuration. But now in addition we'll have the 2^r intersections between each line in S_r^1 and its partner in S_r^2n that lies immediately above it along s. Thus $K_{r+1} \geq 2K_r + 2^r \geq (r+1) \cdot 2^r$, as asserted.

Remark For additional details and extensions of these results to the case of line segments, we refer the reader to [1]. We note that the proof just given is simpler than the original one in [1].

Problem 5. [5 points]

Show that the topological sweep that computes the arrangement of n lines in the plane can be carried through within the same time and space bounds, even when the topological line is required to proceed *vertically* through each region and can only move horizontally by following arrangement edges. In other words, the topological line should consist entirely of vertical segments crossing faces and portions of arrangement edges. This variant can be used to compute the threads needed for a vertical decomposition of the arrangement.

Solution

It suffices to insure that for each pair of consecutive cut edges, their projections onto the x-axis have some point in common. This is certainly true of the initial leftmost cut (because all the cut edges extend to $-\infty$ on the x-axis), so we simply need to preserve this property on each step.

We call an elementary step "safe" it it preserves this property; see figure 4. Note that the property is local; i.e. given a pair of consecutive cut edges we may check if they form

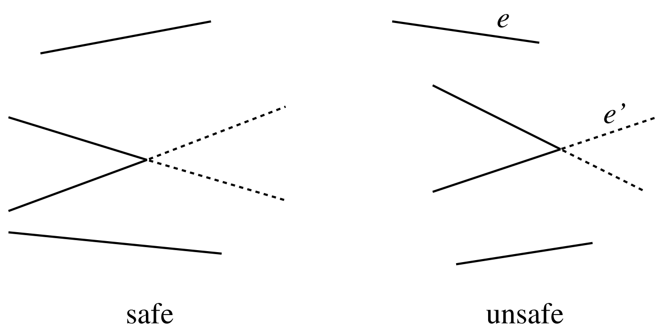

Figure 4: Examples of safe and unsafe elementary steps. The unsafe example is unsafe because there is no vertical line through both e and e'.

a safe elementary step in constant time, by looking at the cut edges immediately above and below.

Our strategy will be to maintain a queue of all the safe elementary steps. After each safe elementary step, at most four nearby elementary steps may be affected (i.e. they may become safe elementary steps and should be added to the queue); so we can update the queue in constant time.

We use the upper and lower horizon tree structures just as before, so this algorithm will have $O(n^2)$ running time if we can show it terminates. All we need to show is that there always exists a safe elementary step until we reach the rightmost cut.

Claim 1. *Consider any non-rightmost set of cut edges, such that currently every consecutive pair of cut edges overlap in their x coordinates. Then there exists a safe elementary step.*

Proof: Since this is not the rightmost cut, at least one of the cut edges has a right endpoint. So let v be the leftmost right endpoint, we claim that there is a safe elementary step at v. We have argued previously that v is an elementary step (see Lemma 2.2 of the paper by Guibas and Edelsbrunner); we need to argue that it is safe.

Consider the cut edge e immediately above the cut edges of v. Since v is leftmost of all right endpoints, e extends to the right of v. Thus after the elementary step at v, edge e will still overlap with the new edge below it, because they overlap at v. A similar argument applies to the edge below v, and clearly the two new edges out of v will also overlap.

Since these are the only pairs of edges changed by the elementary step at v, the elementary step at v is safe. $\square$

Problem 6. [15 points]

Show how to implement the topological sweep we discussed in class using only a *single* horizon tree, say the upper horizon tree. The crucial step is to discover efficiently a vertex of the tree where an elementary step can be carried out.

Solution

Say that line b is the parent of line a if b is the line of higher slope that cuts off a in the upper horizon tree; similarly say a is a child of b. Note a line may have many children but at most one parent. The lines are ordered by their cut edges; say a is immediately above b if the cut edge of a is immediately above the cut edge of b.

Claim 1. *If line b has any children, then the line a immediately above b is a child of b.*

Proof: If a had a higher slope than b, then it would cut off all higher lines before b, and b would have no children. Thus a must have a lower slope than b, and a will be a child of b unless some other line c of higher slope cuts off a before it reaches b. But this c would also cut b, and again b would have no children. $\square$

Thus for every parent b there is a child a such that a, b are consecutive. Call this a *child-parent pair*.

Claim 2. *The uppermost child-parent pair (a, b) is an elementary step.*

Proof: Suppose not; then there is some other line c that intersects either a or b before their intersection point. Line c cannot come from below b, because then c would cut off a before b, and b would not be the parent of a.

Thus c comes from above a. Let d be the parent of c; it could be that $d = a$ or d is some other line, but it is certain that d is above b. Since d has a child, then by the previous lemma d must be part of a child-parent pair above (a, b), which contradicts the choice of (a, b). $\square$

Clearly we can test in constant time whether two adjacent lines of the cut are a child-parent pair in the UHT. Also, after an elementary step between lines m_i and m_{i+1}, the only possible new child-parent pairs are (m_{i-1}, m_i) and (m_{i+1}, m_{i+2}) (where the m_j are labeled after the ES).

This leads us to the following algorithm. We remember the index j of the pair (m_j, m_{j+1}) where the last ES occurred (initially zero). After each update to the UHT, we back up to the previous pair (m_{j-1}, m_j), and then scan downwards looking for a child-parent pair. Clearly the first such pair found will be the uppermost.

We must now show that this algorithm still runs in $O(n^2)$ time. Each ES decrements the index j by one, and then increments it by one for each unsuccessful child-parent test. But we always have $j \leq n$, thus the total number of unsuccessful tests is $O(n^2) + n = O(n^2)$. The number of successful tests is also $O(n^2)$, so we are done.

References

[1] J. Pach and M. Sharir, "On vertical visibility in arrangements of segments and the queue size in the Bentley-Ottmann line sweeping algorithm", *SIAM Journal on Computing*, vol. 25, no. 3, pp. 460–470, (1991).