Main topics are line_arrangements and zone_theorem.

Prelude

- Homework 2 was due today. Please submit on eLearning ASAP if you haven’t already.

Line Arrangements

- Let L be a finite set of lines in the plane.
- The lines subdivide the plane into a cell complex or planar subdivision called the arrangement of L, denote A(L).
- Intersection points are the vertices, segments between intersection points form the edges, and and polygonal regions between lines form faces.
- Like the Voronoi diagram, there are unbounded faces. And like the Voronoi diagram, the easiest way to deal with them is to add a vertex point at infinity and make it the endpoint of all the unbounded edges.
- After doing that, you get a proper embedded planar graph. You can store it as a DCEL.

- Today, we’ll discuss some basic combinatorial properties of line arrangements and how to construct them efficiently. Tuesday, I’ll show you a couple applications of line arrangements which take advantage of point-line duality like we saw earlier in the semester.

Combinatorial Properties

- The combinatorial complexity of an arrangement is the total number of vertices, edges, and faces.
- The arrangement is simple if no three lines intersect at a common point, which is guaranteed by our normal general position assumption for lines.
- Let’s also assume no two lines are parallel.
- With those two assumptions, we can prove the following exact bounds. They only get smaller if the assumptions aren’t true.
- Lemma: Let A(L) be a simple arrangement of n lines L in the plane. Then:
• there are \(\binom{n}{2}\) vertices
• there are \(n^2\) edges
• there are \(\binom{n}{2} + n + 1\) faces

Proof:
• Every pair of lines intersects at exactly one point.
• By induction:
  • One line consists of one unbounded edge.
  • Given \(n\) lines, remove one of them to get an arrangement with \((n-1)^2\) edges.
  • Putting that line back splits the others into \(n - 1\) new edges, and the new line is split into \(n\) new edges total.
  • \((n-1)^2 + (n-1) + n = n^2\).
• By Euler’s formula for planar graphs, \(v - e + f = 2\) where \(v\), \(e\), and \(f\) are the number of vertices, edges, and faces, respectively.
  • \(v = \binom{n}{2} + 1\) and \(e = n^2\).
  • \(f = 2 - v + e = 2 - (1 - \binom{n}{2}) + n^2 = 2 - (1 + n(n-1)/2) + n^2 = 1 + n^2 / 2 + n / 2 = 1 + n(n-1)/2 + n = \binom{n}{2} + n + 1\).

As an aside, similar properties hold in higher dimensions. There, we would consider arrangements of hyperplanes that make a polyhedral cell complex where \(d\) hyperplanes make a vertex, \(d - 1\) make an edge, \(d - 2\) make a face, and so on. The combinatorial complexity is \(\Theta(n^d)\).

Incremental Construction

• If we’re going to use line arrangements, we should probably figure out how to construct them.
• We’ll use another incremental algorithm. However, this one is not randomized.
• The main reason we get away with a deterministic algorithm is that we can afford an \(O(n^2)\) time construction, since the arrangement has that complexity.
• Let \(L = \{ell_1, \ldots, ell_n\}\). We’ll add each line \(ell_i\) in \(O(i)\) time for a total running time of \(O(n^2)\).
• Let \(L_i = \{ell_1, \ldots, ell_i\}\) and \(A(L_i)\) be the arrangement of the first \(i\) lines.
• Say it’s time to insert line \(ell_i\). We first find the leftmost (unbounded) face of \(A(L_{i-1})\) containing the line. To do that, observe that the lines at \(x = -\infty\) are sorted top to bottom by increasing order of their slopes. We just compare the slope of \(ell_i\) to all others in \(O(i)\) time to find where it falls in the order.
• \(ell_i\) cuts through a sequence of \(i - 1\) edges from the other lines. We need to figure out which ones they are. Once we do so, we can split them and update the DCEL in \(O(1)\) time per edge we cut.
• The surprising thing is that we don’t need to do anything particularly clever to find the
edges we’re going to split.

- What we’ll do is iteratively walk along each of the faces that our new line cuts through. We can walk around a face in \( O(1) \) time per edge using the DCEL.
  - Say we figure out where \( \ell_i \) enters an edge on the left side of a face.
  - We walk counterclockwise along the face’s edges until we find the other edge that intersects \( \ell_i \).
  - Then we jump to the other side of that edge to walk along the next face.
  - See (a) below.

- Everything that isn’t the walk takes \( O(i) \) time. So how long do we spend walking along the faces?
  - Naively, we split \( i - 1 \) edges, so we pass through \( i \) faces.
  - Each face is bounded by at most \( i \) lines, so a single face traversal takes \( O(i) \) time.
  - But together that implies \( O(i^2) \) time for all the face traversals. We need a better argument.

### Zone Theorem

- Let \( L \) be a set of \( n \) lines and \( A \) be their arrangement.
- Let \( \ell \) be any line outside of \( L \). The zone of \( \ell \) in \( A \), denoted \( Z_A(\ell) \), is the set of faces in \( A \) intersected by \( \ell \). See (b) above.
- If we get a good bound on the total complexity/number of edges in the zone for a the line \( \ell_i \), then we learn how long those walks take.
- Zone Theorem: The total number of edges in all faces of the zone \( Z_A(\ell) \) is at most \( 6n \).
- So, if we’re inserting \( \ell_i \) into an arrangement of \( i - 1 \) lines, we only walk around \( O(i) \) edges.

**Proof:**

- For simplicity, let’s rotate the plane so \( \ell \) is horizontal.
- We’ll assume none of the \( n \) lines are parallel to \( \ell \).
- Split the edges into two groups. First are left bounding edges for which an incident zone face lies in their right halfplane (so they bound the left side of the face). There’s also right bounding edges.
- We’ll prove there are at most \( 3n \) left bounding edges and \( 3n \) right bounding edges
  - Edges crossed by \( \ell \) are both left and right bounding, so we’re overcounting by a bit.
We’ll proceed using induction.

- For \( n = 1 \), there’s exactly one left bounding edge (the whole line of \( L \)) and \( 1 \leq 3 = 3n \).
- For higher \( n \), consider the rightmost line of \( A(L) \) to intersection \( ell \). Call it \( ell_1 \). Let \( L' \) be the other \( n - 1 \) lines of \( L \), and let \( A' \) be their arrangement.

Inductively, there are at most \( 3(n - 1) \) left bounding edges in \( Z_{A'}(ell) \).

But what if we add back \( ell_1 \)?

- \( ell_1 \) intersects \( ell \) within the rightmost face of \( Z_{A'}(ell) \).
- All the edges of the rightmost face are left bounding edges.
- \( ell_1 \) contains a brand new left bounding edge, and it splits \( e_a \) and \( e_b \) into two left bounding edges each for a net increase in 3 left bounding edges.
- Are there any other new left bounding edges?
- Any left bounding edges from \( ell_1 \) lying above \( e_a \) lie in the region bounded by \( ell_1 \) and \( e_a \)’s line. But that whole region lies above the zone.
- You can say a similar thing for trying to find new edges below \( e_b \).

**Uses and a Caveat**

- Building an arrangement like this is useful for a variety of applications, some of which also use point-line duality.
- For some of these examples, you need to remove the general position assumptions on the lines, but the details aren’t too hard. Again, the textbook has the details. Here are a few example problems you can solve easily in \( O(n^2) \) time using line arrangements, without going into details.
  - General position test: Given a set of \( n \) points in the plane, determine whether any are collinear.
  - Minimum area triangle: Given a set of \( n \) points in the plane, determine the minimum area triangle with vertices selected from the points.
  - Visibility graph: Given line segments in the plane, two points are visible if the interior of the line segment joining them intersects none of the segments. Given \( n \) non-intersecting line segments, compute the visibility graph which has endpoints for vertices and an edge between pairs of endpoints that are visible to one another.
- Maximum stabbing line: Given $n$ line segments in the plane, compute the line $ell$ that stabs the maximum number of line segments.
- Ham sandwich cut: Given $n$ red points and $m$ blue points, find a single line $ell$ that simultaneously bisects both point sets.
  - This is always possible, no matter how the points are arranged.
  - In fact, given $d$ sets of colored points in $\mathbb{R}^d$, we can use a single $d-1$ dimensional hyperplanes to bisect every color set.
  - In other words, we can cut a sandwich with bread, ham, and cheese in half with a single chop no matter how the ingredients are arranged.
- Unfortunately, the arrangement based algorithms also require $O(n^2)$ space to actually store the arrangement, which is fine for something like visibility graph that has that output size anyway. Not so great a general position test.
- Next Tuesday, we'll briefly discuss a strategy to avoid the space usage for some of these problems at the cost of an extra $O(\log n)$ in running time.
- We'll also go over some applications of planar arrangements and point-line duality in more detail.