Trapezoidation Decomposition

Input:N= Set of line segments in plane.

Output: $H(N)$= Trapezoidal Decomposition (i.e. a planar graph).

Technique: PLANE SWEEP

Basic Idea

- Scan the segments *using a vertical line* from the left to the right.

- Keep track of endpoints and intersections.

- Invariant: Trapezoidal diagram on the left of the scan line.

- Think in terms of time, i.e., scan line at time t.

- $H_t(N)$ refers to the trapezoidal diagram to the left of the scan line at time t.

- Aim : To compute $H_\infty(t) = H(N)$.

Critical Points

- Time advancement is done only at the critical points.

- *Critical points*: Endpoints and intersection points of segments.

- We need to predict the next critical points.

- Endpoints are easy.

- What about the intersection points.

Frontiers

Frontier List: List of segments in N intersecting the scan line.

- The segments in the frontier list can be linearly ordered w.r.t. the y-direction.

Observation: When the scan line is about to pass an intersection $v = a \cap b$, the segments a and b must be consecutive in the linear order on the frontier list.

We maintain the frontier throughout the scanning process.

Event Schedule

Initially: Endpoints of all segments ordered w.r.t x-coordinate.

Intermediate Step:
(a) All endpoints to the right of the scan line.
(b) Each intersection point to the right of the scan line that is contained in two consecutive segments of the frontier.
(c) Some additional intersection points to the right of scan line.

What is the next event: The point with the least x-coordinate in the event schedule.

Algorithm maintains the following

1. A partially constructed partition $H_t(n)$.

2. The vertically ordered frontier list of segments intersecting the current scan line.

3. The event schedule. (Also sort them w.r.t. x-coordinate. For each point in the event schedule we maintain the pointers to the respective segments.)

Search, *Delete*, *Add* in the frontier list/event schedule can be done in $O(\log n)$ time.

Algorithm

Initialization:
a) $H(N)$ and frontier list are empty.
b) Event schedule contains all the endpoints of the segments ordered w.r.t. x-coordinate.

While event-schedule is not empty do
Begin
Let p be the next point in the event schedule.
p is either (1) Left endpoint (2) Right endpoint (3) Intersection point.
Perform the respective steps.
End.

Left Endpoint

Let p be the end point of a segment e

1. Add e to the frontier list.

2. Let a and b be the segments just above and below e in the frontier list. (Insert e between a and b.)
   If a and e (similarly, e and b) intersects to the right of the scan line, add their intersection point in the event queue.

3. Update H at its right boundary by adding the segment e and the vertical attachment through p spanning between a and b.

Right Endpoint

Let p be the end point of a segment e

1. Delete e to the frontier list.

2. Let a and b be the segments just above and below e in the frontier list.
   Update H at its right boundary by adding the vertical attachment through p spanning between a and b.
   If a and b intersects to the right of the scan line, add their intersection point in the event queue.

Intersection point

Suppose p is the intersection point of two segment f and g

1. Switch orders of f and g in the frontier list.

2. Let f be above g after the switch in the frontier list.
   Let a and b be the segments above f and below g, respectively.
   Update the partition H at its right boundary so as to indicate that f and g intersect at p.
   Also add the vertical attachment through p spanning between a and b
   If a and f (similarly, b and g) intersect to the right of the scan line, then ass their intersection point in the event queue.

Analysis

1. Sorting of the endpoints: $O(n \log n)$ time.

2. Main body executes once for each critical point.

   (a) $O(1)$ updates and searches in frontier/event schedule: $O(\log n)$ time.

   (b) Number of critical points $= (2n + k)$; where k is the number of intersection points.

Total Running Time: $O((n + k) \log n)$

Theorem

(Bentley-Ottman, 1979)

The trapezoidal decomposition formed by n segments in the plane can be computed in $O((n+k)\log n)$ time, where k is the number of intersections. In particular, all intersections of the given n segments can be computed in $O((k+n)\log n)$ time.

Much More

1. Best known is $O(n \log n + k)$ time (deterministic).
   (sweep, segment trees, amortization arguments, uses $O(n+k)$ space)
   (Chazelle and Edelsbrunner, 1988.)

2. $O(n \log n + k)$ randomized - simple and practical (see later in the course)

3. Computes number of intersections in $O(n^{1.695} \log n)$ time using $O(n)$ storage (Chazelle, 1983)

4. If segments are non-intersecting!

5. If segments form a simple polygon!

Voronoi Diagram

Input:$N = \{p_1, \cdots, p_n\}$= Set of points in the plane.

Output: $H(N)$= Voronoi Diagram.

Voronoi Polygon: $Vor(p_i) = \{q \| p_i - q| \leq |p_j - q|, \forall j \neq i\}$

Applications: Nearest neighbor queries, computational morphology, facility location, path planning.

Geometric Properties of Voronoi Diagrams

Half-Plane formulation: $Vor(p_i) = \cap_{j \neq i} H(p_i, p_j)$, where $H(p_i, p_j)$ represent the half space formed by the bisector of p_i and p_j, containing p_i.

$Vor(p_i)$ are convex.

Voronoi Vertices: Equidistant from three points, or where three Voronoi cell intersect.
It is the center of the circle passing through these three points.

Convex Hull: A cell of the Voronoi diagram is unbounded if and only if the corresponding site lies on the convex hull. (a point is on the convex hull, if and only if, it is the closest point from some point at infinity)

Delaunay Triangulation

Assume points in general position, no 4 cocircular ...

- Each Voronoi vertex is incident to 3 edges of the Voronoi diagram.
- Voronoi Diagram is a planar graph.
- Consider its dual graph, i.e., connect two sites if and only if they share an edge in the Voronoi Diagram.
- Faces of the resulting Dual Graph will be Triangles (ignore the external face!)
- This triangulation is termed as Delaunay Triangulation.

Properties of Delaunay Triangulation

Convex Hull: The exterior face of DT is the convex hull of the point set.

Circumcircle Property: The circumcircle of any triangle in DT is empty.

Empty circle property: Two sites p_i and p_j are connected by a DT edge if and only if there is an empty circle passing through these two points. (Note that the center of this circle passes through a Voronoi edge, since the center is equidistant from both the points)

Closest pair property: The closest pair of sites in P are Delaunay neighbors.

More Properties of Delaunay Triangulation

Max-Min Angle: Among all triangulations of the point set, DT maximizes the minimum angle. (only for 2D)

Minimum Spanning Tree Property: MST of the point set is a subgraph of DT.

$MST \subseteq RNG \subseteq GG \subseteq DT$

Algorithmic Side of Voronoi Diagrams

1. Divide-and-Conquer: $O(n \log n)$ but hard to implement.

2. Randomized Incremental': $O(n \log n)$ w.h.p. and simple to implement.

3. Fortune's plane sweep: $O(n \log n)$ clever but simple to implement.

4. Reduction to Convex Hull Problem in a higher dimension.

Divide-and-Conquer Algorithm

Main Steps:

1. Sort the points w.r.t. x-coordinate.

2. Split the point set by a vertical line into two subsets L and R of equal size.

3. Recursively, construct the Voronoi Diagram of L and R.

4. Merge the two diagrams into a single diagram, by computing the contour and discarding the portion of $VD(L)$ that is to the right of the contour, and the portion of $VD(R)$ that is to the left of the contour.

Analysis

Merging in Linear Time

$$T(n) = 2T(\frac{n}{2}) + O(n)$$

$$T(1) = O(1)$$

$$T(n) = O(n \log n)$$

How to Merge in Linear Time

Idea:

Let $V(L)$ and $V(R)$ represent the Voronoi Diagram of the sets L and R.

Note that L and R are separated by a vertical line.

Compute the CONTOUR (or Dividing Chain) and discard the portion of $V(L)$ to the right of the contour line.
Similarly, discard the portion of $V(R)$ to the left of the contour.

What is the Contour

Lemma: The contour consists of a single polygonal curve, which is monotone with respect to the y-axis.

Proof:

- The contour separates the plane into two regions, those points whose nearest neighbor lies in L from those points whose nearest neighbor lies in R.
- Locally the contour consists of points that are equidistant from 2 points, one from L and other from R.
- Contour is formed from pieces that are perpendicular bisectors, with one point in L and one in R.

- Piecewise it is a polygonal curve.

- No 4-points CO-circular, each Voronoi Vertex is of degree at most 3.

- However, contour separates the plane into two types of regions, so it can contain only the vertices of degree 2.

- It can only be made of *disjoint* union of closed curves and *unbounded* curves.

- Since L and R are separated by a vertical line, the contour can be oriented w.r.t. points in R (or L) (say counterclockwise w.r.t. points in R), such that each segment in the contour points in $-y$ directions.

- No horizontal cusps.

- No closed curves.

- Only vertically monotone unbounded curves.

One more Lemma

Lemma: The topmost(bottommost) edge of the contour is the perpendicular bisector for the two points forming the upper (lower) tangent of the left hull and right hull.

Proof: The vertices of the hull correspond to unbounded Voronoi polygons.

Possible Approach

1. Compute Convex Hull of L and R.

2. Compute the upper/lower tangents of the convex hull of L and R.

3. Trace the contour from top to bottom.

4. Always we are in Voronoi Polygons of two points, one from the left and one from the right, and we remain on their perpendicular bisectors!

5. To begin with, we are in Voronoi Polygons of l_0 and r_0 and we trace the perpendicular bisector of l_0 and r_0 in the negative y-direction till we intersect either the boundary of $V(l_0)$ or $V(r_0)$.

6. Suppose we intersect the boundary of $V(r_0)$ first. Now r_1 plays the role of r_0 and we start by tracing the boundary of l_0 and r_1.

How do we get efficiency

Problem: How to find which Voronoi Polygon we intersect?

Approach: Use structure of Voronoi Cells and Contour: *Convexity* and *Monotonicity*.

Theorem

The Voronoi Diagram of n-cite in the plane can be computed in $O(n \log n)$ time.

Delaunay Triangulation and Convex Hulls

Delaunay Triangulation in dimension d can be computed using the **Convex Hull** in dimension $d+1$.

Main Steps (2-dimensions):

1. Project each point $p_i = (x_i, y_i)$ to $p'_i = (x_i, y_i, x_i^2 + y_i^2)$.

2. Compute the 3-dimensional Convex Hull of p'_i's.

3. Project the convex hull back to 2-dimension.

4. Projection is Delaunay Triangulation.

Why this works?

There is a connection between faces of the convex hull and triangles in DT

Delaunay Condition: Three points p, q, and r form a DT if and only if the circumcircle of these points contain no other point of the set N.

Convex Hull condition: Three points p', q', and s' form a face of the convex hull if and only if the plane passing through them has all points of the projected set lying on one side.

Connection between **emptiness** of circumcircle and half-spaces

LEMMA

Lemma: Consider 4 distinct points p, q, r and s in the plane, and let p', q', r', and s', be their projections on the parabola. The point s lies inside within the circumcircle of p, q, and r if and only if s' lies on the lower side of the plane passing through p', q', and r'.

Proof:

1. Intersection of an arbitrary lower half space with the paraboloid when projected back to the plane is the interior of a circle.

2. Consider the projection of p, q, r to the parabola, then the lower half space, defined by the plane passing through p', q', r', and the reflection of

the hafspace to the plane. This results in a circumcircle passing through p, q, r.

3. $s \in circle(p, q, r)$ if and only if the projection s' belongs to the lower half space of the plane p', q', s'.

Theorem

Given a set of points N in the plane (assume no 4 CO-circular), and given 3 points $p, q, r \in S$, the triangle pqr is a triangle in DT if and only of triangle $p'q'r'$ is a face of the lower convex hull of the projected set.

Theorem for Voronoi Diagrams

VD and DT are Dual of each other.

Convex Hulls and Convex Polytopes are Dual.

Points and Lines are Dual.

Project points to Parabola and consider the tangential plane.

Theorem: Given a set of points N (no 4 CO-circular), let H denote the set of upper half spaces obtained by the projection. Then the Voronoi Diagram is obtained by projecting the convex polytope to the plane.

Proof: Key is that the intersection line of two half-spaces is a perpendicular bisector.