Computational Geometry Lecture Notes by David M. Mount.

{Lecture 11}

Voronoi Diagrams.

A Voronoi diagram encodes proximity information, that is, what is close to what.

Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of points in the plane, or more generally, which we call <u>sites</u>.

Let $\|pq\| = \left(\sum_{i=j}^{d} (p_j - q_j)^2\right)^{\frac{1}{2}}$ denot the <u>Euclidean distance</u> between two points p and q

Define $V(p_i)$, the <u>Voronoi cell</u> for p_i, to be the set of points q in the plane that are closer to p_i than to any other site, that is

$$V(p_i) = \{ q \in \mathbb{R}^d : \|p_i q\| < \|p_j q\|, \forall j \neq i \}.$$

the Voronoi cells of two distinct points of P are disjoint.

The union of the closure of the Voronoi cells defines a cell complex, which is called the <u>Voronoi diagram</u> of P, and is denoted $Vor(P)$.

The cells of the Voronoi diagram are convex polyhedra.

Applications:

- Nearest neighbour queries
- Computational morphology and shape analysis (medial axis)
- Center-based clustering
- Neighbours and interpolation (natural neighbour interpolation)

Properties of the Voronoi diagram (in the plane):

☆ - Empty circle properties:

Each point on an edge of the Voronoi diagram is equidistant from its two nearest neighbours p_i and p_j. Thus, there is a circle centered at any such point where p_i and p_j lie on this circle, and no other site is interior to the circle.

- Voronoi vertices: the vertex at which three Voronoi cells $V(p_i)$, $V(p_j)$, $V(p_k)$ intersect called a Voronoi vertex is equidistant from all sites. Thus it is the center of the circle passing through the sites, and this circle contains no other sites in its interior.

convex hull. A cell of the Voronoi diagram is unbounded if and only if the corresponding site lies on the convex hull.

size: Let n denote the number of sites, then the Voronoi diagram has exactly n faces. It follows from Euler's formula $v - e + f = 2$ The number of Voronoi vertices is roughly $2n$ and the number of edges is roughly $3n$

$3V = 2e$

$V - \frac{3}{2}V + n = 2$ $\quad e = 3n-6$

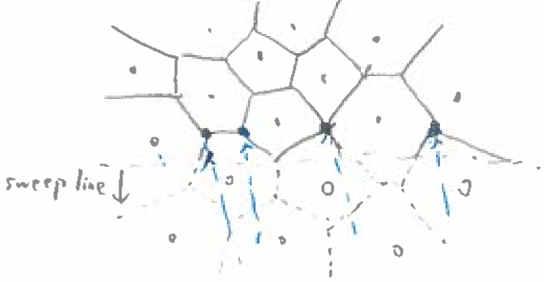
$-\frac{1}{2}V = n - 2$

$V = 2n-4$

Fortune's Algorithm (Steven Fortune)

$O(n \log n)$ runtime

$O(n)$ space



beach line: The set of points q that are equidistant from the sweep line to their nearest site above the sweep line is called the beach line.

The set of points that are equidistant from a point (a site) and a line (the sweep line) is a parabola.
The beach line consists of the lower envelope of these parabolas, one for each site.
Because the parabolas are x-monotone, so is the beach line.

The point where two arcs of the beach line intersect is called a breakpoint.

Lemma The beach line is an x-monotone curve made up of parabolic arcs. The breakpoints of the beach line line on Voronoi edges of the final diagram.

Sweep-line Status:

The algoritm maintains the current location (y-coordinate) of the sweep line.
It stores, in left-to-right order, the sequence of sites that define the beach line.

Events:

Site events: when sweep line passes over a new site a new parabolic arc will be inserted into beach line.

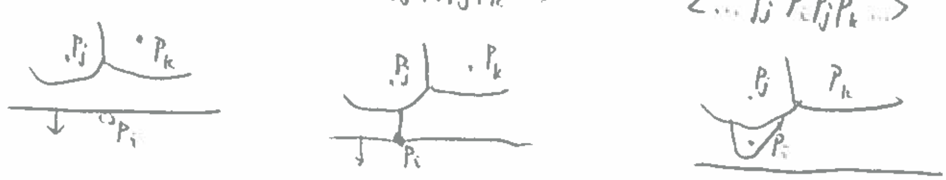
Voronoi vertex events: (circle events)
When the length of an arc of the beach line shrinks to zero, the arc disappears and a new Voronoi vertex will be created at this point.

Site events:

Prior to event — $\langle \dots p_j p_j \dots \rangle$

At the event — $\langle \dots p_j p_i p_j p_k \dots \rangle$

After the event — $\langle \dots p_j p_i p_j p_k \dots \rangle$



(1) advance sweep line
(2) replace $\langle \dots p_j \dots \rangle$ with $\langle \dots p_j, p_i, p_j \rangle$
(3) create a new (dangling) edge in the Voronoi diagram, which lies on the bisector between p_i and p_j
(4) some old triples that involved p_j may be deleted and new triples involving p_i may be inserted

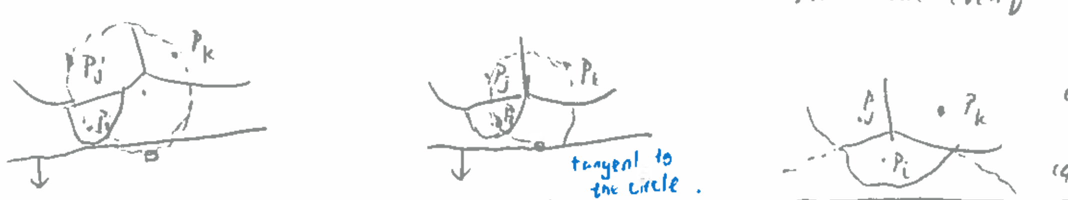
The maximum number of arcs on the beach line can be at most $2n-1$. (each site creates 2 new arcs except the last one)
The sites can be presorted by the y-coordinates and inserted as a batch into the event priority queue.

Voronoi vertex events:

Prior to event — $\langle \dots p_j p_i p_j p_k \dots \rangle$

At the event — $\langle \dots p_j p_i p_k \dots \rangle$

After the event — $\langle \dots p_j p_i p_k \dots \rangle$



(1) delete entry for p_j from beach line status
(2) create a new vertex in Voronoi diagram
(3) create a new (dangling) edge for the bisector between p_i and p_k
(4) delete any events that arose from triples involving the arc of p_j, and generate new events corresponding to consecutive triples involving p_i and p_k.

Sweep-line algorithm:

- Voronoi diagram: The partial Voronoi diagram that has been constructed so far will be stored in any reasonable data structure for storing planar subdivisions e.g. a doubly-connected edge list.
- Beach line: The beach line consists of the sorted sequence of sites whose arcs form the beach line.

- Event queue: The event queue is a priority queue with the ability both to insert and delete new events.

{Lecture 12}

Delaunay Triangulations

The Voronoi diagram of a set of sites in the plane is a planar subdivision, that is, a cell complex. The resulting dual graph is a triangulation of the sites called the Delaunay triangulation.

Properties:

1) <u>Convex hull</u>: The boundary of the exterior face of the Delaunay triangulation is the boundary of the convex hull of the point set

2) <u>Circumcircle property</u>: The circumcircle of any triangle in the Delaunay triangulation is empty. (contains no sites of P).

3) <u>Empty circle property</u>: Two sites p_i and p_j are connected by an edge in the Delaunay triangulation, iff. there is an empty circle passing through p_i and p_j.

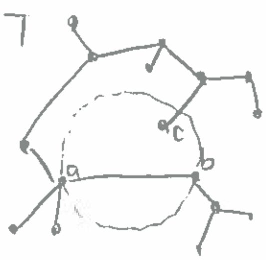
4) <u>Closest pair property</u>: The closest pair of sites in P are neighbours in the Delaunay triangulation.

Given a point set P with n vertices where there are h on the convex hull, it is not
? hard to prove by Euler's formula that the Delaunay triangulation has $2n-2-h$ triangles and $3n-3-h$ edges.

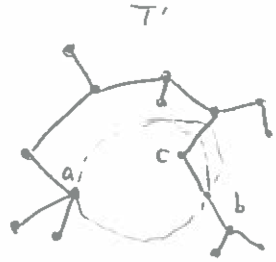
$v + f - e = 2$
$n + f - \frac{3}{2}e = 2$
f

<u>Theorem</u> The <u>minimum spanning tree</u> of a set of points P (in any dimension) is a subgraph of the Delaunay triangulation.

T



T'



$w(T') = w(T) + \|bc\| - \|ab\| < w(T)$.

Does the Delaunay triangulation minimize the total edge length?

The answer is NO.

The triangulation that minimizes total edge weight is called <u>minimum weight triangulation</u>

In 2008 it was proved that this problem is NP-hard.

Spanner Properties:

Consider any point set P and a straight-line graph G whose vertices are the points of P. For any two points $p, q \in P$, let $\delta_G(p,q)$ denote the length of the shortest path from p to q in G, where the weight of each edge is its Euclidean length.

Given any parameter $t \geq 1$, we say that G is a <u>t-spanner</u> if any two points $p, q \in P$, the shortest path between p and q in G is at most a factor t longer than the Euclidean distance between these points, that is

$$\delta_G(p,q) \leq t \, \|pq\|$$

<u>Theorem</u> (Keil and Gutwin)

Given a set of points P in the space, the Delaunay triangulation of P is a t-spanner for $t = 4\pi\sqrt{3}/9 \approx 2.418$.

Maximizing Angles and Edge Flipping:

<u>Theorem</u> Among all triangulations of a given planar point set, the Delaunay triangulation has the lexicographically largest angle sequence, and in particular it maximizes the minimum angle.

{Lecture 13} Delaunay Triangulations: Incremental Construction.

A simple randomized algorithm for constructing the Delaunay triangulation of a set of n sites in the plane. Its expected running time is $O(n \log n)$.

The input consists of a set $P = \{p_1, \dots, p_n\}$ of points in the plane.

The idea is to insert sites in random order, one at a time, and update the triangulation with each new addition. After each insertion, the expected number of structural changes in the diagram is $O(1)$.

We will store each uninserted sites in a bucket according to the triangle in the current triangulation that contains it. We will show that the expected number of times that a site is rebucketed is $O(\log n)$.

Incircle Test:

We claim that d lies in the circumcircle determined by the Δabc if and only if the following determinant is positive.

$$\text{inCircle}(a,b,c,d) = \det \begin{pmatrix} a_x & a_y & a_x^2+a_y^2 & 1 \\ b_x & b_y & b_x^2+b_y^2 & 1 \\ c_x & c_y & c_x^2+c_y^2 & 1 \\ d_x & d_y & d_x^2+d_y^2 & 1 \end{pmatrix} > 0$$

inCircle(a,b,c,d) < 0

inCircle(a,b,c,d) = 0

inCircle(a,b,c,d) > 0

proof. If four points are cocircular then $\exists$ center point $q = (q_x, q_y)$, and radius r, s.t.

$$(a_x - q_x)^2 + (a_y - q_y)^2 = r^2$$

$$\Rightarrow \quad 0 = (-2q_x)\,a_x + (-2q_y)\,a_y + 1\cdot(a_x^2 + a_y^2) + (q_x^2 + q_y^2 - r^2)\cdot 1$$

$$\begin{pmatrix} a_x & a_y & a_x^2+a_y^2 & 1 \\ b_x & b_y & b_x^2+b_y^2 & 1 \\ c_x & c_y & c_x^2+c_y^2 & 1 \\ d_x & d_y & d_x^2+d_y^2 & 1 \end{pmatrix} \begin{pmatrix} -2q_x \\ -2q_y \\ 1 \\ q_x^2+q_y^2-r^2 \end{pmatrix} = 0$$

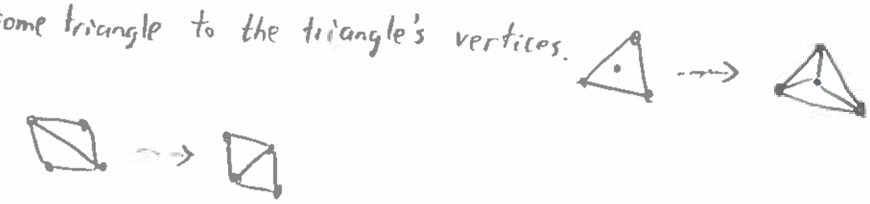
Incremental update:

The algorithm begins by inserting three points that are a huge distance from the other points, such that this massive triangle encloses all the other points.

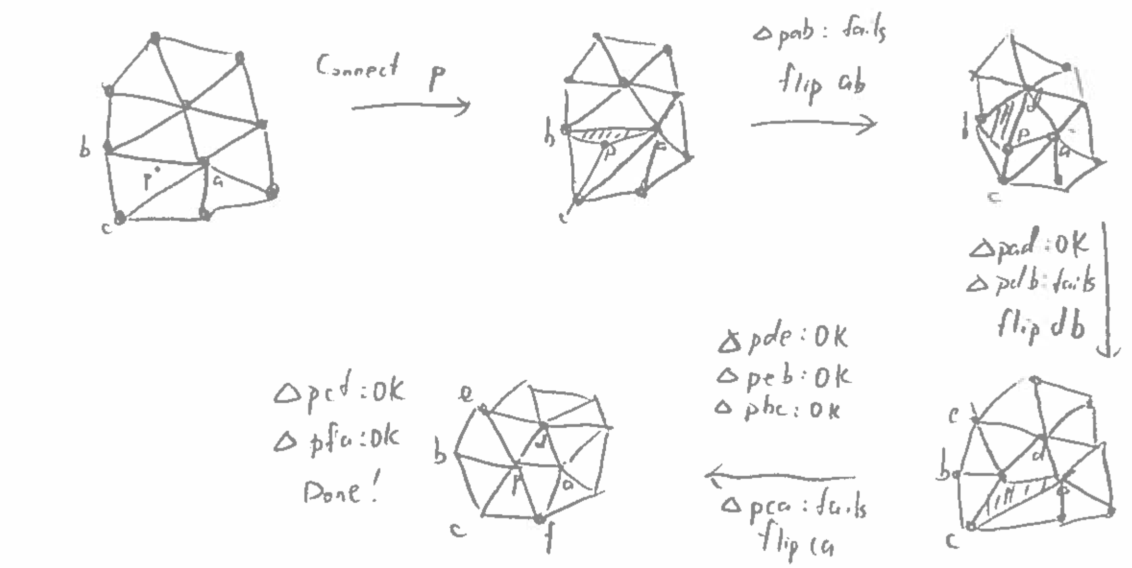
We then insert sites of P one by one. The basic changes are.

(a) Join a site in the interior of some triangle to the triangle's vertices.

(b) Perform an edge flip.



Both operations can be performed in $O(1)$ time assuming that the triangulation is maintained in any reasonable way, say, as a double-connected edge list. ?



For each of the triangles that have been added, we check the vertex of the triangle that lies on the opposite side of the edge that does not include p.

* Randomized Incremental Delaunay Triangulation Algorithm *

```
Insert (p) {
    Find the triangle Δabc containing p;
    Insert edges pa, pb, pc into triangulation;
    SwapTest (ab);                    // check/fix the surrounding edges.
    SwapTest (bc);
    SwapTest (ca);
}

SwapTest (ab) {
    if (ab is an edge on the exterior face) return;
    Let d be the vertex to the right of edge ab;
    if (inCircle (p, a, b, d)) {                // d violates the incircle test
        Flip edge ab for pd;
        SwapTest (ad);                          // check/fix new suspect edges.
        SwapTest (db);
    }
}
```

- A triangulation is <u>locally Delaunay</u> if for each triangle the vertices lying on the opposite side of each edge of the (up to) three neighbouring triangles satisfy the empty circle condition.
- A triangulation is <u>globally Delaunay</u> means that empty circle condition is satisfied for all triangles and all points of P.

Structial Changes: the expected number of edge changes with each insertion is $O(1)$

The total number of changes made in the triangulation for the insertion of p is proportional to the degree of p after the insertion is complete.

Consider the situation after the insertion of the ith site. The expected time to insert ith site is equal to the average degree of a vertex in the triangulation of i sites.

By Euler's formula we know the average degree of a vertex in any planar graph is at most 6.

(n vertices can have at most $3n$ edges, the sum of vertex degrees is equal to twice the number of edges)

Thus, the expected number of edge changes is $O(1)$.

Summing over all n insertions, the total number of structial changes is $O(n)$. Each structial change can be performed in $O(1)$ time.

Rebucketing: the total expected time spent in rebucketing points is $O(n \log n)$.

We will show that the expected number of times that any site is rebucketed is $O(\log n)$.

Let us fix a site $q \in P$. Consider the situation just after the insertion of the ith site

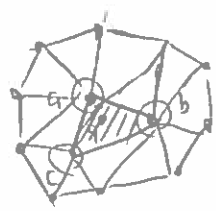
We assert that the probability that q was rebucketed as a result of the last insertion is a most $\frac{3}{i}$.

Let Δ be the triangle containing q after the ith insertion. Δ would have come into existence as a result of the last insertion if and only if one of its three vertices was the last to be added. Thus, the probability that q required rebucketing after the last insertion is at most $3/i$.

After each stage there are $n-i$ points that might subject to rebucketing, and each has probability $\frac{3}{i}$ of being rebucketed. Thus, the expected number of points that require rebucketing as part of the last insertion is at most $(n-i)\frac{3}{i}$.

$$\sum_{i=1}^{n} \frac{3}{i}(n-i) \leq \sum_{i=1}^{n} \frac{3}{i} n = 3n \sum_{i=1}^{n} \frac{1}{i} = 3n \ln n + O(1)$$

Thus, the total expected time spent in rebucketing is $O(n \log n)$.



q would have been rebucketed only if one of a, b, or c was the last to be inserted.