Recall

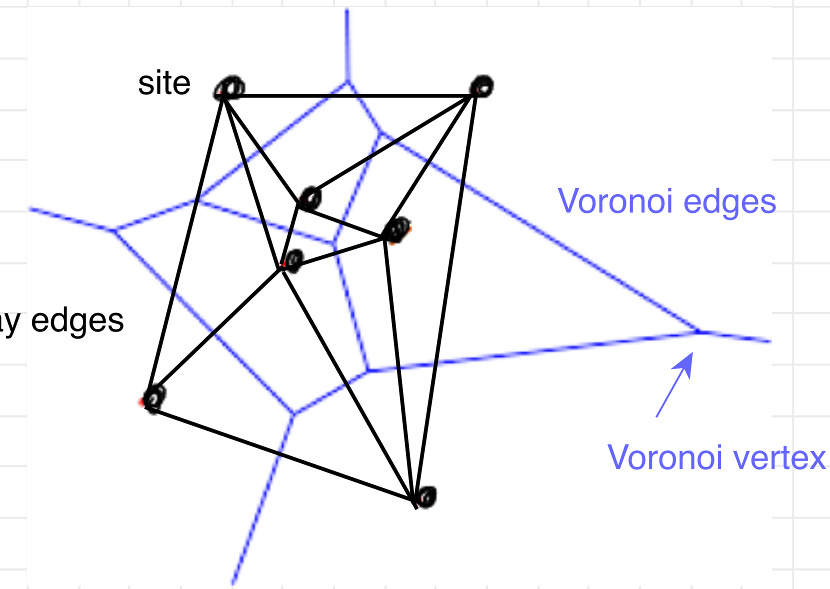
Voronoi diagram

Given points $P = \{p_1, \ldots, p_n\}$ in the plane, the ***Voronoi region*** of p_i is

$$V(p_i) = \{x \in \mathbb{R}^2 : d(x, p_i) \leq d(x, p_j) \forall j \neq i\}$$

p_i is called a ***site***.

The ***Voronoi diagram*** $\mathcal{V}(P)$ consists of all the Voronoi regions



Given points $P = \{p_1, \ldots, p_n\}$ in the plane, the ***Delaunay triangulation*** $\mathcal{D}(P)$ is a graph with vertices $p_1, \ldots, p_n$ and edge (p_i, p_j) iff $V(p_i)$ and $V(p_j)$ share an edge.

$\mathcal{D}(P)$ is the ***planar dual*** of $\mathcal{V}(P)$

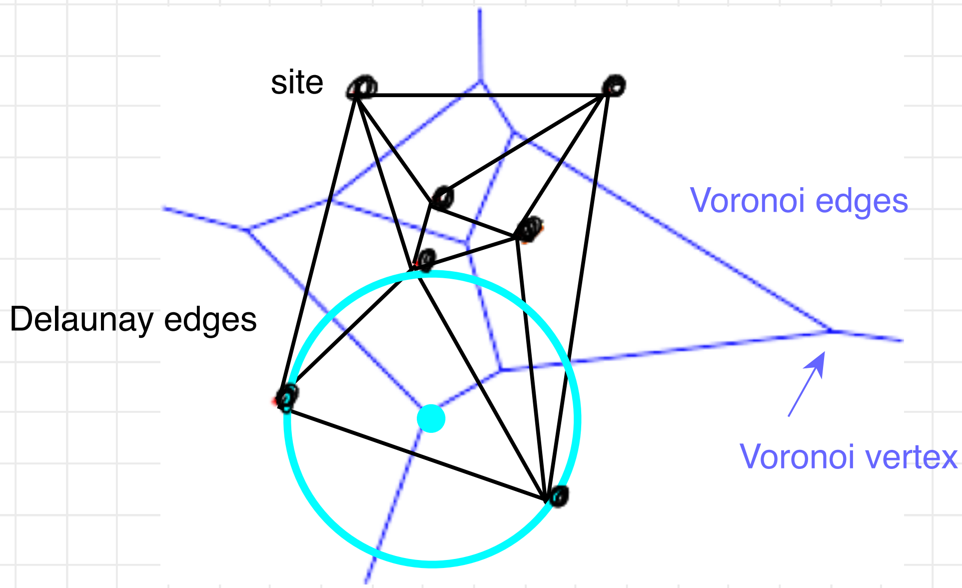
Recall

Properties

Voronoi vertices have degree 3 (we assume no 4 ~~points~~ sites co-circular).
Voronoi cells are convex.

$V(p_i)$ is unbounded iff p_i is on the convex hull of the sites.
There are $\leq 2n$ Voronoi vertices and $\leq 3n$ Voronoi edges.

$\mathcal{D}(P)$
- is a triangulation.
- has an edge (p_i, p_j) iff there is an empty circle through $p_i p_j$.
- has a face $p_i\ p_j\ p_k$ iff there is an empty circle through $p_i\ p_j\ p_k$ (centered at the corresponding Voronoi vertex).

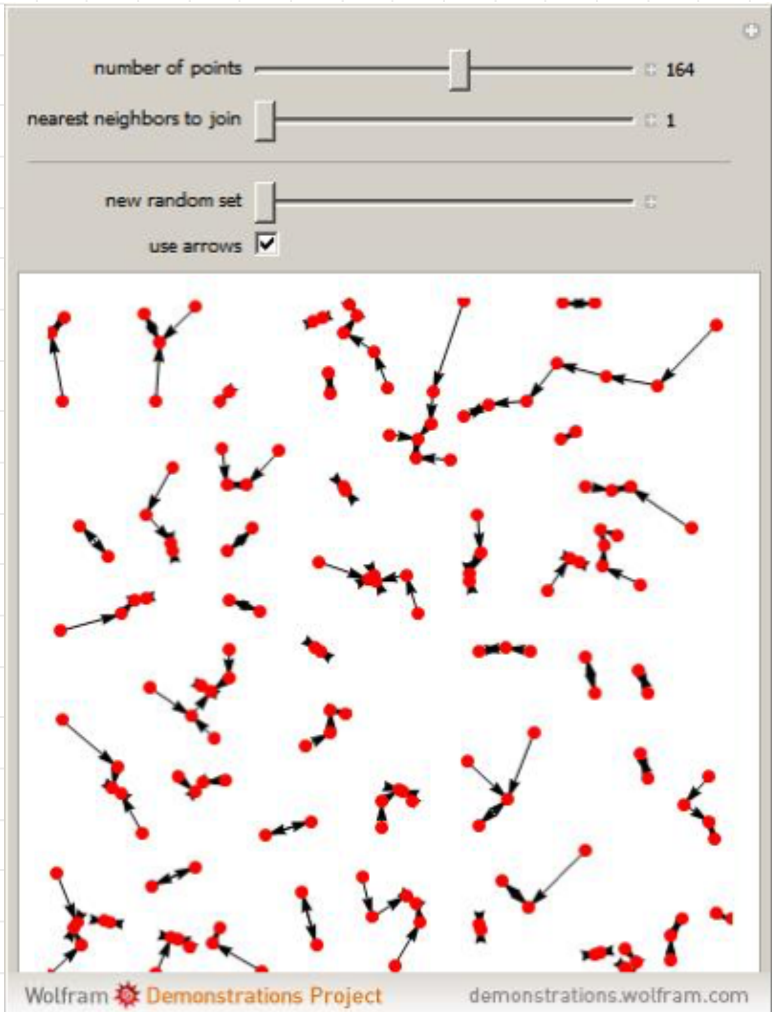


Outline:

- applications of Voronoi diagrams, Delaunay triangulations
- $O(n \log n)$ algorithm for Voronoi diagram
- relationship to convex hull problem

Application of Delaunay triangulations: finding all nearest neighbours

Given n points in the plane find, for each point, its nearest neighbour — gives ***nearest neighbour graph***, a directed graph of out-degree 1.



Many applications, e.g.

in statistical analysis:
find hierarchical clusters using
nearest neighbour chain algorithm

The ***Nearest Neighbour Graph***, NN(P), has vertices P, and a directed edge (u,v) if u's nearest neighbour is v.

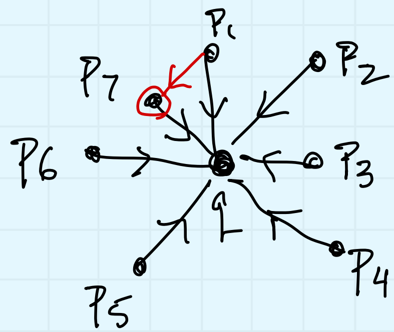
obvious algorithm $O(n^2)$

https://demonstrations.wolfram.com/NearestNeighborNetworks/

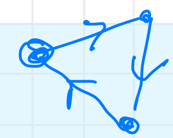
The ***Nearest Neighbour Graph***, NN(P), has vertices P, and a directed edge (u,v) if u's nearest neighbour is v.

Note: break ties so every vertex has out degree 1, and do it to avoid cycles, e.g. choose nearest neighbour of min x, max y.

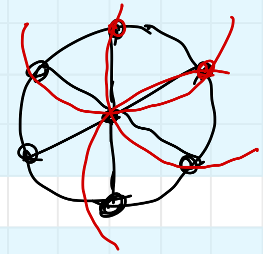
What is the in-degree of a vertex?



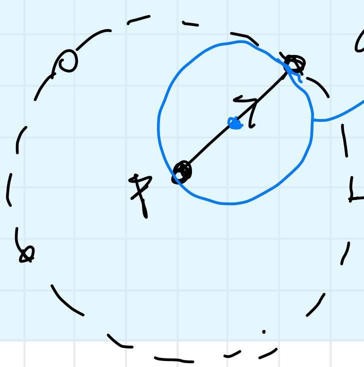
Not possible — p_1's closest neighbour is p_7



answer: max degree 6



Claim. NN(P) ⊆ $\mathcal{D}(P)$



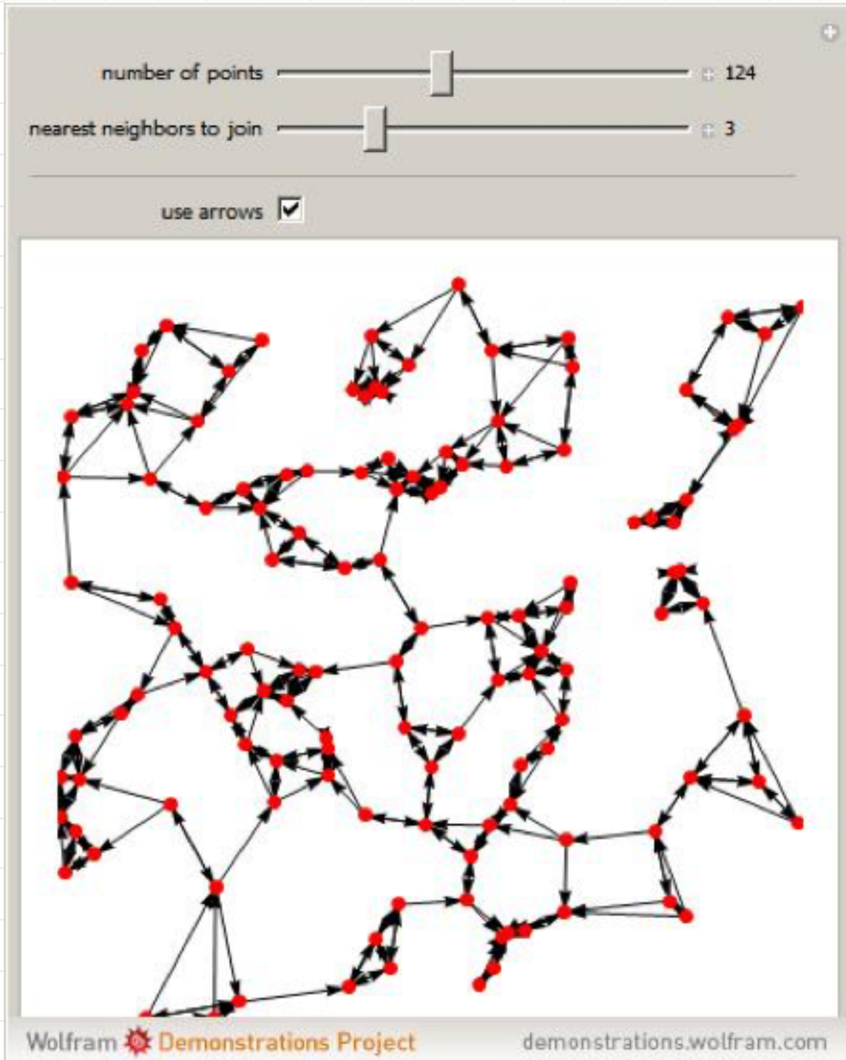
circle with diameter pq — this is empty so pq is an edge of $\mathcal{D}(P)$.

Algorithm to find NN(P)

- find $\mathcal{D}(P)$ — $O(n \log n)$ (to come)
- for every site check its neighbours in $\mathcal{D}(P)$ to find closest.

  $O(n)$ because # edges in $\mathcal{D}(P)$

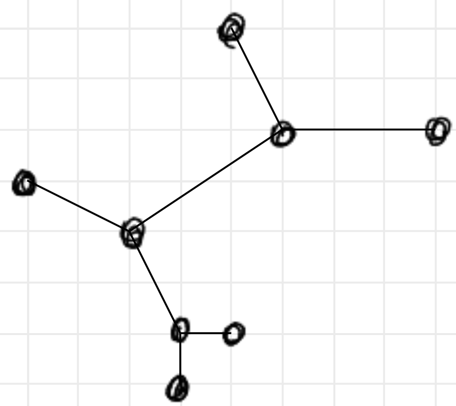
Can also look at *k* nearest neighbours — use *k*-th order Voronoi diagrams (later)



3 nearest neighbours

Application of Delaunay triangulations: finding min spanning trees (MST)

Given points $p_1, \ldots, p_n$ in the plane, find the ***Euclidean minimum spanning tree***
= tree with vertex set $p_1, \ldots, p_n$ of minimum total length



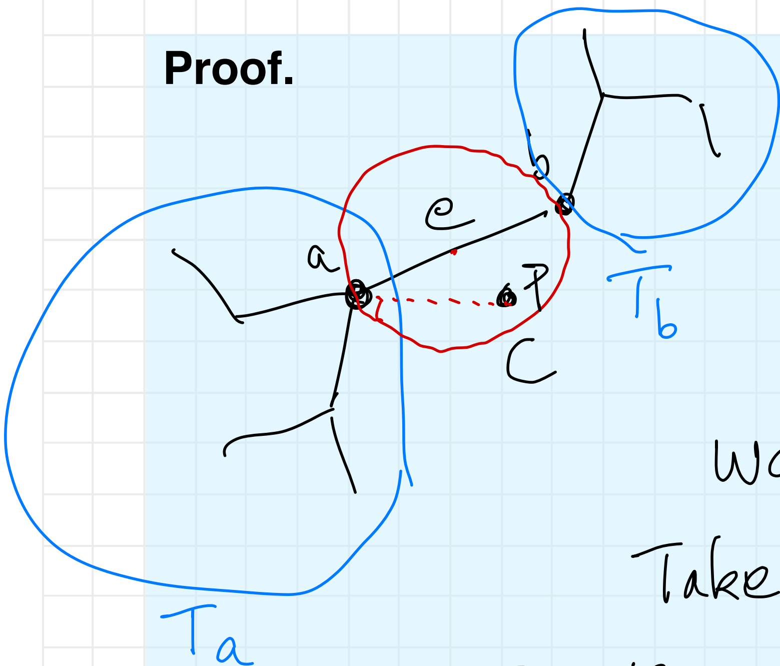
There are good algorithms to find the min weight spanning tree in any edge-weighted graph. But our graph has $O(n^2)$ edges.

Lemma. The minimum spanning tree is a subgraph of the Delaunay triangulation.

Then we can run the graph MST algorithm on the Delaunay triangulation to get an algorithm with total run time $O(n \log n)$.

Lemma. The minimum spanning tree (MST) is a subgraph of the Delaunay triangulation.

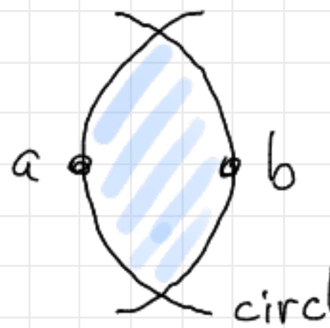
Proof.



Consider edge e of MST
remove e.
Get ~~two~~ trees T_a, T_b
want an empty circle through a and b
Take circle C with diameter ab
Is there a point inside it?
Suppose p inside C,
Suppose $p \in T_b$ (if $p \in T_a$ we'd add pb)
claim $|ap| < |e|$
So adding ap instead of e
gives a spanning tree of smaller weight.
Contradiction. ∴ C empty $e \in \mathcal{D}(P)$

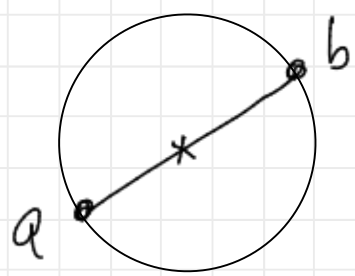
Other Proximity Graphs: Relative Neighbourhood and Gabriel graphs

Relative Neighbourhood Graph (RNG)



edge (a,b) if this ***lune*** is empty,
i.e. there is no point closer to both a and b than d(a,b)

Gabriel Graph (GG)



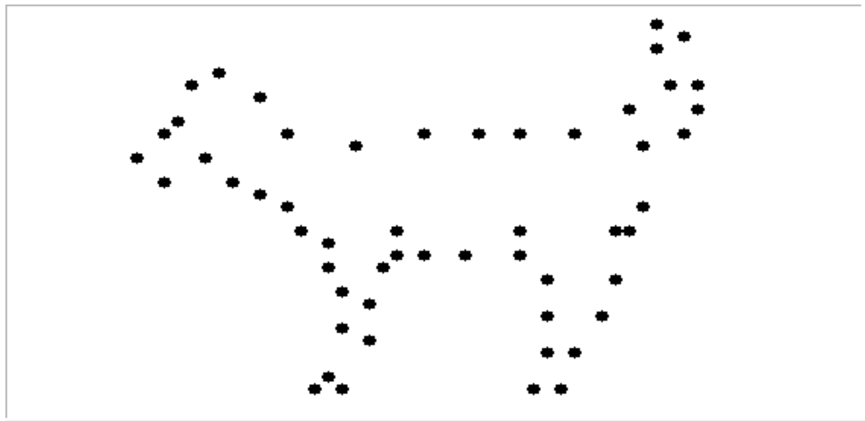
edge (a,b) if the circle with diameter ab is empty

can prove:
NN(P) $\subseteq$ MST(P) $\subseteq$ RNG(P) $\subseteq$ GG(P) $\subseteq$ $\mathcal{D}(P)$

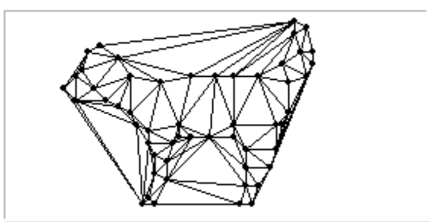
O(n log n) for MST

and all of these can be computed in O(n) time from $\mathcal{D}(P)$ (not obvious)

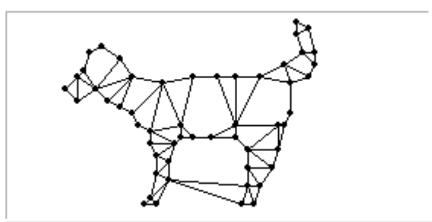
Other Proximity Graphs: Relative Neighbourhood and Gabriel graphs



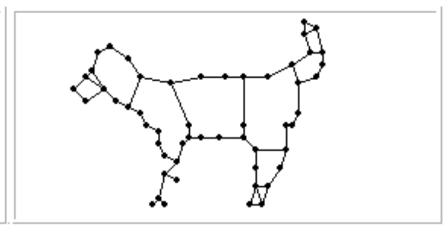
Delaunay Triangulation



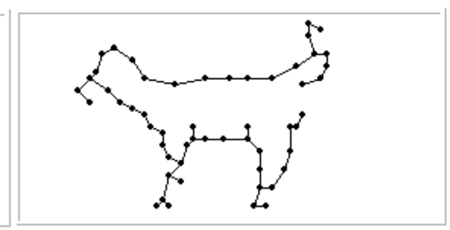
Gabriel Graph



Relative Neighbourhood Graph



Minimum Spanning Tree



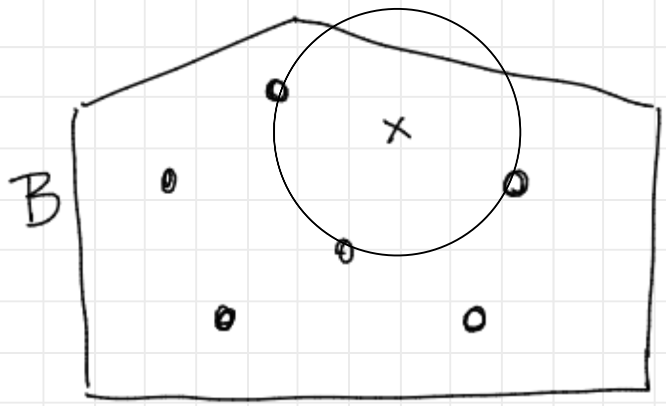
Brendan Colloran

Voronoi diagrams

Application of ~~Delaunay triangulations~~: finding largest empty circle

This is a facility location problem.
(Recall that in Lecture 7 we looked at a different facility location problem — to find the smallest circle enclosing given points.)

Given *n* points in a convex boundary polygon *B*, find the largest empty circle with center in *B*

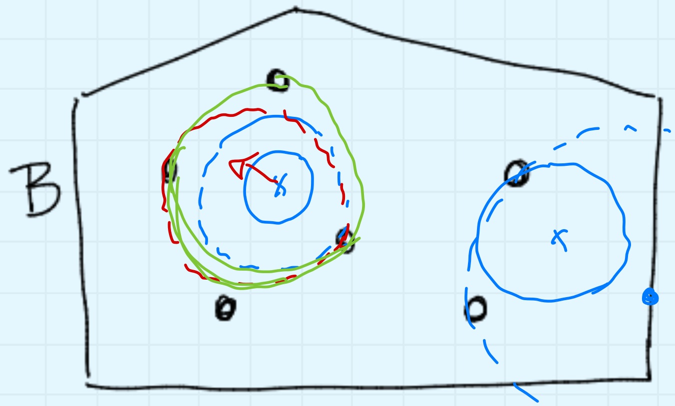


e.g. locate a new store location among existing stores, or
locate a nuclear waste dump among cities

Lemma. The center of the largest empty circle is either

- a Voronoi vertex
- the intersection of a Voronoi edge with the boundary of *B*
- a vertex of *B*

Proof idea. where is the center of largest empty circle

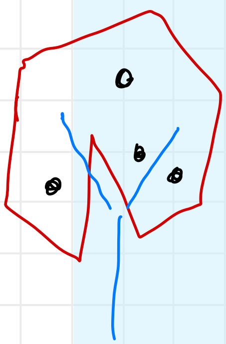


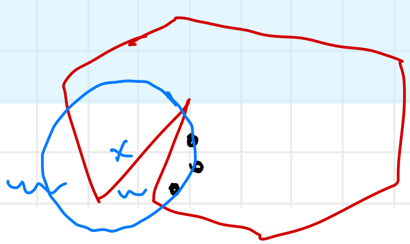
Consider any empty circle C centered at x

We can enlarge C and move x until

- x at Voronoi vertex C goes thru 3 ~~pts~~ sites
- x on boundary of B and on Voronoi edge C goes through 2 sites
- x at vertex of B C goes through 1 site.

what about B non-convex?





whether this is useful might depends on application.

Algorithm for the largest empty circle problem
Input: *n* points in polygon *B* with *k* vertices

- compute Voronoi diagram of the points — $O(n \log n)$
- compute intersection points of Voronoi edges with the polygon — $O(n)$

- try each possible center *p* from the above Lemma

  - Voronoi vertex *p* — $O(n)$
    & check its 3 nearest neighbours

  - intersection point *p* of Voronoi edge *e* and polygon — $O(n)$ of these

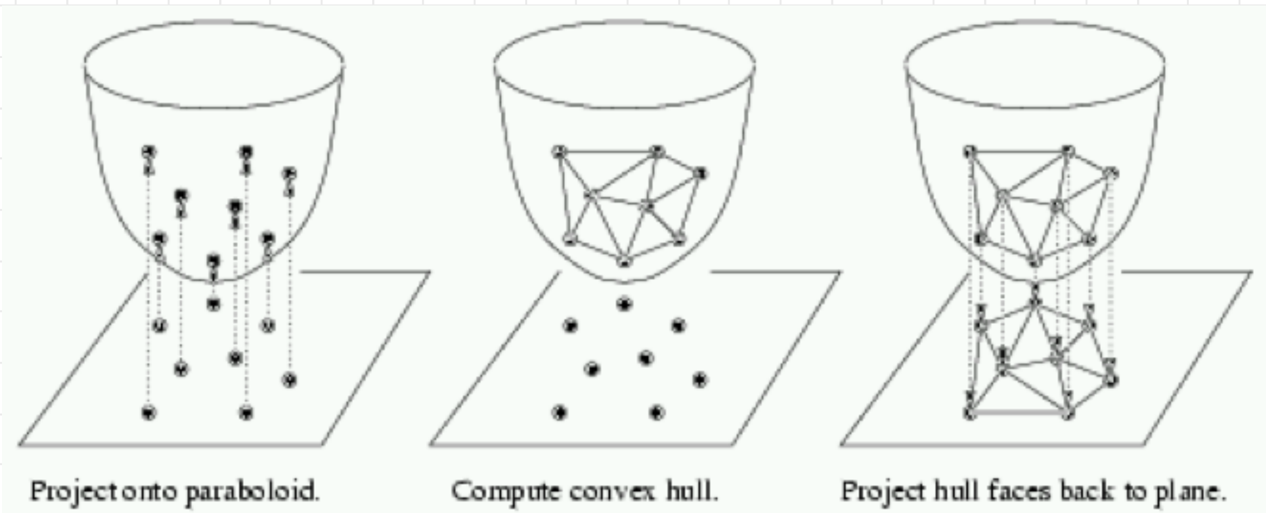
  - polygon vertex *p*
    find which region of $V(P)$ contains p
    preprocess $V(P)$ for planar point location — $O(n \log n)$
    then query every vertex of B — $O(k \log n)$

Runtime:
$O((n+k) \log n)$

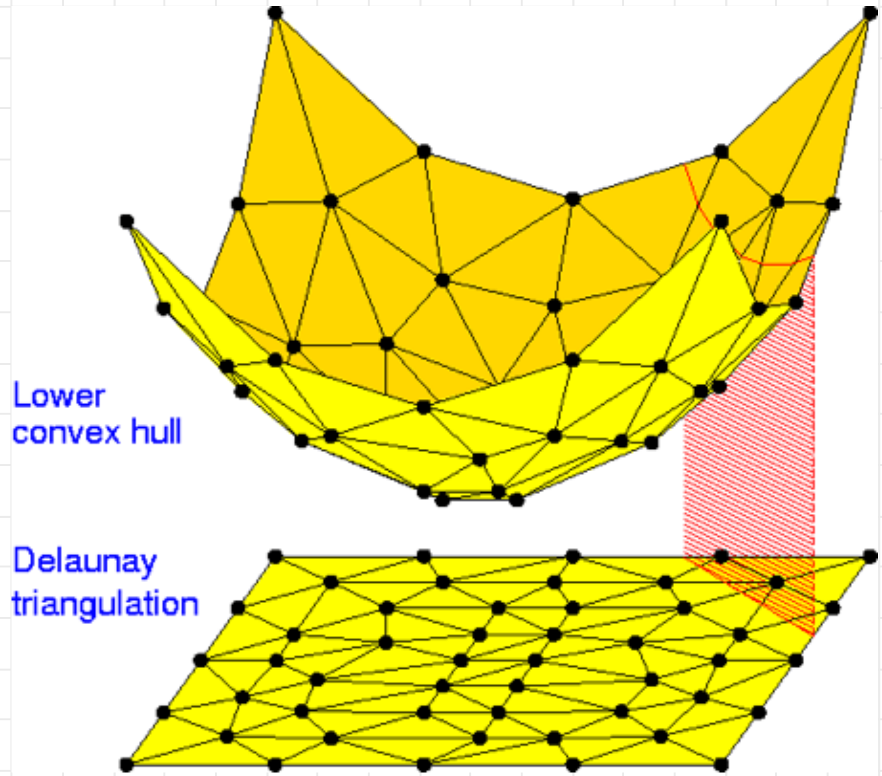
Connection between Voronoi diagram / Delaunay triangulation and Convex Hull

Given $p_1, \ldots, p_n \in \mathbb{R}^2$ project them up onto parabola $z = x^2 + y^2$

$$p = (x_p, y_p) \longmapsto \hat{p} = (x_p, y_p, x_p^2 + y_p^2)$$



Theorem. The lower convex hull of $\hat{p}_1, \cdots, \hat{p}_n$, projected back to the plane, is the Delaunay triangulation of $p_1, \ldots, p_n$



Jonathan Shewchuck

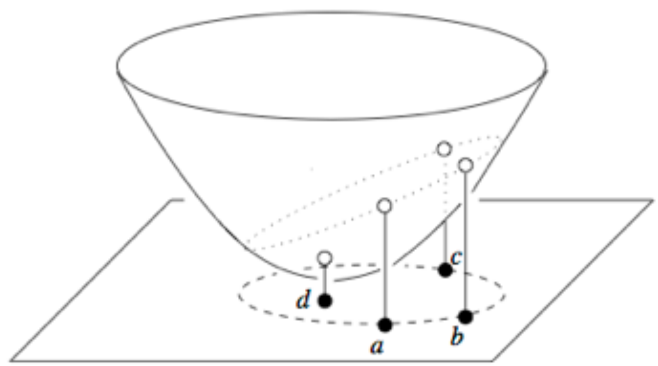


Figure 1.11. Points a, b, c lie on the dashed circle in the x_1x_2-plane and d lies inside that circle. The dotted curve is the intersection of the paraboloid with the plane that passes through $\hat{a}$, $\hat{b}$, $\hat{c}$. It is an ellipse whose projection is the dashed circle.

Herbert Edelsbrunner

Theorem. The lower convex hull of $\hat{p}_1, \dots, \hat{p}_n$, projected back to the plane, is the Delaunay triangulation of $p_1, \dots, p_n$

Proof.

Claim 1. Points in the plane are co-circular iff their projections on the parabola are co-planar.

equation of circle center (p, q) radius r

$$(x-p)^2 + (y-q)^2 = r^2$$

re-arrange

$$\underbrace{(x^2+y^2)}_{z} - 2xp - 2yq + (p^2+q^2-r^2) = 0$$

so this is a plane in $x, y, z = x^2+y^2$.

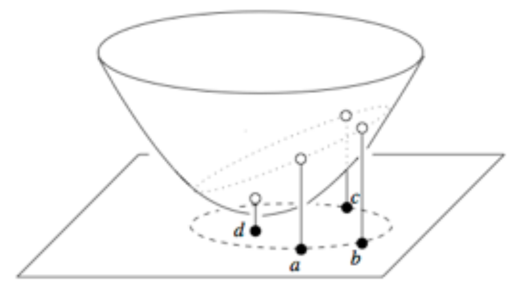


Figure 1.11. Points a, b, c lie on the dashed circle in the x_1x_2-plane and d lies inside that circle. The dotted curve is the intersection of the paraboloid with the plane that passes through $\hat{a}$, $\hat{b}$, $\hat{c}$. It is an ellipse whose projection is the dashed circle.

Claim 1. Points outside the circle map to points above the plane; points inside the circle map to points below the plane.

Theorem. The lower convex hull of $\hat{p}_1, \ldots, \hat{p}_n$, projected back to the plane, is the Delaunay triangulation of $p_1, \ldots, p_n$

Proof. $\hat{a}$ $\hat{b}$ $\hat{c}$ form a face of lower convex hull
iff no sites lie below plane through $\hat{a}, \hat{b}, \hat{c}$
iff no sites lie inside circle thru a, b, c
iff a, b, c form a triangle in $\mathcal{D}(P)$.

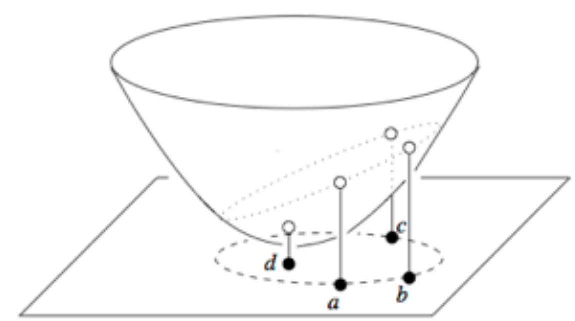


Figure 1.11. Points a, b, c lie on the dashed circle in the $x_1 x_2$-plane and d lies inside that circle. The dotted curve is the intersection of the paraboloid with the plane that passes through $\hat{a}$, $\hat{b}$, $\hat{c}$. It is an ellipse whose projection is the dashed circle.

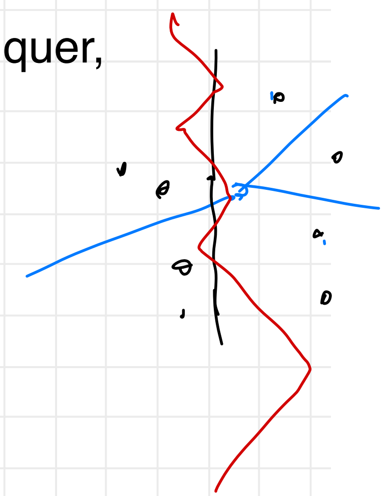
Algorithms to compute Voronoi diagrams / Delaunay triangulations

- we can get either one from the other in O(n) time.

- we can compute the Delaunay triangulation in O(n log n) time using a 3D convex hull algorithm.

- first O(n log n) algorithm to compute Voronoi diagram was divide and conquer, Shamos and Hoey, 1975. The merge step is complicated.

- Steve Fortune, ’87, gave a sweepline algorithm for Voronoi diagram



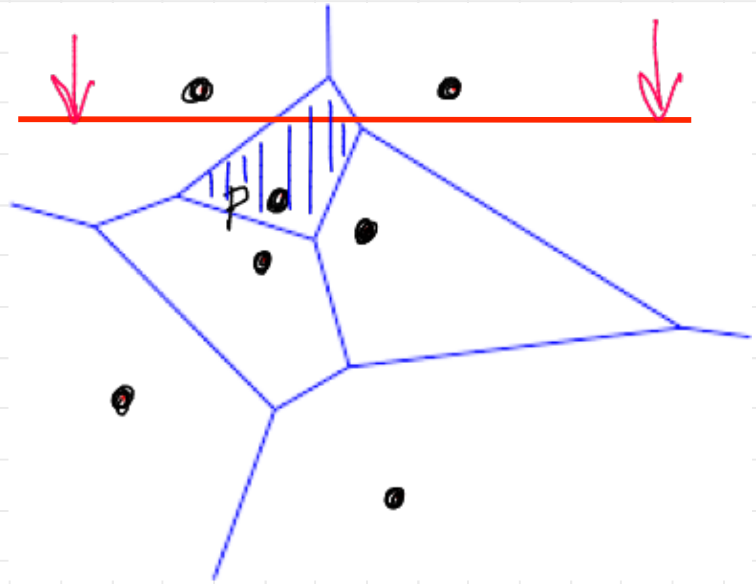
next lecture:

- randomized incremental algorithm to compute the Delaunay triangulation

Fortune's sweepline algorithm for Voronoi diagram

the difficulty with a sweepline approach:

V(p) starts before we reach p

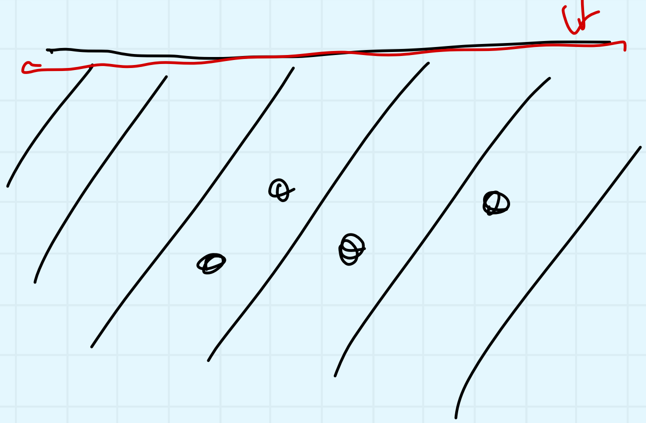


Solution

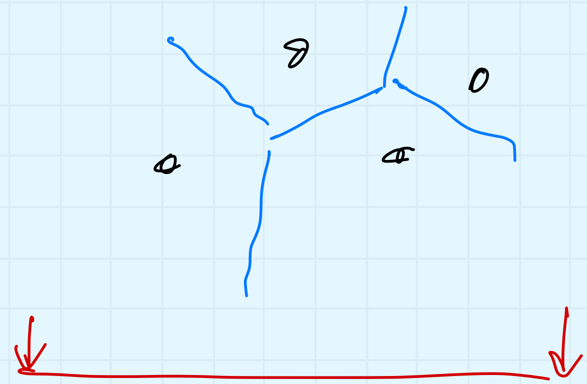
Find the Voronoi diagram of the points PLUS the half plane below the sweep line.

Find the Voronoi diagram of the points PLUS the half plane below the sweep line.

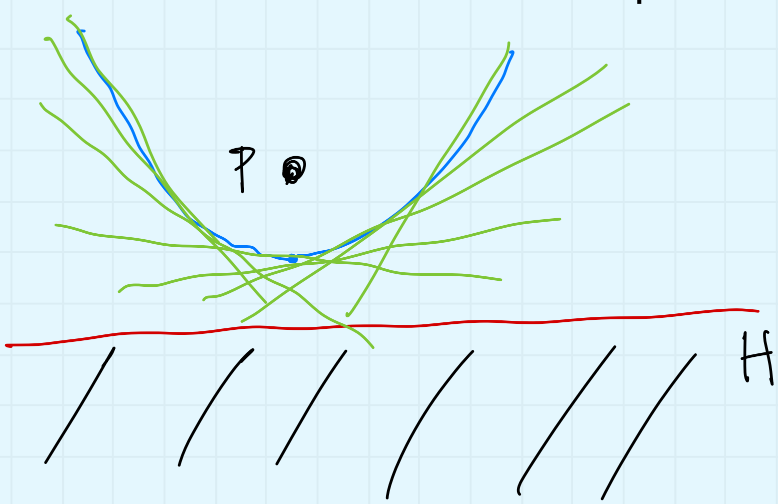
initial situation

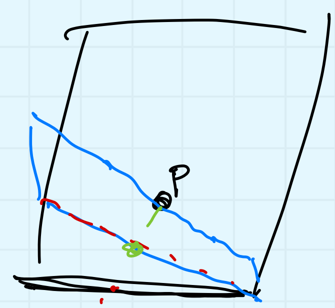


final situation



intermediate situation for one point

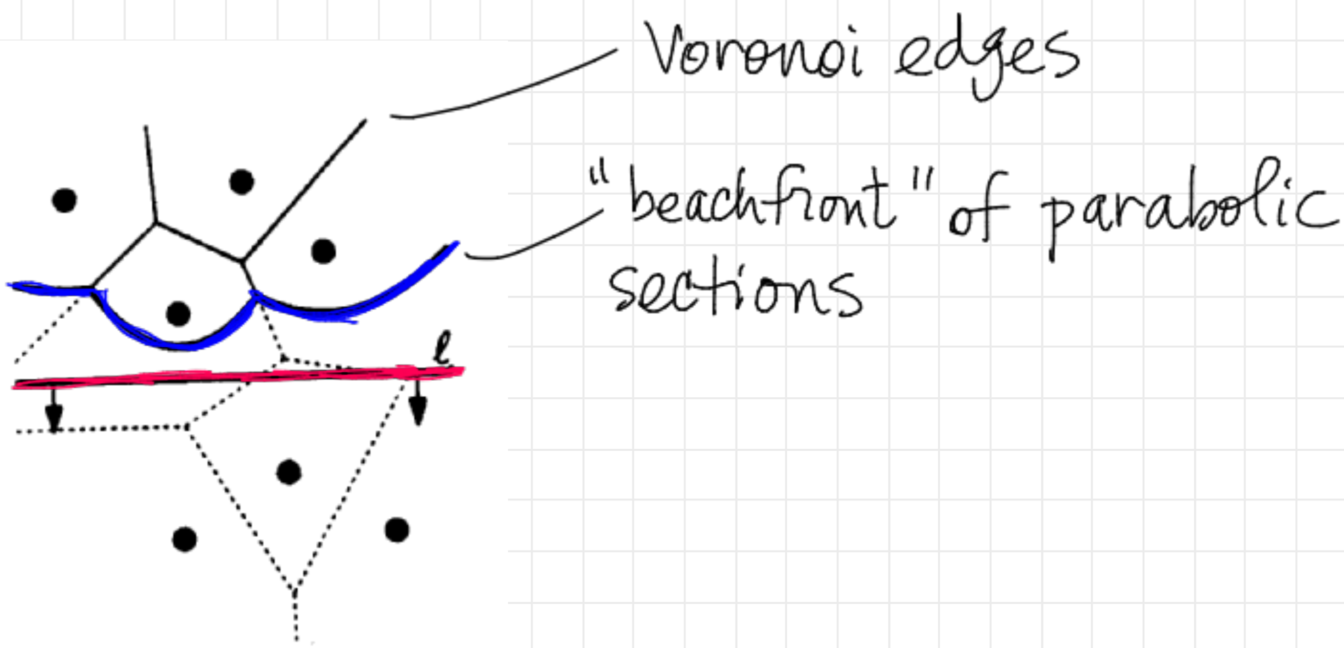




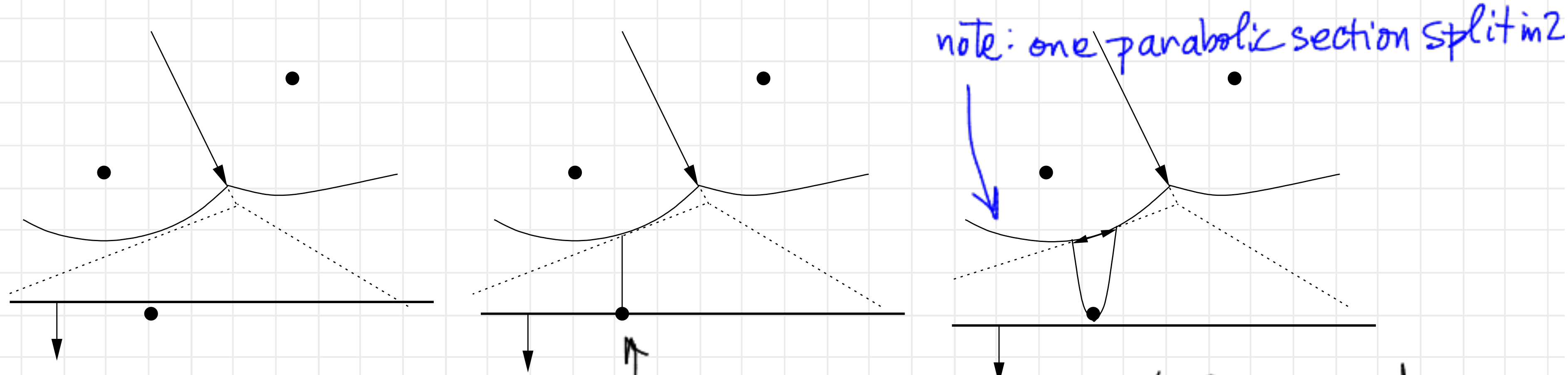
fold paper so bottom edge hits p.

repeat

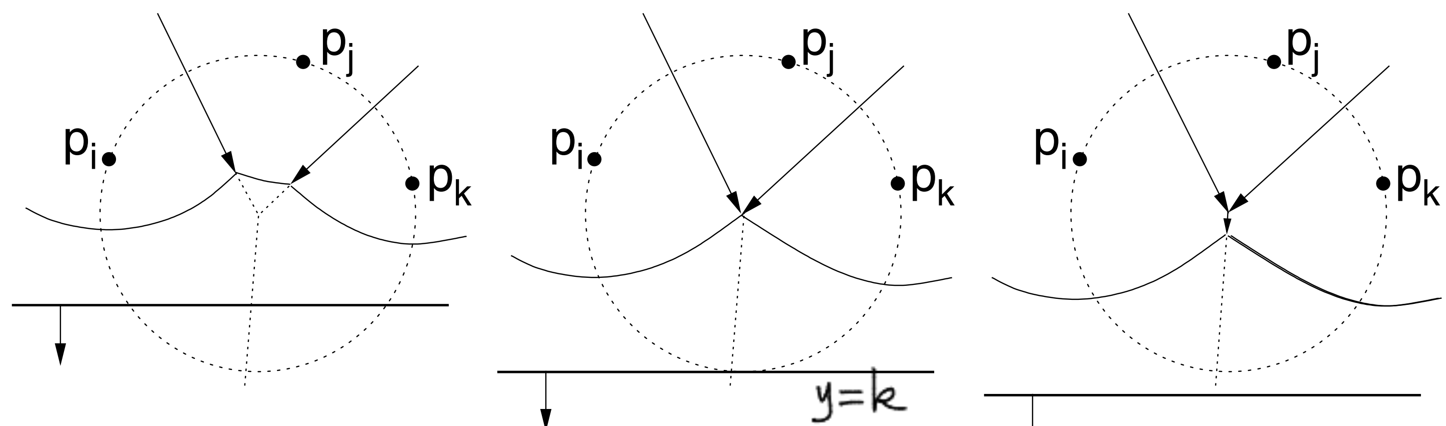
intermediate configuration of Fortune's algorithm



https://www.youtube.com/watch?v=rvmREoyL2F0 update events for Fortune's algorithm



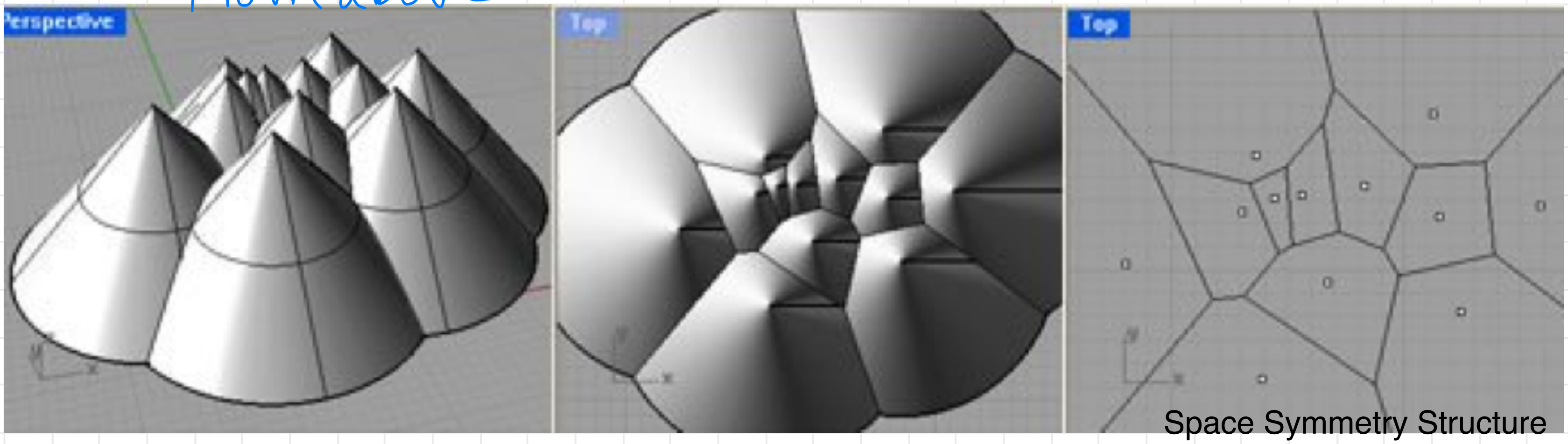
reach a new point ↗ a new parabolic segment appears



a parabolic section vanishes. Our "event list" must include y=k

Another way to visualize Fortune's algorithm

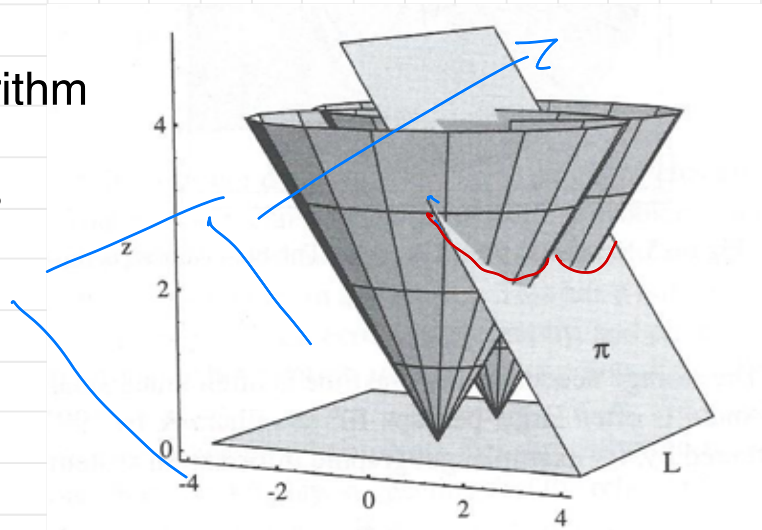
from above



Space Symmetry Structure

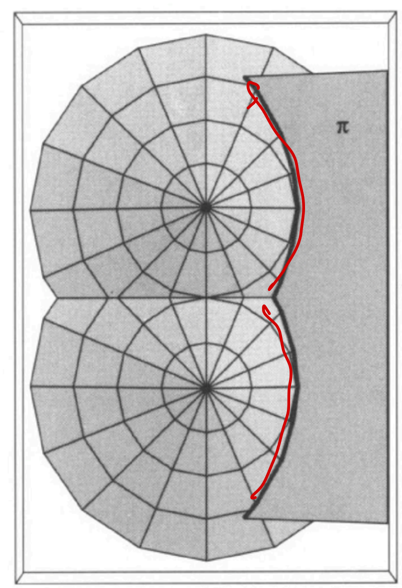
the Voronoi diagram can be viewed as the projection of the upper envelope of cones

and Fortune's algorithm sweeps a plane π across those cones



O'Rourke

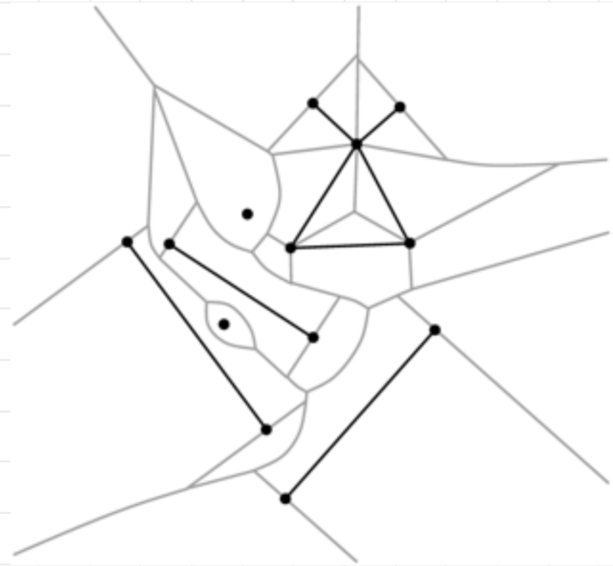
from below.



beach-front

Other versions of Voronoi diagrams

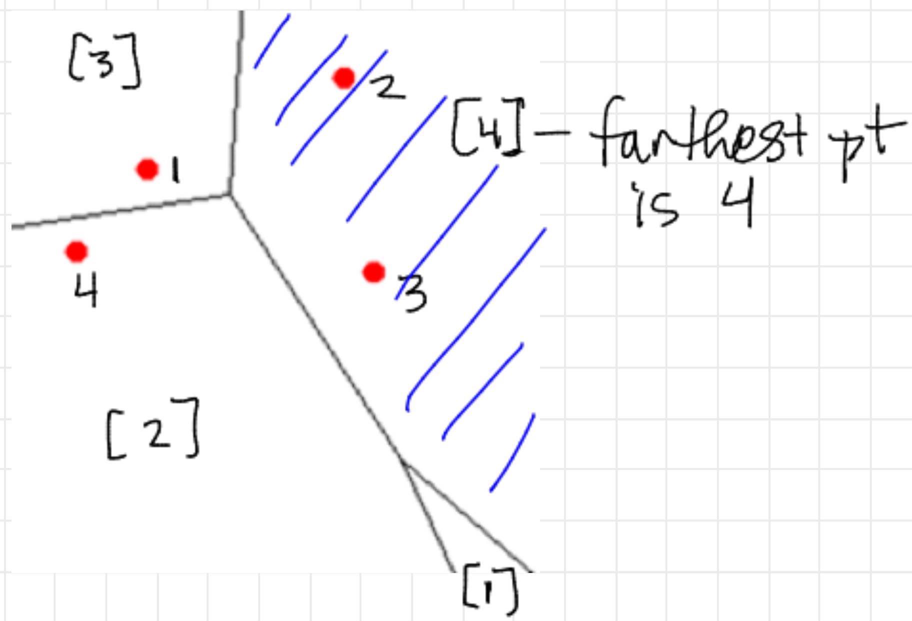
- the sites may be more general than points,
  e.g. line segments, polygons, etc.



CGAL

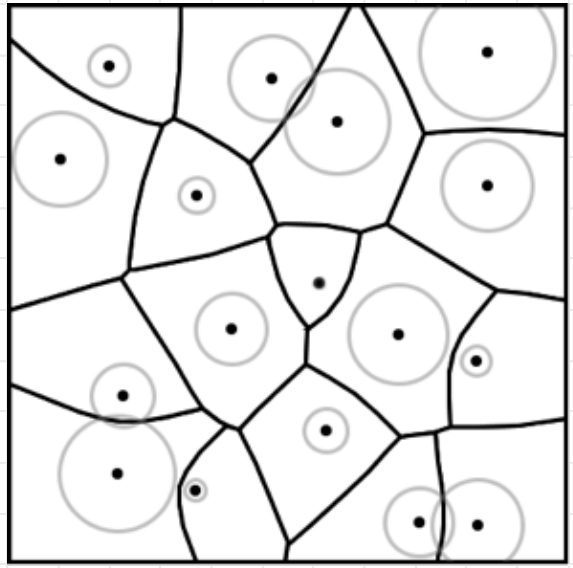
- higher dimensions

- farthest point Voronoi diagrams

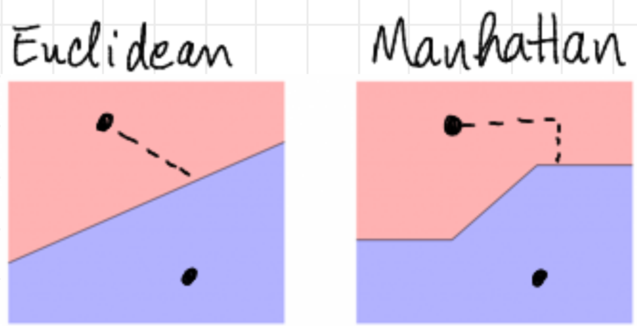


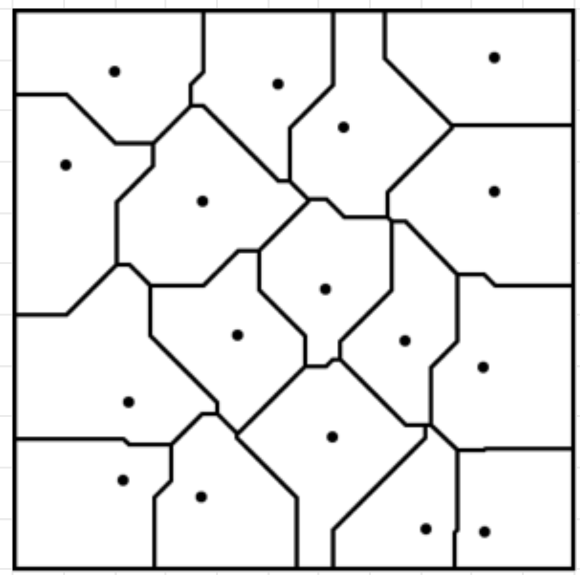
Other versions of Voronoi diagrams

- weighted Voronoi diagrams



- Voronoi diagrams for other distance metrics





Summary

- Voronoi diagram and Delaunay triangulation
- applications to proximity graphs, largest empty circle
- relationship to Convex Hull
- O(n log n) algorithm

References (same as before)

- [CGAA] Chapters 7, 9
- [Zurich notes] Chapters 5, 7 (they start with Delaunay)
- [O'Rourke] Chapter 5
- [Devadoss-O'Rourke] Chapter 4.