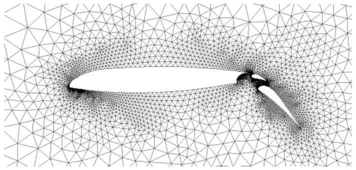
Geometric Algorithms

Reference: Chapters 24-25, Algorithms in C, 2nd Edition, Robert Sedgewick.

Geometric Algorithms

Applications.

- Data mining.
- VLSI design.
- Computer vision.
- Mathematical models.
- Astronomical simulation.
- Geographic information systems.
- Computer graphics (movies, games, virtual reality).
- Models of physical world (maps, architecture, medical imaging).



airflow around an aircraft wing

Reference: http://www.ics.uci.edu/~eppstein/geom.html

History.

- Ancient mathematical foundations.
- Most geometric algorithms less than 25 years old.

Geometric Primitives

Point: two numbers (x, y).

Line: two numbers a and b [ax + by = 1]

any line not through origin

Line segment: two points.

Polygon: sequence of points.

Primitive operations.

- Is a point inside a polygon?
- Compare slopes of two lines.
- Distance between two points.
- Do two line segments intersect?
- Given three points p_1, p_2, p_3, is p_1-p_2-p_3 a counterclockwise turn?

Other geometric shapes.

- Triangle, rectangle, circle, sphere, cone, …
- 3D and higher dimensions sometimes more complicated.

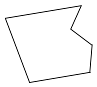
Intuition

Warning: intuition may be misleading.

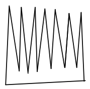
- Humans have spatial intuition in 2D and 3D.
- Computers do not.
- Neither has good intuition in higher dimensions!

Is a given polygon simple?

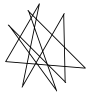
no crossings



1	6	5	8	7	2
7	8	6	4	2	1



1	15	14	13	12	11	10	9	8	7	6	5	4	3	2
1	2	18	4	18	4	19	4	19	4	20	3	20	3	20



1	10	3	7	2	8	8	3	4
6	5	15	1	11	3	14	2	16

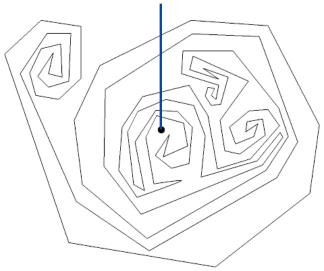
we think of this

algorithm sees this

Polygon Inside, Outside

Jordan curve theorem. [Veblen 1905] Any continuous simple closed curve cuts the plane in exactly two pieces: the inside and the outside.

Is a point inside a simple polygon?



http://www.ics.uci.edu/~eppstein/geom.html

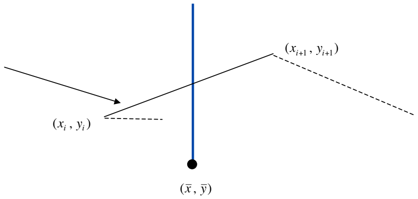
Application. Draw a filled polygon on the screen.

Polygon Inside, Outside: Crossing Number

Does line segment intersect ray?

$$y \;=\; \frac{y_{i+1} - y_i}{x_{i+1} - x_i}\,(x - x_i) \;+\; y_i$$

where $x_i \;\leq\; x \;\leq\; x_{i+1}$

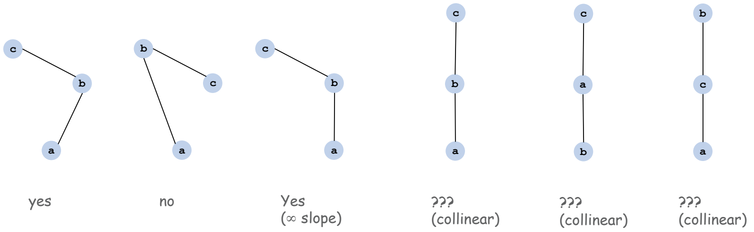


```
public boolean contains(double x0, double y0) {
   int crossings = 0;
   for (int i = 0; i < N; i++) {
      double  slope = (y[i+1] - y[i]) / (x[i+1] - x[i]);
      boolean cond1 = (x[i] <= x0) && (x0 < x[i+1]);
      boolean cond2 = (x[i+1] <= x0) && (x0 < x[i]);
      boolean above = (y0 < slope * (x0 - x[i]) + y[i]);
      if ((cond1 || cond2)  && above) crossings++;
   }
   return (crossings % 2 != 0);
}
```

Implementing CCW

CCW. Given three point a, b, and c, is a-b-c a counterclockwise turn?

- Analog of comparisons in sorting.
- Idea: compare slopes.



Lesson. Geometric primitives are tricky to implement.

- Dealing with degenerate cases.
- Coping with floating point precision.

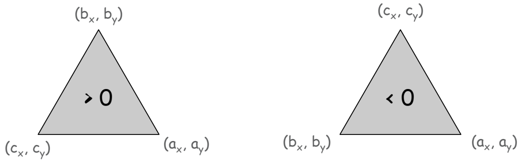
Implementing CCW

CCW. Given three point a, b, and c, is a-b-c a counterclockwise turn?

- Determinant gives twice area of triangle.

$$2 \times Area(a, b, c) \;=\; \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix} \;=\; (b_x - a_x)(c_y - a_y) \;-\; (b_y - a_y)(c_x - a_x)$$

- If area > 0 then a-b-c is counterclockwise.
  If area < 0, then a-b-c is clockwise.
  If area = 0, then a-b-c are collinear.



Immutable Point ADT

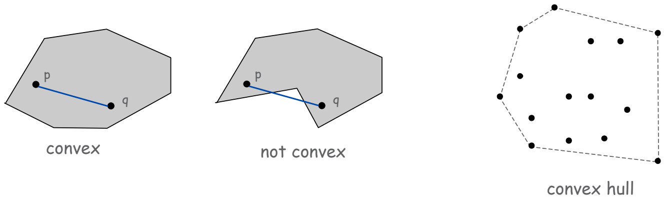
```
public final class Point {
   public final int x;
   public final int y;

   public Point(int x, int y) { this.x = x; this.y = y; }

   public double distanceTo(Point q) {
      return Math.hypot(this.x - q.x, this.y - q.y);
   }

   public static int ccw(Point a, Point b, Point c) {
      double area2 = (b.x-a.x)*(c.y-a.y) - (b.y-a.y)*(c.x-a.x);
      if      (area2 < 0) return -1;
      else if (area2 > 0) return +1;
      else                return  0;
   }

   public static boolean collinear(Point a, Point b, Point c) {
      return ccw(a, b, c) == 0;
   }
}
```

Convex Hull

Convex Hull

A set of points is convex if for any two points p and q in the set, the line segment $\overline{pq}$ is completely in the set.

Convex hull. Smallest convex set containing all the points.



convex

not convex

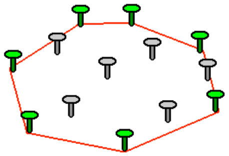
convex hull

Properties.

- "Simplest" shape that approximates set of points.
- Shortest (perimeter) fence surrounding the points.
- Smallest (area) convex polygon enclosing the points.

Mechanical Solution

Mechanical algorithm. Hammer nails perpendicular to plane; stretch elastic rubber band around points.

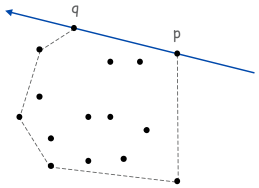


http://www.dfanning.com/math_tips/convexhull_1.gif

Brute Force

Observation 1. Edges of convex hull of P connect pairs of points in P.

Observation 2. Edge $\overrightarrow{pq}$ is on convex hull if all other points are counterclockwise of $\overrightarrow{pq}$.



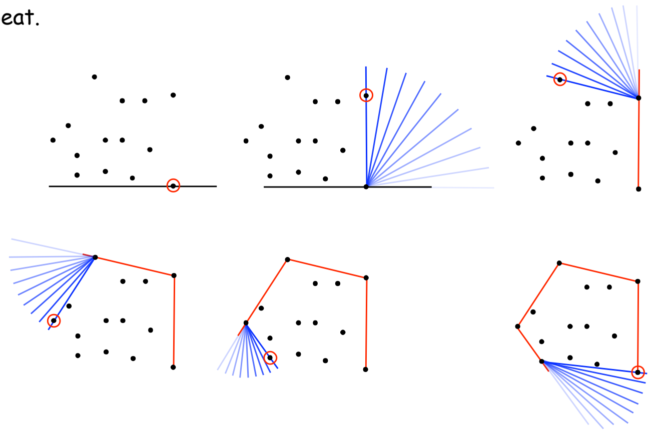
O(N^3) algorithm. For all points p and q in P, check whether $\overrightarrow{pq}$ is an edge of convex hull.

each check requires O(N) ccw calculations, where N is the number of points in P

Package Wrap (Jarvis March)

Package wrap.

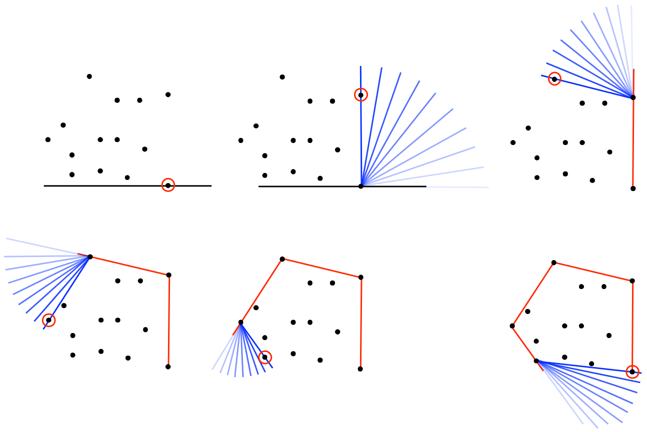
- Start with point with smallest y-coordinate.
- Rotate sweep line around current point in ccw direction.
- First point hit is on the hull.
- Repeat.



Package Wrap (Jarvis March)

Implementation.

- Compute angle between current point and all remaining points.
- Pick smallest angle larger than current angle.
- $\Theta(N)$ per iteration.



How Many Points on the Hull?

Parameters.

- N = number of points.
- h = number of points on the hull.

Package wrap running time. $\Theta(N\,h)$ per iteration.

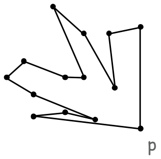
How many points on hull?

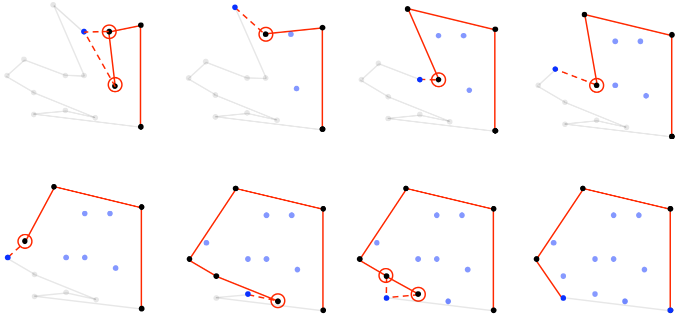
- Worst case: h = N.
- Average case: difficult problems in stochastic geometry.
  - in a disc: $h = N^{1/3}$.
  - in a convex polygon with O(1) edges: h = log N.

Graham Scan: Example

Graham scan.

- Choose point p with smallest y-coordinate.
- Sort points by polar angle with p to get simple polygon.
- Consider points in order, and discard those that would create a clockwise turn.





Graham Scan: Example

Implementation.

- Input: `p[1]`, `p[2]`, …, `p[N]` are points.
- Output: `M` and rearrangement so that `p[1]`, …, `p[M]` is convex hull.

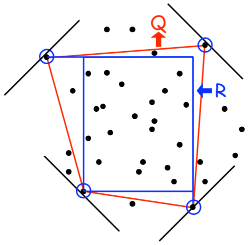
```
// preprocess so that p[1] has smallest y-coordinate
// sort by angle with p[1]

points[0] = points[N];  // sentinel
int M = 2;
for (int i = 3; i <= N; i++) {
   while (Point.ccw(p[M-1], p[M], p[i]) <= 0) {
      M--;
   }         discard points that would create clockwise turn
   M++;
   swap(points, M, i);
}                   add i to putative hull
```

Running time. O(N log N) for sort and $\underbrace{\text{O(N) for rest.}}_{\text{why?}}$

Quick Elimination

Quick elimination.

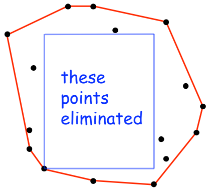
- Choose a quadrilateral Q or rectangle R with 4 points as corners.
- If point is inside, can eliminate.
  - 4 ccw tests for quadrilateral
  - 4 comparisons for rectangle



Three-phase algorithm

- Pass through all points to compute R.
- Eliminate points inside R.
- Find convex hull of remaining points.

Practice. Can eliminate almost all points in linear time.



Convex Hull Algorithms Costs Summary

Algorithm	Running Time
Package wrap	N h
Graham scan	N log N
Quickhull	N log N
Mergehull	N log N
Sweep line	N log N
Quick elimination	N†
Best in theory	N log h

output sensitive running time

asymptotic cost to find h-point hull in N-point set

† assumes "reasonable" point distribution

Convex Hull: Lower Bound

Models of computation.

- Comparison based: compare coordinates.
  (impossible to compute convex hull in this model of computation)

```
(a.x < b.x) || ((a.x == b.x) && (a.y < b.y)))
```

- Quadratic decision tree model: compute any quadratic function of the coordinates and compare against 0.

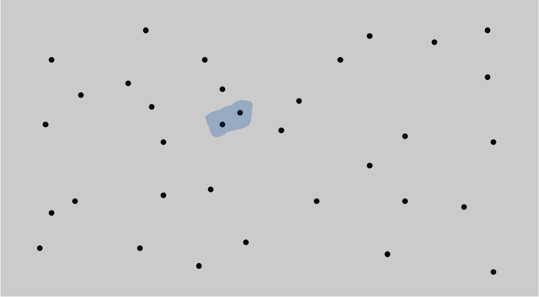
```
(a.x*b.y - a.y*b.x + a.y*c.x - a.x*c.y + b.x*c.y - c.x*b.y) < 0
```

Theorem. [Andy Yao, 1981] In quadratic decision tree model, any convex hull algorithm requires $\Omega(N \log N)$ ops.

even if hull points are not required to be output in counterclockwise order

higher degree polynomial tests don't help either [Ben-Or, 1983]

Closest Pair of Points



Closest Pair of Points

Closest pair. Given N points in the plane, find a pair with smallest Euclidean distance between them.

Fundamental geometric primitive.

- Graphics, computer vision, geographic information systems, molecular modeling, air traffic control.
- Special case of nearest neighbor, Euclidean MST, Voronoi.

fast closest pair inspired fast algorithms for these problems

Brute force. Check all pairs of points p and q with $\Theta(N^2)$ distance calculations.

1-D version. O(N log N) easy if points are on a line.

Assumption. No two points have same x coordinate.

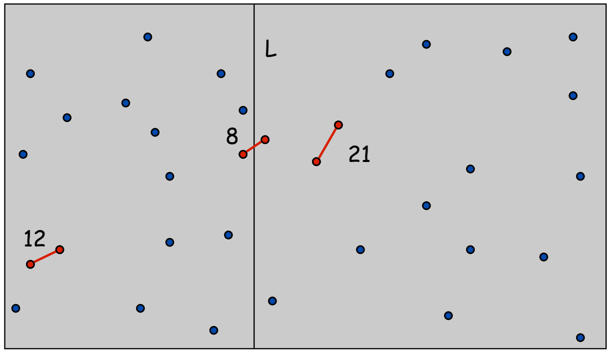
to make presentation cleaner

Closest Pair of Points

Algorithm.

- Divide: draw vertical line L so that roughly $\frac{1}{2}$N points on each side.
- Conquer: find closest pair in each side recursively.
- Combine: find closest pair with one point in each side.
- Return best of 3 solutions.

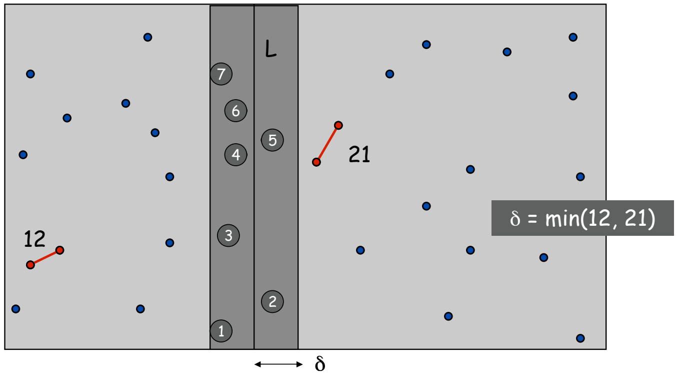
seems like $\Theta(N^2)$



Closest Pair of Points

Find closest pair with one point in each side, assuming that distance < δ.

- Observation: only need to consider points within δ of line L.
- Sort points in 2δ-strip by their y coordinate.
- Only check distances of those within 11 positions in sorted list!



Closest Pair of Points

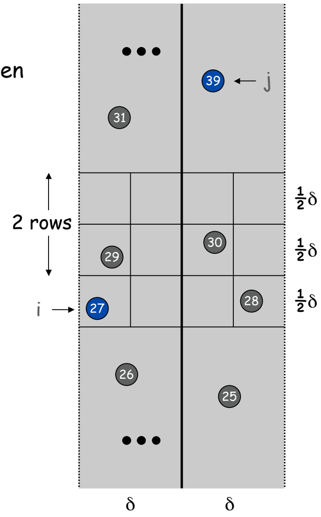
Def. Let s_i be the point in the 2δ-strip, with the i^{th} smallest y-coordinate.

Claim. If $|i - j| \ge 12$, then the distance between s_i and s_j is at least δ.

Pf.

- No two points lie in same $\frac{1}{2}\delta$-by-$\frac{1}{2}\delta$ box.
- Two points at least 2 rows apart have distance $\ge 2(\frac{1}{2}\delta)$. ▪

Fact. Still true if we replace 12 with 7.



Closest Pair Algorithm

```
Closest-Pair(p_1, …, p_n) {
   Compute separation line L such that half the points        O(N log N)
   are on one side and half on the other side.

   δ_1 = Closest-Pair(left half)                              2T(N / 2)
   δ_2 = Closest-Pair(right half)
   δ  = min(δ_1, δ_2)

   Delete all points further than δ from separation line L    O(N)

   Sort remaining points by y-coordinate.                     O(N log N)

   Scan points in y-order and compare distance between        O(N)
   each point and next 11 neighbors. If any of these
   distances is less than δ, update δ.

   return δ.
}
```

Closest Pair of Points: Analysis

Running time.

$$T(N) \le 2T\left(N/2\right) + O(N \log N) \;\Rightarrow\; T(N) = O(N \log^2 N)$$

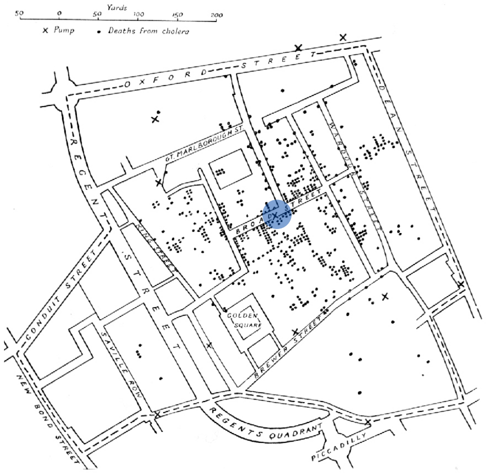
avoid sorting by y-coordinate from scratch

Upper bound. Can be improved to O(N log N).

Lower bound. In quadratic decision tree model, any algorithm for closest pair requires Ω(N log N) steps.

Nearest Neighbor

1854 Cholera Outbreak, Golden Square, London



http://content.answers.com/main/content/wp/en/c/c7/Snow-cholera-map.jpg

Nearest Neighbor

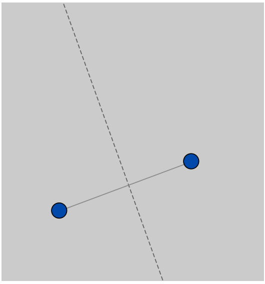
Input. N Euclidean points.

Nearest neighbor problem. Given a query point p, which one of original N points is closest to p?

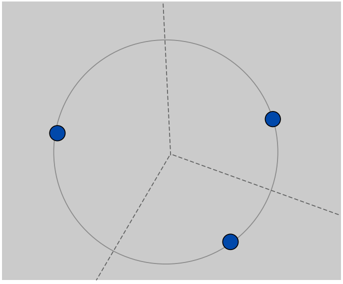
Algorithm	Preprocess	Query
Brute	1	N
Goal	N log N	log N

Voronoi Diagram

Voronoi region. Set of all points closest to a given point.
Voronoi diagram. Planar subdivision delineating Voronoi regions.
Fact. Voronoi edges are perpendicular bisector segments.



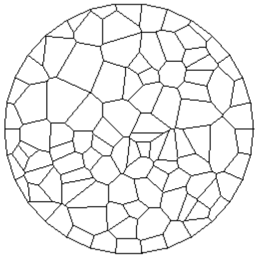
Voronoi of 2 points
(perpendicular bisector)

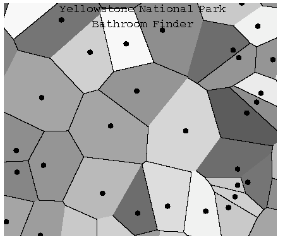


Voronoi of 3 points
(passes through circumcenter)

Voronoi Diagram

Voronoi region. Set of all points closest to a given point.
Voronoi diagram. Planar subdivision delineating Voronoi regions.
Fact. Voronoi edges are perpendicular bisector segments.





Quintessential nearest neighbor data structure.

Voronoi Diagram: Applications

Toxic waste dump problem. N homes in a region. Where to locate nuclear power plant so that it is far away from any home as possible?

↖ looking for largest empty circle
(center must lie on Voronoi diagram)

Path planning. Circular robot must navigate through environment with N obstacle points. How to minimize risk of bumping into a obstacle?

↖ robot should stay on Voronoi diagram of obstacles

Reference: J. O'Rourke. Computational Geometry.

Voronoi Diagram: More Applications

Anthropology. Identify influence of clans and chiefdoms on geographic regions.
Astronomy. Identify clusters of stars and clusters of galaxies.
Biology, Ecology, Forestry. Model and analyze plant competition.
Cartography. Piece together satellite photographs into large "mosaic" maps.
Crystallography. Study Wigner-Setiz regions of metallic sodium.
Data visualization. Nearest neighbor interpolation of 2D data.
Finite elements. Generating finite element meshes which avoid small angles.
Fluid dynamics. Vortex methods for inviscid incompressible 2D fluid flow.
Geology. Estimation of ore reserves in a deposit using info from bore holes.
Geo-scientific modeling. Reconstruct 3D geometric figures from points.
Marketing. Model market of US metro area at individual retail store level.
Metallurgy. Modeling "grain growth" in metal films.
Physiology. Analysis of capillary distribution in cross-sections of muscle tissue.
Robotics. Path planning for robot to minimize risk of collision.
Typography. Character recognition, beveled and carved lettering.
Zoology. Model and analyze the territories of animals.

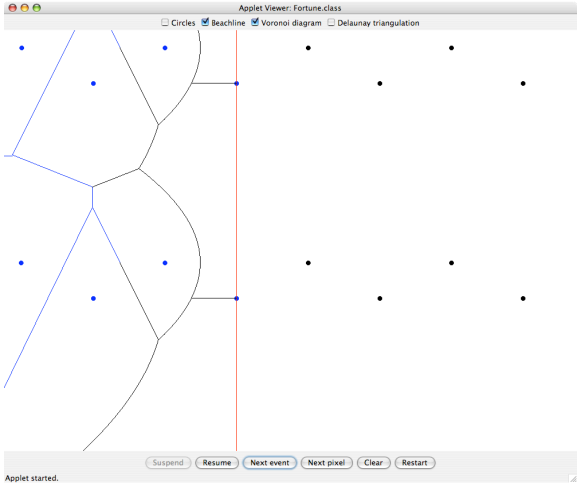
References: http://voronoi.com, http://www.ics.uci.edu/~eppstein/geom.html

Scientific Rediscoveries

Year	Discoverer	Discipline	Name
1644	Descartes	Astronomy	"Heavens"
1850	Dirichlet	Math	Dirichlet tesselation
1908	Voronoi	Math	Voronoi diagram
1909	Boldyrev	Geology	area of influence polygons
1911	Thiessen	Meteorology	Thiessen polygons
1927	Niggli	Crystallography	domains of action
1933	Wigner-Seitz	Physics	Wigner-Seitz regions
1958	Frank-Casper	Physics	atom domains
1965	Brown	Ecology	area of potentially available
1966	Mead	Ecology	plant polygons
1985	Hoofd et al.	Anatomy	capillary domains

Reference: Kenneth E. Hoff III

Fortune's Algorithm



Applet Viewer: Fortune.class

☐ Circles ☑ Beachline ☑ Voronoi diagram ☐ Delaunay triangulation

Applet started.

http://www.diku.dk/hjemmesider/studerende/duff/Fortune

Fortune's Algorithm

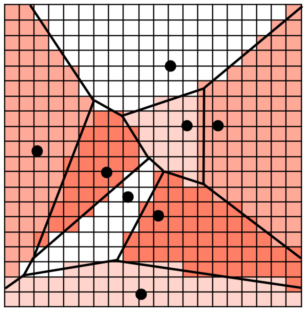
Fortune's algorithm. Sweep-line algorithm can be implemented in O(N log N) time.

but very tricky to get right due to degeneracy and floating point!

Algorithm	Preprocess	Query
Brute	1	N
Fortune	N log N	log N

Discretized Voronoi

Discretized Voronoi. Solve nearest neighbor problem on an N-by-N grid.

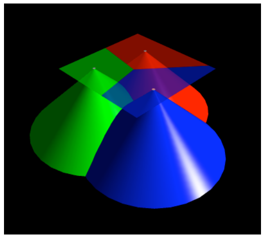


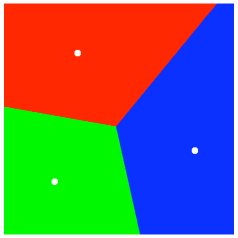
Brute force. For each grid cell, maintain closest point. When adding a new point to Voronoi, update N^2 cells.

Hoff's Algorithm

Hoff's algorithm. Align apex of a right circular cone with sites.

- Minimum envelope of cone intersections projected onto plane is the Voronoi diagram.
- View cones in different colors ⇒ render Voronoi.



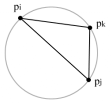


Implementation. Draw cones using standard graphics hardware!

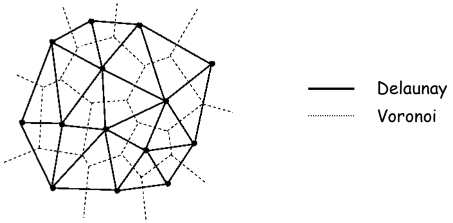
http://www.cs.unc.edu/~geom/voronoi/siggraph_paper/voronoi.pdf

Delaunay Triangulation

Delaunay triangulation. Triangulation of N points such that no point is inside circumcircle of any other triangle.



Fact 0. It exists and is unique (assuming no degeneracy).
Fact 1. Dual of Voronoi (connect adjacent points in Voronoi diagram).
Fact 2. No edges cross ⇒ $O(N)$ edges.
Fact 3. Maximizes the minimum angle for all triangular elements.
Fact 4. Boundary of Delaunay triangulation is convex hull.
Fact 5. Shortest Delaunay edge connects closest pair of points.



Summary

Summary. Many fundamental geometric problems require ingenuity to solve large instances.

Problem	Brute	Cleverness
convex hull	N^2	N log N
closest pair	N^2	N log N
furthest pair	N^2	N log N
Delaunay triangulation	N^4	N log N
polygon triangulation	N^2	N

asymptotic time to solve a 2D problem with N points