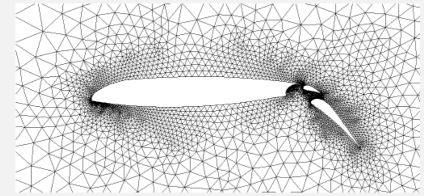
Geometric Primitives

- primitive operations
- convex hull
- closest pair
- voronoi diagram

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

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

History.

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

- primitive operations
- convex hull
- closest pair
- voronoi diagram

Geometric primitives

Point: two numbers (x, y).

Line: two numbers a and b. $[a\,x \;+\; b\,y = 1]$ ← any line not through origin

Line segment: two points.

Polygon: sequence of points.

Primitive operations.

- Is a polygon simple?
- Is a point inside a polygon?
- Do two line segments intersect?
- What is Euclidean distance between two points?
- Given three points p_1, p_2, and p_3, is $p_1 \to p_2 \to p_3$ a counterclockwise turn?

Other geometric shapes.

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

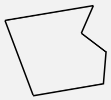
Geometric intuition

Warning: intuition may be misleading.

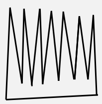
- Humans have spatial intuition in 2D and 3D.
- Computers do not.
- Neither has good intuition in higher dimensions!

Q. Is a given polygon simple? ← no crossings

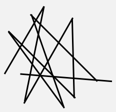
we think of this | algorithm sees this



x	1	6	5	8	7	2
y	7	8	6	4	2	1



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

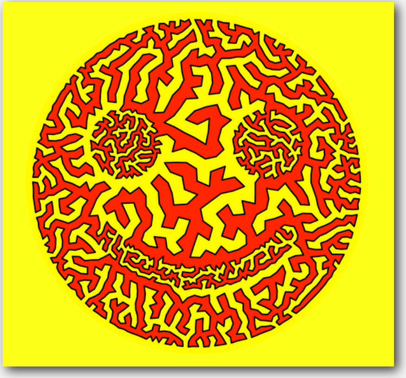


x	1	10	3	7	2	8	8	3	4
y	6	5	15	1	11	3	14	2	16

Polygon inside, outside

Jordan curve theorem. [Jordan 1887, Veblen 1905] Any continuous simple closed curve cuts the plane in exactly two pieces: the inside and the outside.

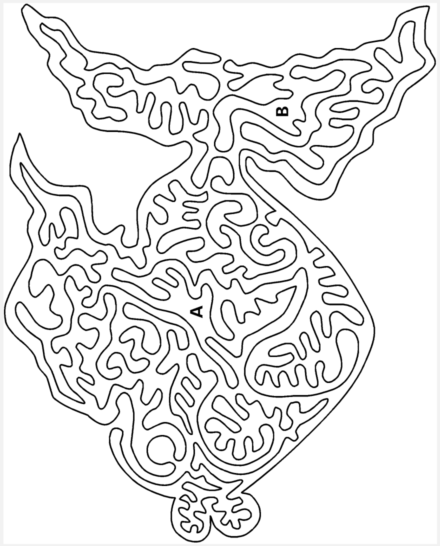
Q. Is a point inside a simple polygon?



Application. Draw a filled polygon on the screen.

Fishy maze

Puzzle. Are A and B inside or outside the maze?

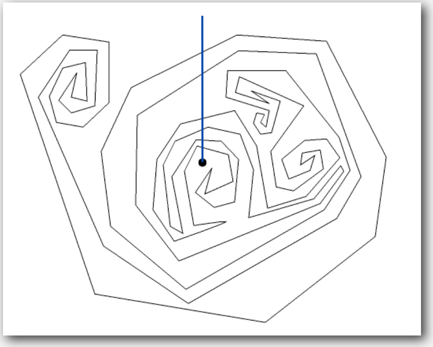


http://britton.disted.camosun.bc.ca/fishmaze.pdf

Polygon inside, outside

Jordan curve theorem. [Jordan 1887, Veblen 1905] Any continuous simple closed curve cuts the plane in exactly two pieces: the inside and the outside.

Q. Is a point inside a simple polygon?



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

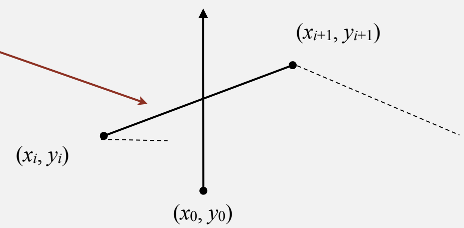
Application. Draw a filled polygon on the screen.

Polygon inside, outside: crossing number

Q. Does line segment intersect ray?

$$y_0 = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x_0 - x_i) + y_i$$

$$x_i \le x_0 \le x_{i+1}$$

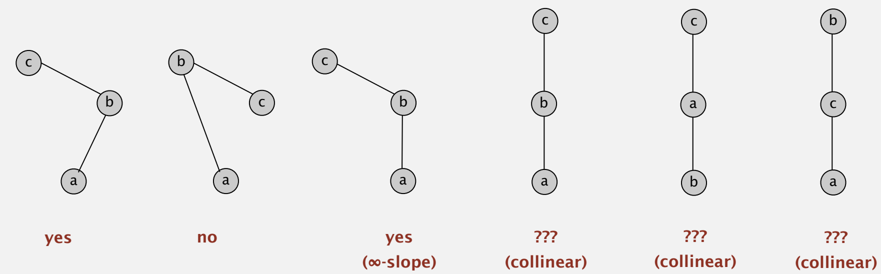



```
public boolean contains(double x0, double y0)
{
    int crossings = 0;
    for (int i = 0; i < N; i++)
    {
       boolean cond1 = (x[i]   <= x0) && (x0 < x[i+1]);
       boolean cond2 = (x[i+1] <= x0) && (x0 < x[i]  );
       double  slope = (y[i+1] - y[i]) / (x[i+1] - x[i]);
       boolean above = (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 $a \to b \to c$ a counterclockwise turn?

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



Lesson. Geometric primitives are tricky to implement.

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

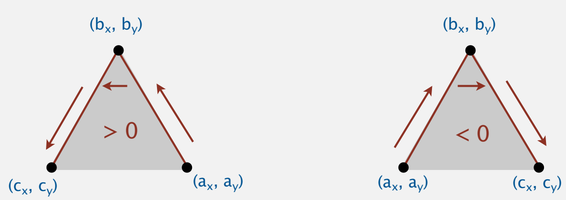
Implementing ccw

CCW. Given three point a, b, and c, is $a \to b \to c$ a counterclockwise turn?

- Determinant gives twice signed 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 $a \to b \to c$ is counterclockwise.
- If area < 0, then $a \to b \to c$ is clockwise.
- If area = 0, then $a \to b \to c$ are collinear.



Immutable point data type

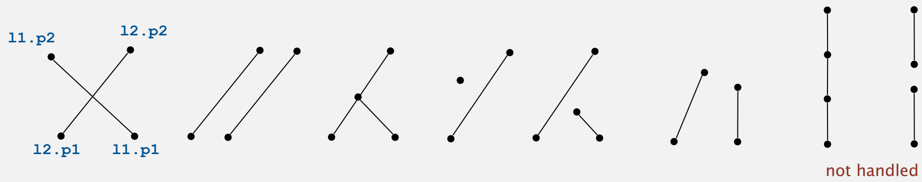
```
public class Point
{
   private final int x;
   private final int y;

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

   public double distanceTo(Point that)
   {
      double dx = this.x - that.x;
      double dy = this.y - that.y;
      return Math.sqrt(dx*dx + dy*dy);
   }

   public static int ccw(Point a, Point b, Point c)
   {
      int 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;  }
}
```

cast to `long` to avoid overflowing an `int`

Sample ccw client: line intersection

Intersect. Given two line segments, do they intersect?

- Idea 1: find intersection point using algebra and check.
- Idea 2: check if the endpoints of one line segment are on different "sides" of the other line segment (4 calls to ccw).



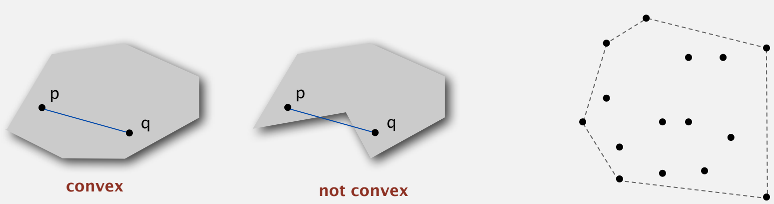

```
public static boolean intersect(LineSegment l1, LineSegment l2)
{
   int test1 = Point.ccw(l1.p1, l1.p2, l2.p1) * Point.ccw(l1.p1, l1.p2, l2.p2);
   int test2 = Point.ccw(l2.p1, l2.p2, l1.p1) * Point.ccw(l2.p1, l2.p2, l1.p2);
   return (test1 <= 0) && (test2 <= 0);
}
```


- primitive operations
- **convex hull**
- closest pair
- voronoi diagram

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.



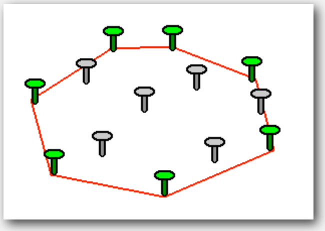
convex not convex convex hull

Properties.

- Shortest perimeter fence surrounding the points.
- Smallest area convex polygon enclosing the points.

Mechanical solution

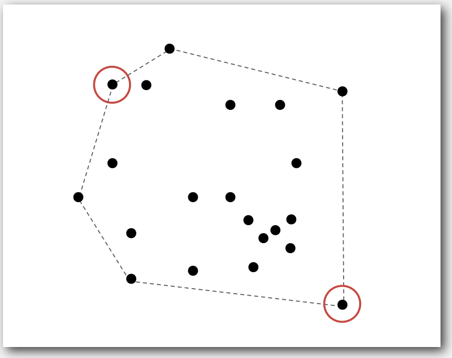
Mechanical convex hull algorithm. Hammer nails perpendicular to plane; stretch elastic rubber band around points.



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

An application: farthest pair

Farthest pair problem. Given N points in the plane, find a pair of points with the largest Euclidean distance between them.

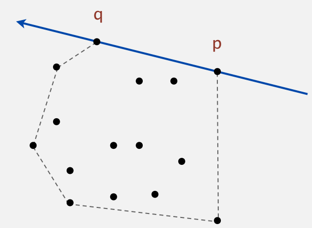


Fact. Farthest pair of points are on convex hull.

Brute-force algorithm

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

Observation 2. $p \rightarrow q$ is on convex hull if all other points are ccw of $\overrightarrow{pq}$.



$O(N^3)$ algorithm. For all pairs of points p and q:

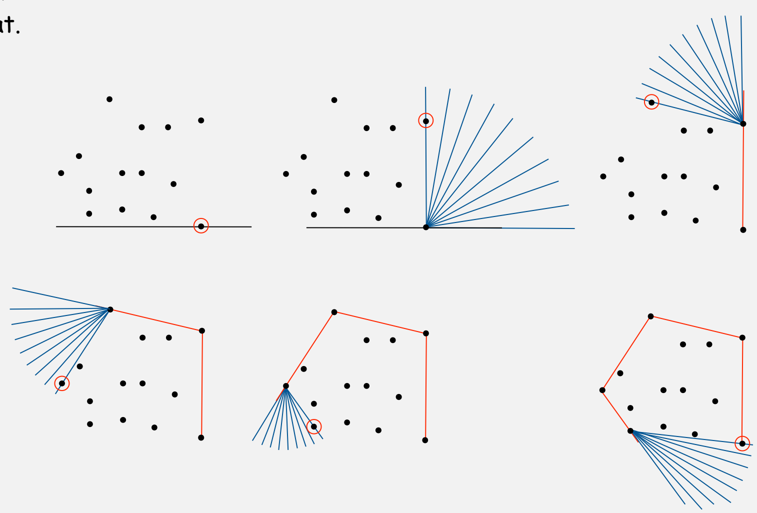
- Compute `Point.ccw(p, q, x)` for all other points x.
- $p \rightarrow q$ is on hull if all values are positive.

Degeneracies. Three (or more) points on a line.

Package wrap (Jarvis march)

Package wrap.

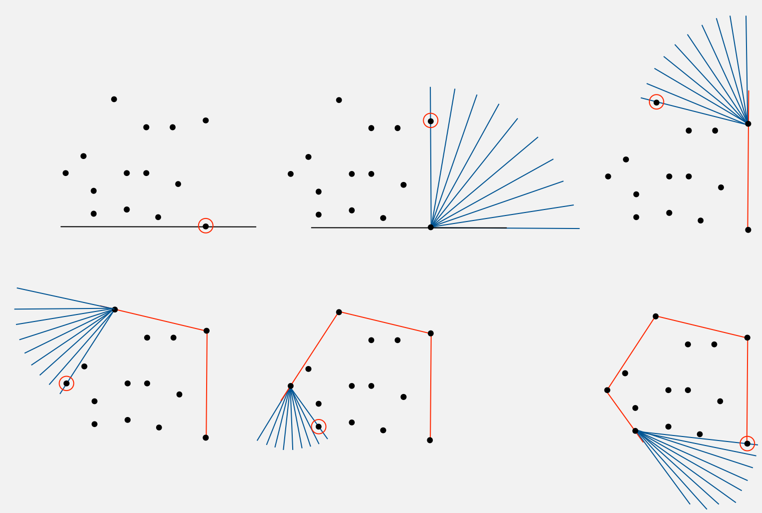
- Start with point with smallest y-coordinate.
- Rotate sweep line around current point in ccw direction.
- First point hit is on the hull.
- Repeat.



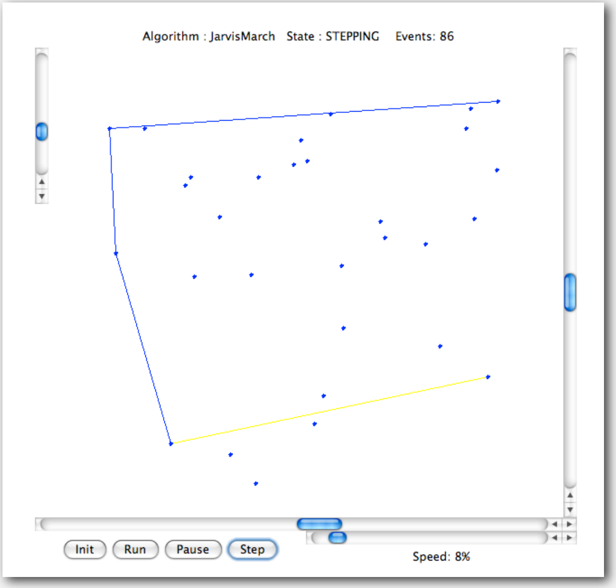
Package wrap (Jarvis march)

Implementation.

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

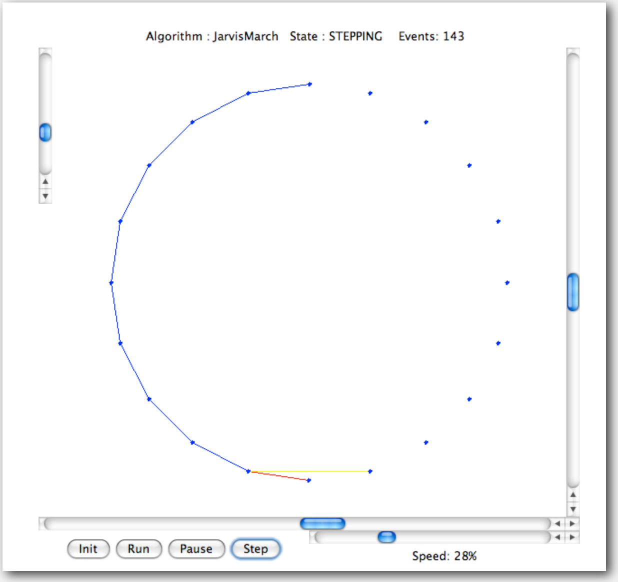


Jarvis march: demo



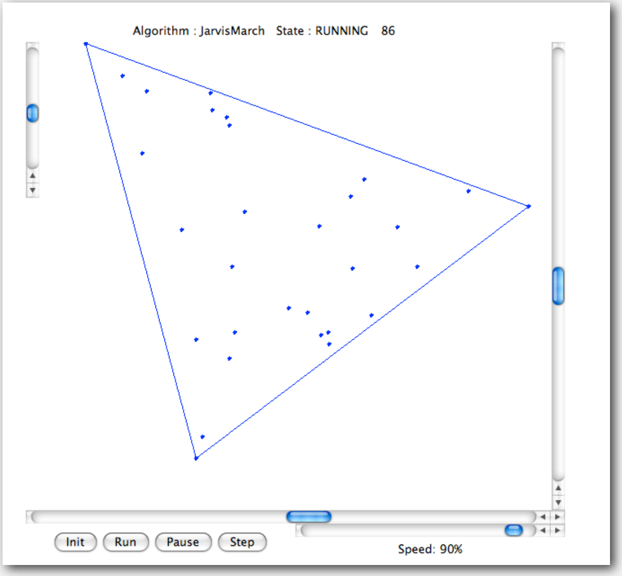
http://www.cs.princeton.edu/courses/archive/fall08/cos226/demo/ah/JarvisMarch.html

Jarvis march: demo



http://www.cs.princeton.edu/courses/archive/fall08/cos226/demo/ah/JarvisMarch.html

Jarvis march: demo



http://www.cs.princeton.edu/courses/archive/fall08/cos226/demo/ah/JarvisMarch.html

How many points on the hull?

Parameters.

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

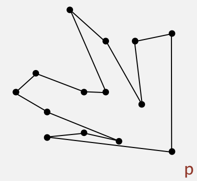
Package wrap running time. $\Theta(N\,h)$.

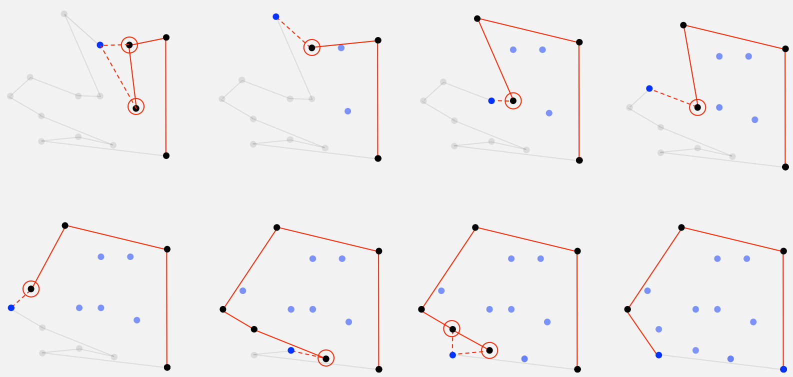
How many points on hull?

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

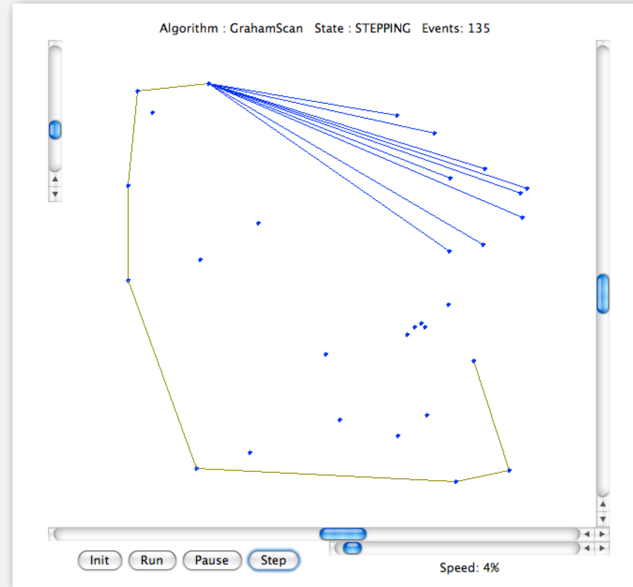
Graham scan

- 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 ccw turn.



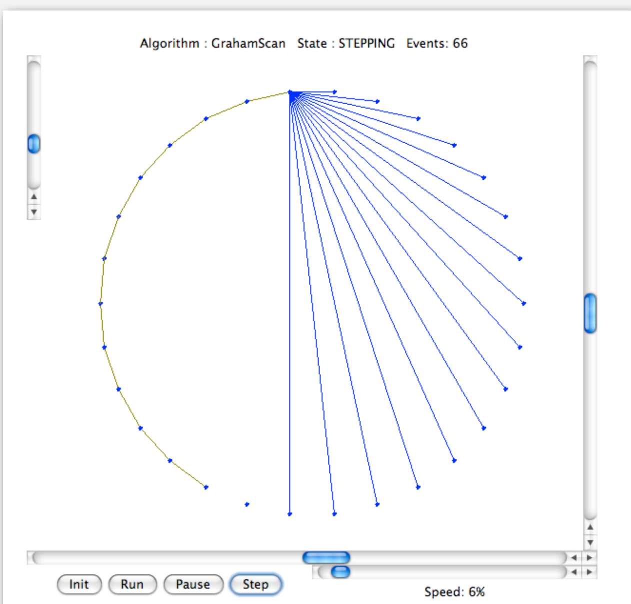


Graham scan: demo



http://www.cs.princeton.edu/courses/archive/fall08/cos226/demo/ah/GrahamScan.html

Graham scan: demo



http://www.cs.princeton.edu/courses/archive/fall08/cos226/demo/ah/GrahamScan.html

Graham scan: implementation

Implementation.

- Input: `p[1]`, `p[2]`, …, `p[N]` are distinct points (not all collinear).
- Output: `M` and rearrangement so that `p[1]`, `p[2]`, …, `p[M]` is convex hull.

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

p[0] = p[N];   // sentinel (p[N] is on hull)
int M = 2;
for (int i = 3; i <= N; i++)
{
   while (Point.ccw(p[M-1], p[M], p[i]) <= 0)
      M--;
   M++;
   swap(p, M, i);  ← add i to putative hull
}
```

discard points that would create clockwise turn

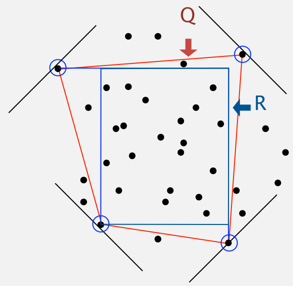
Running time. $N \log N$ for sort and $\underbrace{\text{linear}}_{}$ for rest.

why?

Quick elimination

Quick elimination.

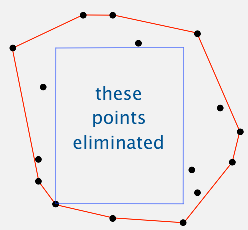
- Choose a quadrilateral Q or rectangle R with 4 points as corners.
- Any point inside cannot be on hull.
  - 4 ccw tests for quadrilateral
  - 4 compares for rectangle



Three-phase algorithm.

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

In practice. Eliminates almost all points in linear time.



Convex hull algorithms costs summary

Order-of-growth of running time to find h-point hull in N-point set.

algorithm	running time	
package wrap	N h	← output sensitive
Graham scan	N log N	
quickhull	N log N	
mergehull	N log N	
sweep line	N log N	
quick elimination	N †	
marriage-before-conquest	N log h	← output sensitive

† **assumes "reasonable" point distribution**

Convex hull: lower bound

Models of computation.

- Compare-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: 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
```

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

Proposition. [Andy Yao, 1981] In quadratic decision tree model, any convex hull algorithm requires $\Omega(N \log N)$ quadratic tests in the worst case.

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

- primitive operations
- convex hull
- **closest pair**
- voronoi diagram

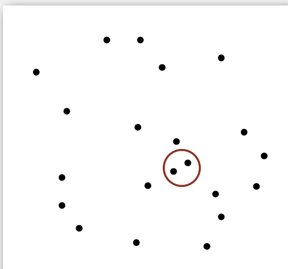
Closest pair

Closest pair problem. Given N points in the plane, find a pair of points with the 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



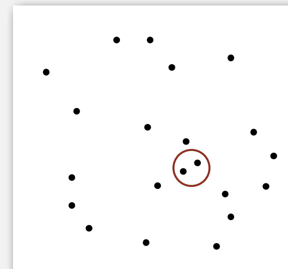
Closest pair

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

Brute force. Check all pairs with N^2 distance calculations.

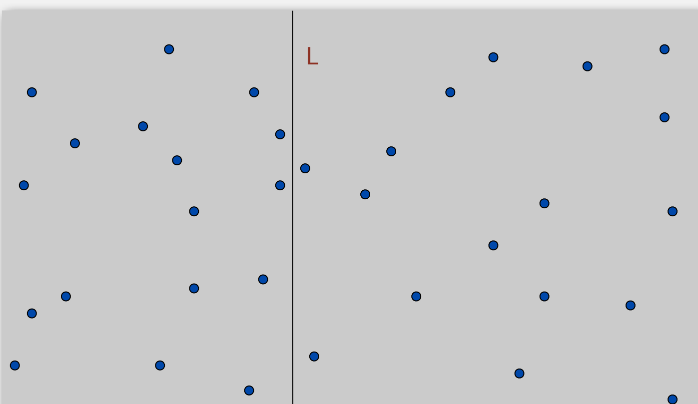
1d version. Easy $N \log N$ algorithm if points are on a line.

Non-degeneracy assumption. No two points have the same x-coordinate.



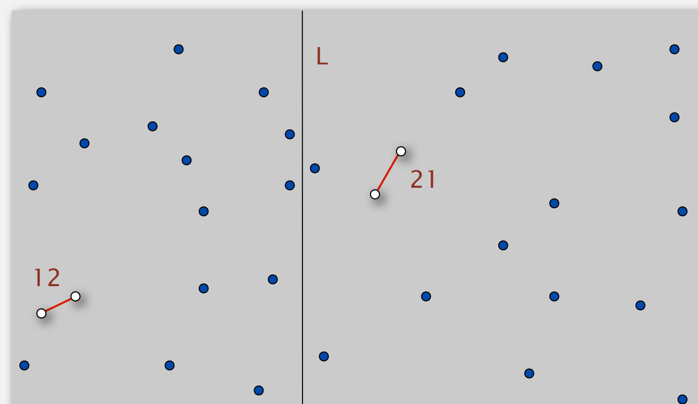
Divide-and-conquer algorithm

- Divide: draw vertical line L so that ~ ½ N points on each side.



Divide-and-conquer algorithm

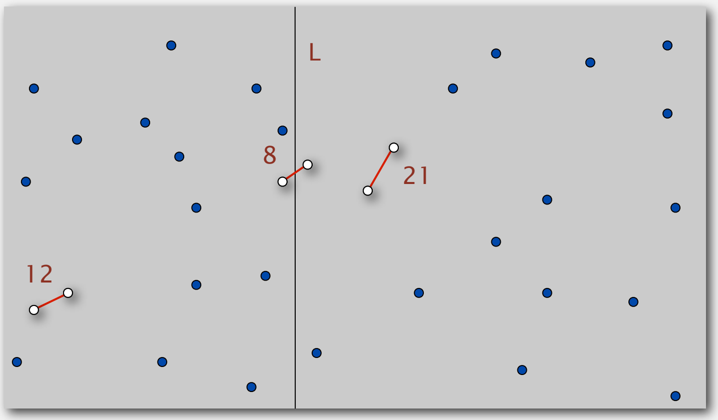
- Divide: draw vertical line L so that ~ ½ N points on each side.
- Conquer: find closest pair in each side recursively.



Divide-and-conquer algorithm

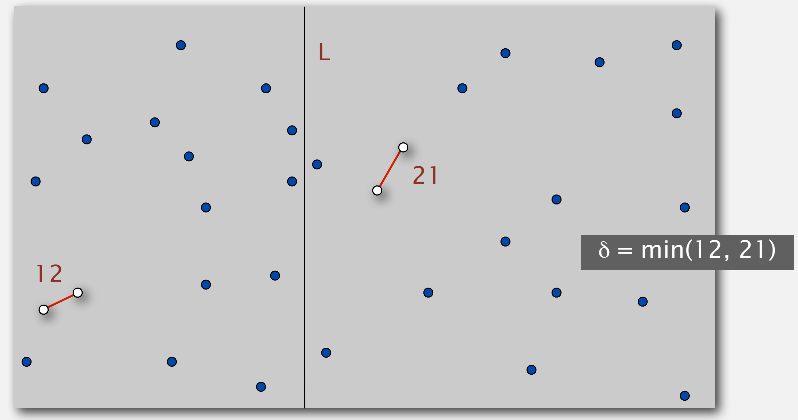
- Divide: draw vertical line L so that ~ ½ N points on each side.
- Conquer: find closest pair in each side recursively.
- Combine: find closest pair with one point in each side.
- Return best of 3 solutions.

seems like $\Theta(N^2)$



How to find closest pair with one point in each side?

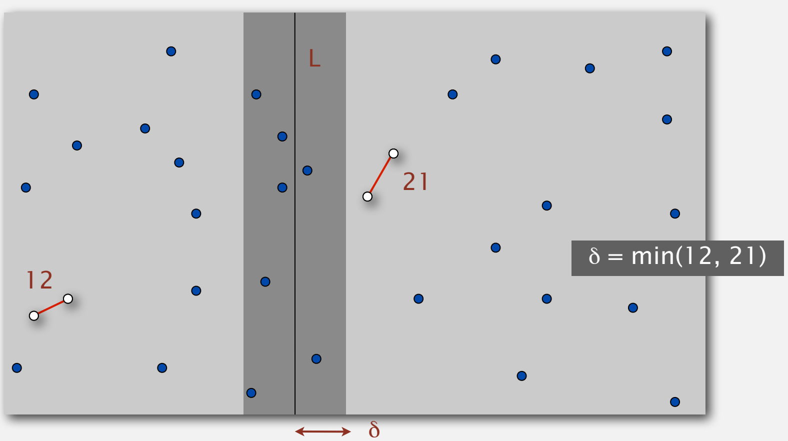
Find closest pair with one point in each side, assuming that distance < δ.



How to find closest pair with one point in each side?

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

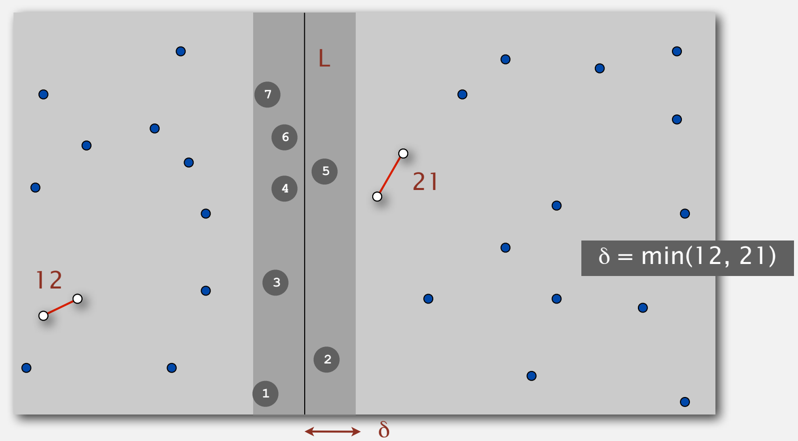
- Observation: only need to consider points within δ of line L.



How to find closest pair with one point in each side?

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 $2\,\delta$-strip by their y-coordinate.

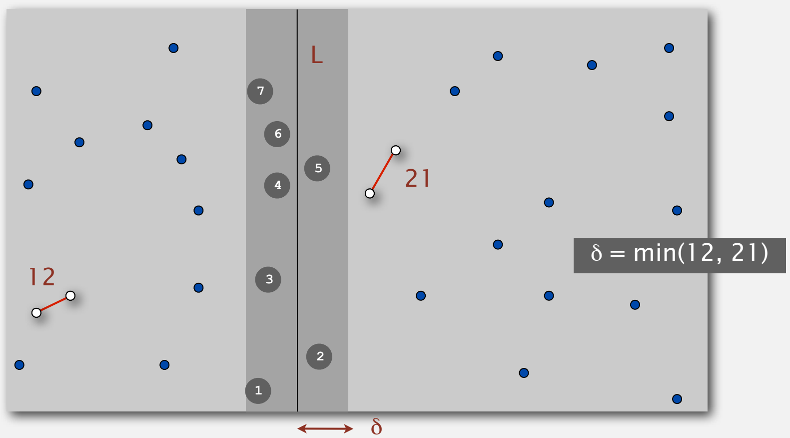


How to find closest pair with one point in each side?

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!

why 11?



How to find closest pair with one point in each side?

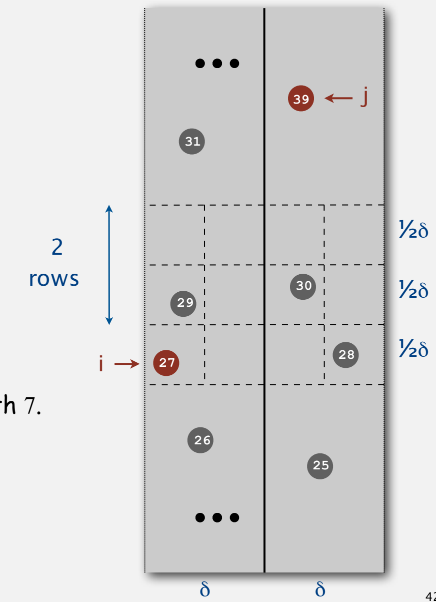
Def. Let s_i be the point in the 2δ-strip, with the i^{th} smallest y-coordinate.

Claim. If $|i - j| \geq 12$, 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 $\geq 2(\frac{1}{2}\delta)$. ▪

Fact. Claim remains true if we replace 12 with 7.



Divide-and-conquer algorithm

```
Closest-Pair(p1, …, pn)
{
   Compute separation line L such that half the points     ← O(N log N)
   are on one side and half on the other side.

   δ1 = Closest-Pair(left half)
   δ2 = Closest-Pair(right half)                           ← 2T(N / 2)
   δ  = min(δ1, δ2)

   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
   each point and next 11 neighbors. If any of these       ← O(N)
   distances is less than δ, update δ.

   return δ.
}
```


Divide-and-conquer algorithm: analysis

Running time recurrence. $T(N) \leq 2T(N/2) + O(N \log N)$.

Solution. $T(N) = O(N(\log N)^2)$.

Remark. Can be improved to $O(N \log N)$.

sort by x- and y-coordinates once
(reuse later to avoid re-sorting)

$(x_1 - x_2)^2 + (y_1 - y_2)^2$

Lower bound. In quadratic decision tree model, any algorithm for closest pair requires $\Omega(N \log N)$ quadratic tests.

- primitive operations
- convex hull
- closest pair
- **voronoi diagram**

1854 cholera outbreak, Golden Square, London

Life-or-death question.

Given a new cholera patient p, which water pump is closest to p's home?

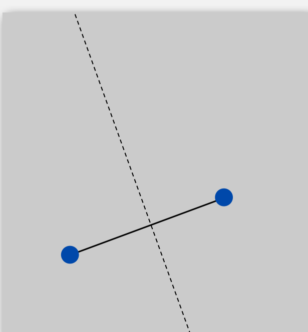
http://content.answers.com/main/content/wp/en/c/c7/Snow-cholera-map.jpg

Voronoi diagram

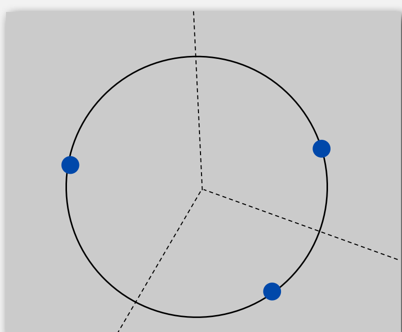
Voronoi region. Set of all points closest to a given point.

Voronoi diagram. Planar subdivision delineating Voronoi regions.

Fact. Voronoi edges are perpendicular bisector segments.



Voronoi of 2 points
(perpendicular bisector)

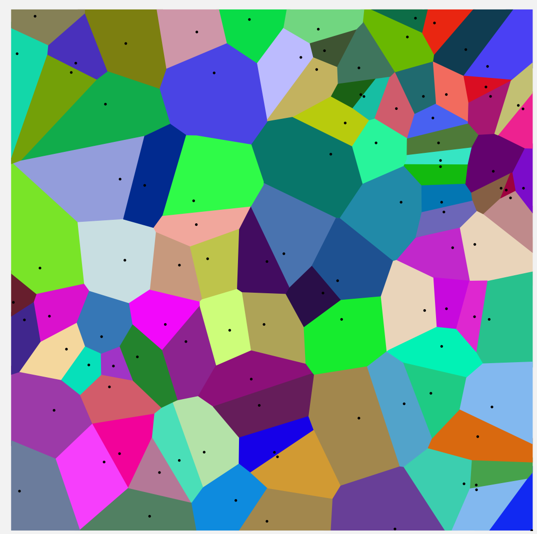


Voronoi of 3 points
(passes through circumcenter)

Voronoi diagram

Voronoi region. Set of all points closest to a given point.

Voronoi diagram. Planar subdivision delineating Voronoi regions.



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.

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

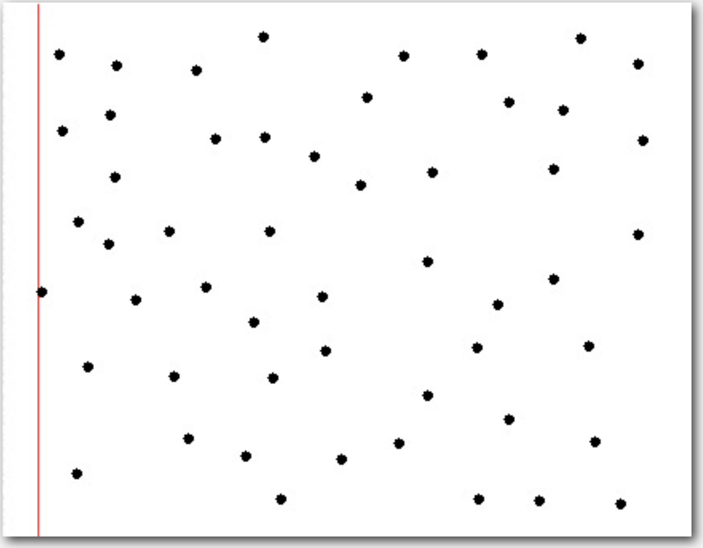
Industrial-strength Voronoi implementation.

- Sweep-line algorithm.
- $O(N \log N)$ time.
- Properly handles degeneracies.
- Properly handles floating-point computations.

algorithm	preprocess	query
brute	1	N
Fortune	N log N	log N

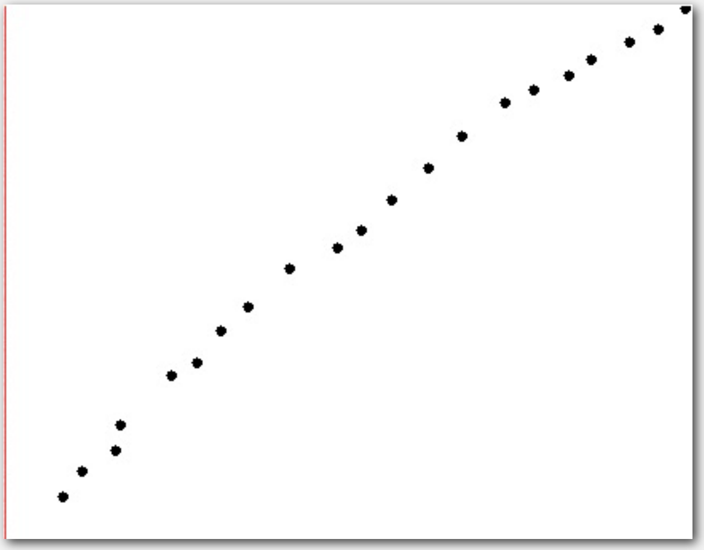
Remark. Beyond scope of this course.

Fortune's algorithm in practice



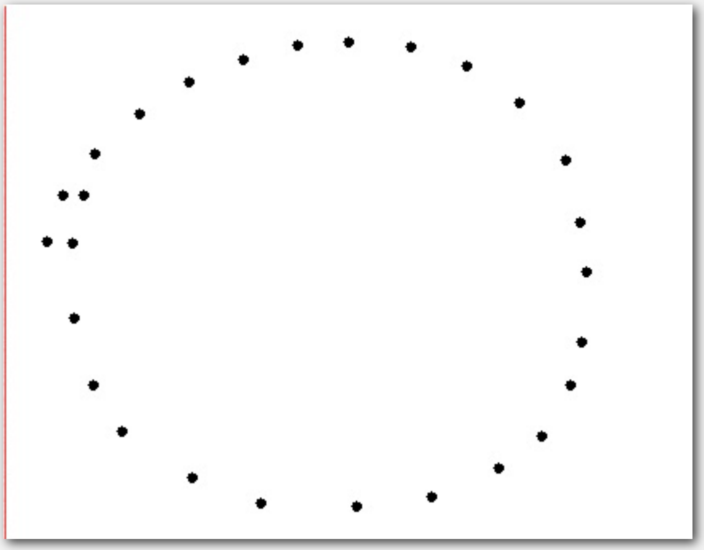
http://www.diku.dk/hjemmesider/studerende/duff/Fortune

Fortune's algorithm in practice



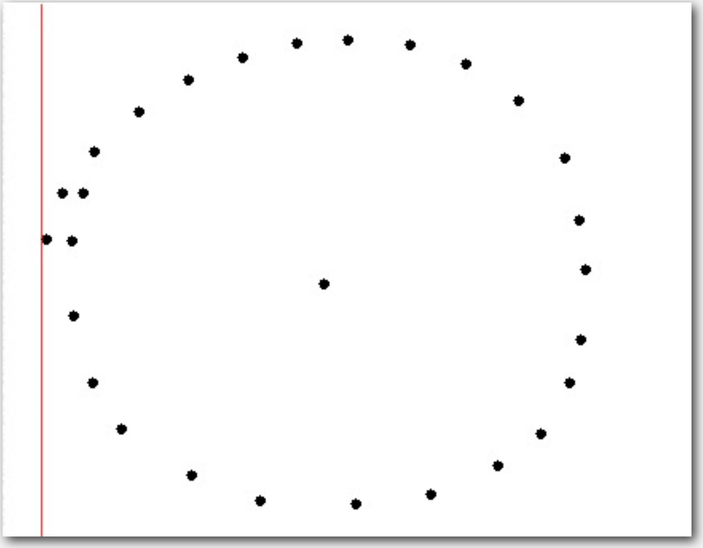
http://www.diku.dk/hjemmesider/studerende/duff/Fortune

Fortune's algorithm in practice



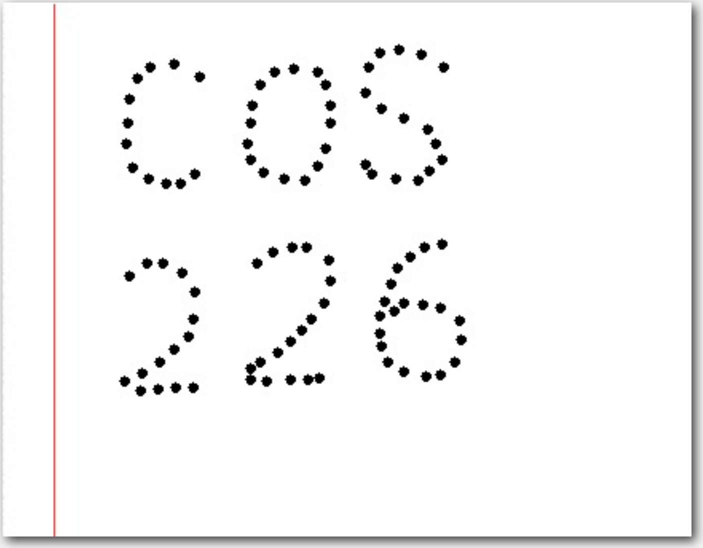
http://www.diku.dk/hjemmesider/studerende/duff/Fortune

Fortune's algorithm in practice



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

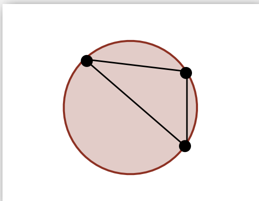
Fortune's algorithm in practice



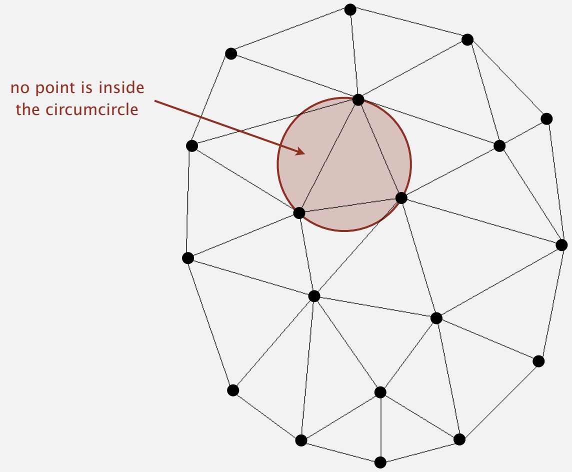
http://www.diku.dk/hjemmesider/studerende/duff/Fortune

Delaunay triangulation

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



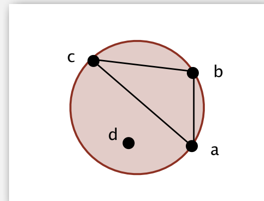
circumcircle of 3 points



Delaunay triangulation of 19 points

Delaunay triangulation

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



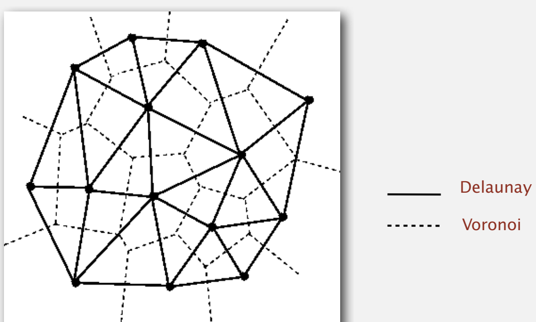
circumcircle of 3 points

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

Proposition. Point d is inside the circumcircle of abc iff $\text{inCircle}(a, b, c, d) < 0$.

Delaunay triangulation properties

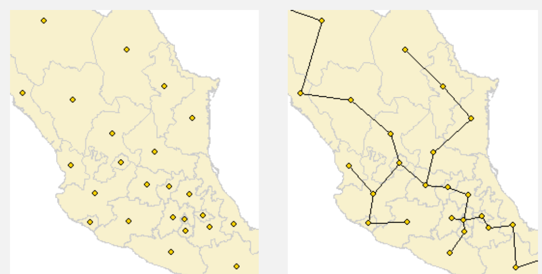
Proposition. It exists and is unique (assuming no degeneracy).
Proposition. Dual of Voronoi (connect adjacent points in Voronoi diagram).
Proposition. No edges cross $\Rightarrow \le 3N$ edges.
Proposition. Boundary of Delaunay triangulation is convex hull.
Proposition. Shortest Delaunay edge connects closest pair of points.
Proposition. Maximizes the minimum angle for all triangular elements.



Degeneracy. No 3 points on a line or 4 on a circle.

Delaunay triangulation application: Euclidean MST

Euclidean MST. Given N points in the plane, find MST connecting them.
[distances between point pairs are Euclidean distances]



Brute force. Compute $N^2 / 2$ distances and run Prim's algorithm.
Ingenuity.
- MST is subgraph of Delaunay triangulation.
- Delaunay has $\le 3N$ edges.
- Compute Delaunay, then use Prim (or Kruskal) to get MST in $N \log N$.

Geometric algorithms summary

Ingenious algorithms enable solution of large instances for numerous fundamental geometric problems.

problem	brute	clever
convex hull	N^2	N log N
farthest pair	N^2	N log N
closest pair	N^2	N log N
Delaunay/Voronoi	N^4	N log N
Euclidean MST	N^2	N log N

order of growth of running time to solve a 2d problem with N points

Note. 3d and higher dimensions test limits of our ingenuity.