Seminar 13

Computational Geometry

TDDD95: APS

Seminar in *Algorithmic Problem Solving*
May 3, 2016

Tommy Färnqvist
Department of Computer and Information Science
Linköping University

Outline

1. **Primitive Operations**
2. **Polygon Area**
3. **Convex Hull**
4. **Closest Pair**

Geometric Primitives

- Point: two numbers (x, y).
- Line: two numbers a and b. $[ax + by = 1]$
- 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 the Euclidean distance between two points?
- Given tree points p_1, p_2, p_3, is $p_1 \to p_2 \to p_3$ a counterclockwise turn?

Geometric Intuition

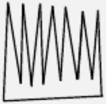
Warning: intution may be misleading.

- 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

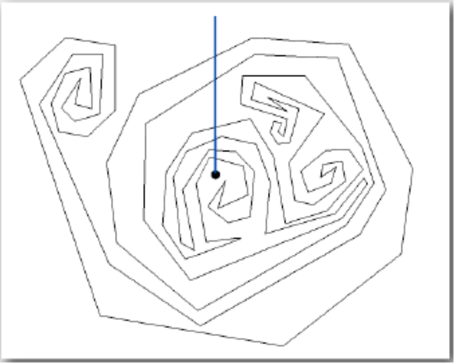
x	1	10	3	7	2	8	8	3	4
y	6	5	15	1	11	3	14	2	16

Polygon inside, outside

Theorem (Jordan curve theorem, Jordan 1886, 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?

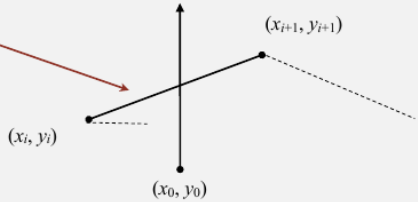


Polygon inside, outside: crossing number

Q. Does line segment intersect ray?

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

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

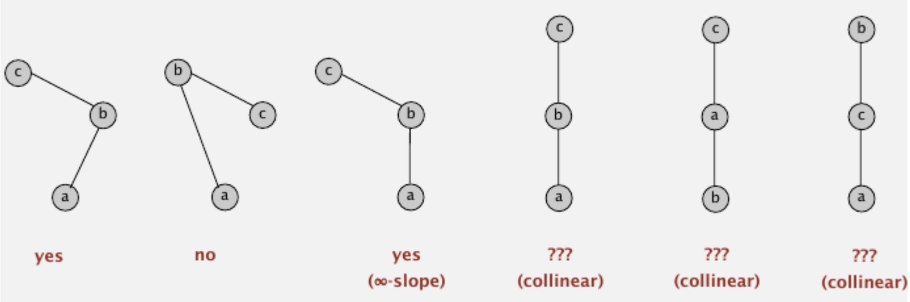



```
public boolean contains(double x0, double y0)
{
    int crossings = 0;
    for (int i = 0; i < N; i++)
    {
       boolean cond1 = (x[i]   <= x0) && (x0 < x[i+1]);
       boolean cond2 = (x[i+1] <= x0) && (x0 < x[i]  );
       double  slope = (y[i+1] - y[i]) / (x[i+1] - x[i]);
       boolean above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

Implementing ccw

CCW. Given three points 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.

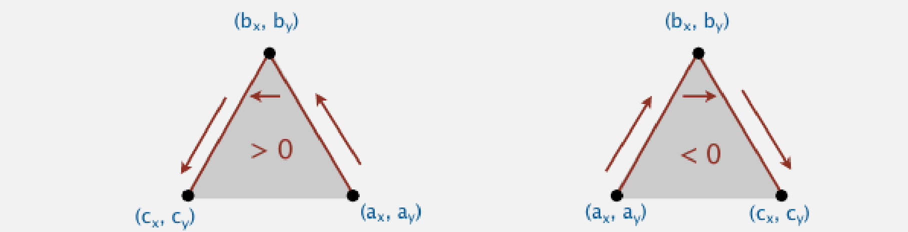
Implementing ccw

CCW. Given three points 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

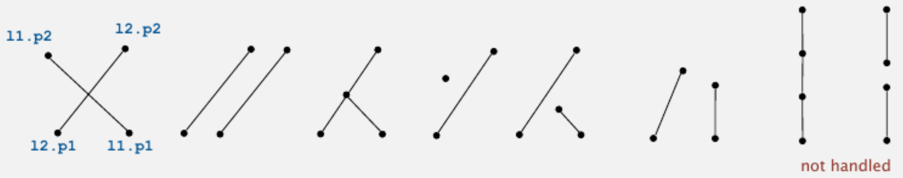
```
public class Point
{
   private final int x;
   private final int y;

   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);
}
```

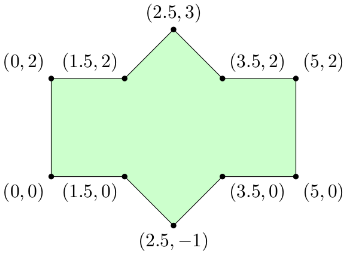
Area of Polygons

Area. The area of any simple polygon is given by the following formula

$$A = \sum_{k=0}^{n} \frac{(x_{k+1} + x_k)(y_{k+1} - y_k)}{2},$$

where n is the number of vertices, (x_k, y_k) is the kth point when labelled in a counterclockwise manner, and $(x_{n+1}, y_{n+1}) = (x_0, y_0)$.

Example



$$\begin{aligned}
A &= \sum_{k=0}^{n} \frac{(x_{k+1} + x_k)(y_{k+1} - y_k)}{2} \\
&= \frac{(1.5+0)(0-0)}{2} + \frac{(2.5+1.5)(-1-0)}{2} + \frac{(3.5+2.5)(0-(-1))}{2} + \frac{(5+3.5)(0-0)}{2} \\
&\quad + \frac{(5+5)(2-0)}{2} + \frac{(3.5+5)(2-2)}{2} + \frac{(2.5+3.5)(3-2)}{2} \\
&\quad + \frac{(1.5+2.5)(2-3)}{2} + \frac{(0+1.5)(2-2)}{2} + \frac{(0+0)(0-2)}{2} \\
&= 0 + (-2) + 3 + 0 + 10 + 0 + 3 + (-2) + 0 + 0 \\
&= 12
\end{aligned}$$

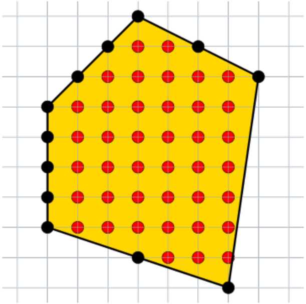
Area of Polygons

Area. The area of any simple polygon with vertices at integer coordinates is given by Pick's theorem:

$$A = I + \frac{R}{2} - 1,$$

where R is the number of integer points on the boundary of the polygon, and I is the number of integer points in the interior of the polygon.

Example

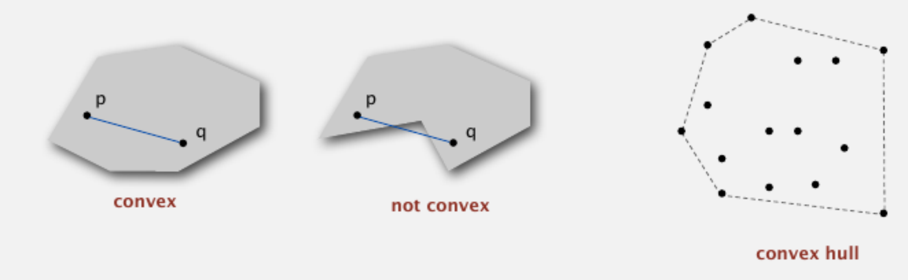


Here, $R = 12$, and $I = 40$, which means the area of the polygon is $40 + 6 - 1 = 45$.

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.

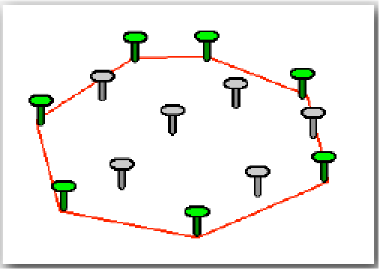
Convex hull. Smallest convex set containing all the points.



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

Mechanical Solution

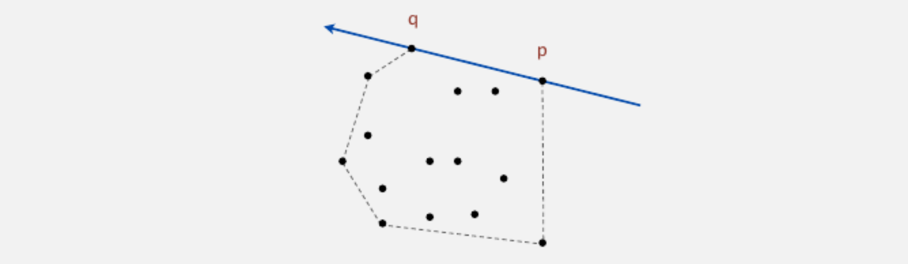
Mechanical convex hull algorithm. Hammer nails perpendicular to the plane; stretch elastic rubber band around points.



Brute-force Algorithm

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

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



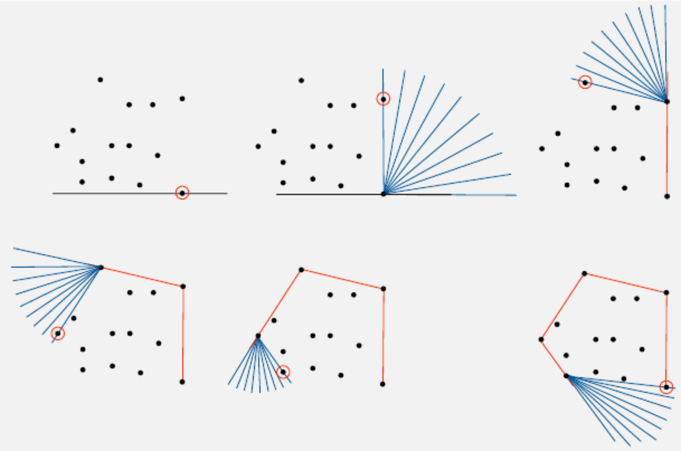
$\mathcal{O}(N^3)$ algorithm. For all pairs of points p and q:

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

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

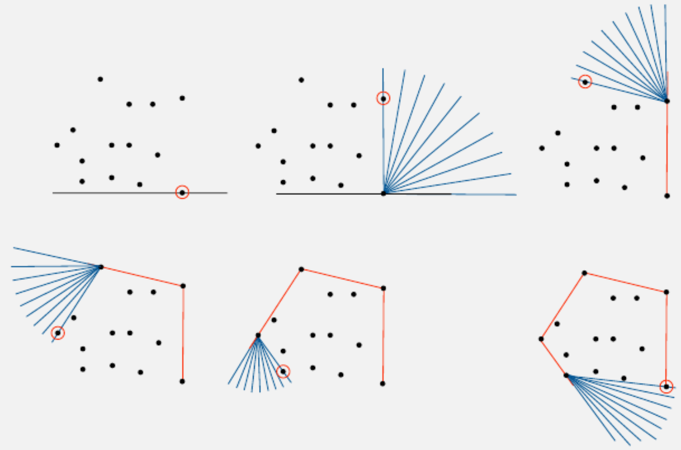
Package Wrap (Jarvis March)

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



Package Wrap (Jarvis March)

- 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.

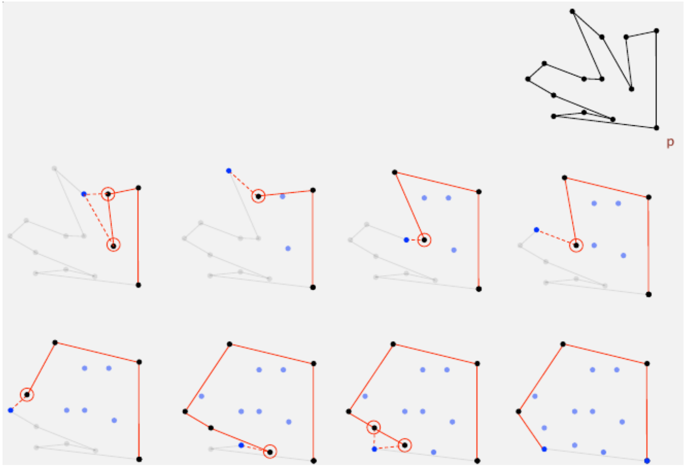
Package wrap running time. $\Theta(Nh)$.

How many points on the 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 $\mathcal{O}(1)$ 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 clockwise turn.



Graham Scan: 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

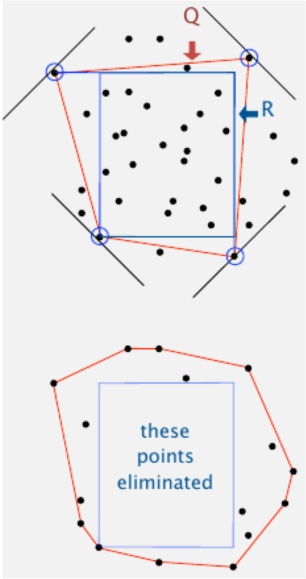
discard points that would create clockwise turn

Running time. $N \log N$ for sort and linear for rest. (Why?)

Quick Elemination

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.

Closest Pair

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

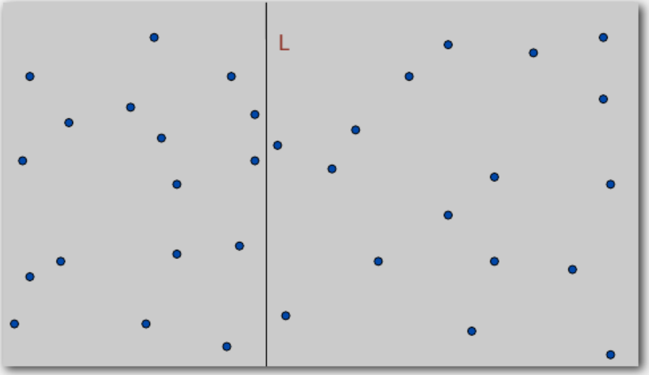
Brute force. Check all pairs with N^2 distance calculations.

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

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

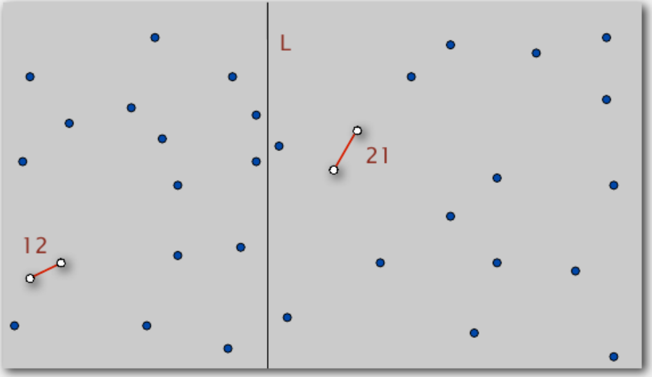
Divide-and-conquer Algorithm

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



Divide-and-conquer Algorithm

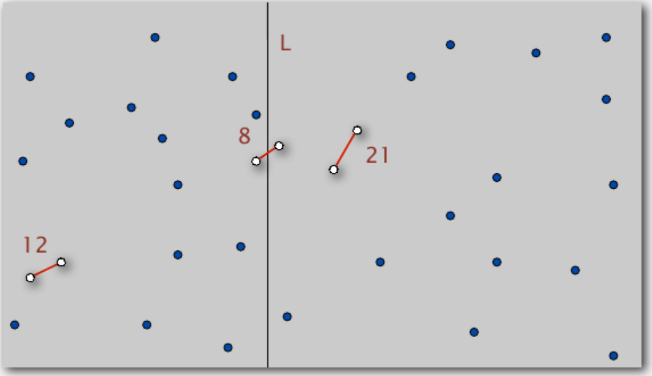
- Divide: draw vertical line L so that ~ ½ N points on each side.
- Conquer: find closest pair in each side recursively.



Divide-and-conquer Algorithm

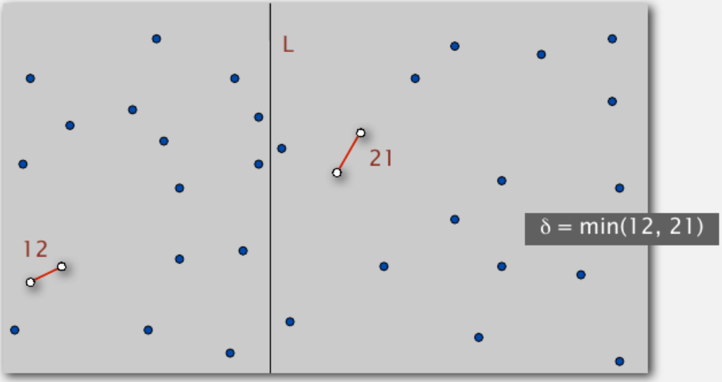
- Divide: draw vertical line L so that $\sim \frac{1}{2} N$ points on each side.
- Conquer: find closest pair in each side recursively.
- Combine: find closest pair with one point in each side.
- Return best of 3 solutions.

seems like $\Theta(N^2)$



Divide-and-conquer Algorithm

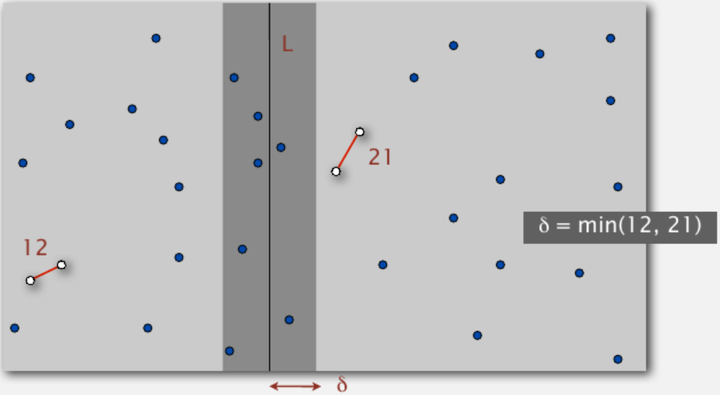
Find closest pair with one point in each side, assuming that distance < δ.



Divide-and-conquer Algorithm

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

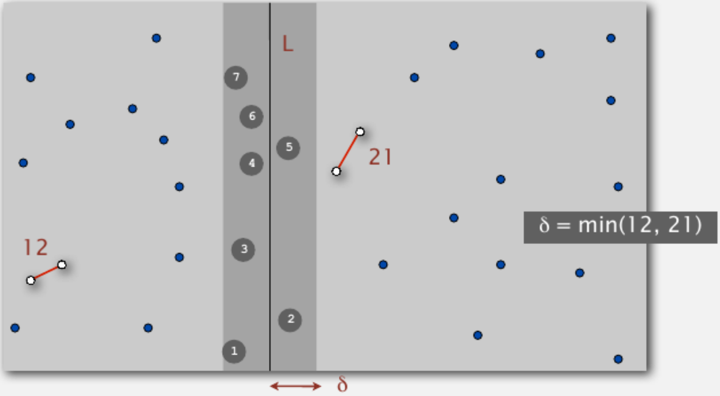
- Observation: only need to consider points within δ of line L.



Divide-and-conquer Algorithm

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.

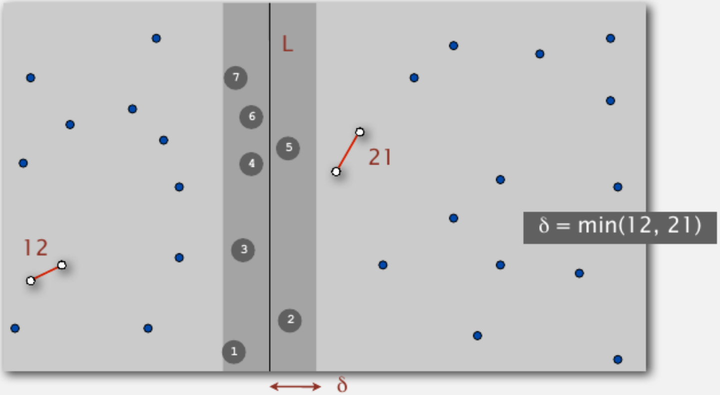


Divide-and-conquer Algorithm

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?



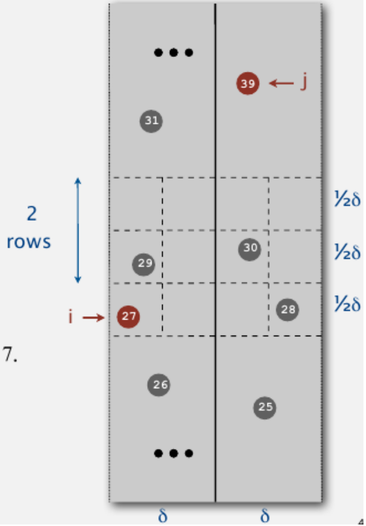
Divide-and-conquer Algorithm

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
   are on one side and half on the other side.                O(N log N)

   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

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.

Next time

Combinatorics/Probability theory…