CSE 5311 Notes 13: Computational Geometry

(Last updated 4/17/17 4:39 PM)

SMALLEST ENCLOSING DISK

See section 4.7 of de Berg (`http://dx.doi.org.ezproxy.uta.edu/10.1007/978-3-540-77974-2`)

Algorithm MINIDISC(P)
Input. A set P of n points in the plane.
Output. The smallest enclosing disc for P.

```
1. Compute a random permutation $p_1, \ldots, p_n$ of $P$.
2. Let $D_2$ be the smallest enclosing disc for $\{p_1, p_2\}$.
3. for $i \leftarrow 3$ to $n$
4.     do if $p_i \in D_{i-1}$
5.         then $D_i \leftarrow D_{i-1}$
6.         else $D_i \leftarrow$ MINIDISCWITHPOINT($\{p_1, \ldots, p_{i-1}\}, p_i$)
7. return $D_n$
```

MINIDISCWITHPOINT(P, q)
Input. A set P of n points in the plane, and a point q such that there exists an enclosing disc for P with q on its boundary.
Output. The smallest enclosing disc for P with q on its boundary.

```
1. Let $D_1$ be the smallest enclosing disc with $q$ and $p_1$ on its boundary.
2. for $j \leftarrow 2$ to $n$
3.     do if $p_j \in D_{j-1}$
4.         then $D_j \leftarrow D_{j-1}$
5.         else $D_j \leftarrow$ MINIDISCWITH2POINTS($\{p_1, \ldots, p_{j-1}\}, p_j, q$)
6. return $D_n$
```

MINIDISCWITH2POINTS(P, q_1, q_2)
Input. A set P of n points in the plane, and two points q_1 and q_2 such that there exists an enclosing disc for P with q_1 and q_2 on its boundary.
Output. The smallest enclosing disc for P with q_1 and q_2 on its boundary.

```
1. Let $D_0$ be the smallest enclosing disc with $q_1$ and $q_2$ on its boundary.
2. for $k \leftarrow 1$ to $n$
3.     do if $p_k \in D_{k-1}$
4.         then $D_k \leftarrow D_{k-1}$
5.         else $D_k \leftarrow$ the disc with $q_1$, $q_2$, and $p_k$ on its boundary
6. return $D_n$
```

Fundamental Predicates

See `http://www.cs.cmu.edu/~quake/robust.html` or O'Rourke's book for more details.

Twice the (signed) area of a triangle $A(T)$ is given by:

$$2A(T) = \begin{vmatrix} x_a & y_a & 1 \\ x_b & y_b & 1 \\ x_c & y_c & 1 \end{vmatrix} = \begin{vmatrix} x_b - x_a & y_b - y_a \\ x_c - x_a & y_c - y_a \end{vmatrix} = (x_b - x_a)(y_c - y_a) - (x_c - x_a)(y_b - y_a)$$

If positive, then points a, b, and c make a left turn (counter-clockwise).

If negative, then points a, b, and c make a right turn (clockwise).

If zero, then points a, b, and c are collinear.

Relationship of a point a to counter-clockwise circle of points b, c, and d.

$$\begin{vmatrix} x_a & y_a & x_a^2 + y_a^2 & 1 \\ x_b & y_b & x_b^2 + y_b^2 & 1 \\ x_c & y_c & x_c^2 + y_c^2 & 1 \\ x_d & y_d & x_d^2 + y_d^2 & 1 \end{vmatrix} \quad \begin{array}{r l} \text{Zero:} & \text{on circle} \\ \text{Positive:} & \text{outside} \\ \text{Negative:} & \text{inside} \end{array}$$

If the vertices $v_i = (x_i, y_i)$ of a polygon are labeled counter-clockwise, the area is:

$$\frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - y_i x_{i+1}) = \frac{1}{2}\sum_{i=0}^{n-1}(x_i + x_{i+1})(y_{i+1} - y_i)$$

Proximity

Closest points in 1-d space (`http://ranger.uta.edu/~weems/NOTES5311/1dclosest.c`)

1. Find median of point set. (Notes 6)
2. Recursively determine closest pair on both left side and right side.
3. Check whether rightmost point on left side and leftmost on right side are a closer pair than 2.

Worst-case: $\Theta(n \log n)$

Closest points in 2-d space (`http://ranger.uta.edu/~weems/NOTES5311/2dclosest.c`)

Brute-force: $\Theta\left(n^2\right)$

Divide-and-conquer:

1. Draw vertical line to divide into equal-size subsets.

2. Recursively find closest pair for left and right sides. Let δ be the smaller of the two distances.

3. Find closest pair among points within δ of the dividing line (the *seam*).

Since the point set is not random, details must assure that $\Theta\left(n^2\right)$ behavior is avoided.

Base Case: If $n \leq 3$ (or some other constant), use brute-force.

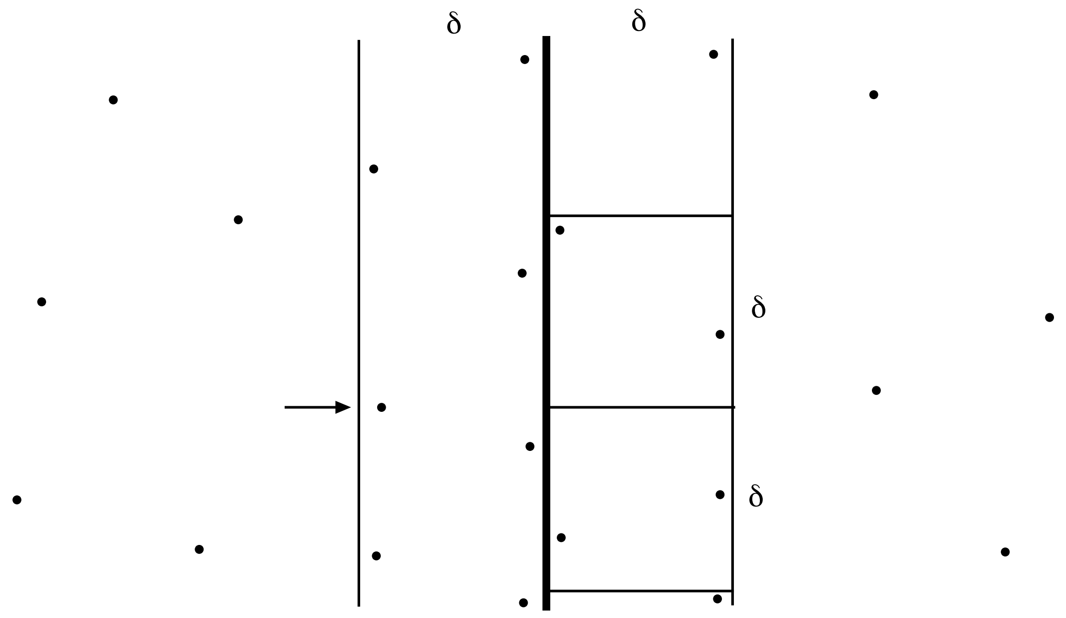
To support the "divides" and the seam processing, the set of points is preprocessed:

1. Create array with points sorted by x-coordinate.

2. Create second array with points sorted by y-coordinate. Also include cross-references to x-ordered array.

When a "divide" by a vertical line is needed, the first array is trivial to split and the second array is split by using the cross-references.

The y-ordered array facilitates finding the closest pair across the seam.



For a given left-side seam point, the distances to at most six right-side seam points are needed.

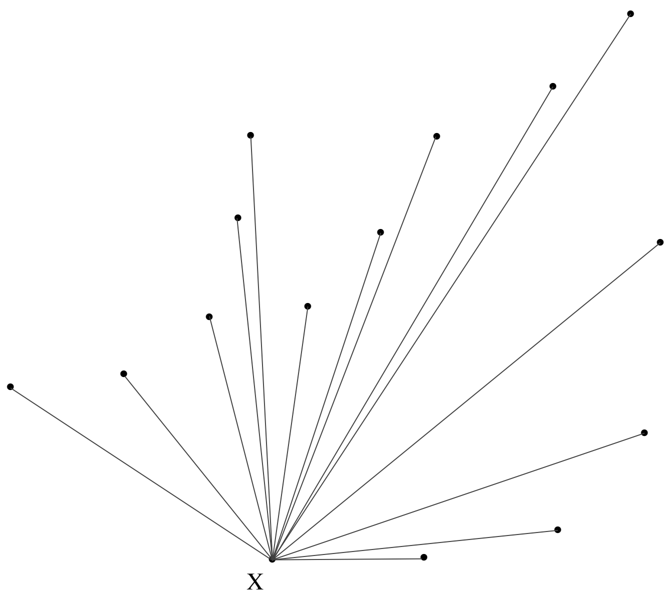
$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) = \Theta(n \log n)$$

CONVEX HULLS

Determine smallest convex polygon that includes all points in a 2-d set.

Graham scan - Based on *angular sweep* w.r.t. the (leftmost) bottom point X and maintaining stack with convex hull.

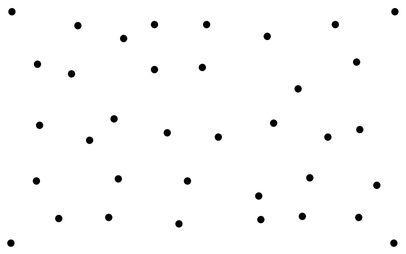
```
1. Find X.

2. Sort by angle w.r.t. X.  Comparisons by testing "turns" and breaking collinear cases by
   taking farthest point first.  (No arctangents needed.)

3. Push X and first two sweep points.

4. for each point P in sorted order

       while next-to-top-of-stack, top-of-stack, and P do not make a left turn

           Discard top-of-stack (it's not in convex hull)

       Push P
```



Jarvis march (rubberbanding or gift-wrapping)

Runs in $\Theta(nh)$ time where h is the number of hull points

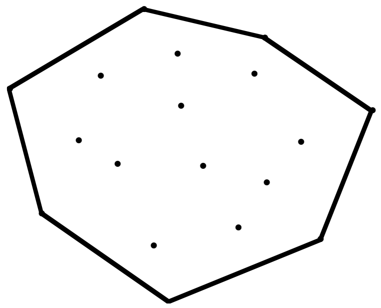
Good for cases like:



Same initial point X as Graham scan.

Also need (leftmost) top point Y.

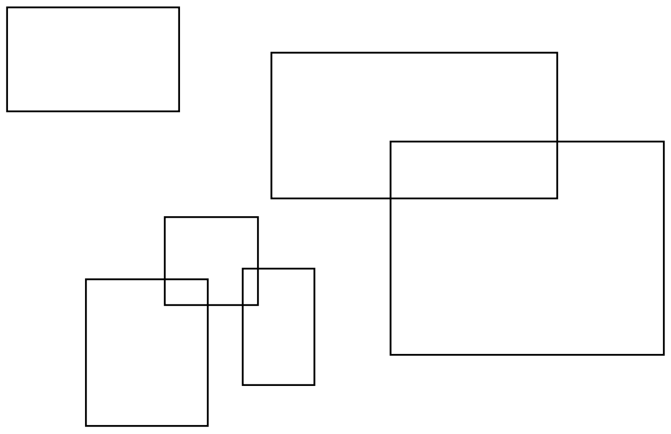
Each successive hull point is found by finding minimum angle WRT the last point and the horizon.



Convex hull may be used to find the diameter of a point set using $\Theta(h)$ additional time.
https://en.wikipedia.org/wiki/Rotating_calipers

SWEEP-LINE ALGORITHMS

Simple example: Intersection of rectilinear rectangles



Idea: Sweep a vertical line from left-to-right and store vertical cross-section (sweep-line status).

Preprocessing: Sort left and right edges by x-coordinate (event-point schedule).

Algorithm: Sweep across x dimension

Left edge: Check for intersection. Insert in interval tree (CLRS, 14.3, p. 350)

Right edge: Delete from interval tree

Runs in $\Theta(n \log n + n \log m) = \Theta(n \log n)$ (m is max rects in tree)

Difficulty: What if two rectangles "touch"? Treat as intersecting or not by how ties are handled.

More significant example: 2-d closest pairs

Idea: Incrementally determine δ for the leftmost k points. Maintain y-ordered BST of points whose x-distance from point $k + 1$ is $< \delta$.

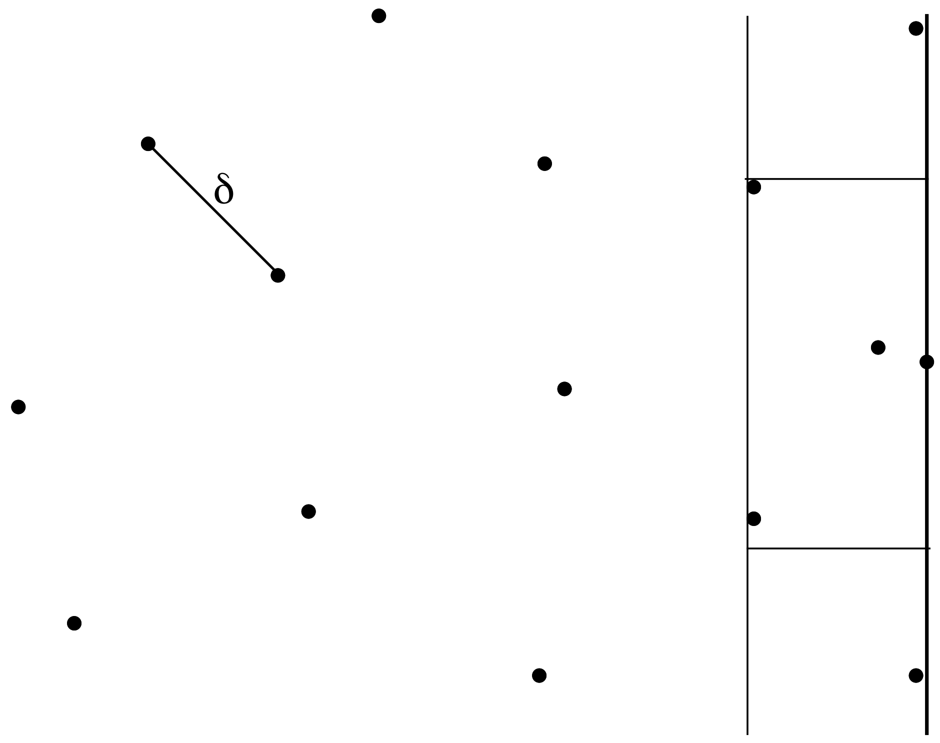
Preprocessing: Sort points by x-coordinate.

Processing point $k + 1$:

1. Delete BST points that are at least δ to the left of point $k + 1$.

2. Examine BST points that are no more than δ below or above point $k + 1$ and check for improving δ.

3. Insert point $k + 1$ into BST. (If BST already has a point with same y-coordinate, replace it).



Time: $\Theta(n \log n)$

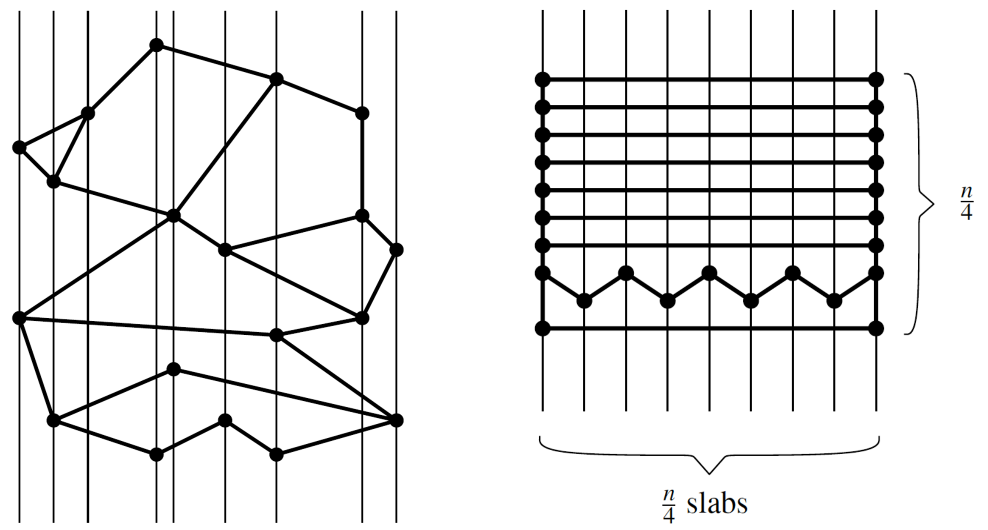
Reference: K. Hinrichs, J. Nievergelt, and P. Schorn, "Plane-Sweep Solves the Closest Pair Problem Elegantly", *Information Processing Letters* 26 (1988), 255-261.

Plane Partitions and Point Location - (aside)

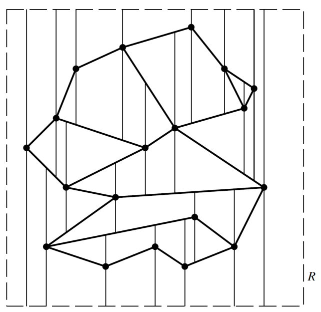
`https://en.wikipedia.org/wiki/Point_location`

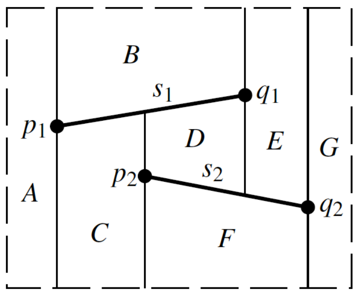
Chapter 6 of M. de Berg et.al., *Computational Geometry*, 3rd ed.

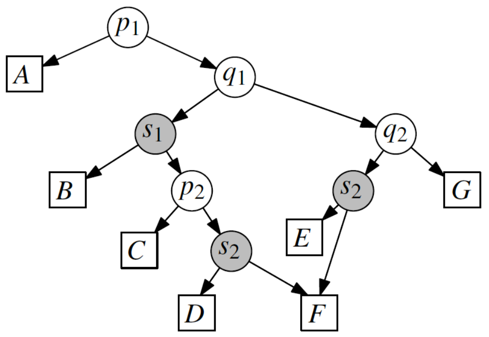
Polygonal partitioning by slabs ($\Theta(n^2)$ space, $\Theta(\log n)$ time)



Trapezoidal maps ($\Theta(n)$ space, $\Theta(\log n)$ expected time)







Triangular decomposition (used in incremental flip method for Delaunay Triangulation, de Berg Chapter 9, p. 203)

EUCLIDEAN MINIMUM SPANNING TREES

Voronoi Diagram - post office problem. Divides plane into convex regions, each containing points closest to some given point (blue lines). (Chapter 7 of M. de Berg et.al.)

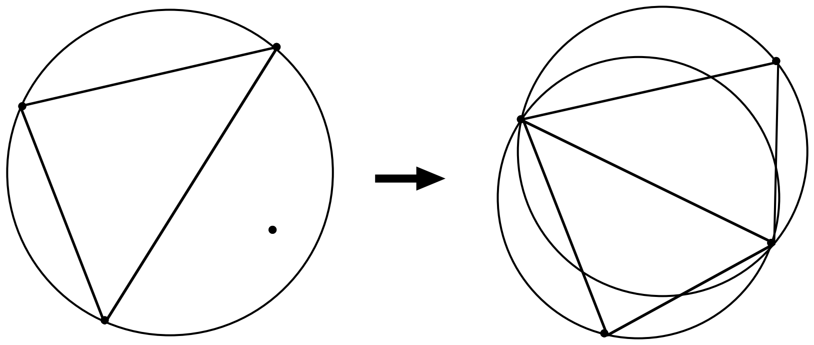
Fortune's Sweep-Line ("beachline") achieves $\Theta(n \log n)$ time

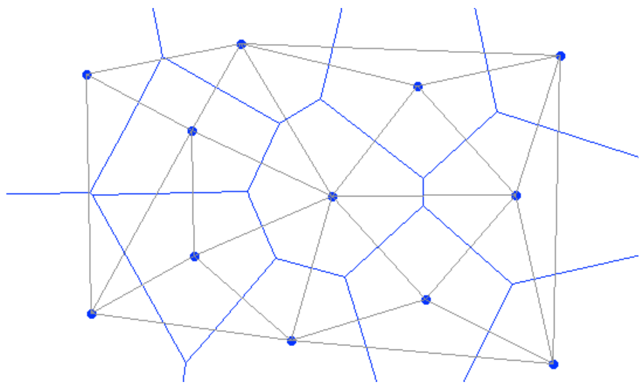
Delaunay Triangulation (has $\Theta(n)$ size, must include EMST edges, Chapter 9 of M. de Berg et.al.)

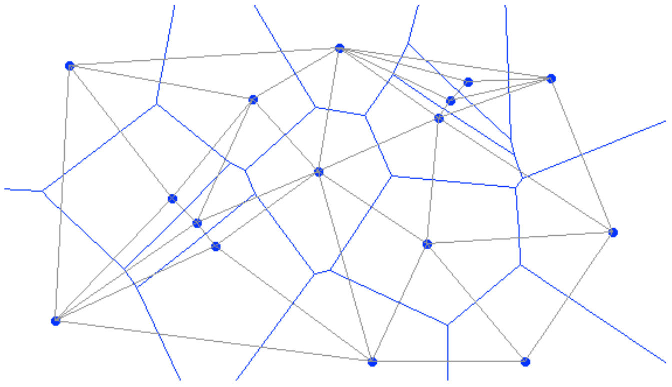
Connects vertices for adjacent Voronoi regions (black lines between input points).

May transform an arbitrary triangulation to a DT in $\Theta(n^2)$ time using flips based on incircle test and the following property:

Three points are vertices of a Delaunay triangle iff the circle that passes through the three points contains no other input point.





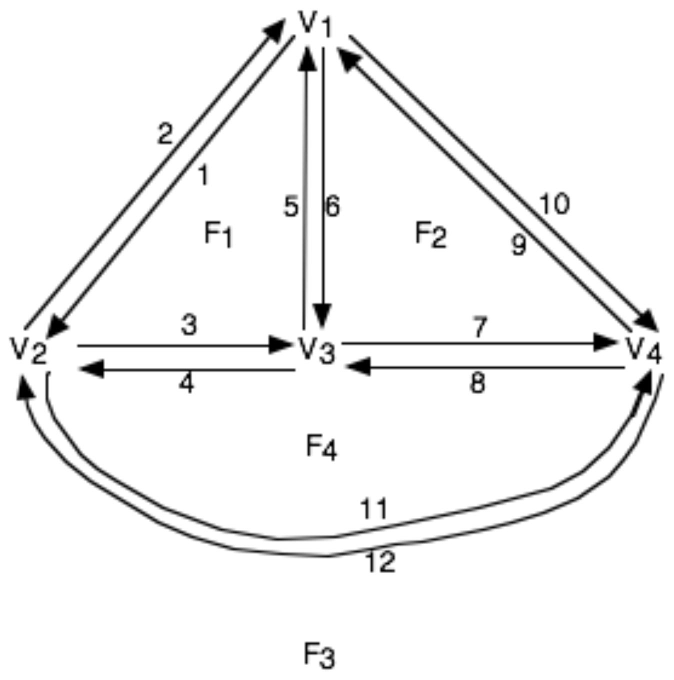


Doubly-Connected Edge List - DCEL (AKA Twin/Half-Edge Data Structure, $V - E + F = 2$).
de Berg, section 2.2

Vertex #	Cooordinates	Incident Edge#
1	0 0 0	1
2	1 0 0	2
3	0 1 0	4
4	0 0 1	8

Face #	Edge
1	1
2	6
3	10
4	11

Edge #	Origin (Tail)	Twin	Incident Face	Next	Prev
1	1	2	1	3	5
2	2	1	3	10	12
3	2	4	1	5	1
4	3	3	4	11	8
5	3	6	1	1	3
6	1	5	2	7	9
7	3	8	2	9	6
8	4	7	4	4	11
9	4	10	2	6	7
10	1	9	3	12	2
11	2	12	4	8	4
12	4	11	3	2	10



Aside: What do -

Determining whether three values in a set of integers add to 0
Determining whether there are three values from different sets of integers that add to 0
Determining whether no three points in a set are collinear
Determining whether no three lines in a set intersect
Determining the area of overlapping triangles in the plane

have in common? They are 3SUM-hard . . .

A. Gajentaan and M.H. Overmars, "On a Class of $O\left(n^2\right)$ Problems in Computational Geometry", *Computational Geometry: Theory and Applications* 5 (1994), 165-185.
`http://www.sciencedirect.com.ezproxy.uta.edu/science/article/pii/0925772195000222`