# CS 164 & CS 266: Computational Geometry

## Lecture 3

## Arrangements of lines

**David Eppstein**
University of California, Irvine

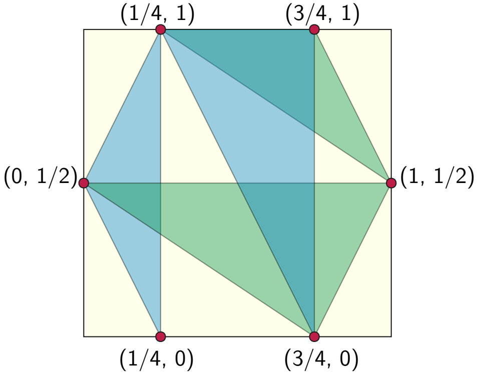
Fall Quarter, 2025

# Minimum-area triangles

# The Heilbronn triangle problem

Find $n$ points in unit square maximizing the minimum area triangle

3 in line $\Rightarrow$ area $= 0$, so we want as far out of line as possible




Example: $n = 6$. For these points, triangle areas are $1/8$, $1/4$, and $3/8$,

so its minimum triangle area is $1/8$.

This point set is optimum: no other 6-point set has minimum $> 1/8$.

# Known bounds on the Heilbronn problem

Sort points by $x$-coordinates $\Rightarrow$ some three points lie in a thin vertical strip $\Rightarrow$ their triangle has area $O(1/n)$

Every sufficiently large point set has a triangle with area

$$\Delta(n) \leq \left(\frac{1}{n}\right)^{\frac{8}{7}+\frac{1}{2000}}$$

[Cohen et al. 2023] slightly improving [Komlós et al. 1981]

Some $n$-point sets have smallest triangle area

$$\Omega\left(\frac{\log n}{n^2}\right)$$

[Komlós et al. 1982]

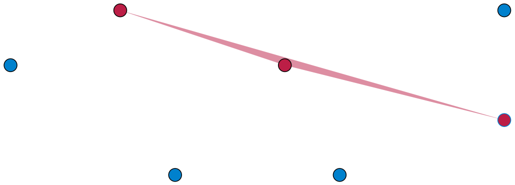
Unsolved problem: close this big gap!

# How to test quality of Heilbronn triangle solution?

Algorithmic problem:

Given $n$ points,
find their minimum-area triangle

Special case: detect an area-zero triangle – is this input in general position?



Naïve algorithm:

Use the triangle-area primitive on all triples

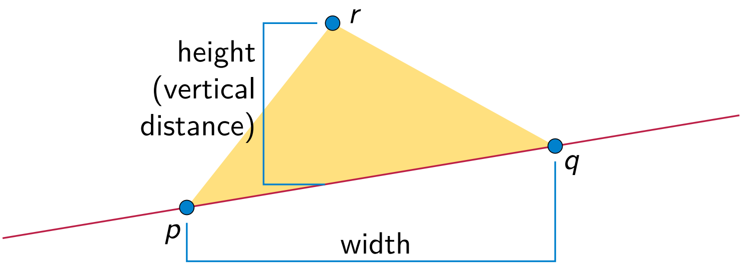
Time $= O(n^3)$

We can do much better!

# From triangle area to vertical distance

For each two (non-vertically-aligned) points $p$, and $q$:

- We can find the line $\ell : y = mx + b$ through $p$ and $q$
- Area of $pqr$ is $\frac{1}{2}$width$\cdot$height where width is $|x_p - x_q|$ and height (vertical distance of $r$ from $\ell$) is $|y_r - (mx_r + b)|$, the difference between the actual $y$-coordinate of $r$ and the $y$-coordinate predicted by the line given $x_r$
- Do this for all $r \Rightarrow$ find the smallest triangle using $p$ and $q$
- But this is still $O(n^3)$ because there are $O(n^2)$ lines to try



# Duality of vertical distance

Given points with coordinates $(x, y)$
and non-vertical lines with coordinates $(m, b)$
$\Rightarrow$ (signed) vertical distance is $y - mx - b$

Given points with coordinates $(m, -b)$
and non-vertical lines with coordinates $(x, -y)$
$\Rightarrow$ (signed) vertical distance is $-b - mx + y$

Using point coordinates for lines and vice versa
(and negating the second coordinate)
doesn't change our calculation!

# Original and dual problems

## Original problem:

Given $n$ points, no two $x$-coordinates equal

Find the non-vertical line through each pair

Find third given point with minimum vertical distance to it

(Min-area triangle will be one of the triples we find)

## Equivalent dual problem:

Given $n$ non-vertical lines, no two slopes equal

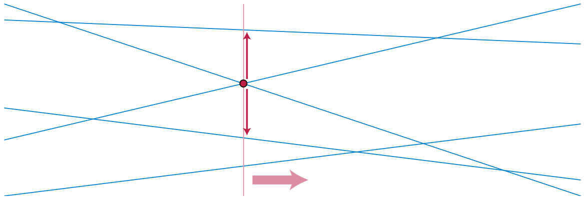
Find the crossing point of each two lines

Find third given line with minimum vertical distance to it

(Min-area triangle will still be one of the triples we find)

# Plane sweep

Use plane sweep for finding crossings of line segments, on the $n$ given lines



Each time it sweeps over a crossing, use binary search tree (of lines meeting vertical sweep line) to find the two lines directly above and directly below the crossing; one of the two will have minimum vertical distance

$O(n^2)$ crossings

$\Rightarrow$ $O(n^2)$ priority queue and binary search tree operations

$\Rightarrow$ total time $O(n^2 \log n)$

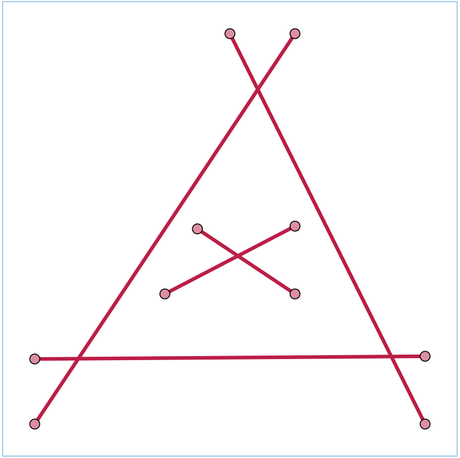
We can do better!

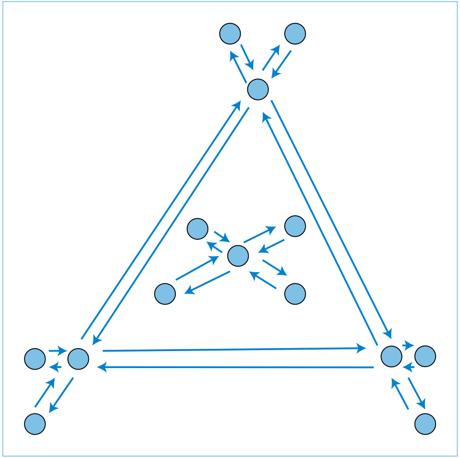
# Review from last time: DCEL

# Half-edges

Represent each edge of an arrangement by two directed edges ("half-edges"), like the two directions of a two-way road

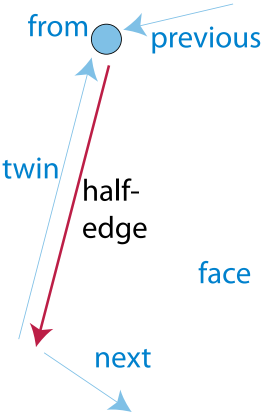
(But like in England or Japan where they drive on the left)





# Doubly-connected edge list

Object-oriented, with objects for vertices, half-edges, and faces




Each half-edge stores:

- Pointer to twin half-edge from same edge
- The vertex it comes from (Can find other vertex from twin)
- The face on its side of the edge
- The next and previous half-edges in the cycle around its face

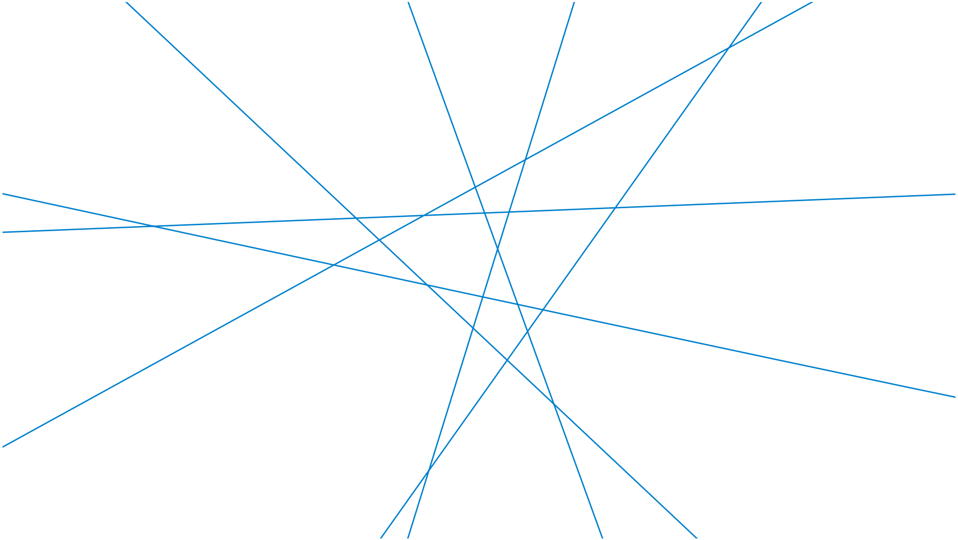
Each vertex stores:

- Its coordinates
- One of the half-edges it touches

Each face stores:

- A half-edge on its outer boundary
- A list of half-edges, one for each internal boundary

## An arrangement of lines



With no two parallel lines (general position), every two lines cross

Convenient to include one more line at infinity, proj. coords $(0, 0, 1)$

# How to handle unbounded edges and faces?

Use DCEL in the projective plane!

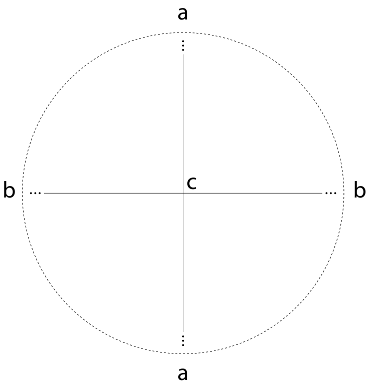
Include the line at infinity $(0, 0, 1)$ as part of the arrangement

(That way edges and faces don't "wrap around" to the other side of the Euclidean plane)

Ray $=$ line segment with one endpoint on the line at infinity

Unbounded cells have one edge on the line at infinity

# DCEL for x and y axes + line at infinity




Vertices a: (0,1,0), b: (1,0,0), c: (0,0,1)

Three faces: triangles in northwest, northeast, southwest, southeast

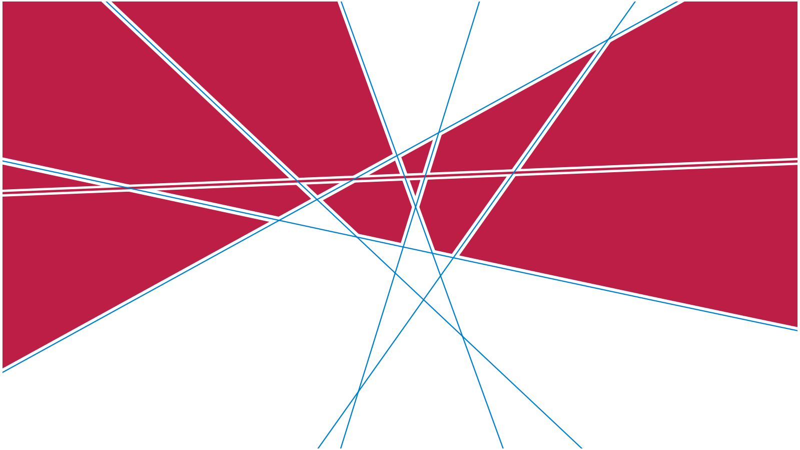
Six twin pairs of half-edges:

- Upper $y$-axis between northwest and northeast
- Lower $y$-axis between southwest and southeast
- Left $x$-axis between northwest and southwest
- Right $x$-axis between northeast and southeast
- Line @ infinity between northeast and southwest
- Line @ infinity between northwest and southeast

# Arrangement construction and complexity

# Incremental construction algorithm

- Start with DCEL describing two lines + the line at infinity
- For each line $\ell_i$ in arbitrary order:
  - Walk along line at infinity to find start point of new line
  - For each face crossed by the new line, walk around the face to find where it exits the face, and use that information to split the face into two smaller faces



# Combinatorial complexity of arrangements of lines

By understanding how many things are in the arrangement
we can analyze algorithms that do something for each thing

$$\#\text{ vertices (crossings)} \leq \binom{n}{2} = O(n^2)$$

Each line is divided into 2 rays and $\leq n-2$ segments $\Rightarrow$

$$\#\text{ rays} = 2n$$

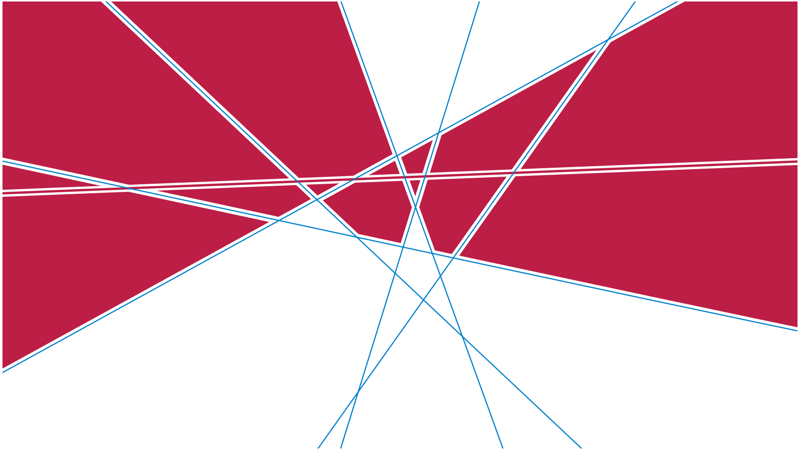
$$\#\text{ segments} \leq n(n-2) = O(n^2)$$

Each face has a bottom vertex or extends infinitely far down
Infinite-downward faces are separated by $n$ lines $\Rightarrow$

$$\#\text{ faces} \leq \binom{n}{2} + n + 1 = O(n^2)$$

# Zones

Zone = faces touched by a line = new faces after inserting the line



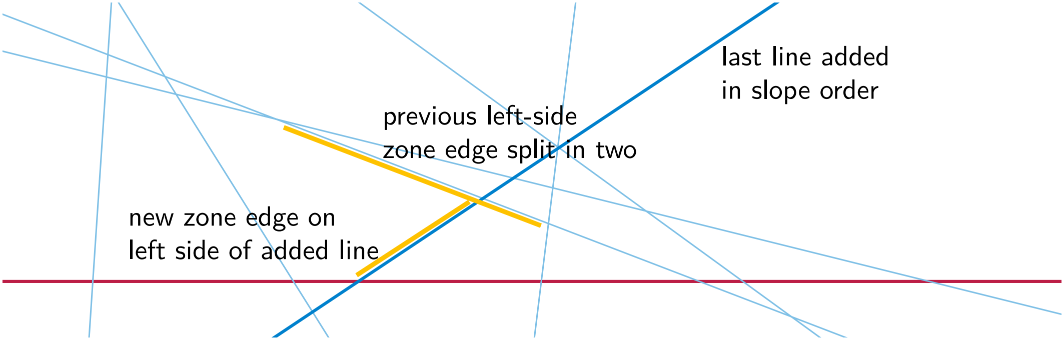
Its complexity (sum of # edges in each face) controls the time taken for walks across faces when adding the line to the arrangement

Zone theorem: For every zone, the complexity is $O(n)$

# Ideas for proof of zone theorem

Rotate so the given line is horizontal
Add other lines to zone, sorted by slope (negative to positive)

Each line creates $\leq 2$ edges on left sides of lines, above given line:
One on new line, one formed by splitting an existing edge



# Summing up zones

Zone has:

- $\leq 2n$ half-edges above the horizontal line, on the left side of each line
- $\leq 2n$ half-edges above the horizontal line, on the right side of each line
- $\leq 2n$ half-edges below the horizontal line, on the left side of each line
- $\leq 2n$ half-edges right the horizontal line, on the right side of each line
- $\leq 2n$ half-edges on the horizontal line itself

Total: $\leq 10n$ by this analysis

Actual max complexity is $\lfloor 9.5n \rfloor - 3$ [Bern et al. 1991; Pinchasi 2011]

# Analysis of incremental arrangement construction

Each zone has complexity $O(n)$

Adding each line to the arrangement takes time proportional to its zone (in the arrangement so far), $O(n)$

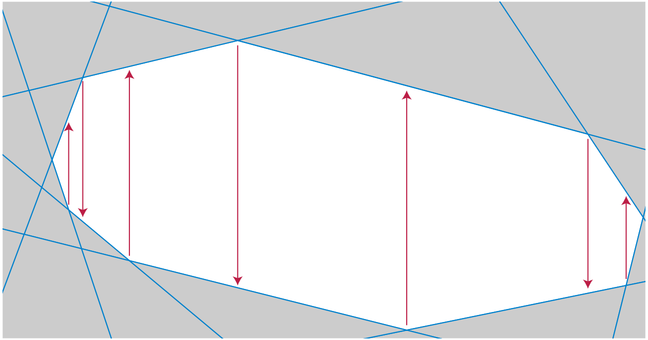
Constructing the whole arrangement takes time $O(n^2)$

# To find the minimum-area triangle

For each face of arrangement:

- Merge top and bottom left-to-right sequences of vertices to find top edge above each bottom vertex, and bottom edge below each top vertex
- Each edge-vertex pair gives you a candidate triple of input lines

Return the smallest triangle among the candidate triples



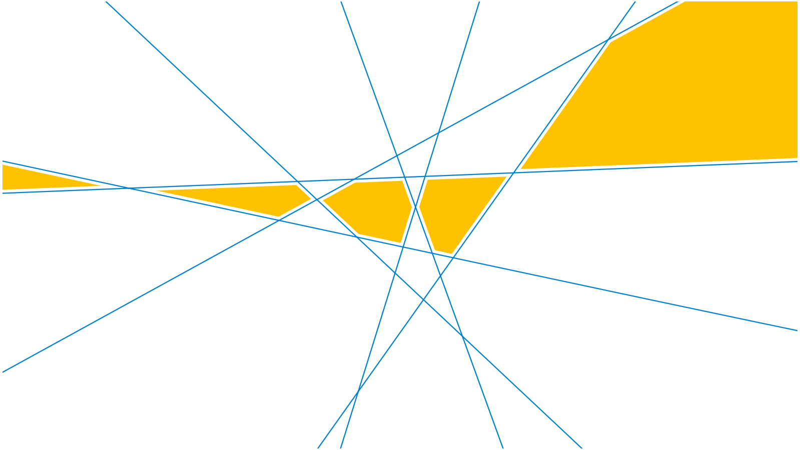
Time = linear in DCEL = $O(n^2)$

# The middle level problem

Middle level = faces with $\lfloor n/2 \rfloor$ lines above and $\lceil n/2 \rceil$ lines below

Big unsolved problem: What is its complexity?

Dual: In how many ways can we bisect $n$ points by a line?



$O(n^{4/3})$ [Dey 1998] $\Omega\left(ne^{c(\log n)^{1/2}}\right)$ [Tóth 2001]

# References


Marshall W. Bern, D. Eppstein, Paul E. Plassman, and Frances F. Yao. Horizon theorems for lines and polygons. In J. E. Goodman, R. Pollack, and W. Steiger, editors, *Discrete and Computational Geometry: Papers from the DIMACS Special Year*, volume 6 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, pages 45–66. American Mathematical Society, 1991. URL `https://ics.uci.edu/~eppstein/pubs/BerEppPla-DCG-91.pdf`.

Alex Cohen, Cosmin Pohoata, and Dmitrii Zakharov. A new upper bound for the Heilbronn triangle problem. Electronic preprint arxiv:2305.18253, 2023.

Tamal K. Dey. Improved bounds for planar *k*-sets and related problems. *Discrete & Computational Geometry*, 19(3):373–382, 1998. doi: 10.1007/PL00009354.

János Komlós, János Pintz, and Endre Szemerédi. On Heilbronn's triangle problem. *Journal of the London Mathematical Society*, 24(3):385–396, 1981. doi: 10.1112/jlms/s2-24.3.385.

János Komlós, János Pintz, and Endre Szemerédi. A lower bound for Heilbronn's problem. *Journal of the London Mathematical Society*, 25(1):13–24, 1982. doi: 10.1112/jlms/s2-25.1.13.

Rom Pinchasi. The zone theorem revisited. Unpublished manuscript, 2011. URL `https://graphics.stanford.edu/courses/cs268-16-fall/Notes/zonespl.pdf`.

Géza Tóth. Point sets with many *k*-sets. *Discrete & Computational Geometry*, 26(2):187–194, 2001. doi: 10.1007/s004540010022.