CS 532: 3D Computer Vision
14th Set of Notes

Instructor: Philippos Mordohai
Webpage: www.cs.stevens.edu/~mordohai
E-mail: Philippos.Mordohai@stevens.edu
Office: Lieb 215

Lecture Outline

- Triangulating a Polygon (cont. from Week 11)

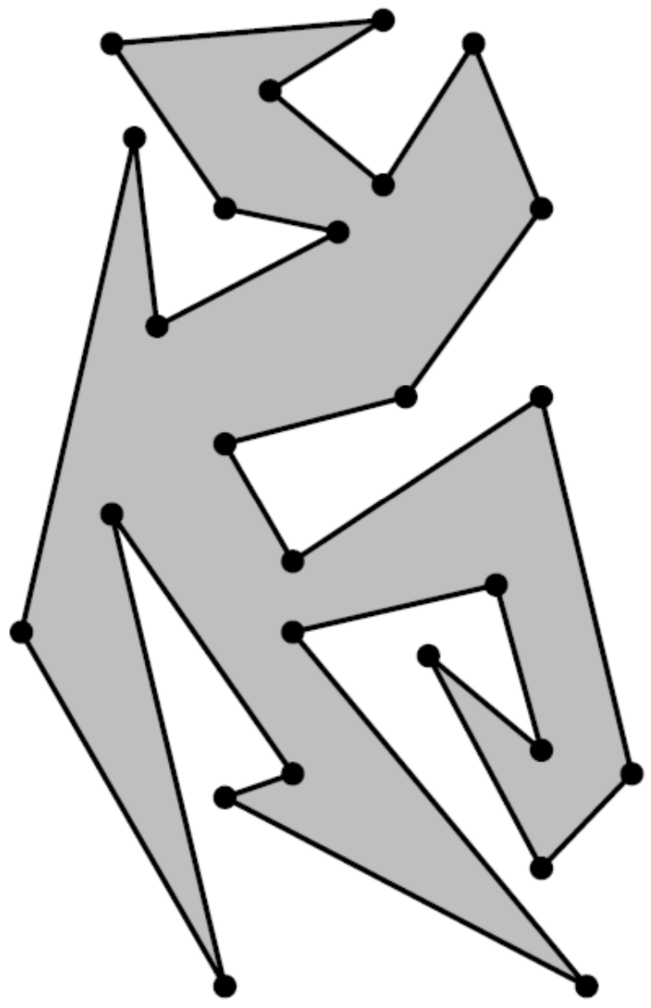
- Voronoi diagrams
- Delaunay triangulations
  - David M. Mount, CMSC 754: Computational Geometry lecture notes, Department of Computer Science, University of Maryland, Spring 2012
    - Lectures 11, 12 and 13
  - Slides by:
    - M. van Kreveld (Utrecht University)

Triangulating a Polygon (cont. from Week 11)

Slides by M. van Kreveld (Utrecht University)

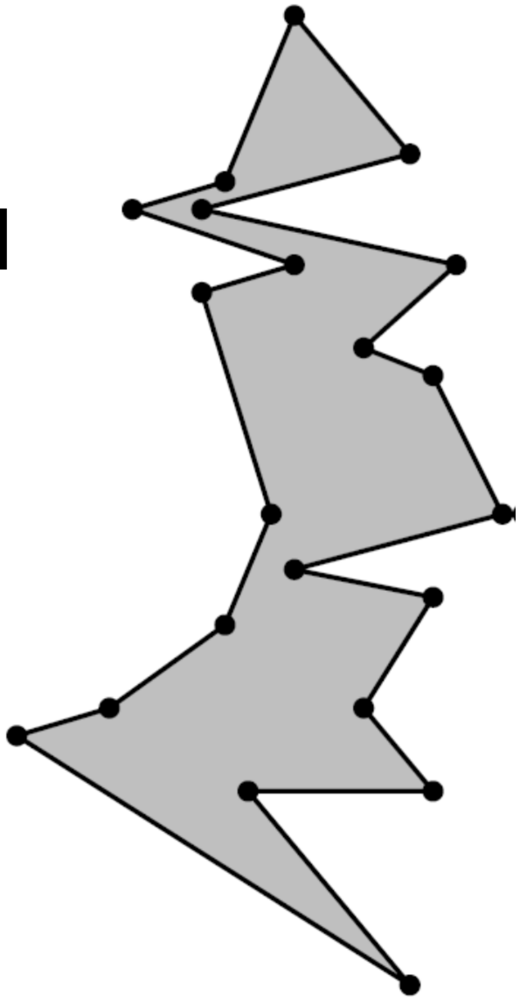
Two-Ears for Triangulation

- Using the two-ears theorem: (an ear consists of three consecutive vertices u, v, w where uw is a diagonal)
- Find an ear, cut it off with a diagonal, triangulate the rest iteratively



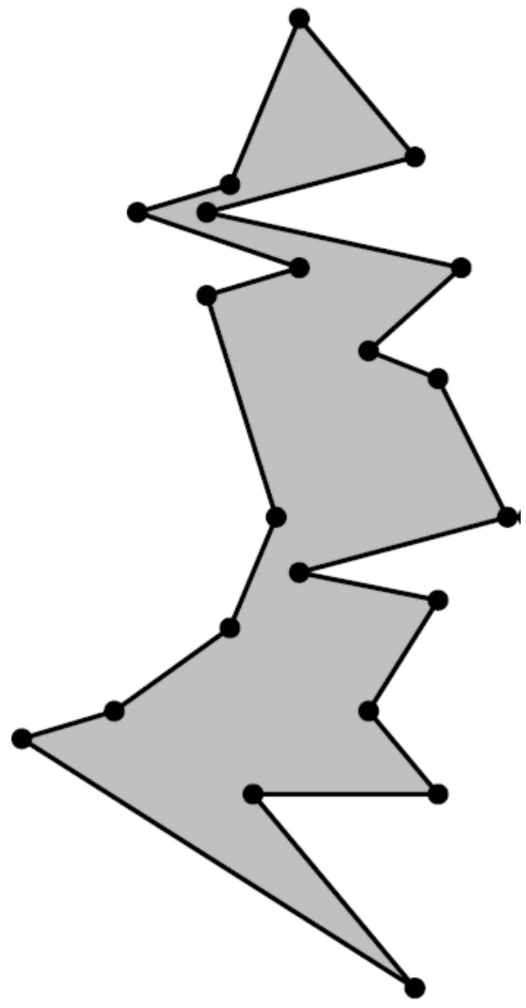
Overview

- A simple polygon is y-monotone iff any horizontal line intersects it in a connected set (or not at all)
- Use plane sweep to partition the polygon into y-monotone polygons
- Then triangulate each y-monotone polygon



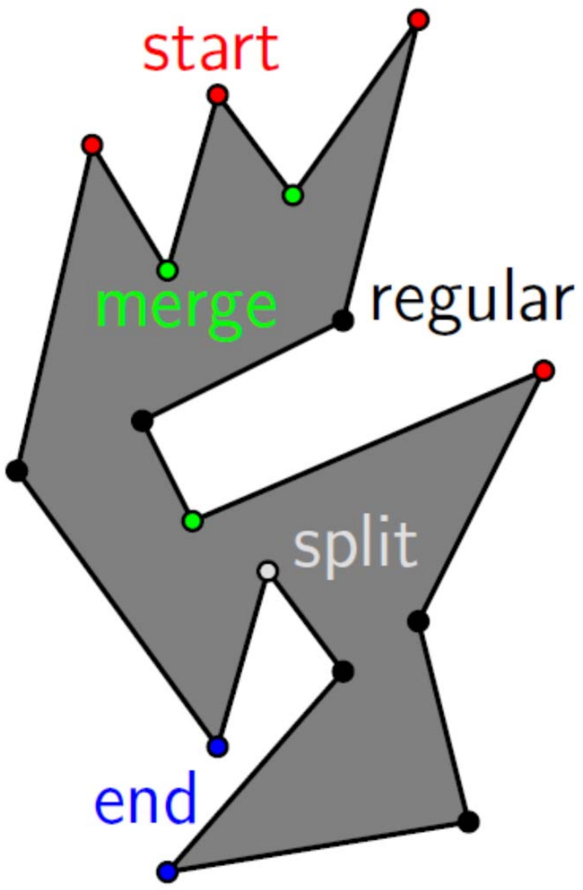
Monotone Polygons

- A y-monotone polygon has a top vertex, a bottom vertex, and two y-monotone chains between top and bottom as its boundary
- Any simple polygon with one top vertex and one bottom vertex is y-monotone



Vertex Types

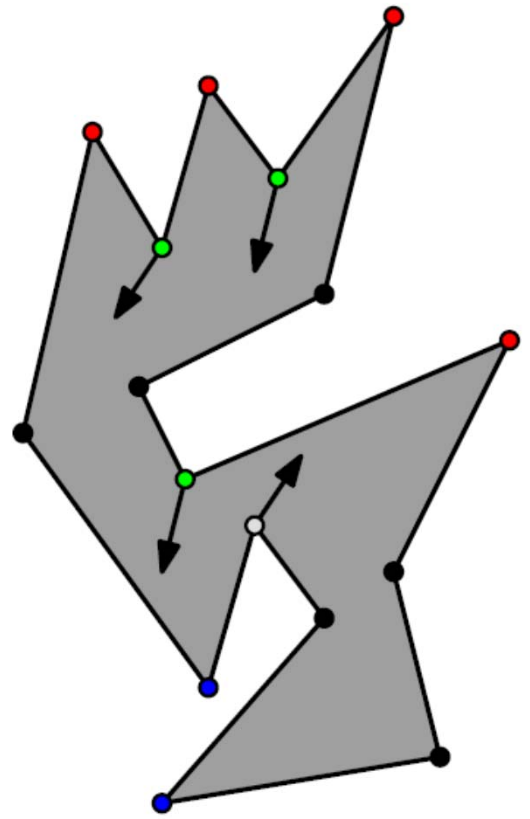
- What types of vertices does a simple polygon have?
  - start
  - stop
  - split
  - merge
  - regular
- … imagining a sweep line going top to bottom



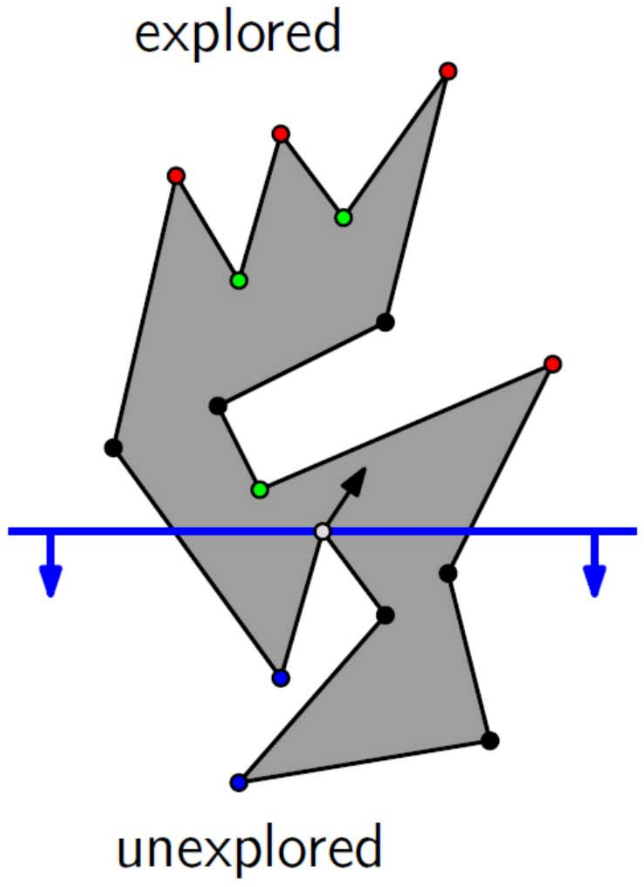
Sweep Ideas

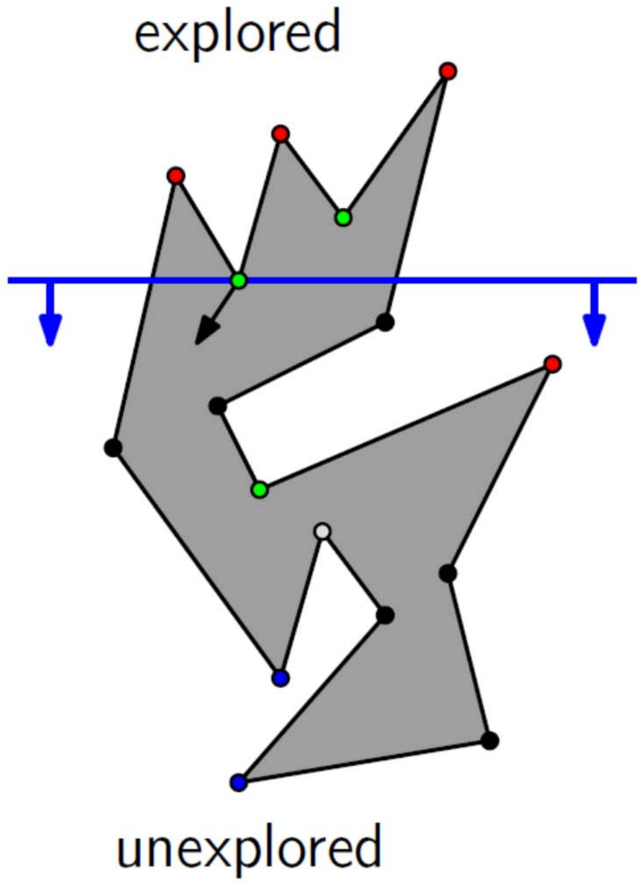
- Find diagonals from each merge vertex down, and from each split vertex up

- A simple polygon with no split or merge vertices can have at most one start and one end vertex, so it is y-monotone



Sweep Ideas

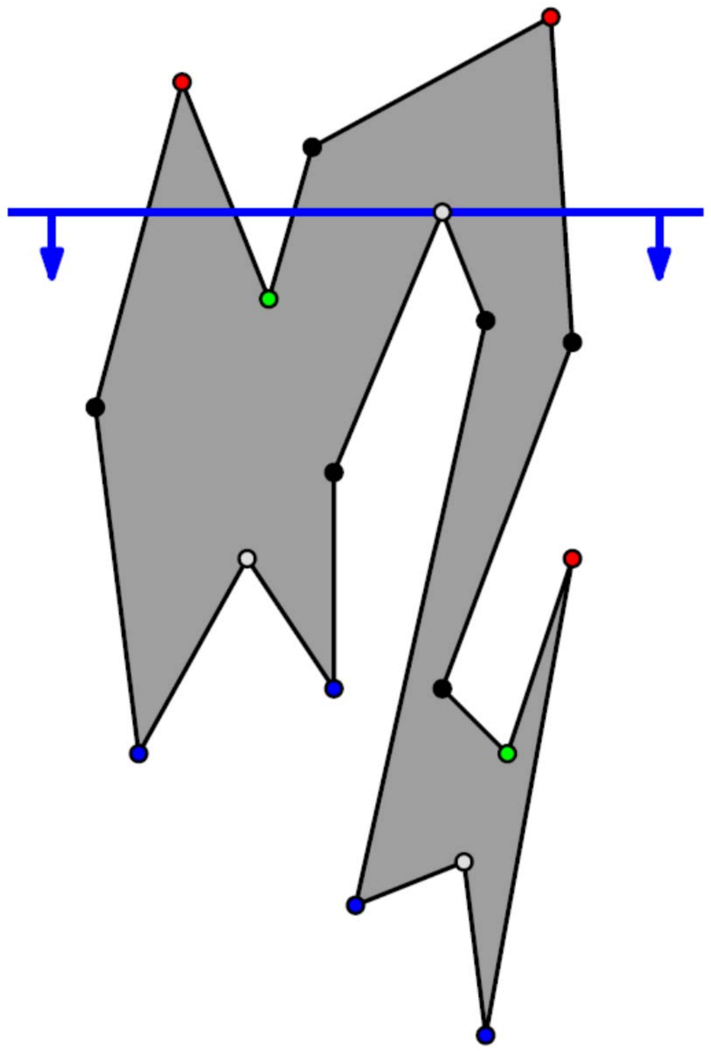




Sweep Ideas

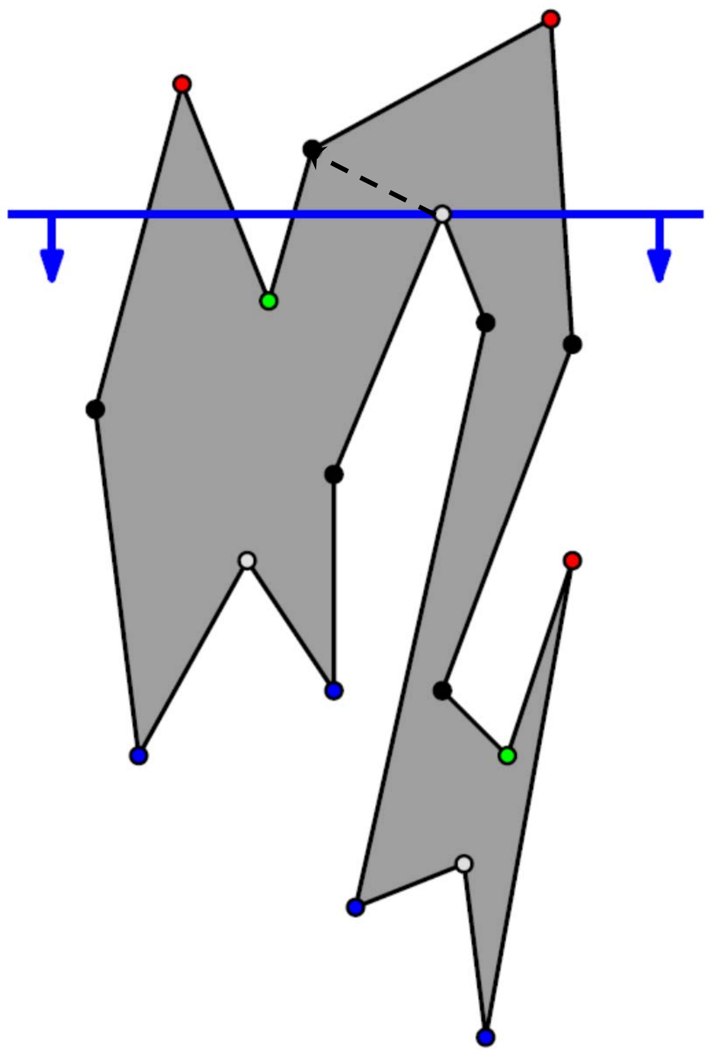
- Where can a diagonal from a split vertex go?

- Perhaps the upper endpoint of the edge immediately left of the merge vertex?



Sweep Ideas

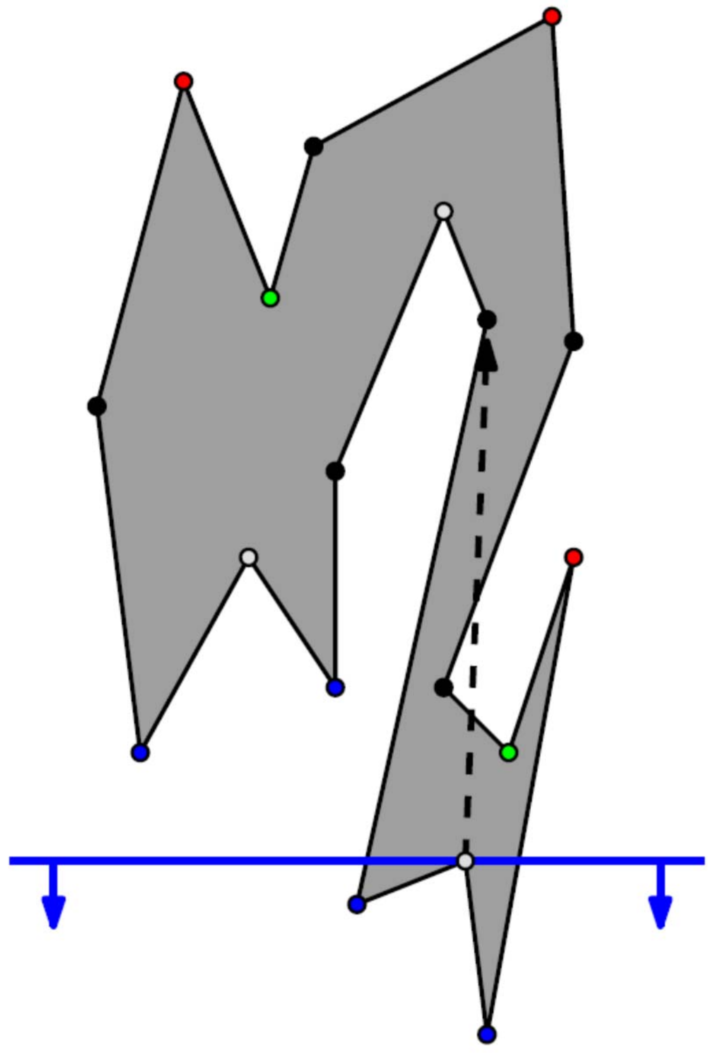
- Where can a diagonal from a split vertex go?
- Perhaps the upper endpoint of the edge immediately left of the merge vertex?



Sweep Ideas

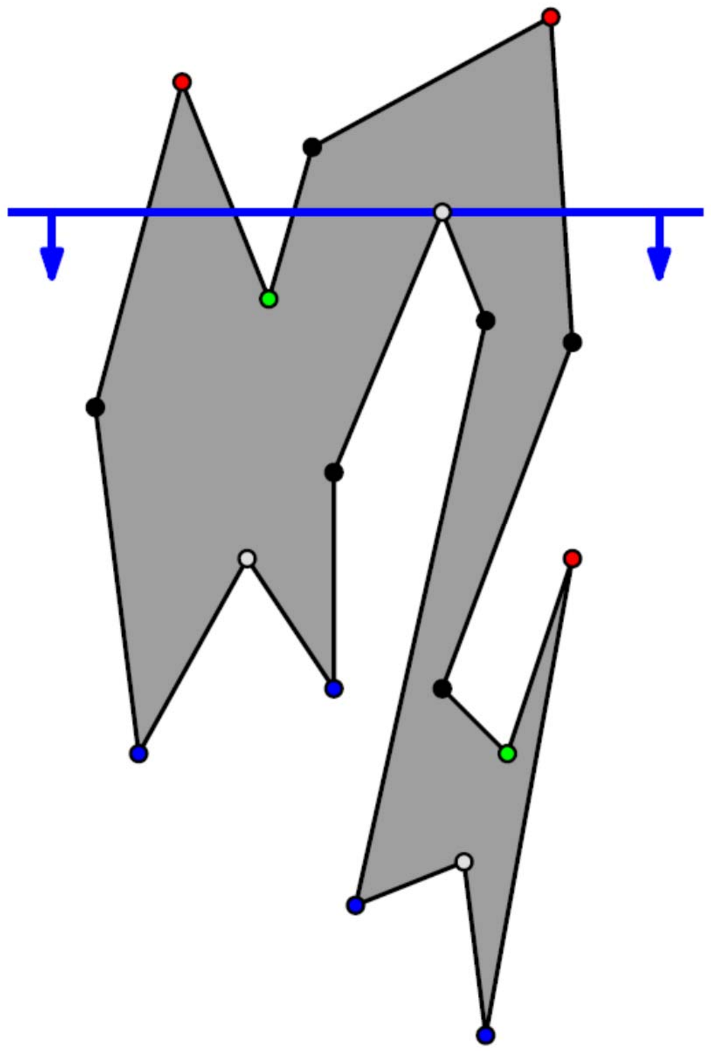
- Where can a diagonal from a split vertex go?

- Perhaps the upper endpoint of the edge immediately left of the merge vertex?



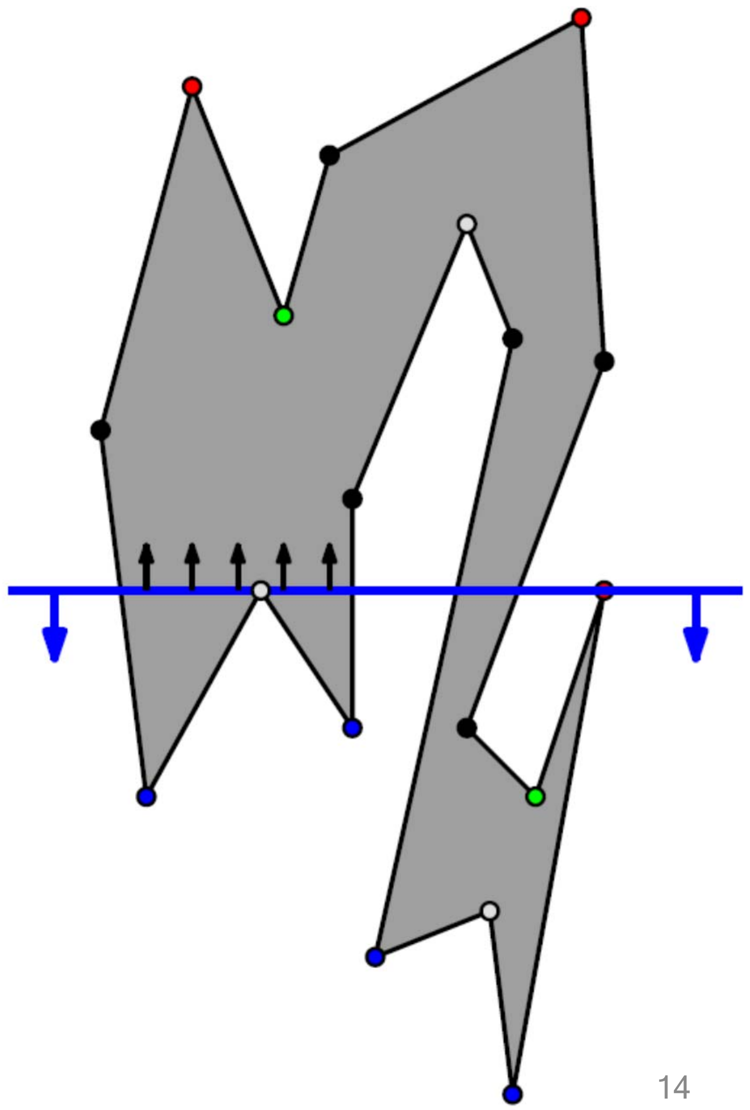
Sweep Ideas

- Where can a diagonal from a split vertex go?
- Perhaps the last vertex passed in the same component?



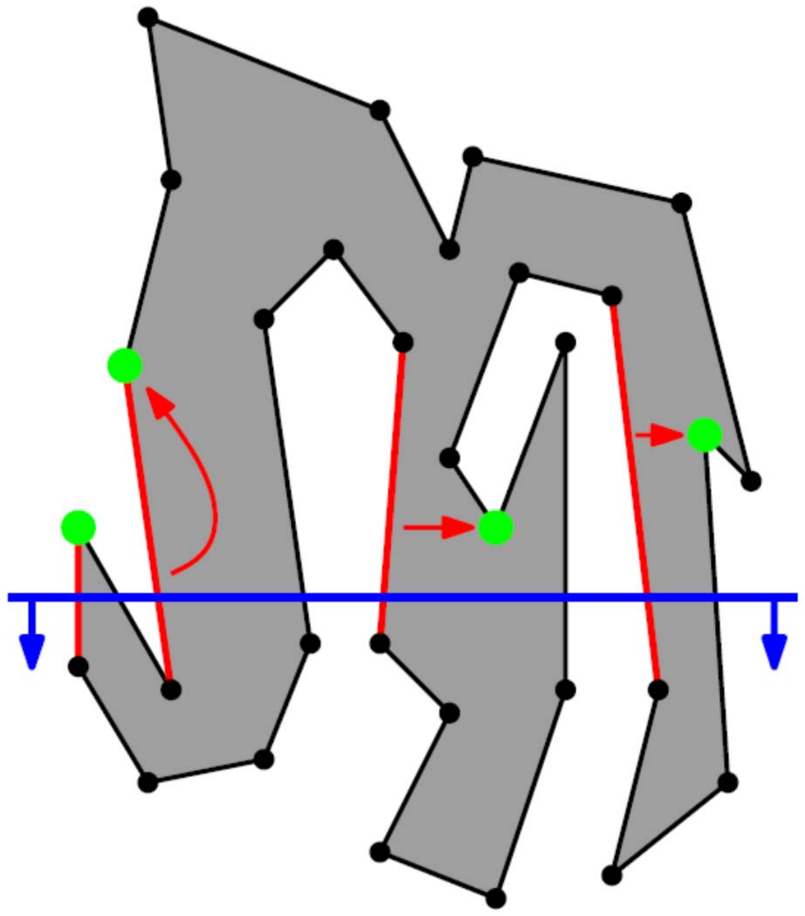
Sweep Ideas

- Where can a diagonal from a split vertex go?
- Perhaps the last vertex passed in the same component?



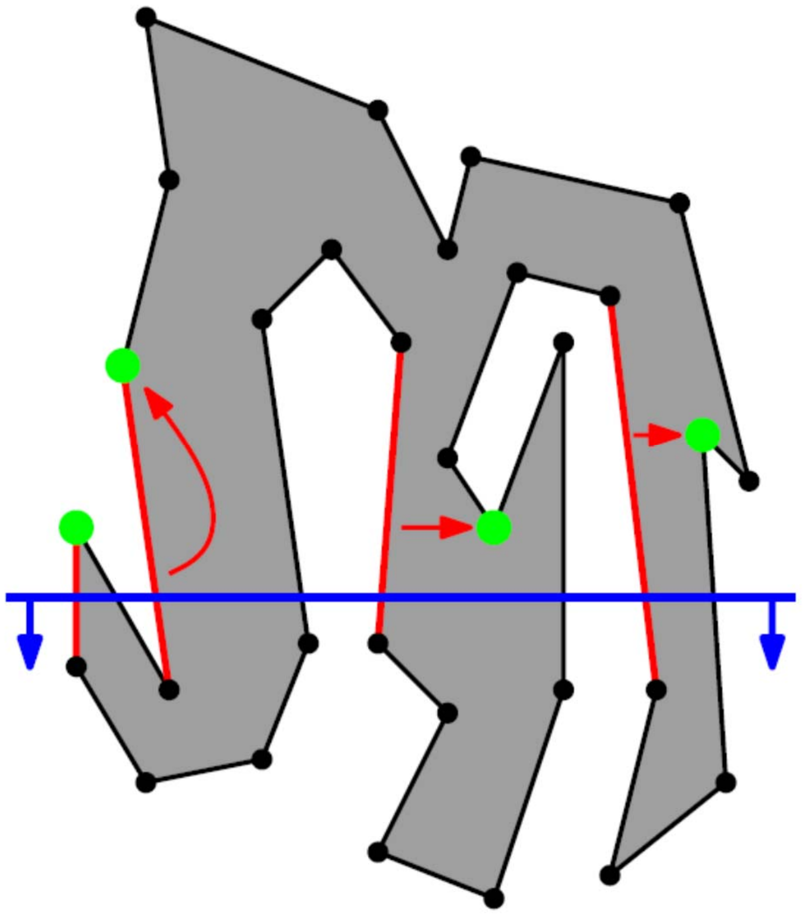
Helpers of Edges

- The helper of an edge e is the lowest vertex v above the sweep line such that the horizontal line segment connecting e and v is inside the polygon
- Note that helper(e_j) can be the upper endpoint of e_j itself



Status of Sweep

- The status is the set of edges intersecting the sweep line that have the polygon to their right, sorted from left to right
- Each edge has a helper: the last vertex passed in that component



Status Structure, Event List

- The status structure stores all edges that have the polygon to the right, with their helper, sorted from left to right in the leaves of a balanced binary search tree
- The events happen only at the vertices: sort them by y-coordinate and put them in a list (or array, or tree)

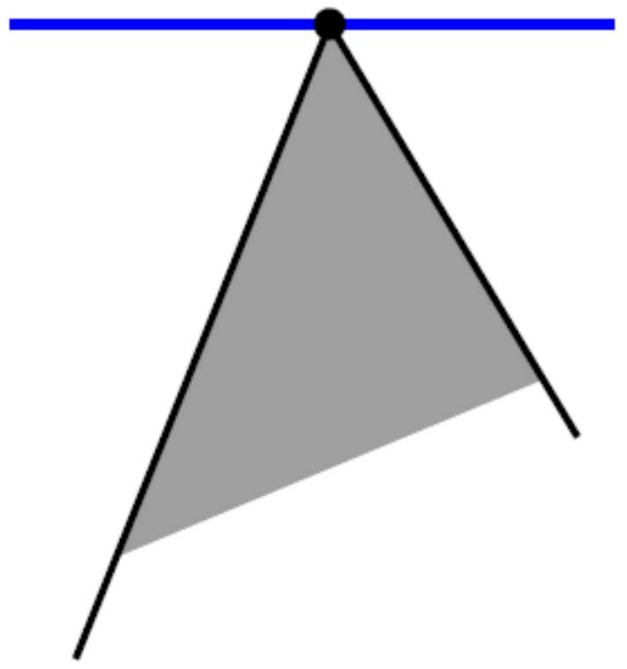
Main Algorithm

- Initialize the event list (all vertices sorted by decreasing y-coordinate) and the status structure (empty)

- While there are still events in the event list, remove the first (topmost) one and handle it

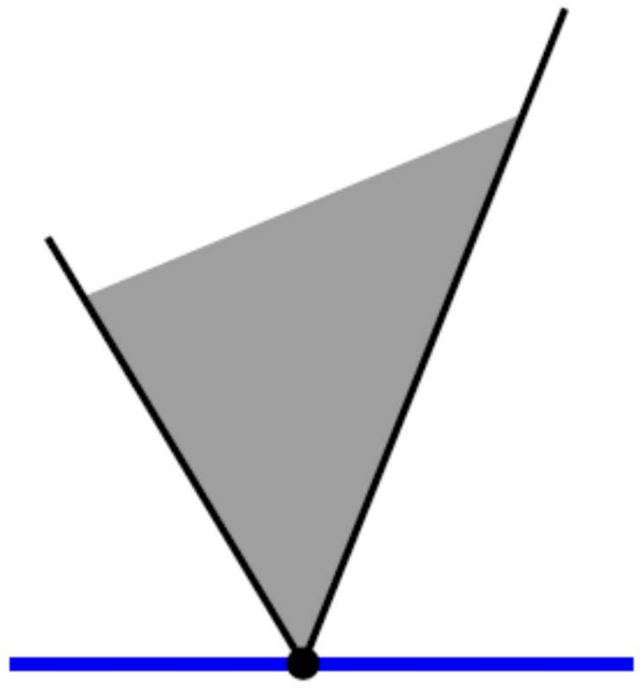
Event Handling

- Start vertex v:
  - Insert the counterclockwise incident edge in T with v as the helper



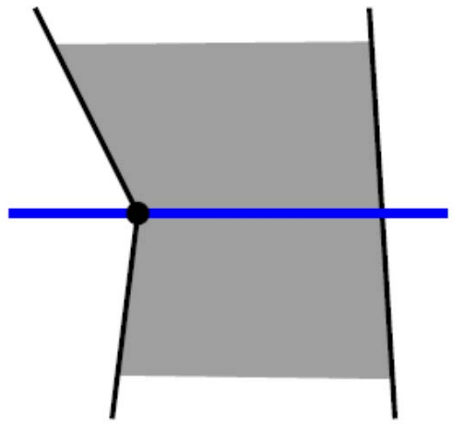
Event Handling

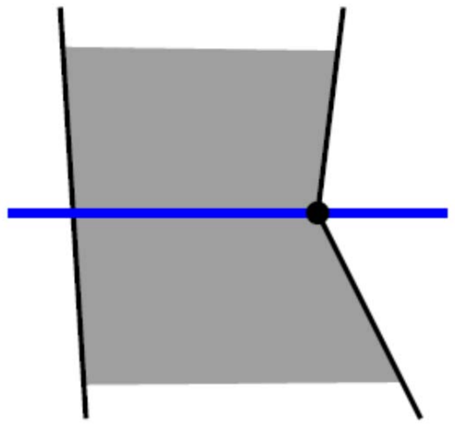
- End vertex v:
  - Delete the clockwise incident edge and its helper from T



Event Handling

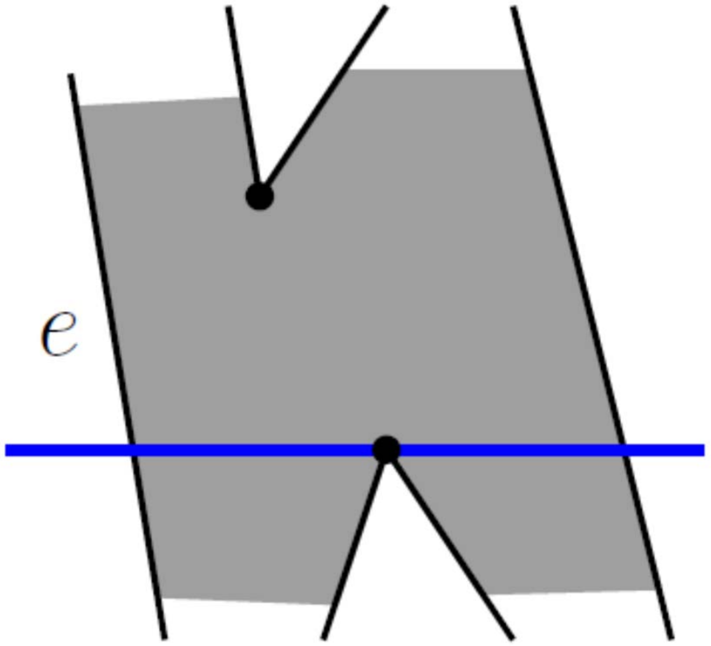
- Regular vertex v:
  - If the polygon is to the right of the two incident edges, then replace the upper edge by the lower edge in T, and make v the helper
  - If the polygon is to the left of the two incident edges, then find the edge e directly left of v, and replace its helper by v





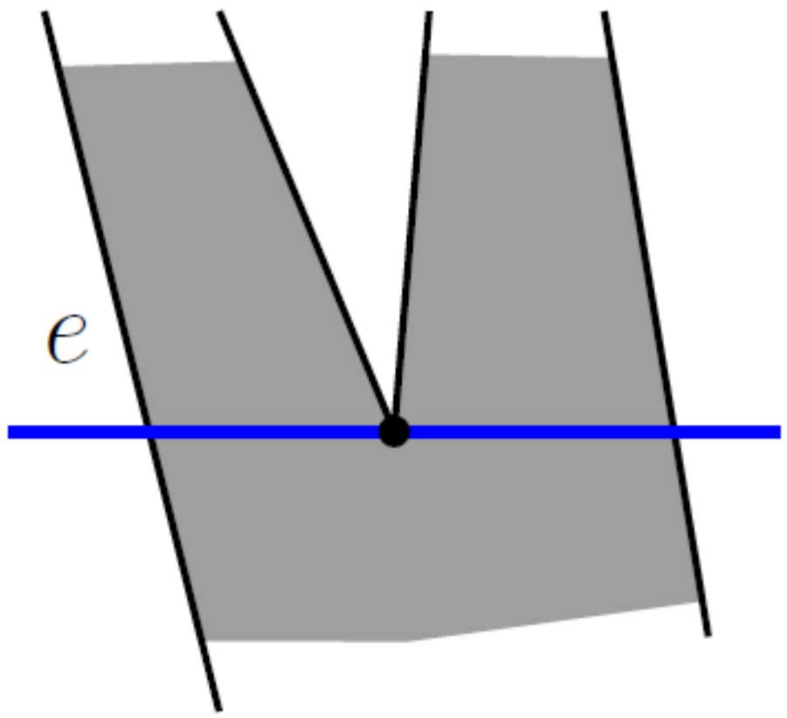
Event Handling

- Split vertex v:
  - Find the edge e directly left of v, and choose as a **diagonal** the edge between its helper and v
  - Replace the helper of e by v
  - Insert the edge counterclockwise from v in T, with v as its helper



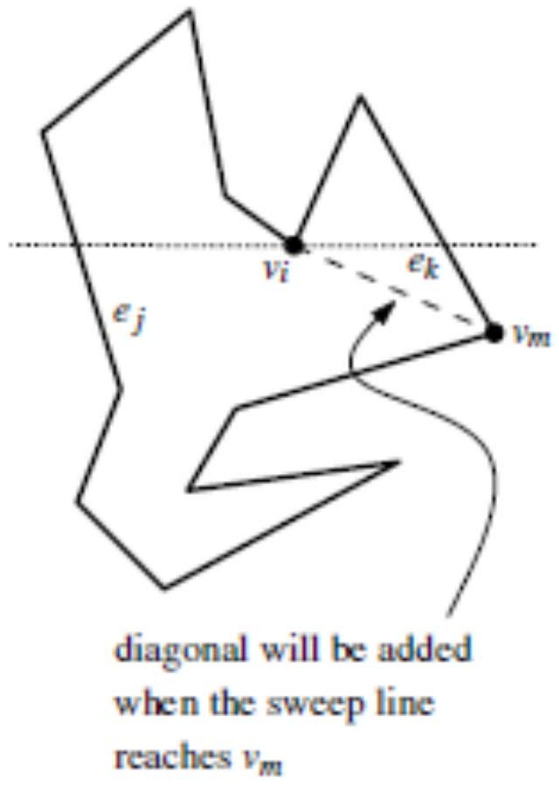
Event Handling

- Merge vertex v:
  - Remove the edge clockwise from v from T
  - Find the edge e directly left of v, and replace its helper by v



Event Handling

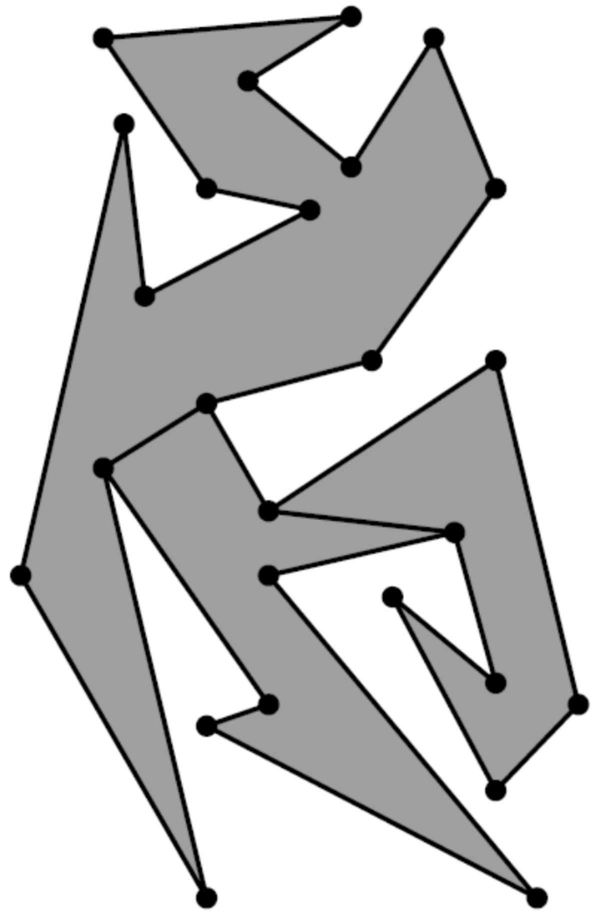
- Diagonal insertion
  - When we reach vertex v_m that replaces v_i as the helper of e_j, check whether old helper is a merge vertex
  - If yes, add diagonal between old and new helper



Efficiency

- Sorting all events by y-coordinate takes O(n log n) time

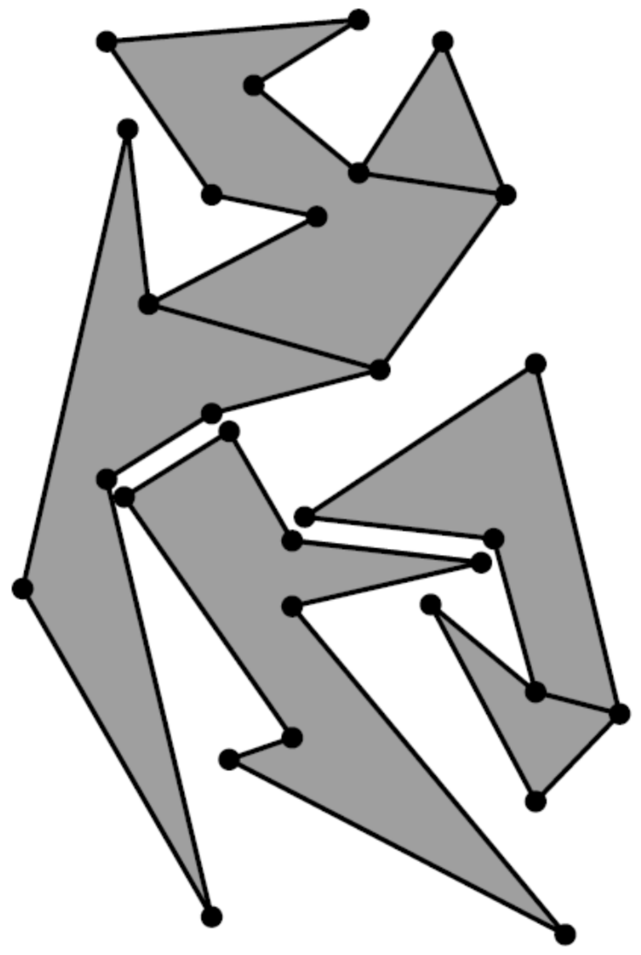
- Every event takes O(log n) time, because it only involves querying, inserting and deleting in T



More Sweeping

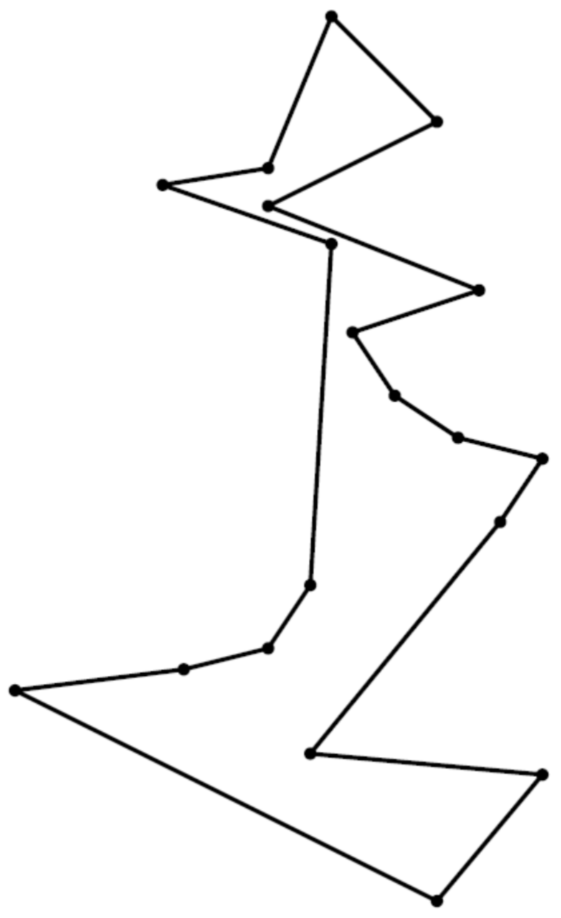
- With an upward sweep in each subpolygon, we can find a diagonal down from every merge vertex (which is a split vertex for an upward sweep!)

- This makes all subpolygons y-monotone



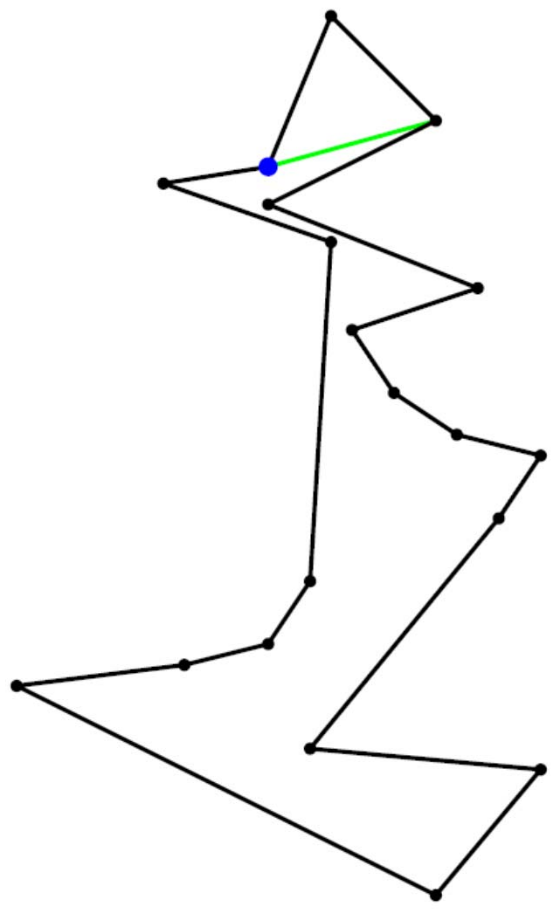
Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



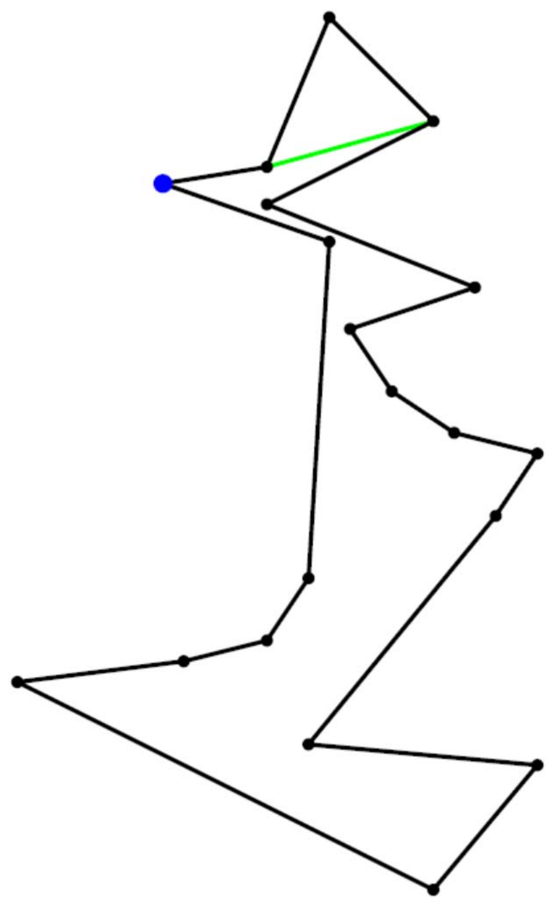
Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



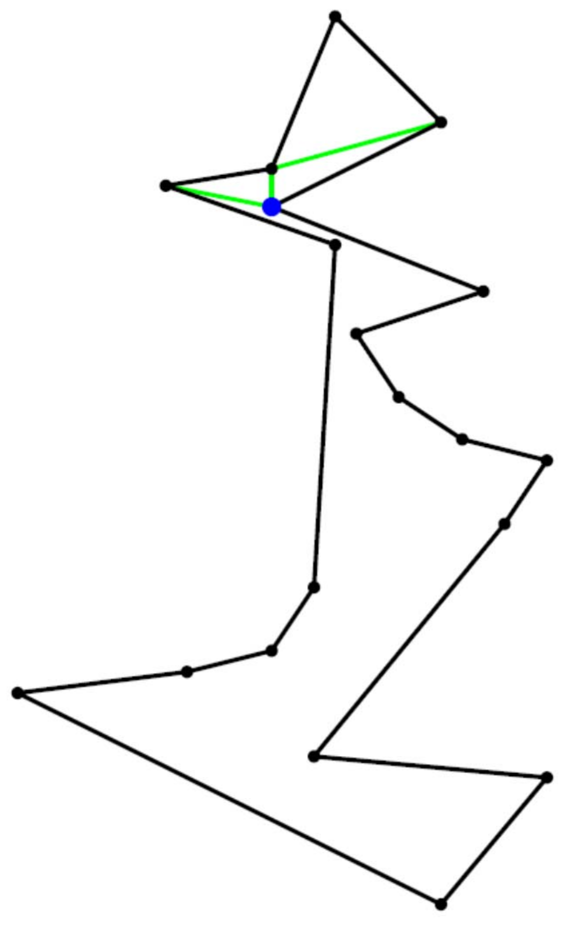
Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



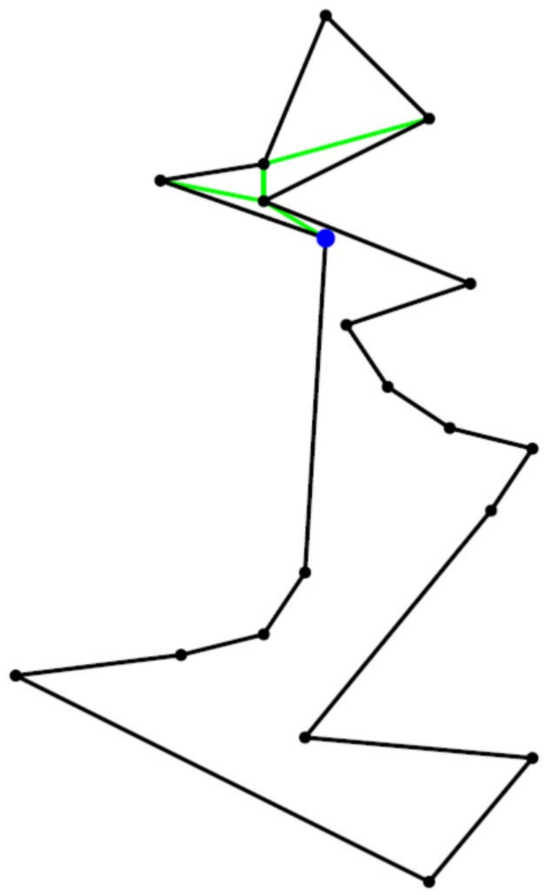
Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



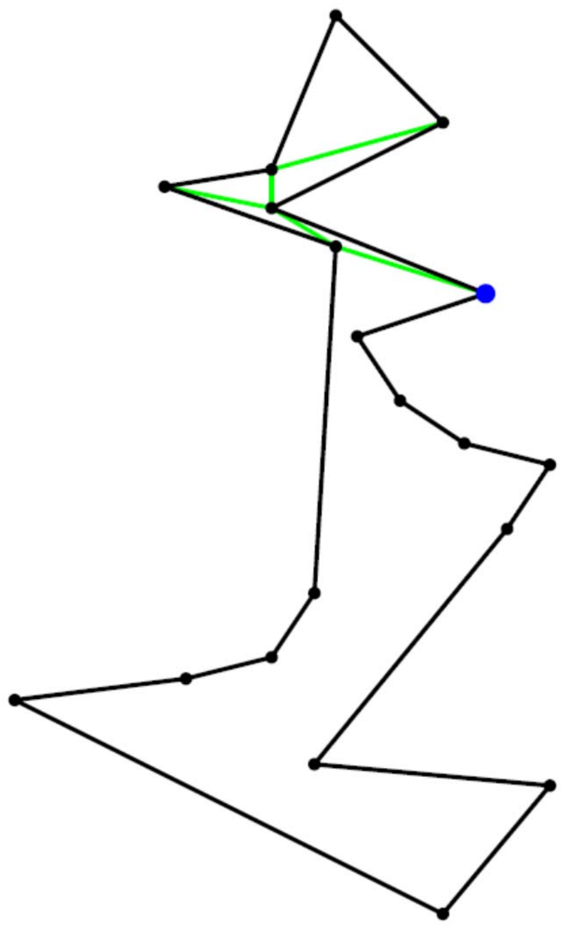
Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



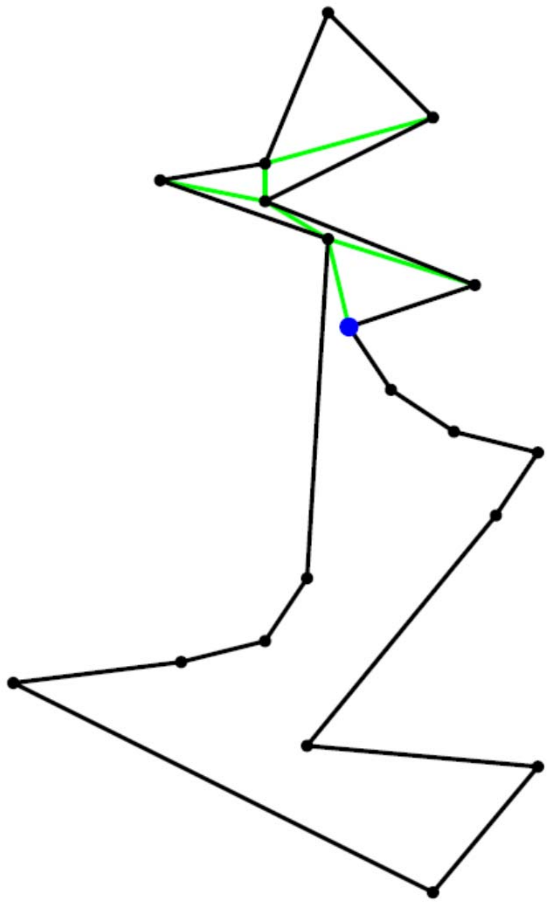
Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



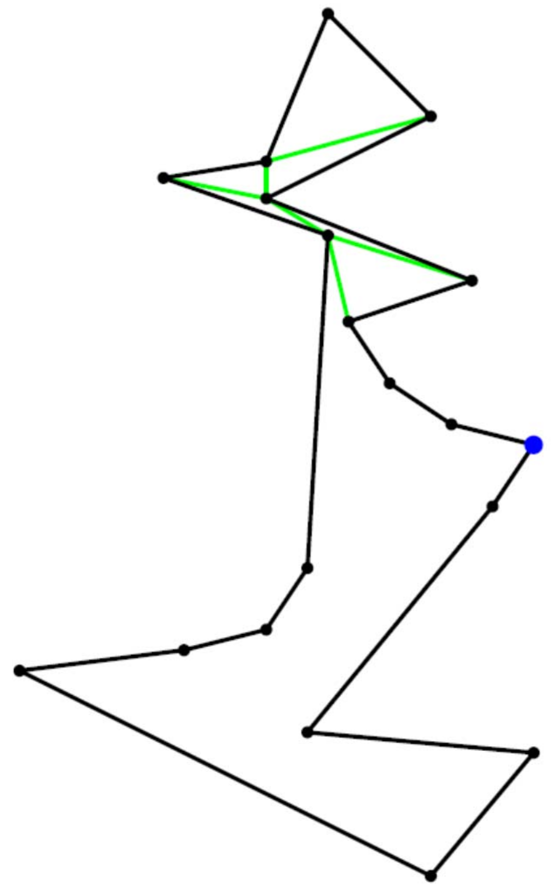
Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



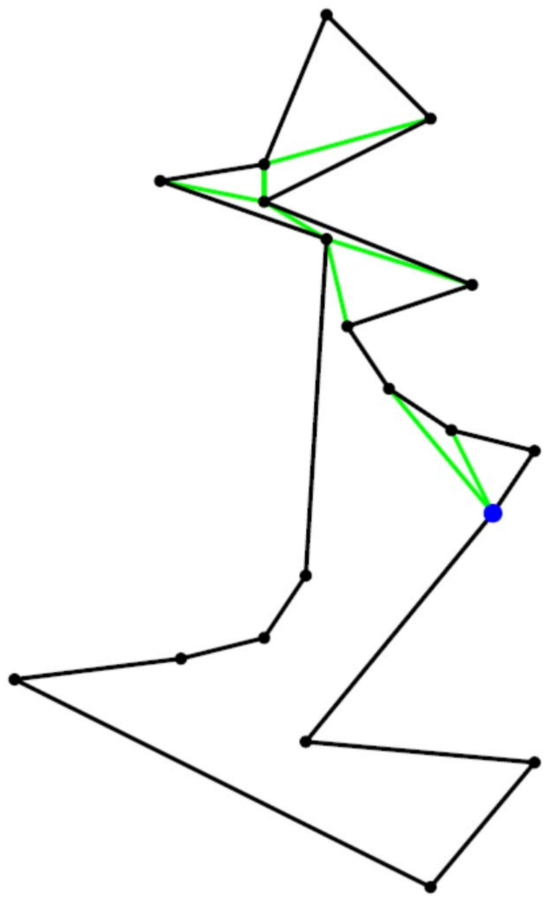
Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



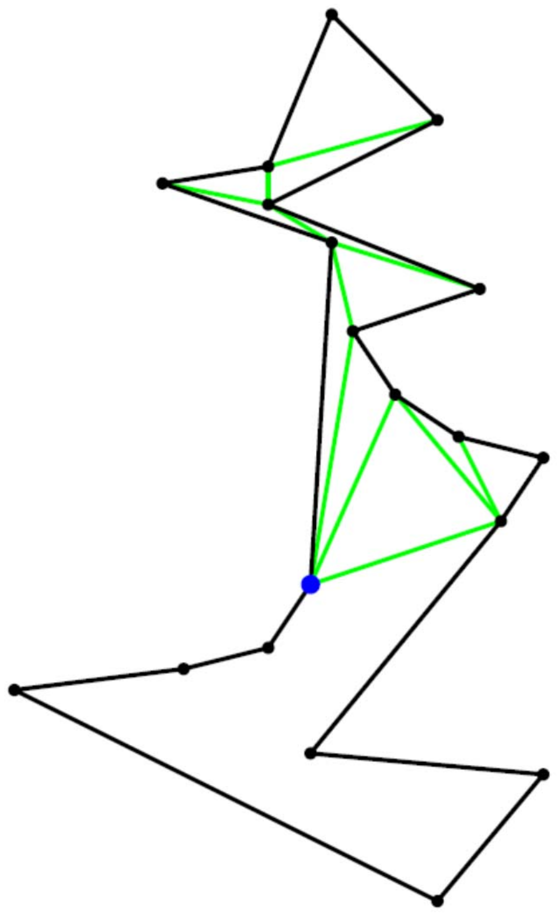
Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



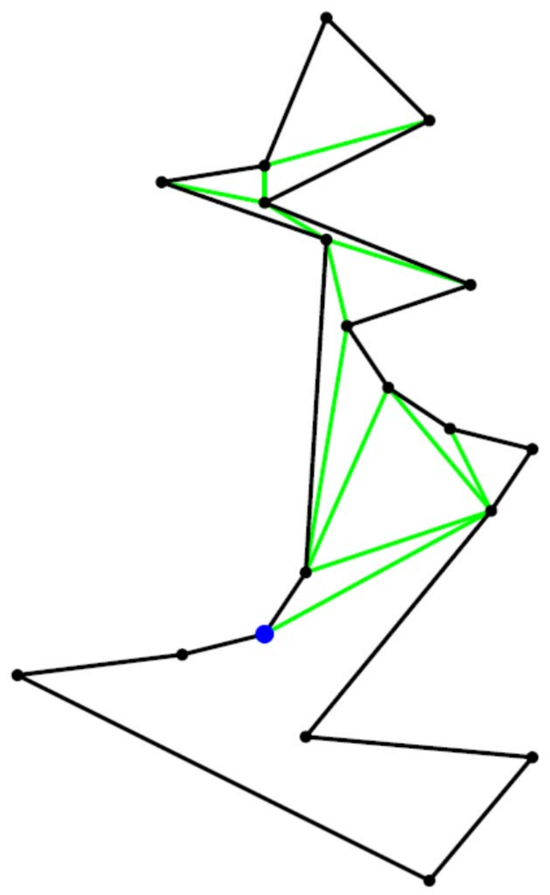
Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



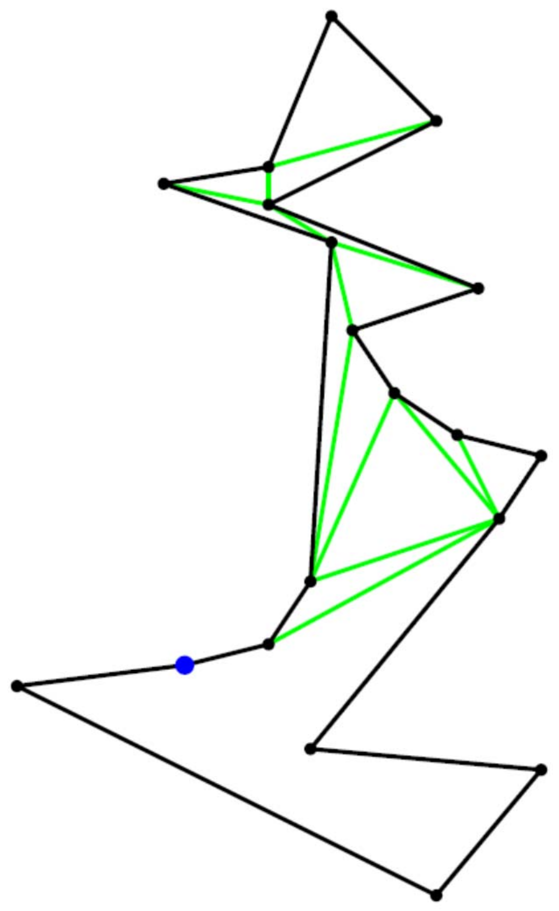
Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



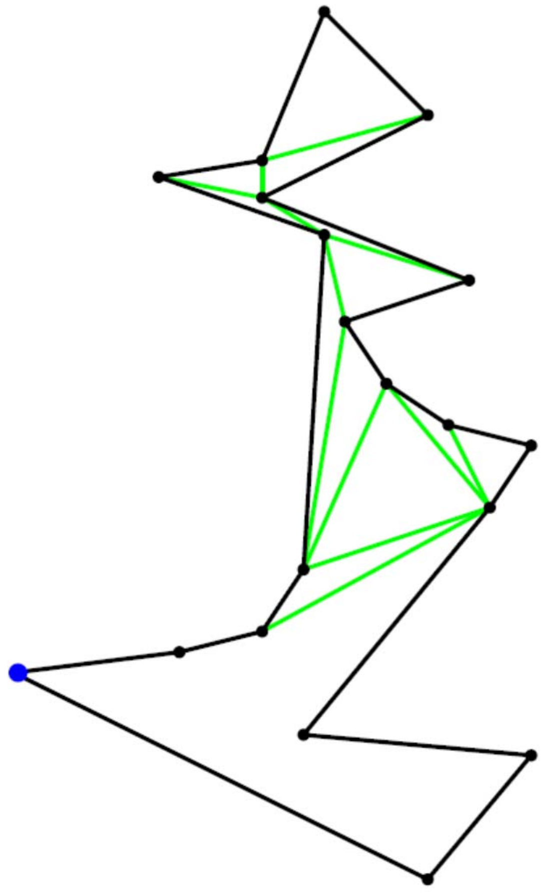
Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



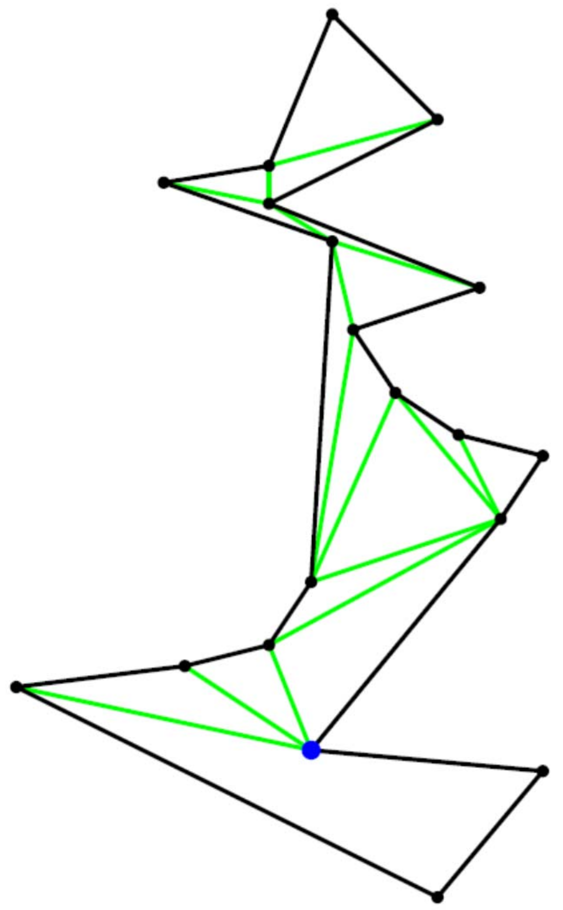
Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



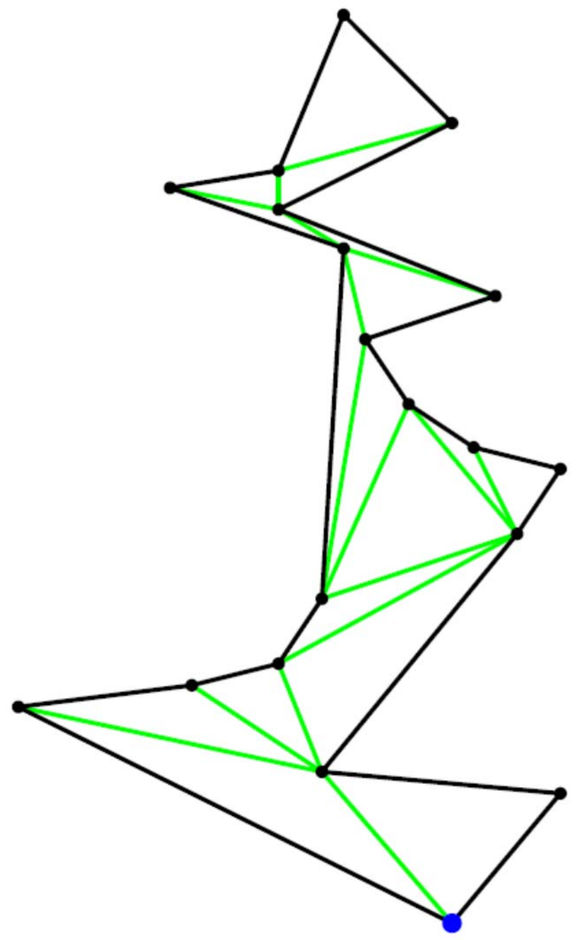
Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



Triangulating a Monotone Polygon

- How to triangulate a y-monotone polygon?



The Algorithm

- Sort the vertices top-to-bottom by a merge of the two chains
- Initialize a stack. Push the first two vertices
- Take the next vertex v, and triangulate as much as possible, top-down, while popping the stack
- Push v onto the stack

Result

- Theorem: A simple polygon with n vertices can be partitioned into y-monotone pieces in O(n log n) time
- Theorem: A monotone polygon with n vertices can be triangulated O(n) time

- Can we immediately conclude:
- A simple polygon with n vertices can be triangulated in O(n log n) time ???

Result

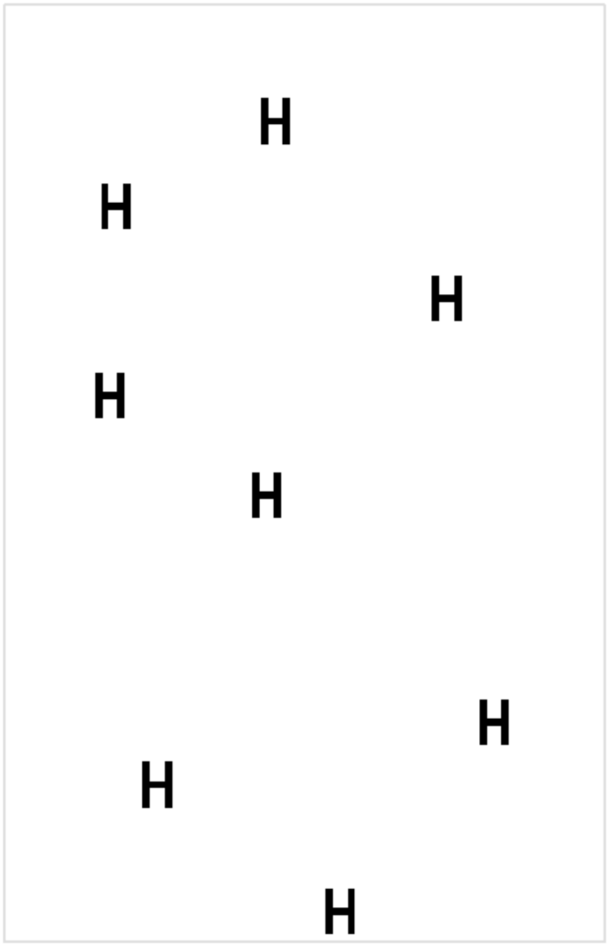
- We need to argue that all y-monotone polygons that we will triangulate have O(n) vertices together
- Initially we had n edges. We add at most n-3 diagonals in the sweeps. These diagonals are used on both sides as edges. So all monotone polygons together have at most 3n-6 edges, and therefore at most 3n-6 vertices
- Hence we can conclude that triangulating all monotone polygons together takes only O(n) time
- Theorem: A simple polygon with n vertices can be triangulated O(n log n) time

Voronoi Diagrams

Slides by M. van Kreveld
(Utrecht University)

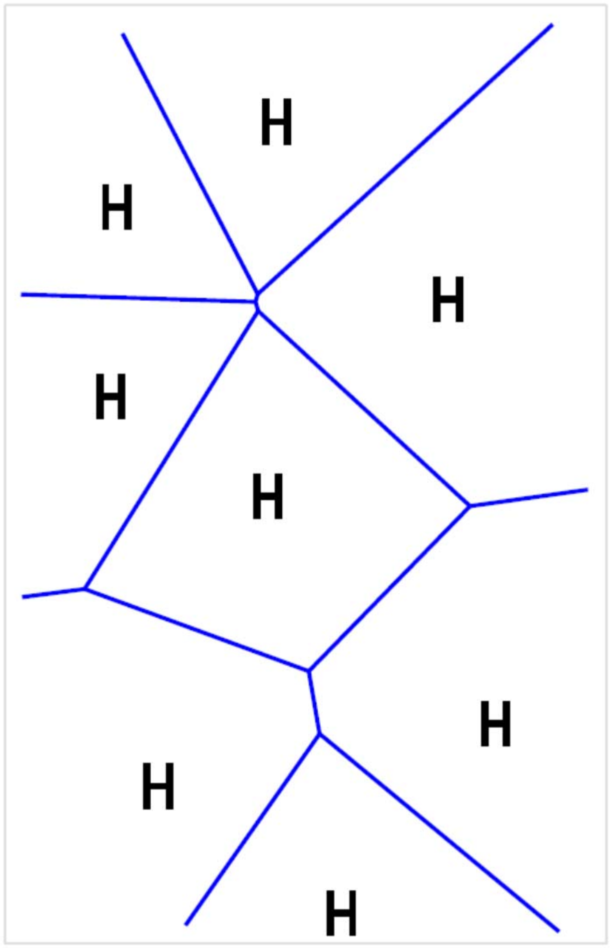
Voronoi Diagram

- Given ambulance posts, where should the ambulance come from in case of an emergency somewhere?



Voronoi Diagram

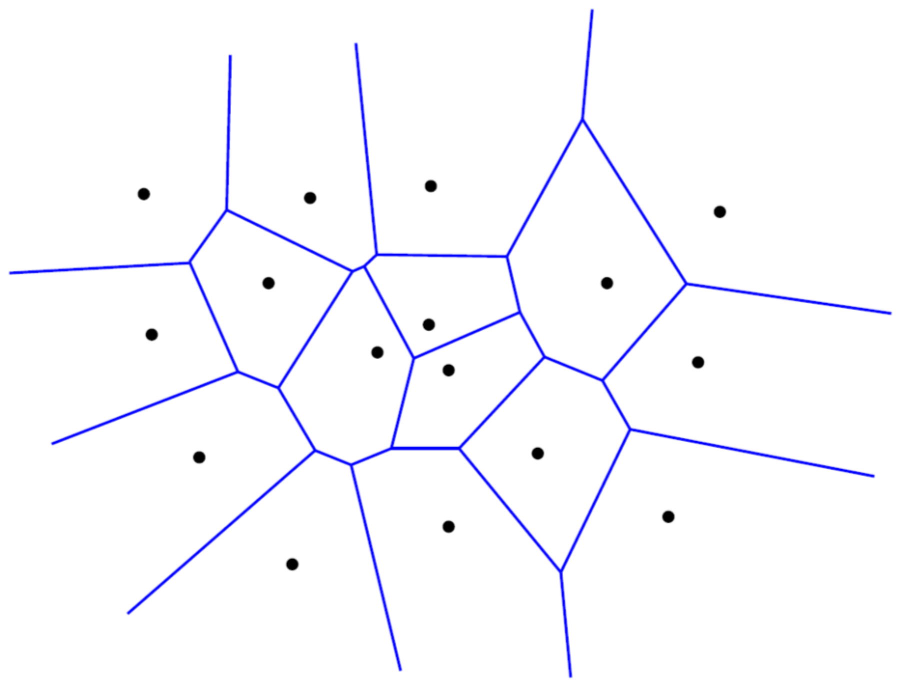
- Given ambulance posts, where should the ambulance come from in case of an emergency somewhere?



Voronoi Diagram

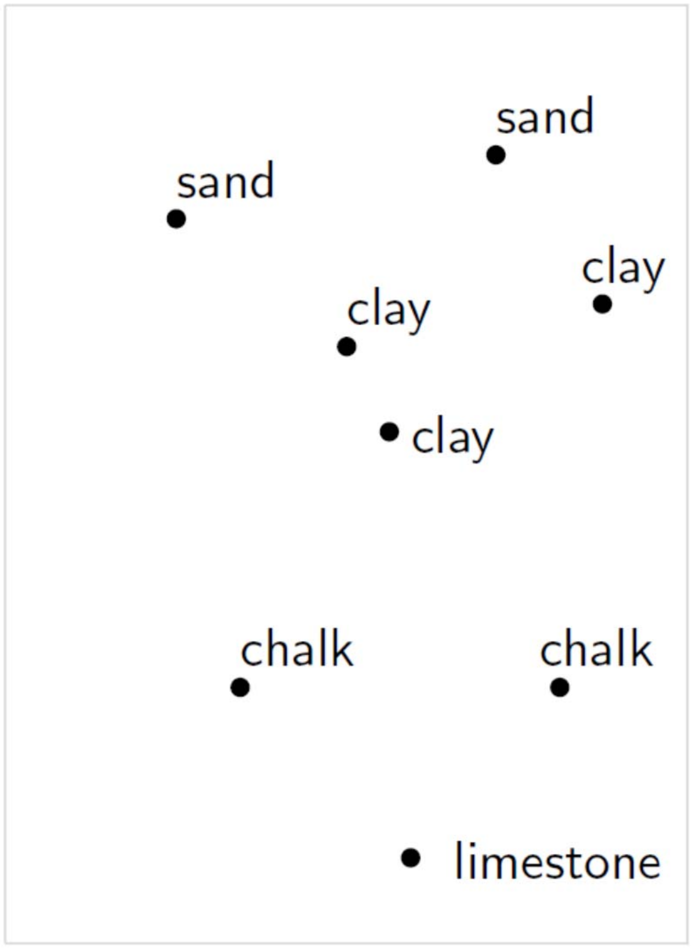
Definition

- The Voronoi diagram induced by a set of points (called sites):
- Subdivision of the plane where the faces correspond to the regions where one site is closest



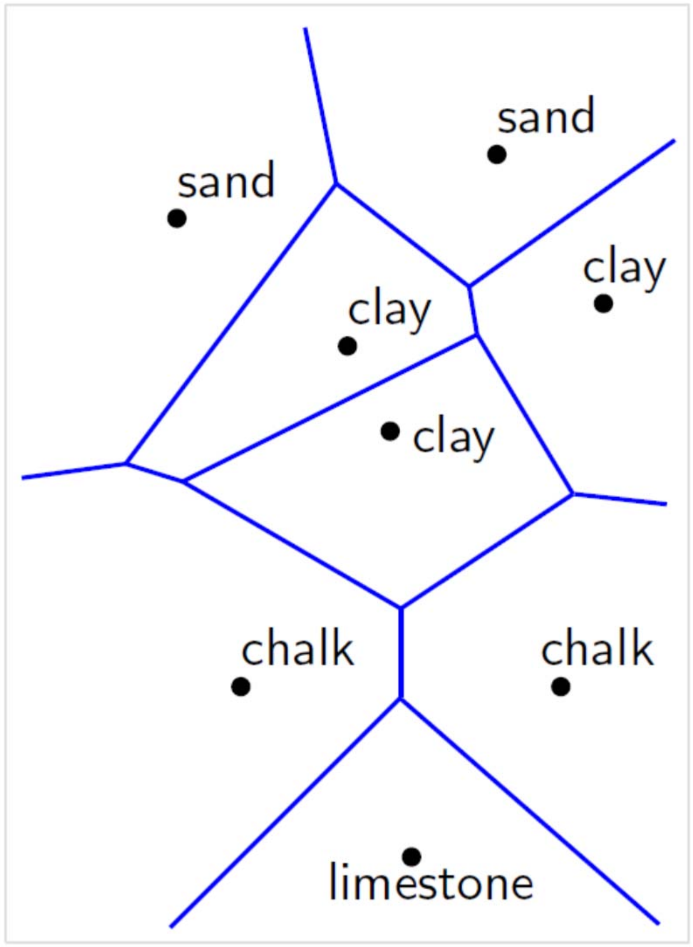
Spatial Interpolation

- Suppose we tested the soil at a number of sample points and classified the results



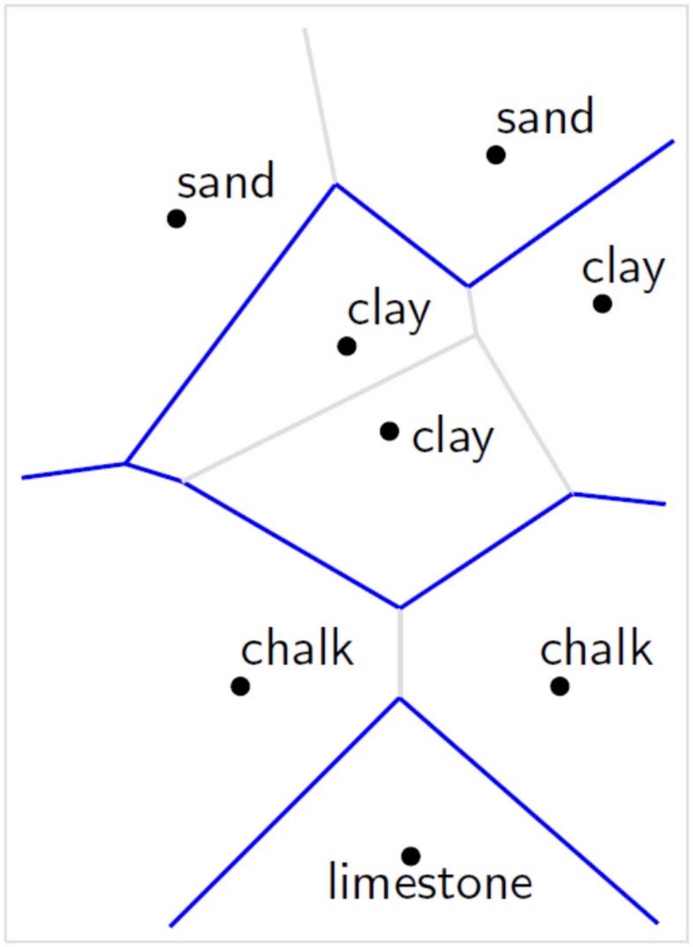
Spatial Interpolation

- Suppose we tested the soil at a number of sample points and classified the results



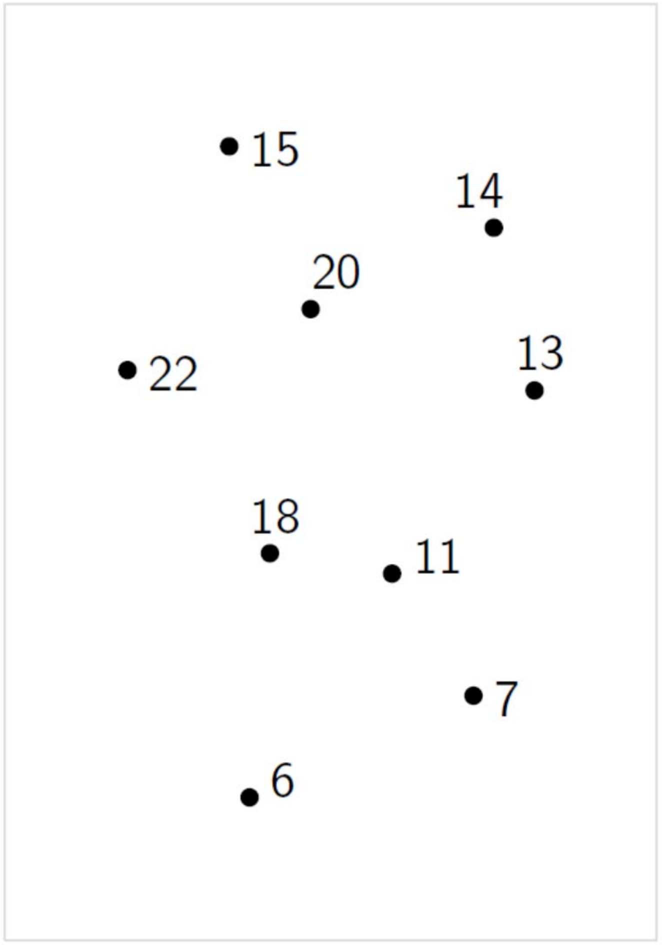
Spatial Interpolation

- Suppose we tested the soil at a number of sample points and classified the results



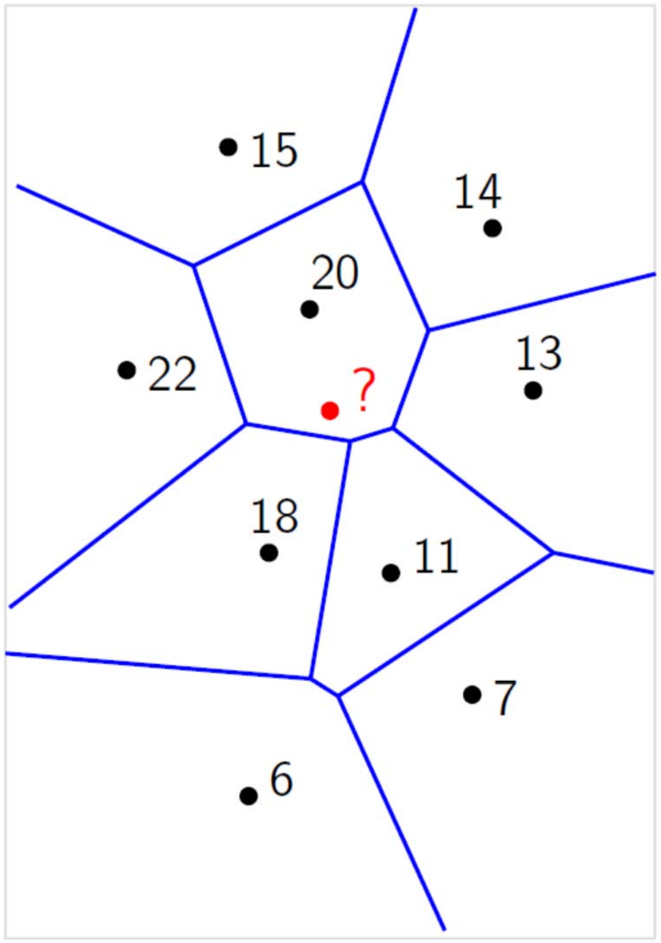
Spatial Interpolation

- Suppose we measured lead concentration at a number of sample points



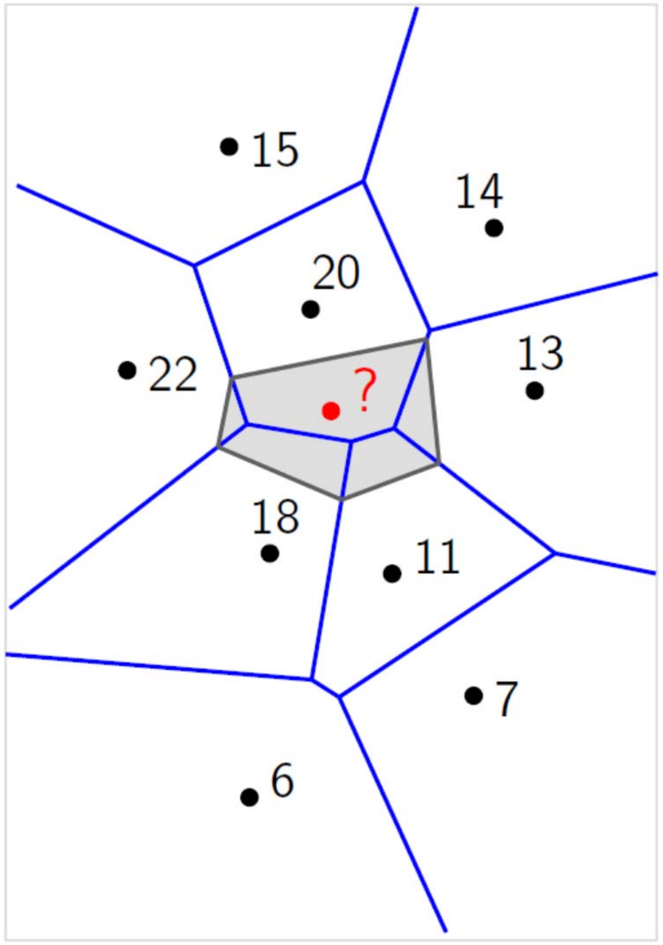
Spatial Interpolation

- Suppose we measured lead concentration at a number of sample points



Spatial Interpolation

- Suppose we measured lead concentration at a number of sample points (natural neighbor interpolation)

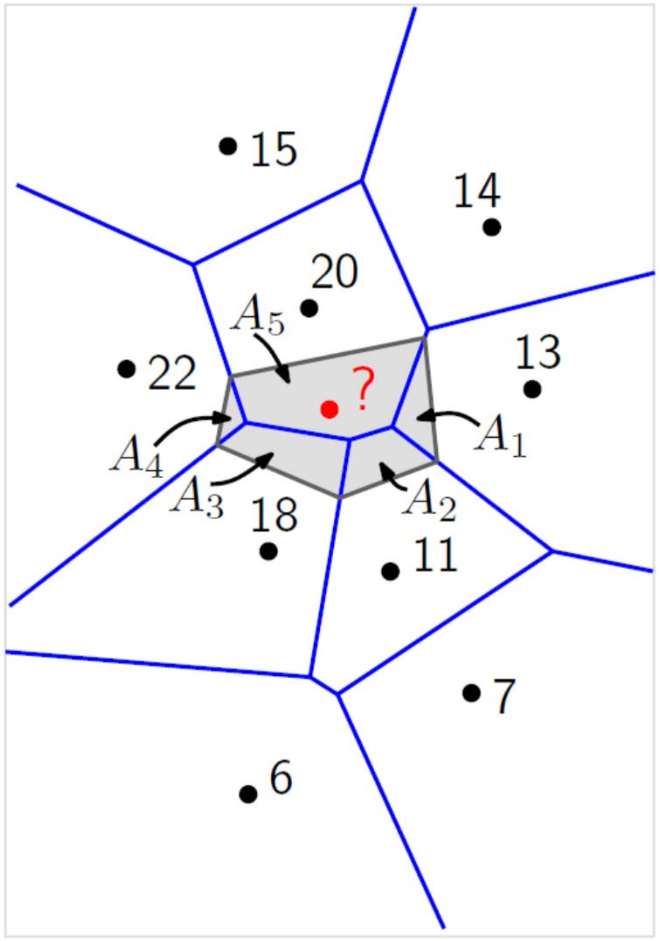


Spatial Interpolation

Natural neighbor interpolation

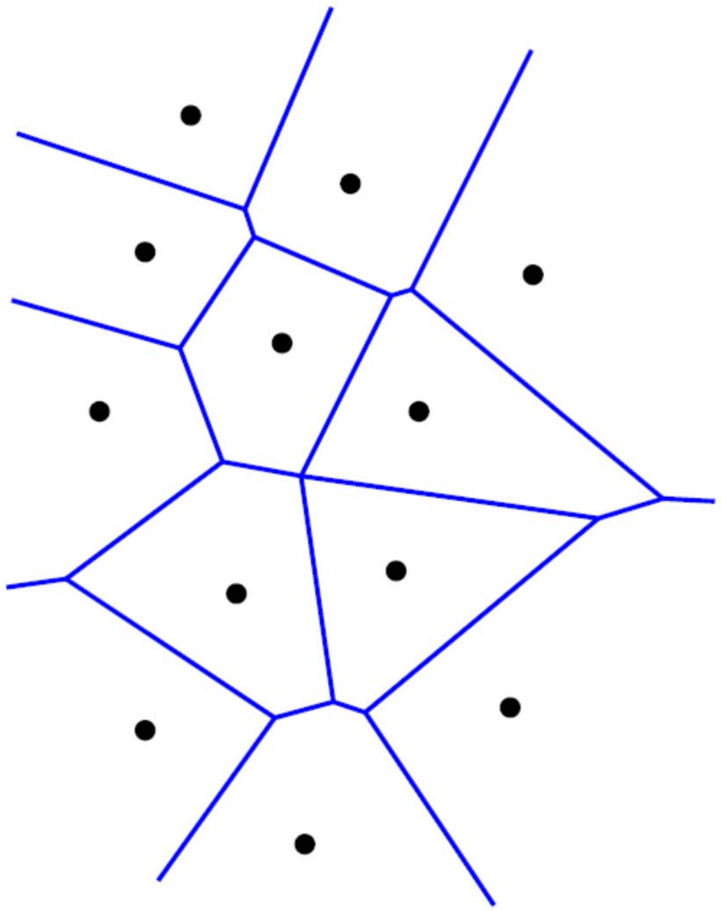
- Let $A_T = A_1 + A_2 + \cdot\cdot\cdot + A_5$
- The interpolated value is:

$$\frac{A_1}{A_T}13 + \frac{A_2}{A_T}11 + \cdots + \frac{A_5}{A_T}20$$



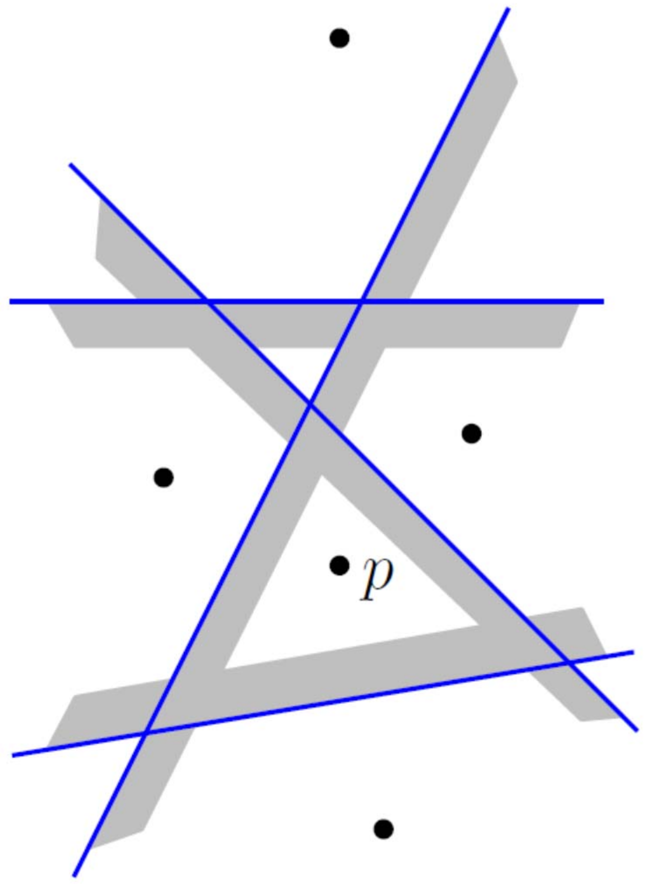
Observations

- Edges are parts of bisectors
- Some edges are half-infinite
- Some cells are unbounded



Every Voronoi cell is the intersection of n−1 half-planes, if there are n sites

=> all cells are convex and have up to n−1 edges in the boundary



Structure

- The Voronoi diagram of n sites has the following structure:
- If all n sites lie on a line, then the Voronoi cell boundaries are parallel lines, so the “graph” is disconnected
- Otherwise, the Voronoi cell boundaries form a connected “graph”

Complexity

- **Theorem:** The Voronoi diagram on f sites in the plane has at most 2n−5 Voronoi vertices and at most 3n−6 Voronoi edges (including lines and half-lines)
- **Proof:** If the sites are colinear, then it is trivial
- Otherwise, we will use Euler's formula for planar graphs

Complexity

- Euler’s formula for planar graphs: a connected planar graph with n_v vertices, n_e edges, and n_f faces satisfies:

$$n_v - n_e + n_f = 2$$

- However, a Voronoi diagram is not a proper graph

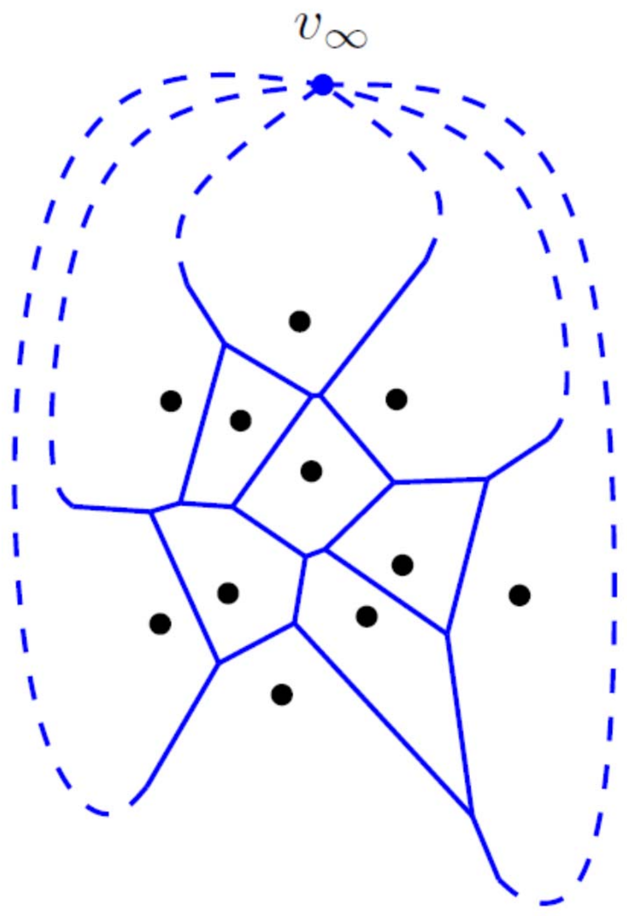
Complexity

We make it proper by connecting all half-infinite edges to a new vertex v_∞

n_v = no. of Voronoi vertices VV +1

n_e = no. of Voronoi edges VE

n_f = no. of Voronoi cells = n, the number of sites



Complexity

- Substitution in Euler's formula $n_v - n_e + n_f = 2$

gives:

$$(VV + 1) - VE + n = 2$$

- Every edge is incident to exactly 2 vertices, and every vertex is incident to at least 3 edges

- Sum-of-degree-of-all-vertices = 2 VE
- Sum-of-degree-of-all-vertices ≥ 3 VV

- Thus 2 VE ≥ 3 VV

Complexity

The combination of

$$(VV + 1) - VE + n = 2$$

and

$$2\ VE \geq 3\ (VV+1)$$

gives the desired bounds $VV \leq 2n-5$ and $VE \leq 3n-6$

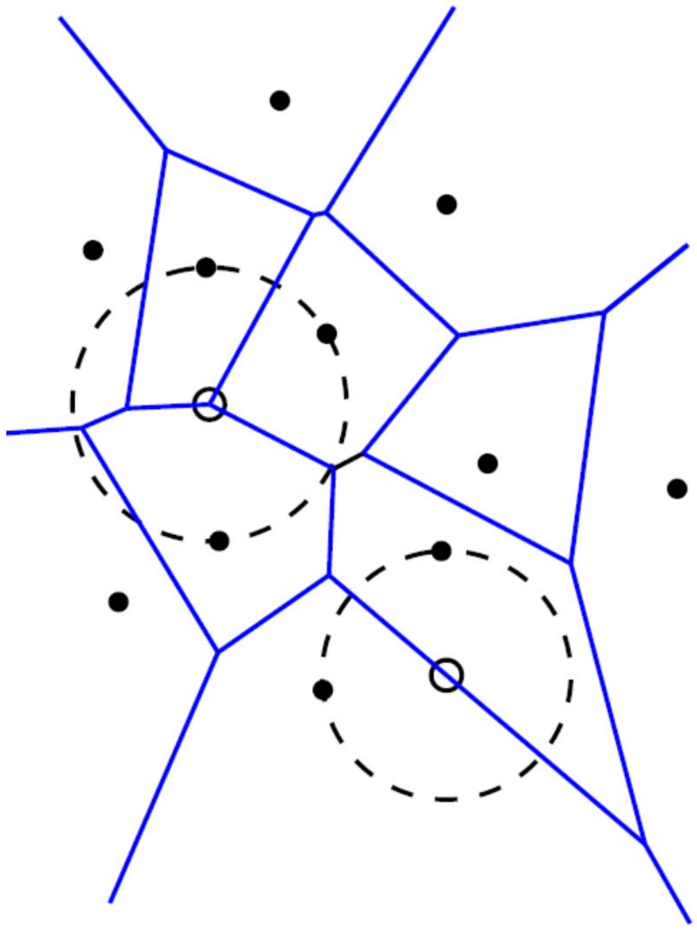
Time Complexity

- **Theorem:** The Voronoi diagram of a set of n point sites in the plane can be computed in O(nlogn) time
- Algorithms
  - Compute the intersection of n−1 half-planes for each site, and "merge" the cells into the diagram
  - Divide-and-conquer (1975, Shamos & Hoey)
  - Plane sweep (1987, Fortune)
  - Randomized incremental construction (1992, Guibas, Knuth & Sharir)

Empty Circle Property

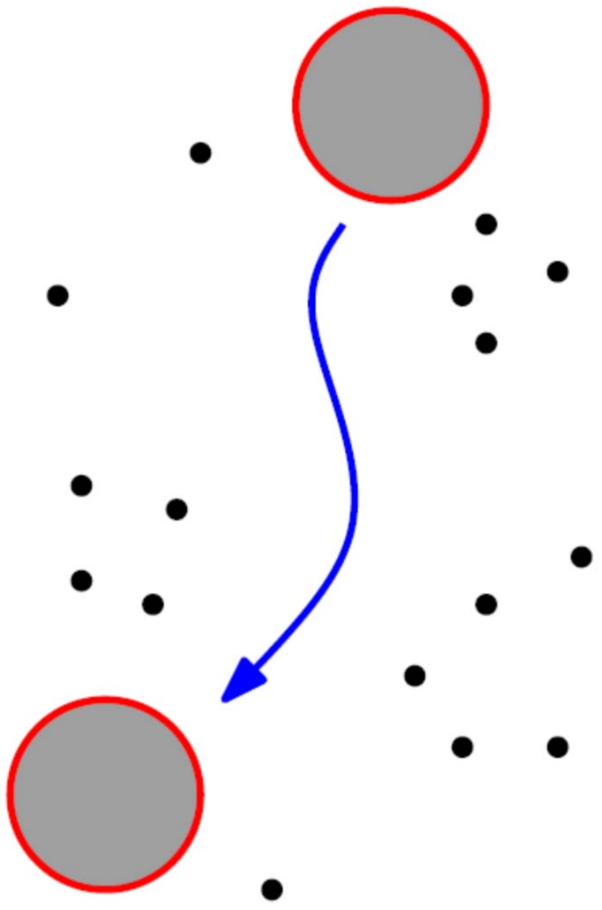
- Every Voronoi vertex is the center of an empty circle through 3 sites
- Every point on a Voronoi edge is the center of an empty circle through 2 sites

- Prove it!



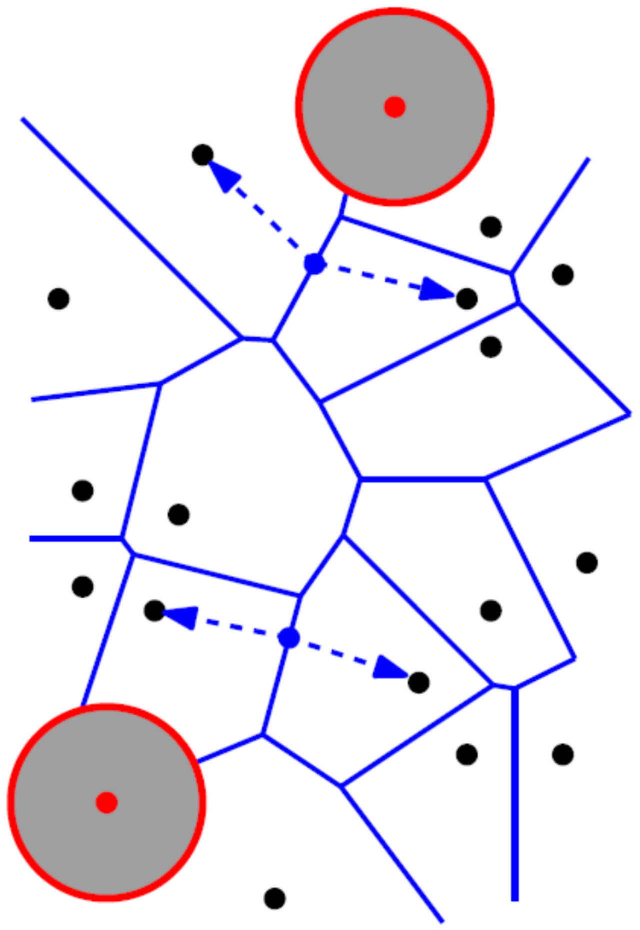
Motion Planning for a Disc

- Can we move a disc from one location to another amidst obstacles?



Motion Planning for a Disc

- Since the Voronoi diagram of point sites is locally “furthest away” from those sites, we can move the disc if and only if we can do so on the Voronoi diagram



Delaunay Triangulations

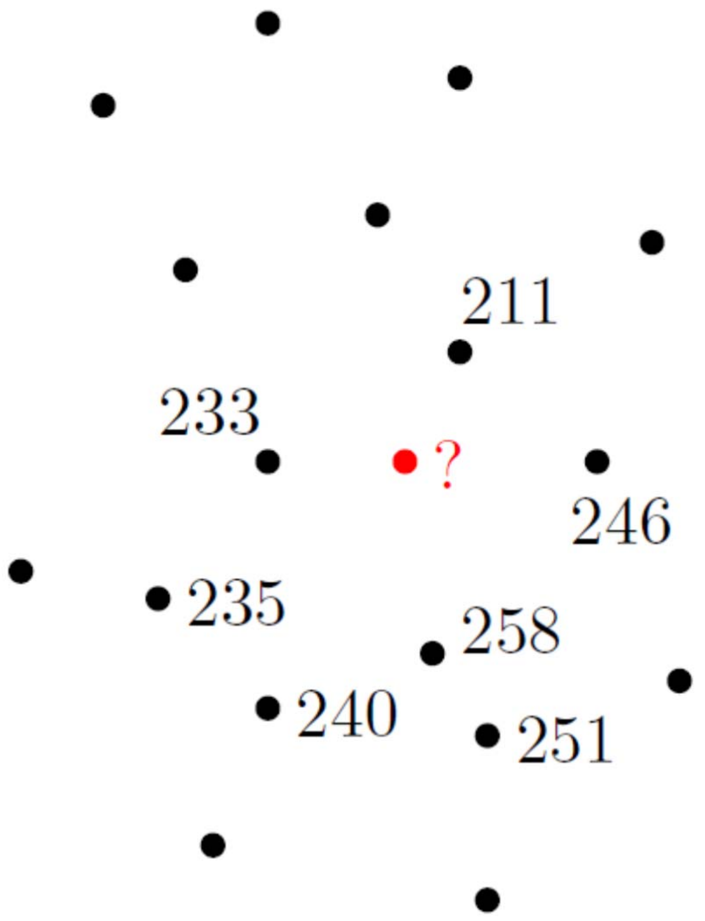
Slides by M. van Kreveld
(Utrecht University)

Motivation: Terrains by Interpolation

To build a model of the terrain surface, we can start with a number of sample points where we know the height.

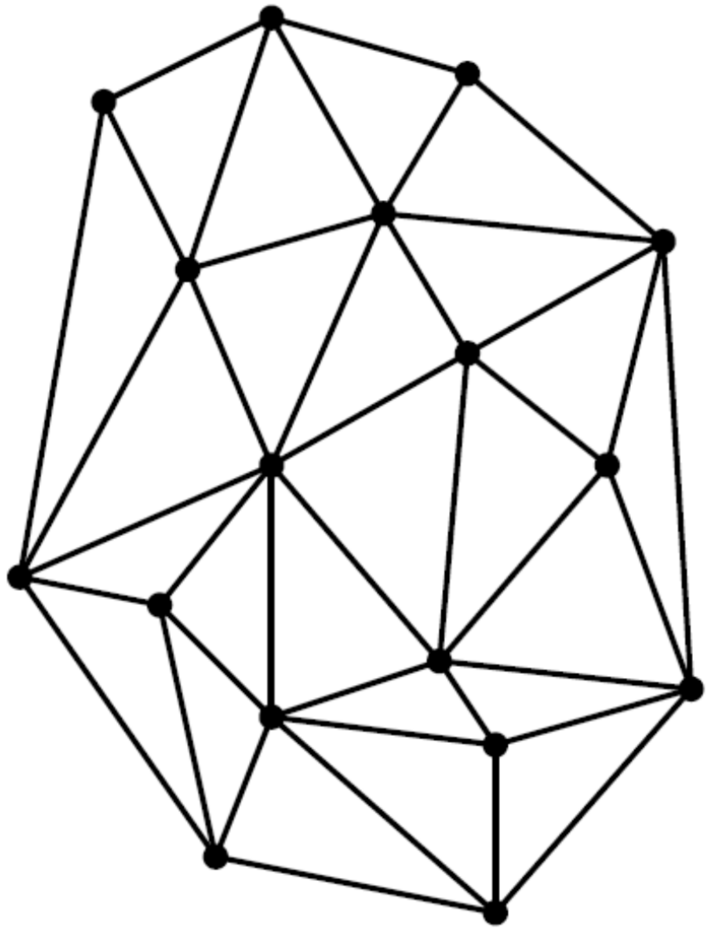
Motivation: Terrains

- How do we interpolate the height at other points?
  - Nearest neighbor interpolation
  - Piecewise linear interpolation by a triangulation
  - Natural neighbor interpolation

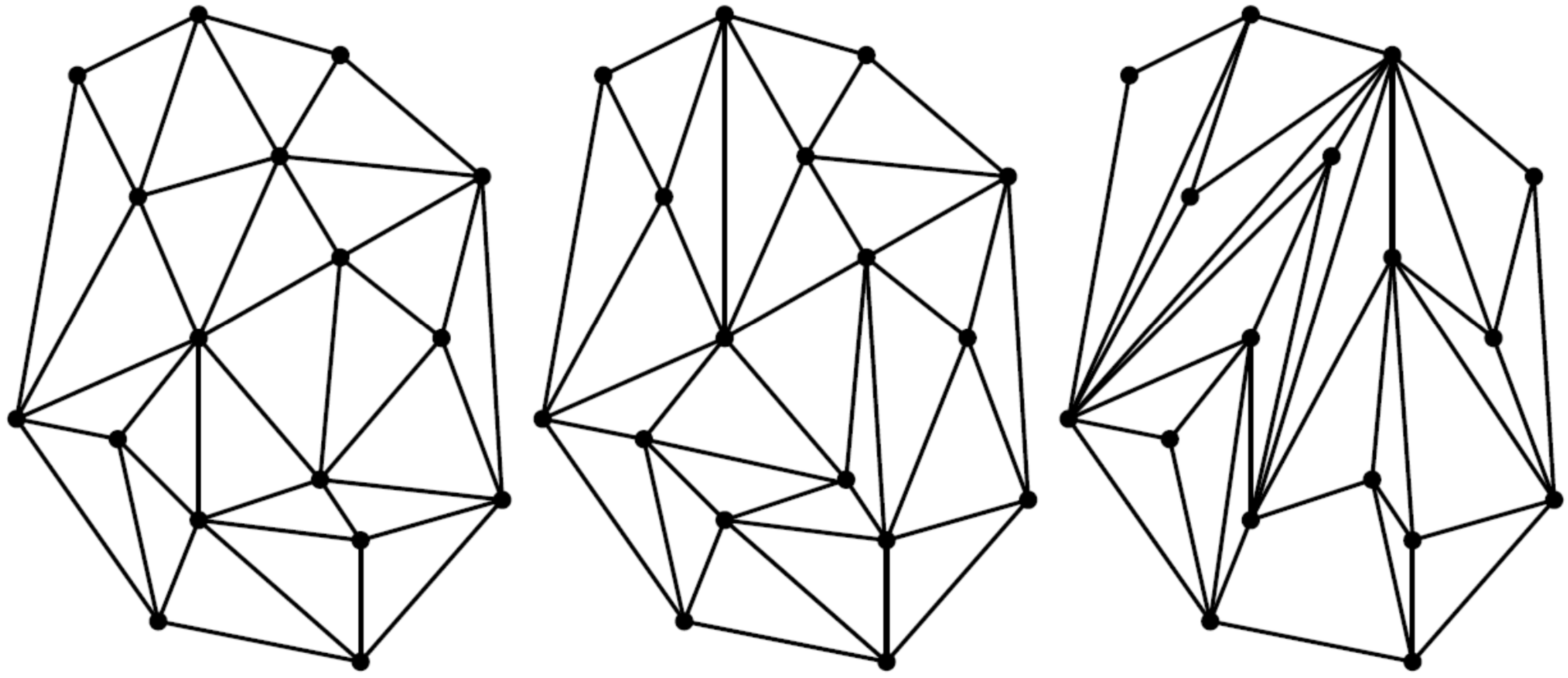


Triangulation

- Let P = {p_1, …, p_n} be a point set
- A triangulation of P is a maximal planar subdivision with vertex set P
- Complexity:
    - 2n-2-k triangles
    - 3n-3-k edges
- where k is the number of points in P on the convex hull of P



Which Triangulation?

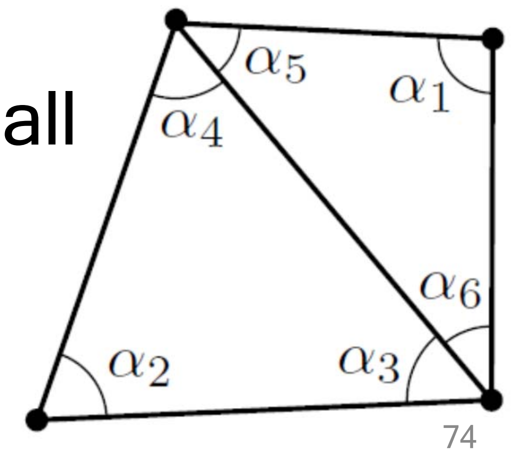


Triangulation

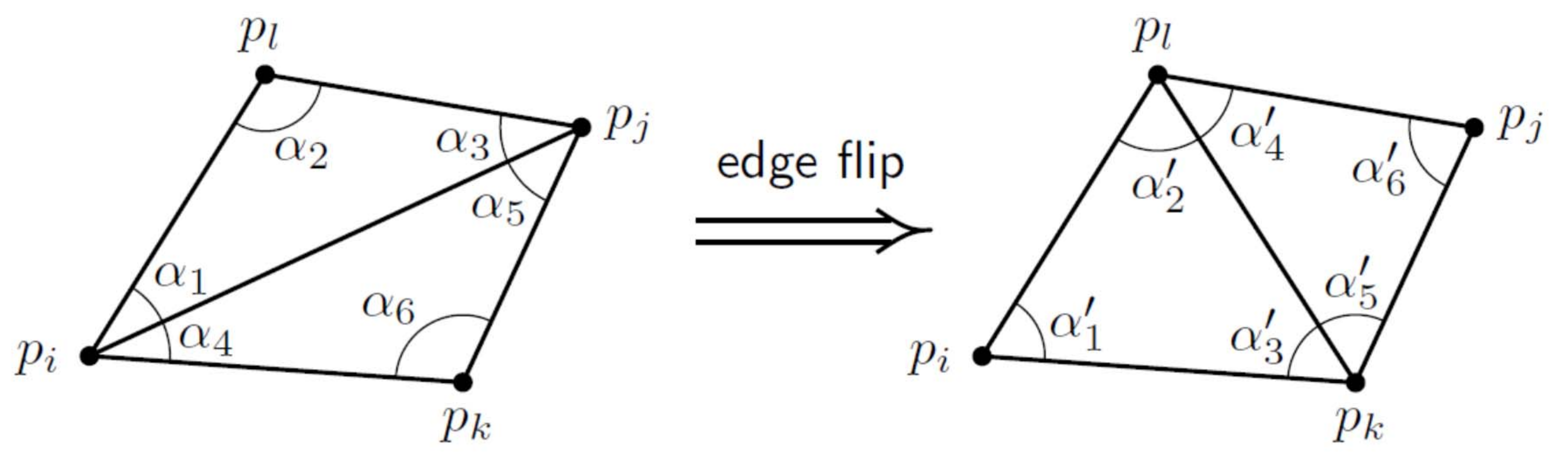
- For interpolation, it is good if triangles are not long and skinny. We will try to use large angles in our triangulation.

Angle Vector of a Triangulation

- Let T be a triangulation of P with m triangles. Its angle vector is A(T) = (a_1, …, a_{3m}) where a_1, …, a_{3m} are the angles of T sorted by increasing value.
- Let T' be another triangulation of P. We define A(T) > A(T') if A(T) is lexicographically larger than A(T')
- T is angle optimal if A(T)≥A(T') for all triangulations T' of P.



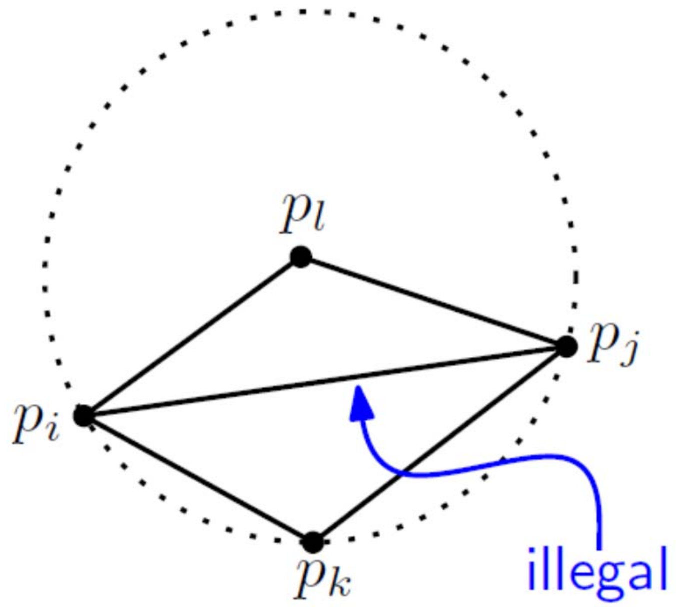
Edge Flipping



- An edge is illegal if $\min\{a_i\} < \min\{a_i'\}$
- Flipping an illegal edge increases the angle vector

Illegal Edges

- An edge p_ip_j is illegal if an only if p_l lies in the interior of the circle C

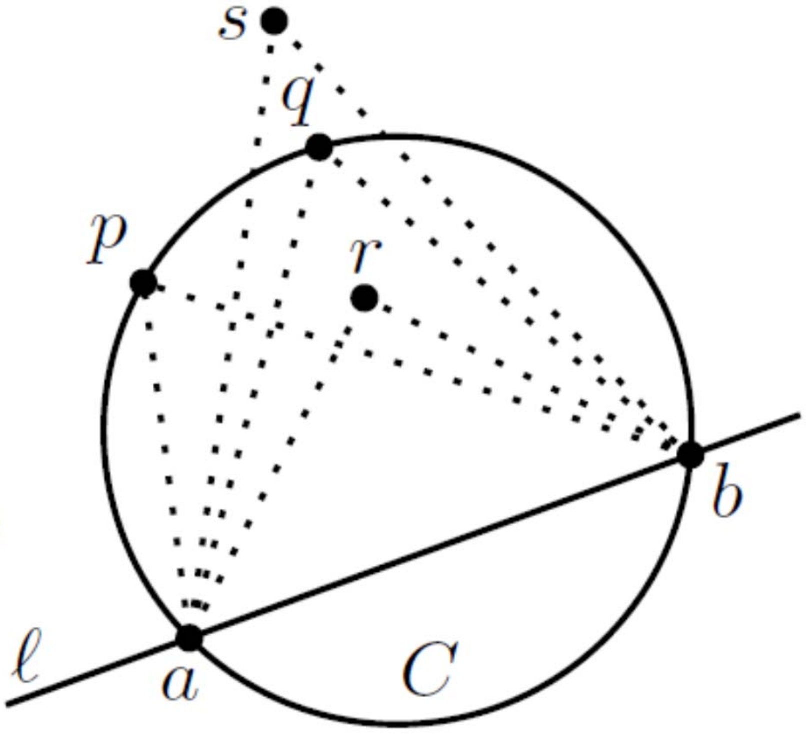


Thales Theorem

- Theorem: Let C be a circle, l a line intersecting C in points a and b, and p, q, r, s points lying on the same side of l. Suppose that p, q lie on C, r lies inside C, and s lies outside C. Then:

$$\measuredangle arb > \measuredangle apb = \measuredangle aqb > \measuredangle asb,$$

- Where $\measuredangle arb$ denotes the smaller angle defined by three points a, b, c.



Legal Triangulations

A legal triangulation is a triangulation that does not contain any illegal edge.

Algorithm LEGALTRIANGULATION($\mathcal{T}$)
Input. A triangulation $\mathcal{T}$ of a point set P.
Output. A legal triangulation of P.

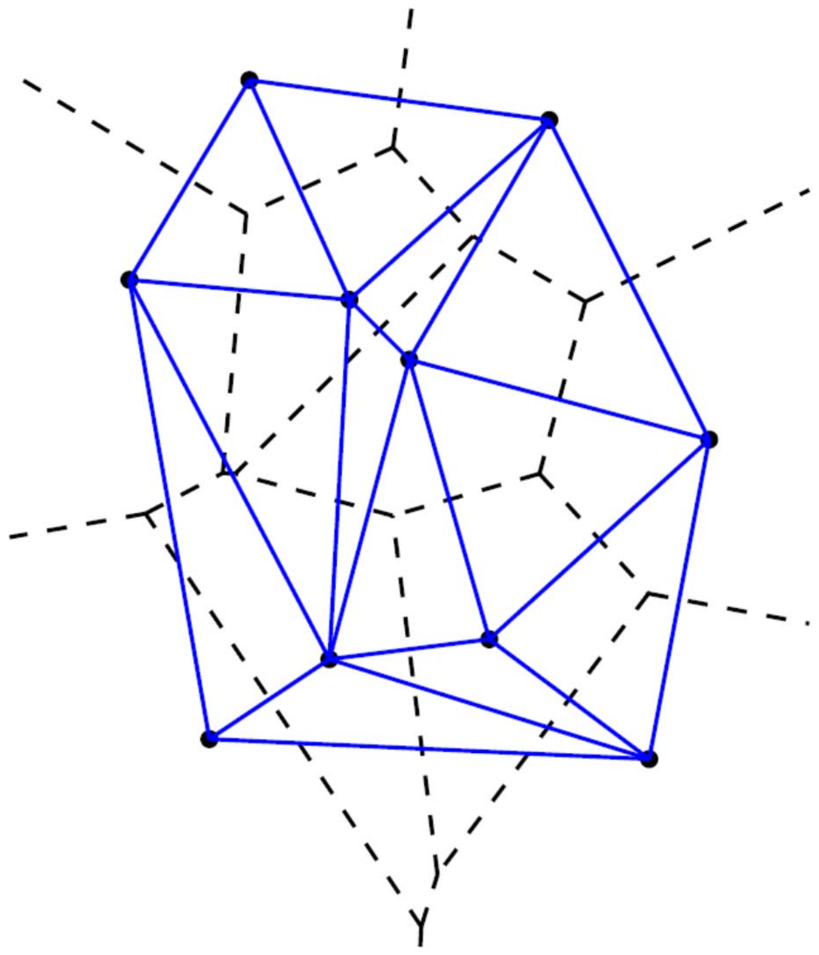
```
1.  while $\mathcal{T}$ contains an illegal edge $\overline{p_ip_j}$
2.      do (* Flip $\overline{p_ip_j}$ *)
3.          Let $p_ip_jp_k$ and $p_ip_jp_l$ be the two triangles adjacent
            to $\overline{p_ip_j}$.
4.          Remove $\overline{p_ip_j}$ from $\mathcal{T}$, and add $\overline{p_kp_l}$ instead.
5.  return $\mathcal{T}$
```

Question: Why does this algorithm terminate?

Voronoi Diagram and Delaunay Graph

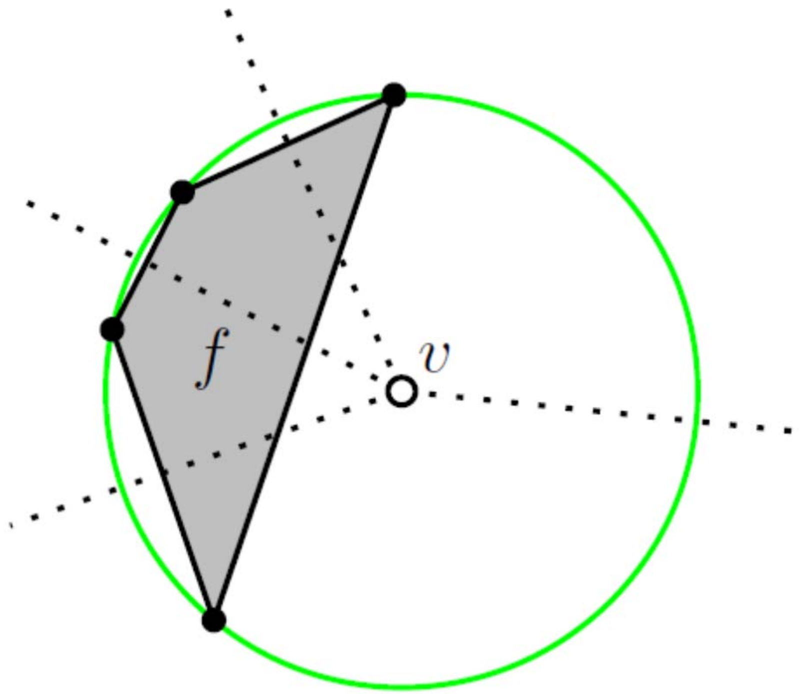
- Let P be a set of n points in the plane
- The Voronoi diagram Vor(P) is the subdivision of the plane into Voronoi cells V(p)

- Let G be the dual graph of Vor(P)
- The Delaunay graph DG(P) is the straight line embedding of G



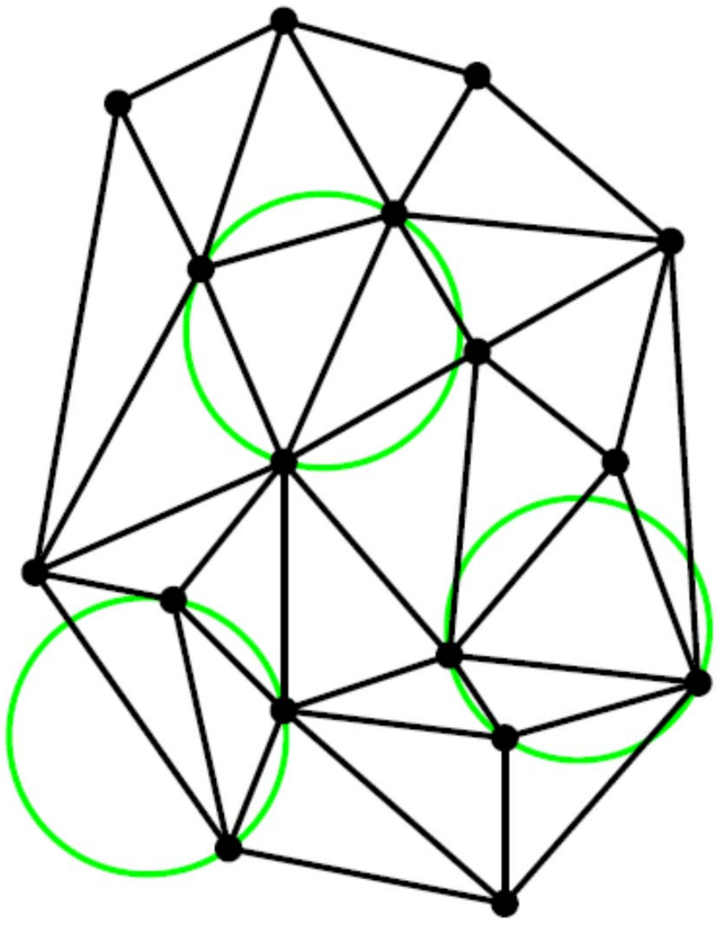
Delaunay Triangulation

- If the point set P is in general position, then the Delaunay graph is a triangulation



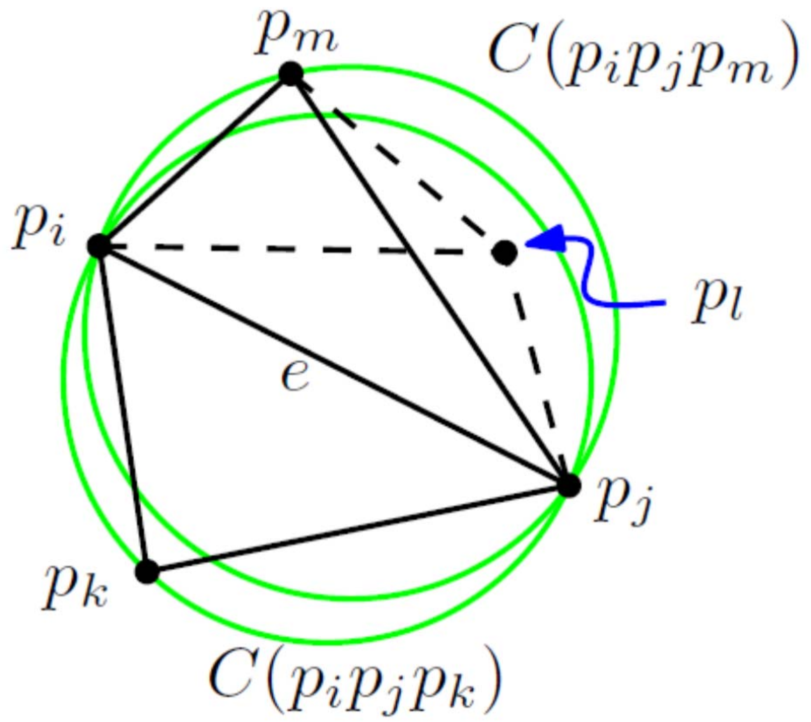
Empty Circle Property

Theorem: Let P be a set of points in the plane, and let T be a triangulation of P. Then T is a Delaunay triangulation of P if and only if the circumcircle of any triangle of T does not contain a point of P in its interior.



Delaunay Triangulations and Legal Triangulations

Theorem: Let P be a set of points in the plane. A triangulation T of P is legal if and only if T is a Delaunay triangulation.



Computing Delaunay Triangulations

- There are several ways to compute the Delaunay triangulation:
  - By iterative flipping from any triangulation
  - By plane sweep
  - By randomized incremental construction
  - By conversion from the Voronoi diagram
- The last three run in O(nlogn) time [expected] for n points in the plane

Incremental Construction

- L. J. Guibas, D. E. Knuth, and M. Sharir, Randomized incremental construction of Delaunay and Voronoi diagrams, Algorithmica,7, 1992, 381–413.
- Notes by D. Mount

- Insert sites in random order and update the triangulation with each addition
- After each insertion the expected number of structural changes in the diagram is O(1)
- The challenge is keeping track of where newly inserted sites are to be placed in the diagram
- Simple solution: put each of the uninserted points into a bucket according to the triangle that contains it in the current triangulation
- Claim that the expected number of times that a site is re-bucketed is O(log n)

In Circle Test

- Assume that no four sites are co-circular
- In circle test is equivalent to a determinant computation
  - Assume that abcd define a counterclockwise convex polygon (abc is the original triangle)
  - If not, d lies inside triangle and the test fails
  - d lies in circumcircle if and only if the following determinant is positive (if it is 0, the points are co-circular)

$$\text{inCircle}(a, b, c, d) = \det \begin{pmatrix} a_x & a_y & a_x^2 + a_y^2 & 1 \\ b_x & b_y & b_x^2 + b_y^2 & 1 \\ c_x & c_y & c_x^2 + c_y^2 & 1 \\ d_x & d_y & d_x^2 + d_y^2 & 1 \end{pmatrix} > 0.$$

Incremental Update

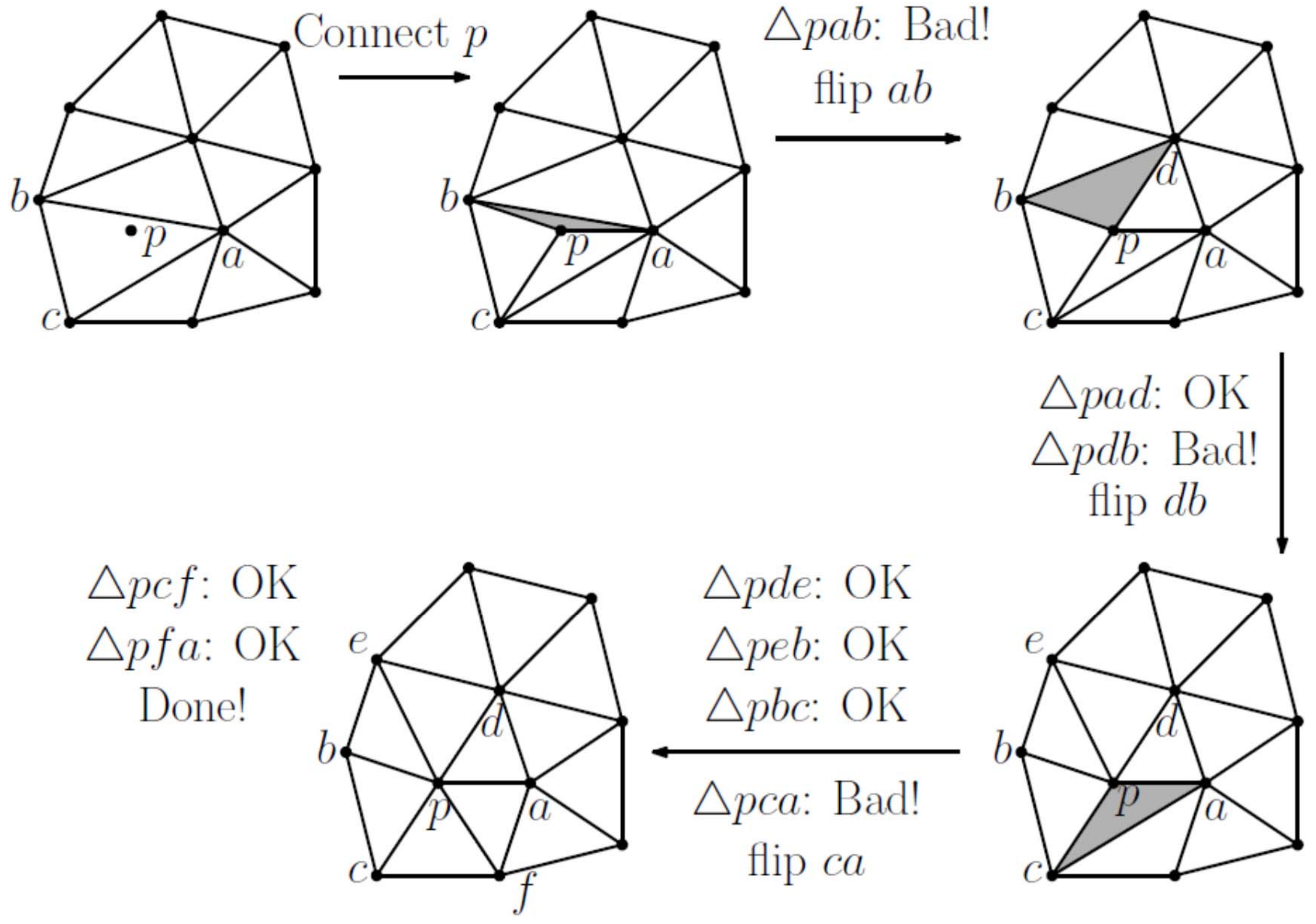
- Create a non-Delaunay triangulation and fix it
- Join new point with the vertices of the triangle that contains it
- Flip edges as needed
- Both can be done in O(1)

- Initialize by enclosing all points in a very large triangle (its vertices must lie outside of all circumcircles of final triangulation)

Incremental Update

- For each new point p, we have created three new triangles
- For each of the triangles that have been added, we check the vertex of the triangle that lies on the opposite side of the edge that does not include p
- If this vertex fails the in circle test, then we swap the edge creating two new triangles that are adjacent to p
- We repeat the same test with these triangles

Incremental Update Example



Details

- This is only a sketch of algorithm
- We would need to prove that a triangulation that is locally Delaunay is also globally Delaunay
- Each time triangles are deleted and new triangles are made, uninserted points must be re-bucketed in O(1) time per point and each point is expected to be re-bucketed O(logn) times