Polygon Partitioning

O'Rourke, Chapter 2

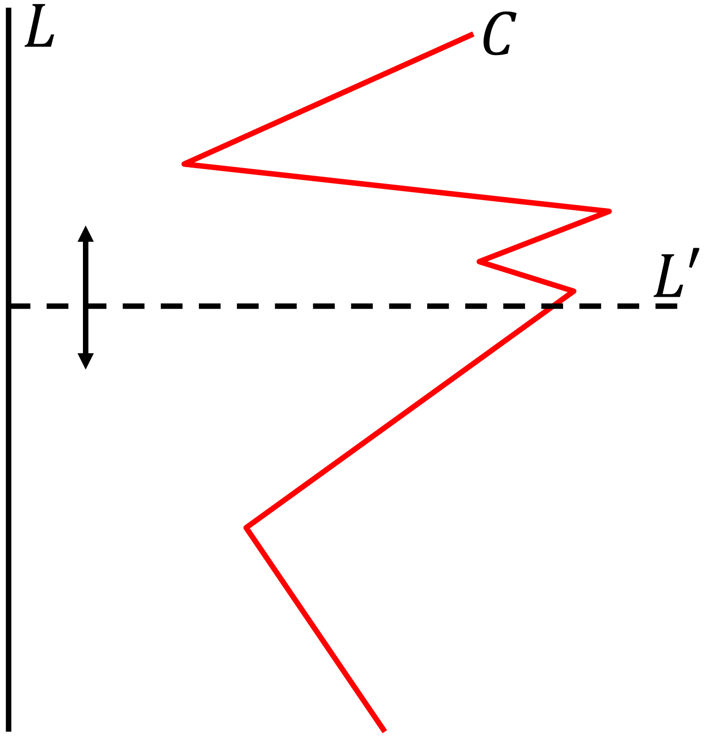
de Berg, Chapter 3

Announcements

- Assignment 1 posted
- TA office hours:
  - Thursday @ 4PM
  - Malone 239

Monotonicity

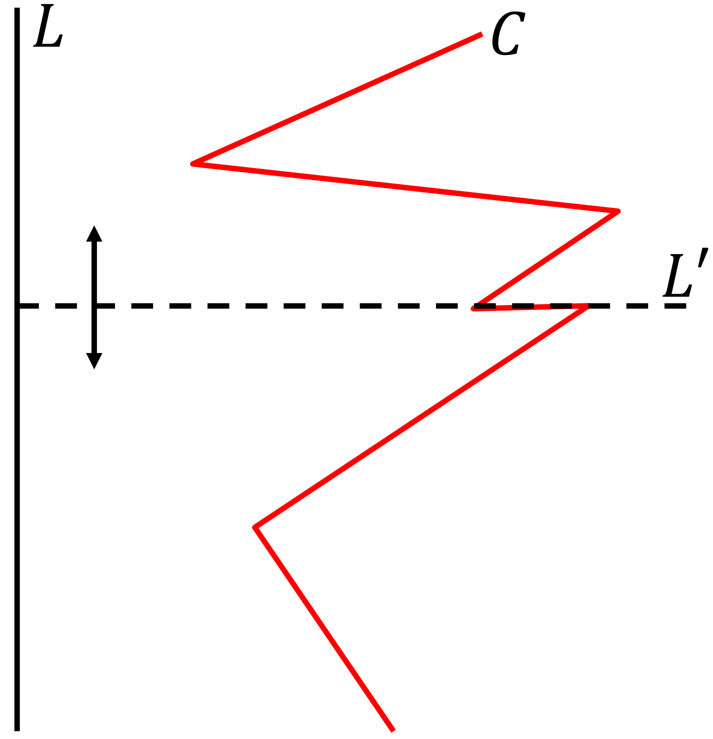
A polygonal chain C is *strictly monotone w.r.t. a line* L if every line L' perp. to L meets C at at most one point.



Monotonicity

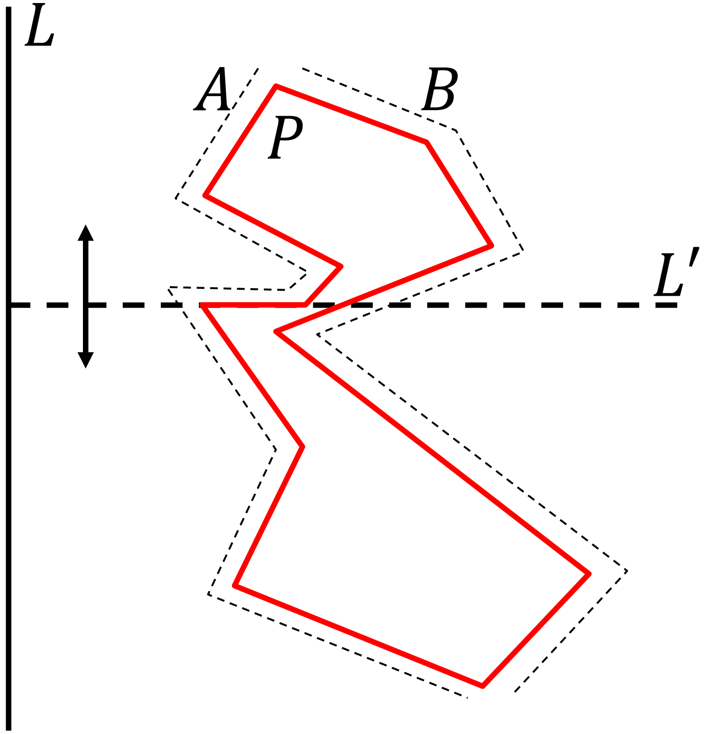
A polygonal chain C is *strictly monotone w.r.t. a line* L if every line L' perp. to L meets C at at most one point.

It is *monotone w.r.t. a line* L if every line L' perp. to L intersects C in at most one connected component.



Monotonicity

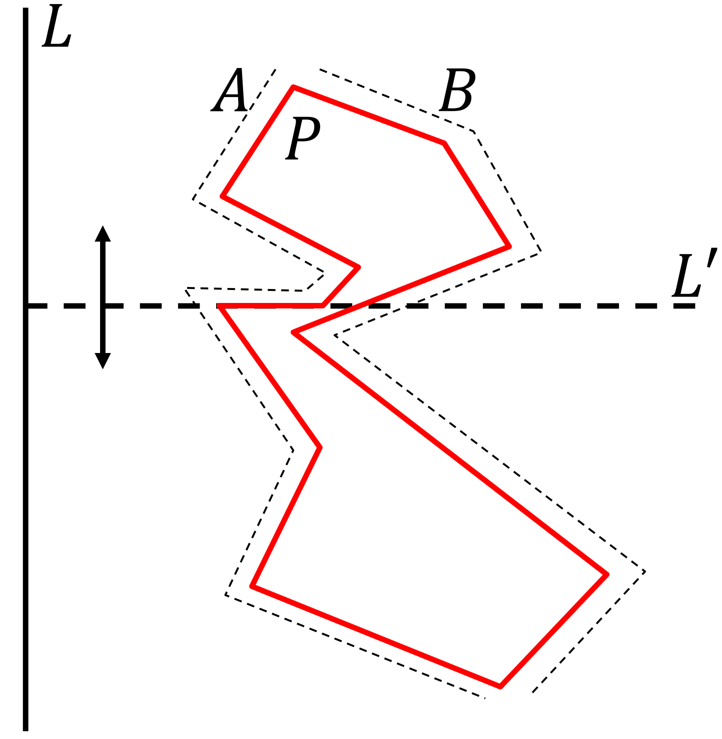
A polygonal P is *monotone w.r.t. a line* L if its boundary can be split into two polygon chains, A and B, such that each chain is monotonic w.r.t. L.



Monotonicity

A polygonal P is *monotone w.r.t. a line* L if its boundary can be split into two polygon chains, A and B, such that each chain is monotonic w.r.t. L.

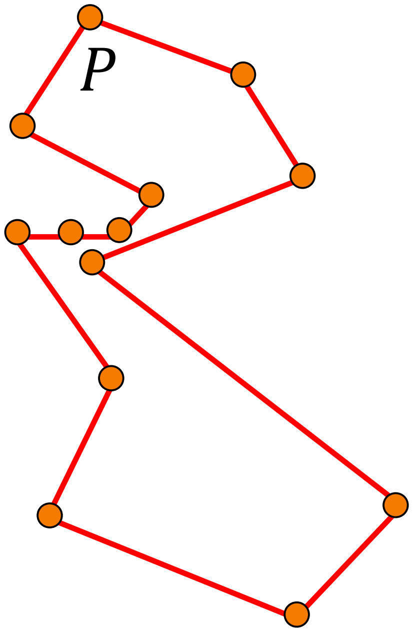
$\Leftrightarrow$ It is monotone w.r.t. L if the intersection of P with any line L' perp. to L has at most two connected components.



Note

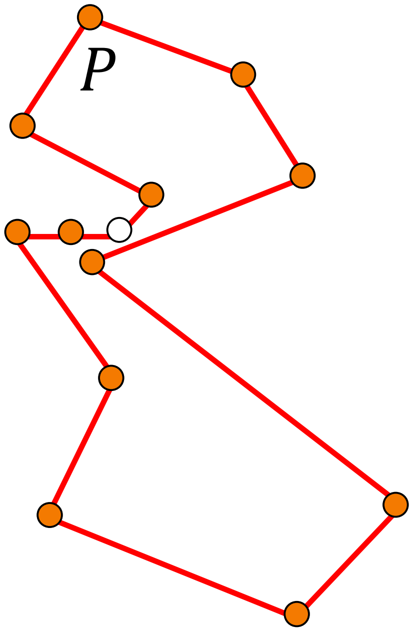
The vertices of a monotone polygon (w.r.t. the vertical axis) can be sorted by y-value in linear time.

- $O(n)$: Compute the highest vertex.
- $O(n)$: Merge the two (sorted) chains.



Interior Cusps

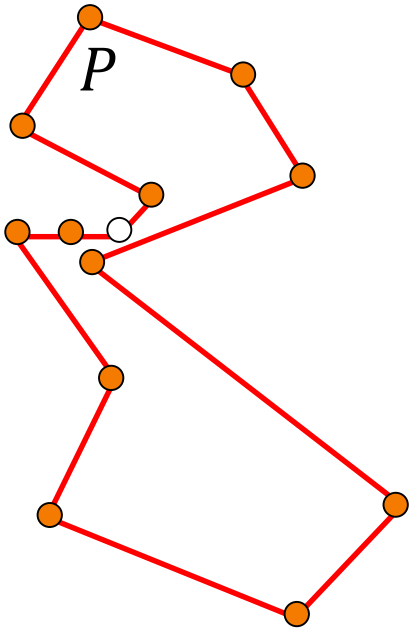
An *interior cusp* of a polygon P (w.r.t. the vertical axis) is a reflex* vertex $v \in P$ whose neighboring vertices are either at or above, or at or below v.



*Recall that reflex vertices have interior angle strictly greater than π.

Claim

*If P has no interior cusps (w.r.t. the vertical axis), it is monotone (w.r.t. the vertical axis).**

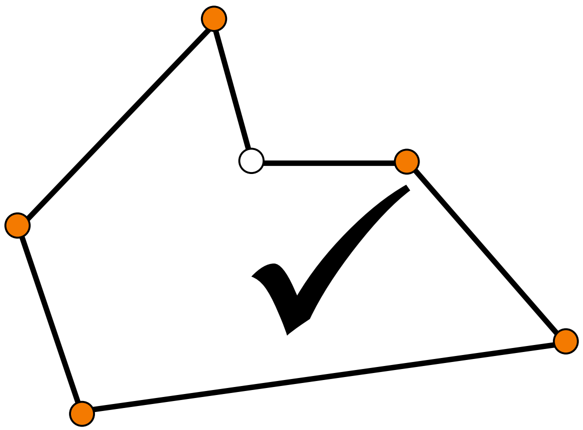


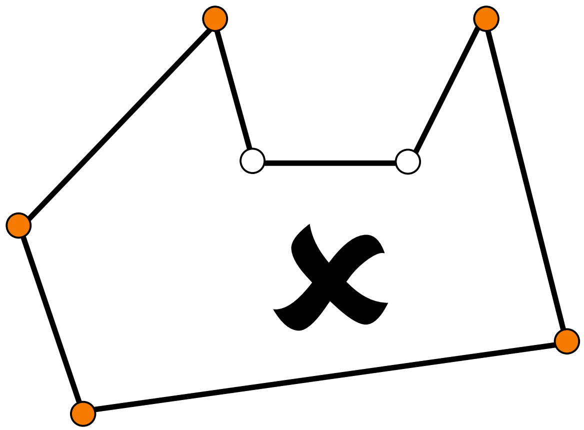
*Note that it can have interior cusps and still be monotone.

Claim

If P has no interior cusps (w.r.t. the vertical axis), it is monotone (w.r.t. the vertical axis).

<u>Note</u>: We cannot change the condition so that interior cusps have to be strictly above



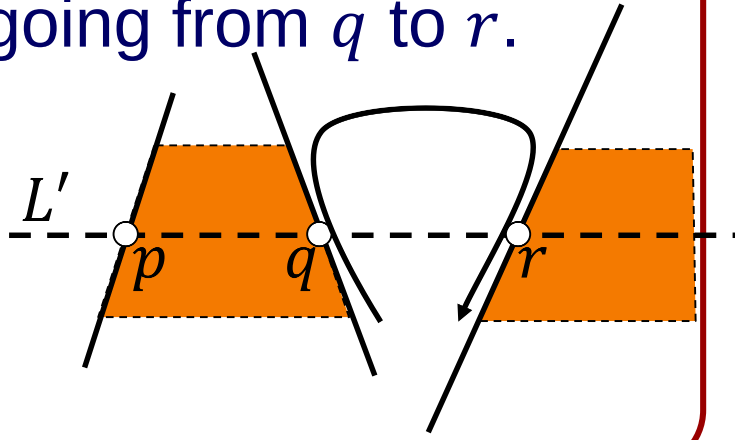


Proof

If it isn't monotone, there will be a line L' intersecting P in three or more points, p, q, and r. (Assume these are the first three.)

WLOG, assume the polygon interior is to the left of q (and right of p and r):

- If the order of the vertices in the polygon is pqr we hit an interior cusp at the top going from q to r.

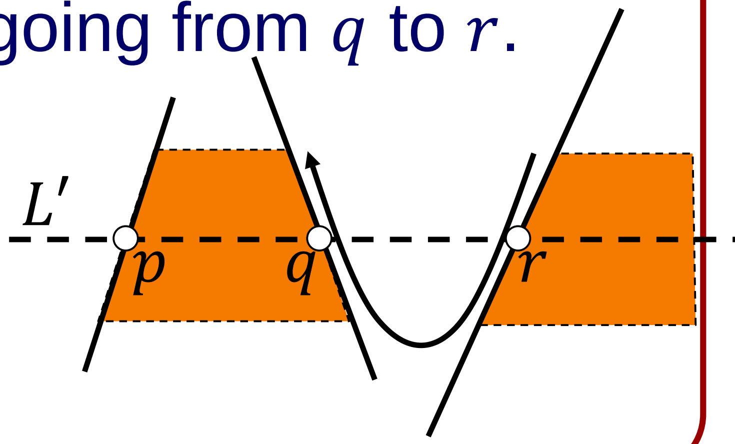


Proof

If it isn't monotone, there will be a line L' intersecting P in three or more points, p, q, and r. (Assume these are the first three.)

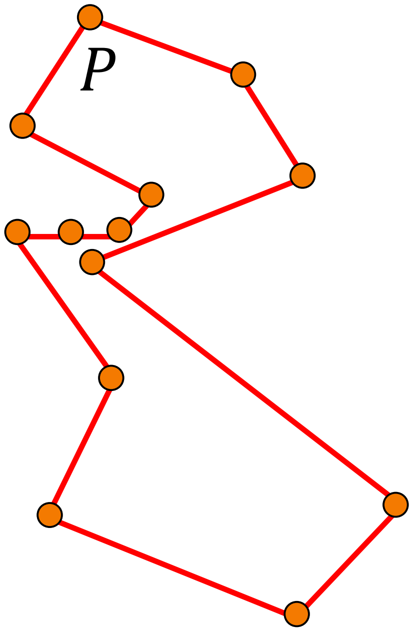
WLOG, assume the polygon interior is to the left of q (and right of p and r):

- If the order of the vertices in the polygon is pqr we hit an interior cusp at the top going from q to r.
- Otherwise, we hit an interior cusp at the bottom going from r to q.



Claim

A monotone polygon can be triangulated in linear time.

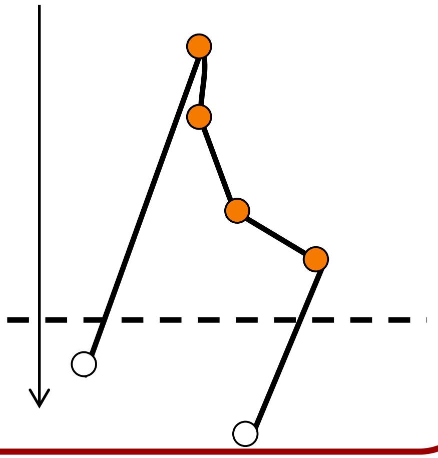


Outline

Invariant

When triangulating from the top vertex, at any y-value, the un-triangulated vertices above y can be broken up into two chains:

- One contains a single vertex
- The other has only reflex vertices.

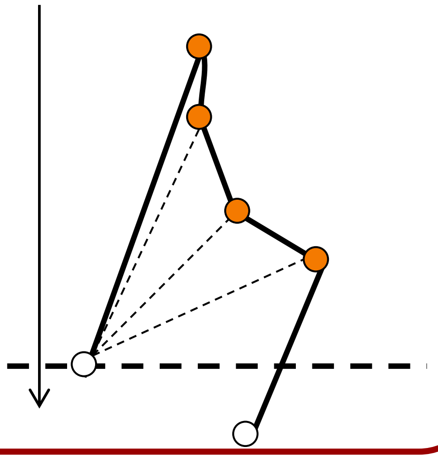


Outline

When we hit the next vertex it can be:

- On the side with one vertex
  - » Connect the vertex to all vertices on the other side and pop off the triangles.

The invariant is preserved!

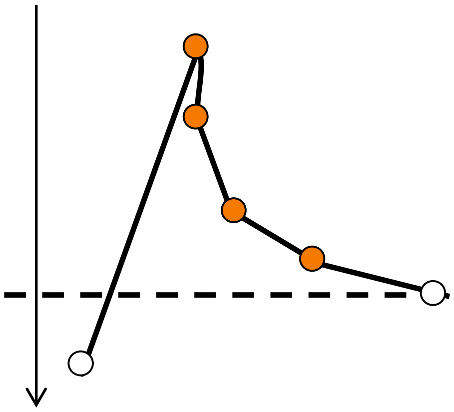


Outline

When you hit the next vertex it can be:

- On the side with reflex vertices
  - » Either the new vertex makes the previous one reflex
    - – Do nothing

The invariant is preserved!



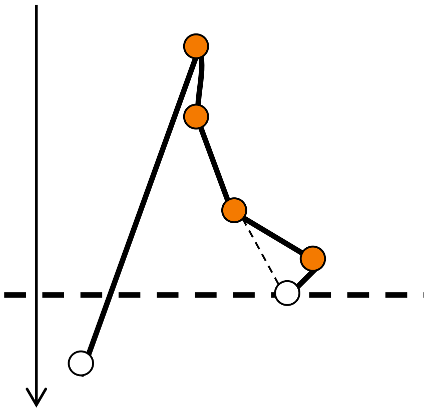
Outline

When you hit the next vertex it can be:

- On the side with reflex vertices
  - » Either the new vertex makes the previous one reflex
    - – Do nothing
  - » Or it doesn't
    - – Recursively connect and pop

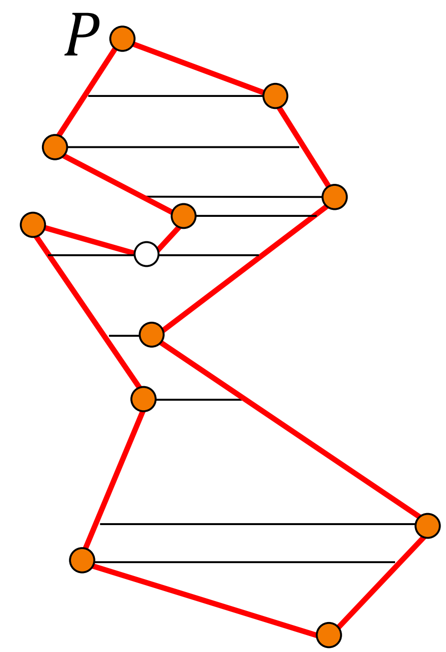
When we can't connect back anymore, we have a new reflex vertex.

The invariant is preserved!



Trapezoidalization

A *horizontal trapezoidalization* is obtained by drawing a horizontal line through every vertex of the polygon.*

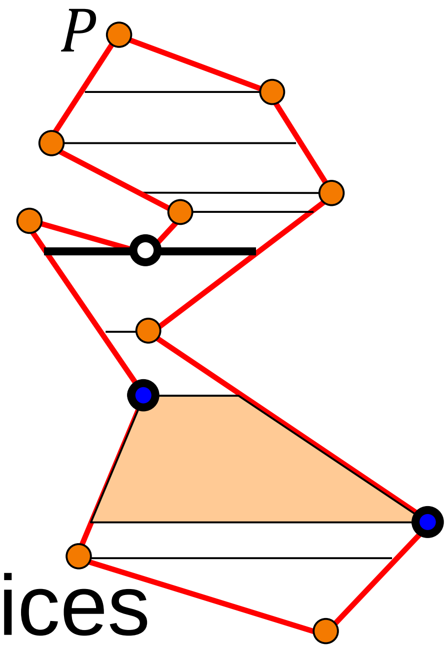


*Assuming distinct vertices have different y-values.

Trapezoidalization

A *horizontal trapezoidalization* is obtained by drawing a horizontal line through every vertex of the polygon.

The *supporting vertices* of a trapezoid are the two vertices of P defining the horizontals of the trapezoid.

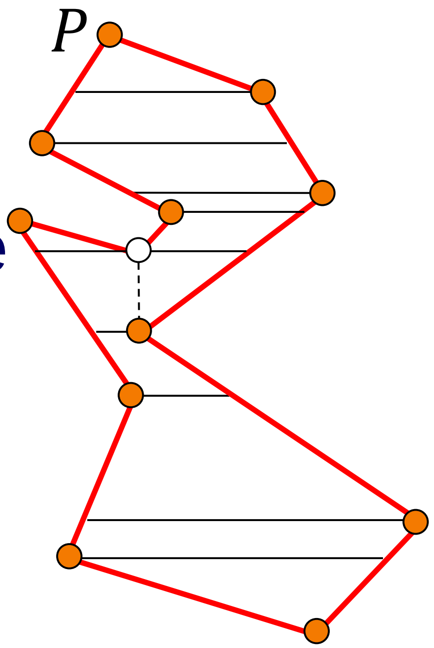


Note:
Interior (vertical) cusps are vertices that are internal to their horizontals.

Trapezoids → Monotone Polygons

Given a trapezoidalization of P, we can obtain a partition into monotone (w.r.t. the vertical axis) polygons:

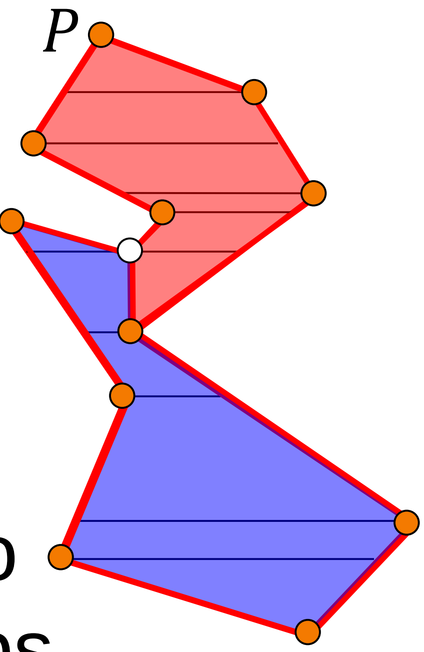
- For upward cusps, connect the supporting vertices on the trapezoid below the cusp
- For downward cusps, connect the supporting vertices on the trapezoid above the cusp.



Trapezoids → Monotone Polygons

Given a trapezoidalization of P, we can obtain a partition into monotone (w.r.t. the vertical axis) polygons:

- For upward cusps, connect the supporting vertices on the trapezoid below the cusp
- For downward cusps, connect the supporting vertices on the trapezoid above the cusp.



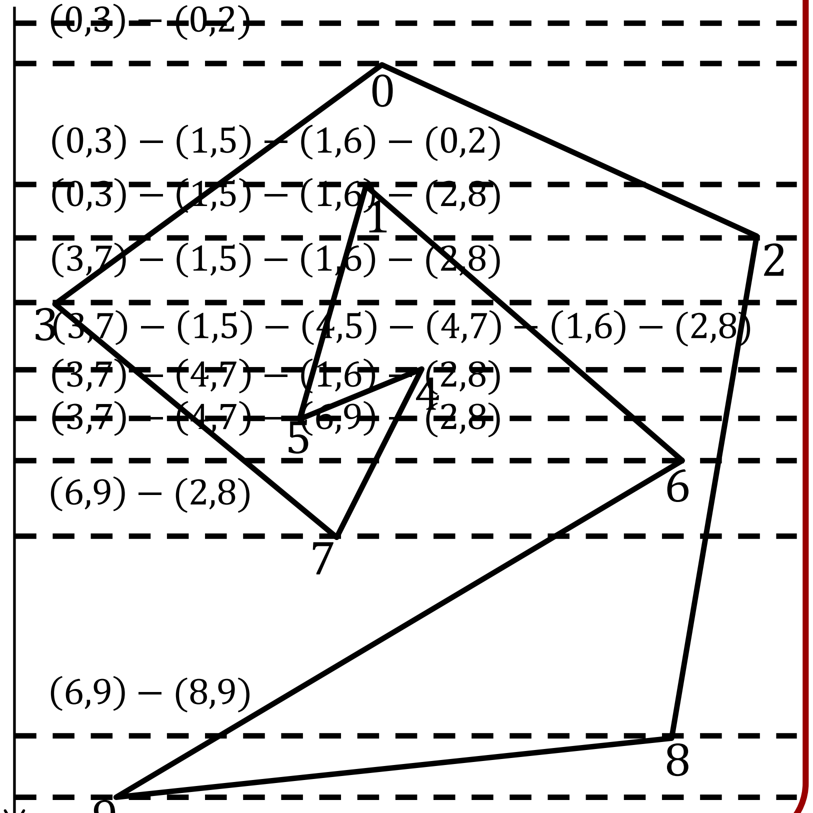
This decomposes the polygon into sub-polygons without interior cusps.
⇒ Each sub-polygon is monotone.

Line/Plane Sweep

Given a polygon P, sweep a horizontal line downwards maintaining a sorted "active edge" list – those edges that are intersected by the current horizontal.

<u>Note</u>:

The list of active edges can only change when the horizontal passes through a vertex.

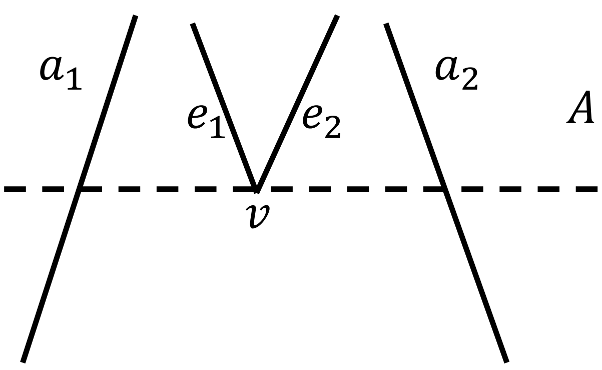


Algorithm

- PlaneSweep(V , $E \subset V \times V$):
  - SortByLargestToSmallestHeight(V)
  - $A \leftarrow \emptyset$
  - For each $v \in V$
    - $(e_1, e_2) \leftarrow$ EndPoints(v)
    - If(Before(v , e_1): Remove(A , e_1)
    - Else: Insert(A , e_1)
    - If(Before(v , e_2): Remove(A , e_2)
    - Else: Insert(A , e_2)

Algorithm

- PlaneSweep(V , $E \subset V \times V$):
  - SortByLargestToSmallestHeight(V)
  - $A \leftarrow \emptyset$
  - For each $v \in V$
    - $(e_1, e_2) \leftarrow$ EndPoints(v)
    - If(Before(v , e_1): Remove(A , e_1)
    - Else: Insert(A , e_1)
    - If(Before(v , e_2): Remove(A , e_2)
    - Else: Insert(A , e_2)



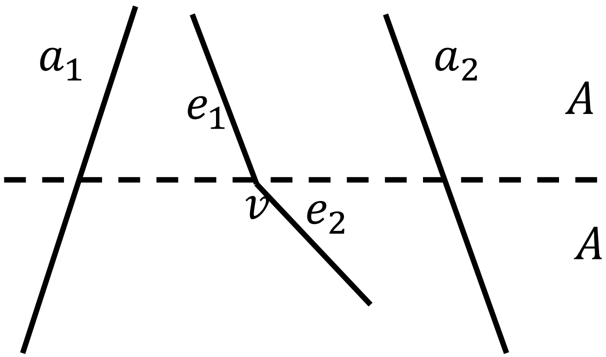
$$A = \cdots a_1 - e_1 - e_2 - a_2 \ldots$$

$$\Downarrow$$

$$A = \cdots a_1 - a_2 \ldots$$

Algorithm

- PlaneSweep(V , $E \subset V \times V$):
  - SortByLargestToSmallestHeight(V)
  - $A \leftarrow \emptyset$
  - For each $v \in V$
    - $(e_1, e_2) \leftarrow$ EndPoints(v)
    - If(Before(v , e_1): Remove(A , e_1)
    - Else: Insert(A , e_1)
    - If(Before(v , e_2): Remove(A , e_2)
    - Else: Insert(A , e_2)



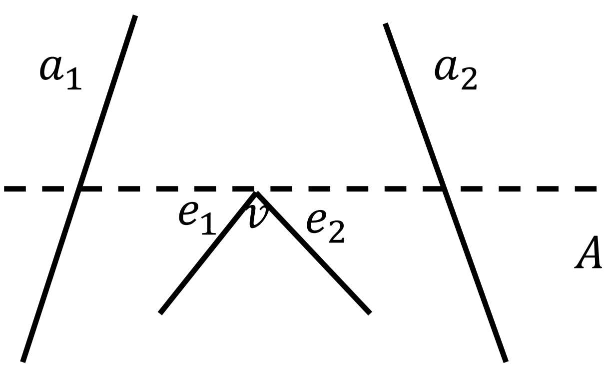
$$A = \cdots a_1 - e_1 - a_2 \ldots$$

$$\Downarrow$$

$$A = \cdots a_1 - e_2 - a_2 \ldots$$

Algorithm

- PlaneSweep(V , $E \subset V \times V$):
  - SortByLargestToSmallestHeight(V)
  - $A \leftarrow \emptyset$
  - For each $v \in V$
    - $(e_1, e_2) \leftarrow$ EndPoints(v)
    - If(Before(v , e_1): Remove(A , e_1)
    - Else: Insert(A , e_1)
    - If(Before(v , e_2): Remove(A , e_2)
    - Else: Insert(A , e_2)



$$A = \cdots a_1 - a_2 \ldots$$

$$A = \cdots a_1 - e_1 - e_2 - a_2 \ldots$$

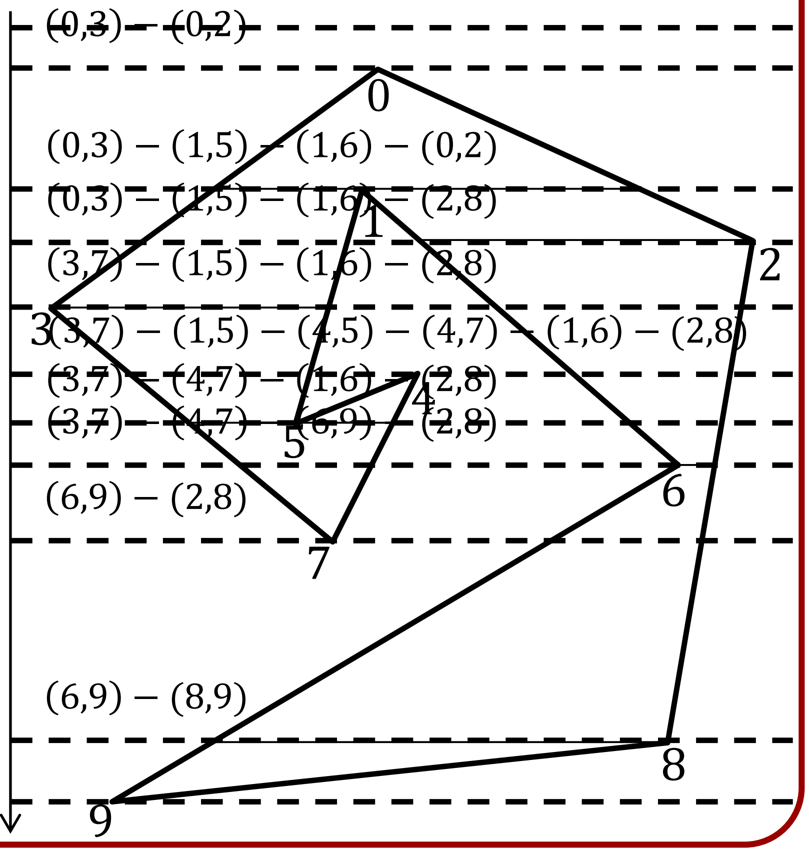
Algorithm

- PlaneSweep(V , $E \subset V \times V$):
  - SortByLargestToSmallestHeight(V) $\boxed{O(n \log n)}$
  - $A \leftarrow \emptyset$
  - For each $v \in V$ $\boxed{O(n)}$
    - $(e_1, e_2) \leftarrow$ EndPoints(v)
    - If(Before(v , e_1): Remove(A , e_1)
    - Else: Insert(A , e_1)
    - If(Before(v , e_2): Remove(A , e_2)
    - Else: Insert(A , e_2)

$O(\log n)$
w/ balanced tree
(e.g. std::map)

Constructing a Trapezoidalization

A trapezoidal partition can be computed in $O(n \log n)$ time by performing a line-sweep and adding (part of) the horizontal to the left and right neighbors as we hit new vertices.



Constructing a Trapezoidalization

Note:

We had assumed that the vertices have different y-coordinates.

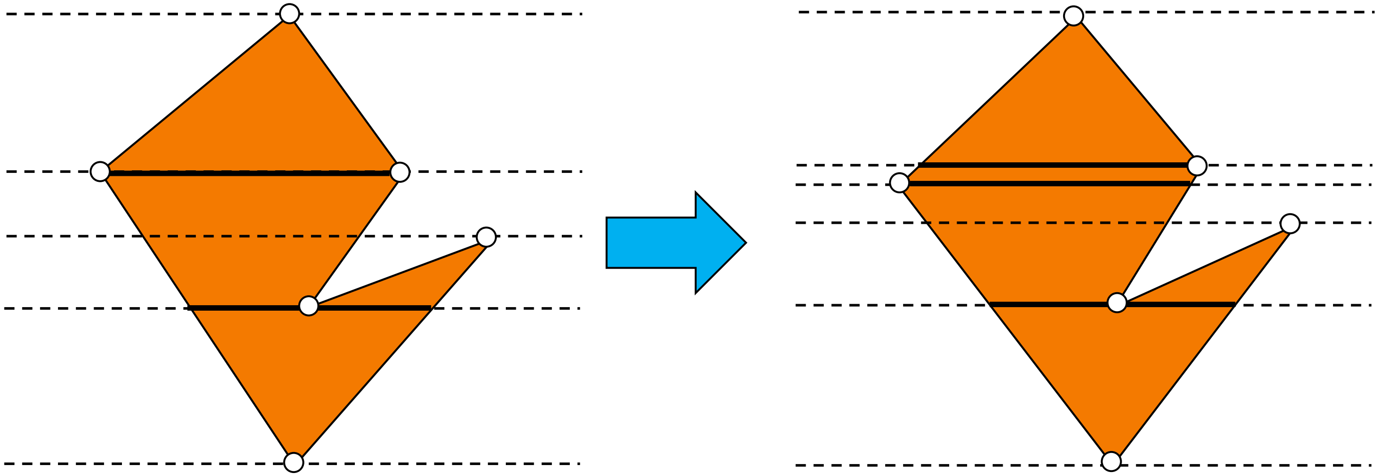
This isn't actually necessary. It suffices to sort lexicographically. (If two vertices have the same y-coordinates then the one with larger x-coordinate is first.)

Constructing a Trapezoidalization

Note:

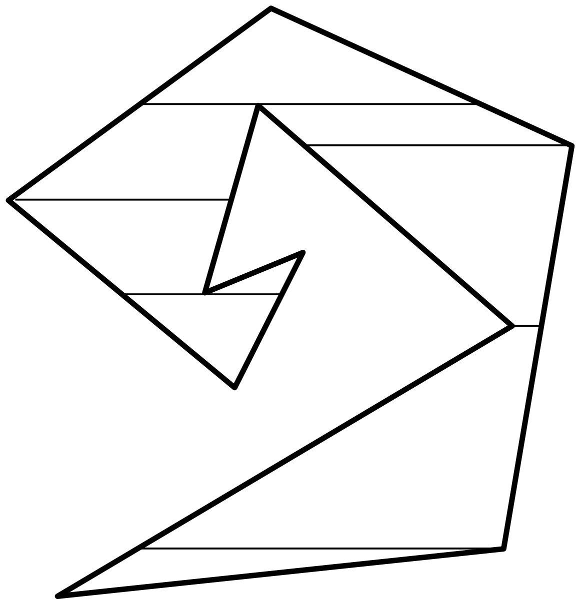
We had assumed that the vertices have different y-coordinates.

Conceptually, this amounts to applying a tiny rotation in the CCW direction.



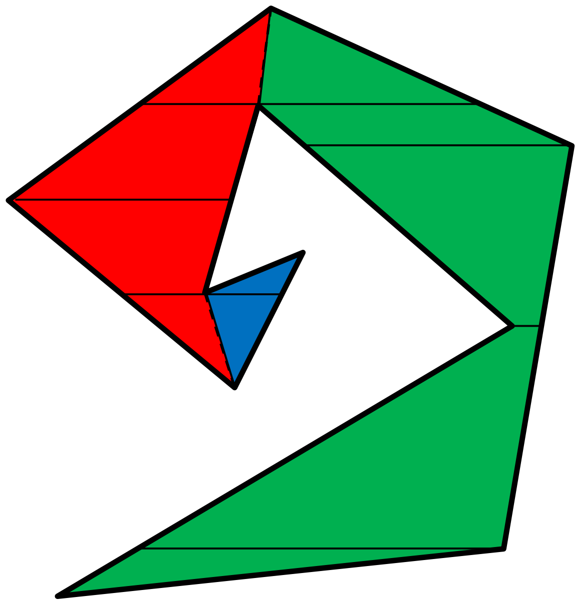
Triangulation

- Triangulate(P):
  - Construct a trapezoidalization



Triangulation

- Triangulate(P):
  - Construct a trapezoidalization
  - Partition into monotone polygons



Triangulation

- Triangulate(P):
  - Construct a trapezoidalization
  - Partition into monotone polygons
  - Triangulate the monotone polygons