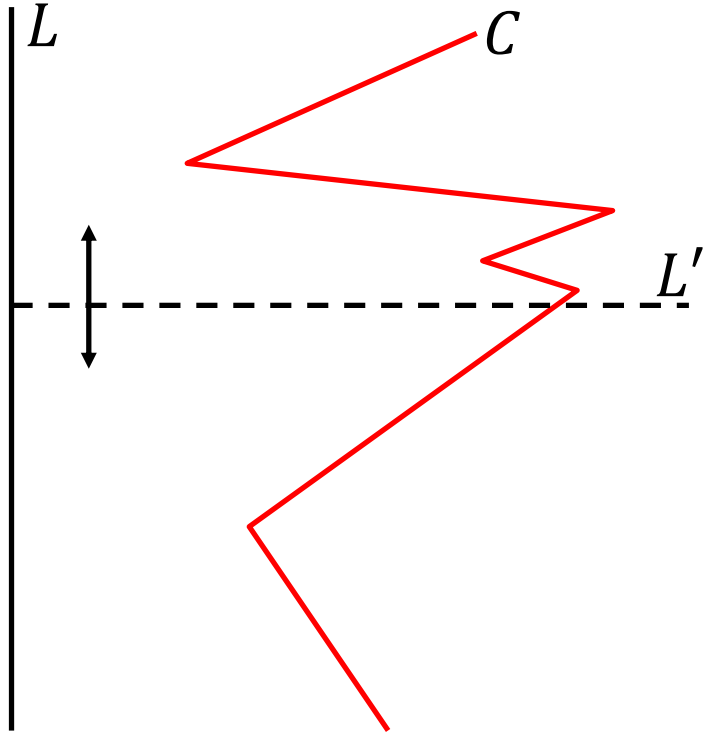
Polygon Partitioning

O'Rourke, Chapter 2

de Berg, Chapter 3

Monotonicity

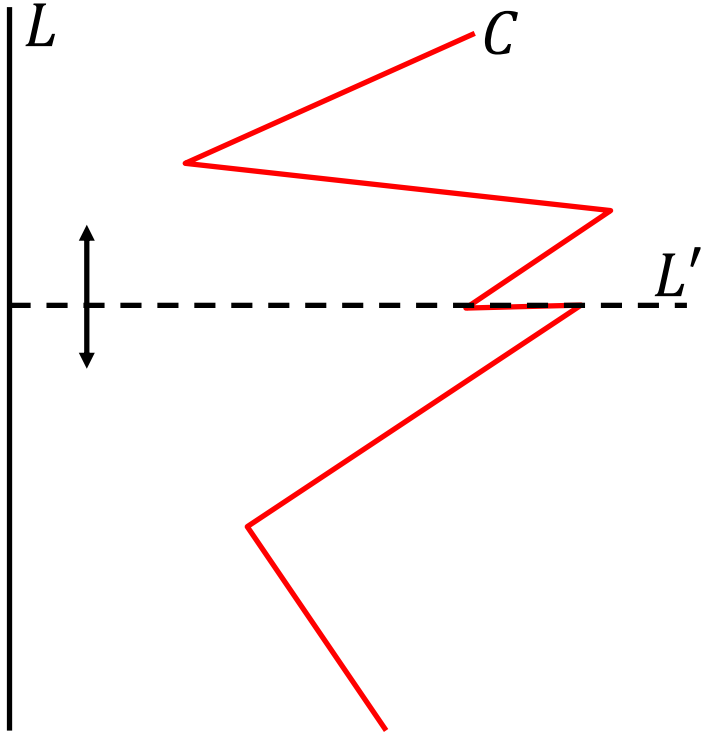
A polygonal chain C is *strictly monotone w.r.t. a line* L if every line L' perp. to L meets C at at most one point.



Monotonicity

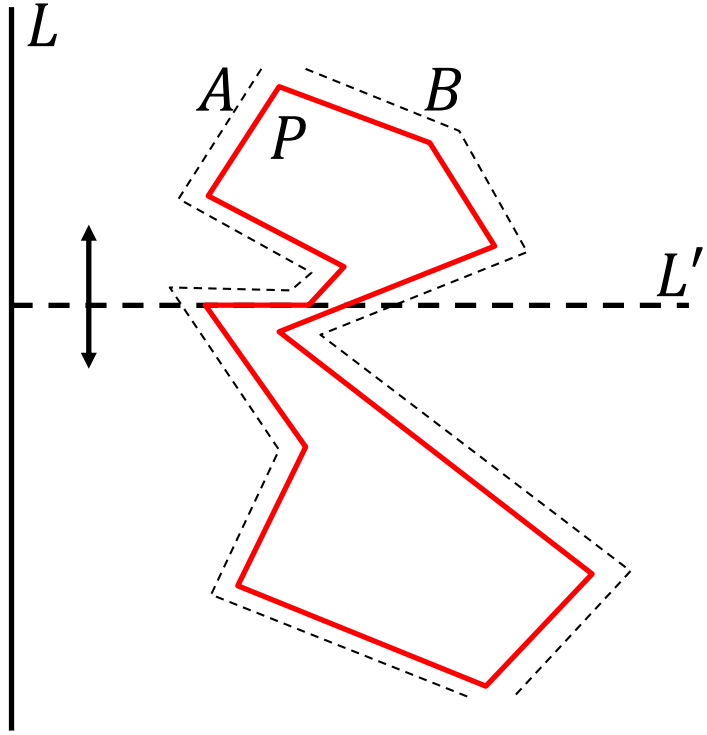
A polygonal chain C is *strictly monotone w.r.t. a line* L if every line L' perp. to L meets C at at most one point.

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

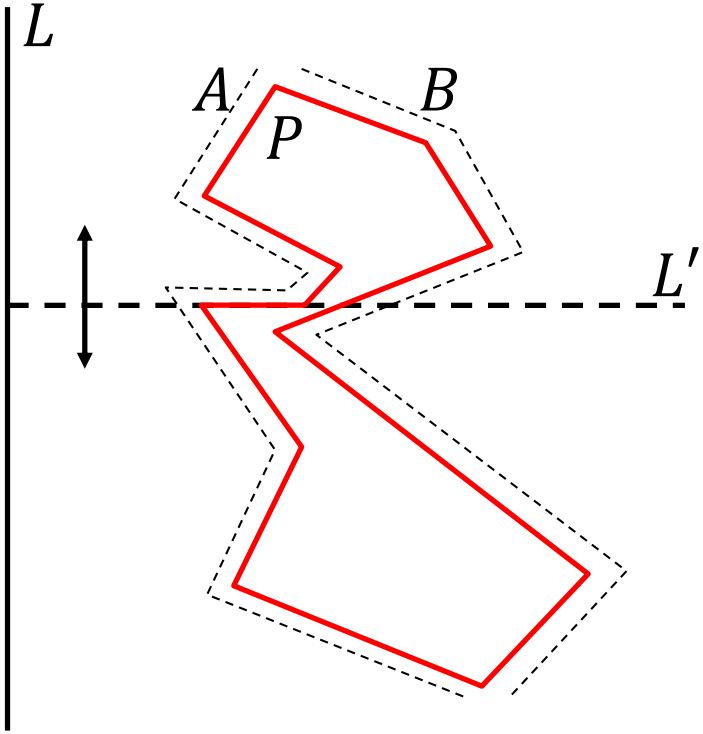
A polygonal P is *monotone w.r.t. a line* L if its boundary can be split into two polygon chains, A and B, such that each chain is monotonic w.r.t. L.



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.

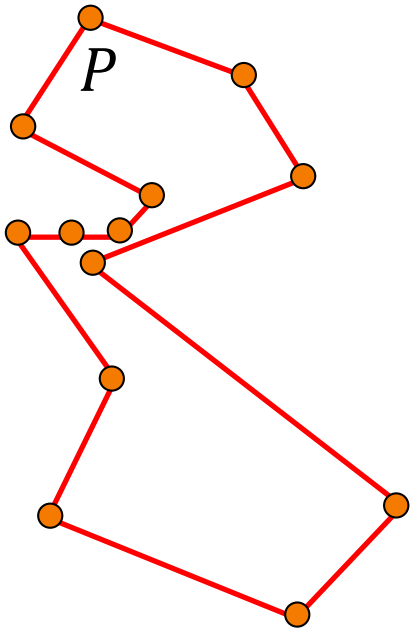
$\Leftrightarrow$ It is monotone w.r.t. L if the intersection of P with any line L' perp. to L has at most two connected components.



Note

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

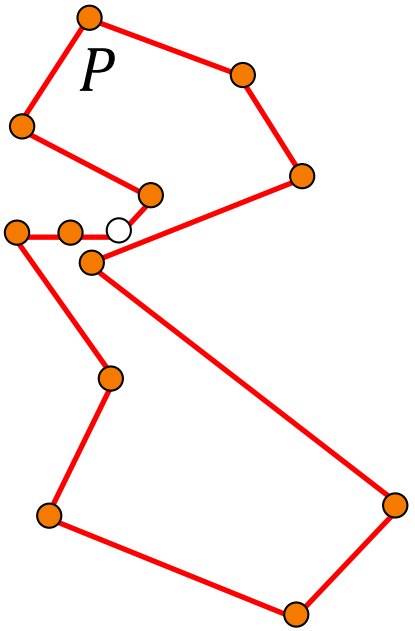
- $O(n)$: Compute the highest vertex.
- $O(n)$: Merge the two (sorted) chains.



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.

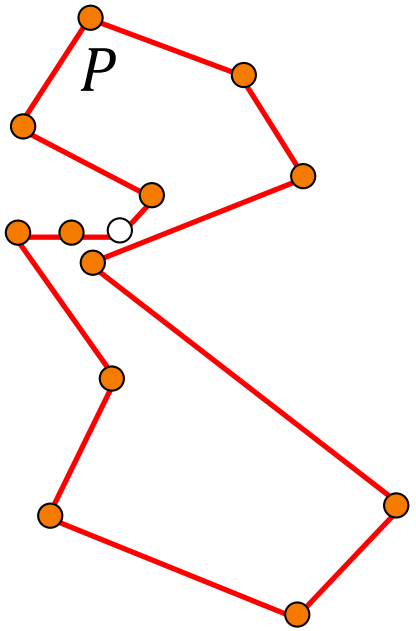
It is a *downward* (resp. *upward*) cusp if it is above (resp. below) the horizontal.



*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).[*]

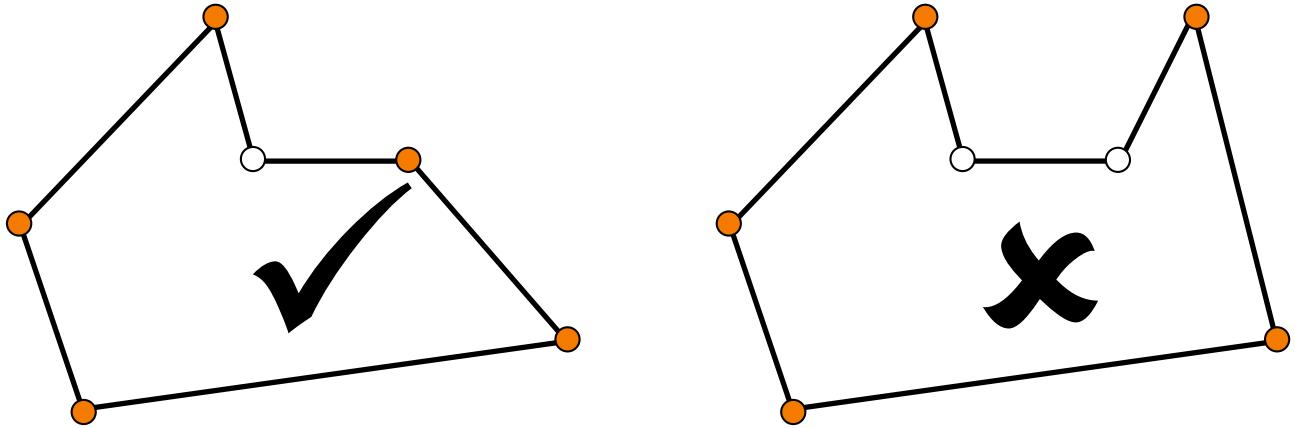


[*]Note that it can have interior cusps and still be monotone.

Claim

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

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

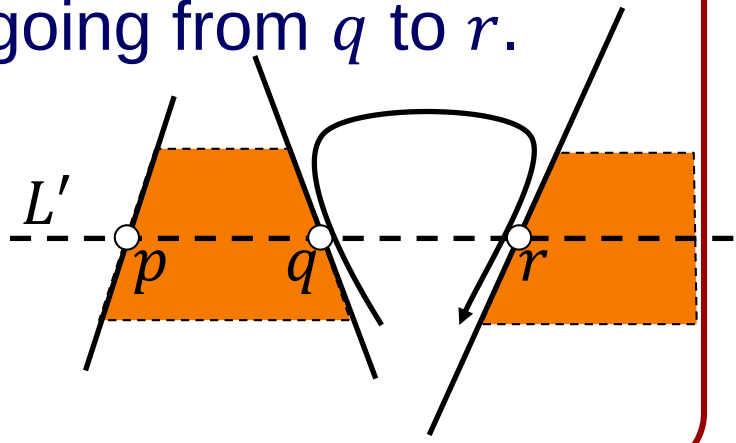


Proof

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.)

The polygon interior is right of p, left of q, and right of 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.



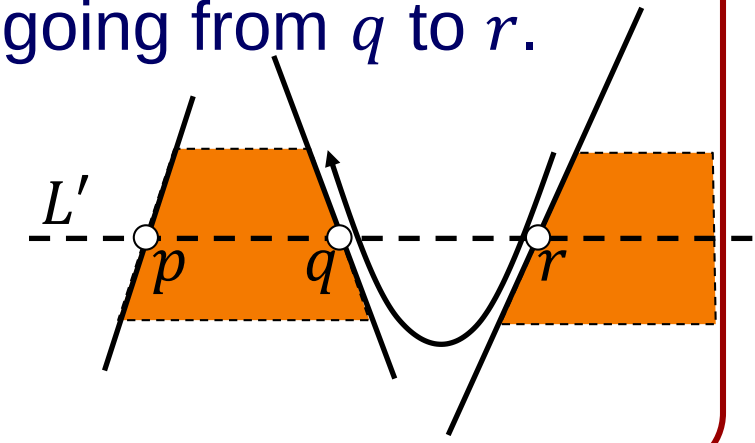
*More precisely, three or more connected components.

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.)

The polygon interior is right of p, left of q, and right of r.

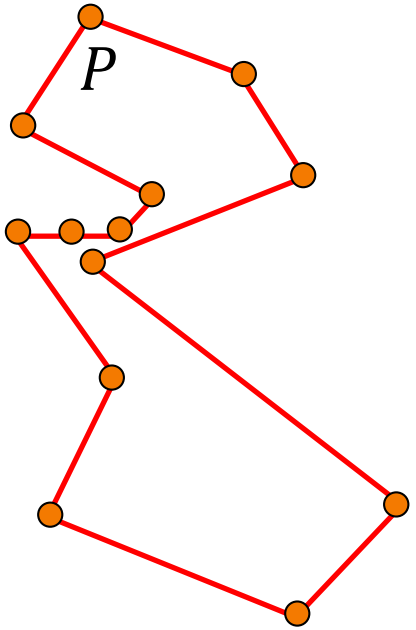
- If the order of the vertices in the polygon is pqr we hit an interior cusp at the top going from q to r.
- Otherwise, we hit an interior cusp at the bottom going from r to q.



*More precisely, three or more connected components.

Claim

A monotone polygon can be triangulated in linear time.

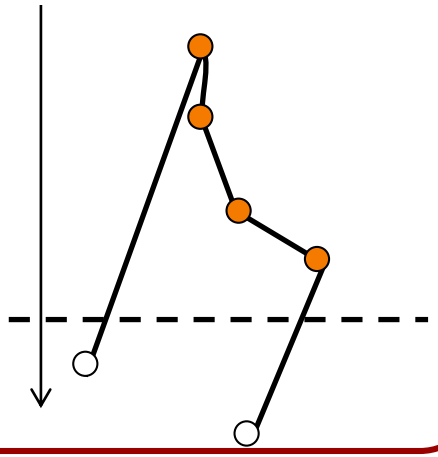


Outline

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.

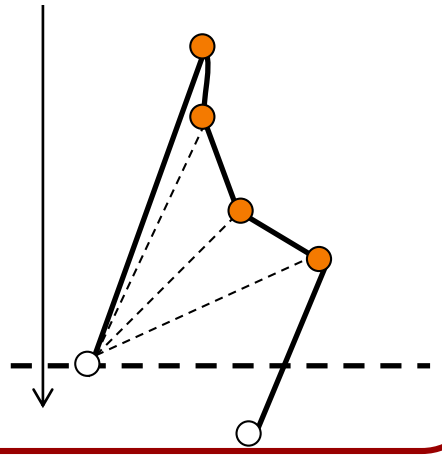


Outline

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!

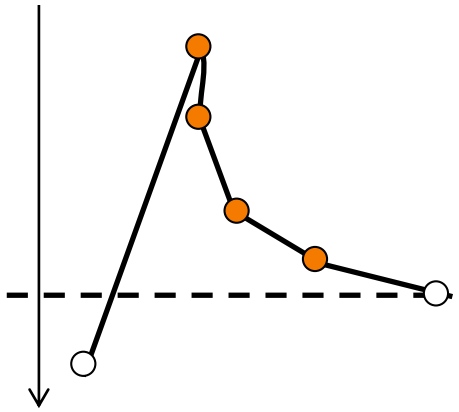


Outline

When we 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!



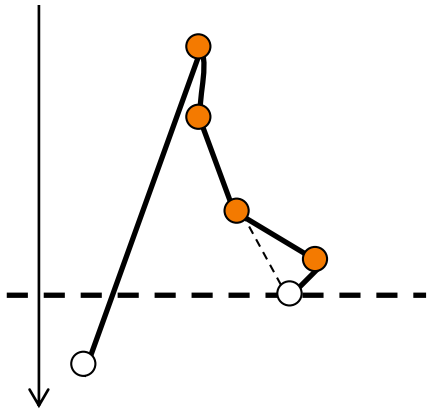
Outline

When we 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

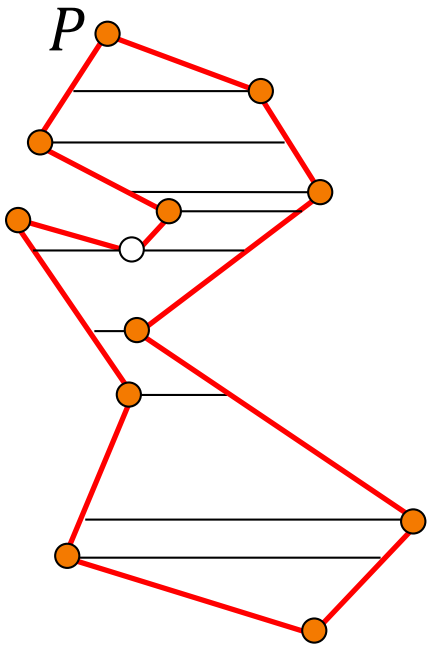
When we can't connect back anymore, we have a new reflex vertex.

The invariant is preserved!



Trapezoidalization

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

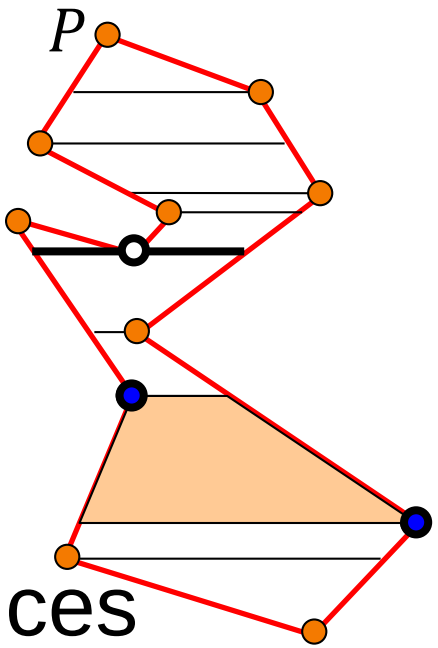


*Assuming distinct vertices have different y-values.

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.

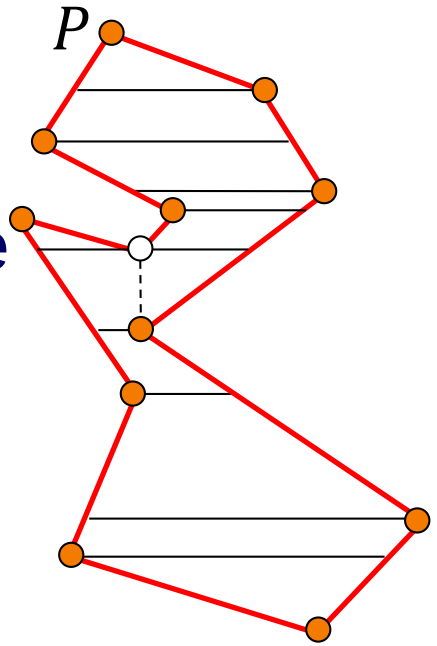


Note:
Interior (vertical) cusps are vertices that are internal to their horizontals.

Trapezoids → Monotone Polygons

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

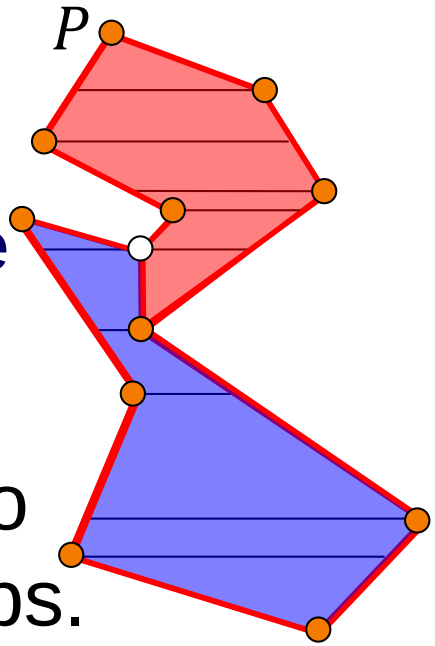
- For upward cusps, connect the supporting vertices on the trapezoid below the cusp
- For downward cusps, connect the supporting vertices on the trapezoid above the cusp.



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.

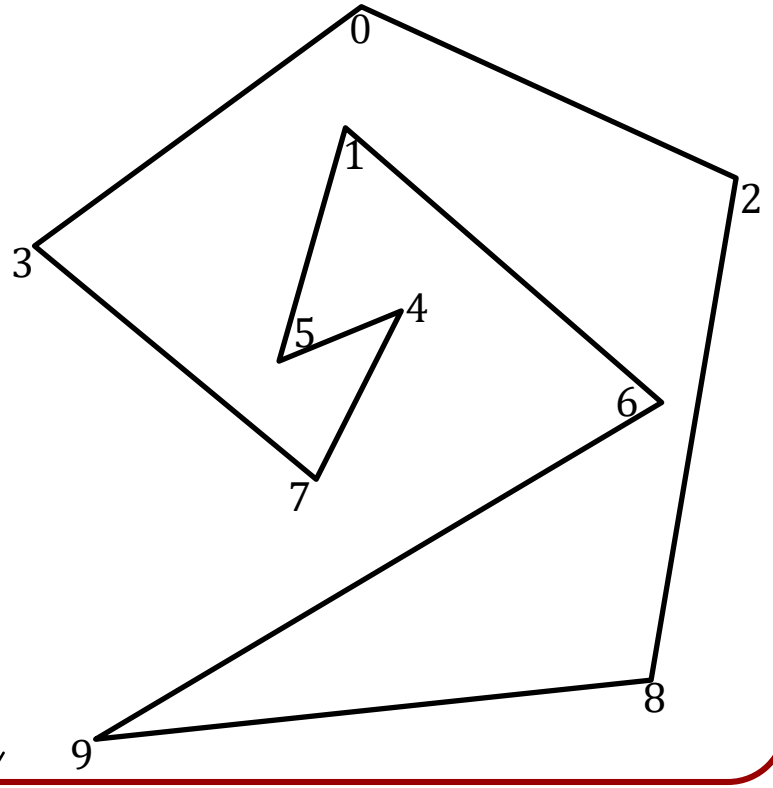
⇒ Each sub-polygon is monotone.

Line/Plane Sweep

Given a polygon P, sweep a horizontal line downwards maintaining a sorted "active edge" list – the edges intersected just below the current horizontal.

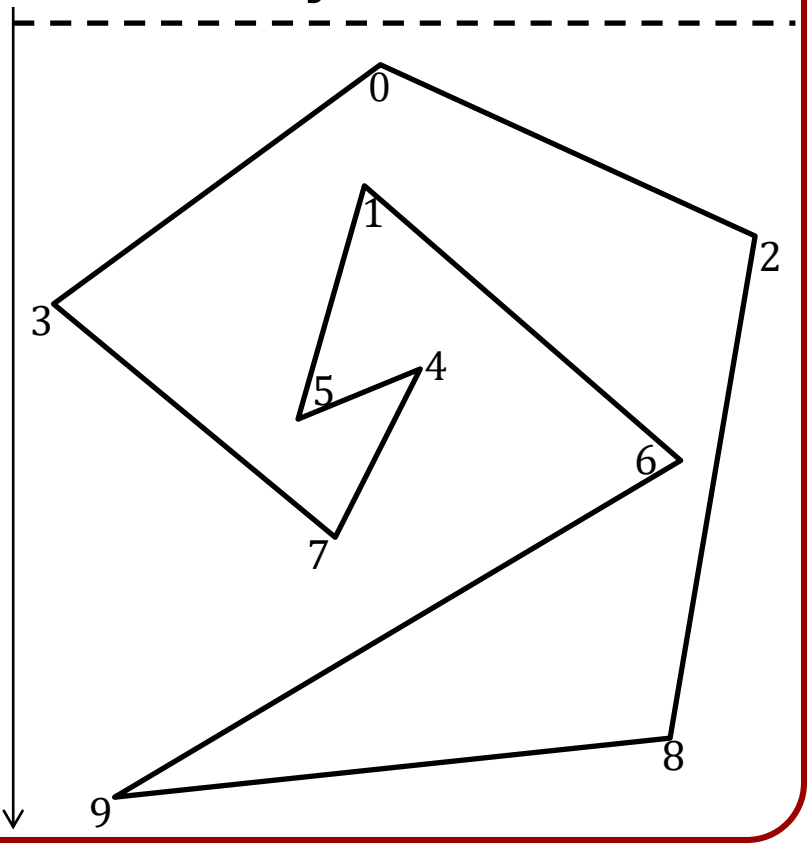
Note:

- The active edge list only changes when the horizontal goes through a vertex.
- The segment between the 1st and 2nd active edge, the 3rd and 4th active edge, etc. is interior.



Line/Plane Sweep

Given a polygon P, sweep a horizontal line downwards maintaining a sorted "active edge" list – the edges intersected just below the current horizontal.

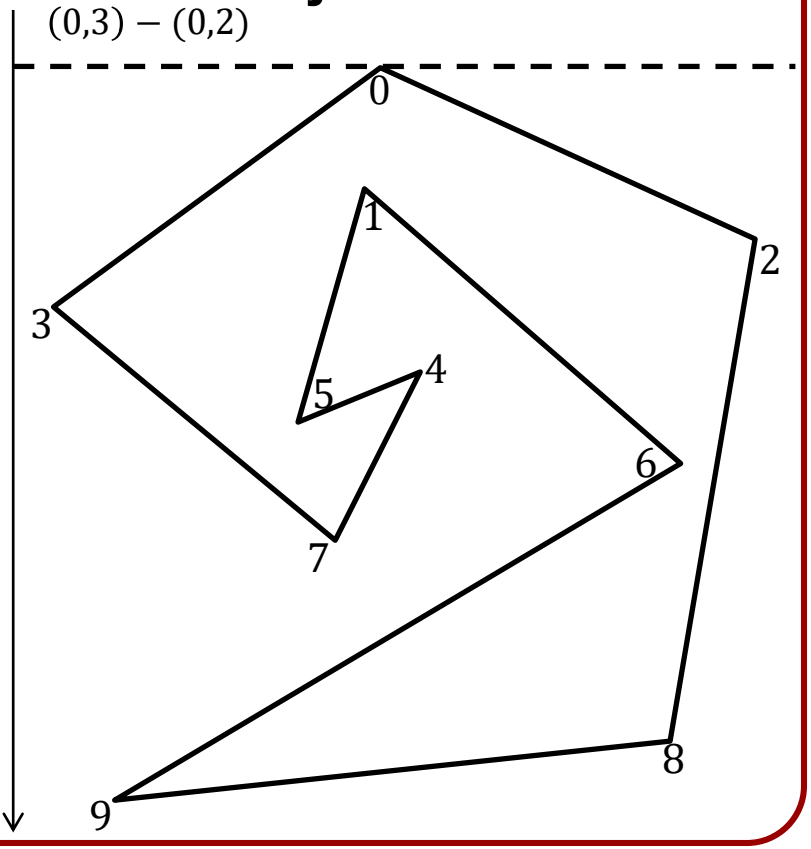


Note:

- The active edge list only changes when the horizontal goes through a vertex.
- The segment between the 1st and 2nd active edge, the 3rd and 4th active edge, etc. is interior.

Line/Plane Sweep

Given a polygon P, sweep a horizontal line downwards maintaining a sorted "active edge" list – the edges intersected just below the current horizontal.



Note:

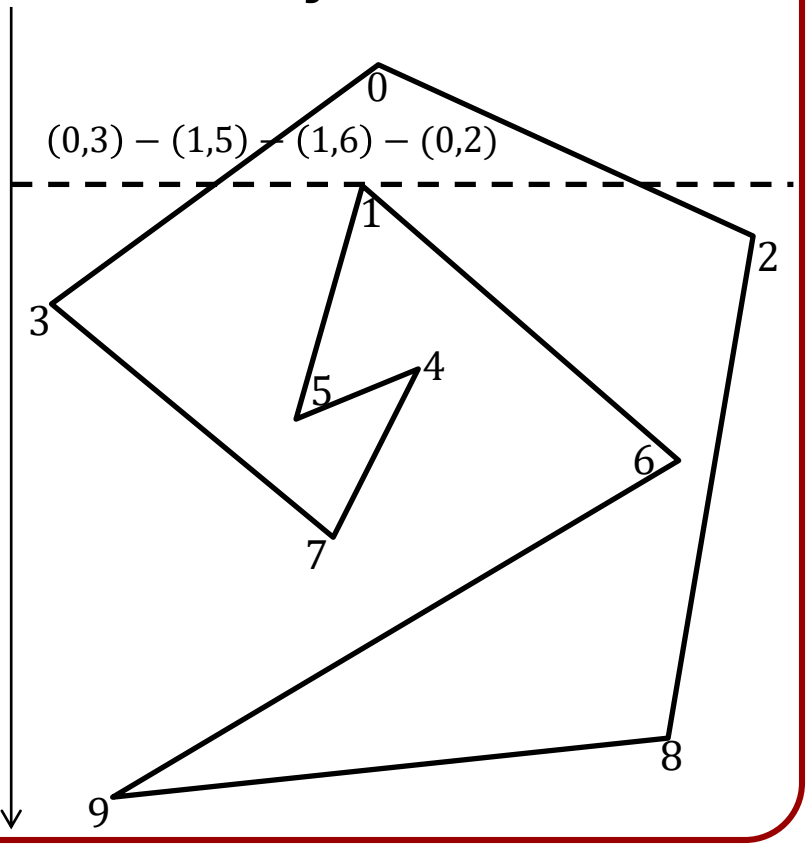
- The active edge list only changes when the horizontal goes through a vertex.
- The segment between the 1st and 2nd active edge, the 3rd and 4th active edge, etc. is interior.

Line/Plane Sweep

Given a polygon P, sweep a horizontal line downwards maintaining a sorted "active edge" list – the edges intersected just below the current horizontal.

<u>Note</u>:

- The active edge list only changes when the horizontal goes through a vertex.
- The segment between the 1st and 2nd active edge, the 3rd and 4th active edge, etc. is interior.

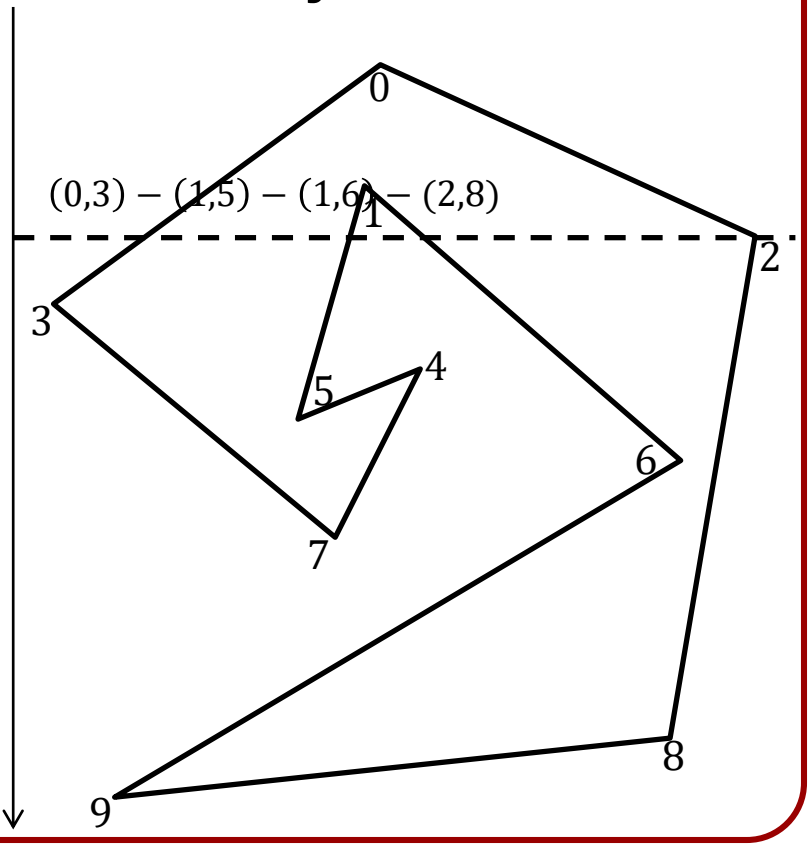


Line/Plane Sweep

Given a polygon P, sweep a horizontal line downwards maintaining a sorted "active edge" list – the edges intersected just below the current horizontal.

Note:

- The active edge list only changes when the horizontal goes through a vertex.
- The segment between the 1st and 2nd active edge, the 3rd and 4th active edge, etc. is interior.

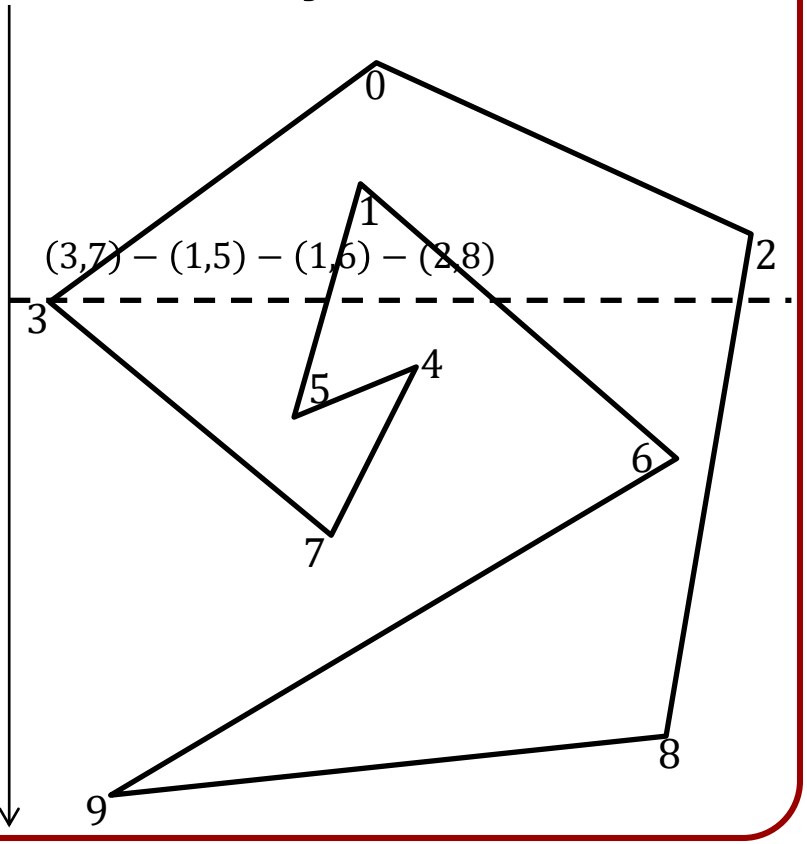


Line/Plane Sweep

Given a polygon P, sweep a horizontal line downwards maintaining a sorted "active edge" list – the edges intersected just below the current horizontal.

Note:

- The active edge list only changes when the horizontal goes through a vertex.
- The segment between the 1st and 2nd active edge, the 3rd and 4th active edge, etc. is interior.

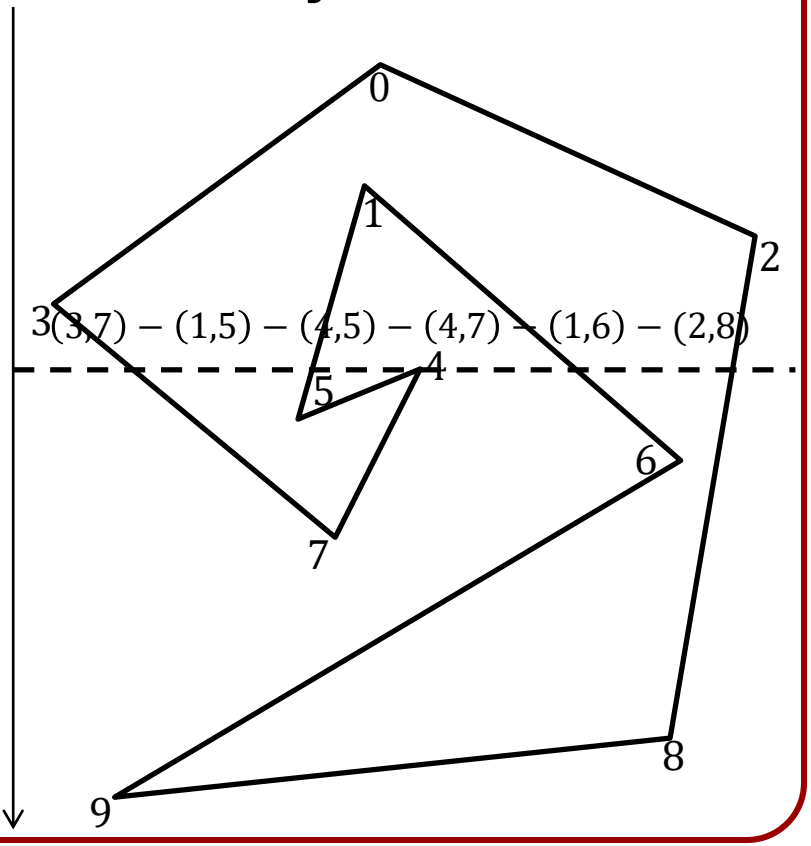


Line/Plane Sweep

Given a polygon P, sweep a horizontal line downwards maintaining a sorted "active edge" list – the edges intersected just below the current horizontal.

Note:

- The active edge list only changes when the horizontal goes through a vertex.
- The segment between the 1st and 2nd active edge, the 3rd and 4th active edge, etc. is interior.

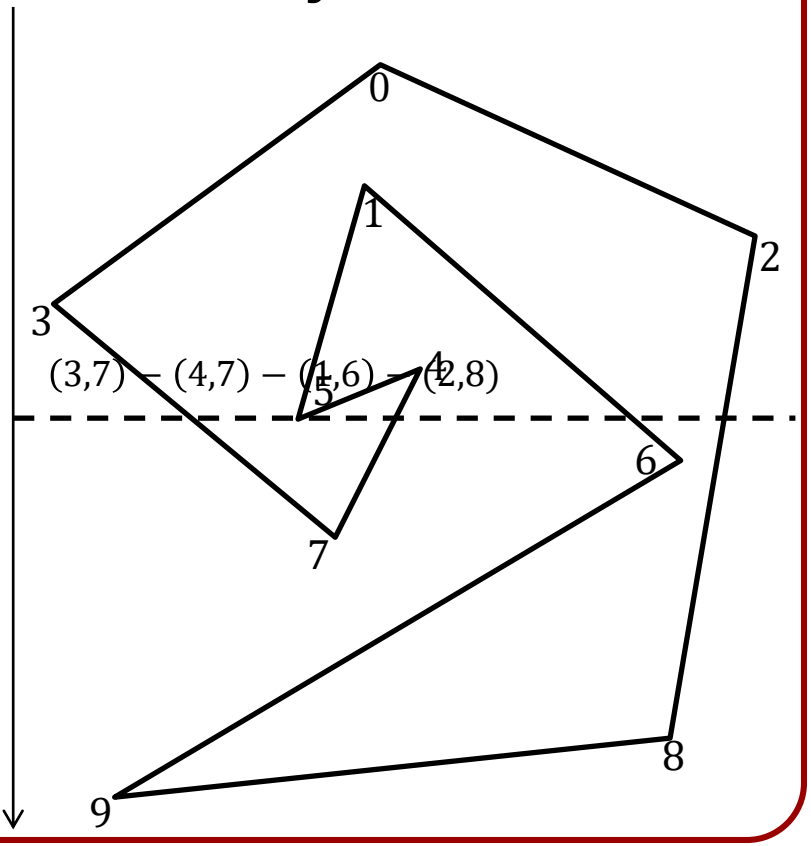


Line/Plane Sweep

Given a polygon P, sweep a horizontal line downwards maintaining a sorted "active edge" list – the edges intersected just below the current horizontal.

Note:

- The active edge list only changes when the horizontal goes through a vertex.
- The segment between the 1st and 2nd active edge, the 3rd and 4th active edge, etc. is interior.

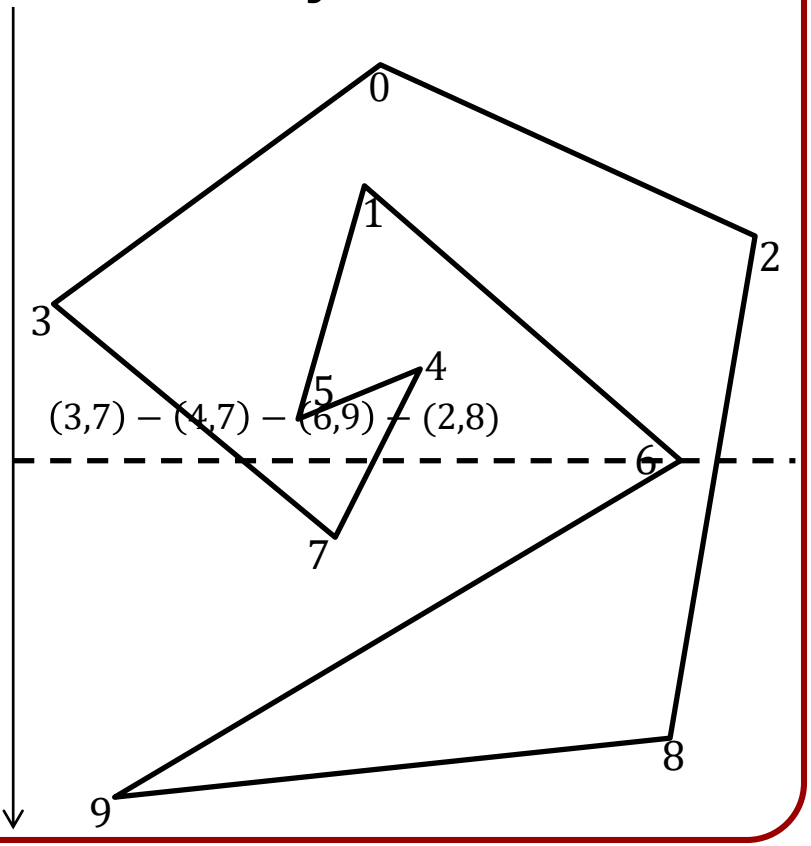


Line/Plane Sweep

Given a polygon P, sweep a horizontal line downwards maintaining a sorted "active edge" list – the edges intersected just below the current horizontal.

Note:

- The active edge list only changes when the horizontal goes through a vertex.
- The segment between the 1st and 2nd active edge, the 3rd and 4th active edge, etc. is interior.

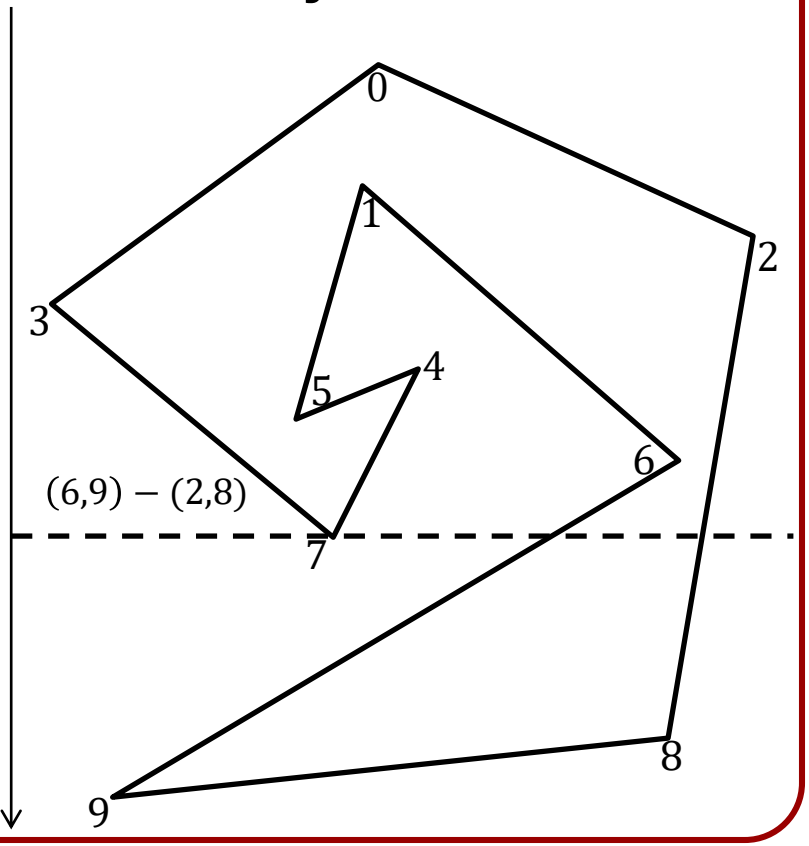


Line/Plane Sweep

Given a polygon P, sweep a horizontal line downwards maintaining a sorted "active edge" list – the edges intersected just below the current horizontal.

Note:

- The active edge list only changes when the horizontal goes through a vertex.
- The segment between the 1st and 2nd active edge, the 3rd and 4th active edge, etc. is interior.

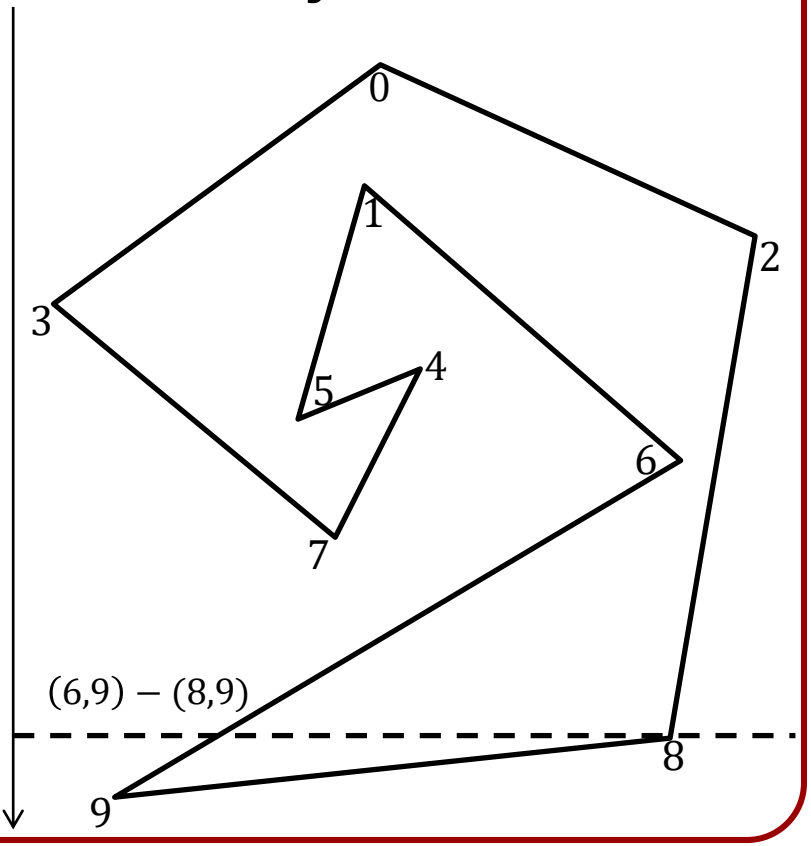


Line/Plane Sweep

Given a polygon P, sweep a horizontal line downwards maintaining a sorted "active edge" list – the edges intersected just below the current horizontal.

Note:

- The active edge list only changes when the horizontal goes through a vertex.
- The segment between the 1st and 2nd active edge, the 3rd and 4th active edge, etc. is interior.

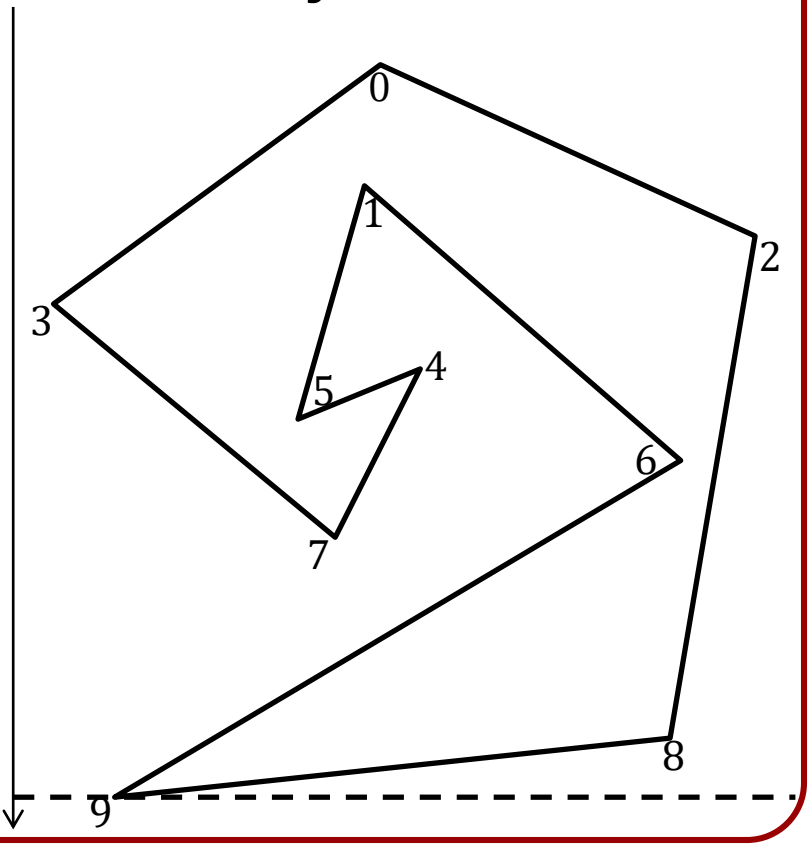


Line/Plane Sweep

Given a polygon P, sweep a horizontal line downwards maintaining a sorted "active edge" list – the edges intersected just below the current horizontal.

Note:

- The active edge list only changes when the horizontal goes through a vertex.
- The segment between the 1st and 2nd active edge, the 3rd and 4th active edge, etc. is interior.

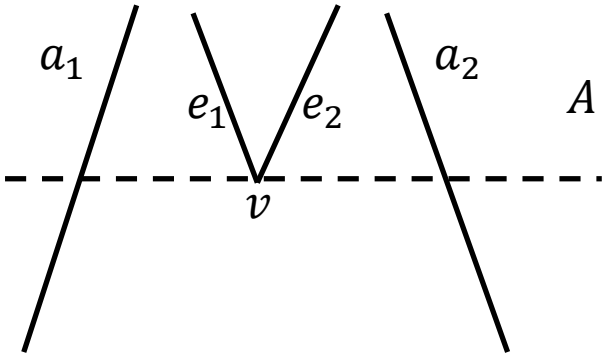


Algorithm

- PlaneSweep(V , $E \subset V \times V$):
  - SortByLargestToSmallestHeight(V)
  - $A \leftarrow \emptyset$
  - For each $v \in V$
    - $(e_1, e_2) \leftarrow$ Edges(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 $v \in V$
    - $(e_1, e_2) \leftarrow$ Edges(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)

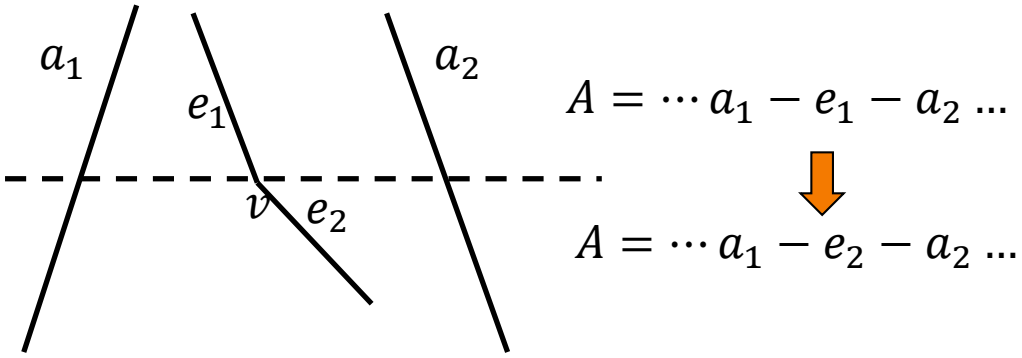


$$A = \cdots a_1 - e_1 - e_2 - a_2 \ldots$$

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

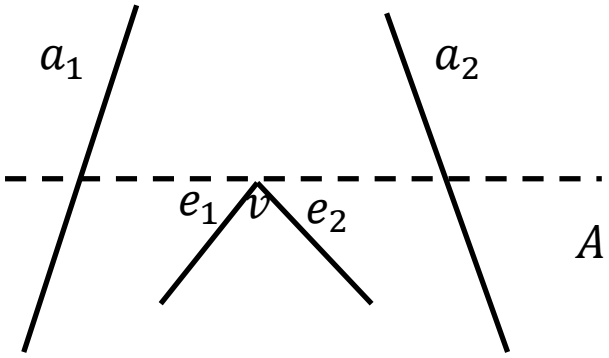
Algorithm

- PlaneSweep(V , $E \subset V \times V$):
  - SortByLargestToSmallestHeight(V)
  - $A \leftarrow \emptyset$
  - For each $v \in V$
    - $(e_1, e_2) \leftarrow$ Edges(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 $v \in V$
    - $(e_1, e_2) \leftarrow$ Edges(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)
  - $A \leftarrow \emptyset$
  - For each $v \in V$
    - $(e_1, e_2) \leftarrow$ Edges(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)

$O(n \log n)$

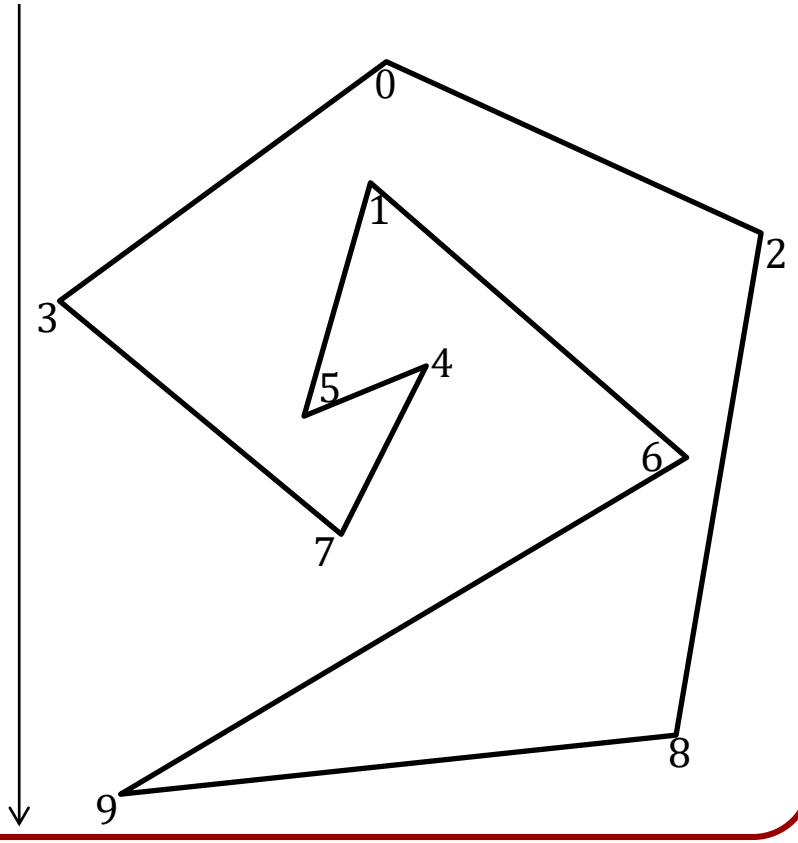
$O(n)$

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

$O(n \log n)$ Monotone Partition

Perform a line-sweep, track active trapezoids, and add diagonals between supporting vertices if one of the vertices is either:

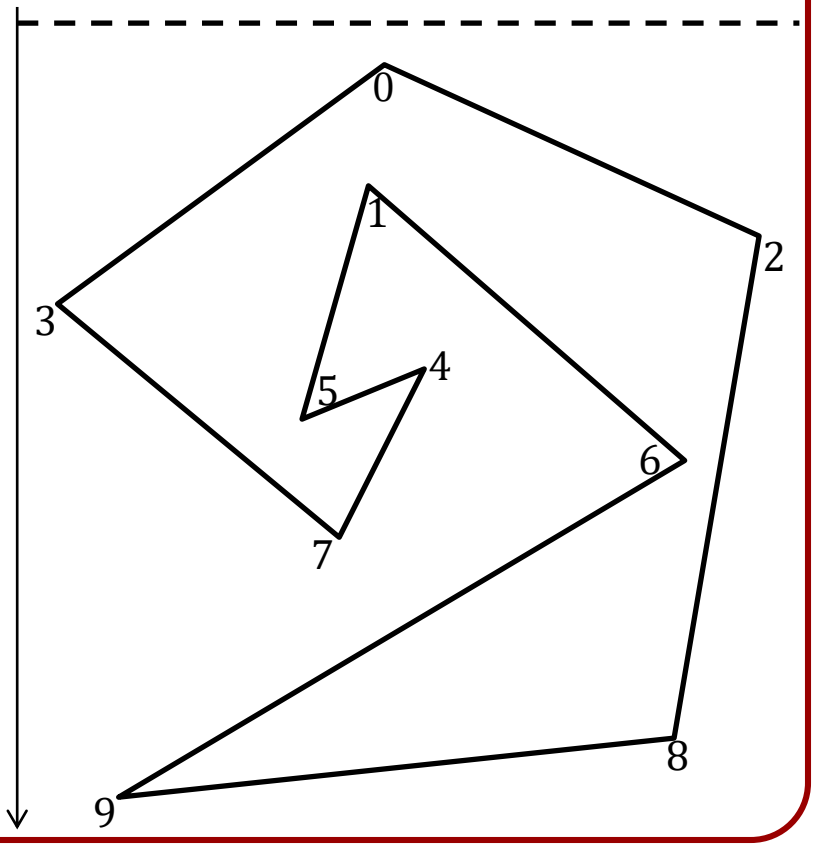
1. up cusp at bottom
2. down cusp at top



$O(n \log n)$ Monotone Partition

Perform a line-sweep, track active trapezoids, and add diagonals between supporting vertices if one of the vertices is either:

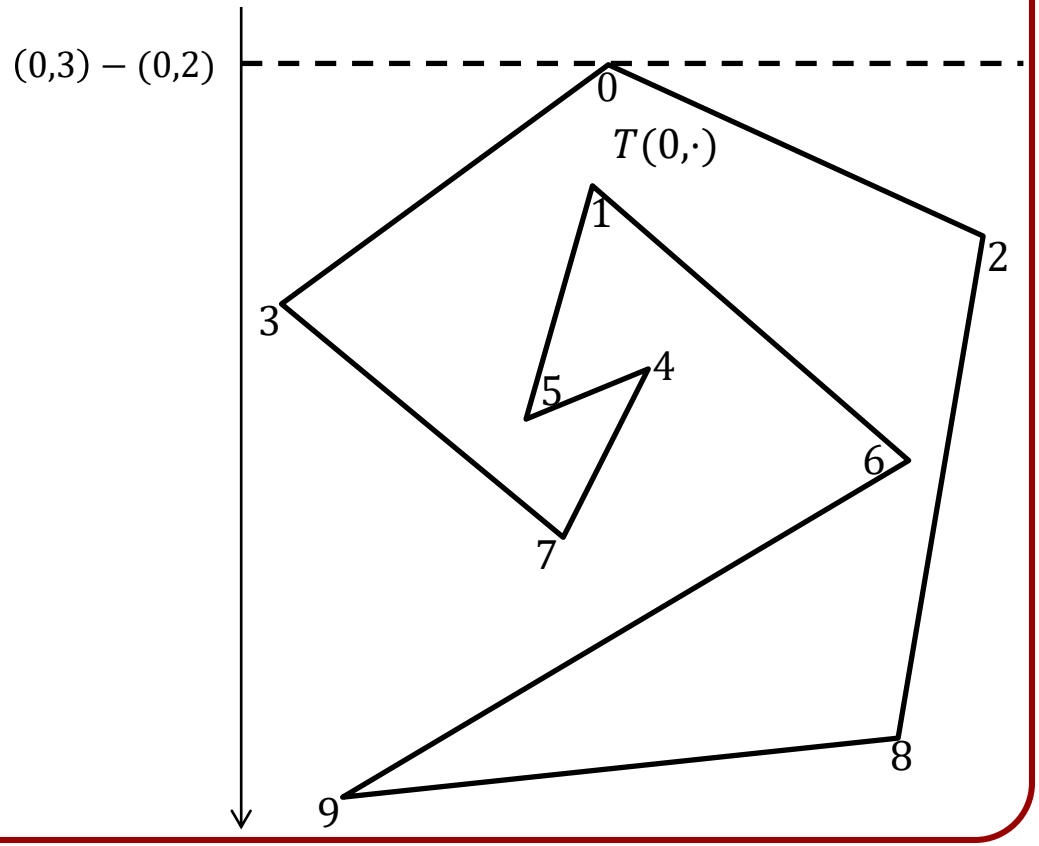
1. up cusp at bottom
2. down cusp at top



$O(n \log n)$ Monotone Partition

Perform a line-sweep, track active trapezoids, and add diagonals between supporting vertices if one of the vertices is either:

1. up cusp at bottom
2. down cusp at top

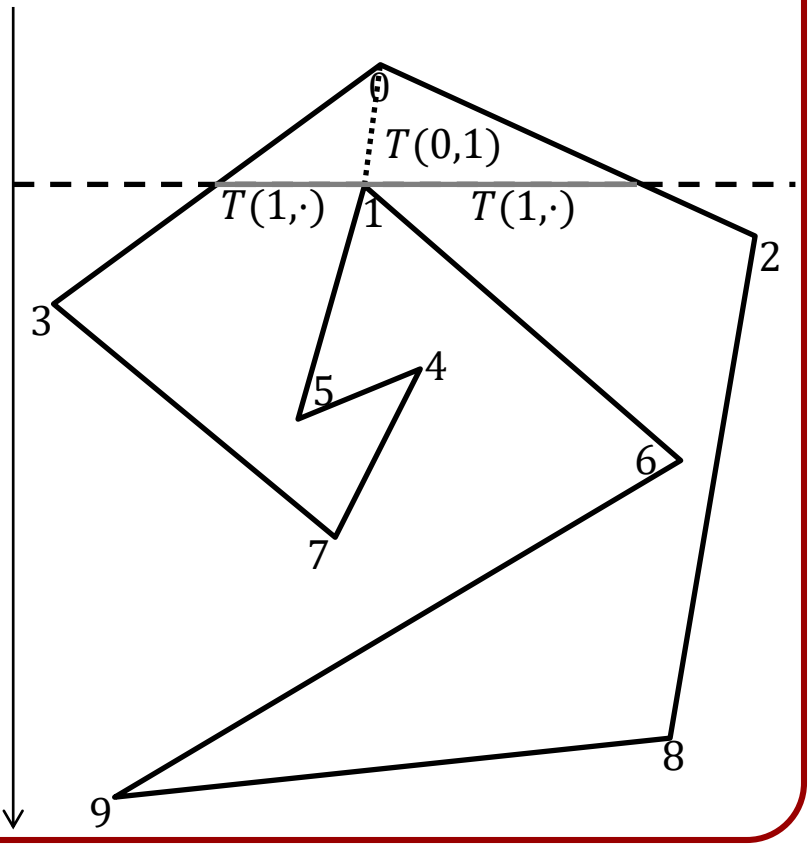


Note:

1. There is a trapezoid between the 1st and 2nd active edges, between the 3rd and 4th active edges, etc.
2. The index of the top supporting vertex is the largest index of the edge vertices, less than or equal to the current vertex of the sweep.

1. up cusp at bottom
2. down cusp at top

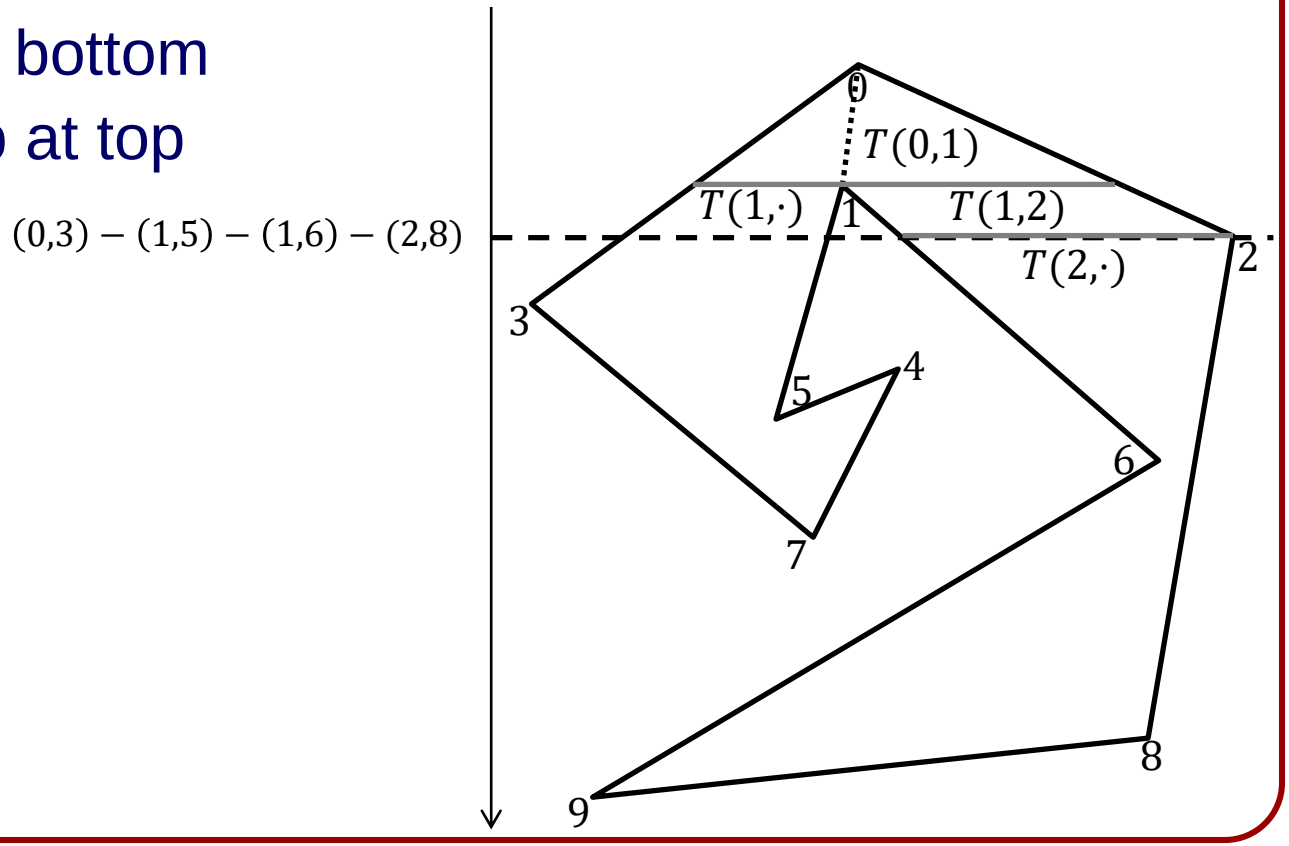
$(0,3) - (1,5) - (1,6) - (0,2)$



$O(n \log n)$ Monotone Partition

Perform a line-sweep, track active trapezoids, and add diagonals between supporting vertices if one of the vertices is either:

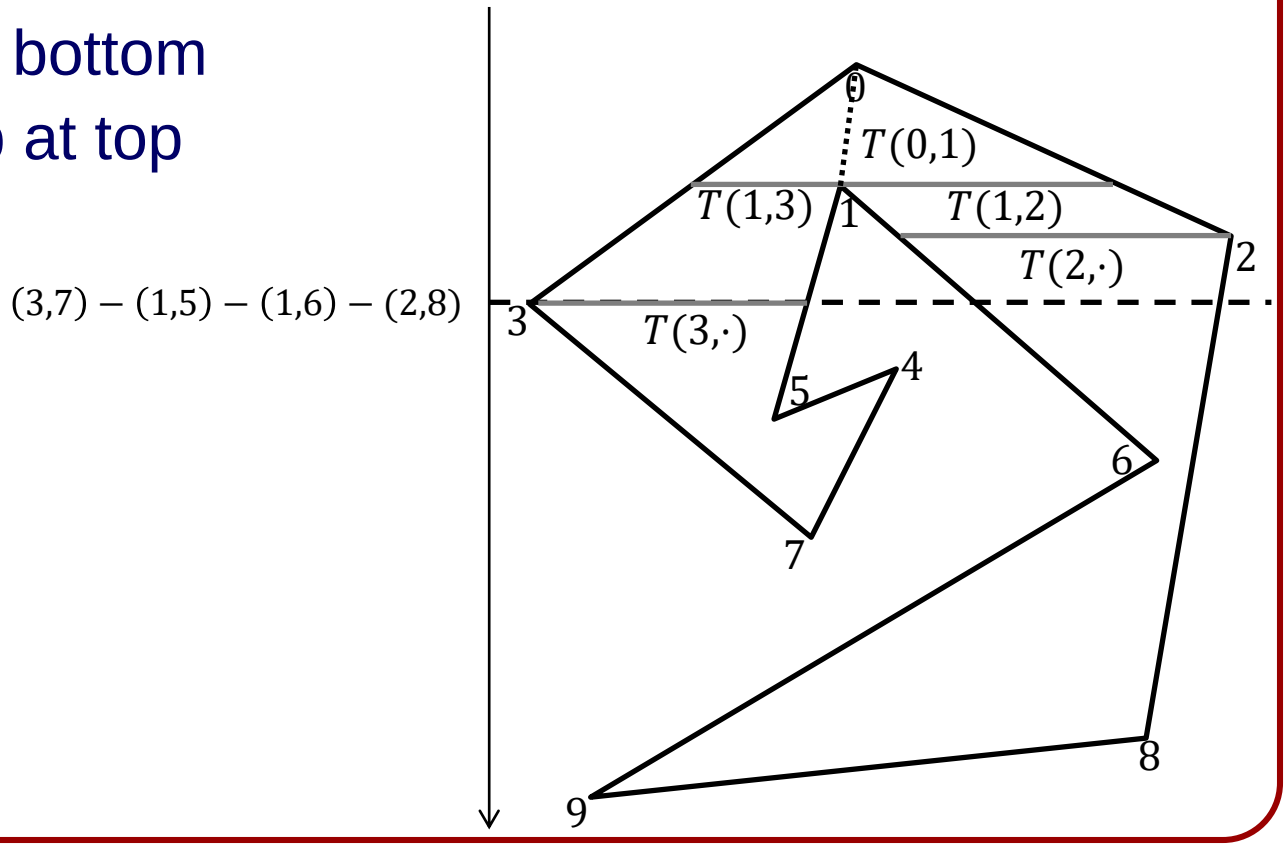
1. up cusp at bottom
2. down cusp at top



$O(n \log n)$ Monotone Partition

Perform a line-sweep, track active trapezoids, and add diagonals between supporting vertices if one of the vertices is either:

1. up cusp at bottom
2. down cusp at top

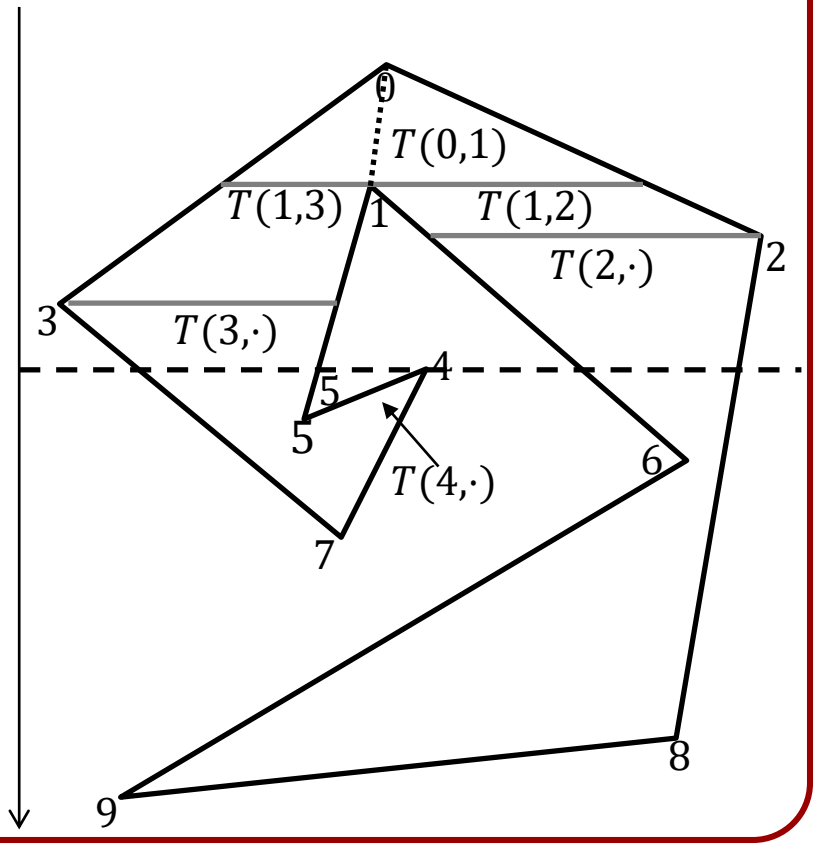


$O(n \log n)$ Monotone Partition

Perform a line-sweep, track active trapezoids, and add diagonals between supporting vertices if one of the vertices is either:

1. up cusp at bottom
2. down cusp at top

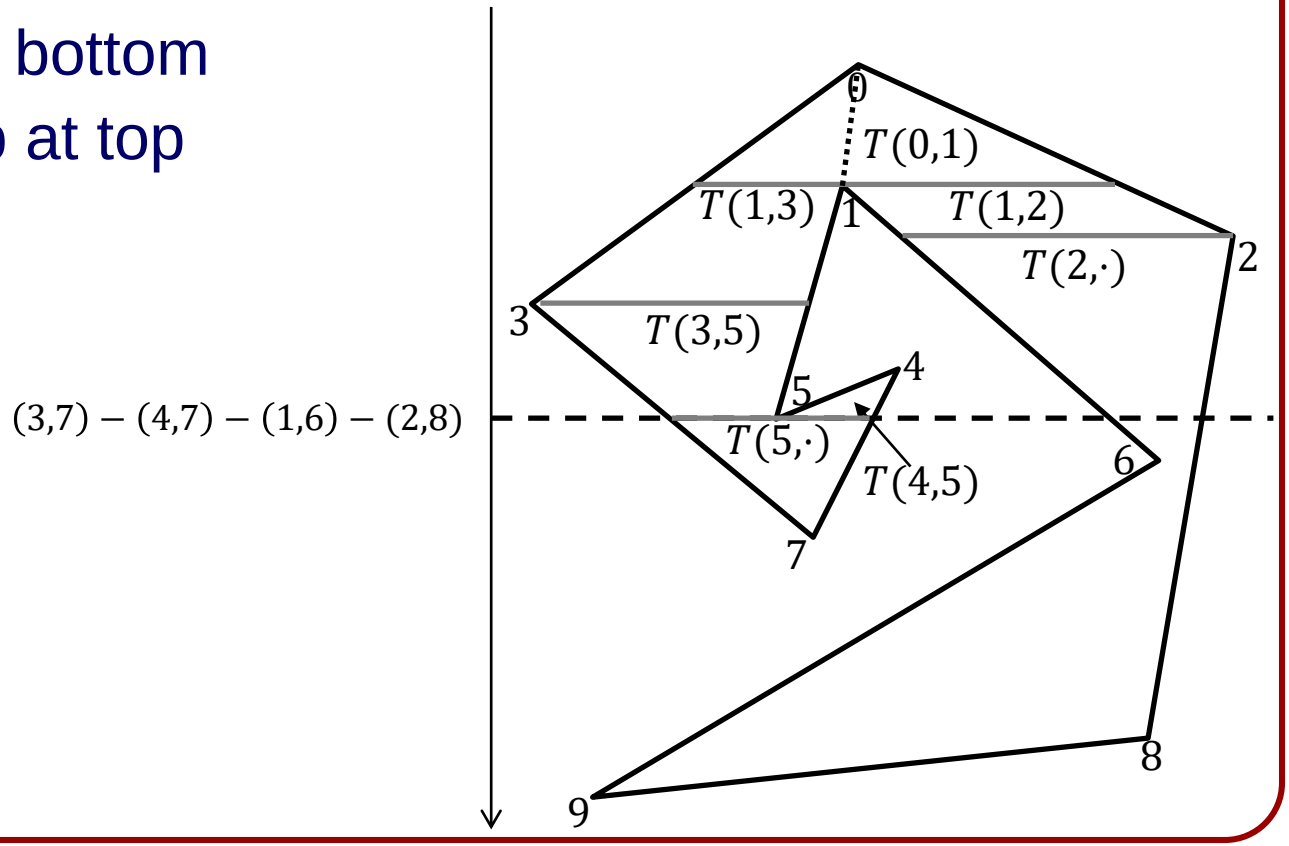
$(3,7) - (1,5) - (4,5) - (4,7) - (1,6) - (2,8)$



$O(n \log n)$ Monotone Partition

Perform a line-sweep, track active trapezoids, and add diagonals between supporting vertices if one of the vertices is either:

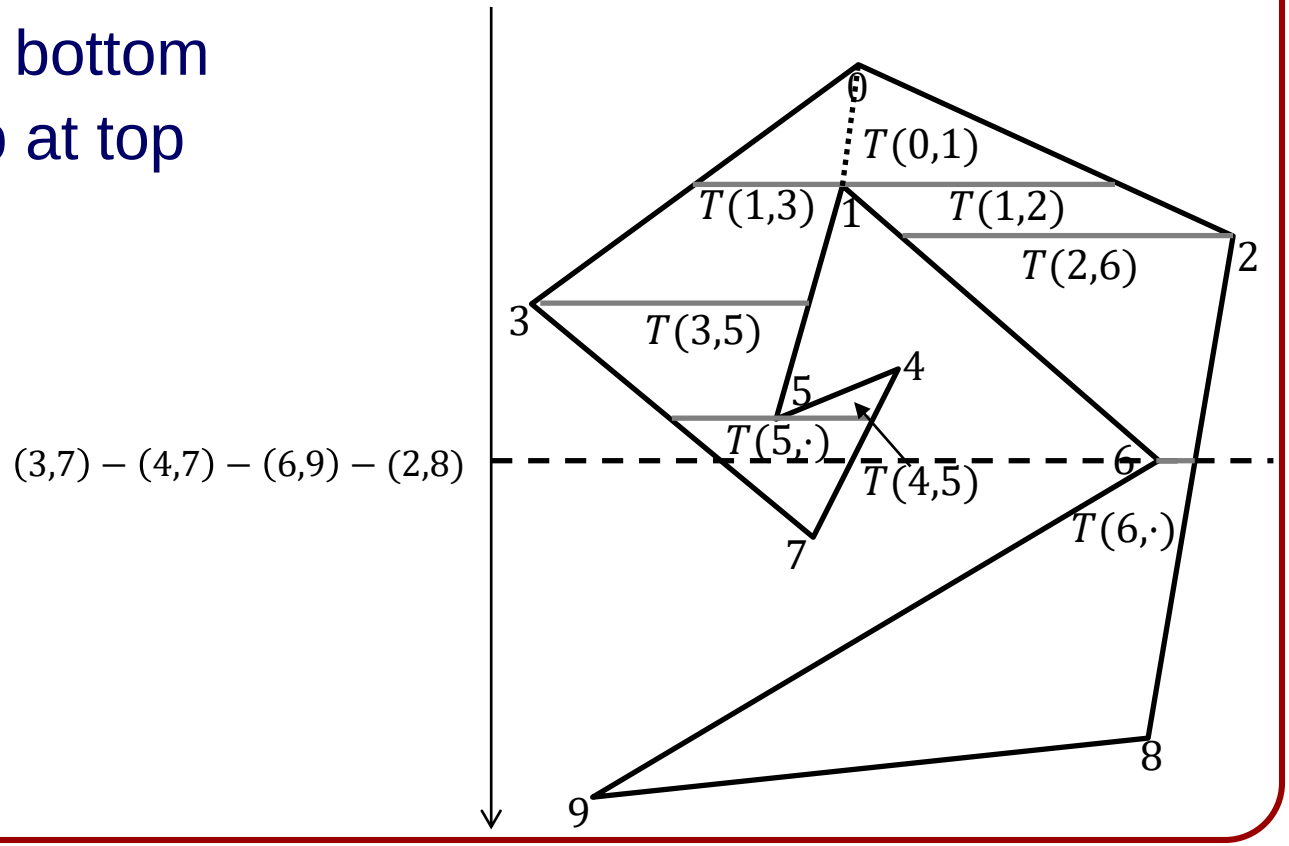
1. up cusp at bottom
2. down cusp at top



$O(n \log n)$ Monotone Partition

Perform a line-sweep, track active trapezoids, and add diagonals between supporting vertices if one of the vertices is either:

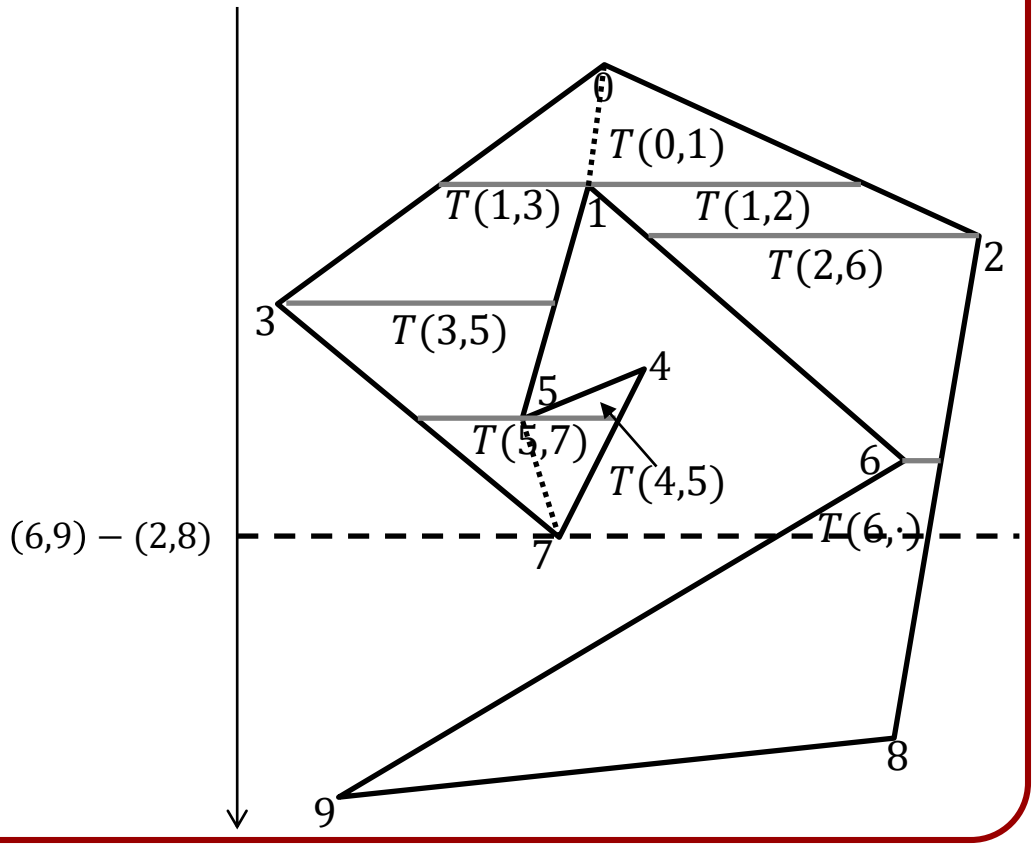
1. up cusp at bottom
2. down cusp at top



$O(n \log n)$ Monotone Partition

Perform a line-sweep, track active trapezoids, and add diagonals between supporting vertices if one of the vertices is either:

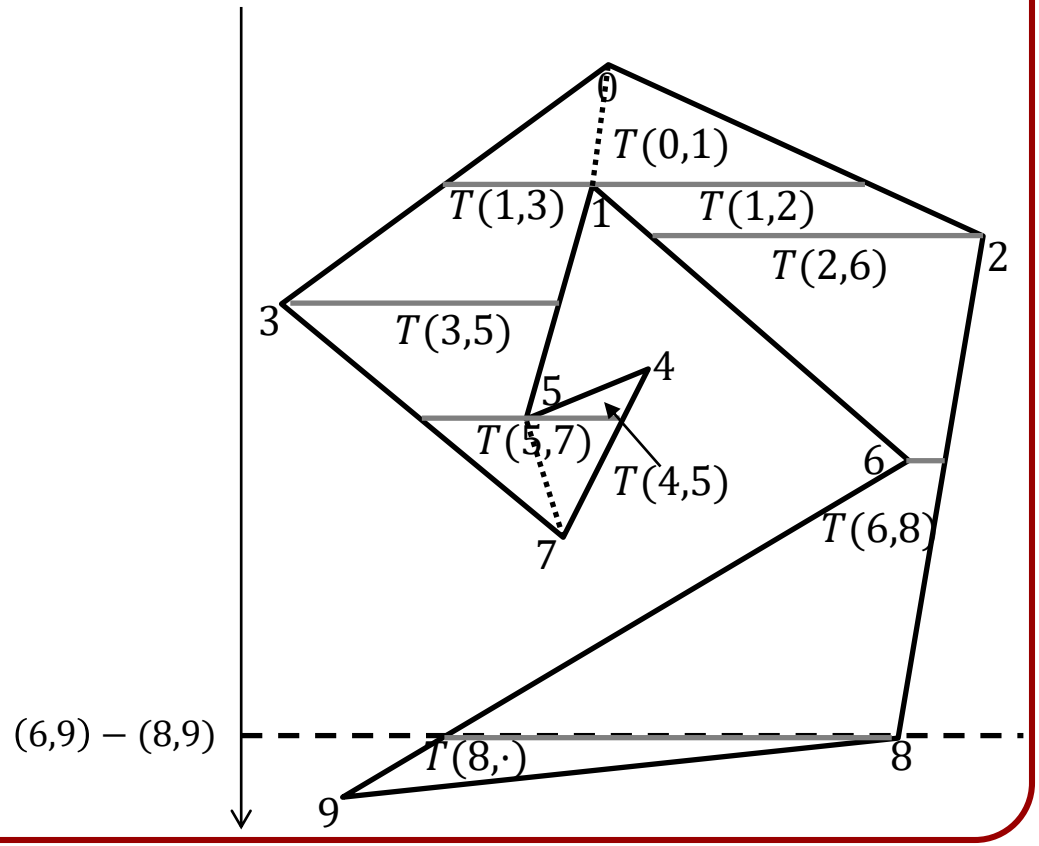
1. up cusp at bottom
2. down cusp at top



$O(n \log n)$ Monotone Partition

Perform a line-sweep, track active trapezoids, and add diagonals between supporting vertices if one of the vertices is either:

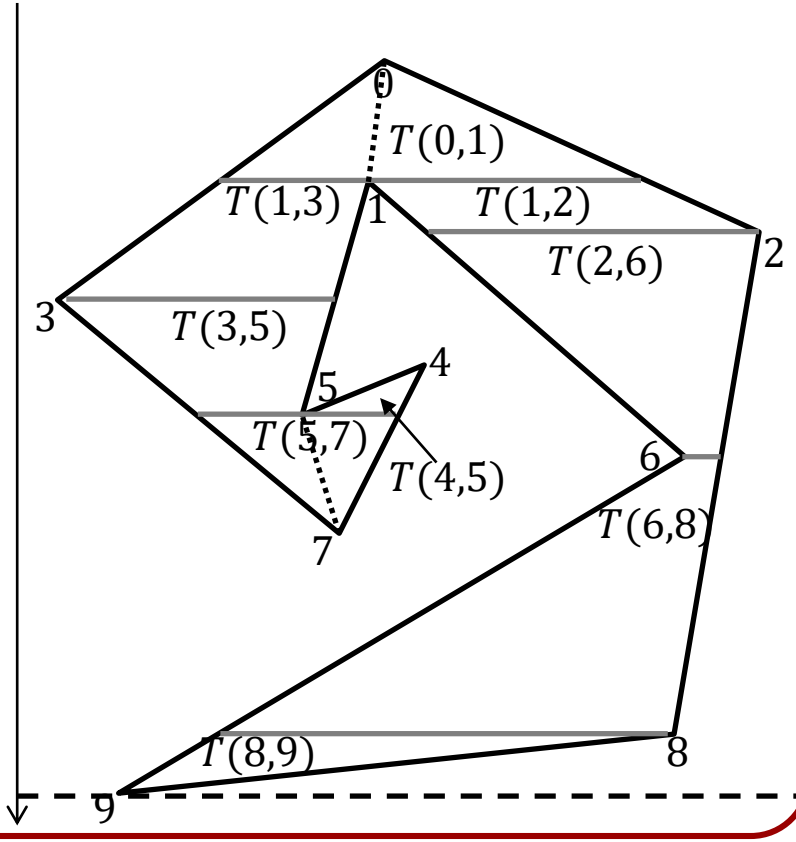
1. up cusp at bottom
2. down cusp at top



$O(n \log n)$ Monotone Partition

Perform a line-sweep, track active trapezoids, and add diagonals between supporting vertices if one of the vertices is either:

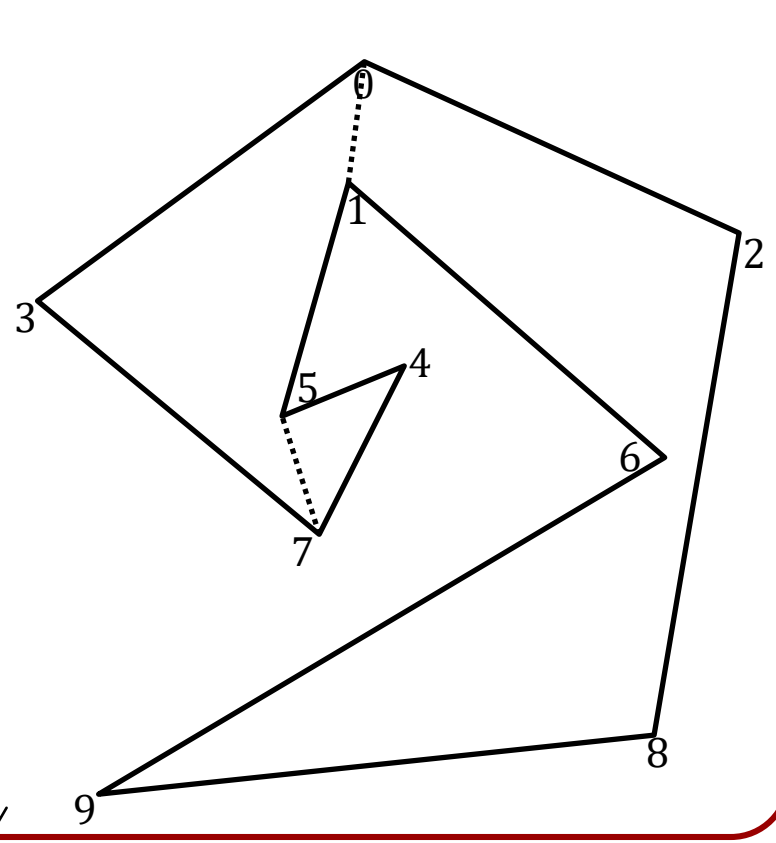
1. up cusp at bottom
2. down cusp at top



$O(n \log n)$ Monotone Partition

Perform a line-sweep, track active trapezoids, and add diagonals between supporting vertices if one of the vertices is either:

1. up cusp at bottom
2. down cusp at top



$O(n \log n)$ Monotone Partition

Note:

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

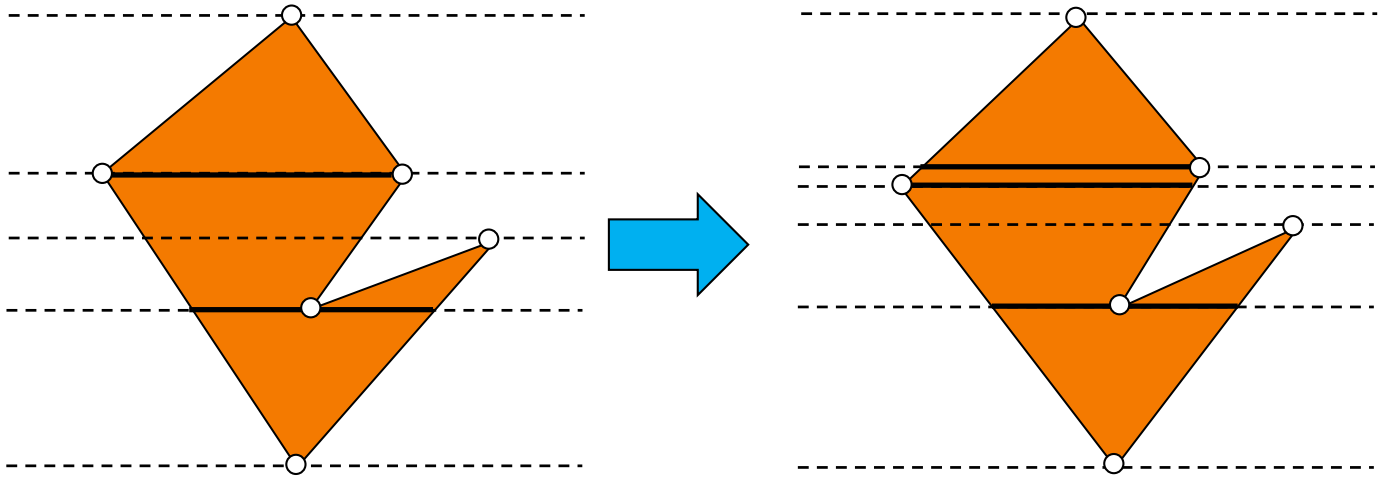
This isn't necessary. It suffices to sort lexicographically. (If two vertices have the same y-coordinates then the one with larger x-coordinate is first.)

$O(n \log n)$ Monotone Partition

Note:

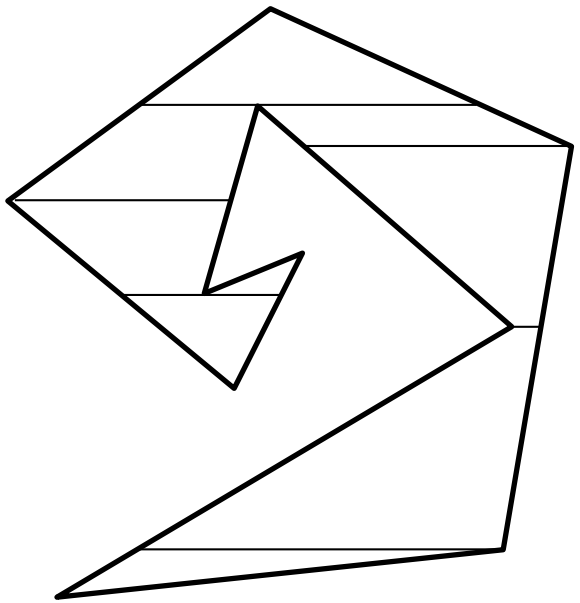
We had assumed that the vertices have different y-coordinates.

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



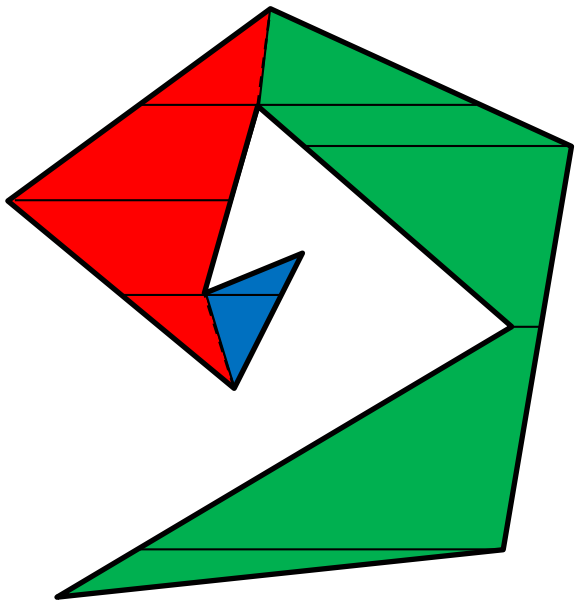
Triangulation

- Triangulate(P):
  - Partition into monotone polygons



Triangulation

- Triangulate(P):
  - Partition into monotone polygons



Triangulation

- Triangulate(P):
  - Partition into monotone polygons
  - Triangulate the monotone polygons