Computational Geometry
[csci 3250]

Laura Toma

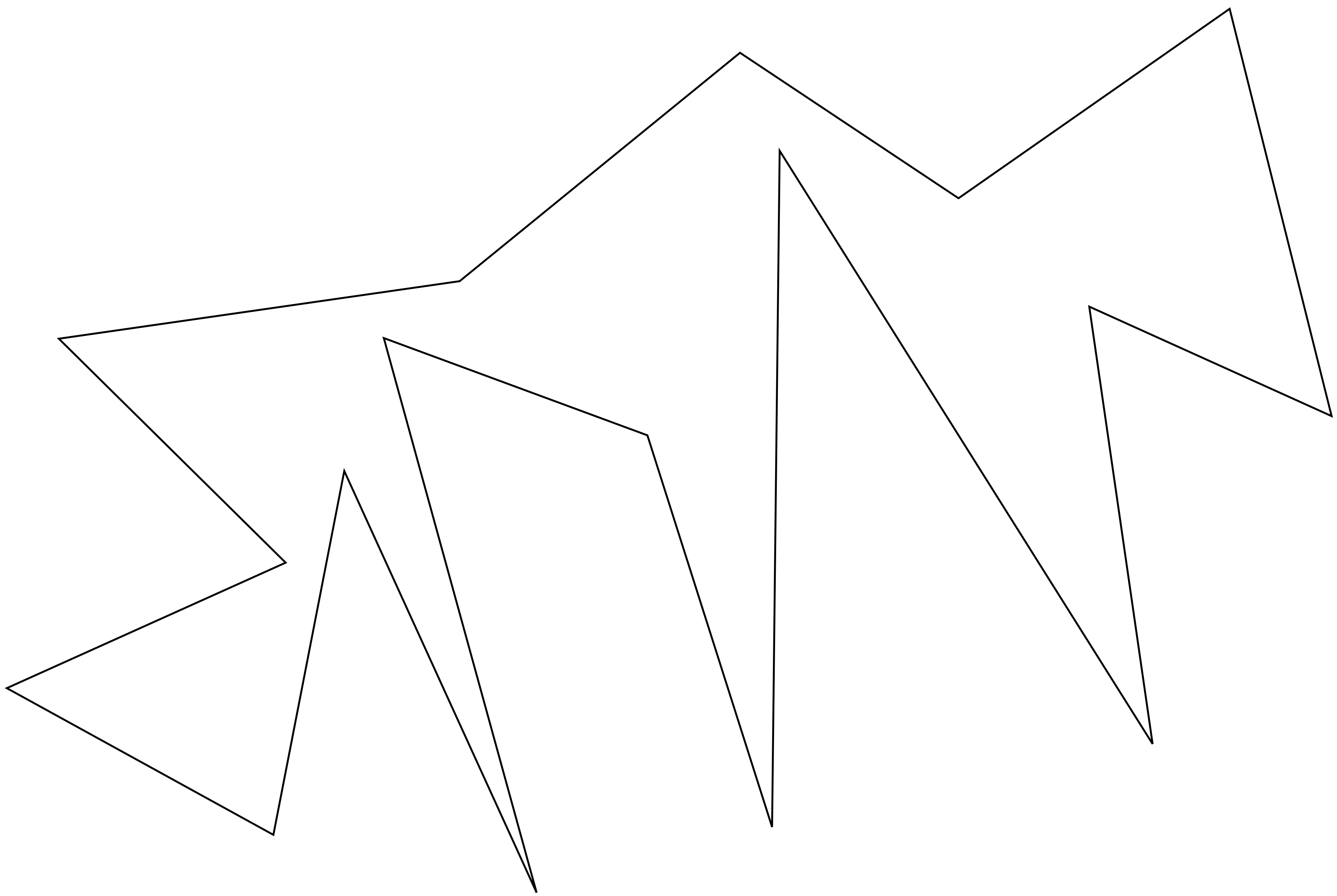
Bowdoin College

Polygon Triangulation

Polygon Triangulation

The problem: Triangulate a given polygon.

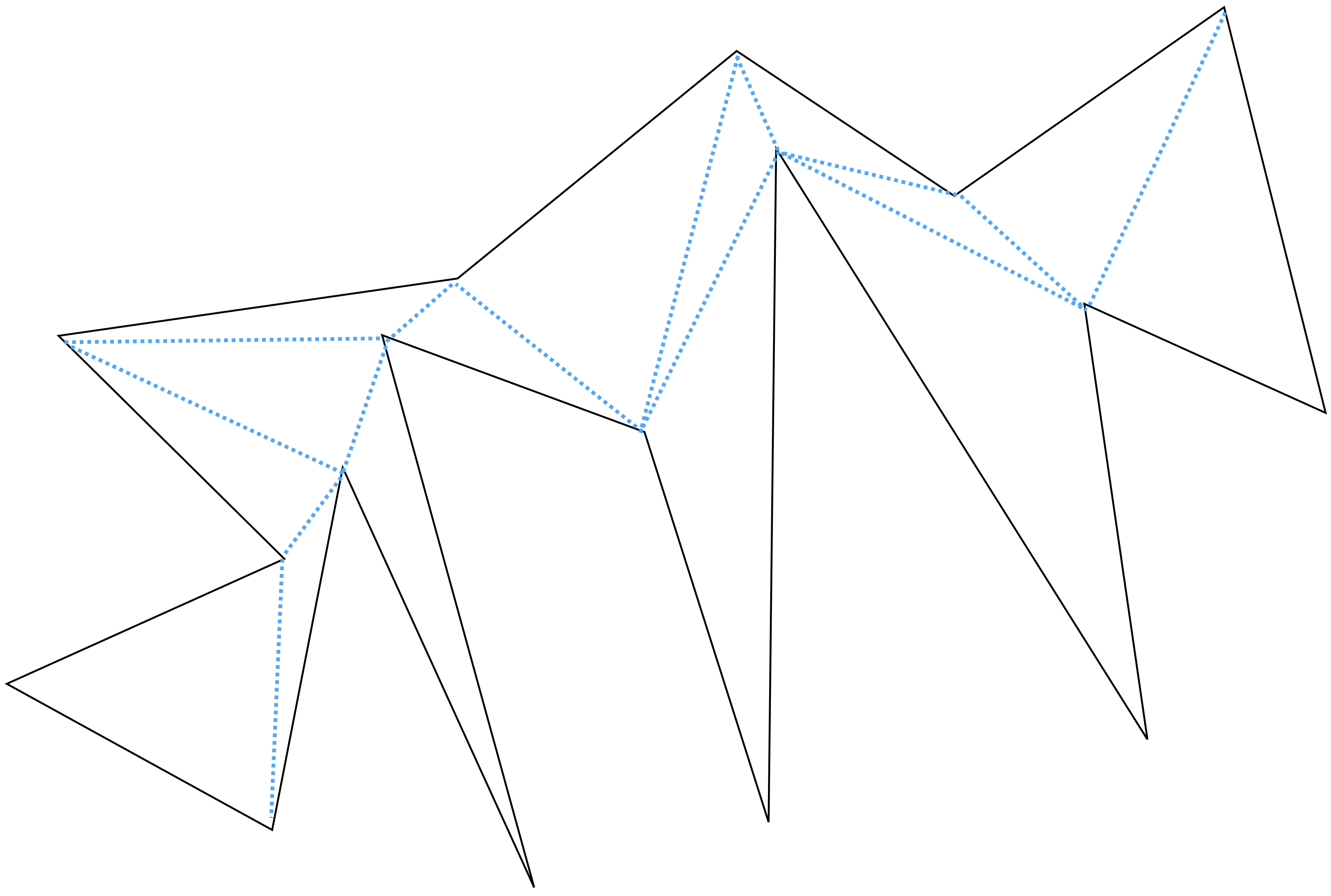
(output a set of diagonals that partition the polygon into triangles).



Polygon Triangulation

The problem: Triangulate a given polygon.

(output a set of diagonals that partition the polygon into triangles).



Known Results

- Given a simple polygon P, a **diagonal** is a segment between 2 non-adjacent vertices that lies entirely within the interior of the polygon.

- Theorem: Any simple polygon with $n>3$ vertices contains (at least) a diagonal.

- Theorem: Any polygon can be triangulated.

- A set of 3 consecutive vertices v_{i-1}, v_i, v_{i+1} defines an ear if $v_{i-1}v_{i+1}$ is a diagonal.

- Theorem: Any polygon has at least two ears.

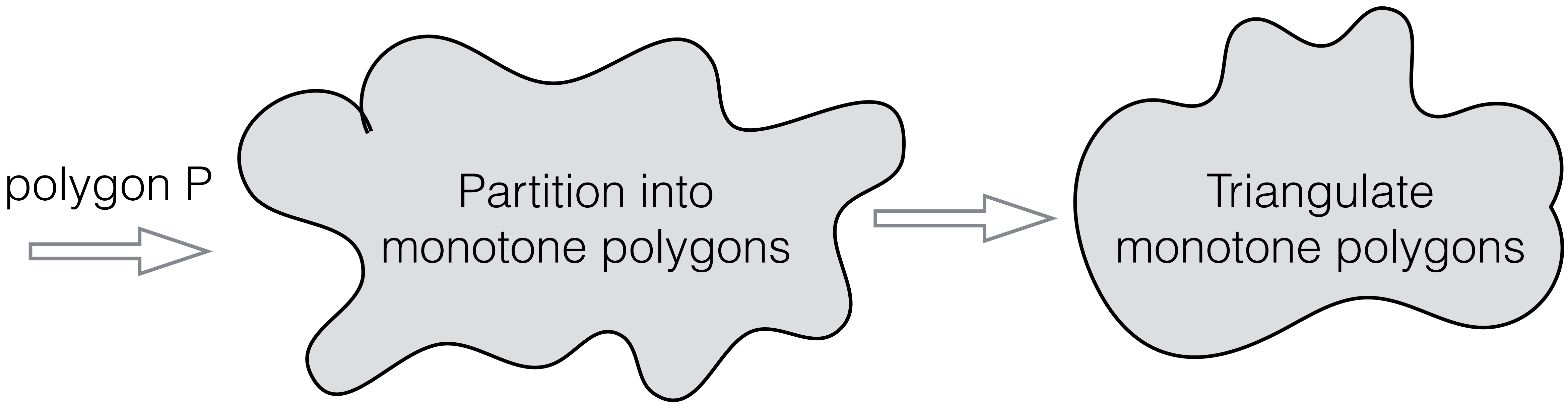
First steps

- Triangulation by identifying ears

  - Idea: Find an ear, output the diagonal, delete the ear tip, repeat.
  - Analysis:
    - checking whether a vertex is ear tip or not: $O(n)$
    - finding an ear $O(n)$
    - overall $O(n^3)$

  - Can be improved to $O(n^2)$
    - Idea: When you remove a ear tip from the polygon, only the adjacent vertices might change their ear status

Towards an O(n lg n) Polygon Triangulation Algorithm

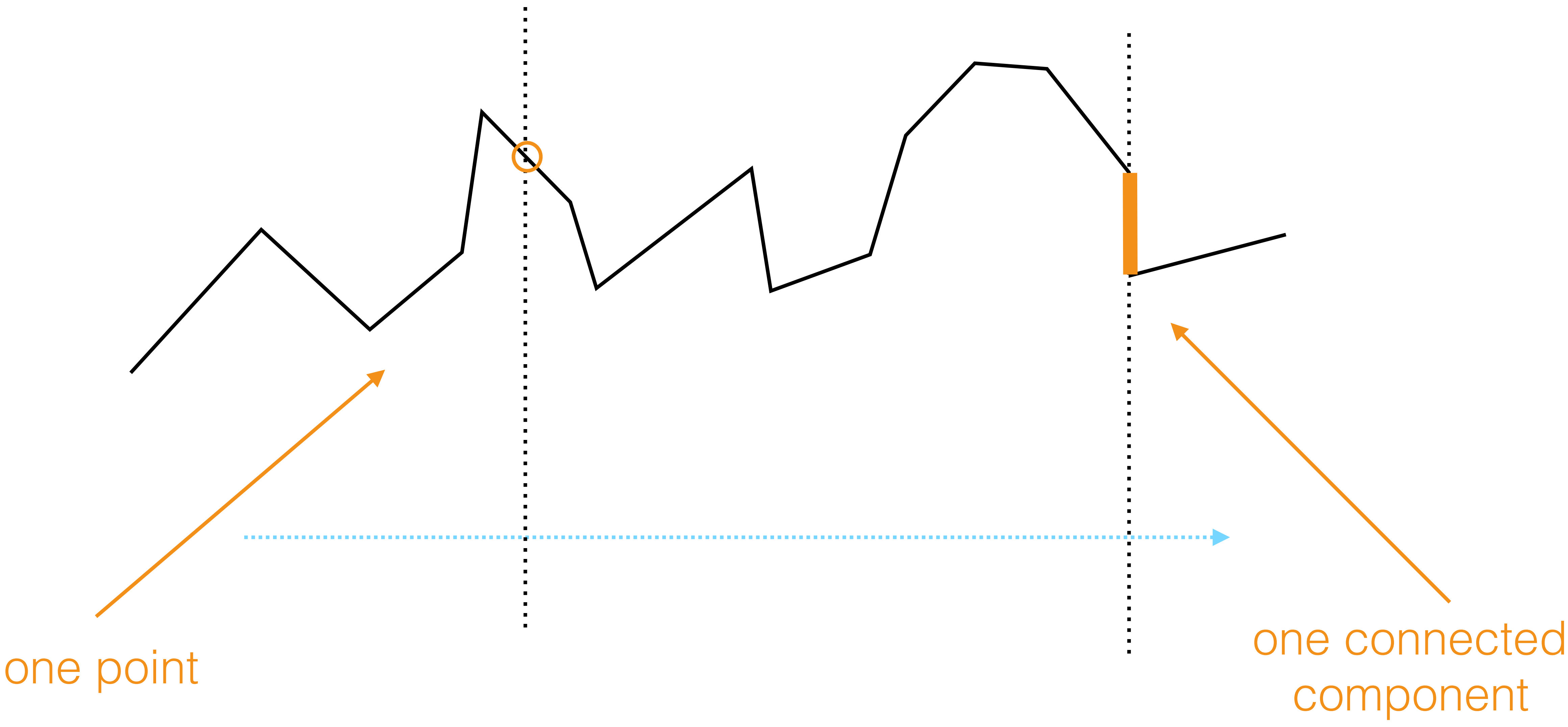


Monotone chains

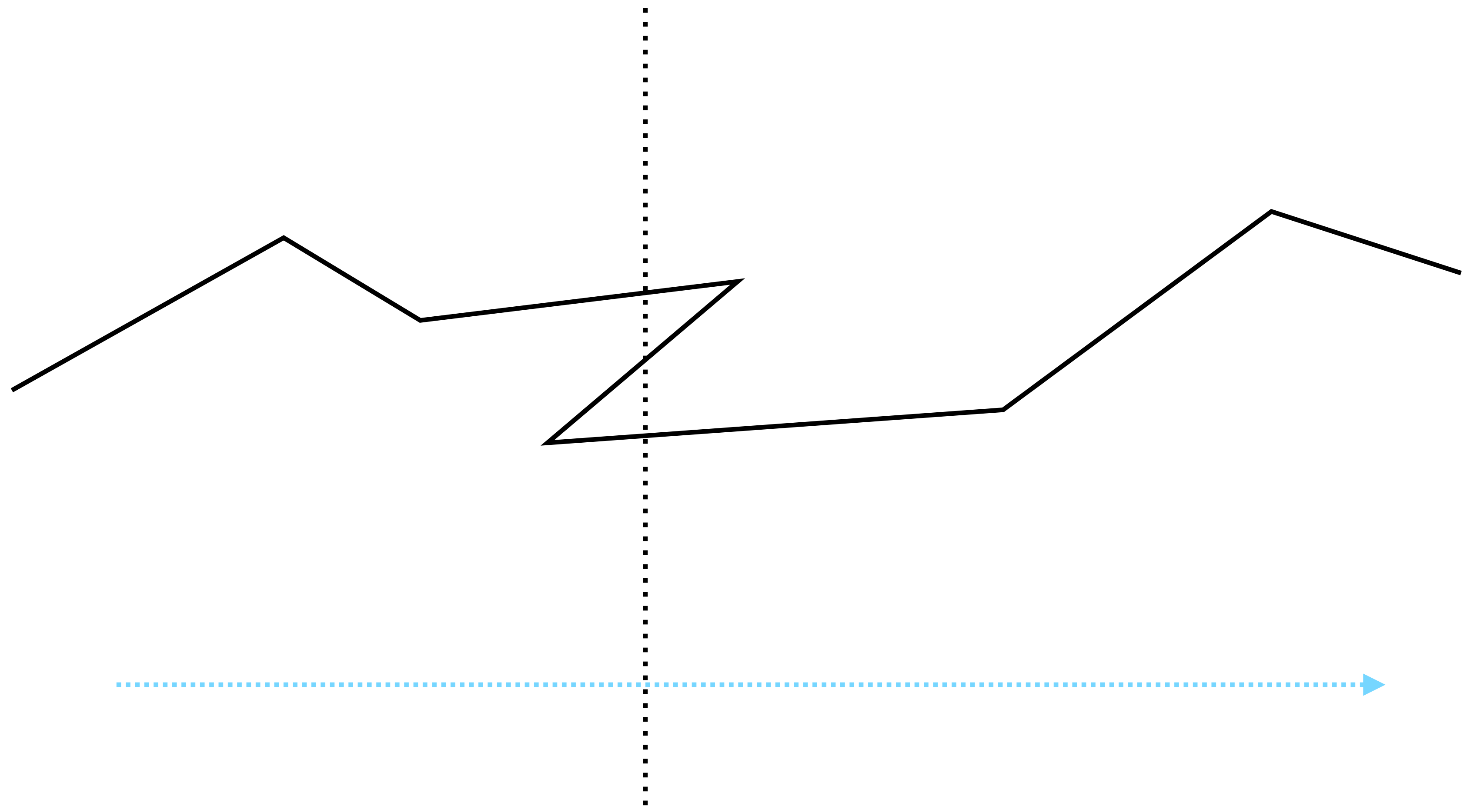
A polygonal chain is **x-monotone** if any line perpendicular to x-axis intersects it in one point (one connected component).

Monotone chains

A polygonal chain is **x-monotone** if any line perpendicular to **x-axis** intersects it in one point (one connected component).



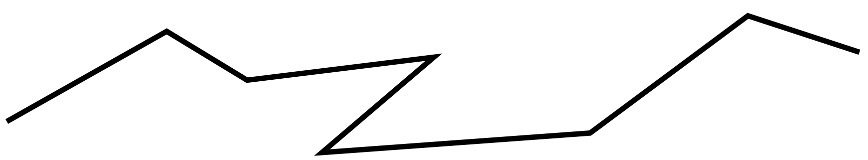
Monotone chains



Not x-monotone

Monotone chains

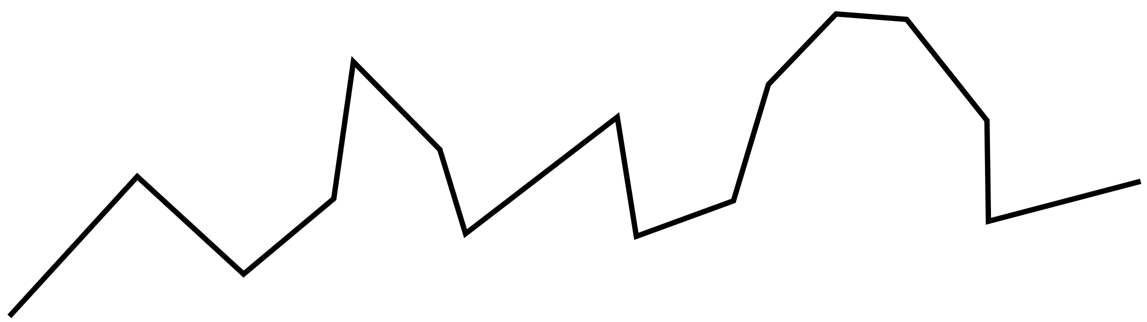
x-monotone



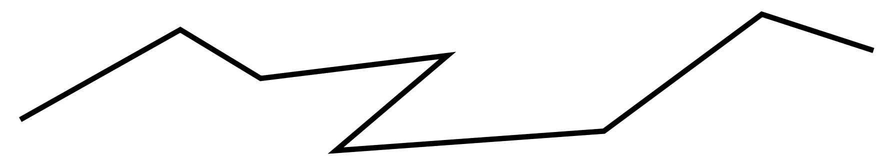
not x-monotone

- Let u and v be the points on the chain with min/max x-coordinate.
- The vertices on the boundary of an x-monotone chain, going from u to v, are in x-order.

Monotone chains



x-monotone

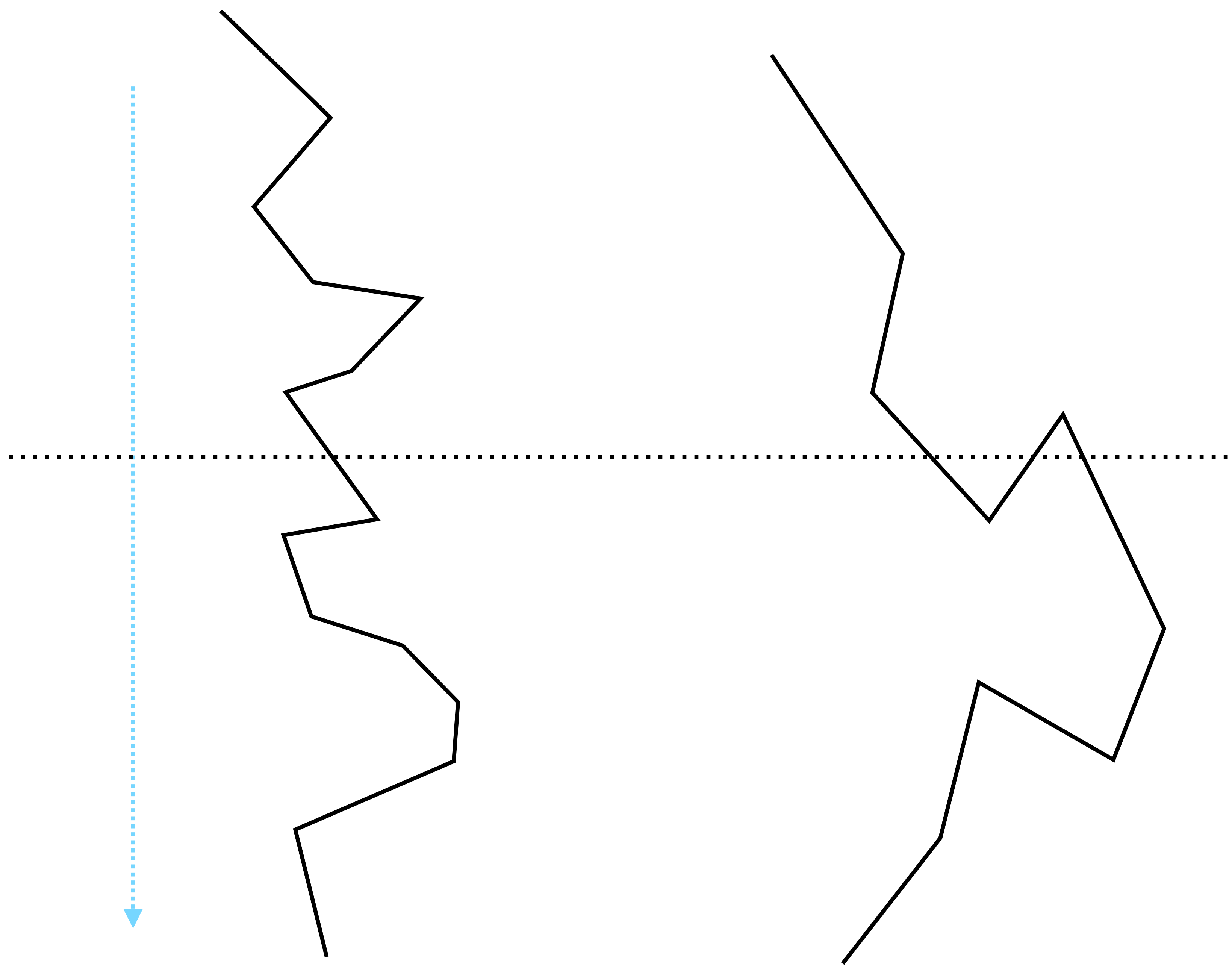


not x-monotone

As you travel along this chain, your x-coordinate is staying the same or increasing

Monotone chains

A polygonal chain is **y-monotone** if any line perpendicular to **y-axis** intersects it in one point (one connected component).

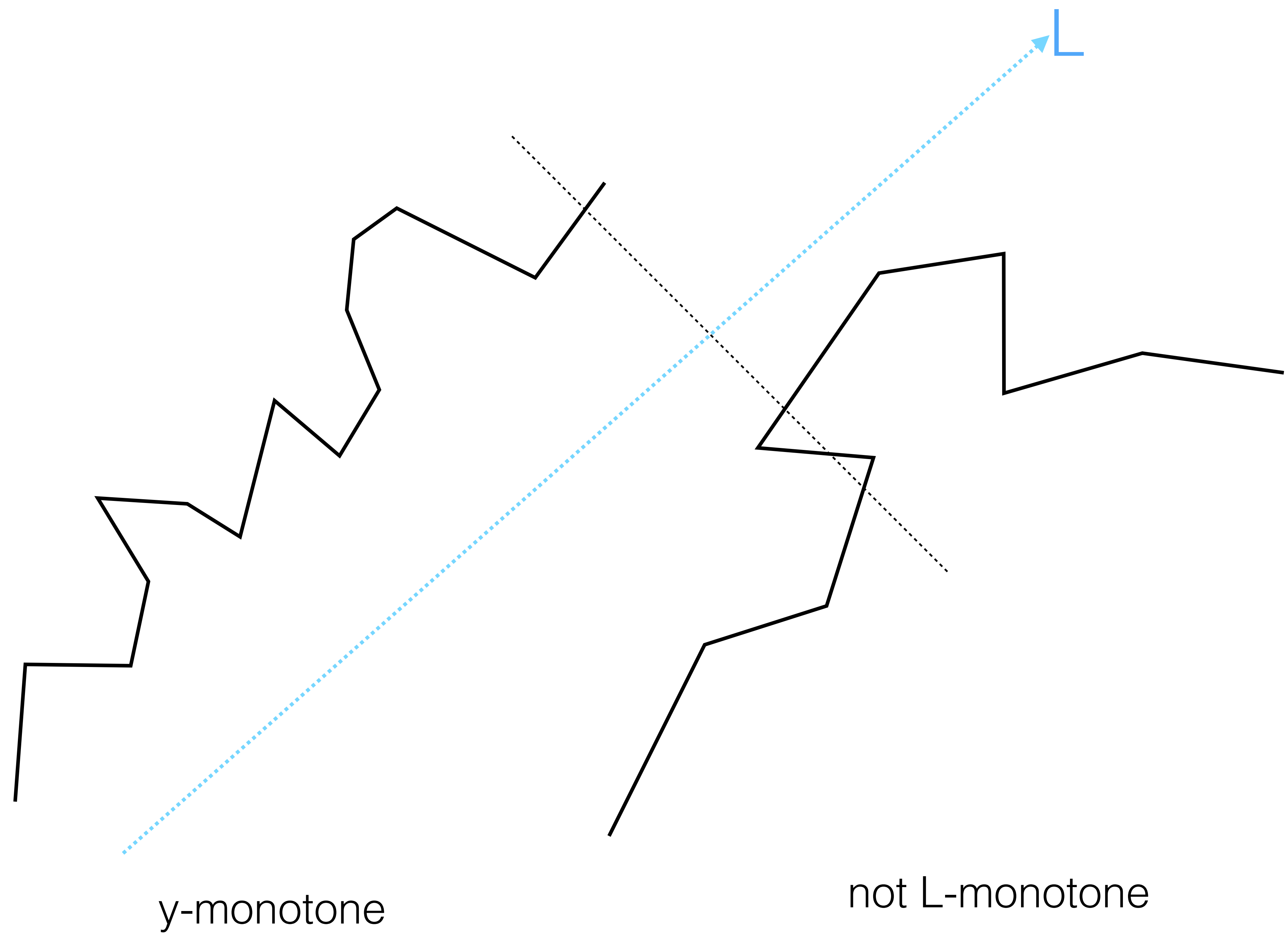


y-monotone

not y-monotone

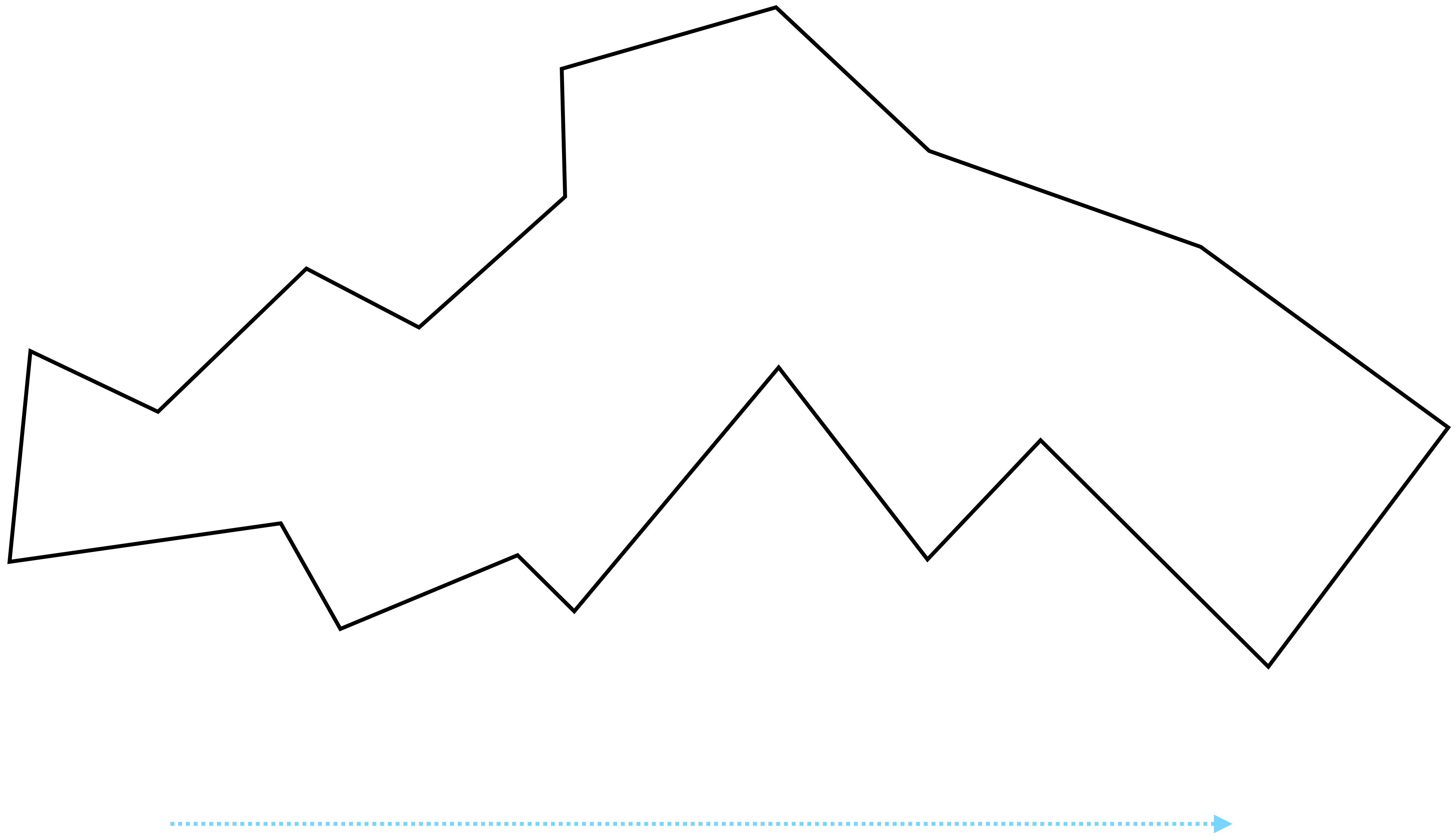
Monotone chains

A polygonal chain is **L-monotone** if any line perpendicular to **line L** intersects it in one point (one connected component).



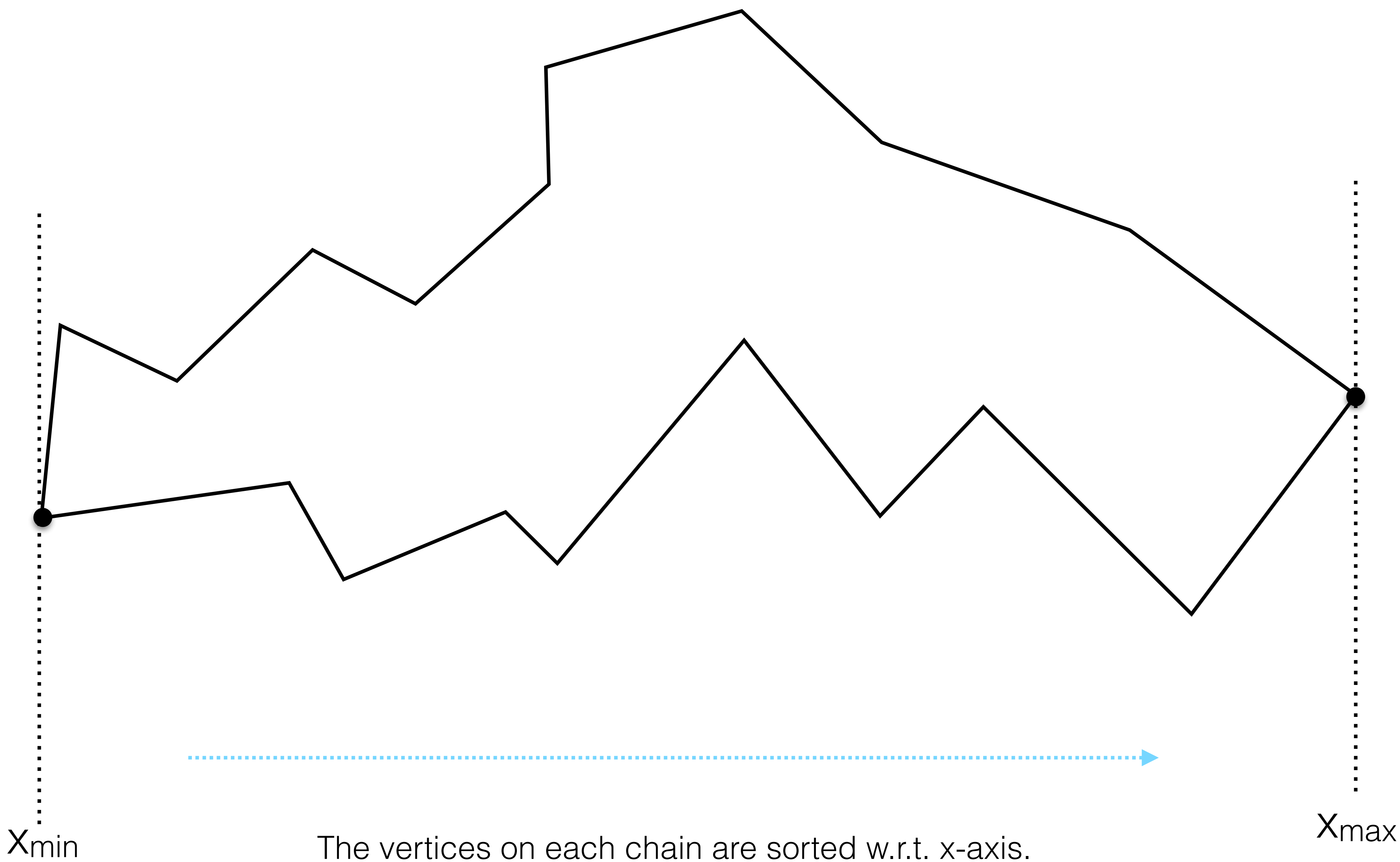
Monotone polygons

A polygon is **x-monotone** if its boundary can be split into two x-monotone chains.



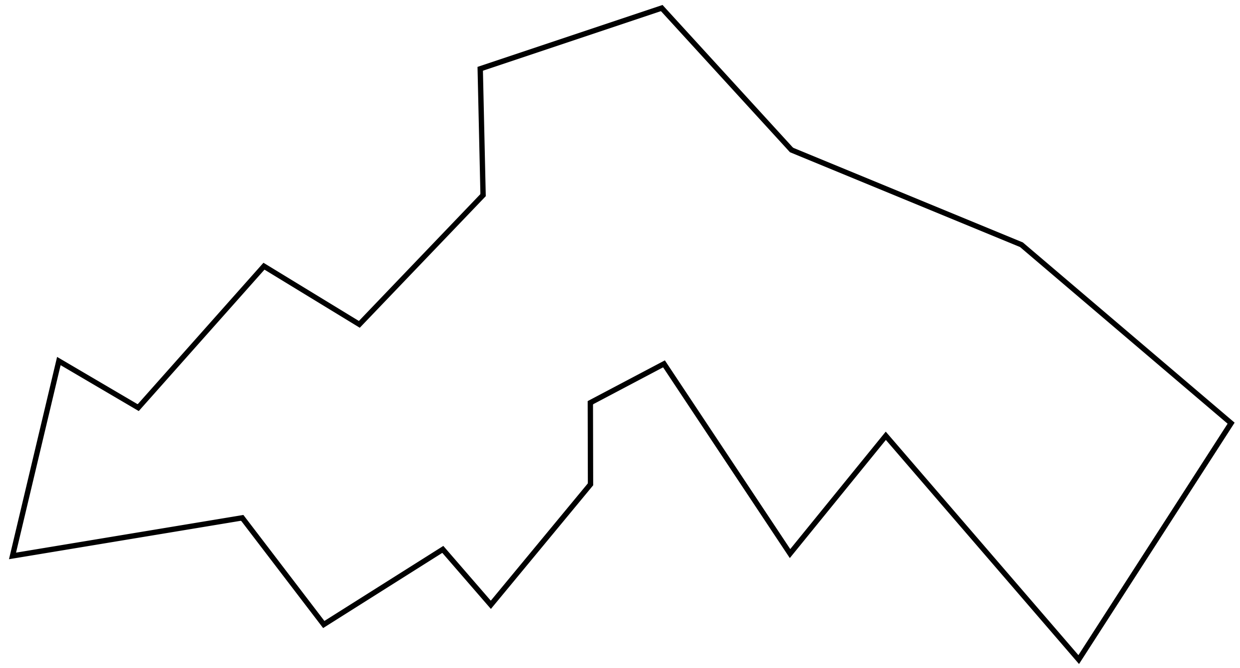
Monotone polygons

A polygon is **x-monotone** if its boundary can be split into two x-monotone chains.

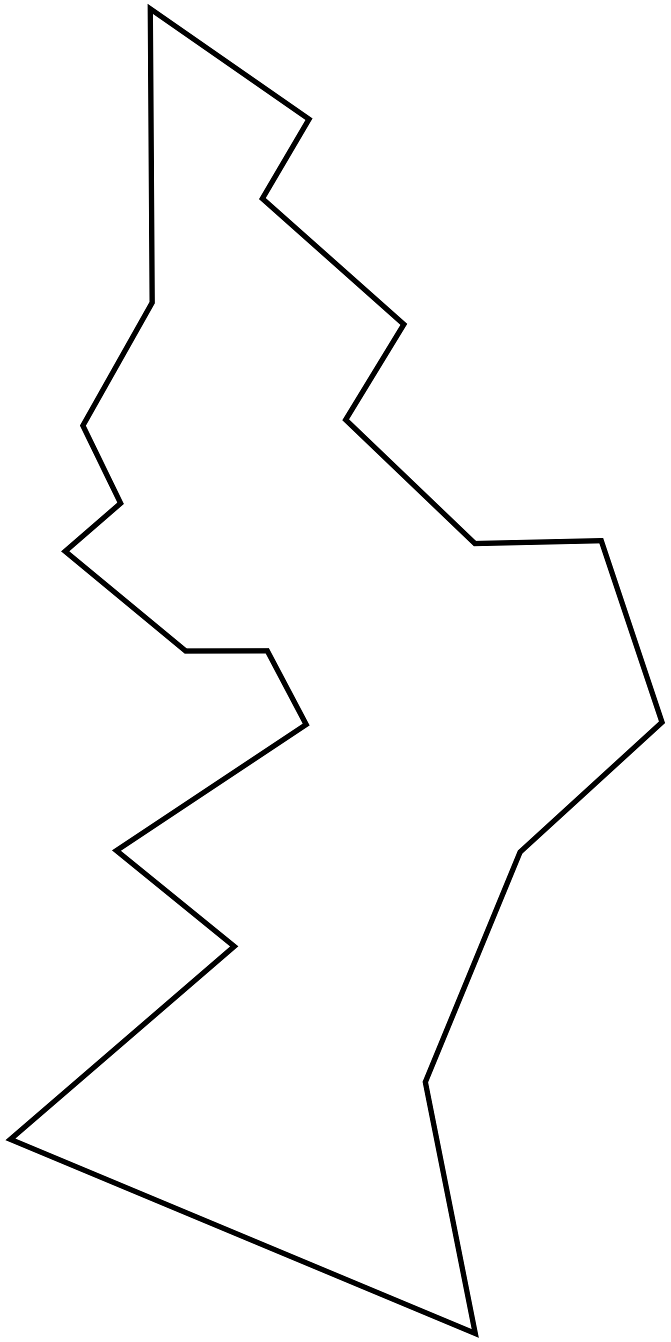


The vertices on each chain are sorted w.r.t. x-axis.

Monotone polygons



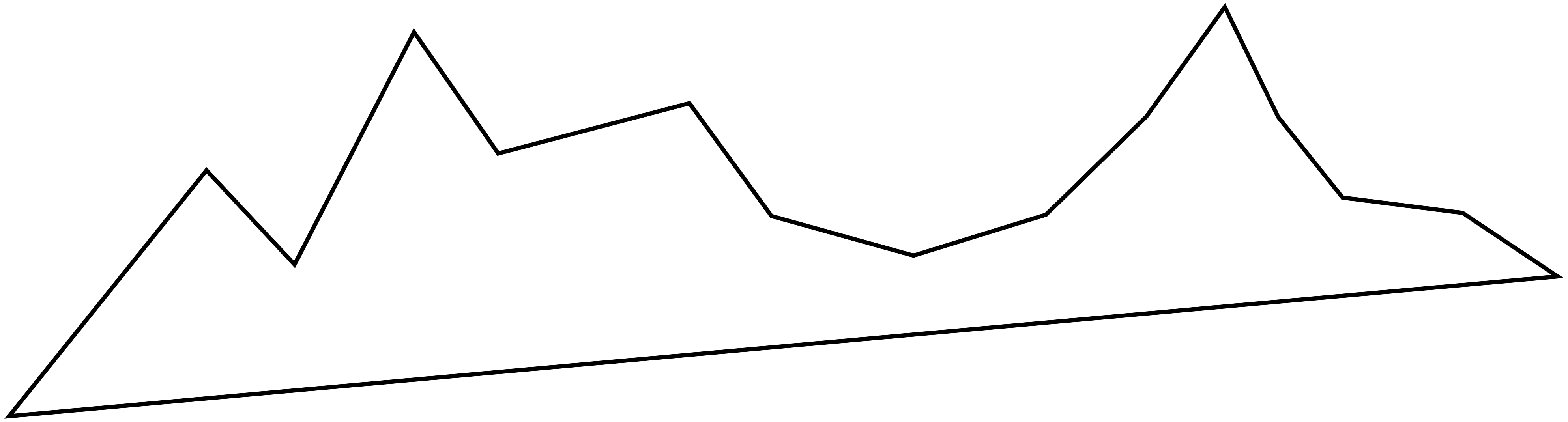
x-monotone



y-monotone

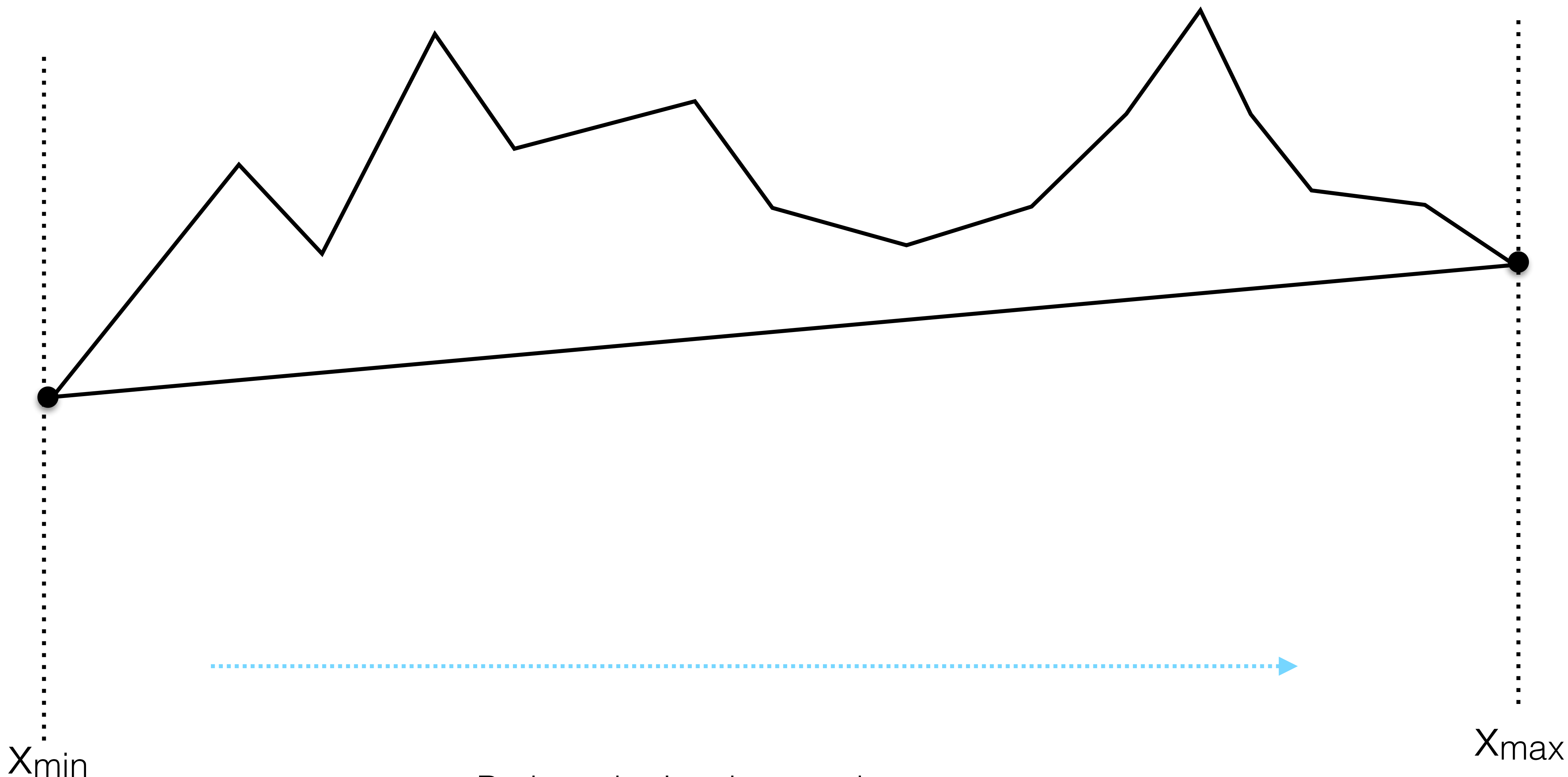
Monotone Mountains

A polygon is an **x-monotone mountain** if it is monotone and one of the two chains is a single segment.



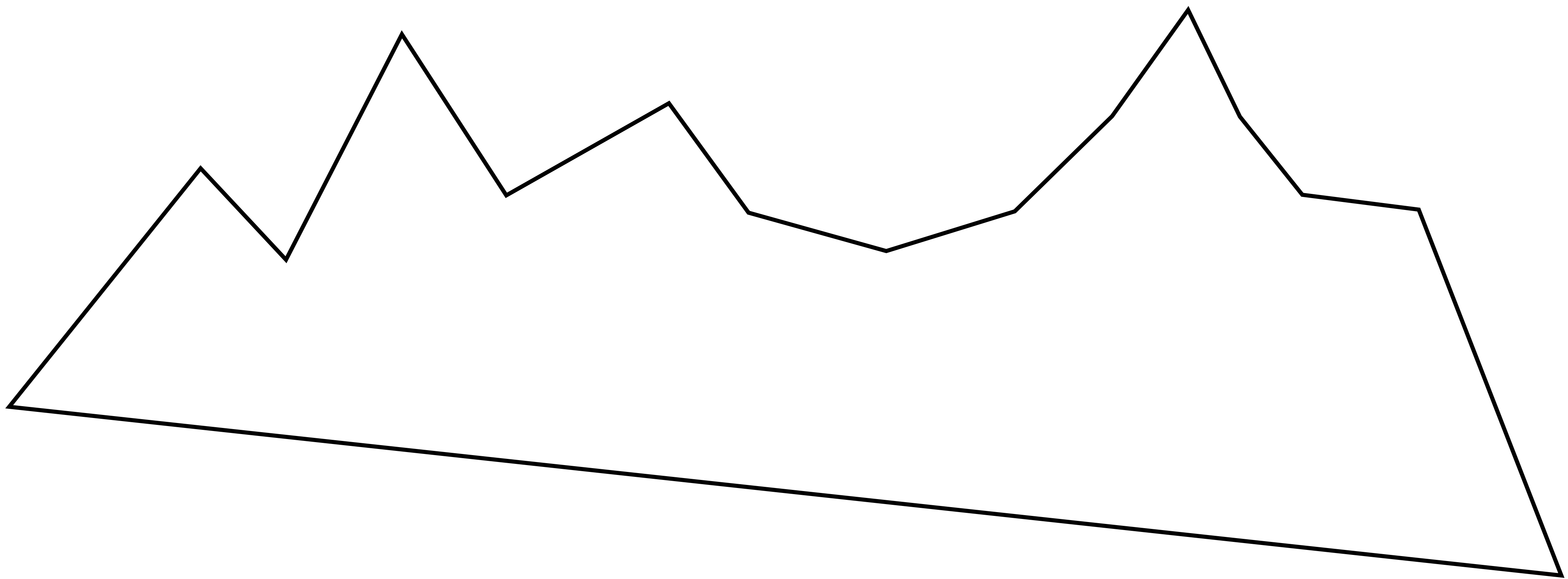
Monotone Mountains

A polygon is an **x-monotone mountain** if it is monotone and one of the two chains is a single segment.



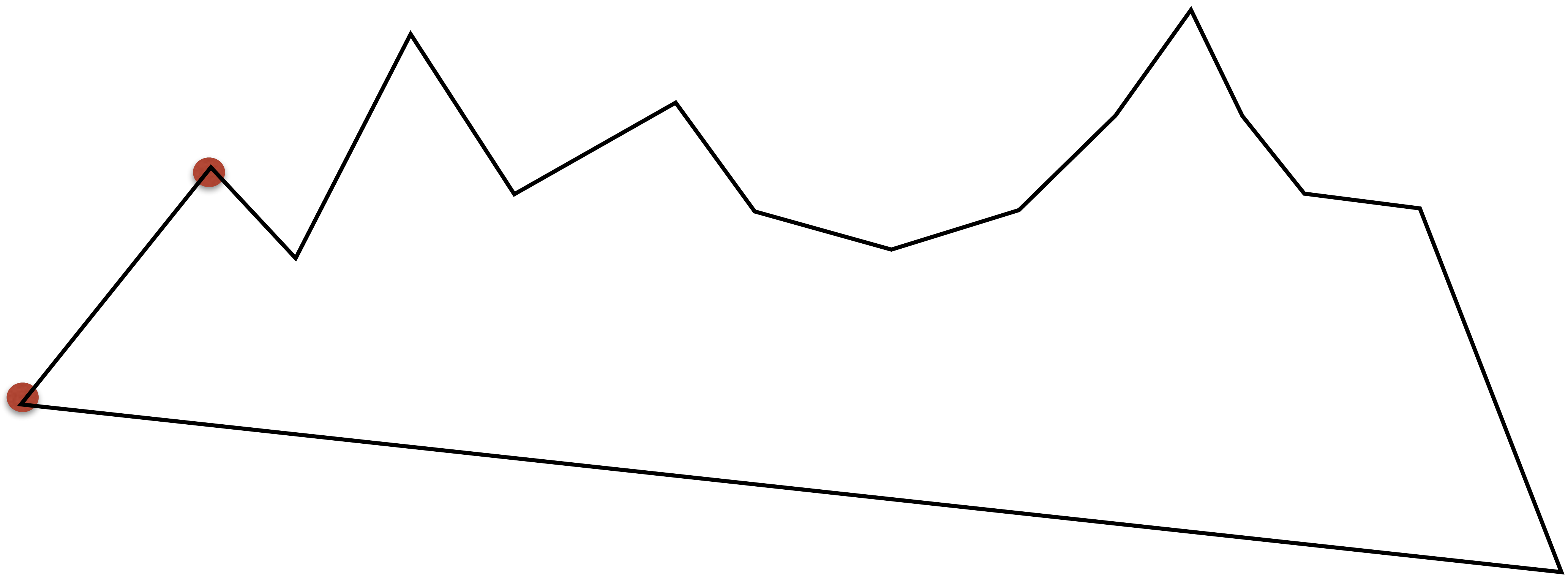
Both endpoints have to be convex.

Monotone mountains are easy to triangulate!

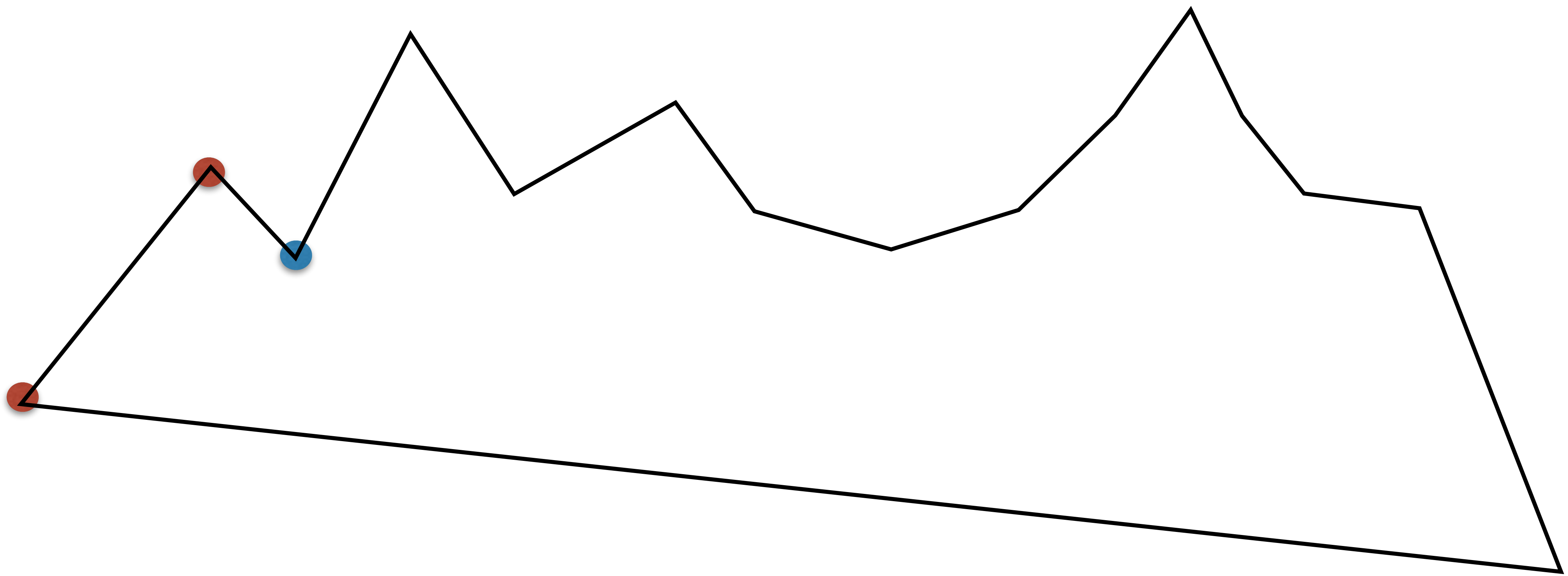


Class work: come up with an algorithm and analyze it.

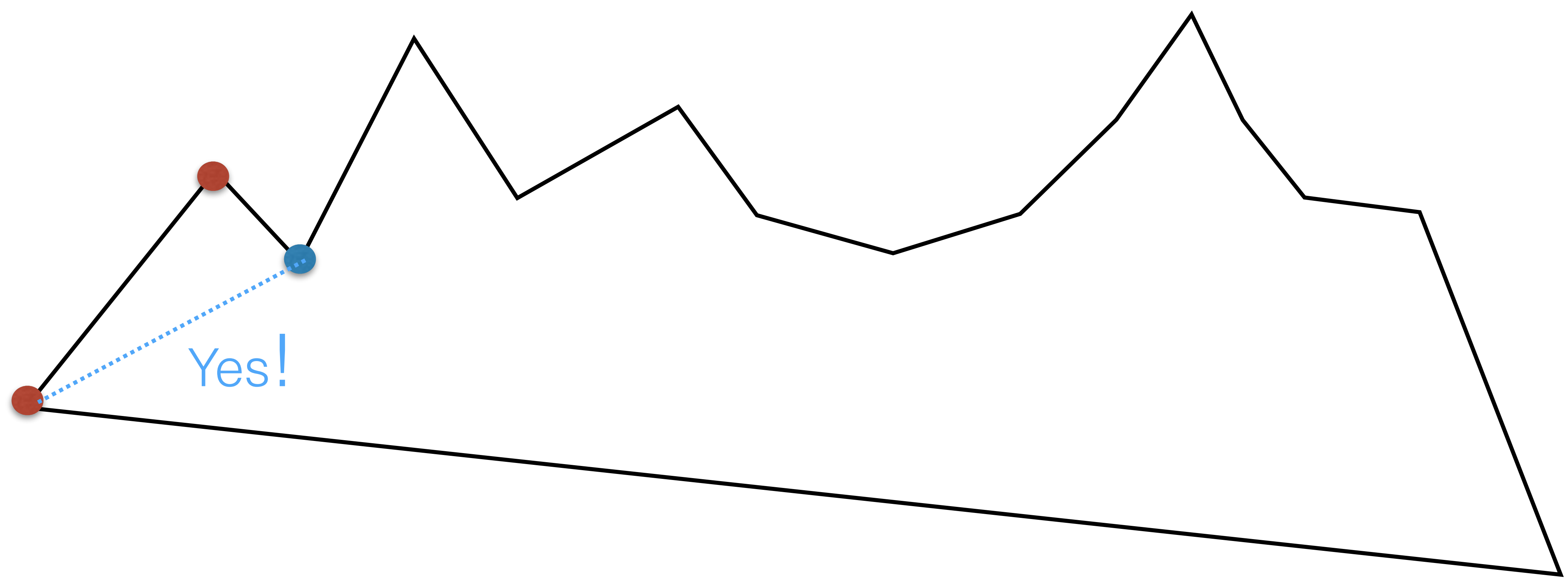
Monotone mountains are easy to triangulate!



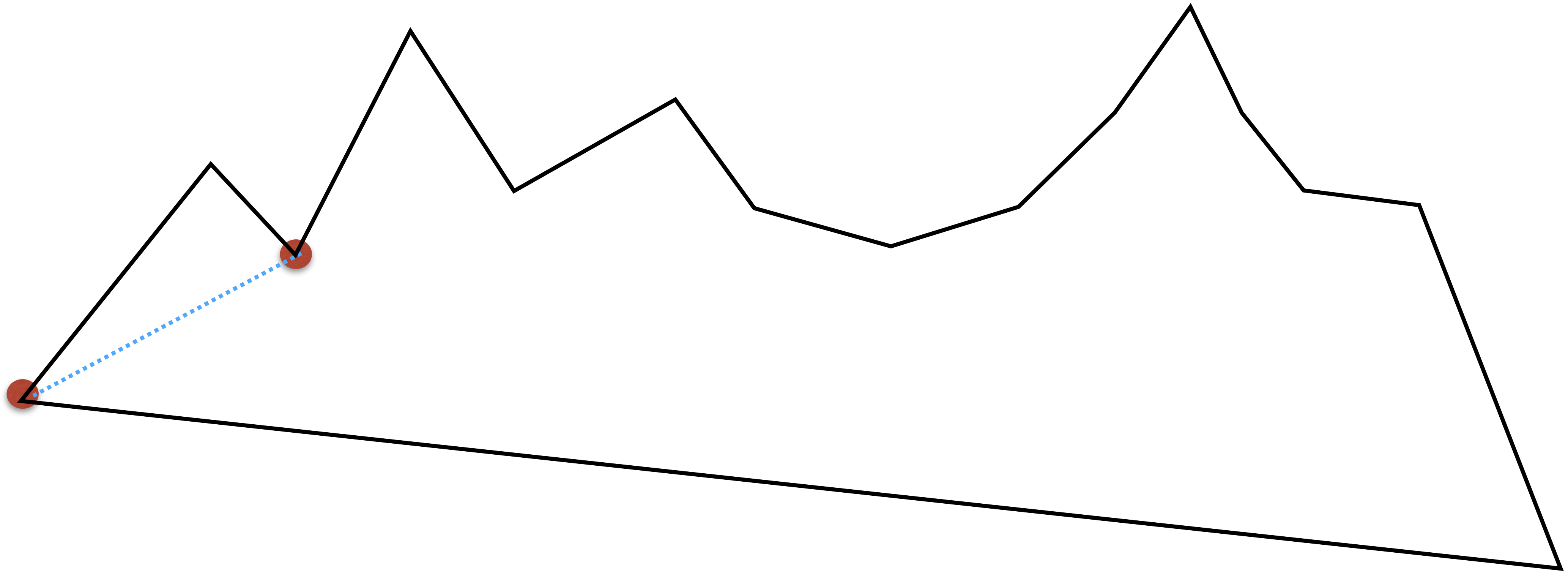
Monotone mountains are easy to triangulate!



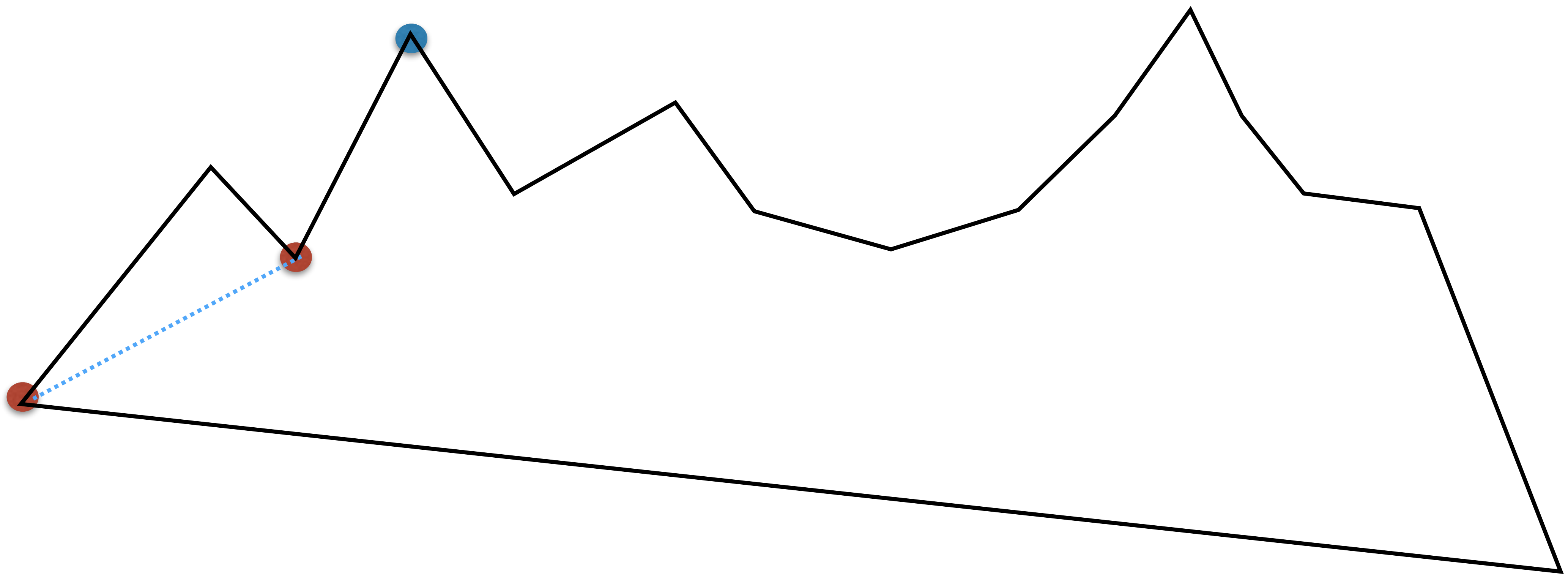
Monotone mountains are easy to triangulate!



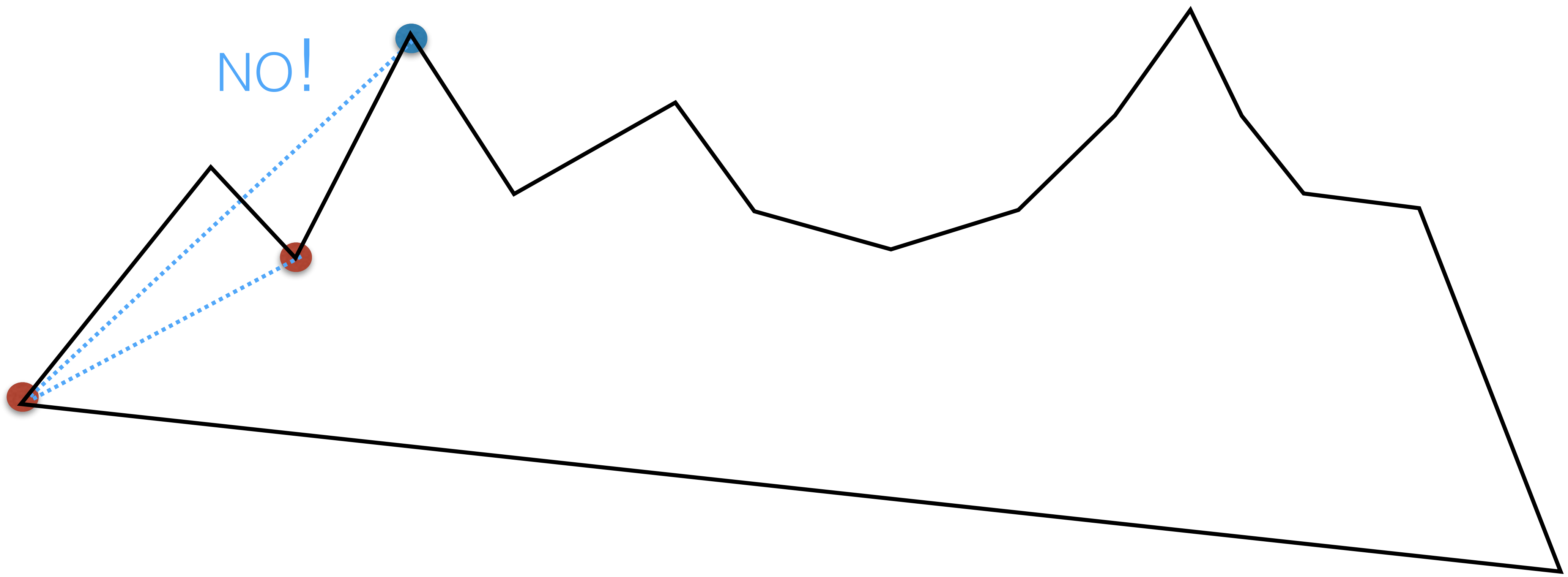
Monotone mountains are easy to triangulate!



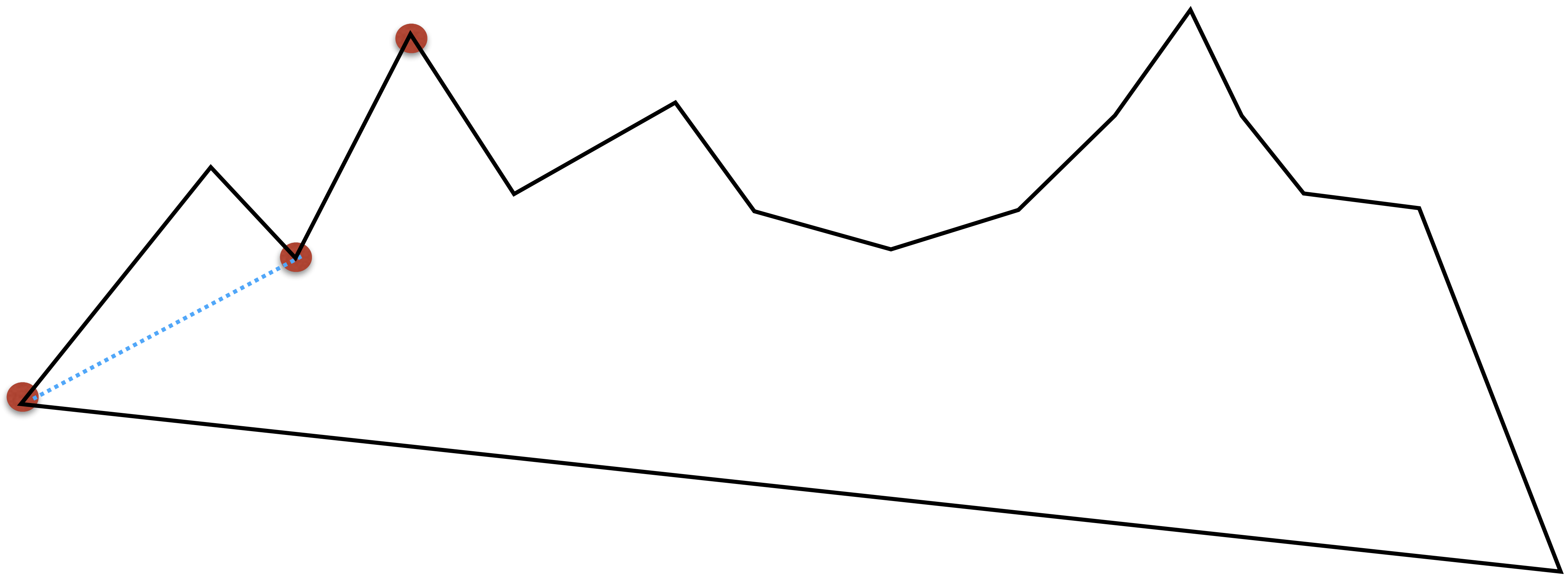
Monotone mountains are easy to triangulate!



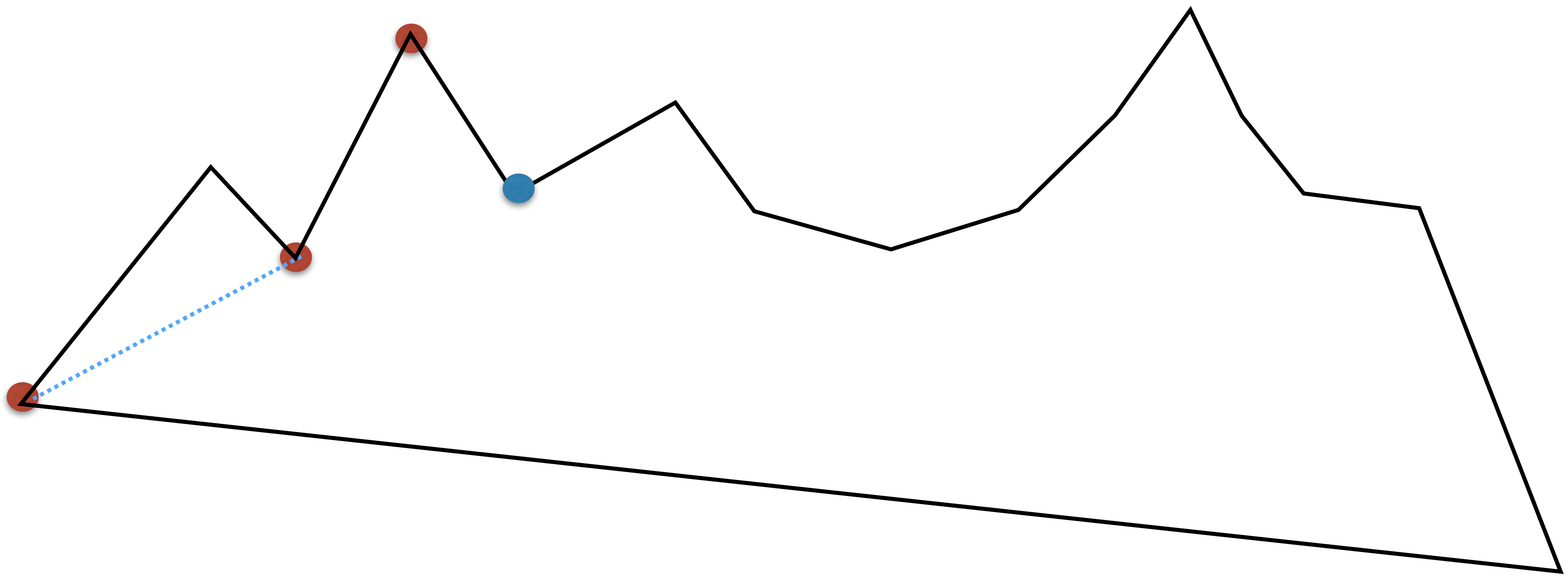
Monotone mountains are easy to triangulate!



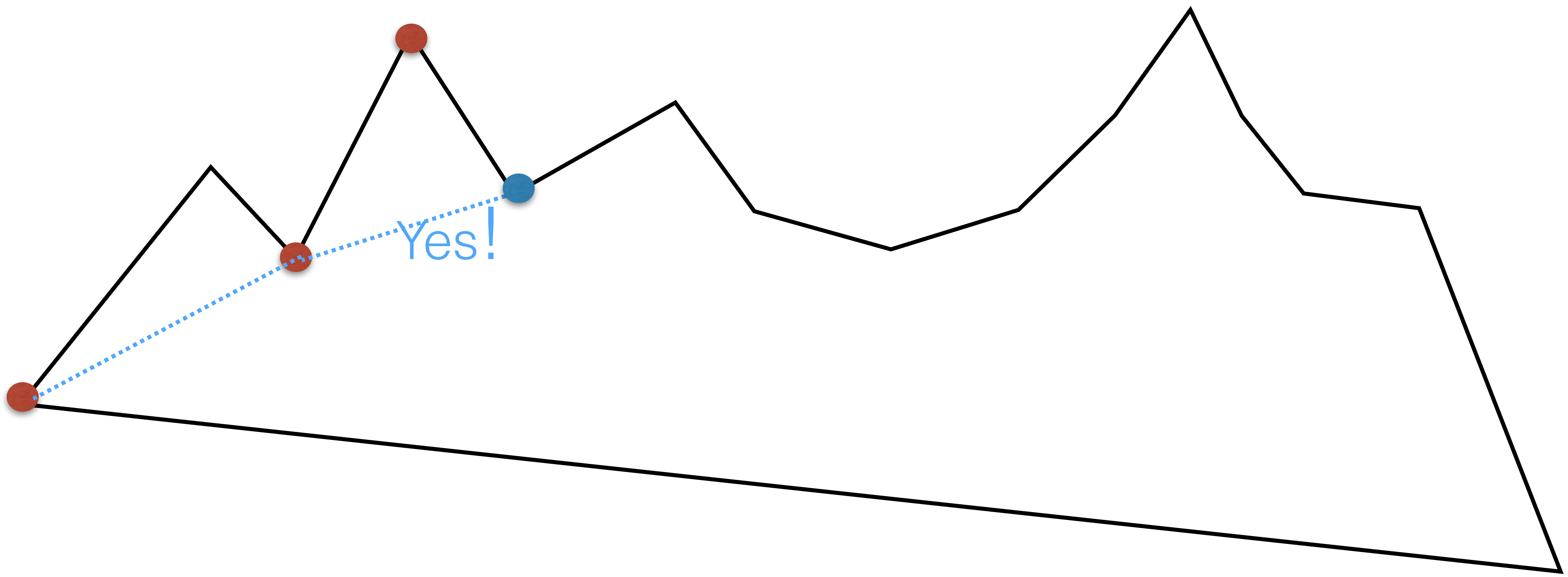
Monotone mountains are easy to triangulate!



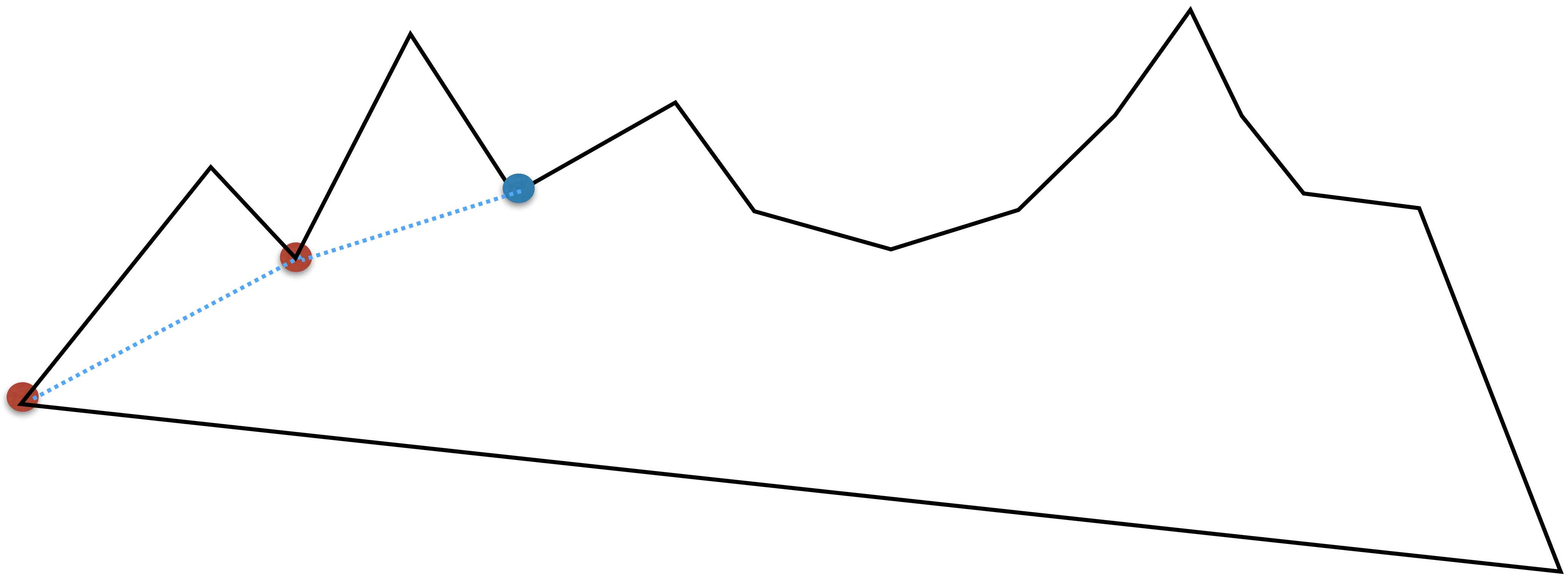
Monotone mountains are easy to triangulate!



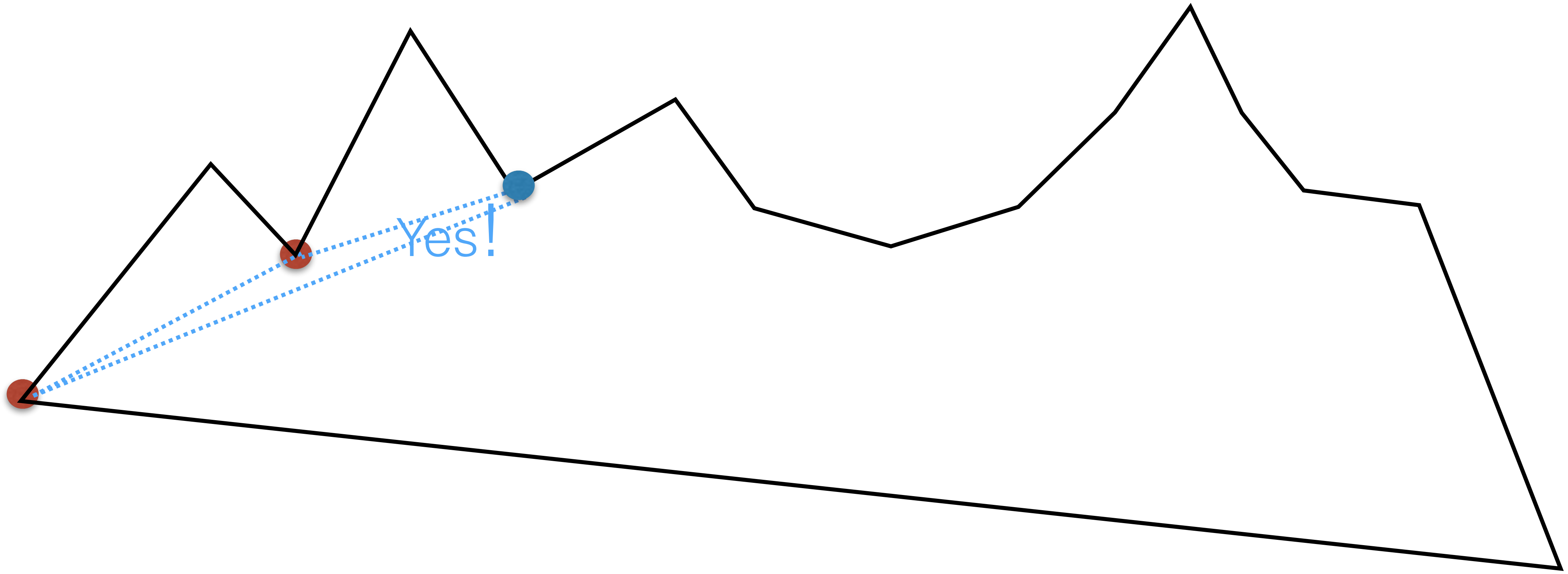
Monotone mountains are easy to triangulate!



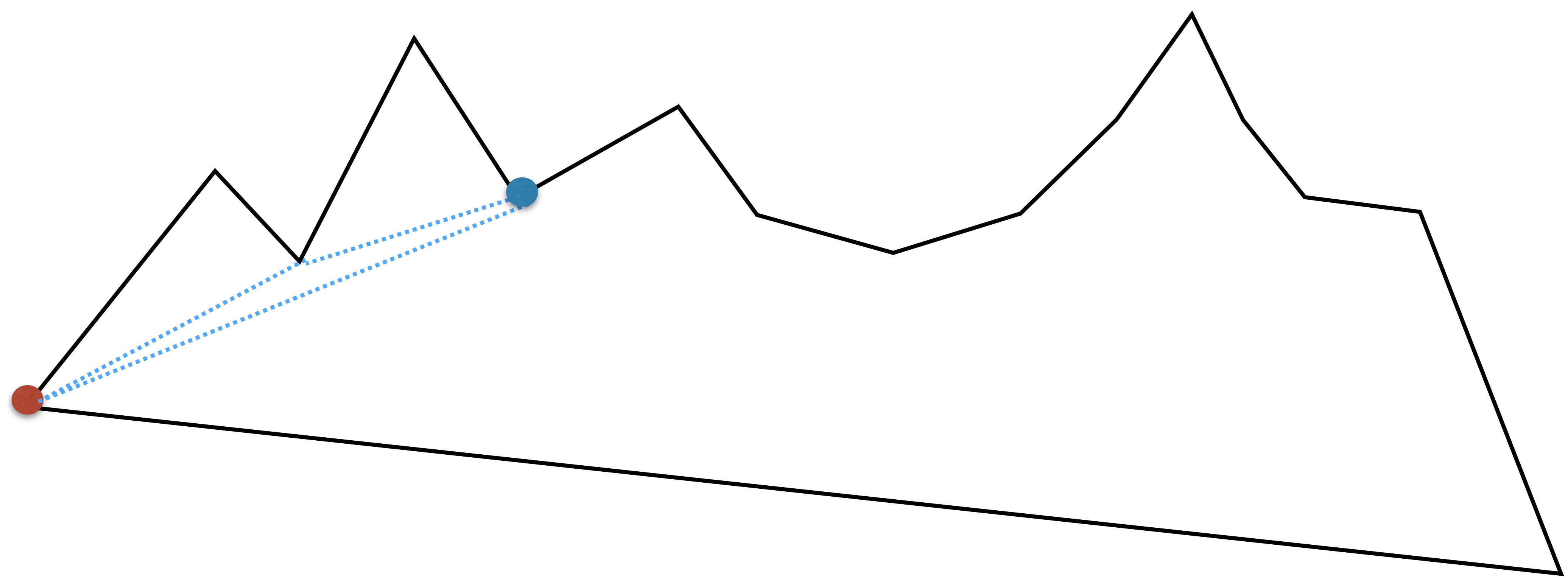
Monotone mountains are easy to triangulate!



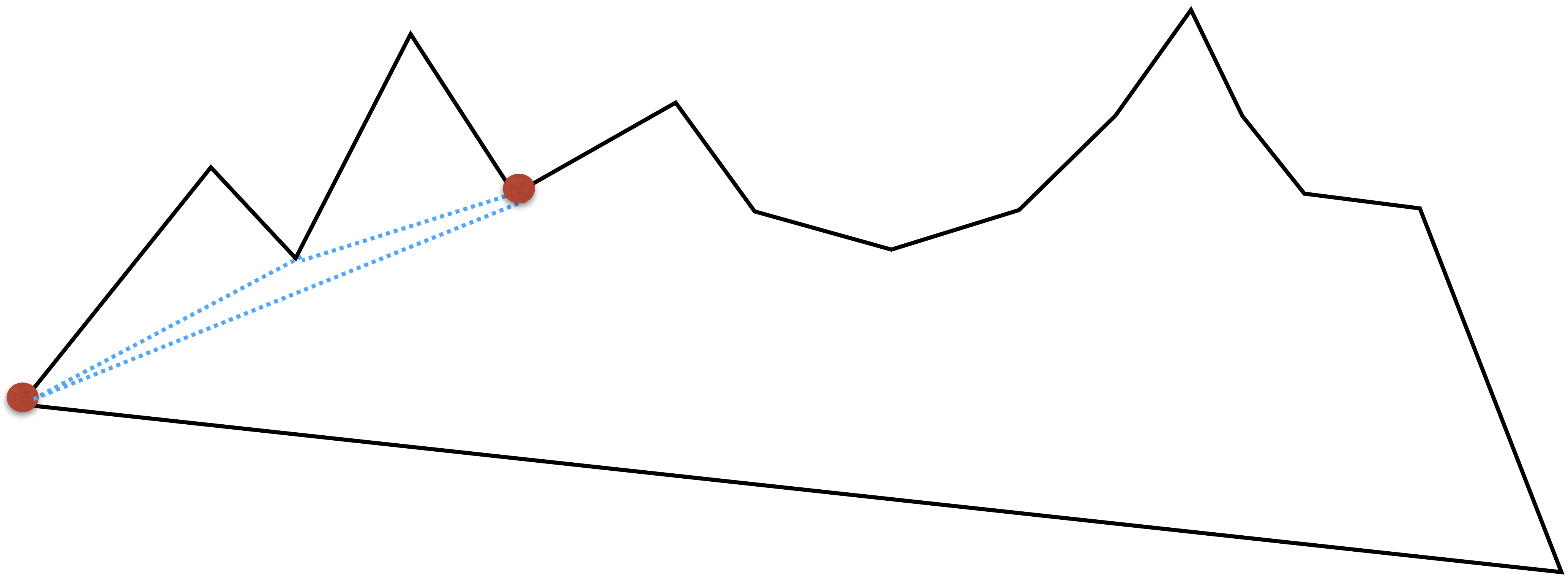
Monotone mountains are easy to triangulate!



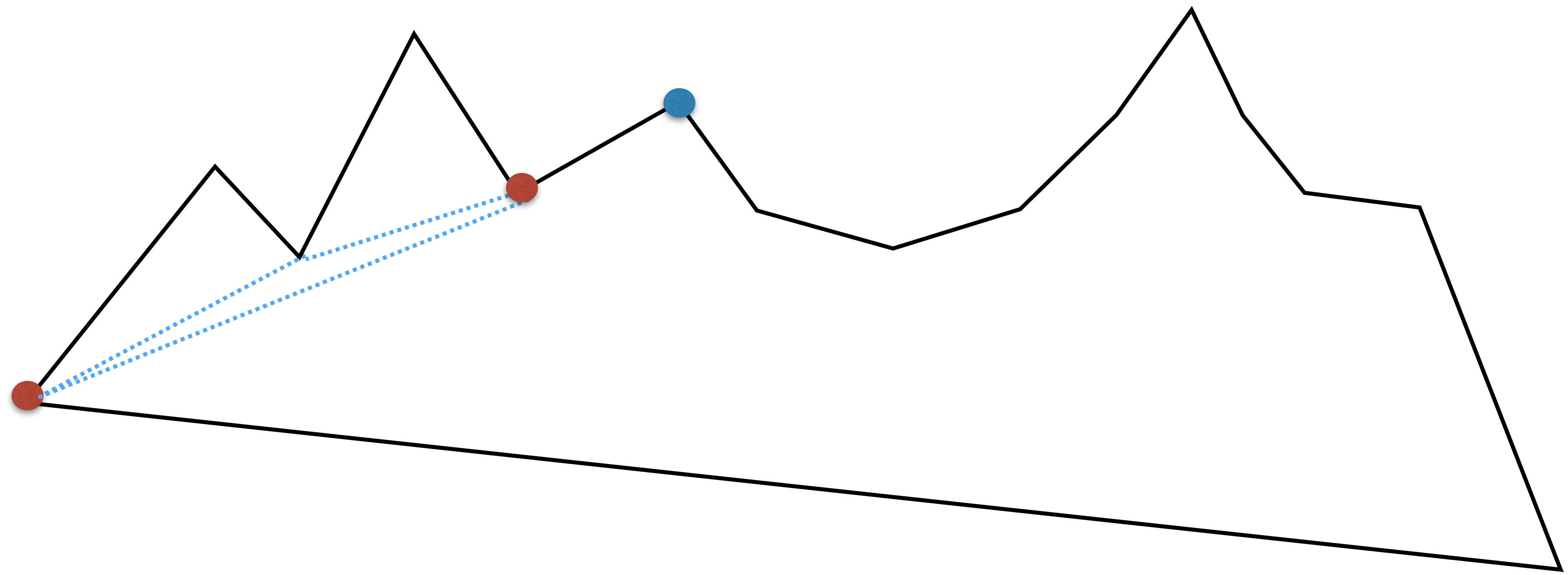
Monotone mountains are easy to triangulate!



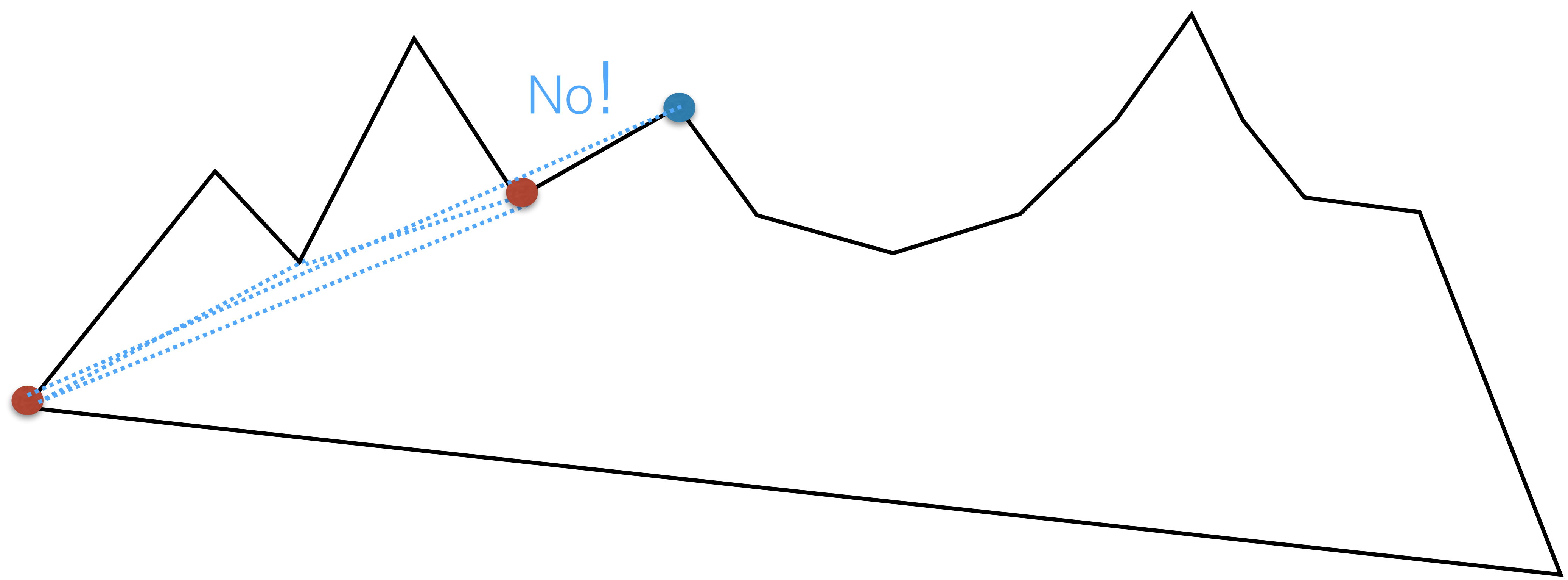
Monotone mountains are easy to triangulate!



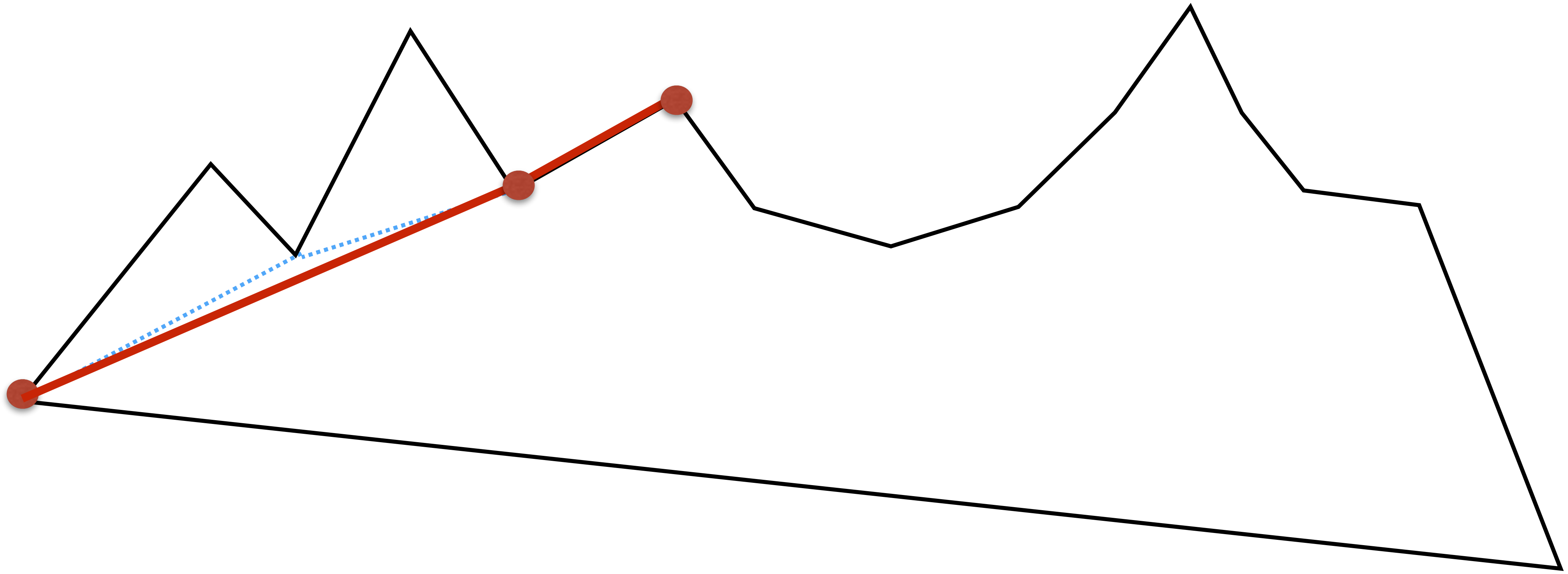
Monotone mountains are easy to triangulate!



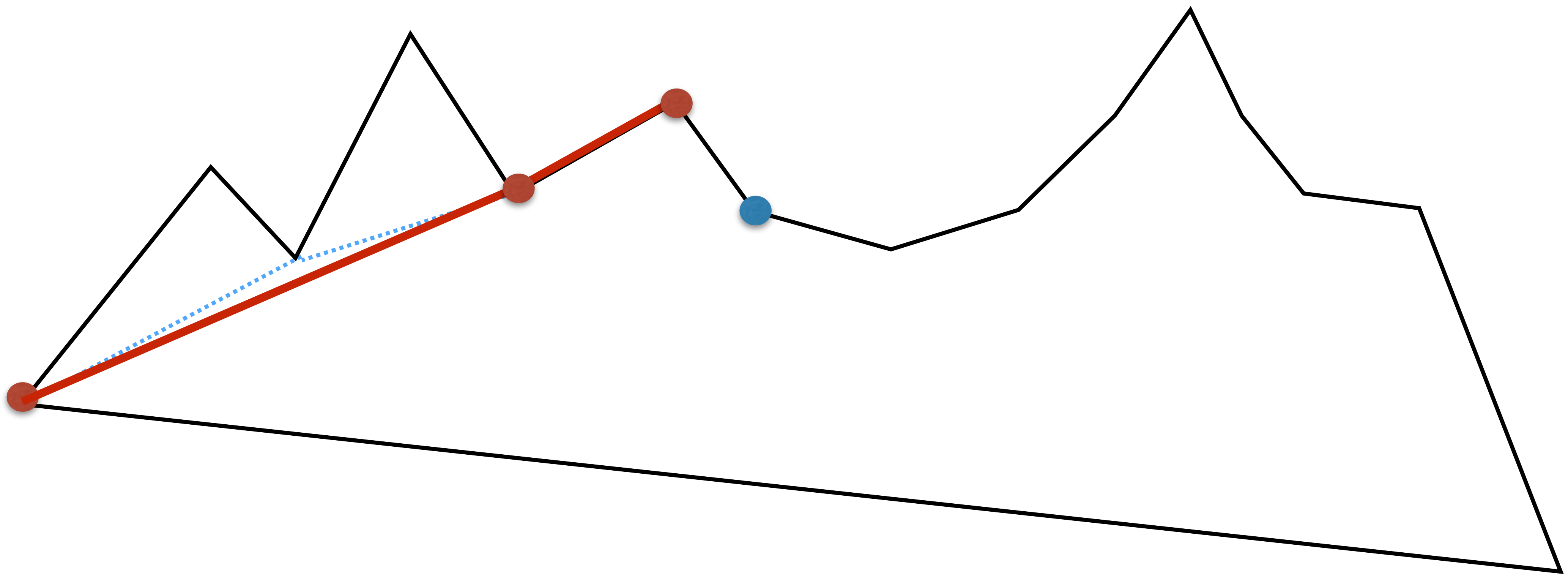
Monotone mountains are easy to triangulate!



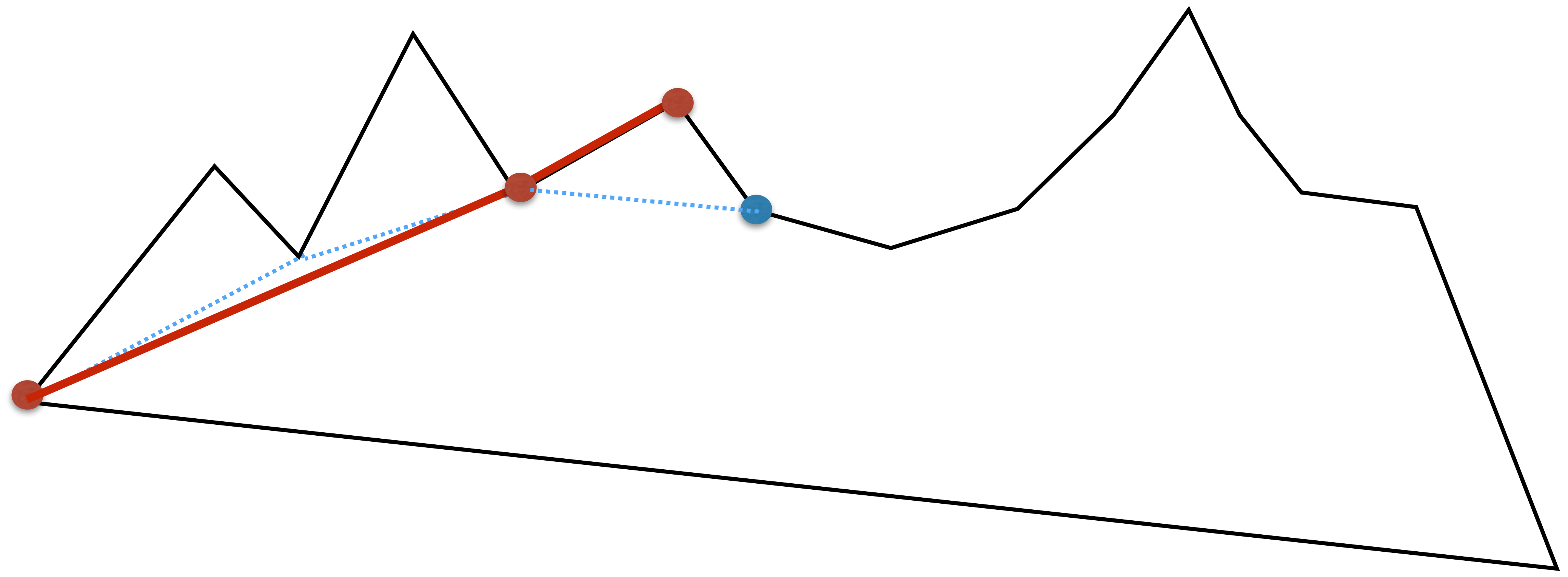
Monotone mountains are easy to triangulate!



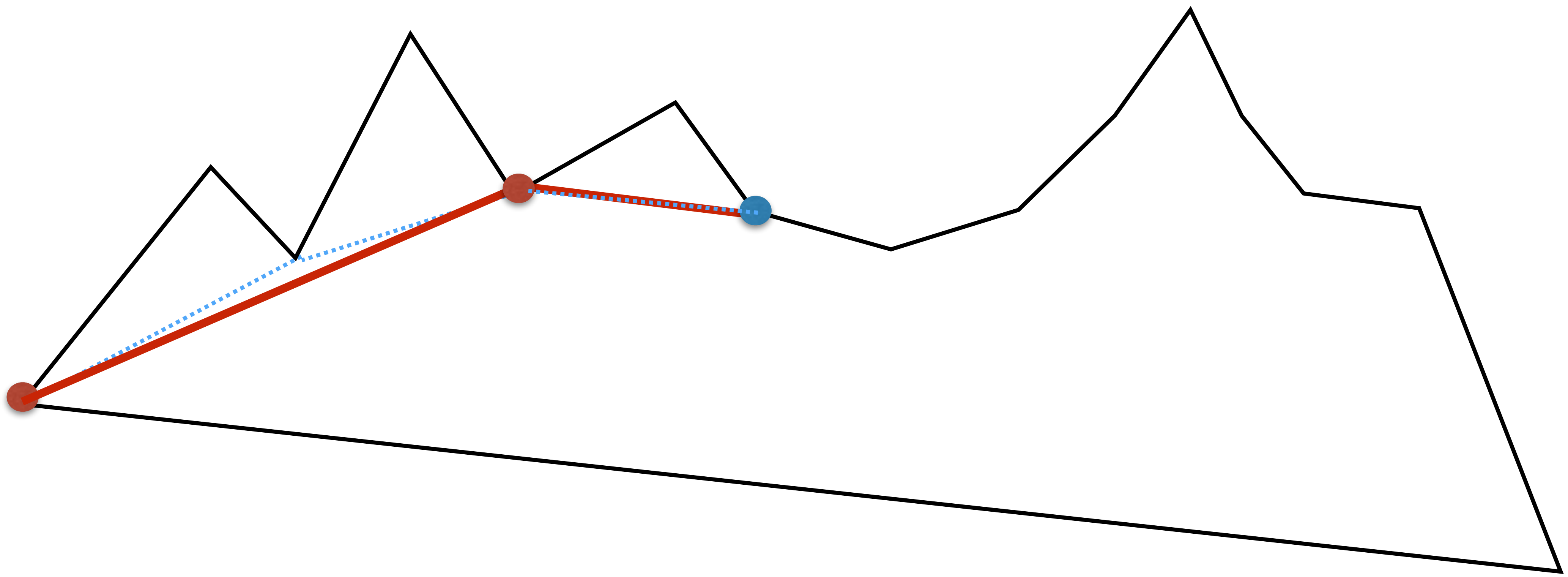
Monotone mountains are easy to triangulate!



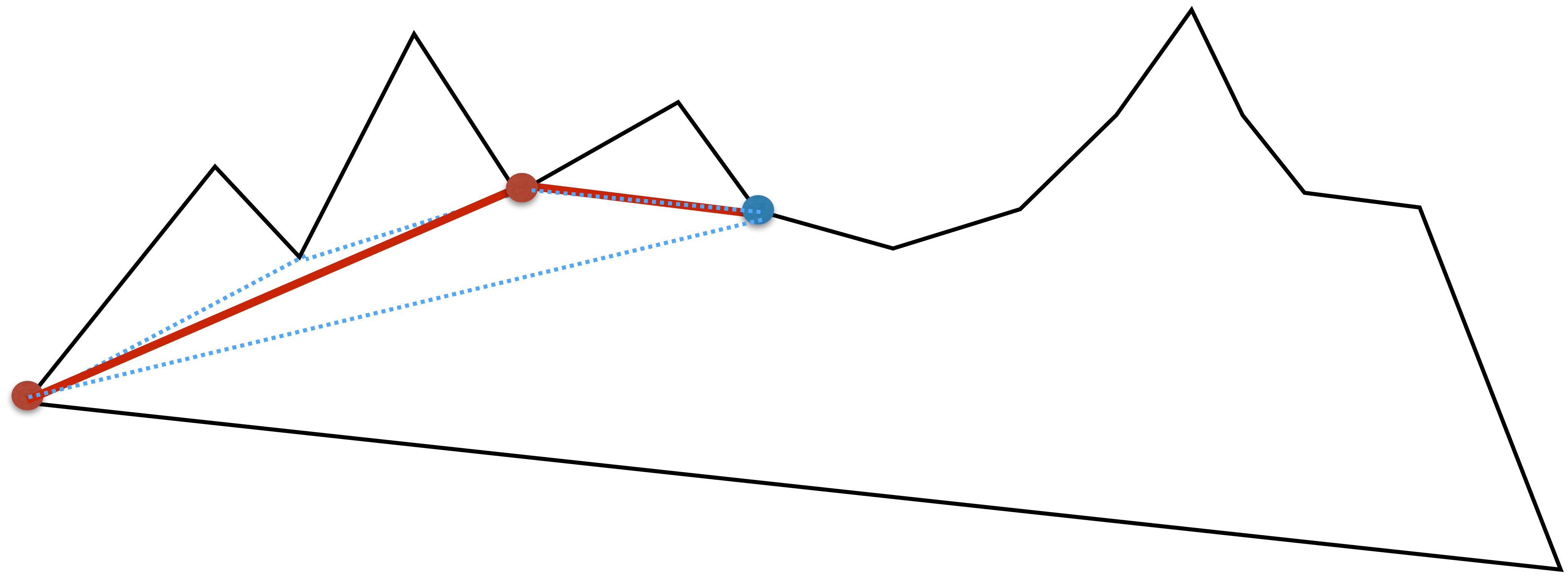
Monotone mountains are easy to triangulate!



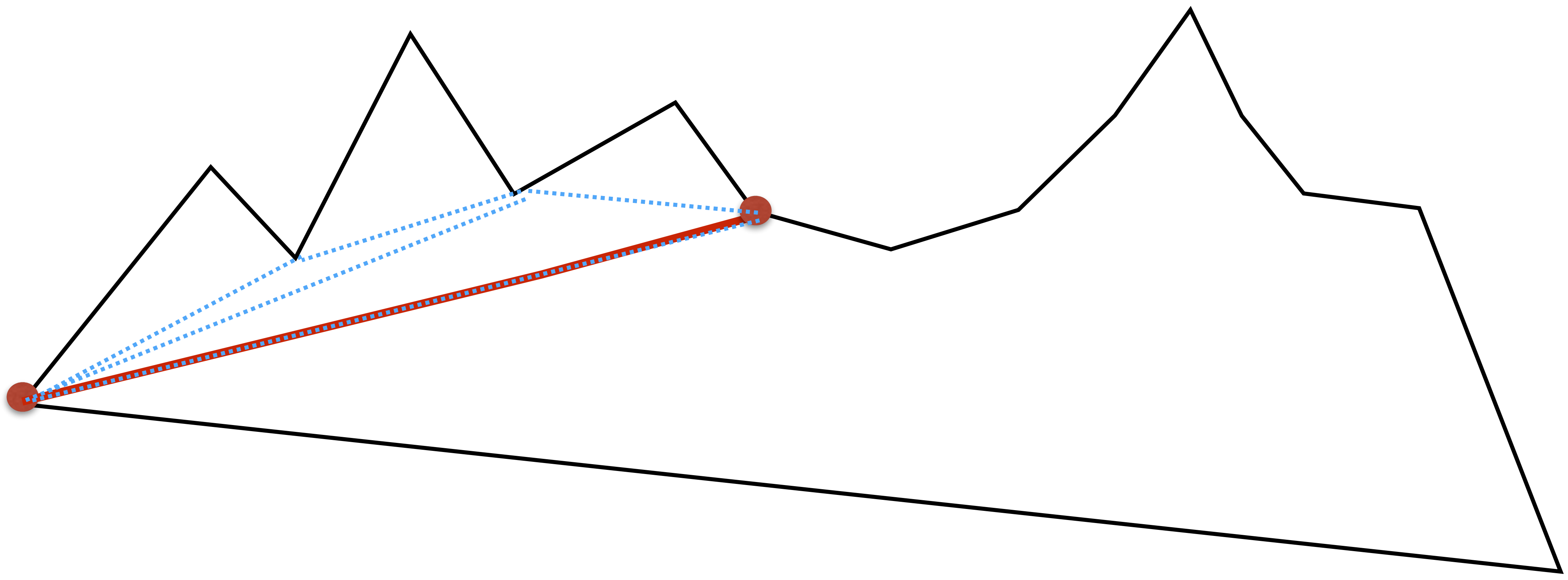
Monotone mountains are easy to triangulate!



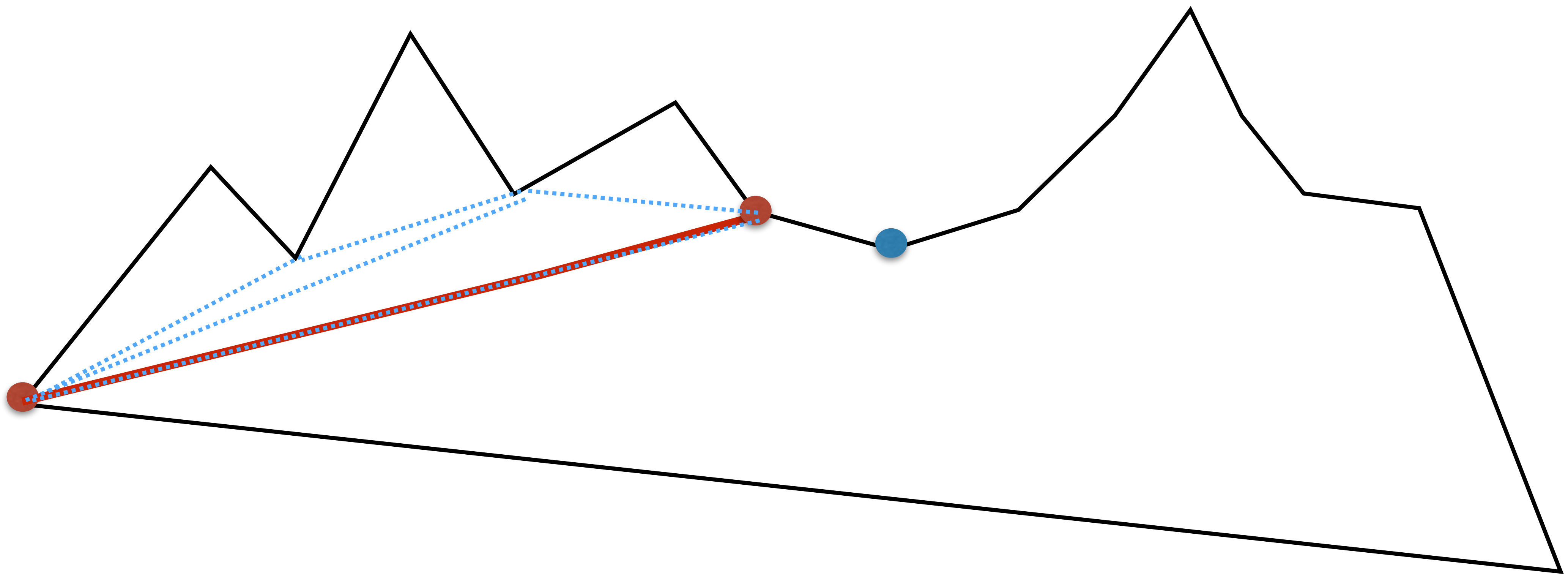
Monotone mountains are easy to triangulate!



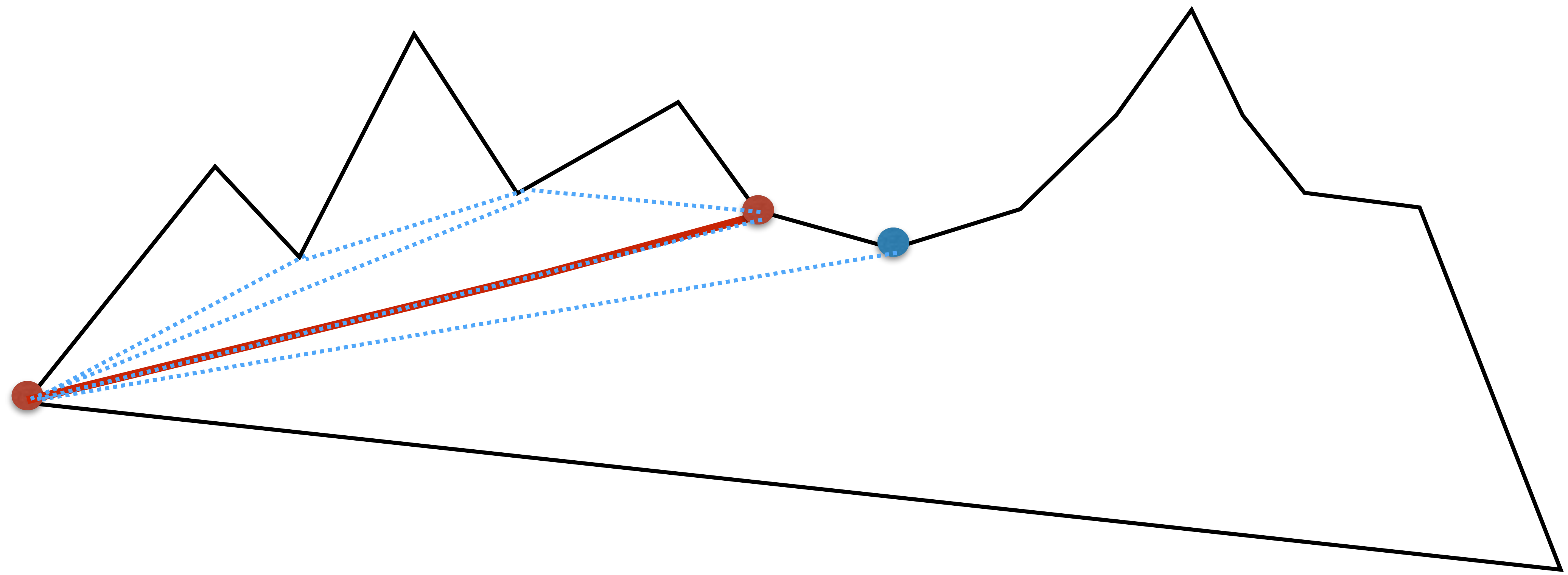
Monotone mountains are easy to triangulate!



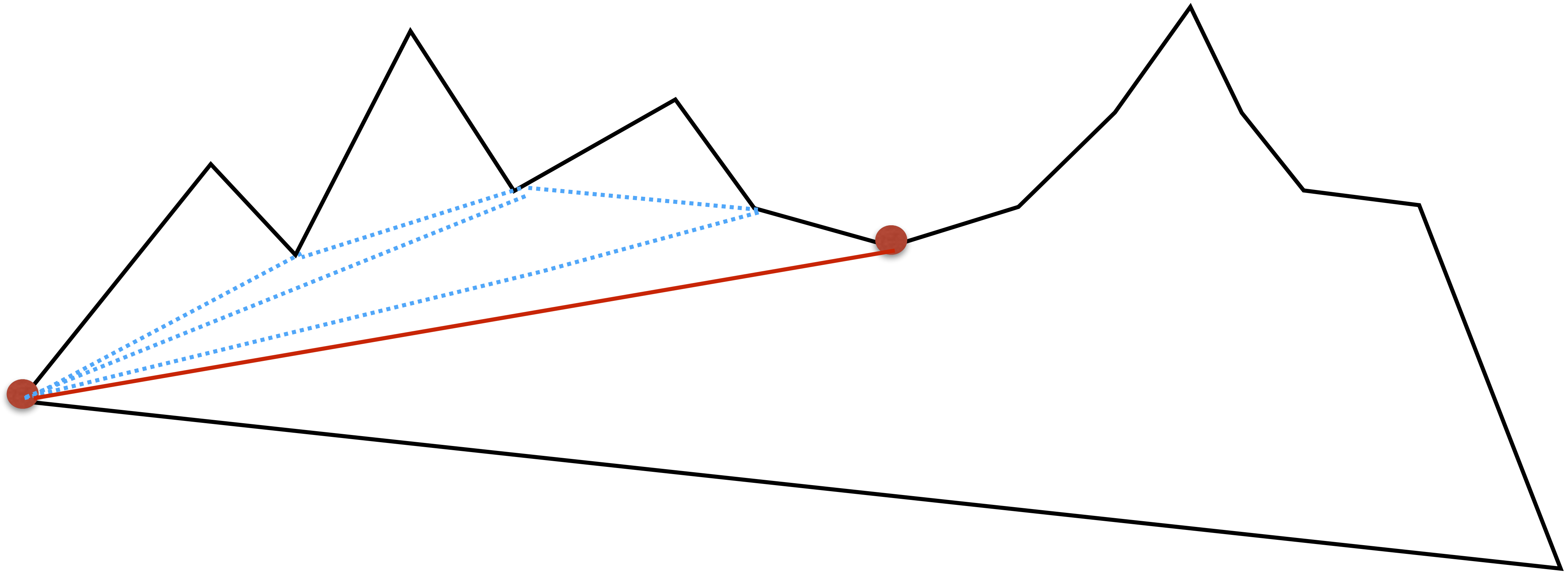
Monotone mountains are easy to triangulate!



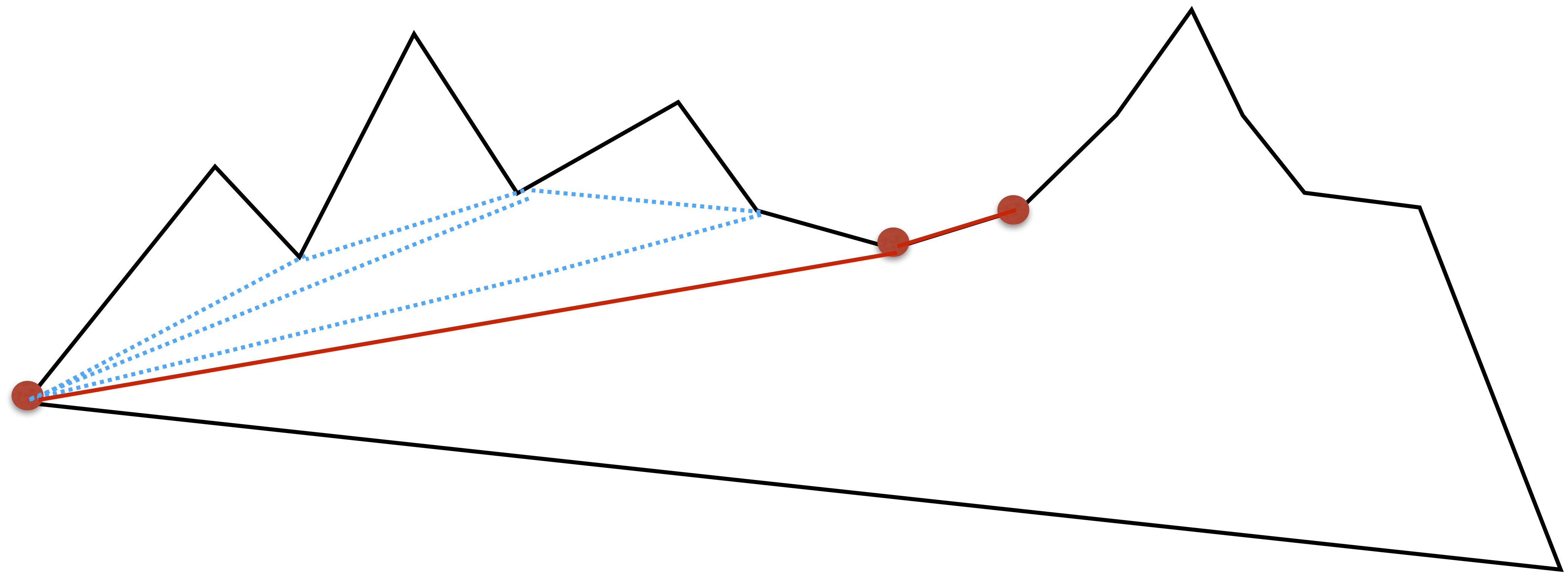
Monotone mountains are easy to triangulate!



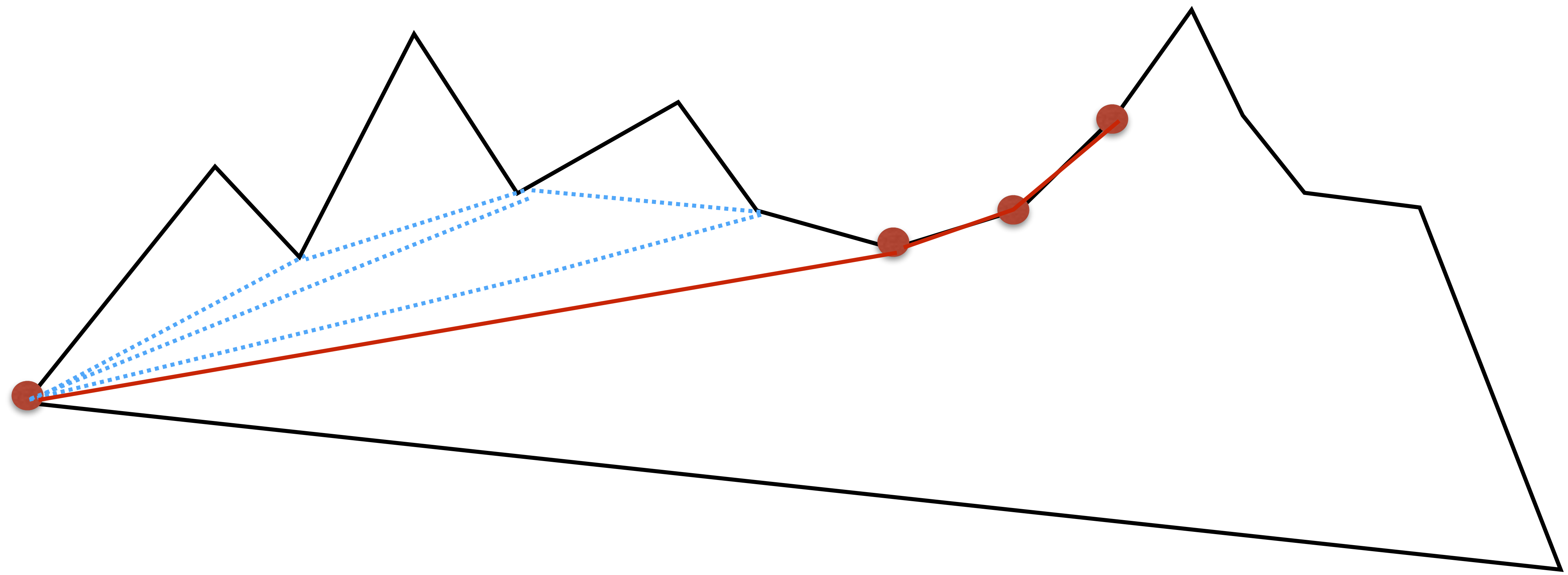
Monotone mountains are easy to triangulate!



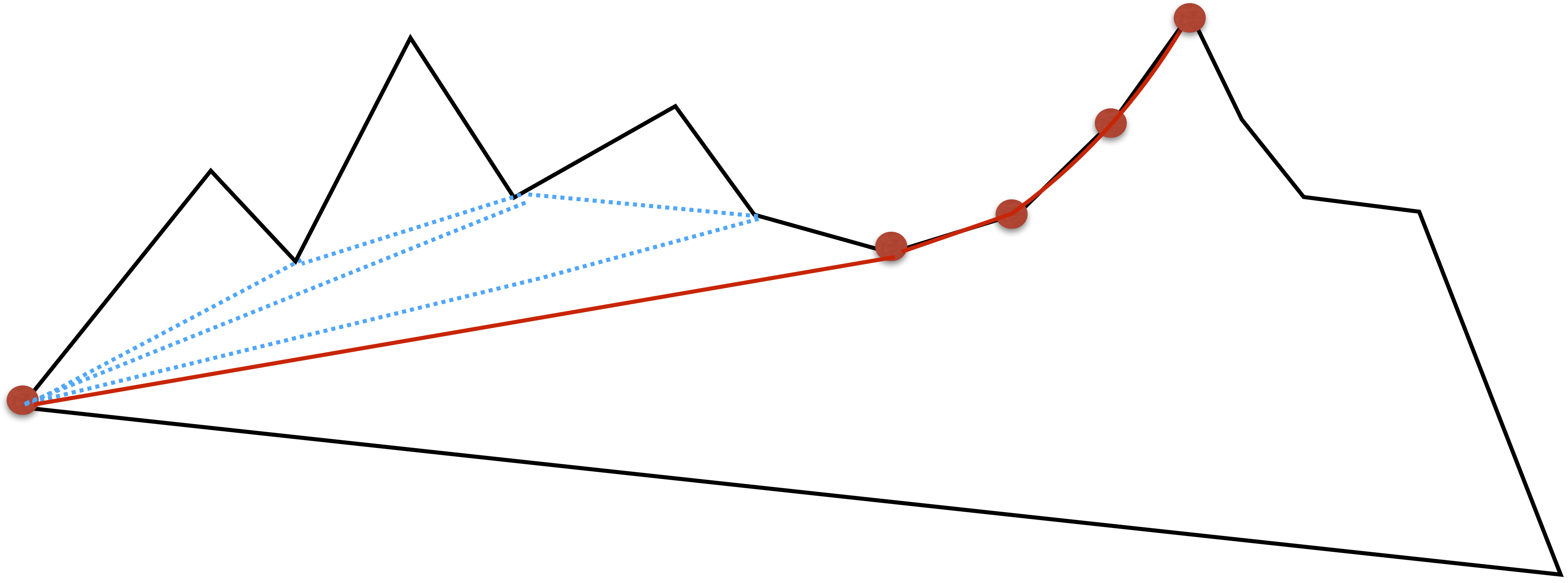
Monotone mountains are easy to triangulate!



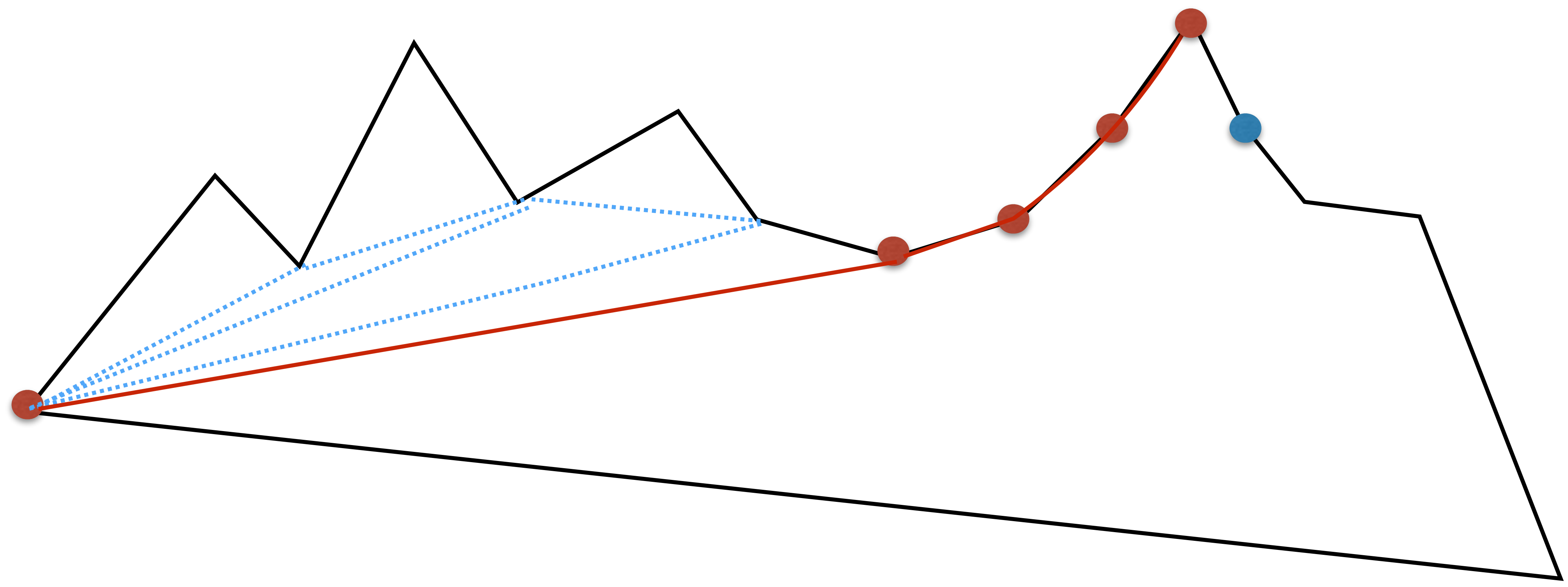
Monotone mountains are easy to triangulate!



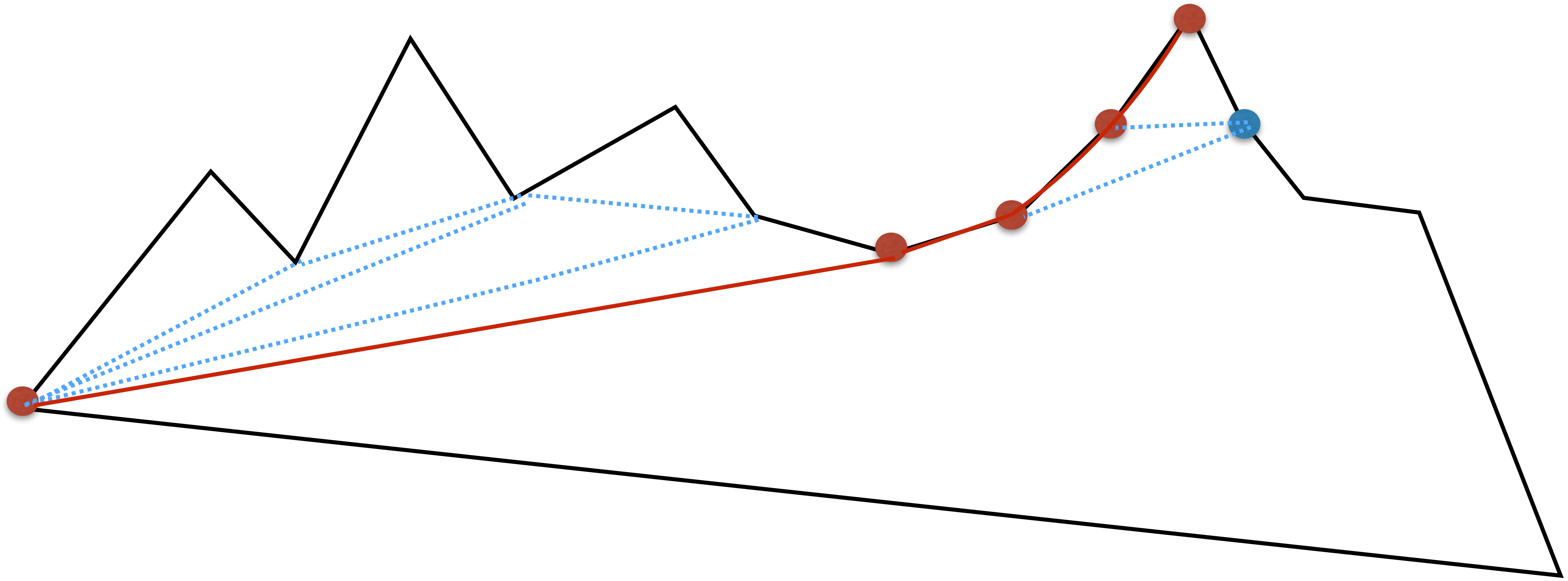
Monotone mountains are easy to triangulate!



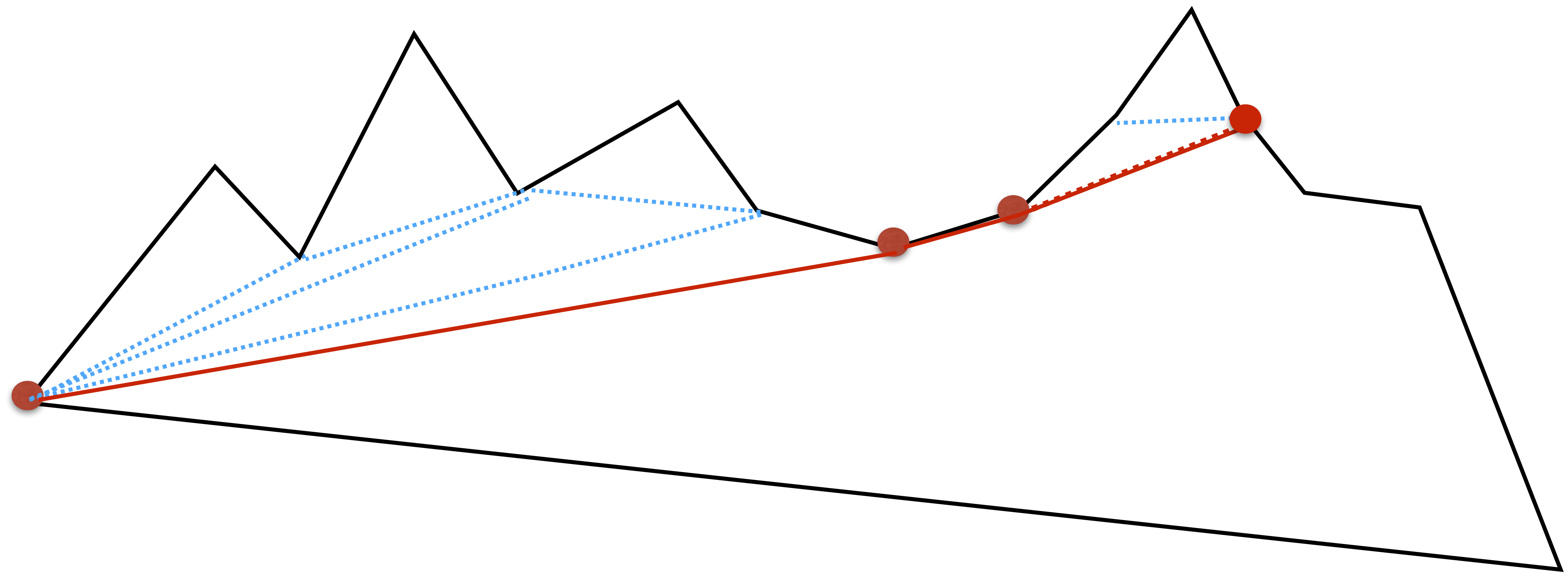
Monotone mountains are easy to triangulate!



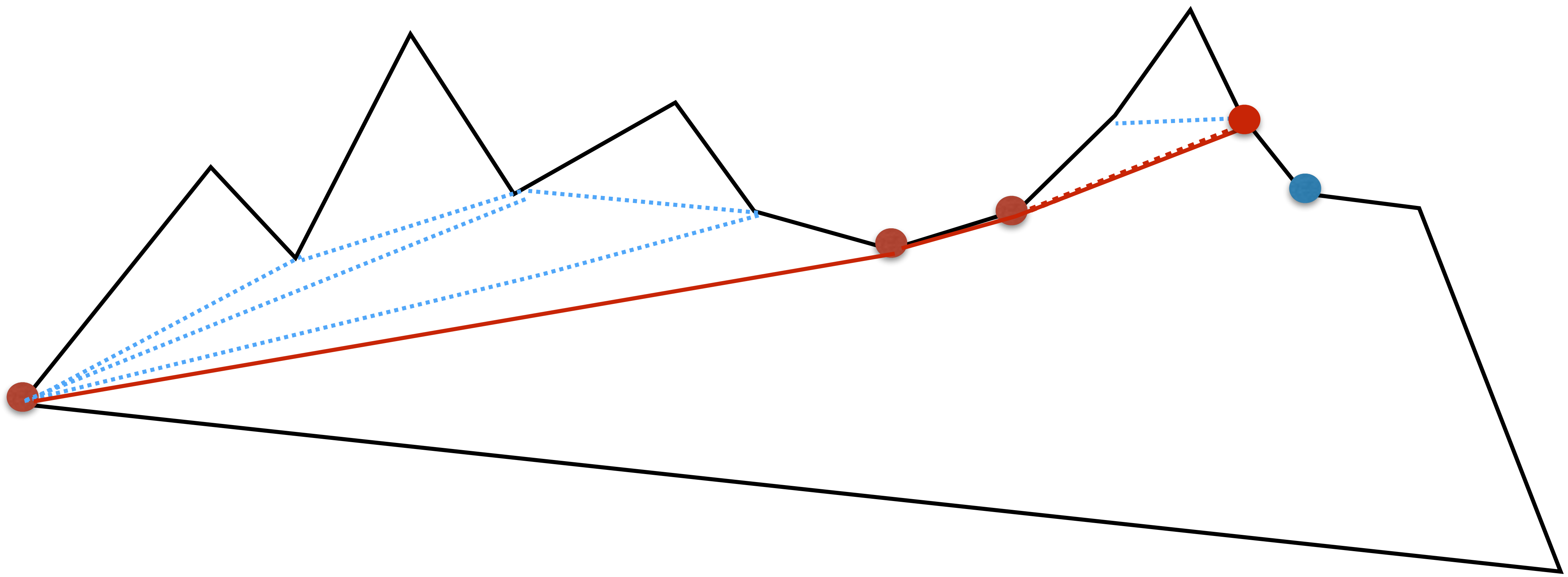
Monotone mountains are easy to triangulate!



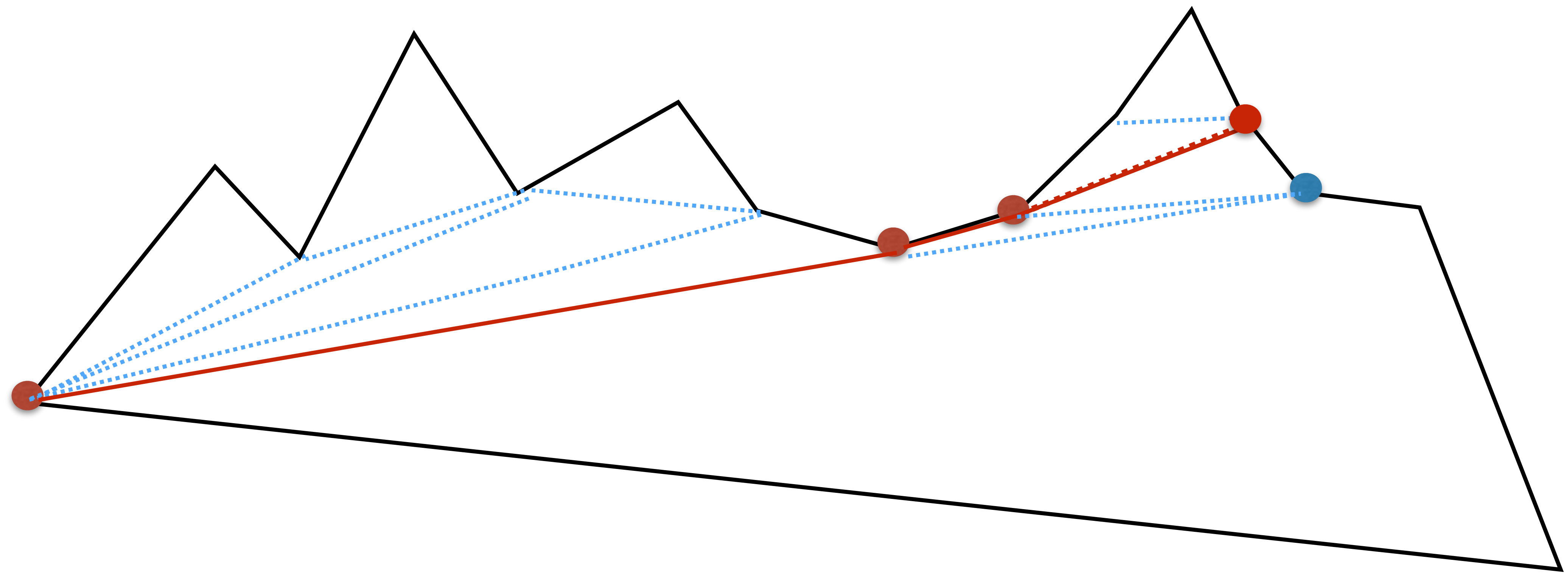
Monotone mountains are easy to triangulate!



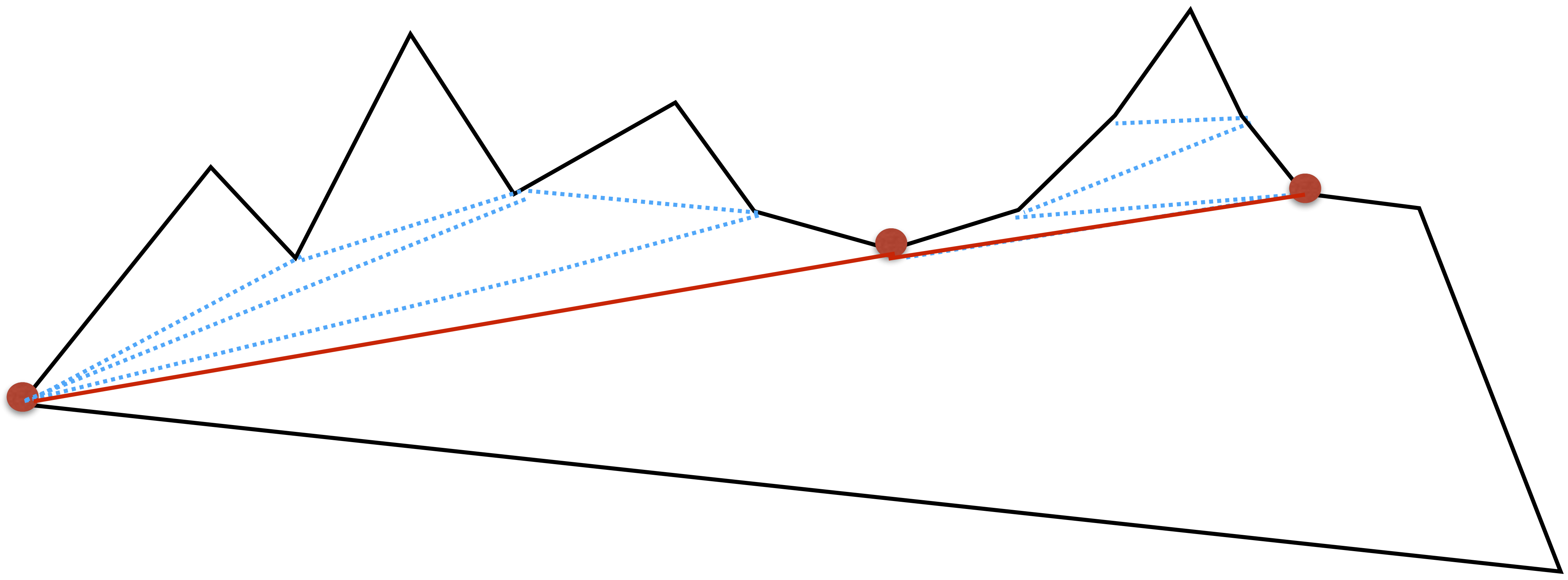
Monotone mountains are easy to triangulate!



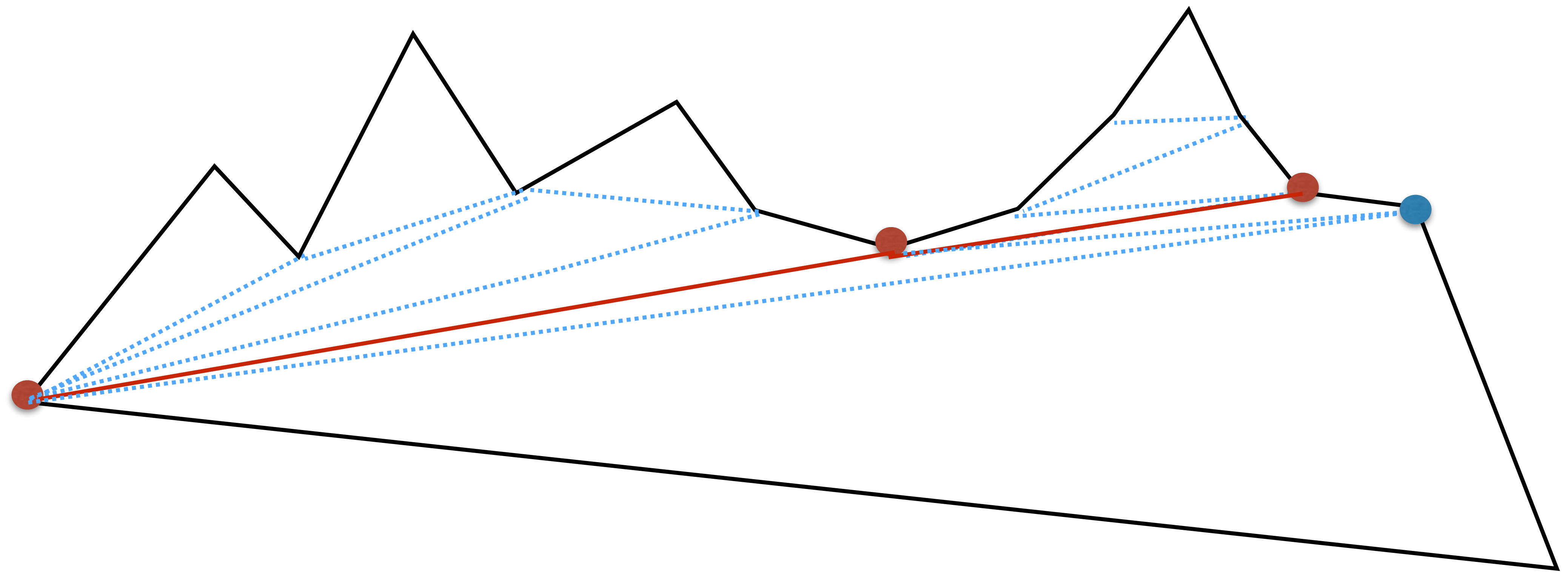
Monotone mountains are easy to triangulate!



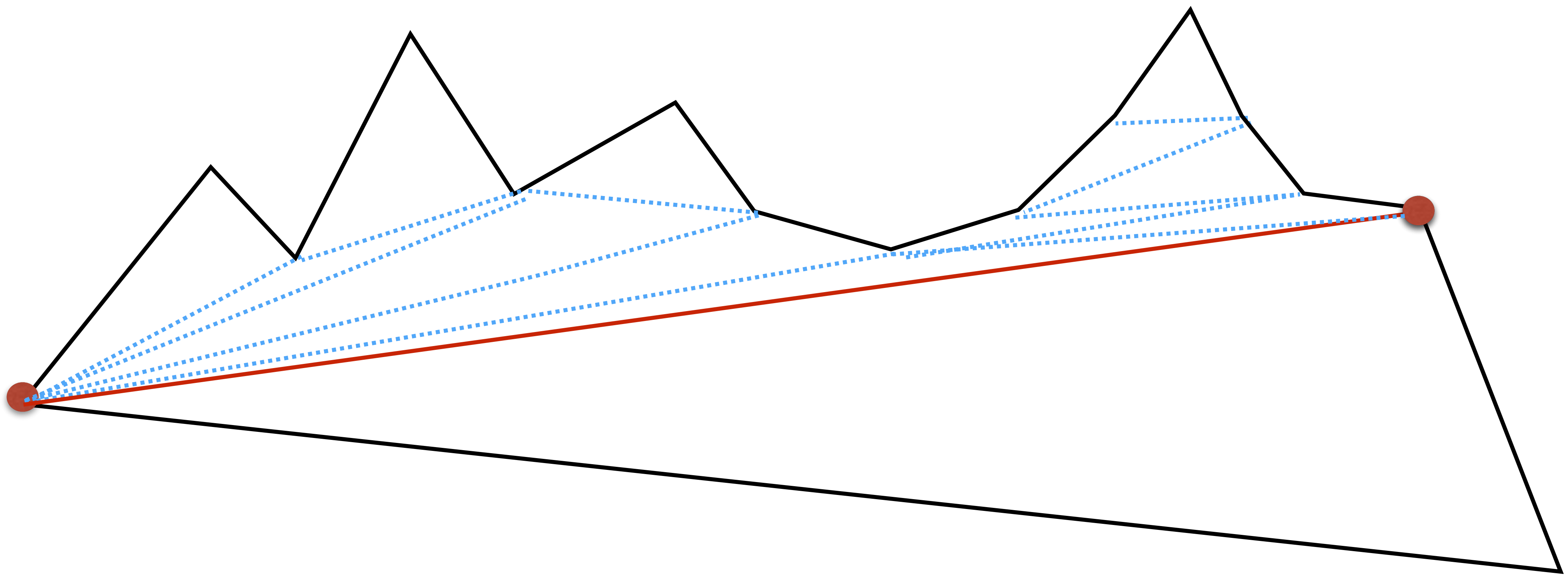
Monotone mountains are easy to triangulate!



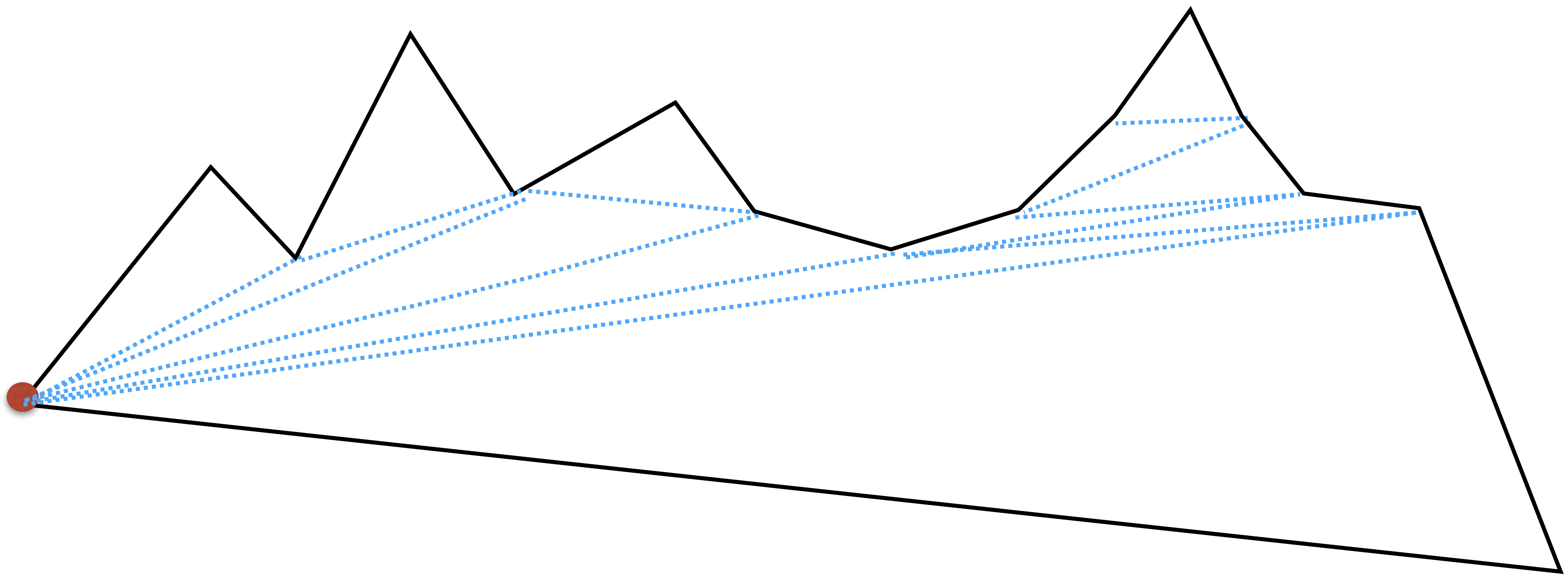
Monotone mountains are easy to triangulate!



Monotone mountains are easy to triangulate!



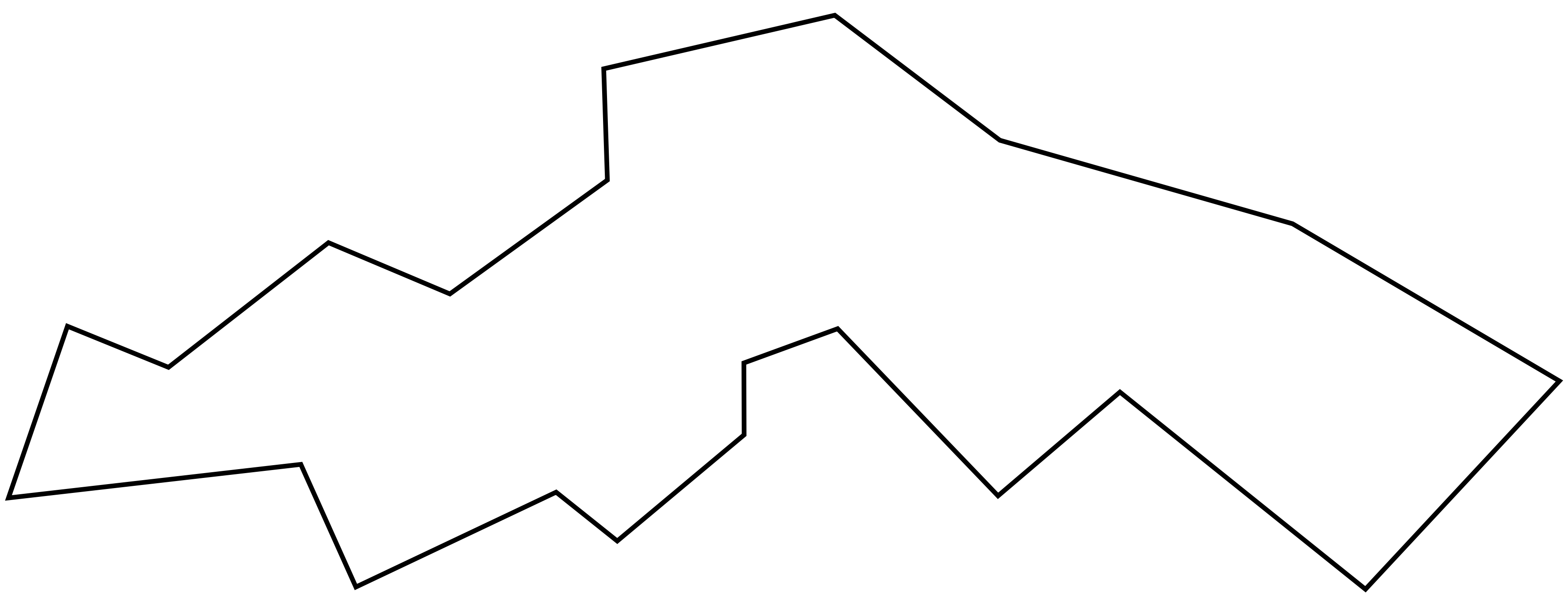
Monotone mountains are easy to triangulate!



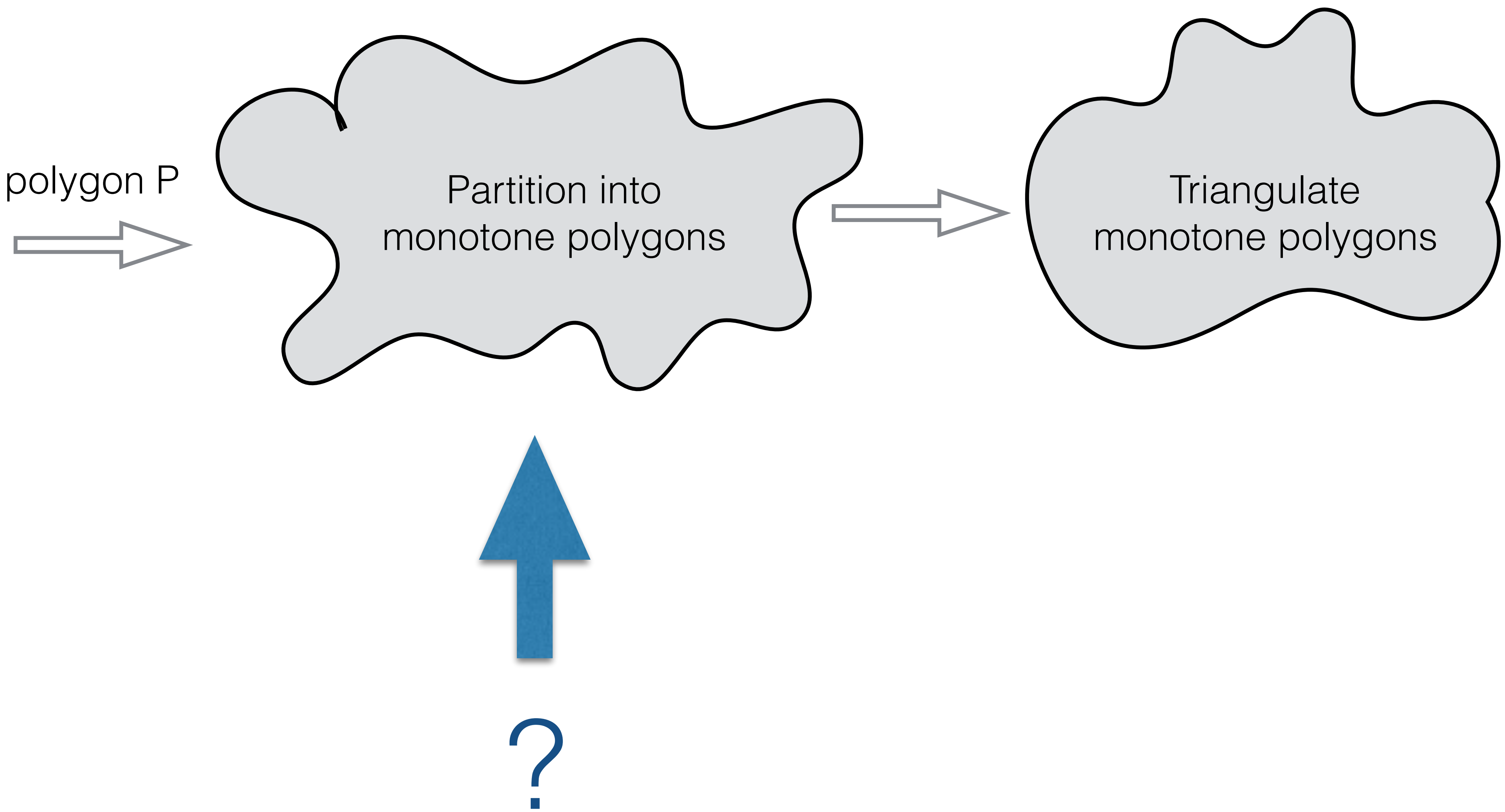
Analysis: O(n) time

Triangulating Monotone Polygons

Similar idea, O(n) time

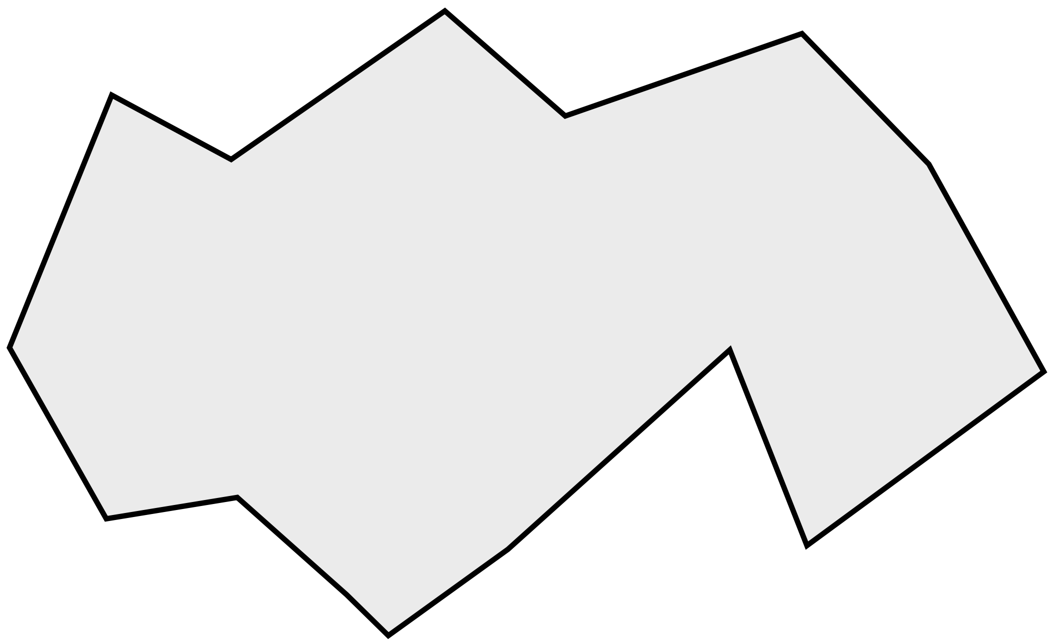


Towards an O(n lg n) Polygon Triangulation Algorithm

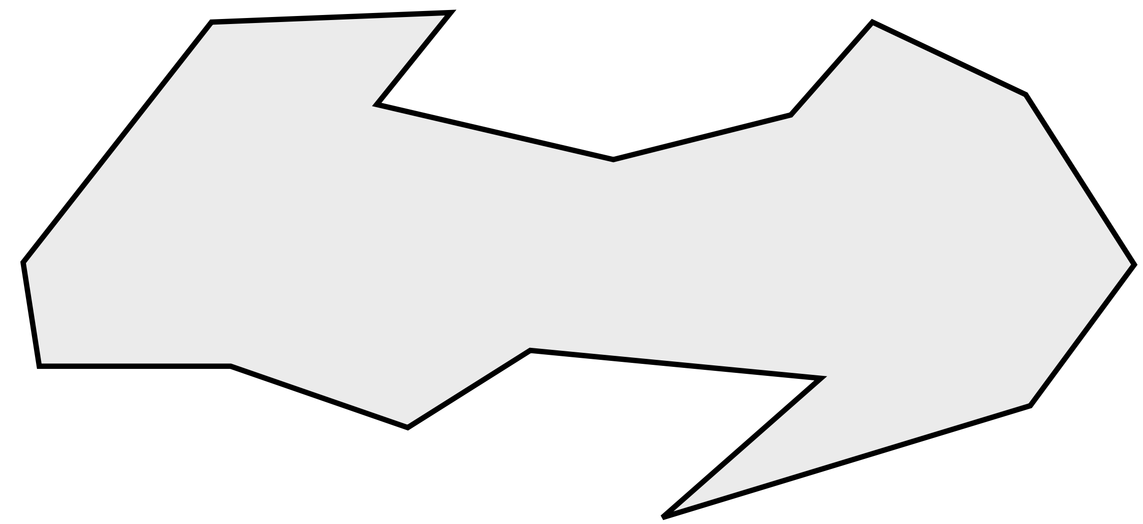


How can we partition a polygon in monotone pieces?

Intuition



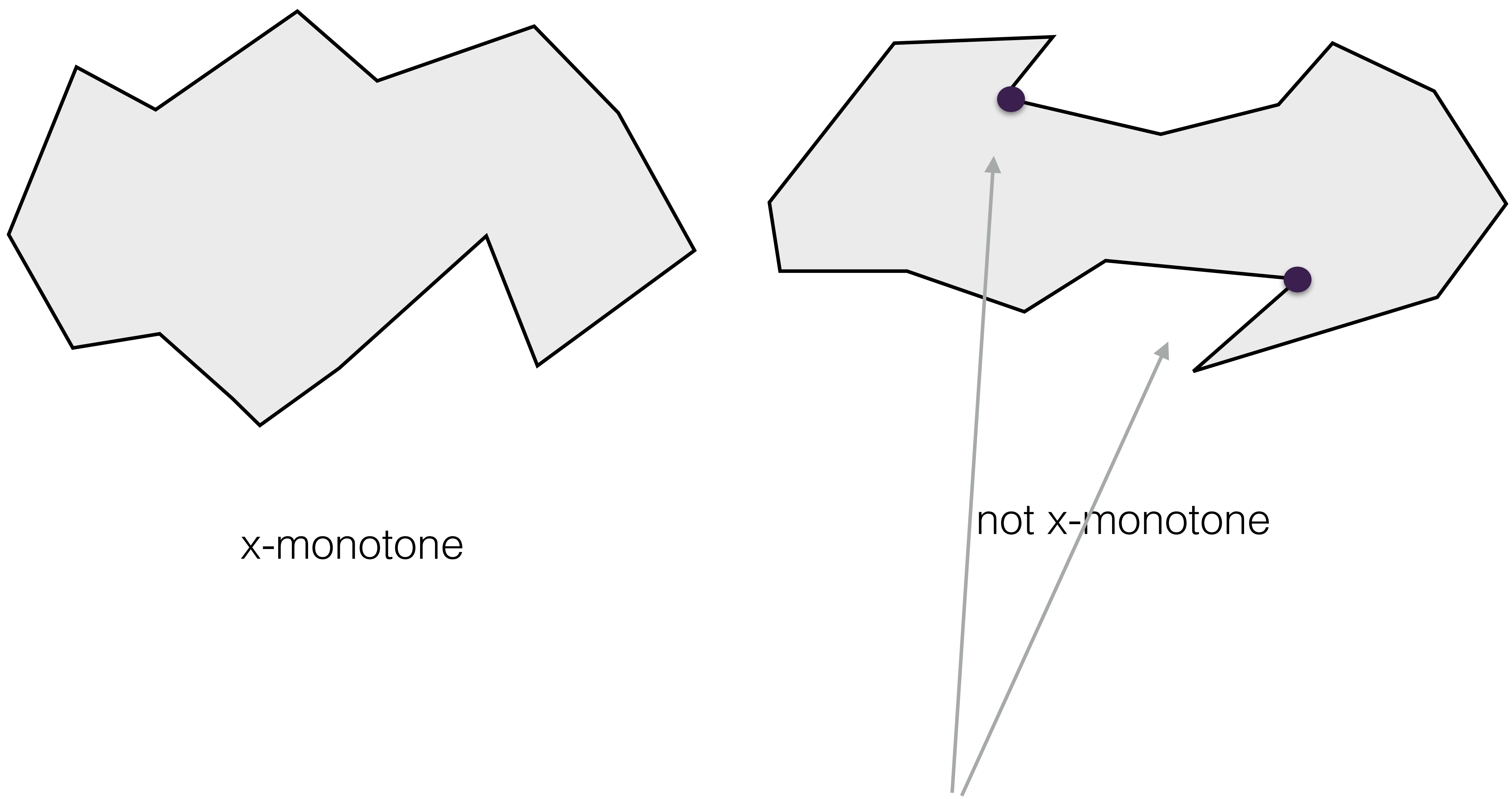
x-monotone



not x-monotone

What makes a polygon **not** monotone?

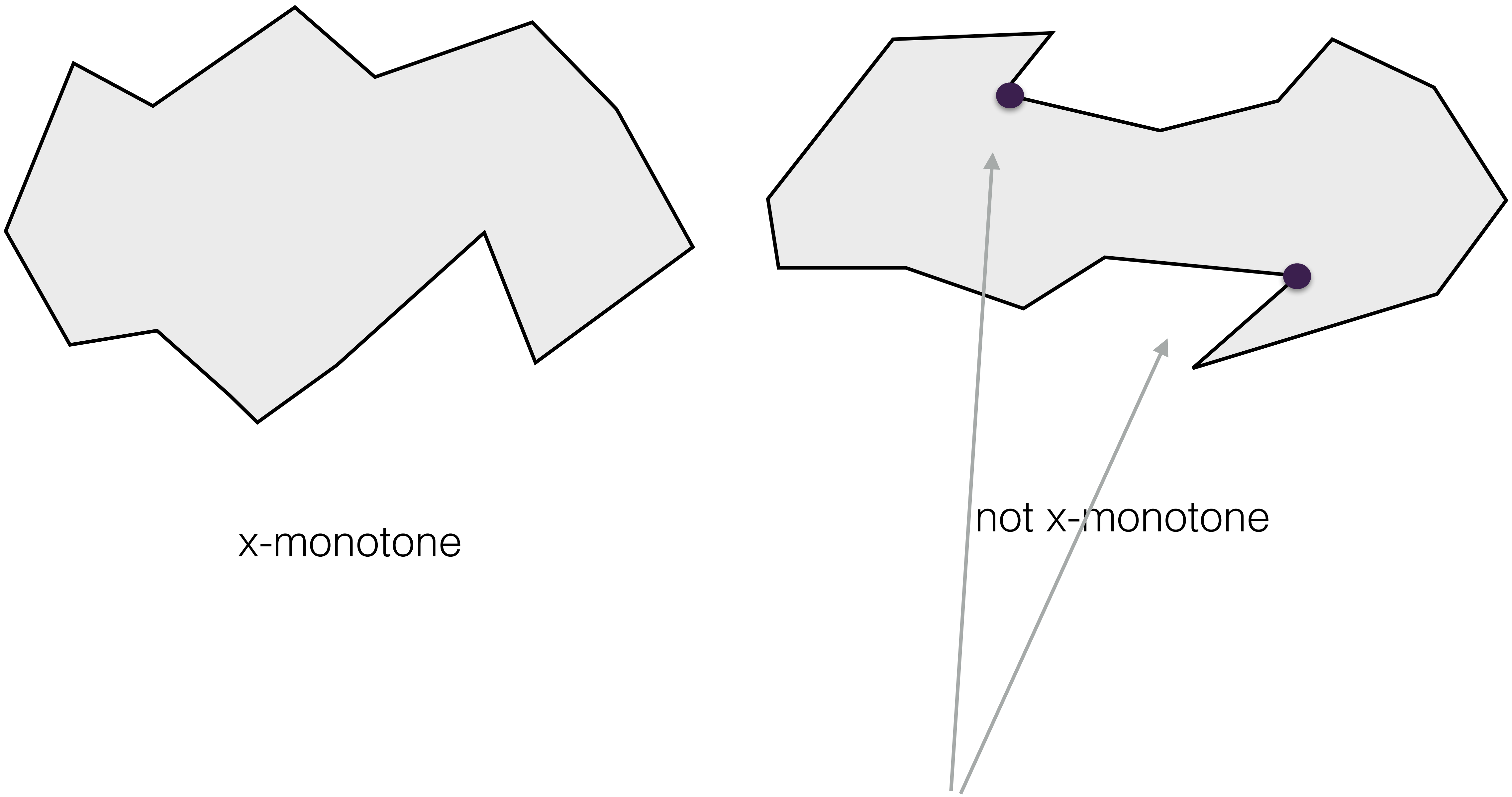
Intuition



What makes a polygon **not** monotone?

Intuition

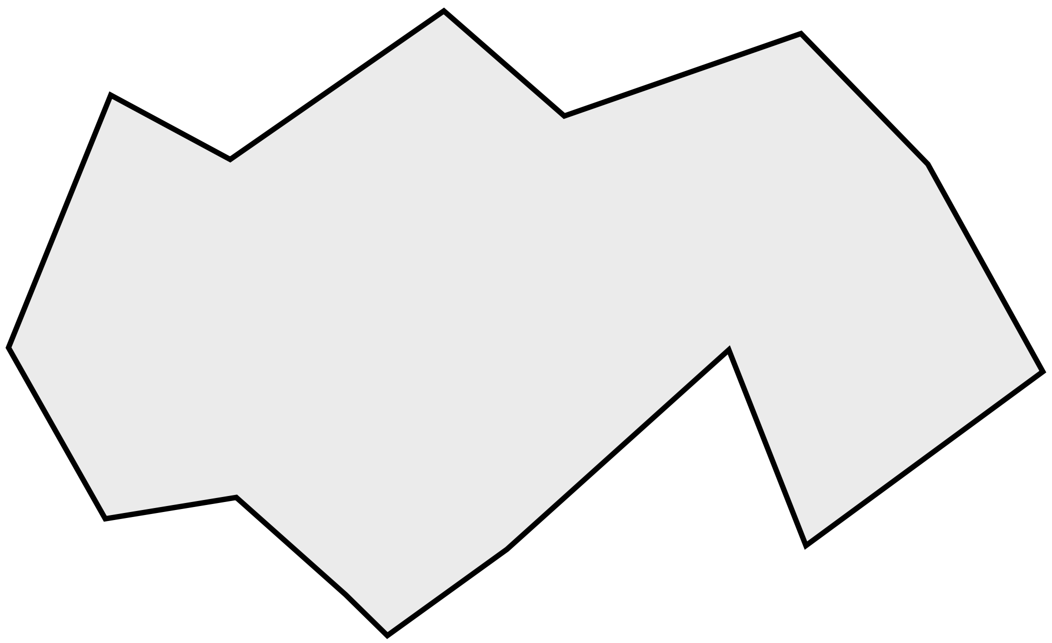
Cusp: a reflex vertex v such that the vertices before and after are both smaller or both larger than v (in terms of x-coords).



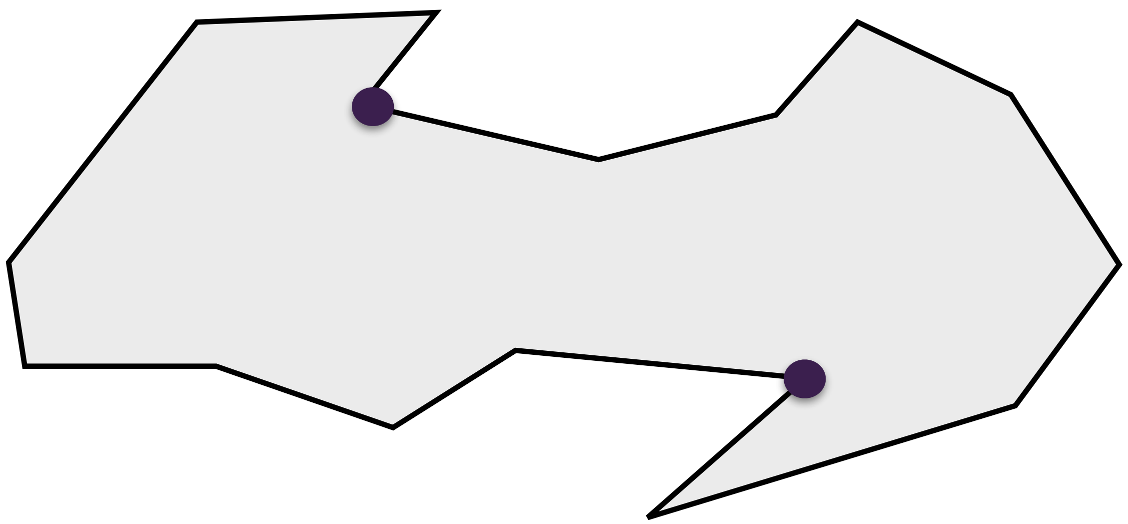
What makes a polygon **not** monotone?

Intuition

Cusp: a reflex vertex v such that the vertices before and after are both smaller or both larger than v (in terms of x-coords).



x-monotone



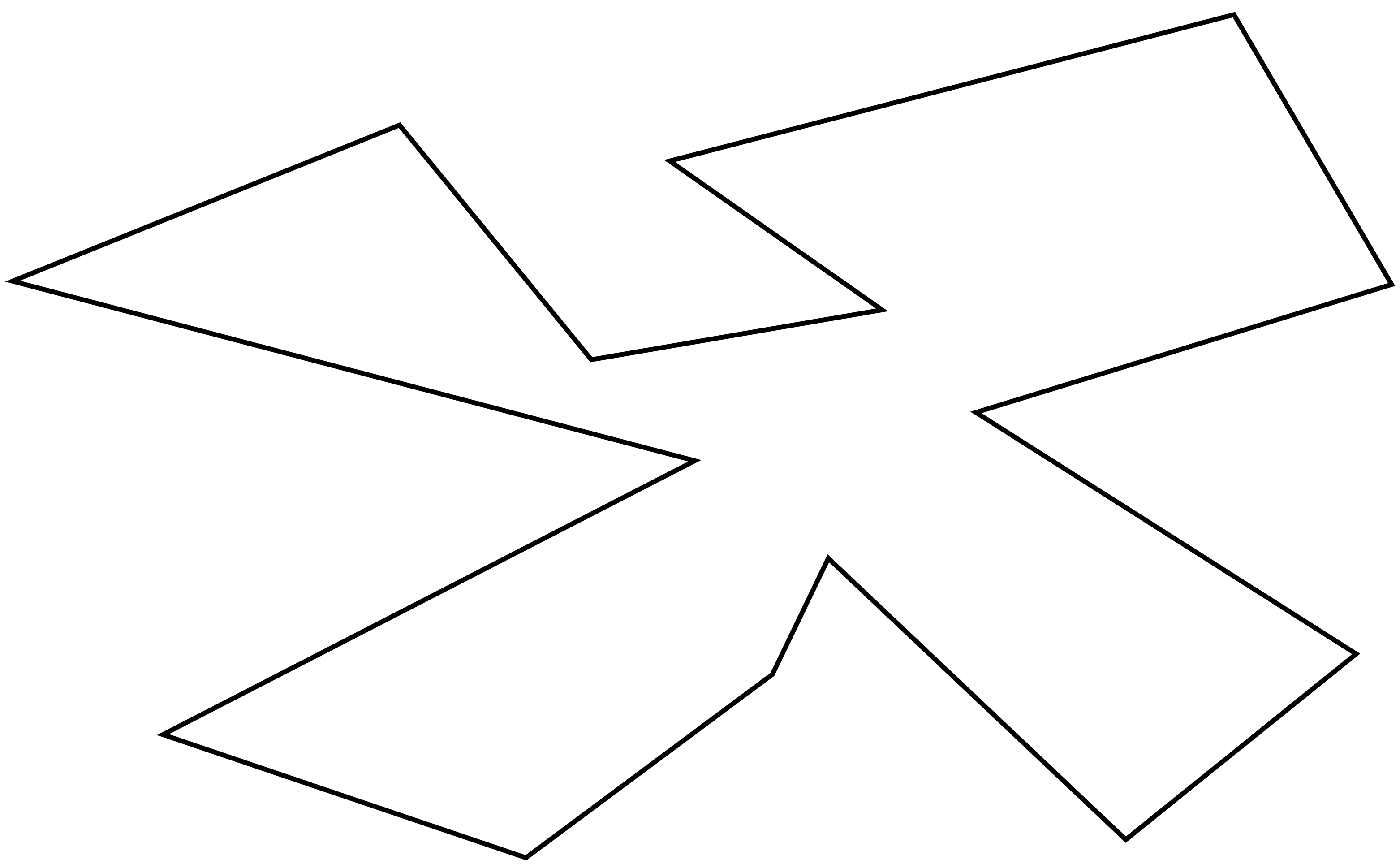
not x-monotone

- Theorem: If a polygon has no cusps, then it's monotone.
- Proof: maybe later..

We’ll get rid of cusps using a trapezoidalization of P.

Trapezoid partitions

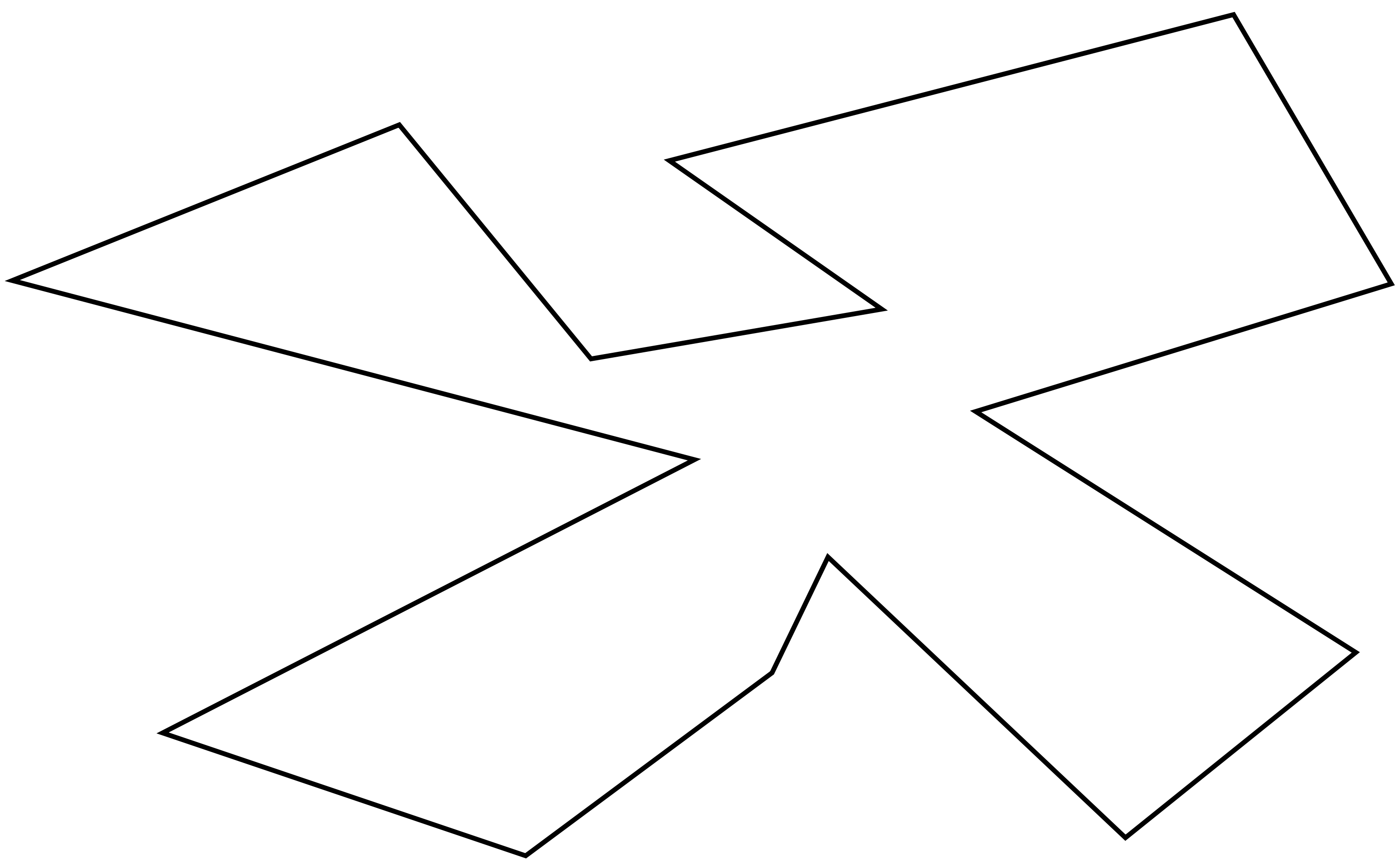
Compute a trapezoidalization (trapezoid partition) of the polygon.



Trapezoid partitions

Compute a trapezoidalization (trapezoid partition) of the polygon.

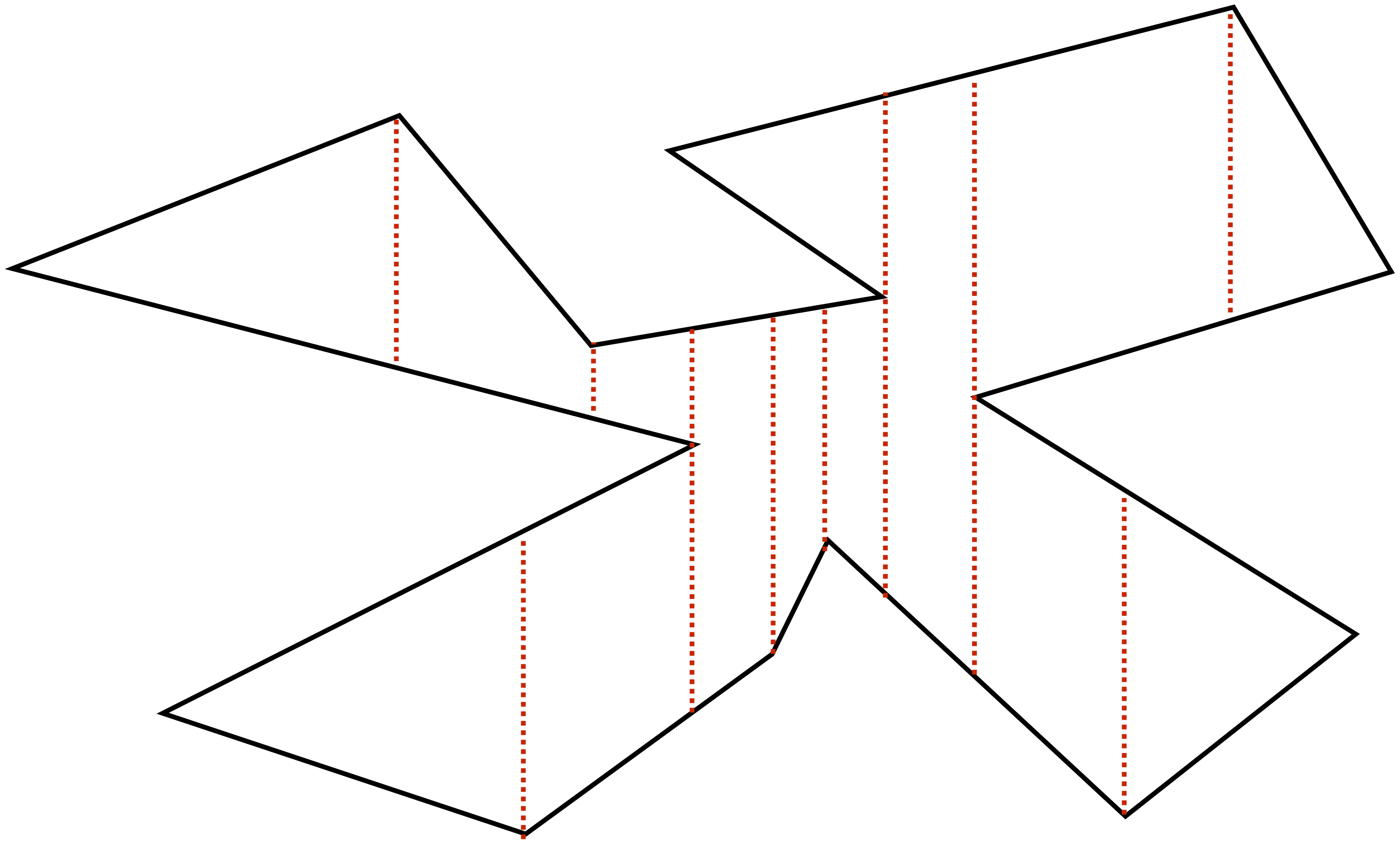
- if polygon is above vertex, shoot vertical ray up until reaches boundary
- if polygon is below vertex, shoot down
- if polygon is above and below vertex, shoot both up and down



Trapezoid partitions

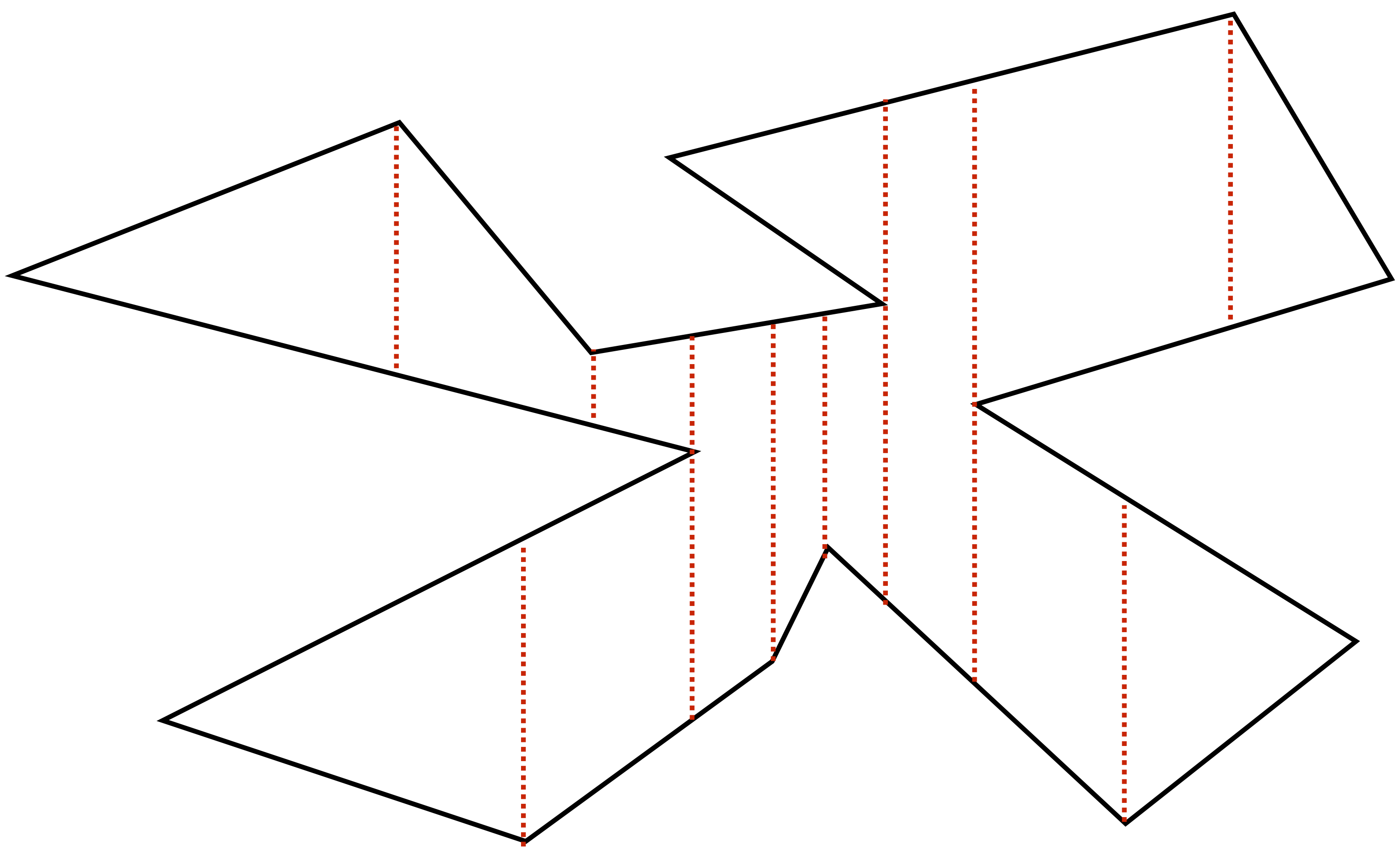
Compute a trapezoidalization (trapezoid partition) of the polygon.

- if polygon is above vertex, shoot vertical ray up until reaches boundary
- if polygon is below vertex, shoot down
- if polygon is above and below vertex, shoot both up and down



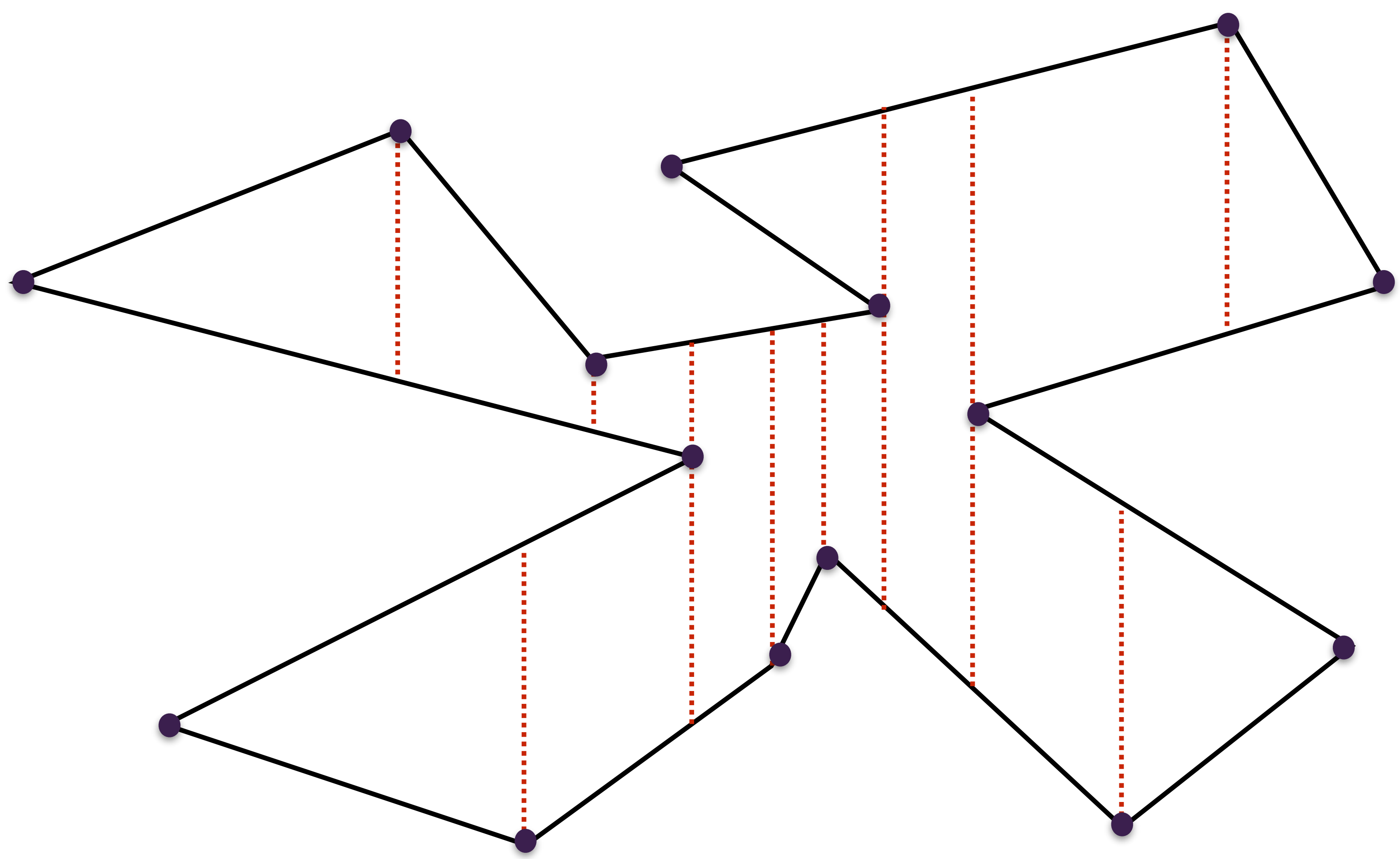
Trapezoid partitions

- Each polygon in the partition is a trapezoid:
  - It has one or two threads as sides.
  - If it has two, then they must both hit the same edge above, and the same edge below.
- At most one thread through each vertex => O(n) threads => O(n) trapezoids



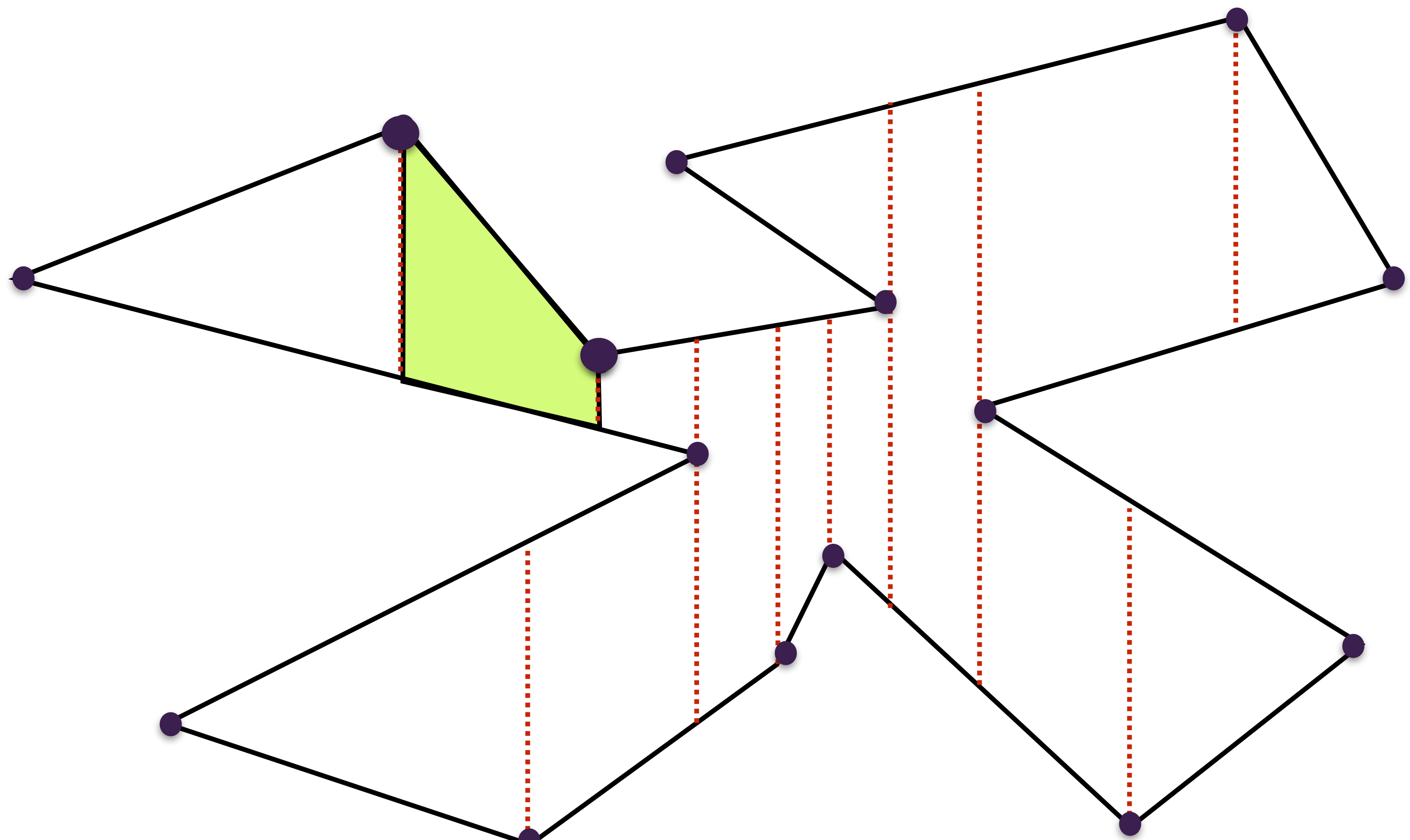
Trapezoid partitions

- Each trapezoid has precisely two vertices of the polygon, one on the left and one on the right. They can be on the top, bottom or middle of the trapezoid.



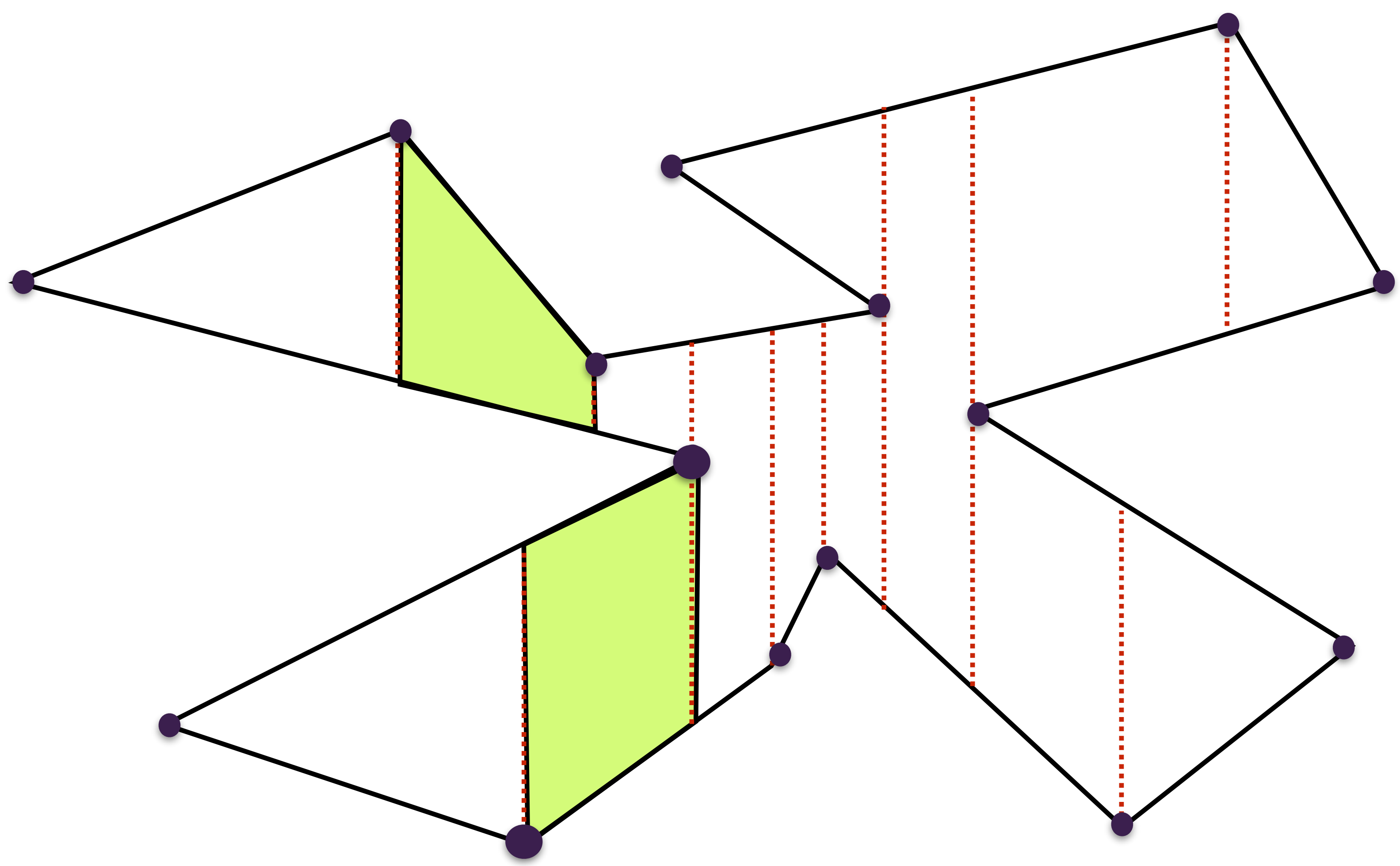
Trapezoid partitions

- Each trapezoid has precisely two vertices of the polygon, one on the left and one on the right. They can be on the top, bottom or middle of the trapezoid.



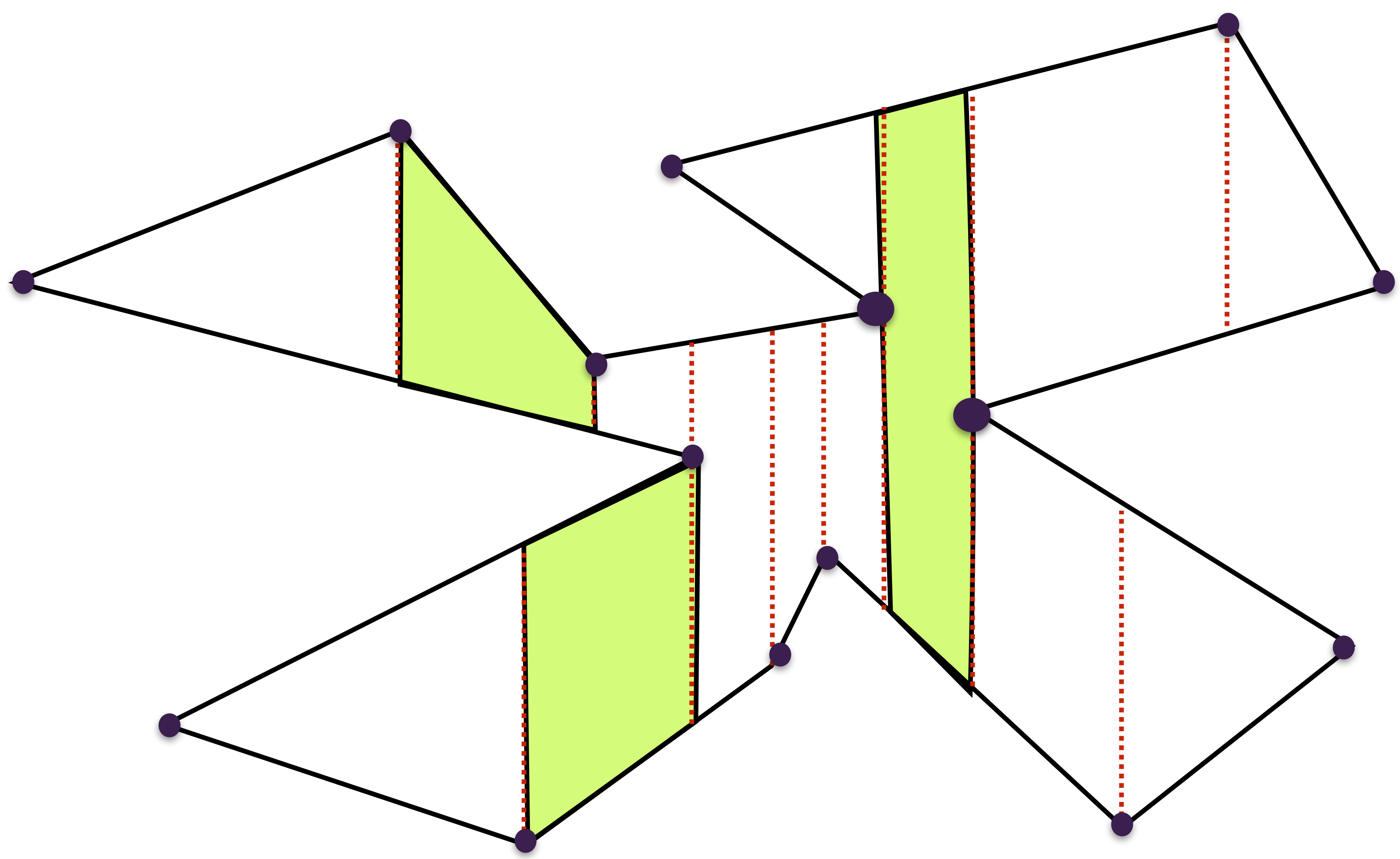
Trapezoid partitions

- Each trapezoid has precisely two vertices of the polygon, one on the left and one on the right. They can be on the top, bottom or middle of the trapezoid.



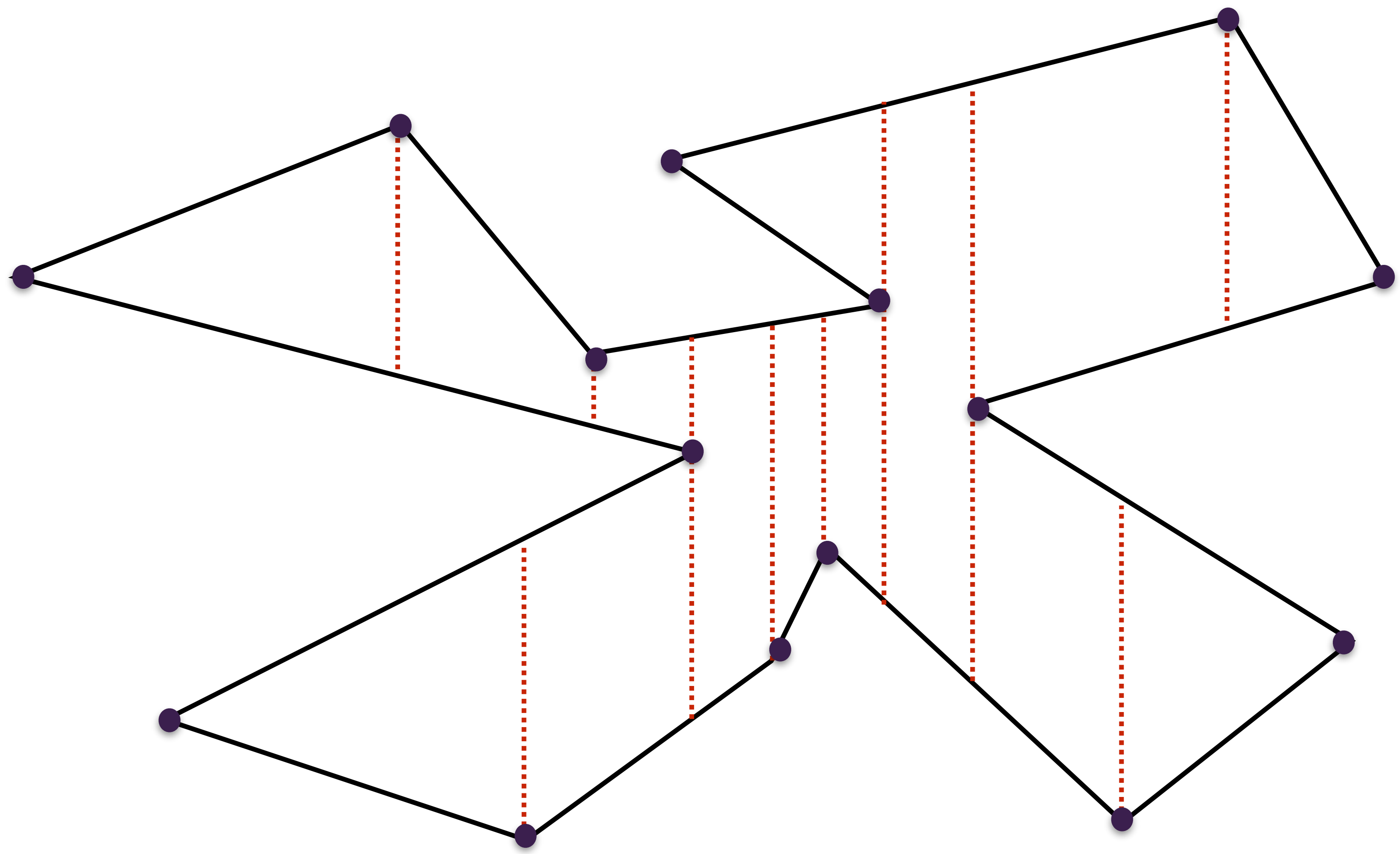
Trapezoid partitions

- Each trapezoid has precisely two vertices of the polygon, one on the left and one on the right. They can be on the top, bottom or middle of the trapezoid.



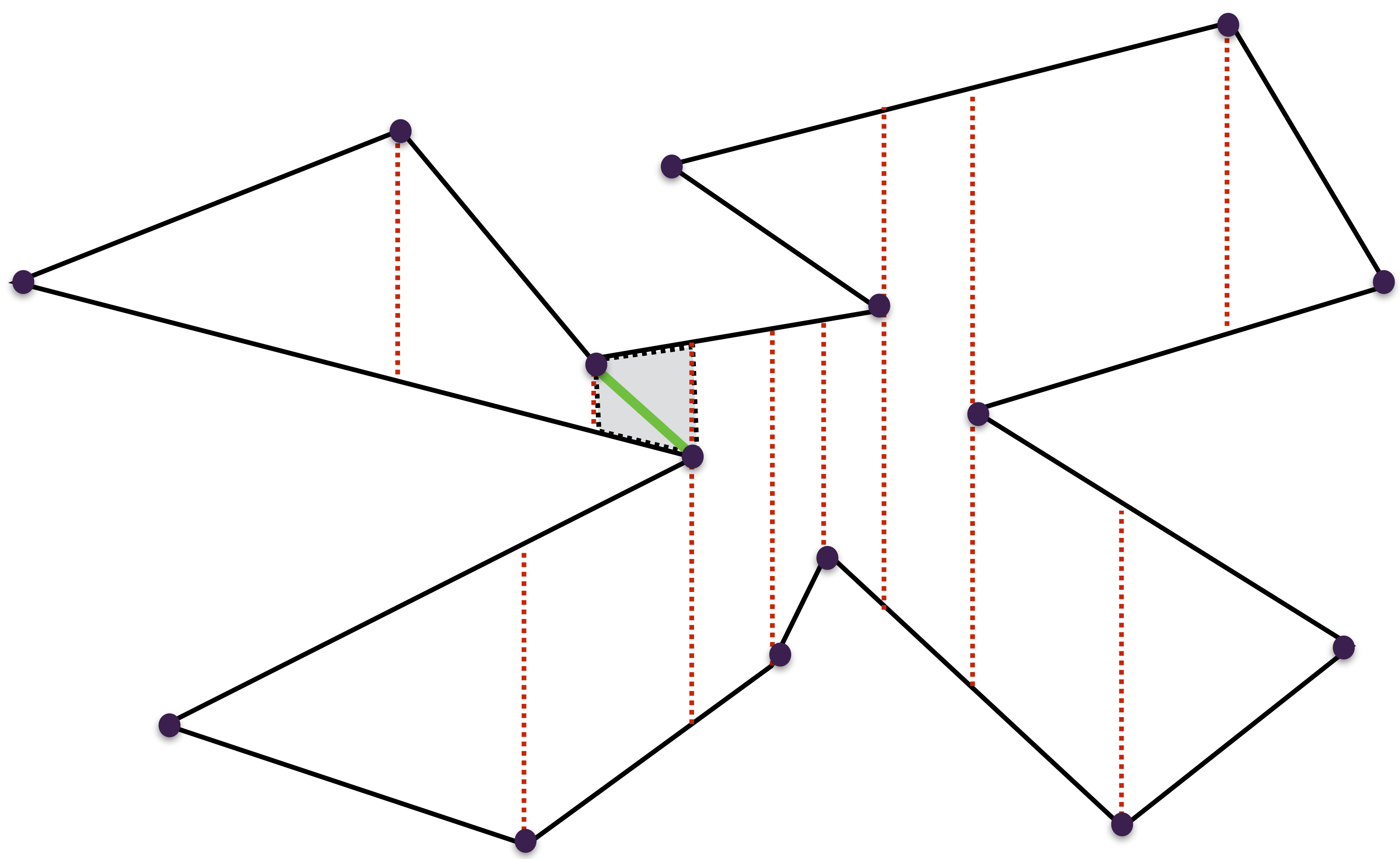
Trapezoid partitions

- In each trapezoid: if its two vertices are not on the same edge, they define a diagonal.



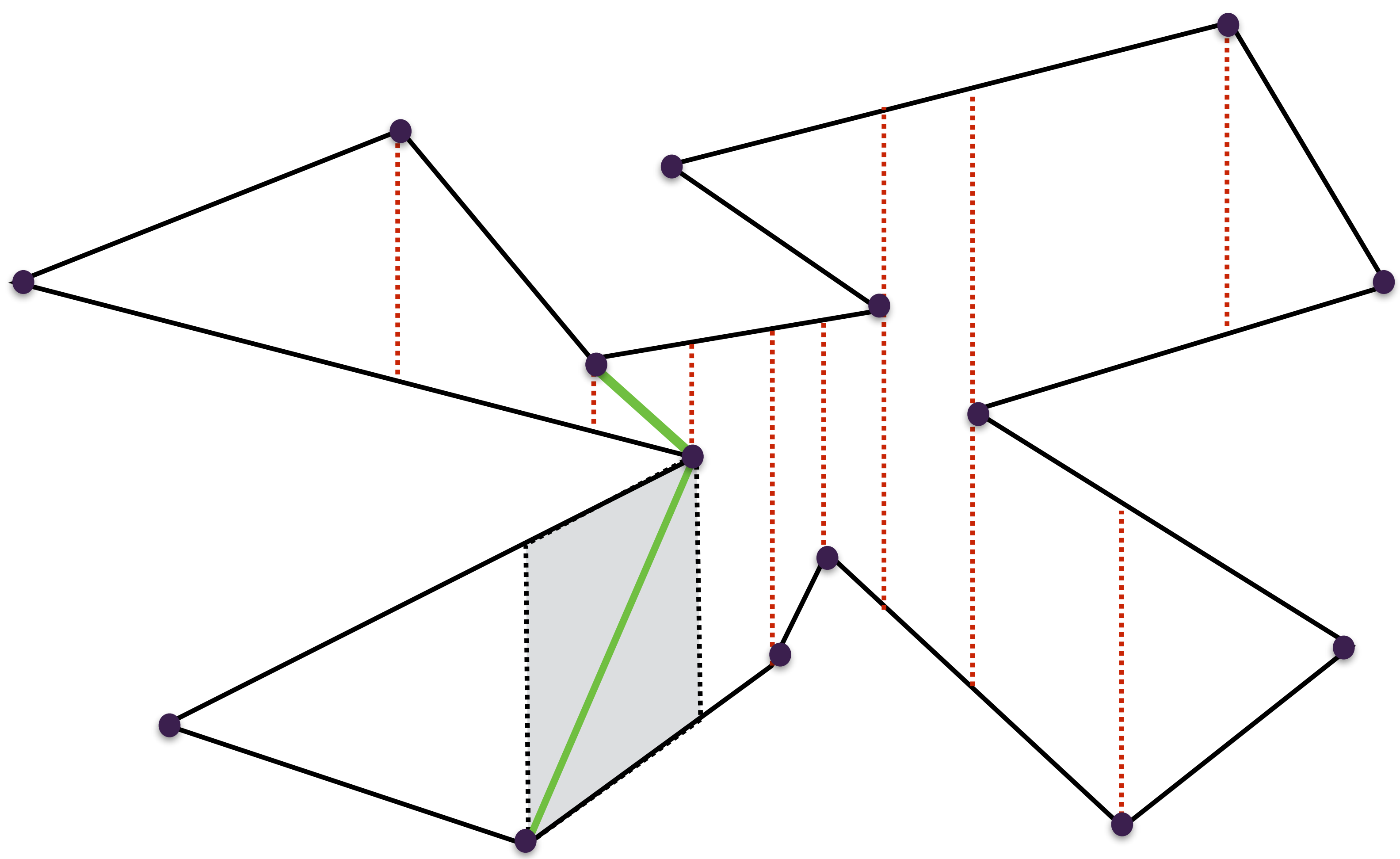
Trapezoid partitions

- In each trapezoid: if its two vertices are not on the same edge, they define a diagonal.



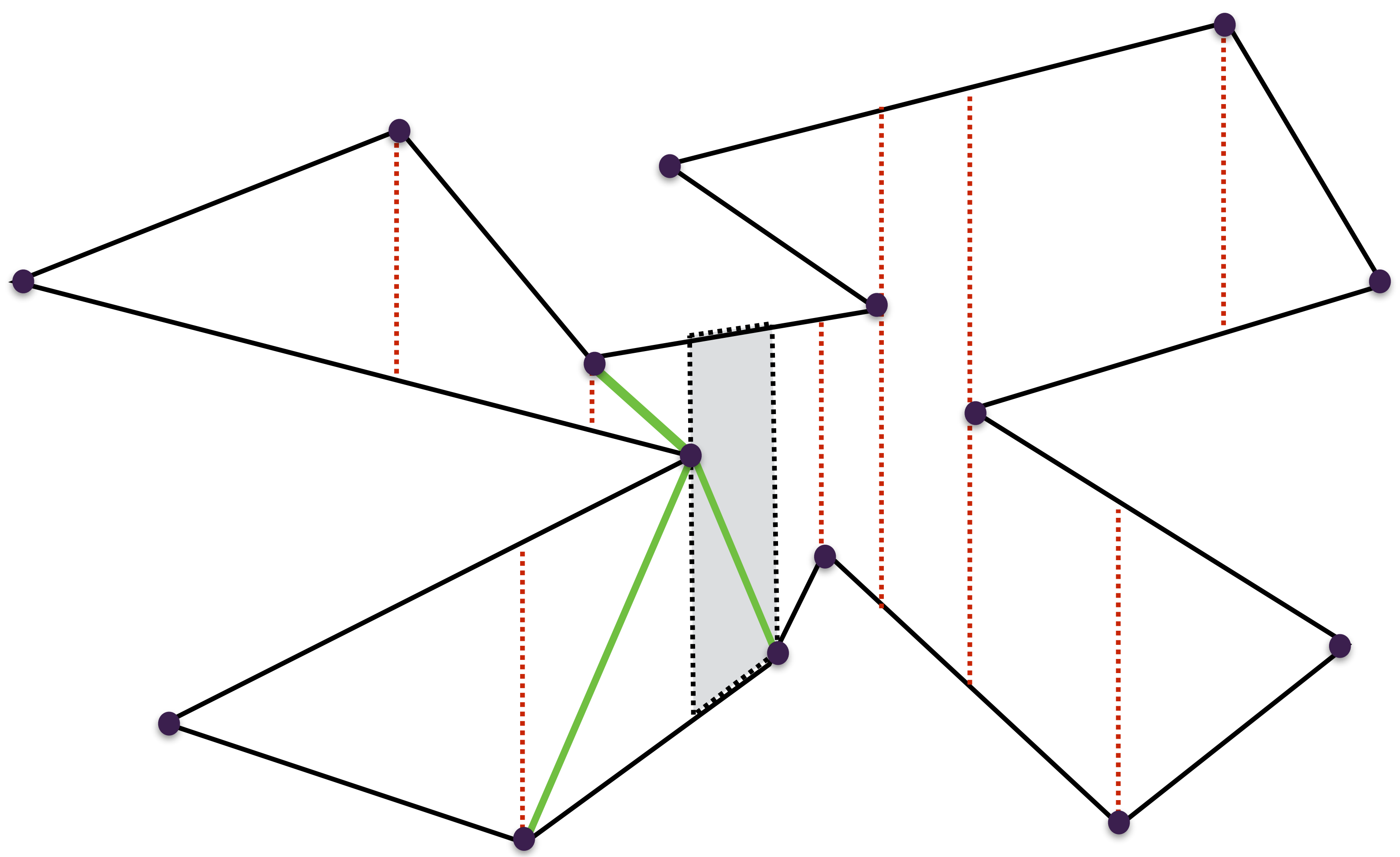
Trapezoid partitions

- In each trapezoid: if its two vertices are not on the same edge, they define a diagonal.



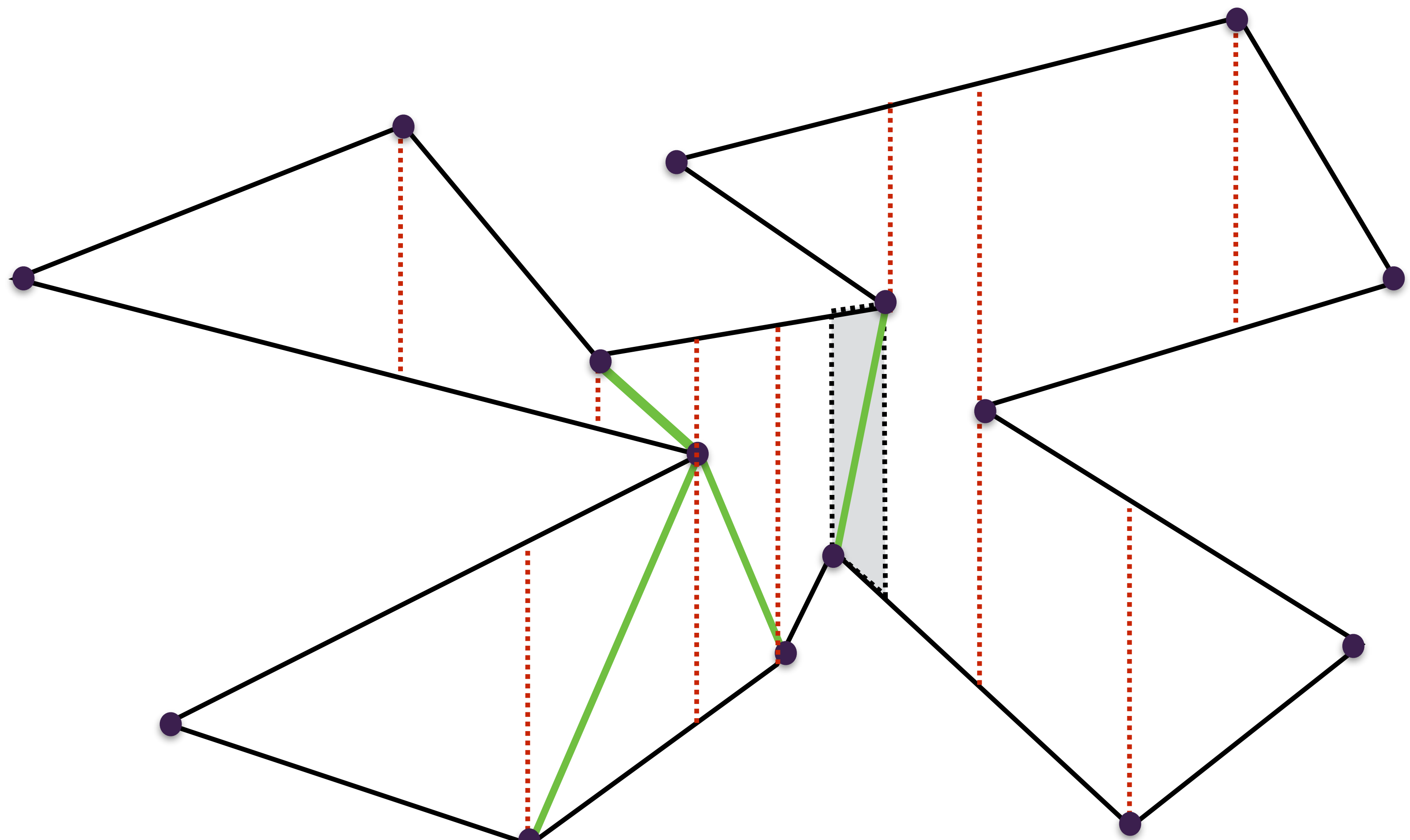
Trapezoid partitions

- In each trapezoid: if its two vertices are not on the same edge, they define a diagonal.



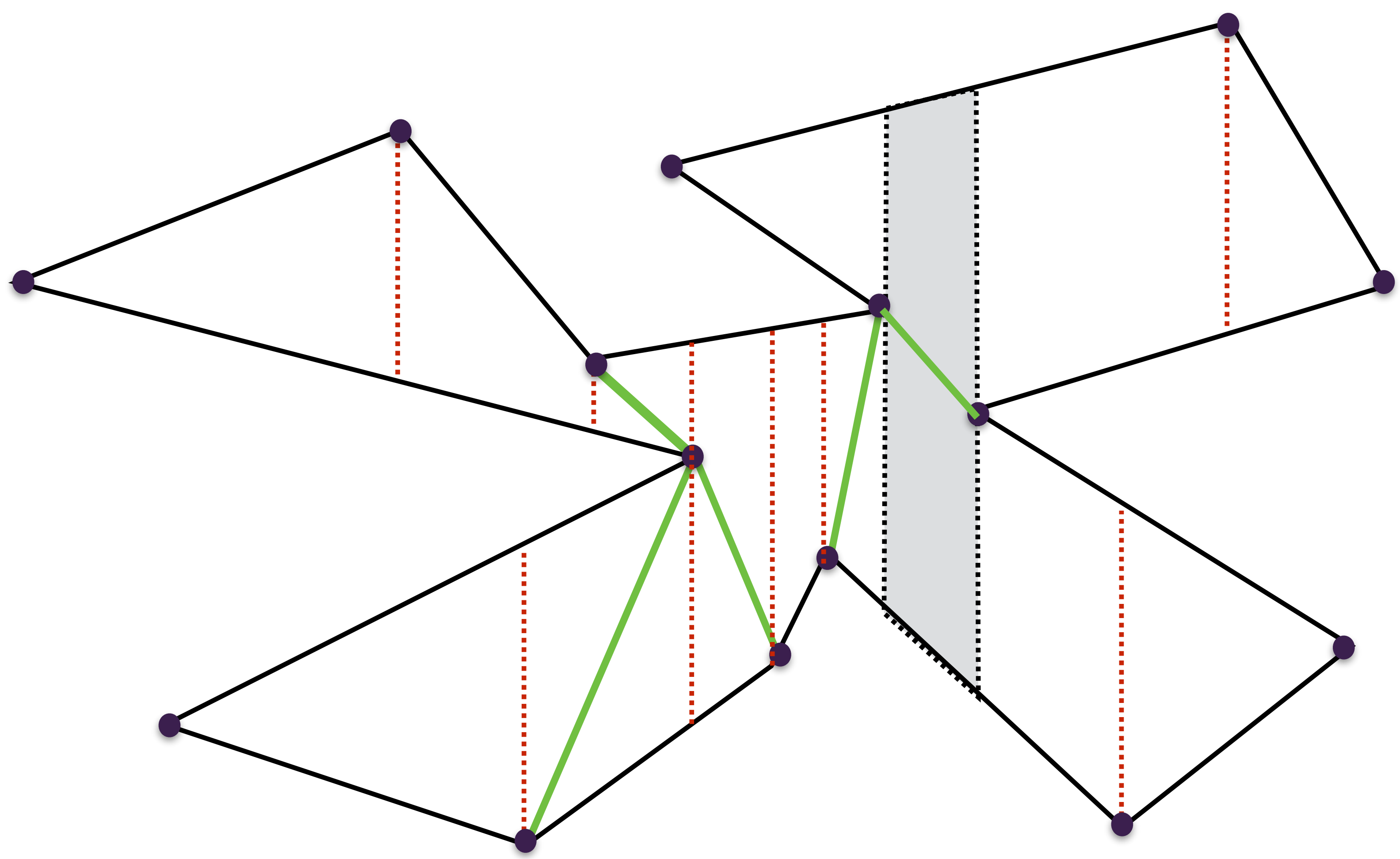
Trapezoid partitions

- In each trapezoid: if its two vertices are not on the same edge, they define a diagonal.



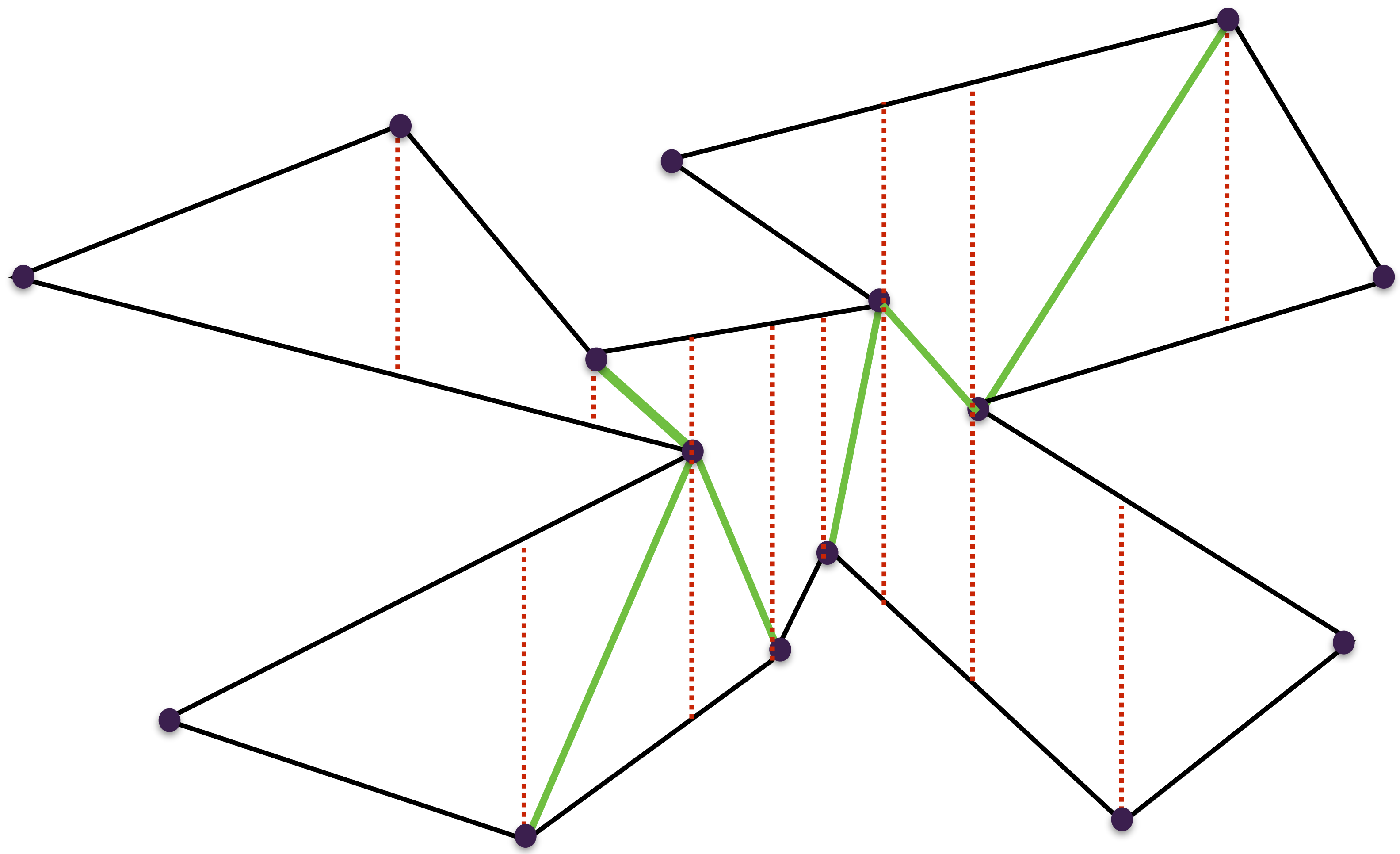
Trapezoid partitions

- In each trapezoid: if its two vertices are not on the same edge, they define a diagonal.



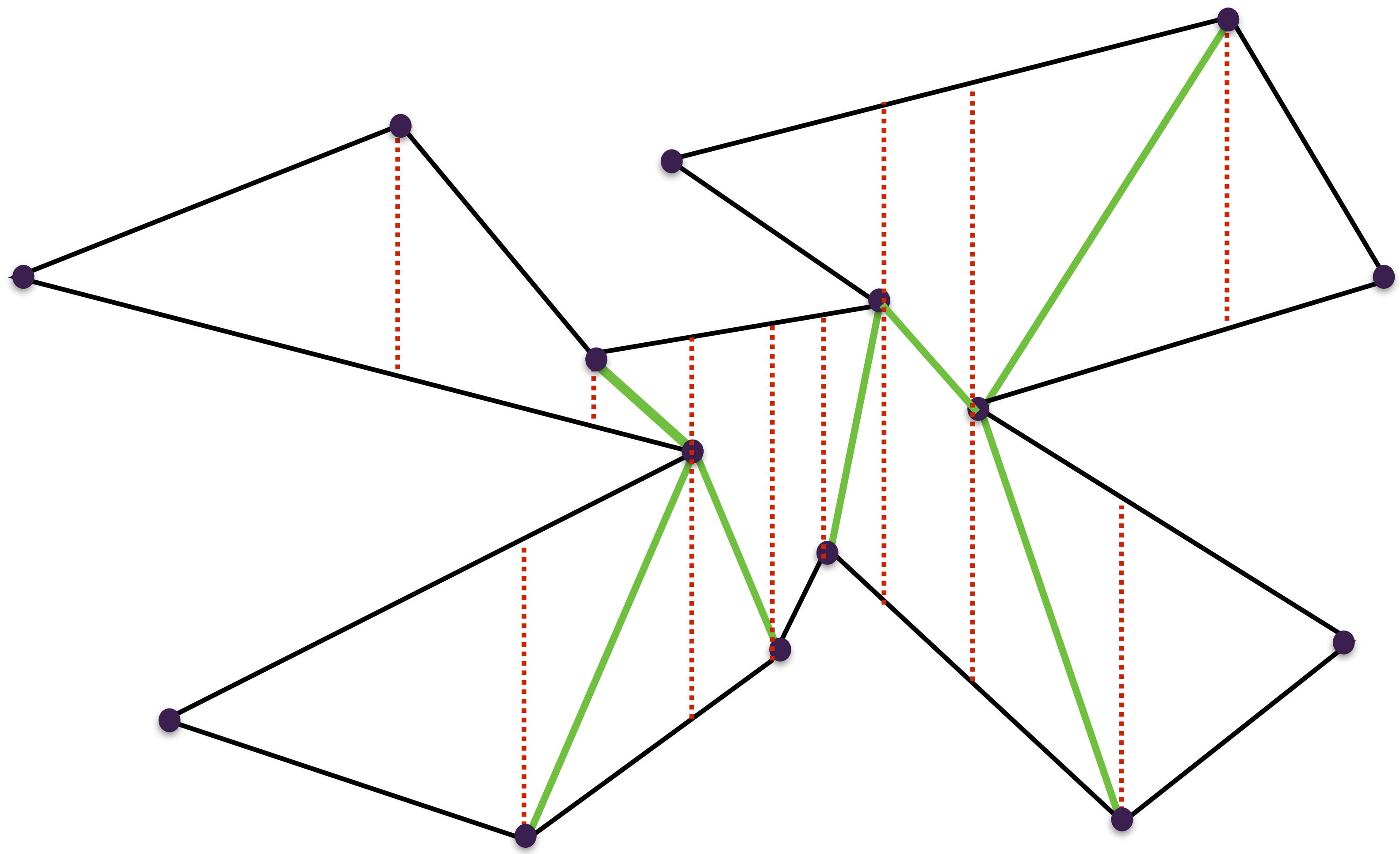
Trapezoid partitions

- In each trapezoid: if its two vertices are not on the same edge, they define a diagonal.



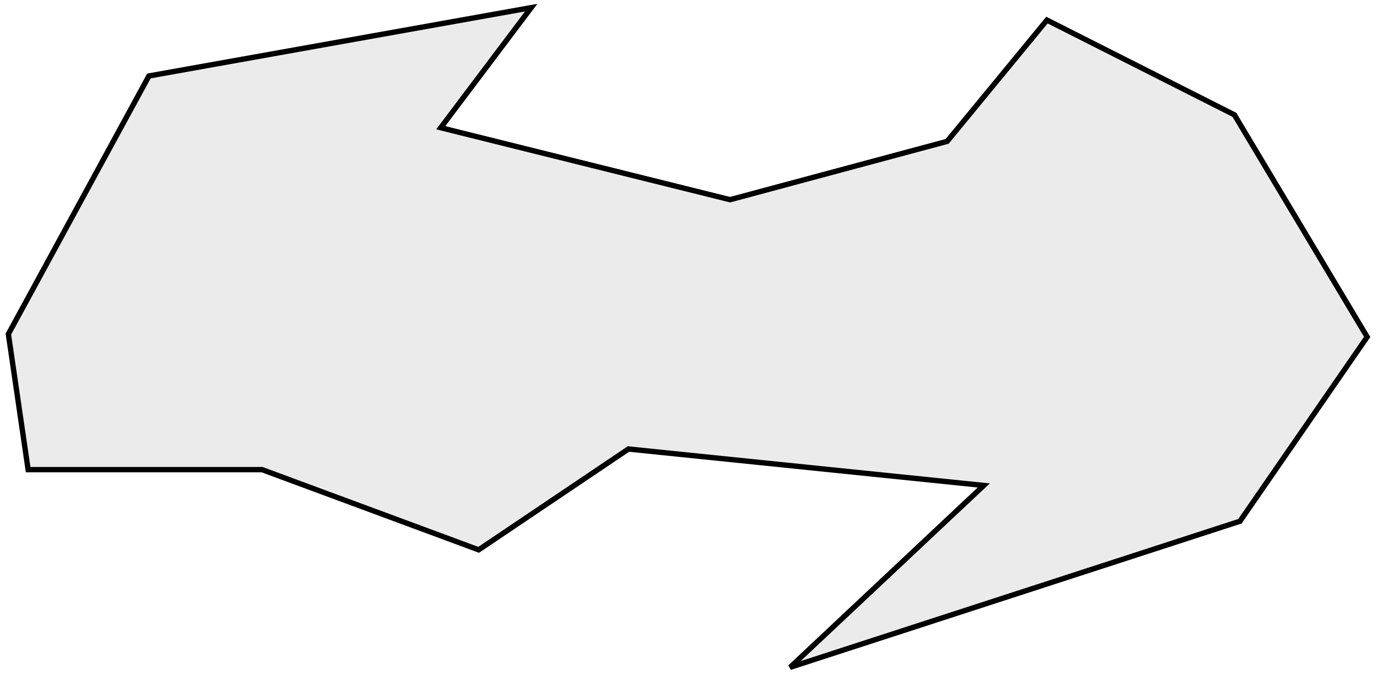
Trapezoid partitions

- In each trapezoid: if its two vertices are not on the same edge, they define a diagonal.



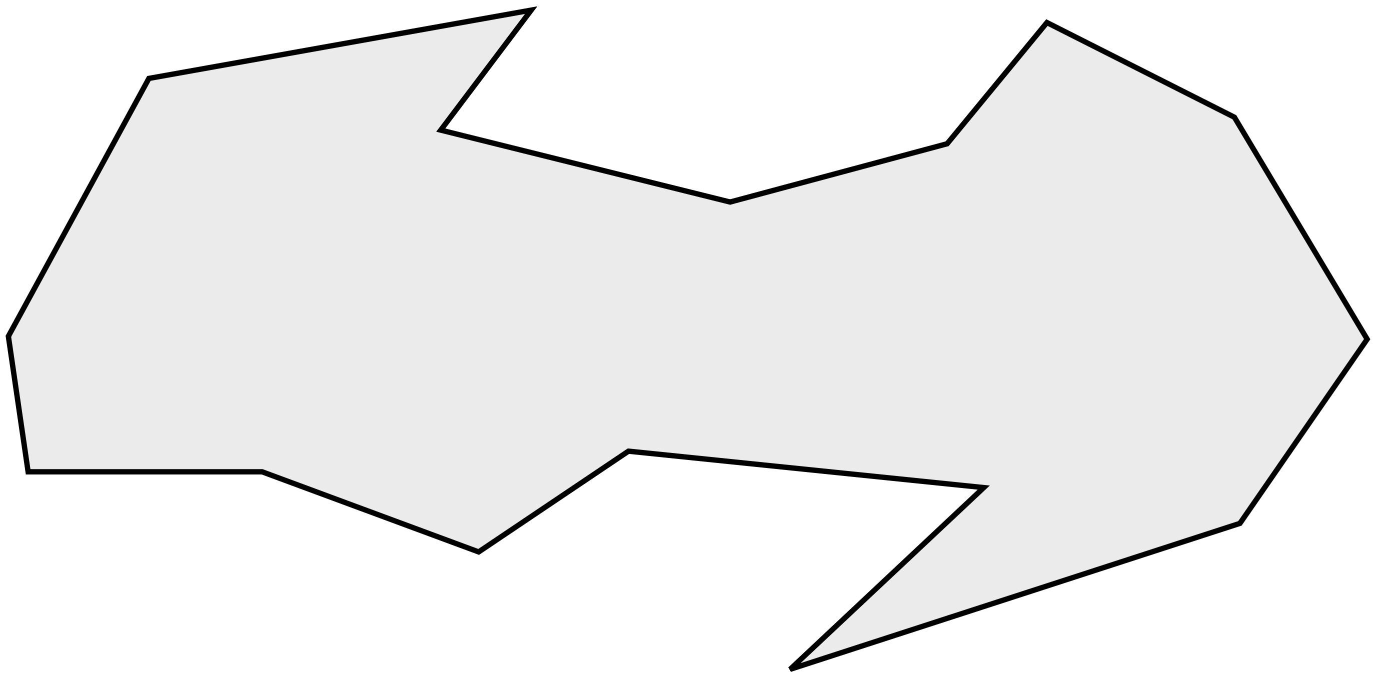
We’ll use the trapezoid partition of P
to get rid of cusps and split it into monotone polygons

Removing cusps



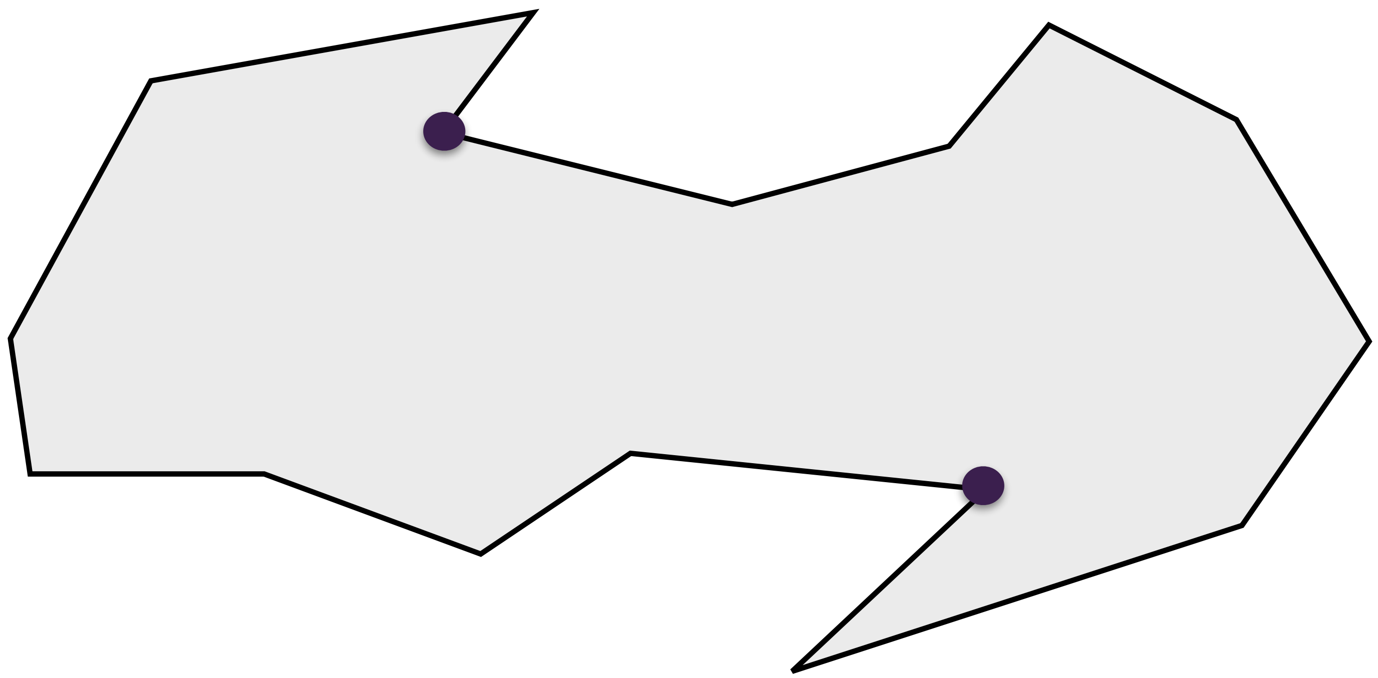
1. Compute a trapezoid partition of P

Removing cusps



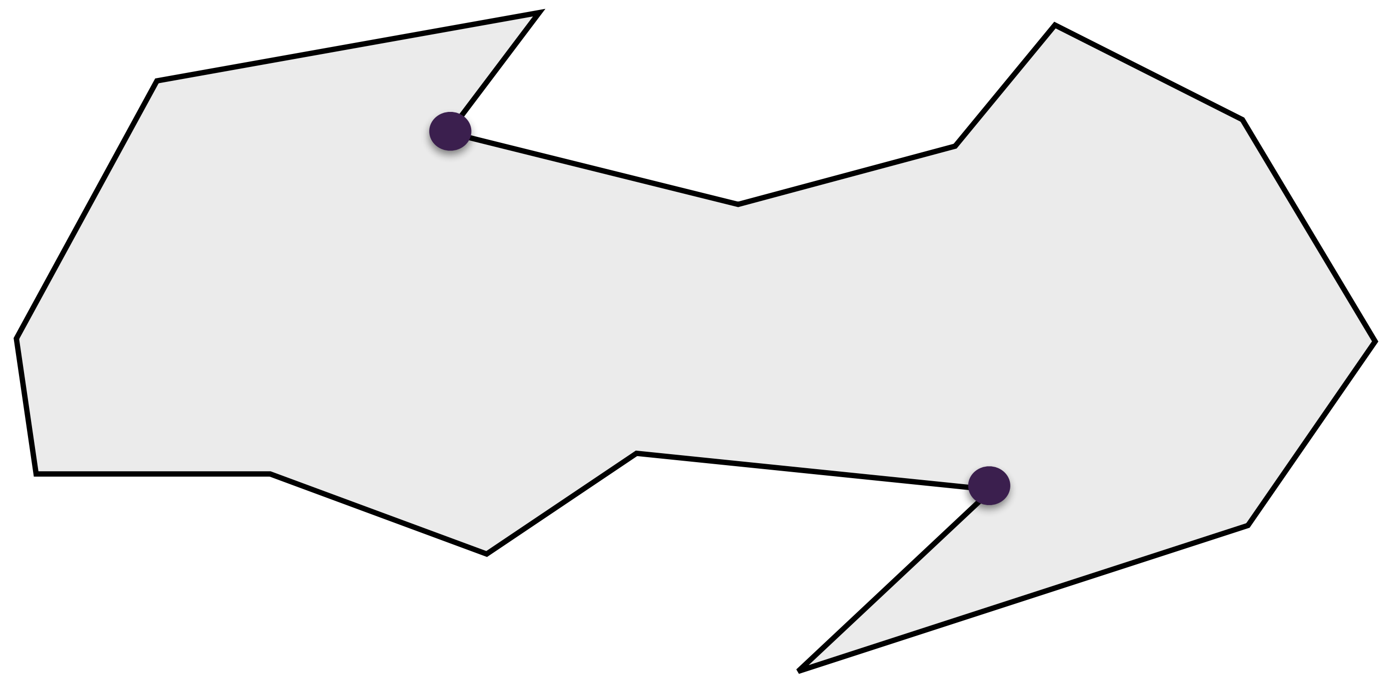
1. Compute a trapezoid partition of P
2. Identify cusp vertices

Removing cusps



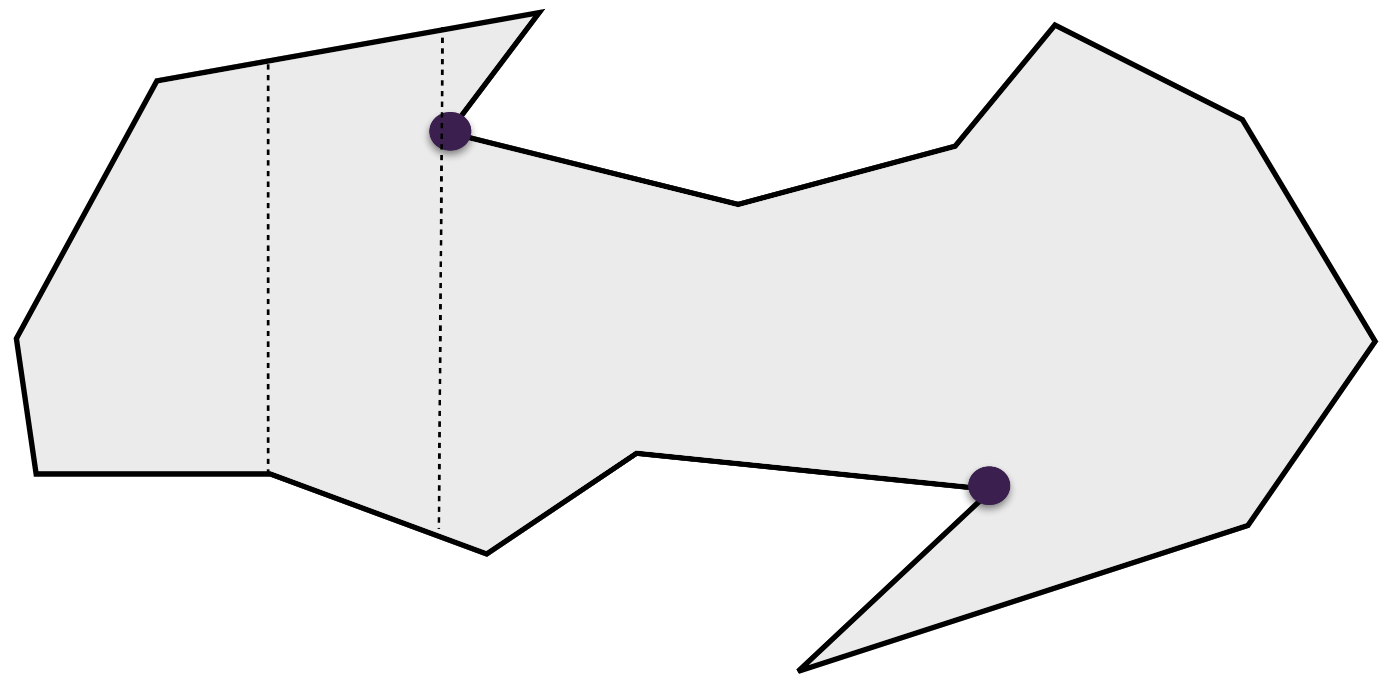
1. Compute a trapezoid partition of P
2. Identify cusp vertices

Removing cusps



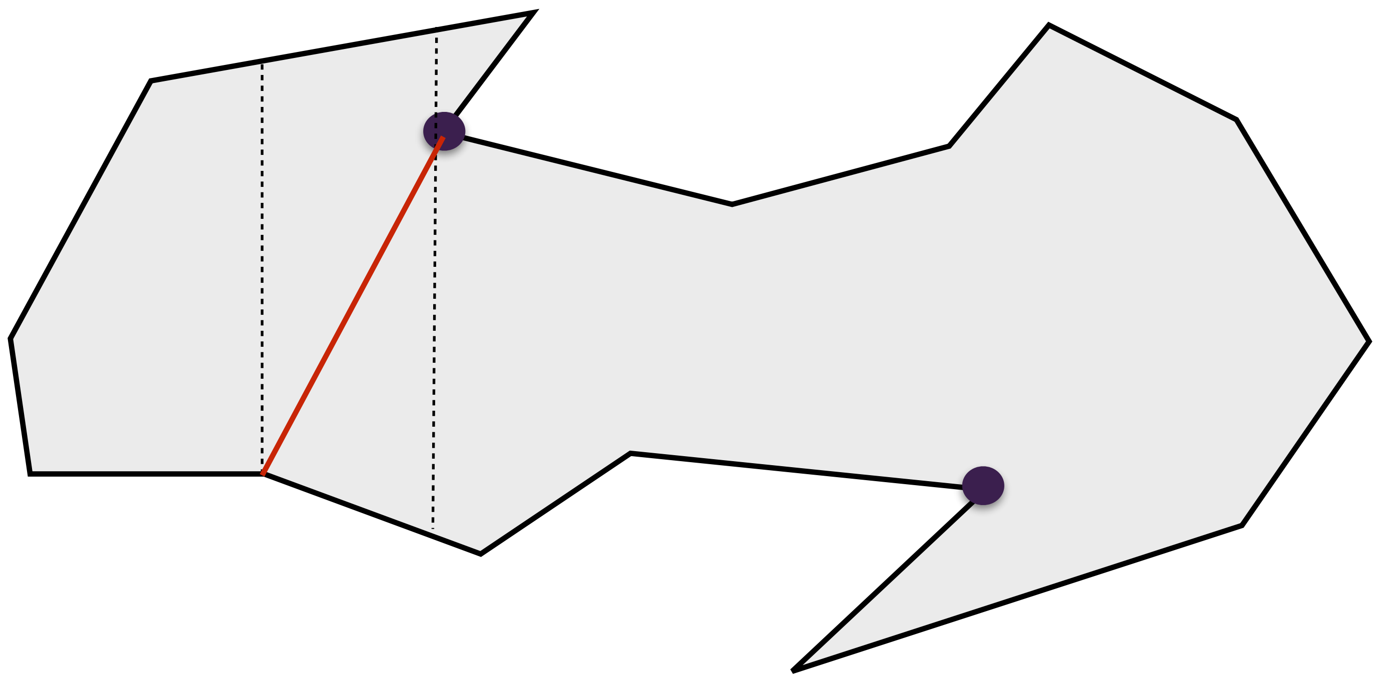
1. Compute a trapezoid partition of P
2. Identify cusp vertices
3. For each cusp vertex, add diagonal in trapezoid before/after the cusp

Removing cusps



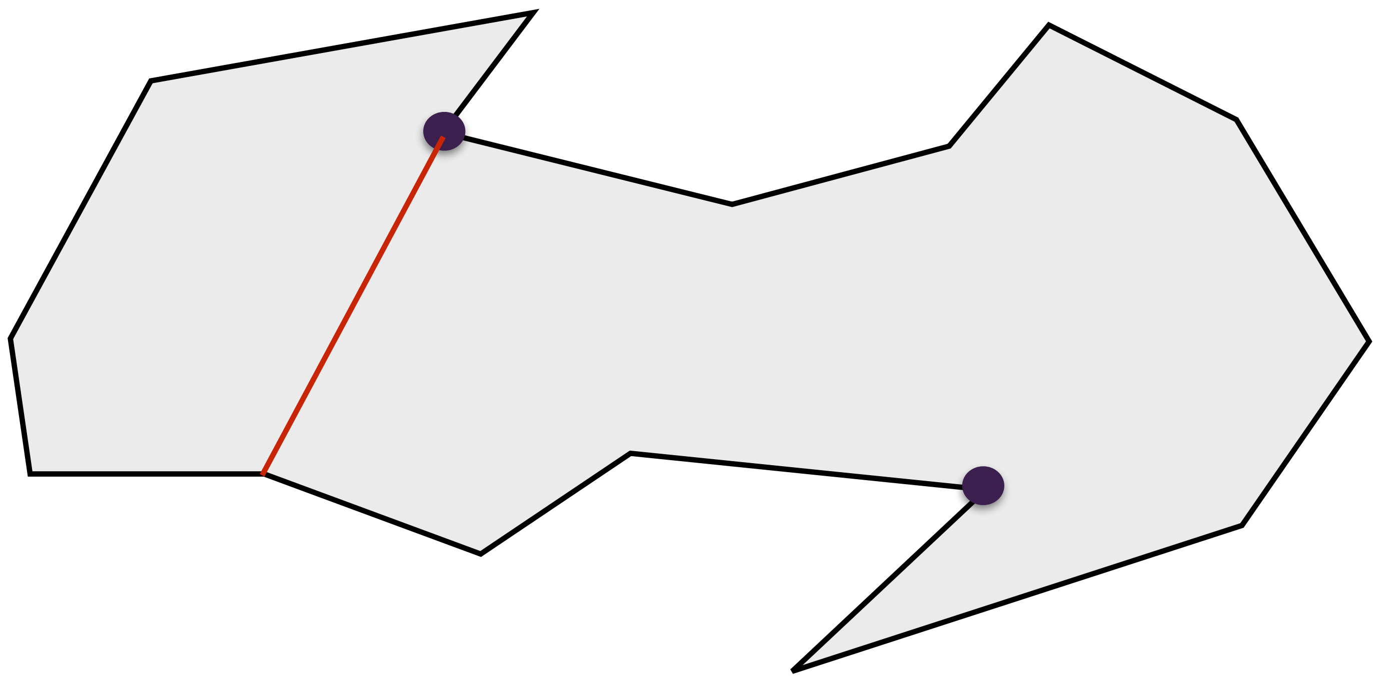
1. Compute a trapezoid partition of P
2. Identify cusp vertices
3. For each cusp vertex, add diagonal in trapezoid before/after the cusp

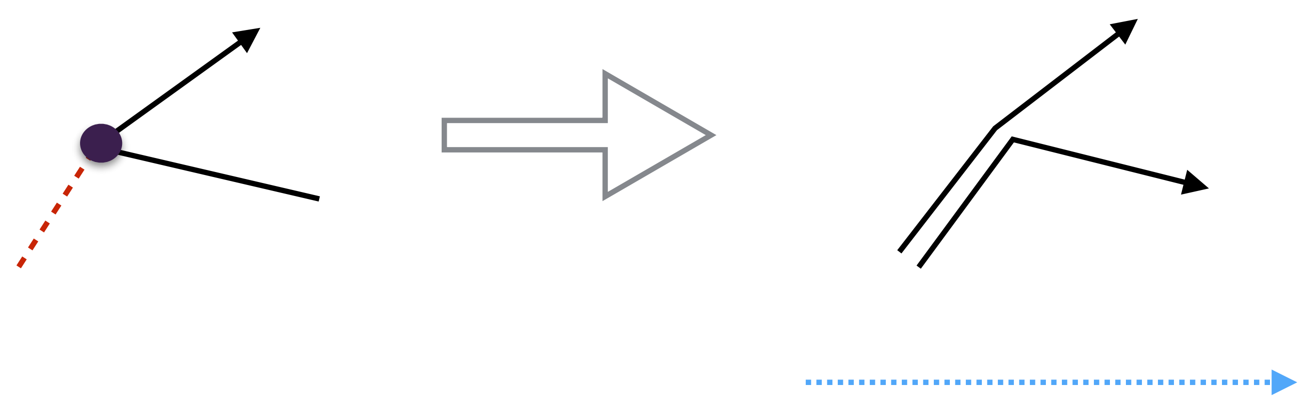
Removing cusps



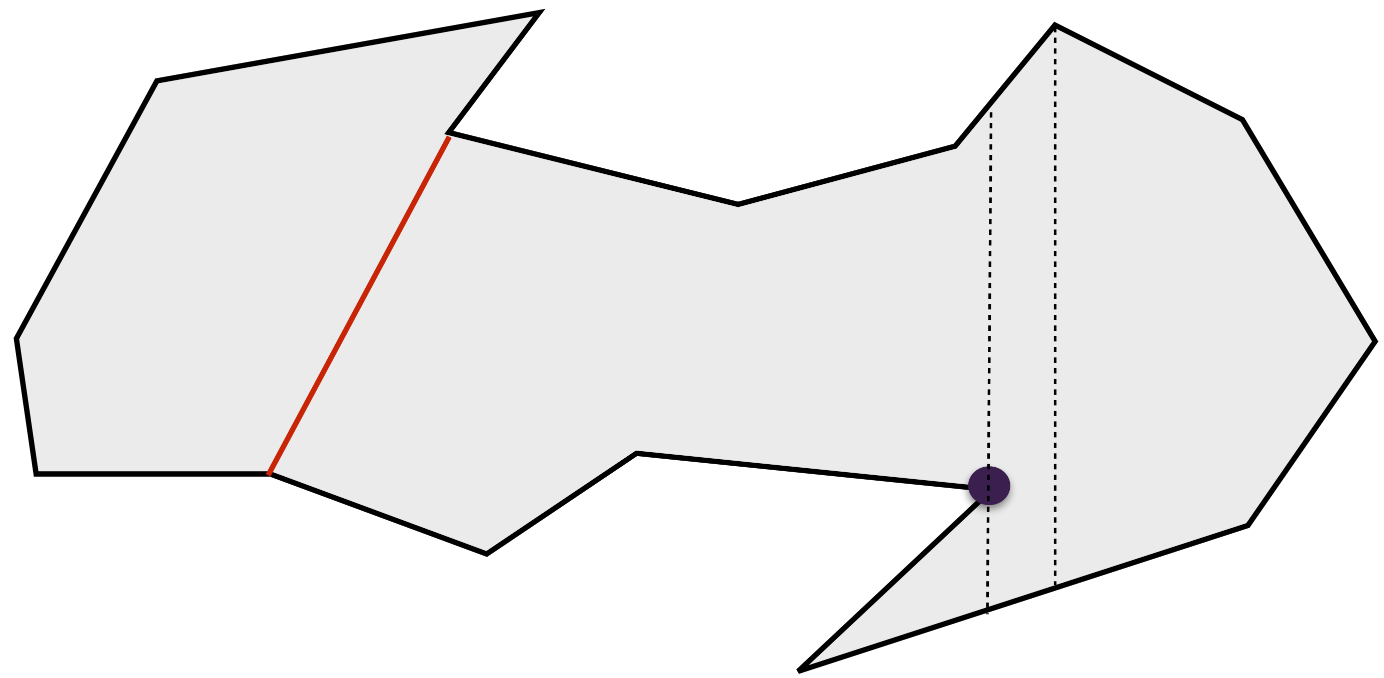
1. Compute a trapezoid partition of P
2. Identify cusp vertices
3. For each cusp vertex, add diagonal in trapezoid before/after the cusp

Removing cusps



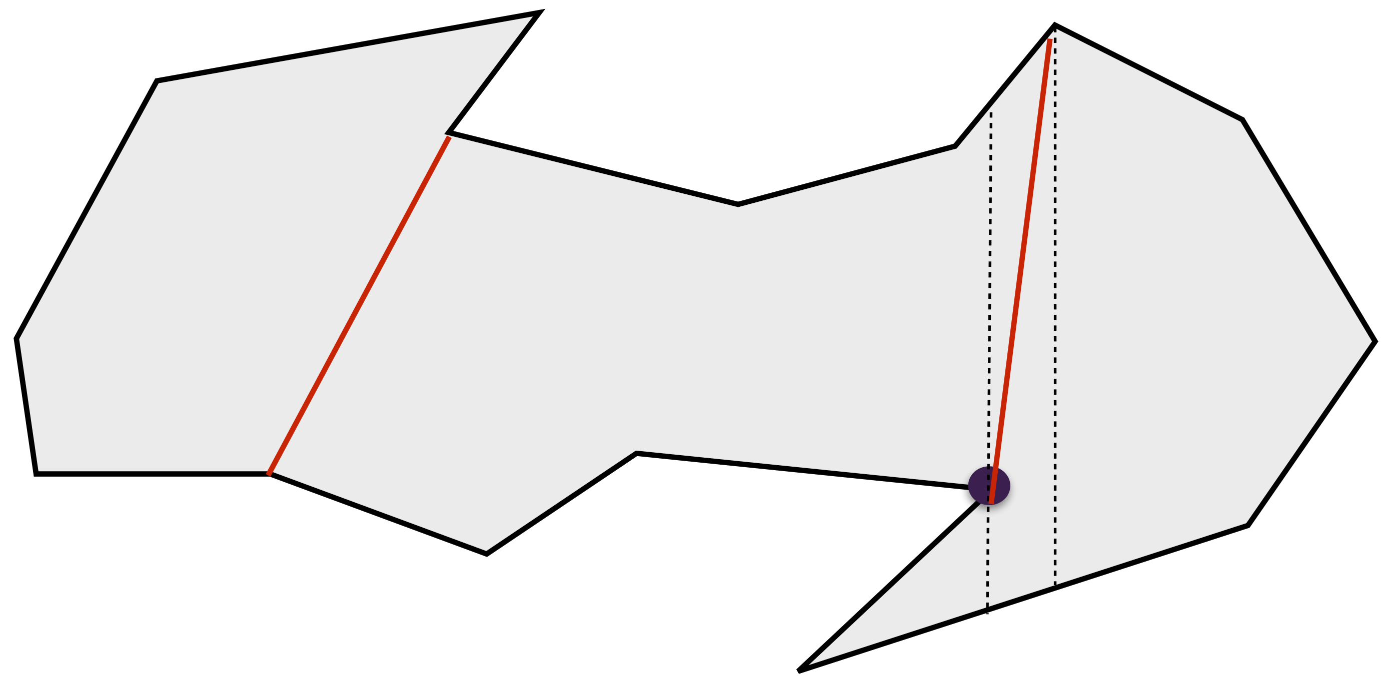


Removing cusps



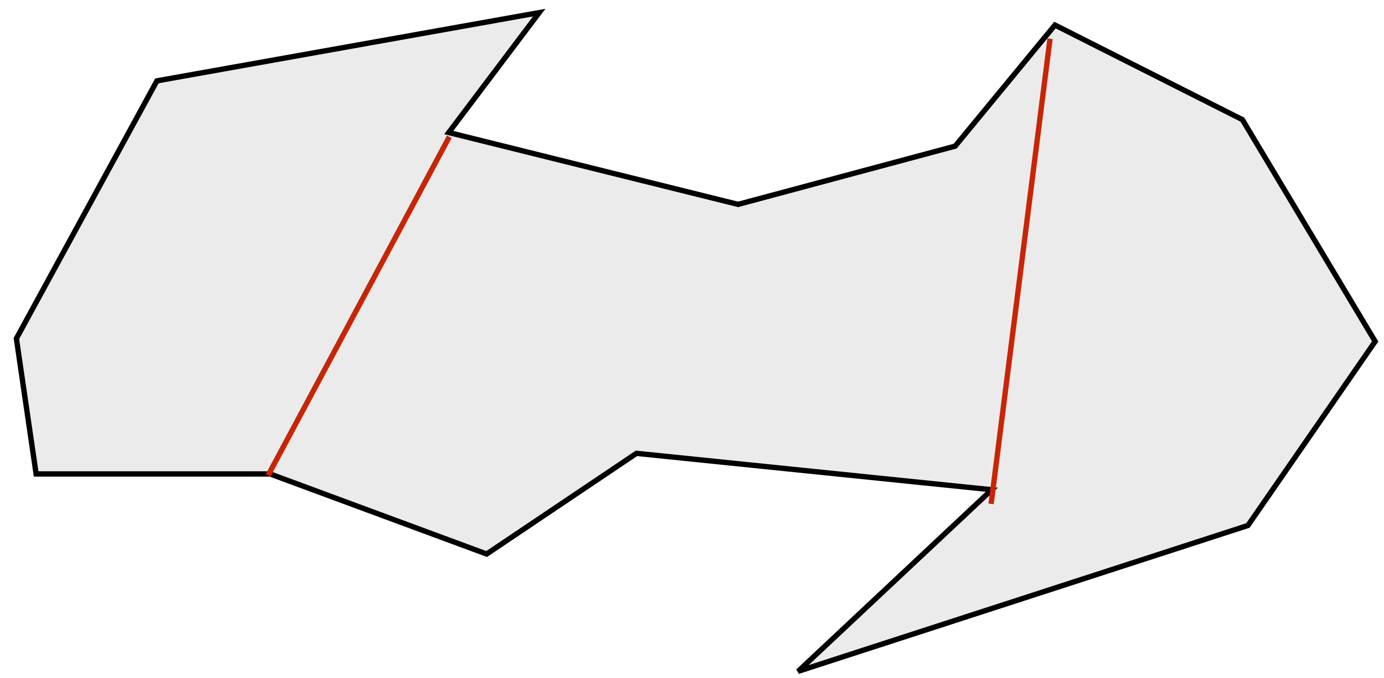
1. Compute a trapezoid partition of P
2. Identify cusp vertices
3. For each cusp vertex, add diagonal in trapezoid before/after the cusp

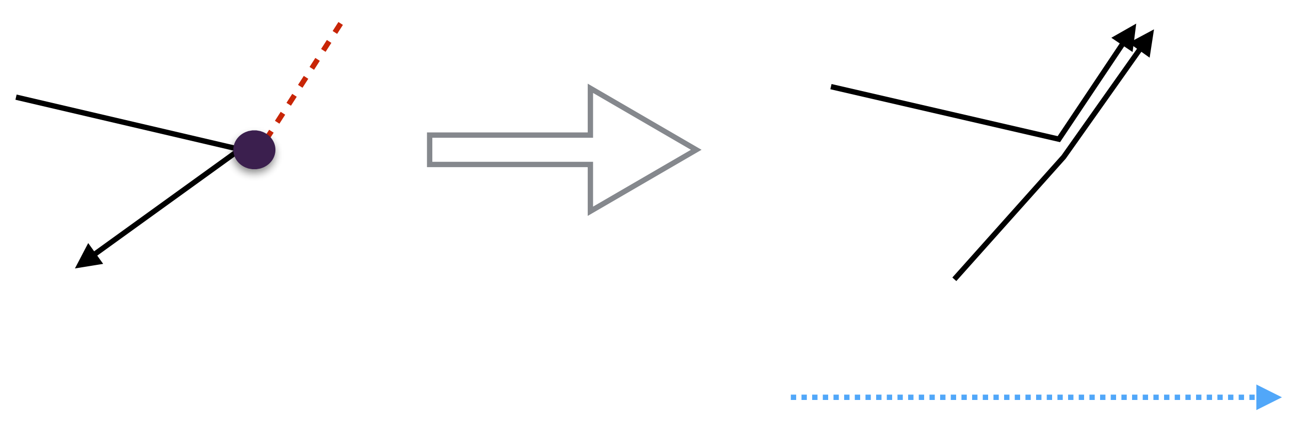
Removing cusps



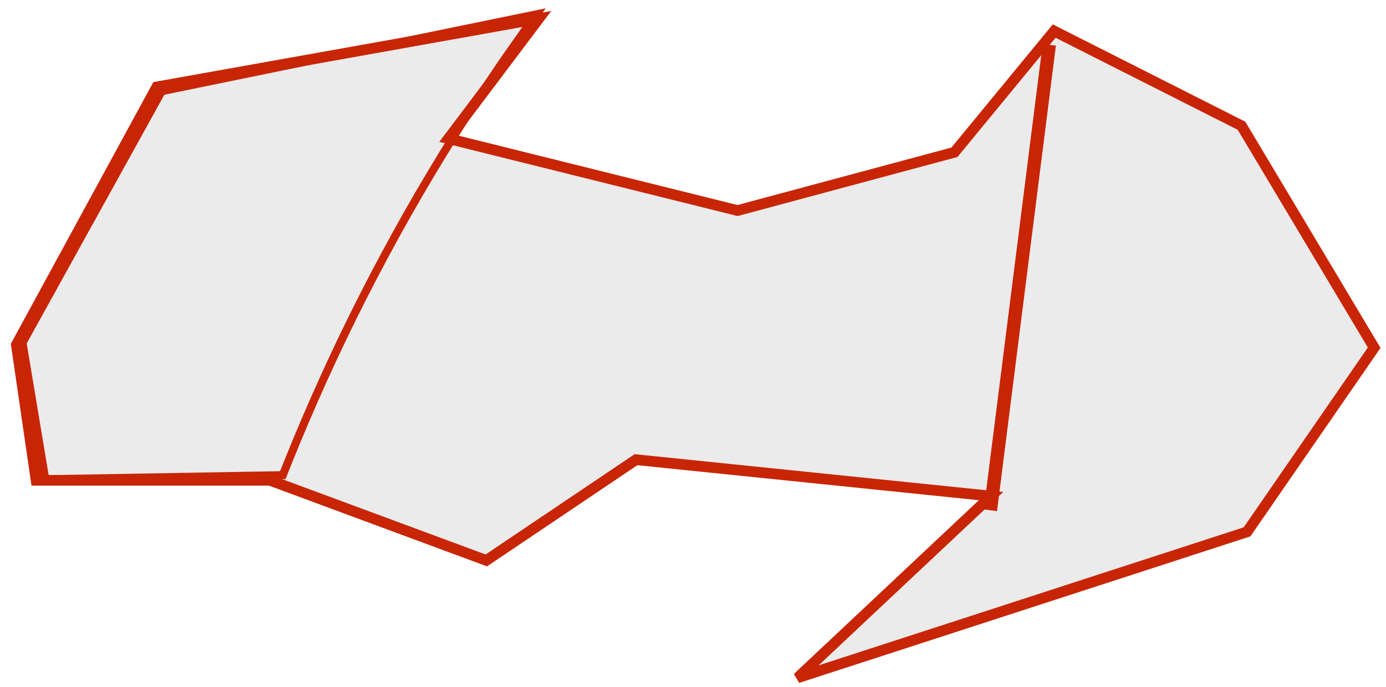
1. Compute a trapezoid partition of P
2. Identify cusp vertices
3. For each cusp vertex, add diagonal in trapezoid before/after the cusp

Removing cusps





Removing cusps



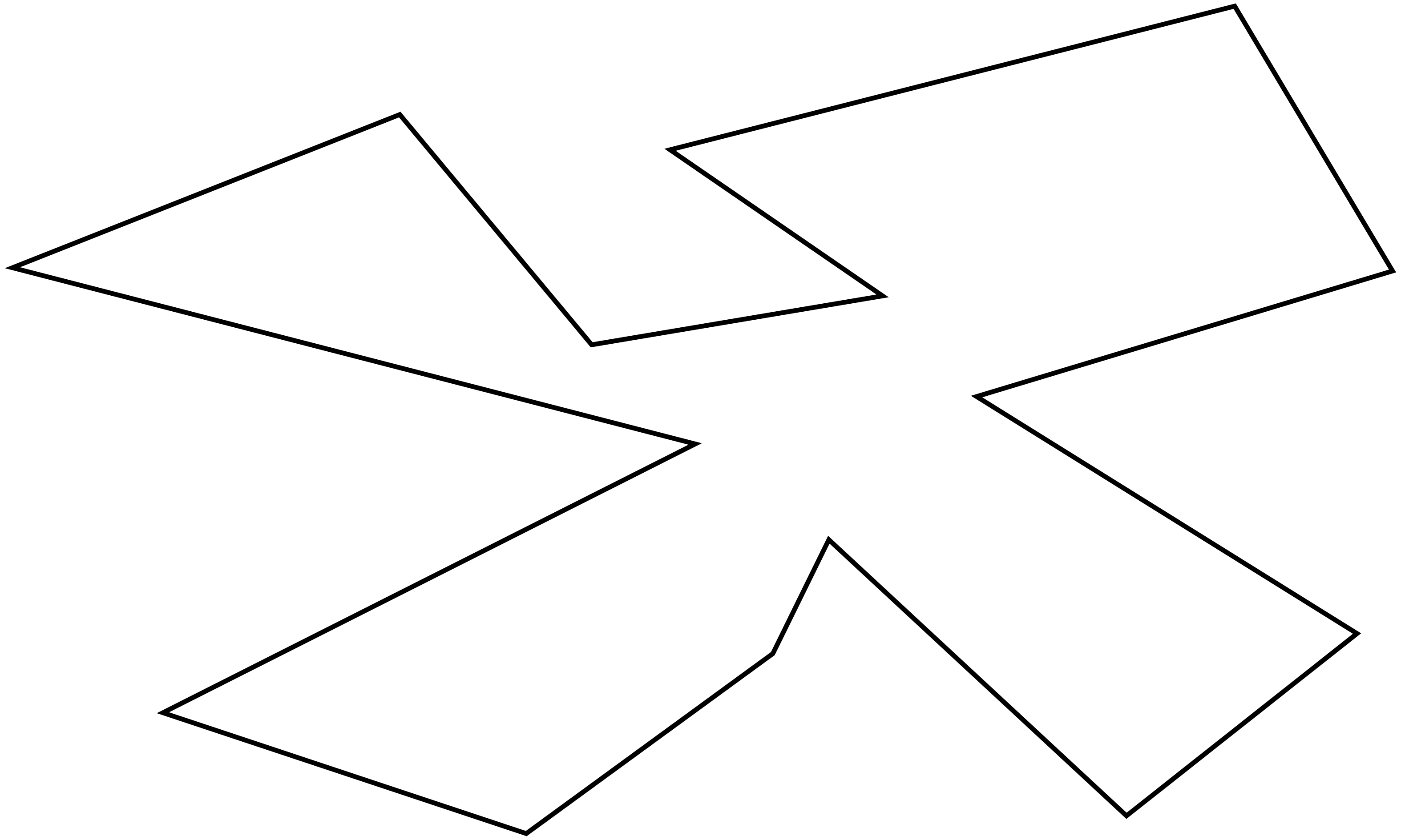
1. Compute a trapezoid partition of P
2. Identify cusp vertices
3. For each cusp vertex, add diagonal in trapezoid before/after the cusp

This creates a partition of P.

The resulting polygons have no cusps and thus are monotone (by theorem).

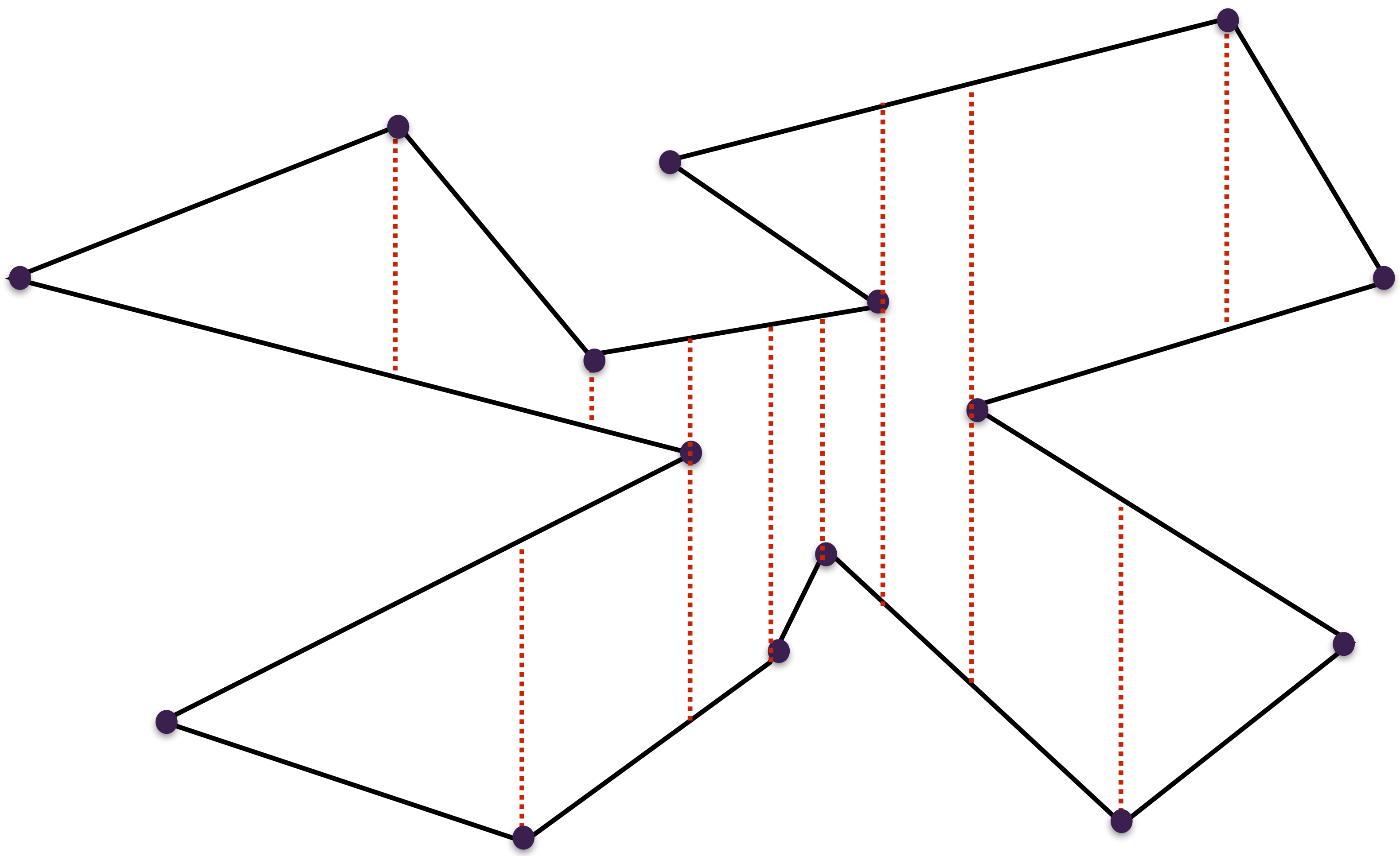
Removing cusps

- Another example



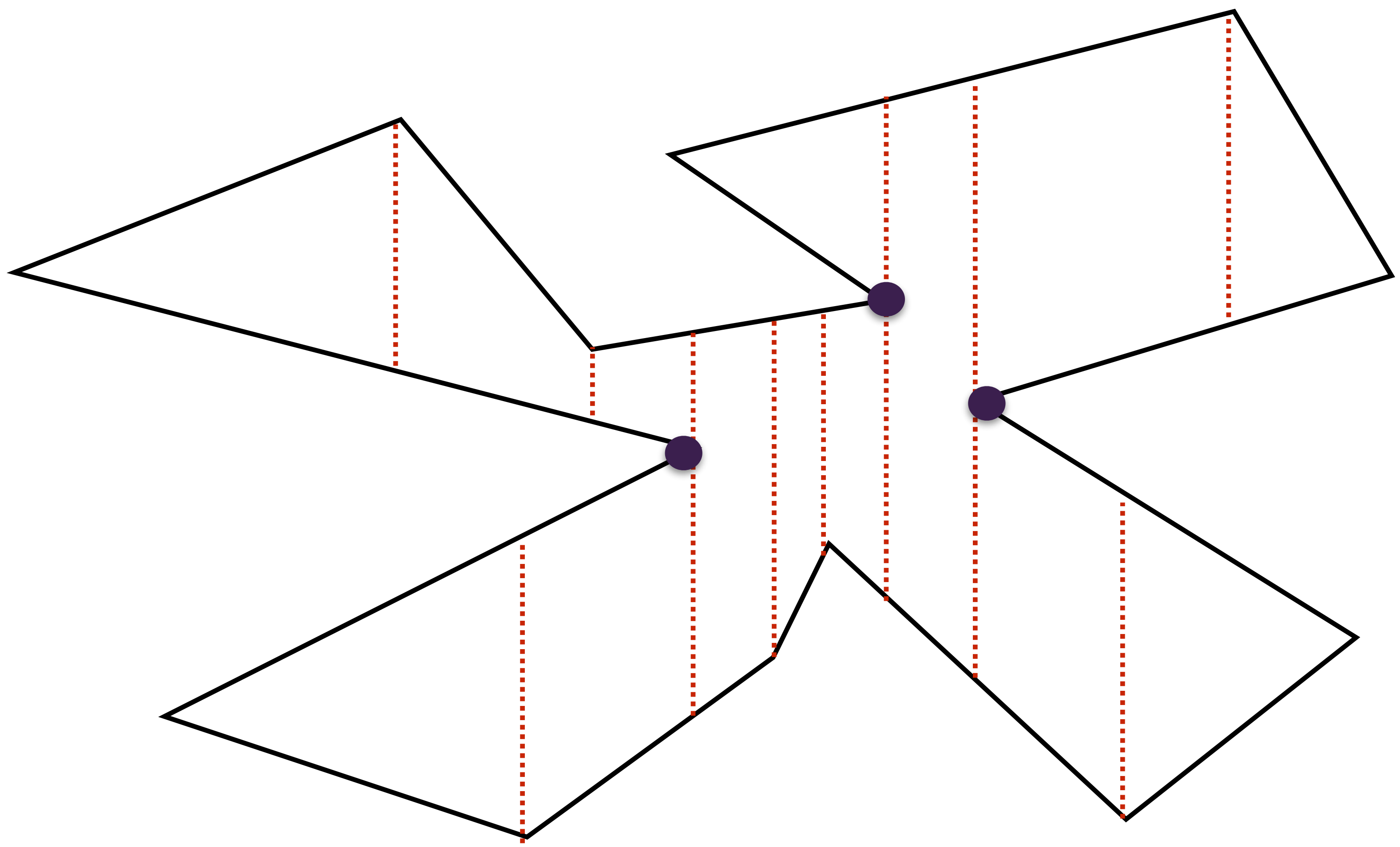
Removing cusps

- Another example



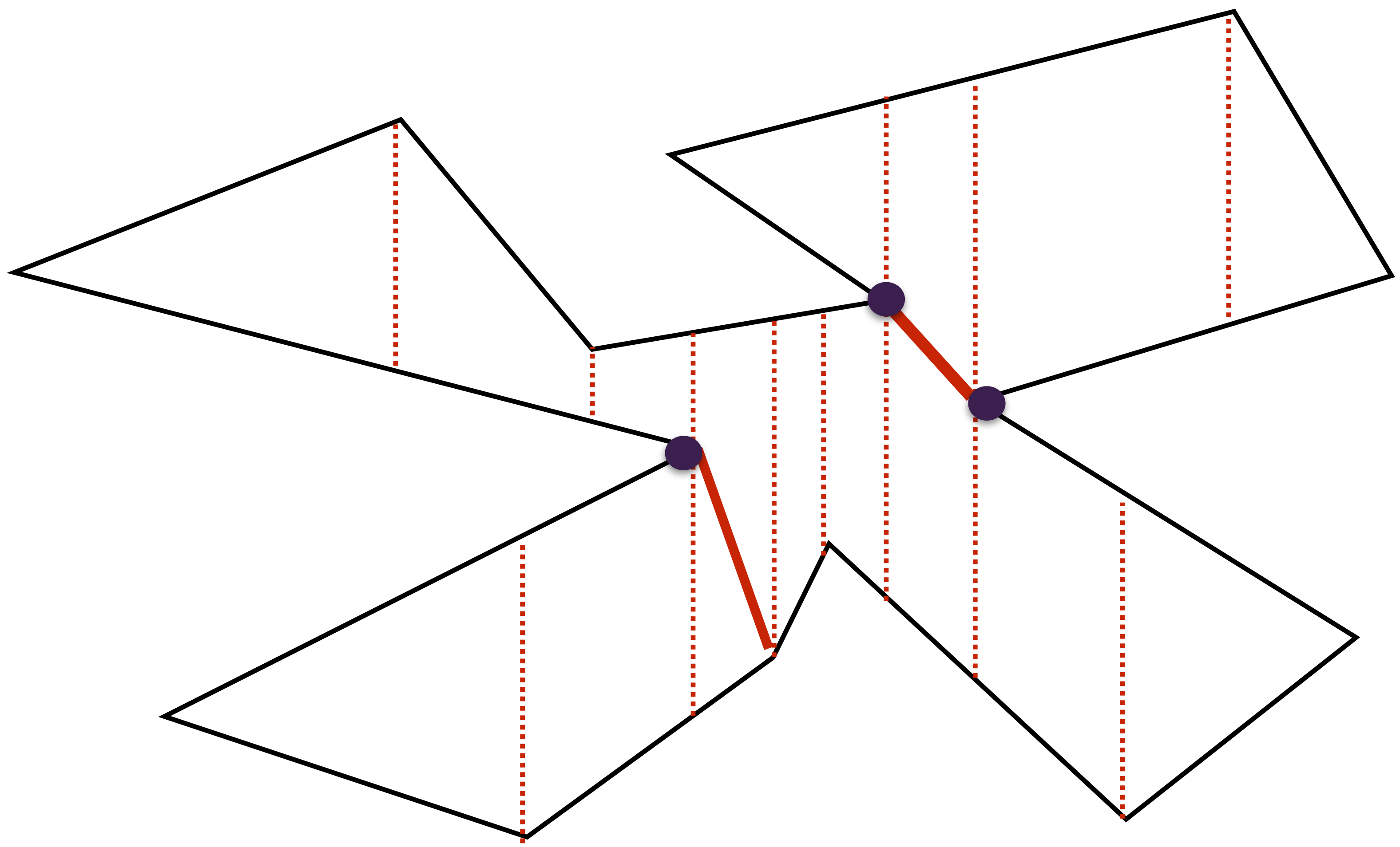
1. Compute a trapezoid partition of P

Removing cusps



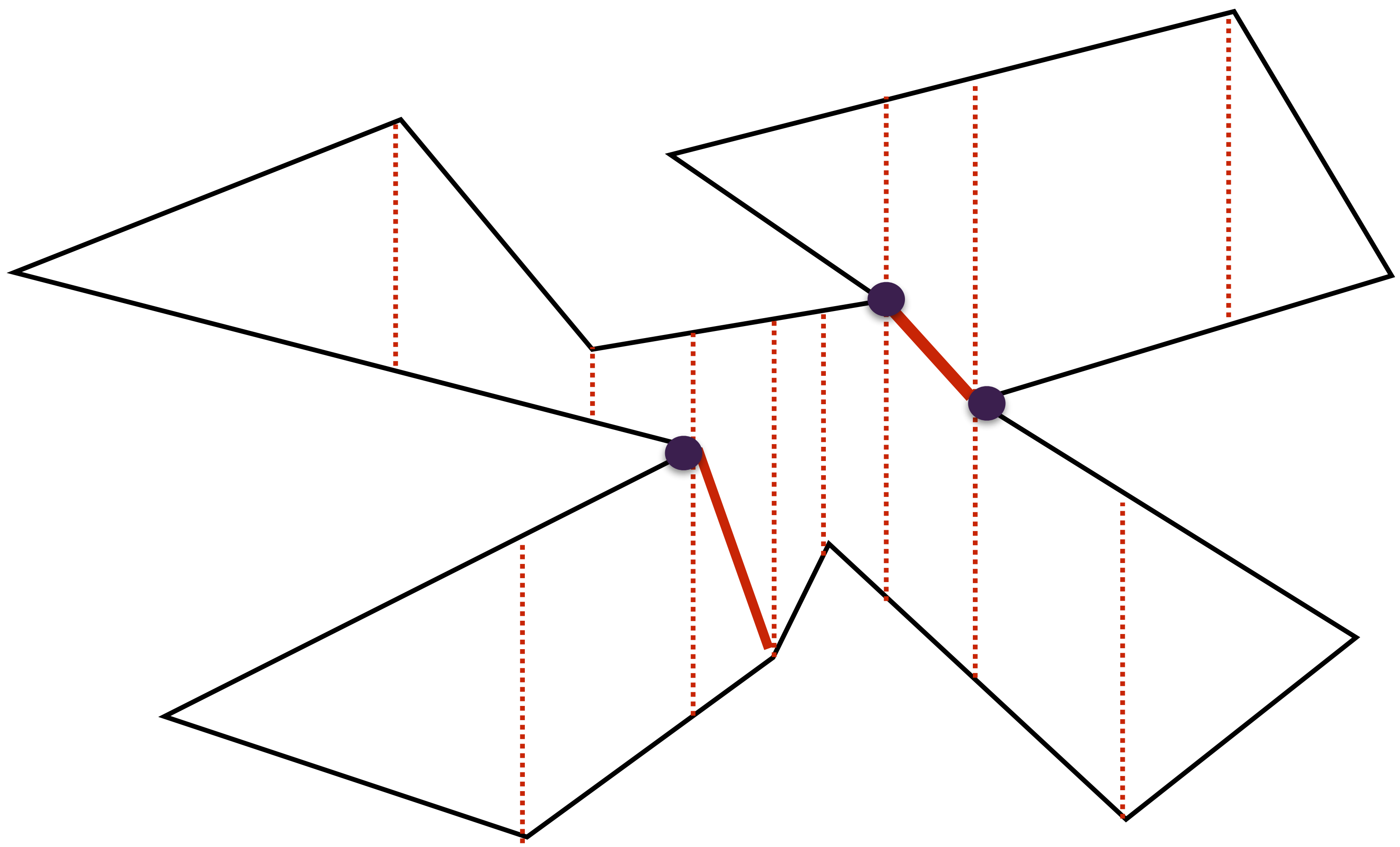
1. Compute a trapezoid partition of P
2. Identify cusp vertices

Removing cusps



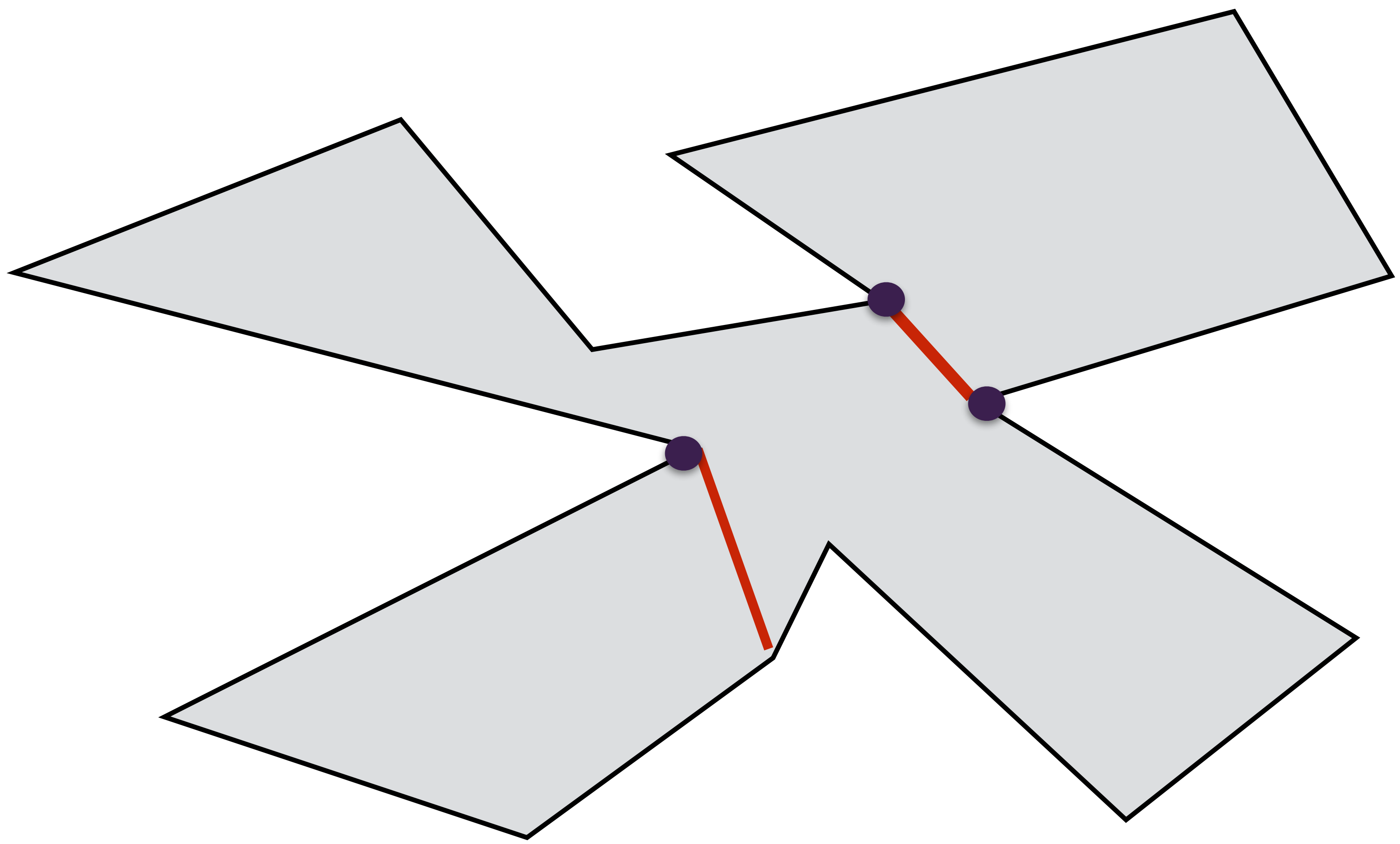
1. Compute a trapezoid partition of P
2. Identify cusp vertices
3. Add obvious diagonal before/after each cusp

Removing cusps



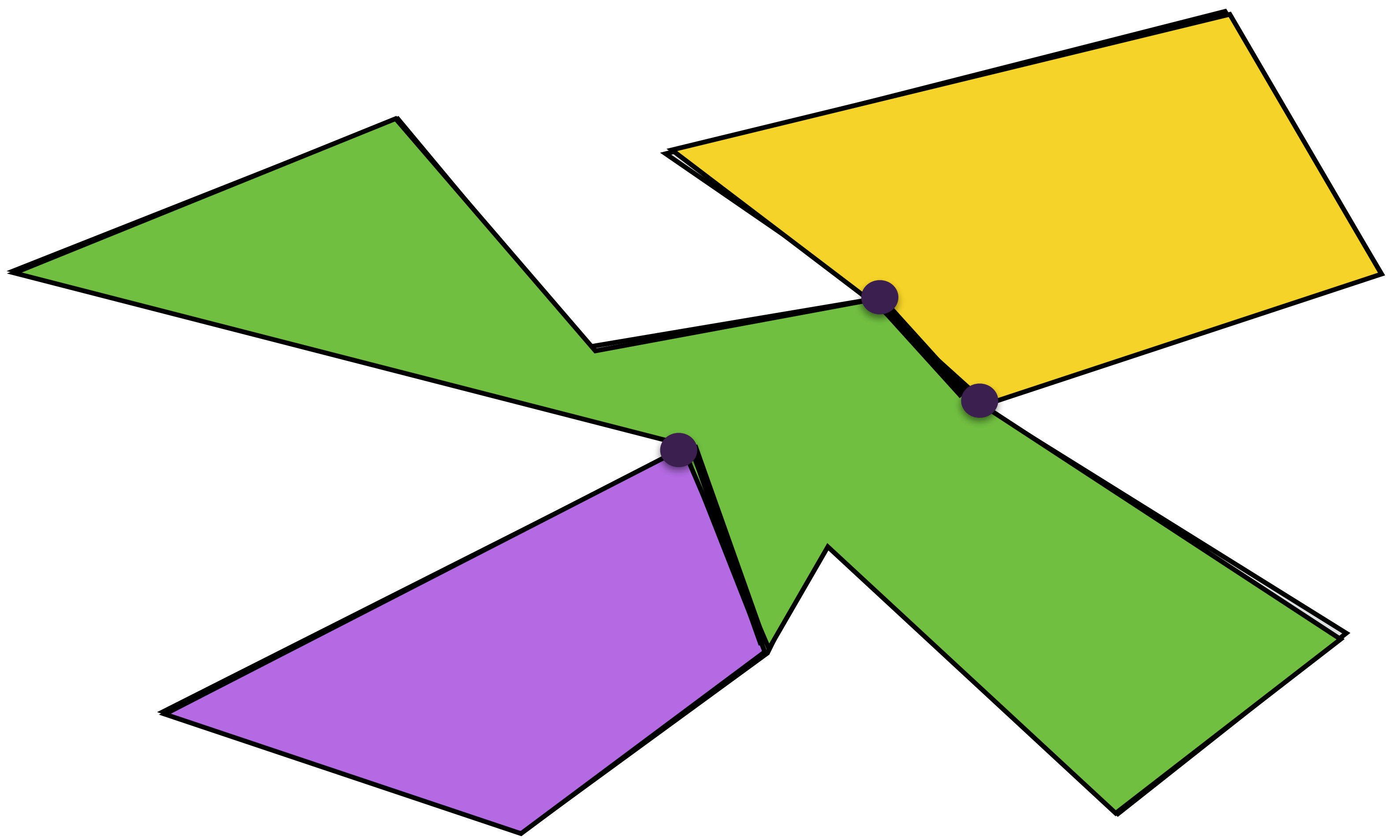
This partitions the polygon into monotone pieces.

Removing cusps



This partitions the polygon into monotone pieces.

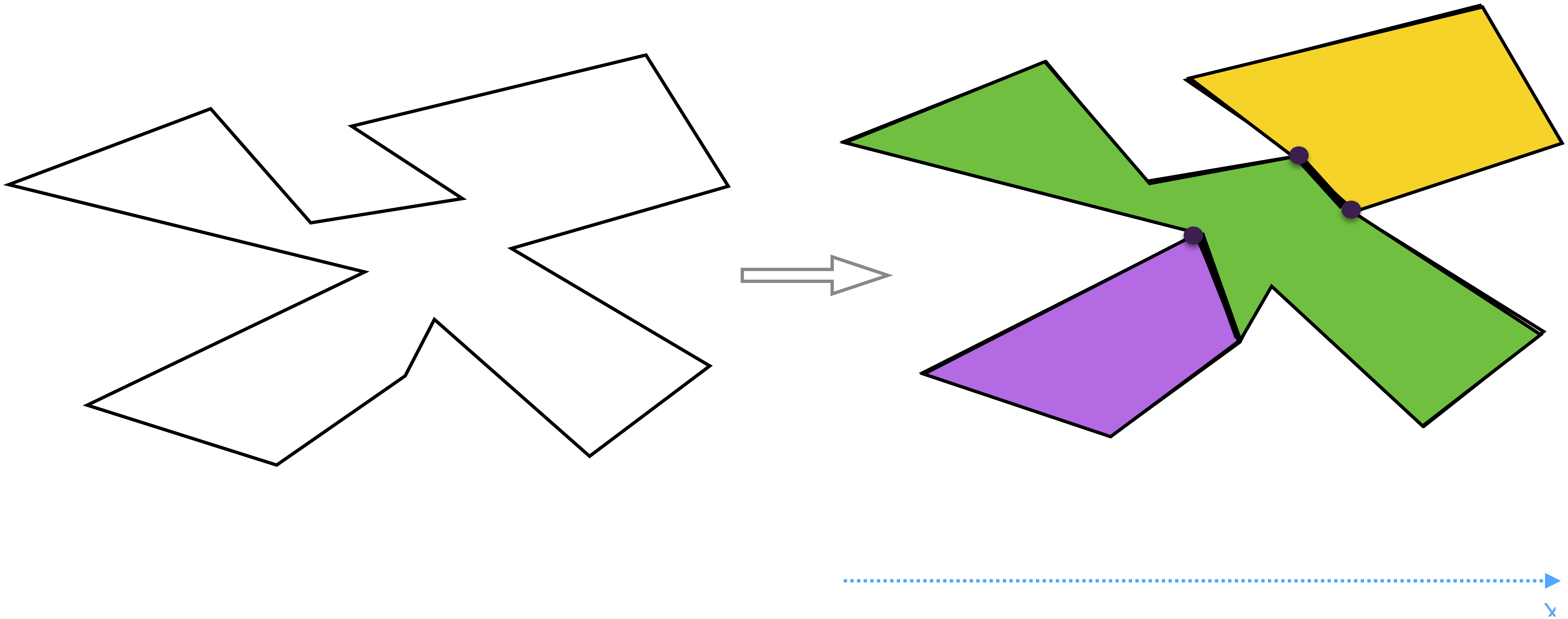
Removing cusps



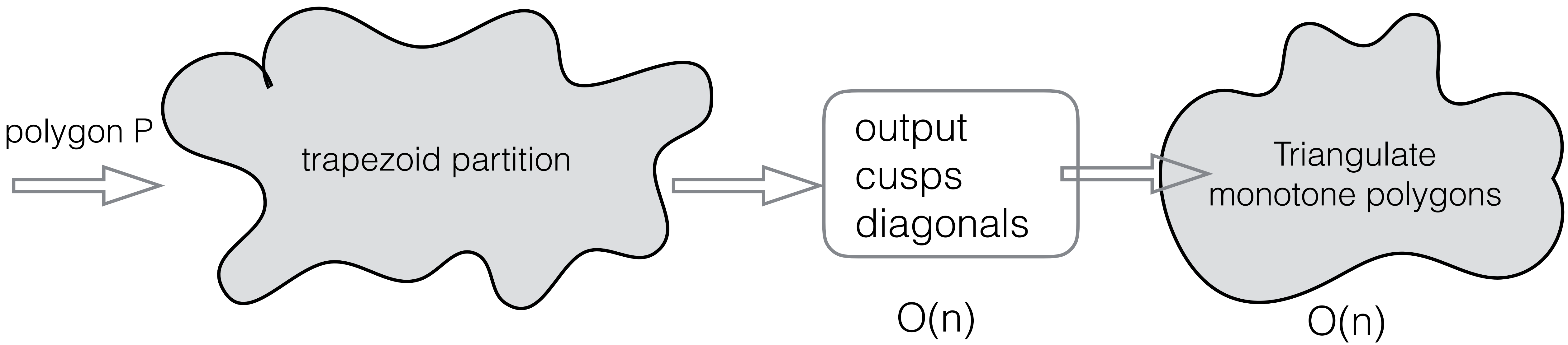
This partitions the polygon into monotone pieces.

Partition P into monotone polygons

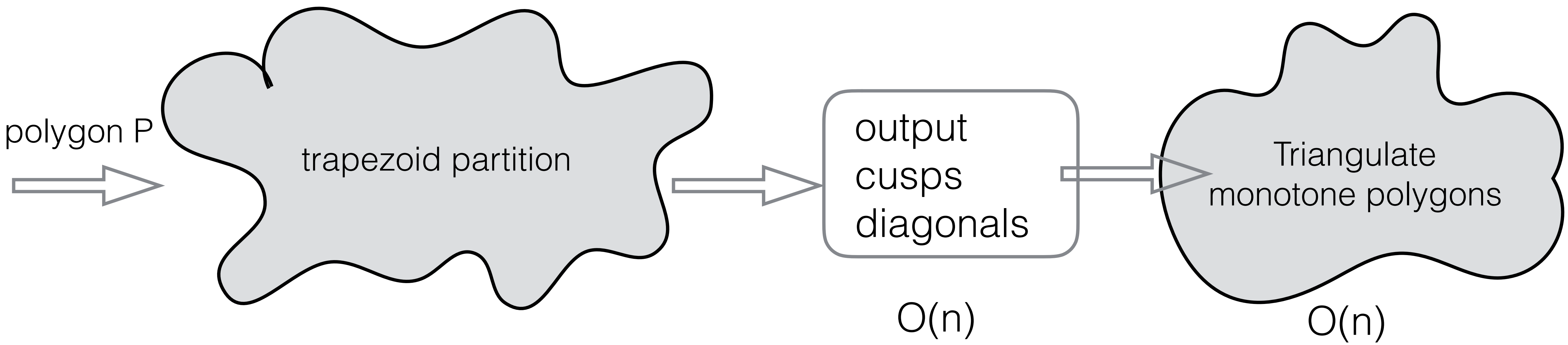
1. Compute a trapezoid partition of P
2. Identify cusp vertices
3. Add obvious diagonal before/after each cusp



An O(n lg n) Polygon Triangulation Algorithm

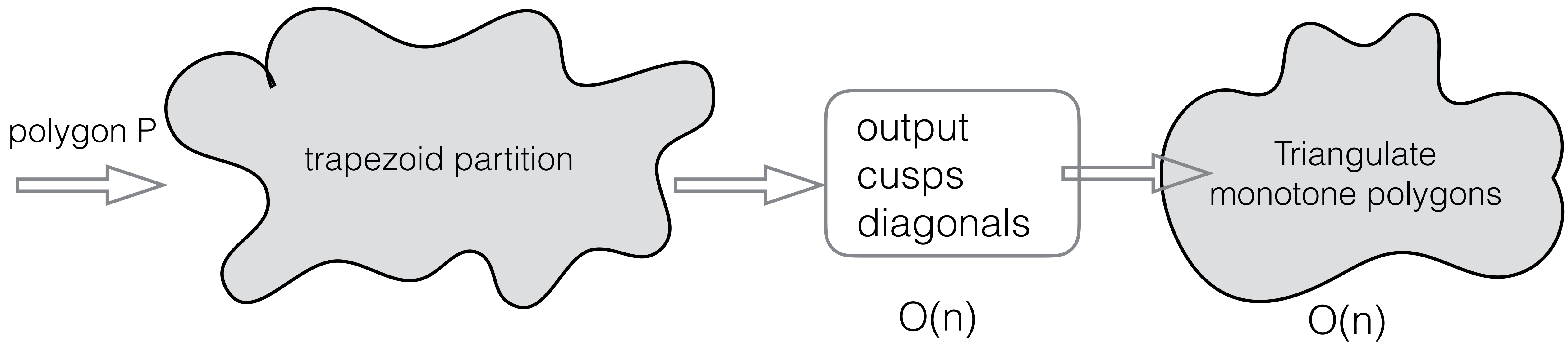


An O(n lg n) Polygon Triangulation Algorithm



Given a trapezoid partition of P, we can triangulate it in O(n) time.

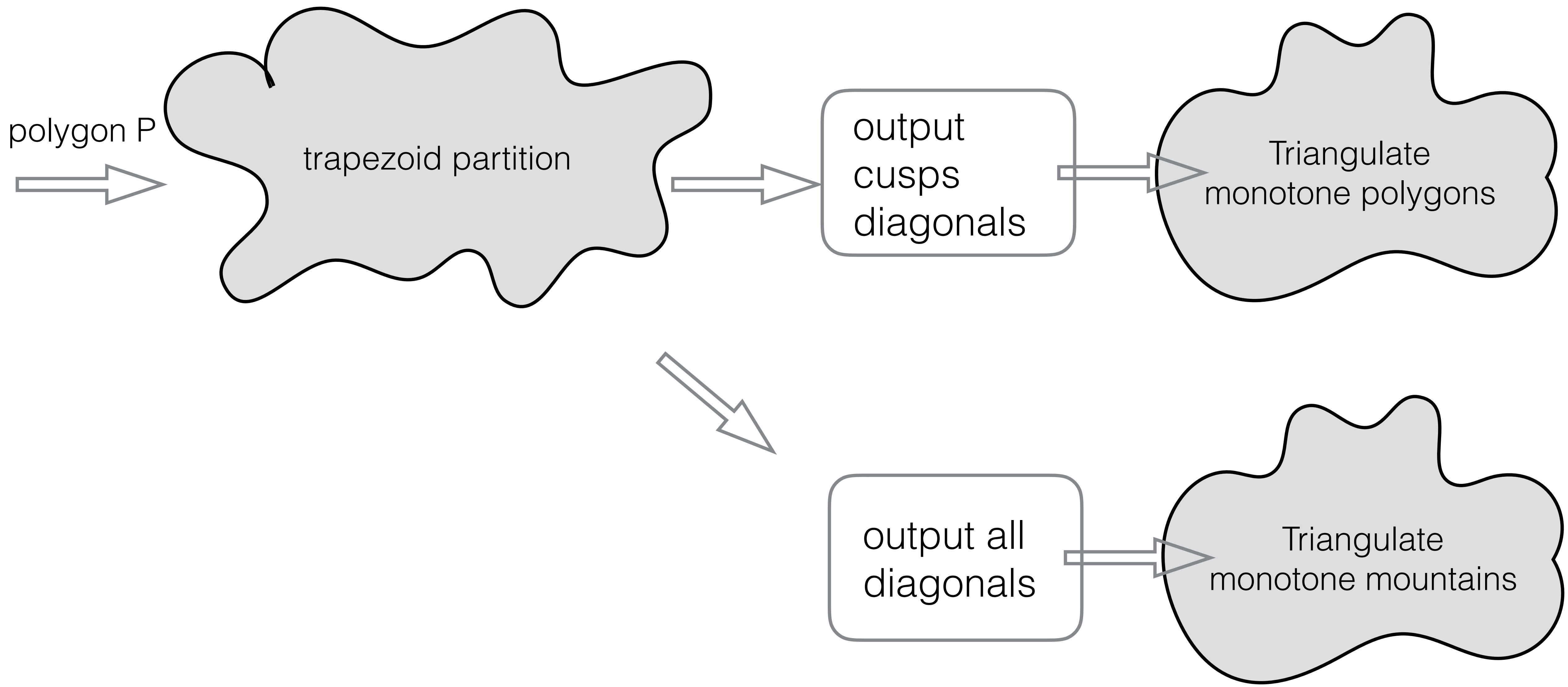
An O(n lg n) Polygon Triangulation Algorithm



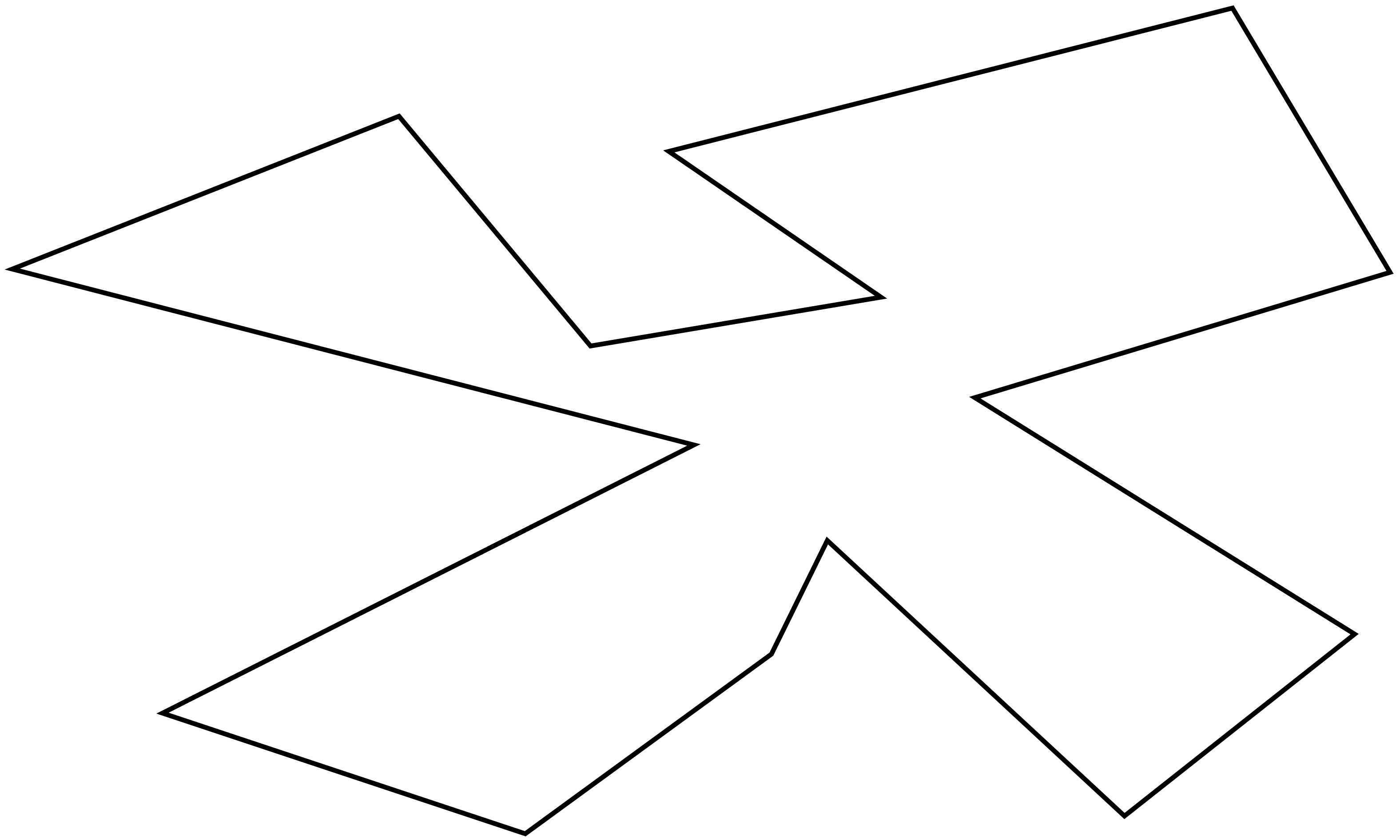
Given a trapezoid partition of P, we can triangulate it in O(n) time.

Actually there's even a simpler way to do this.

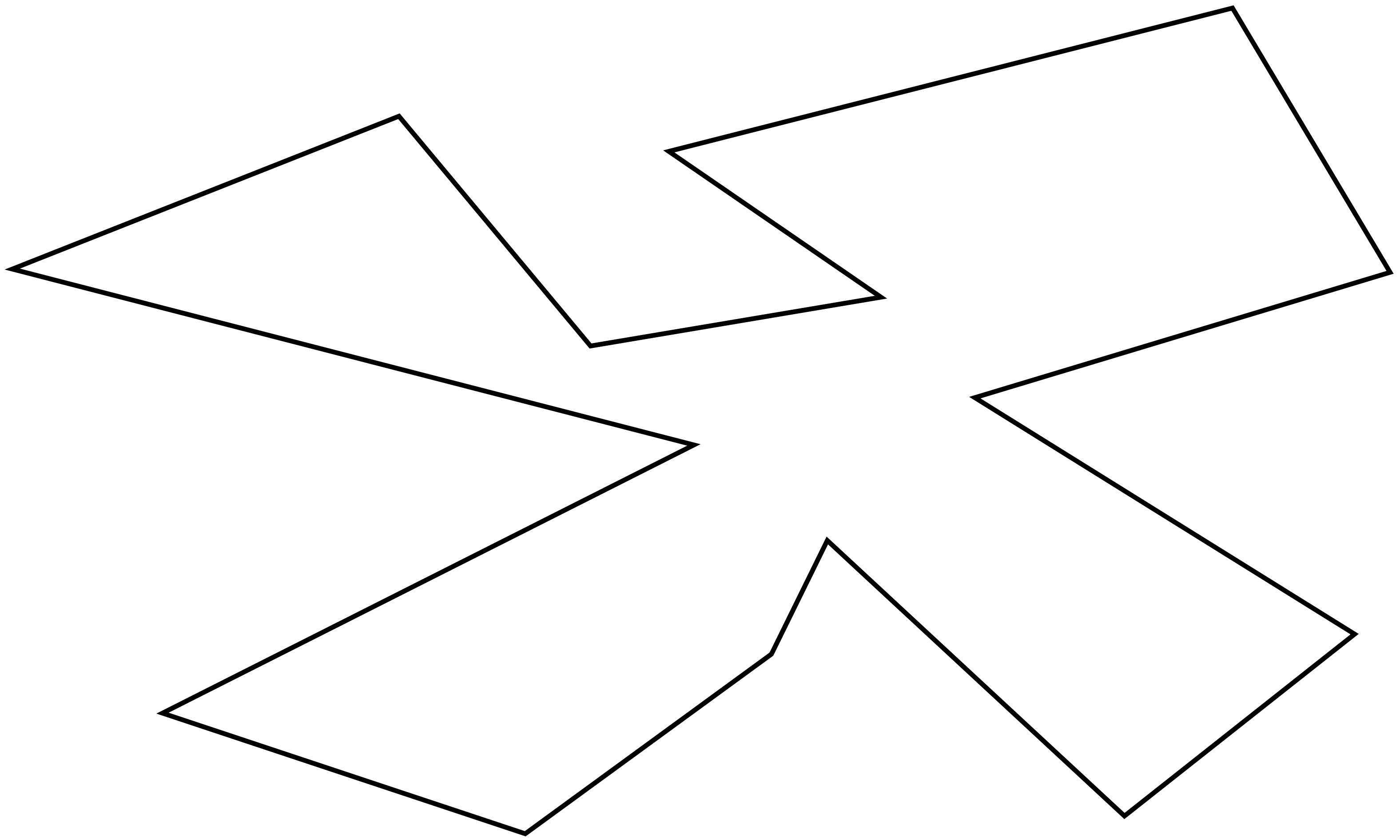
An O(n lg n) Polygon Triangulation Algorithm



Generating monotone mountains

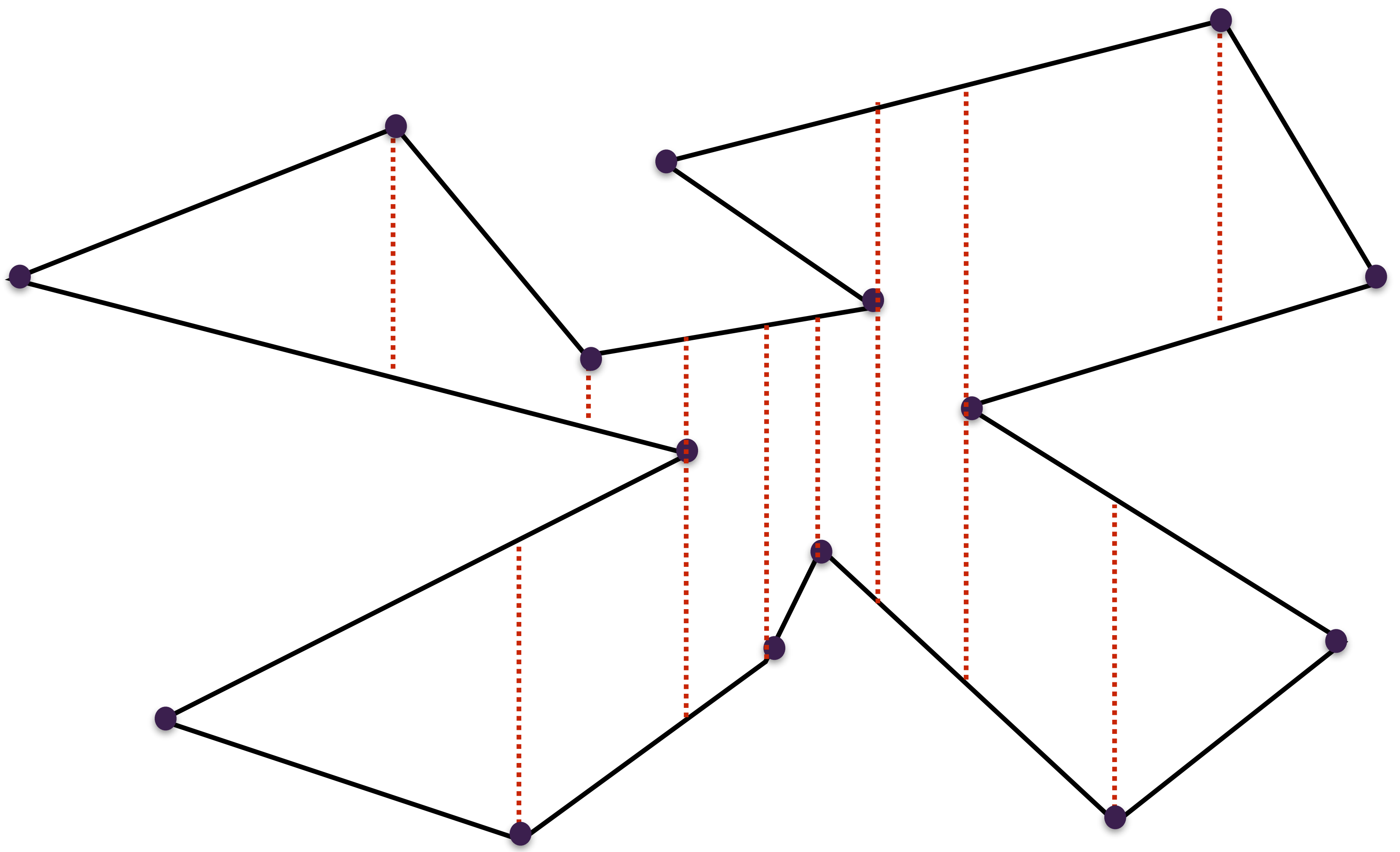


Generating monotone mountains



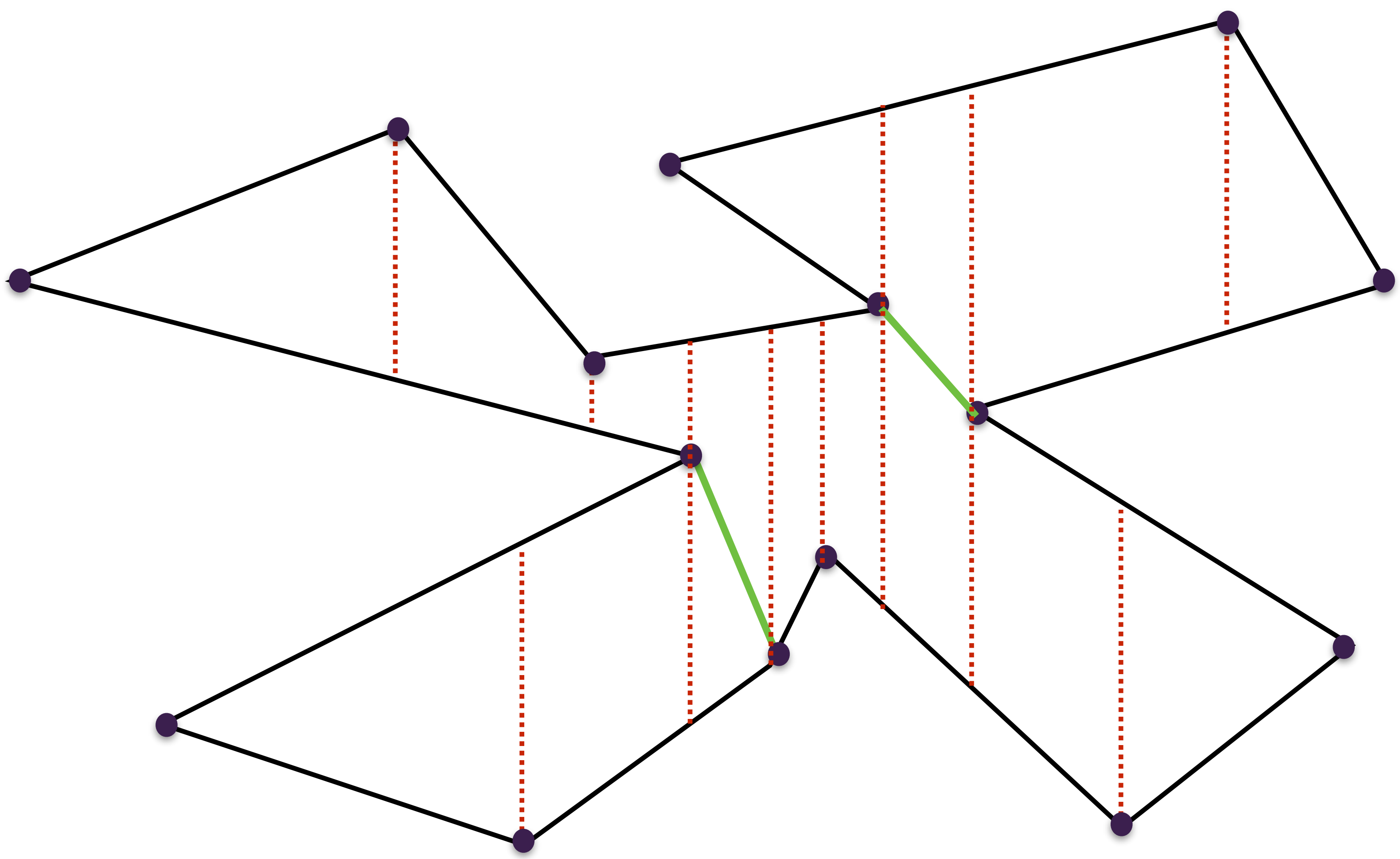
1. Compute a trapezoid partition of P

Generating monotone mountains



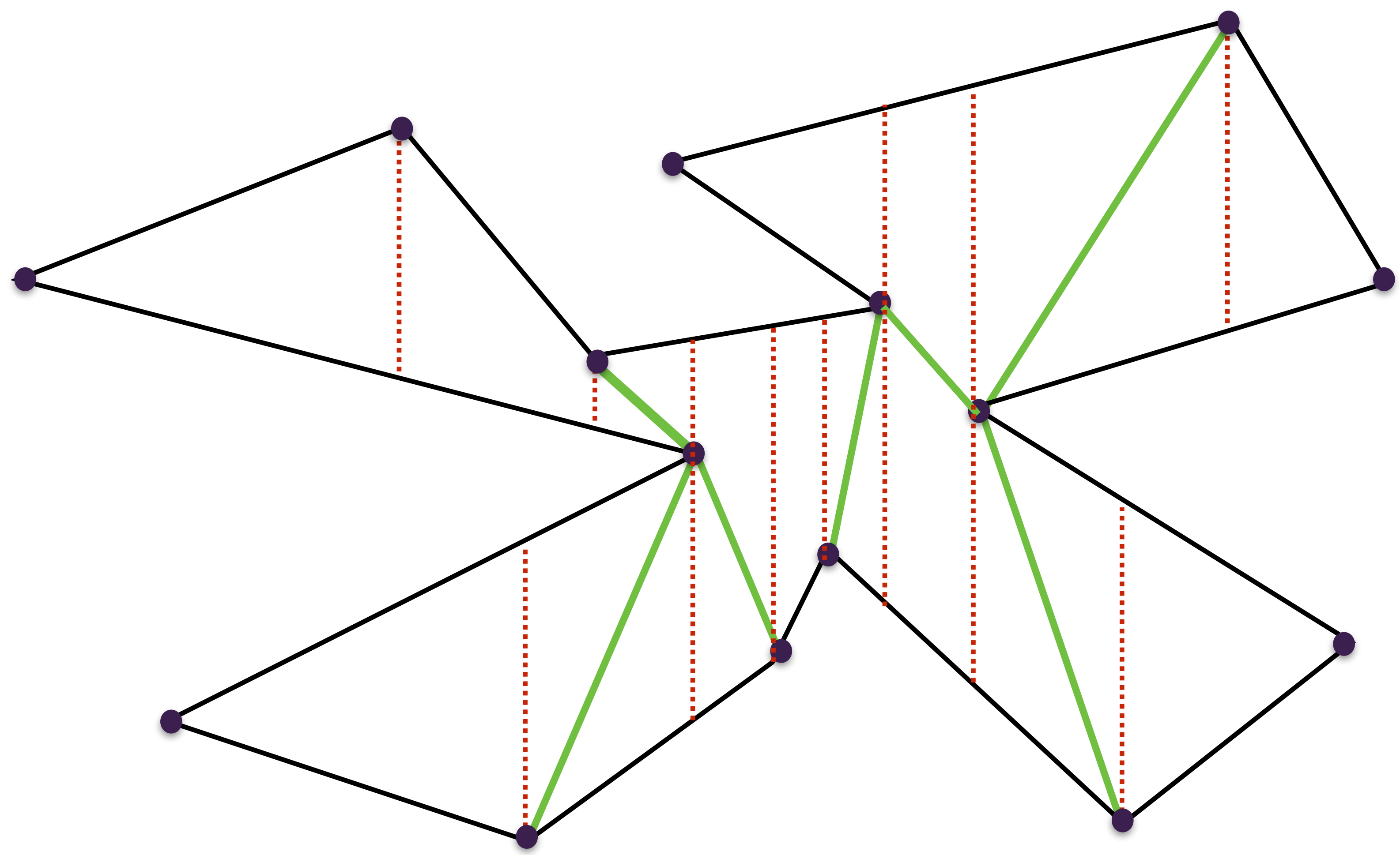
1. Compute a trapezoid partition of P

Generating monotone mountains



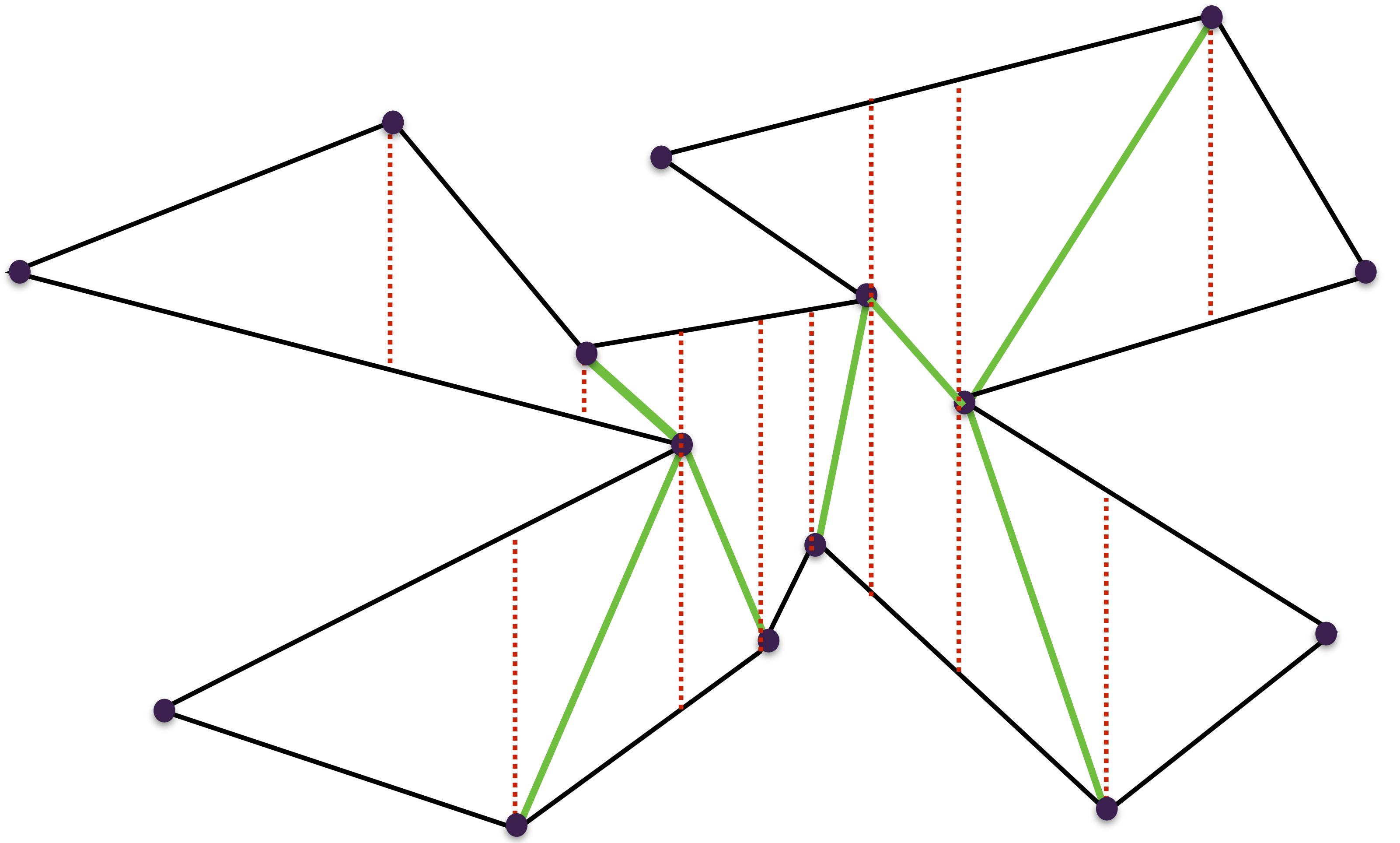
1. Compute a trapezoid partition of P
2. Output **all** diagonals.

Generating monotone mountains



1. Compute a trapezoid partition of P
2. Output **all** diagonals.

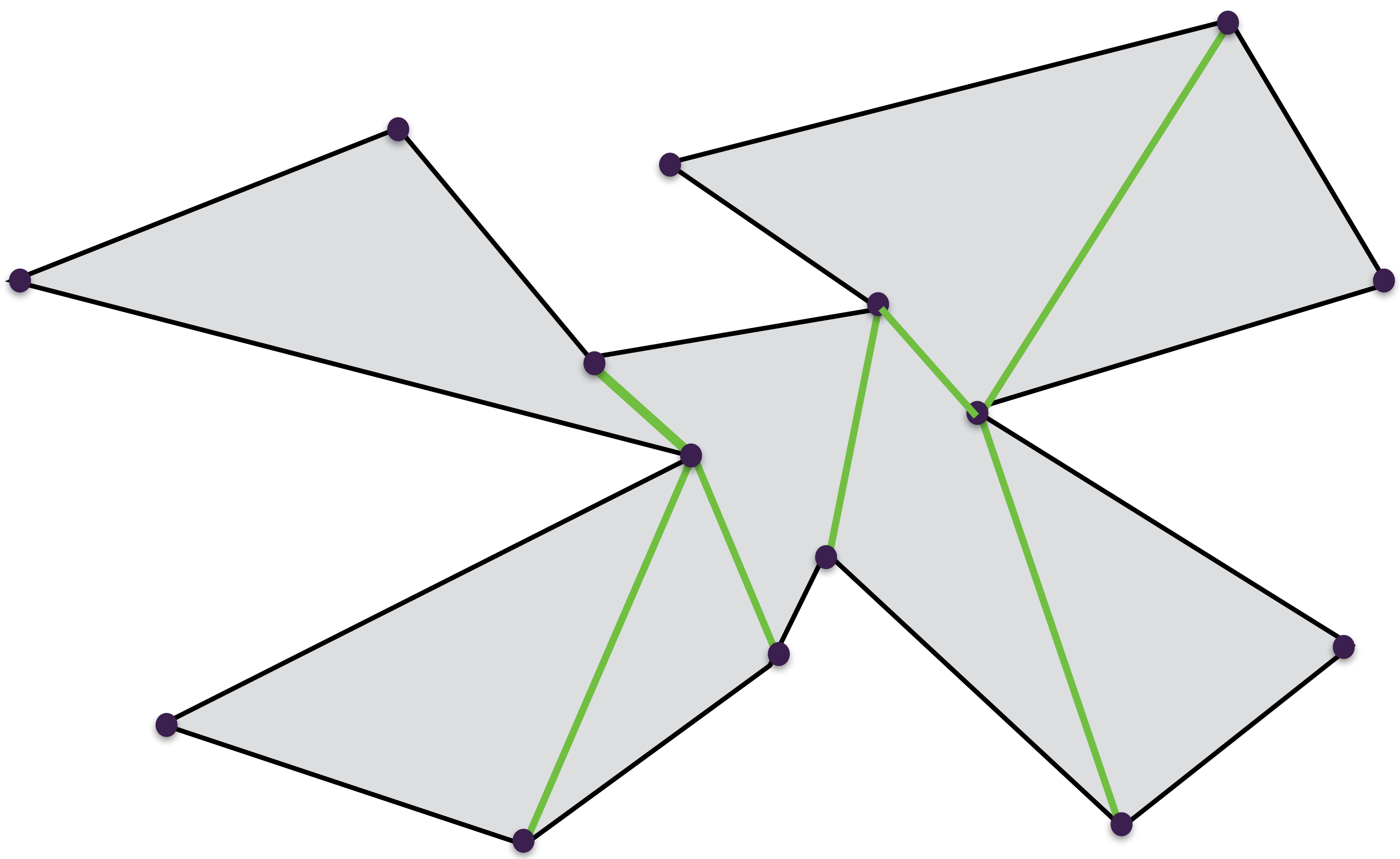
Generating monotone mountains



1. Compute a trapezoid partition of P
2. Output **all** diagonals.

   The diagonals partition the polygon into monotone mountains.

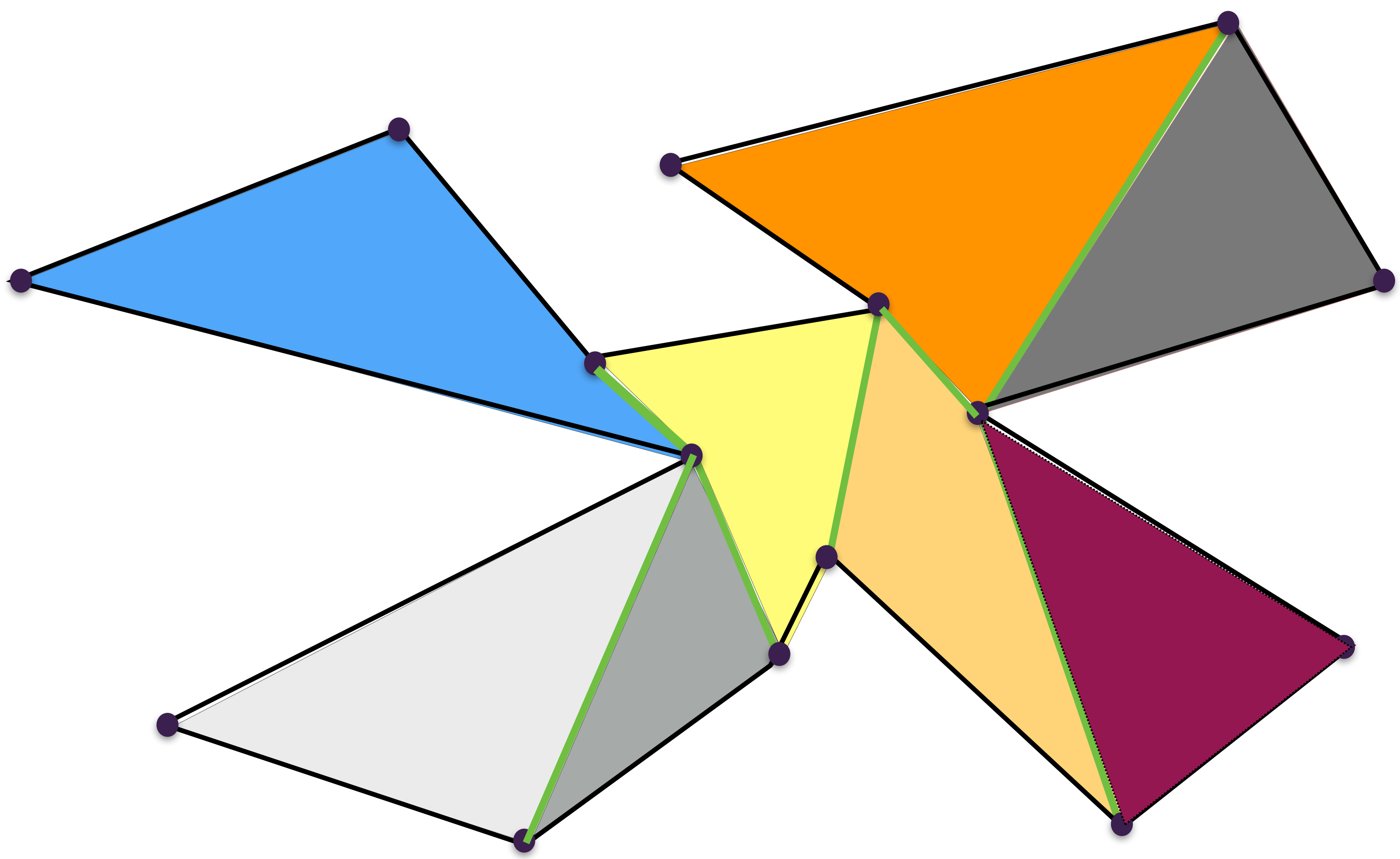
Generating monotone mountains



1. Compute a trapezoid partition of P
2. Output **all** diagonals.

   The diagonals partition the polygon into monotone mountains.

Generating monotone mountains

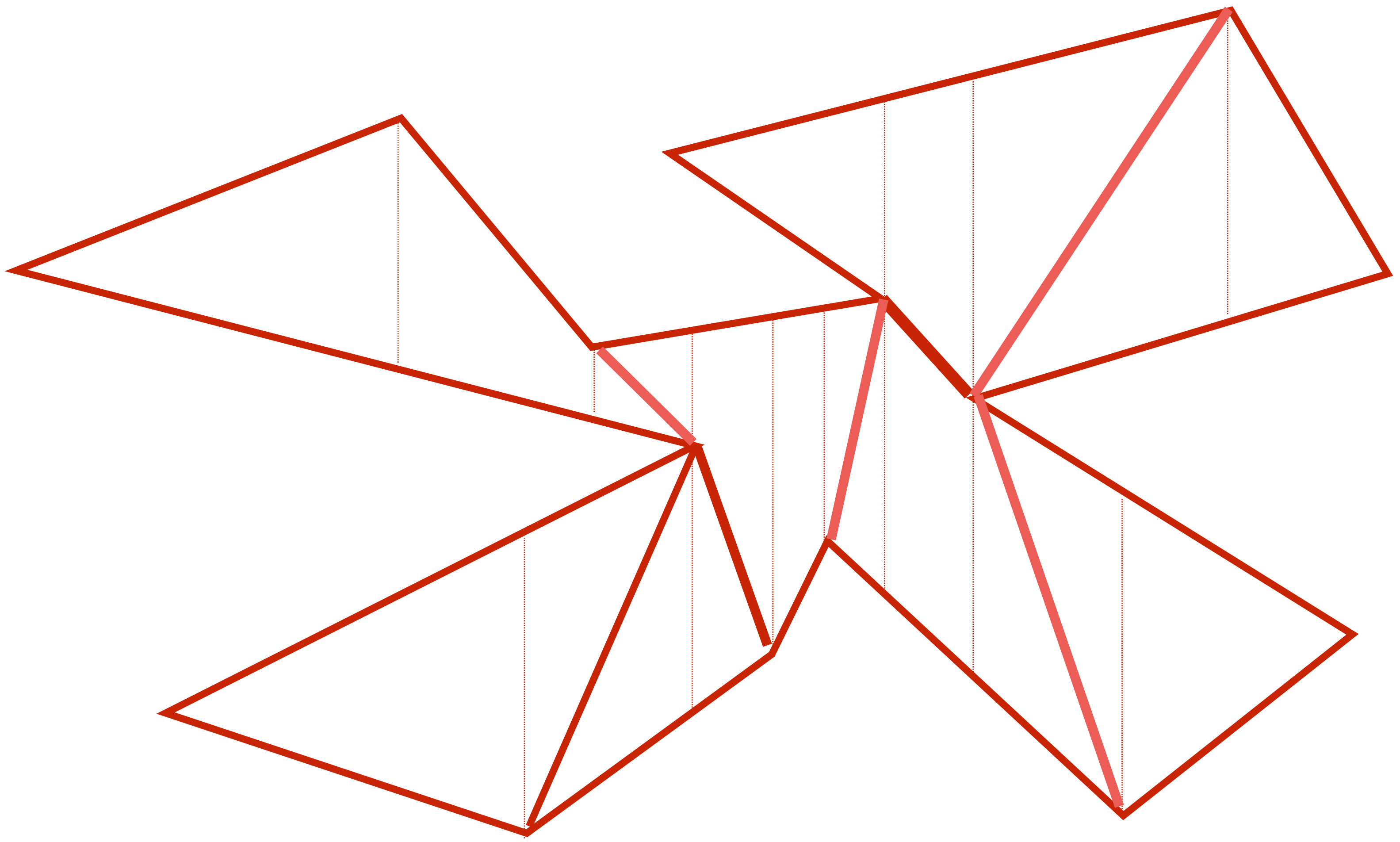


1. Compute a trapezoid partition of P
2. Output **all** diagonals.

   The diagonals partition the polygon into monotone mountains.

Generating monotone mountains

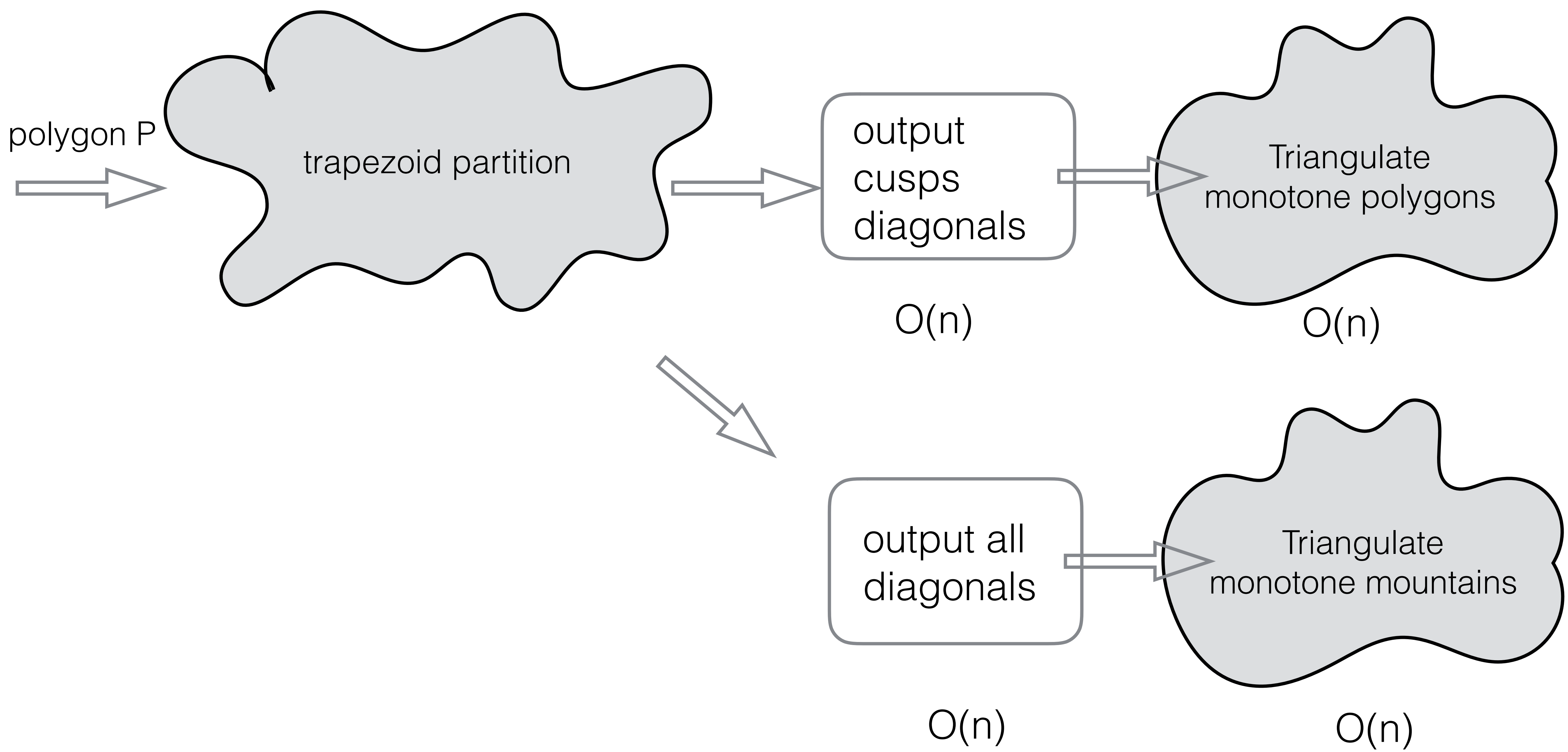
Claim: The diagonals partition the polygon into monotone mountains.



Proof:

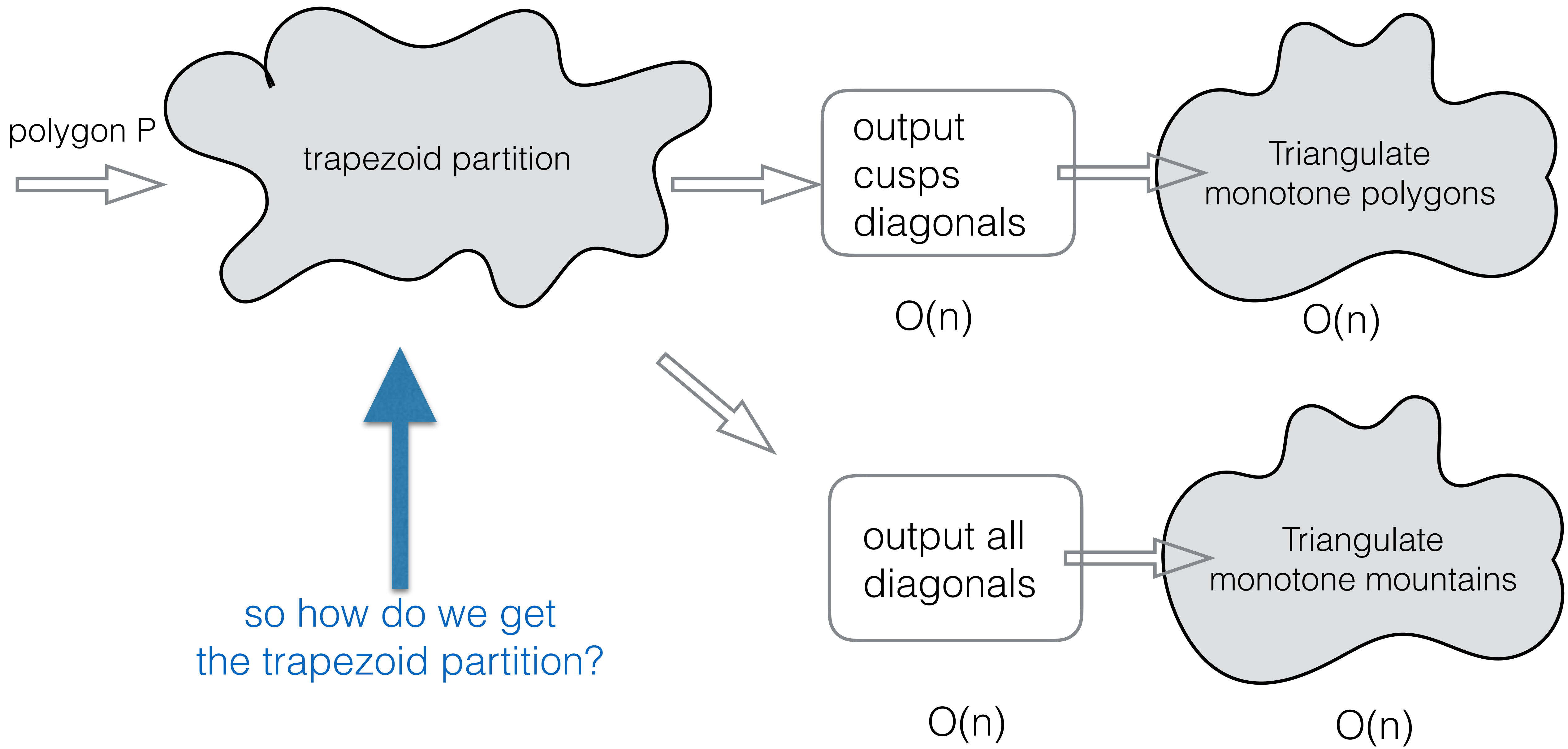
- Each polygon is monotone
- One of the chains must be a segment (because if it had another point, that point would generate a diagonal)

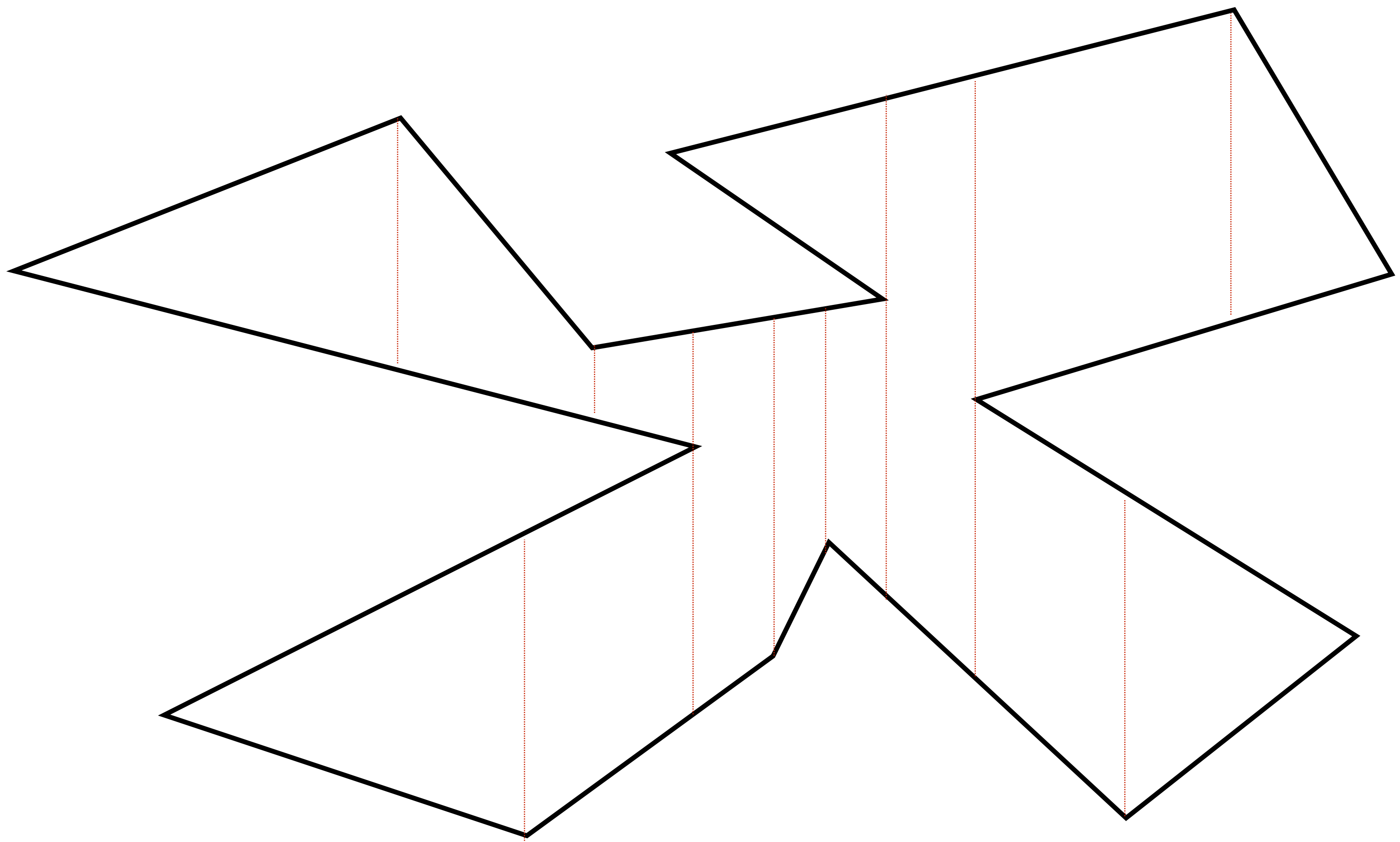
An O(n lg n) Polygon Triangulation Algorithm



Given a trapezoid partition of P, we can triangulate it in O(n) time.

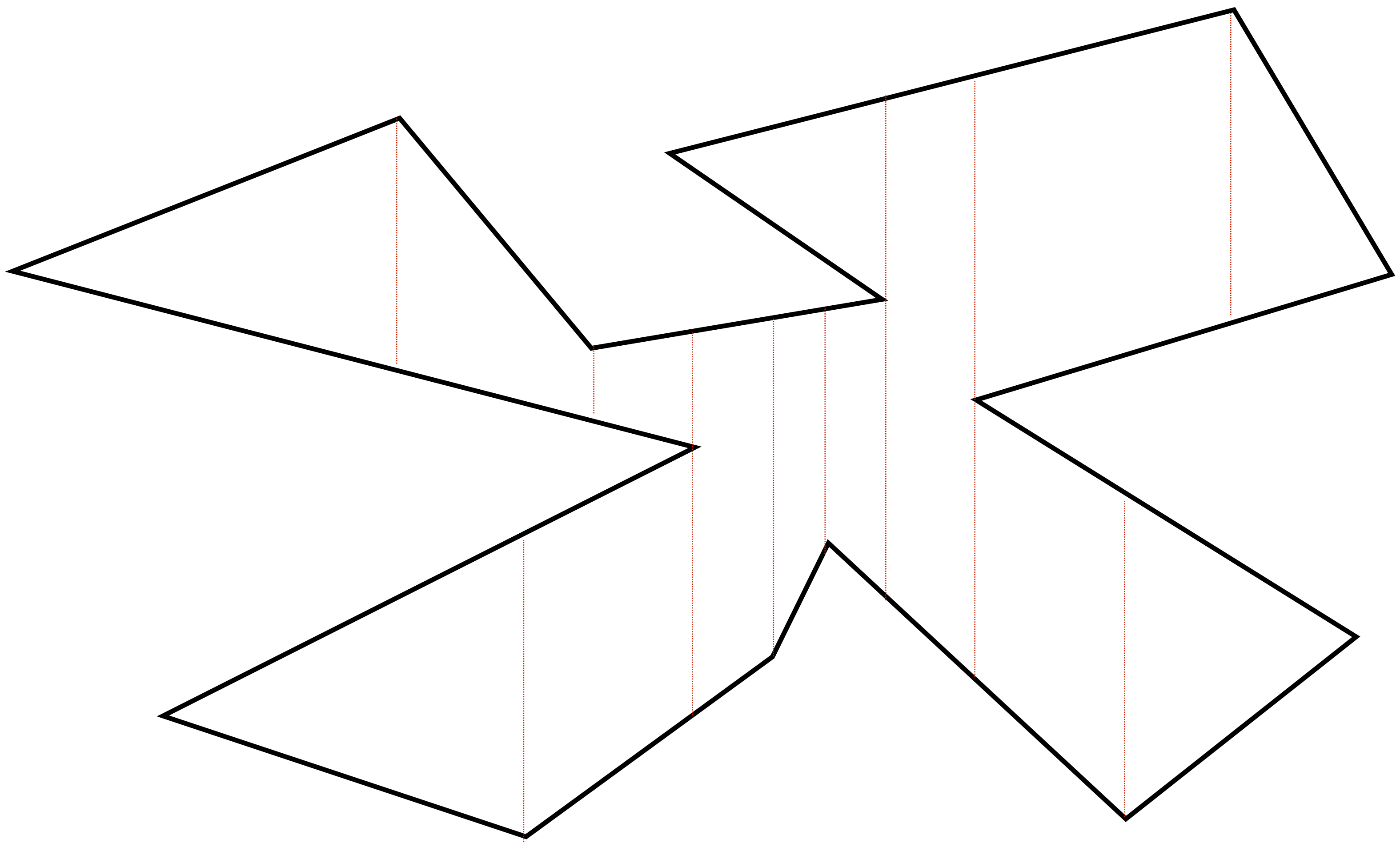
An O(n lg n) Polygon Triangulation Algorithm





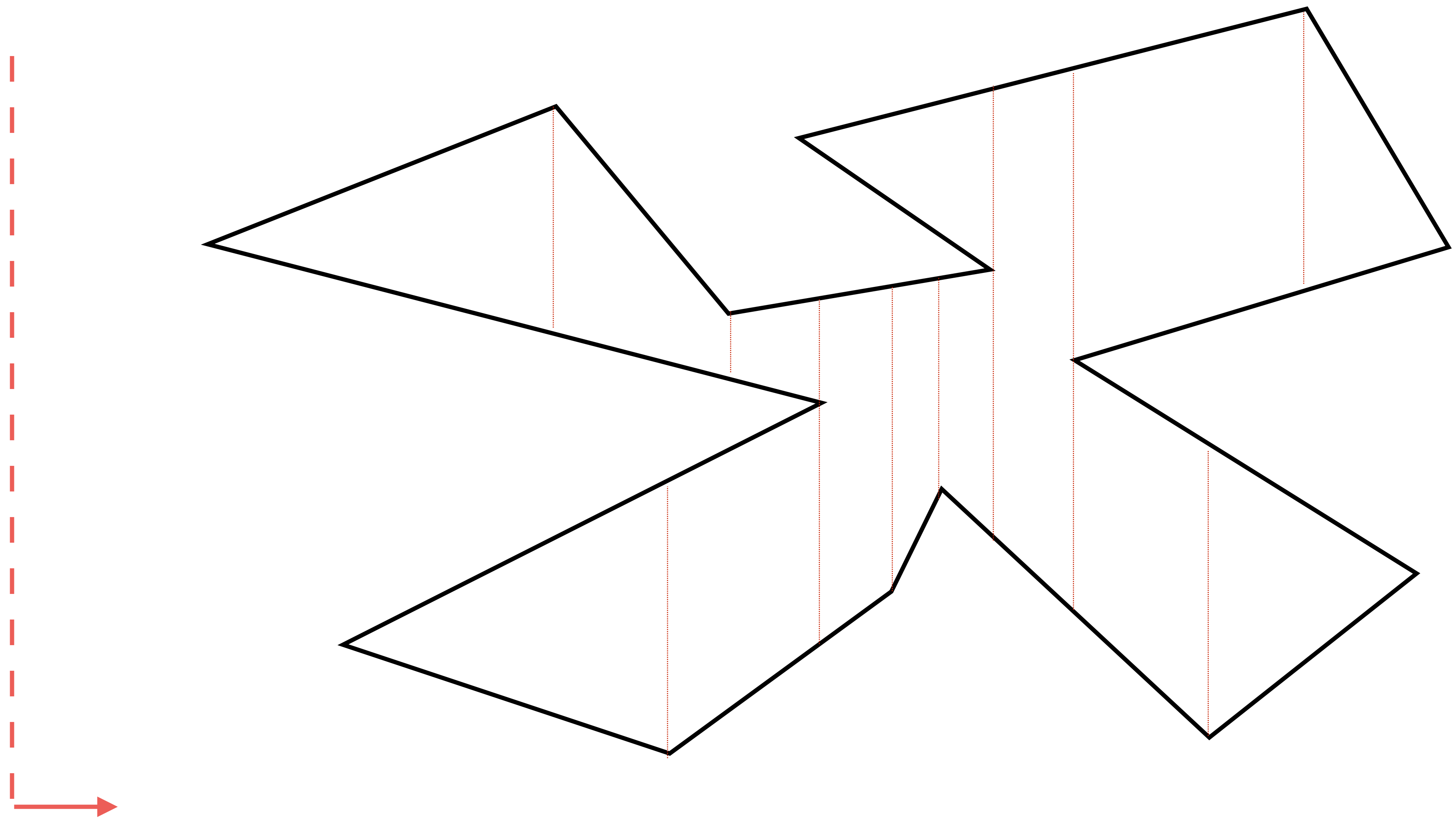
Given a polygon P, how do we compute a trapezoid partition?

Computing the trapezoid partition in O(n lg n)



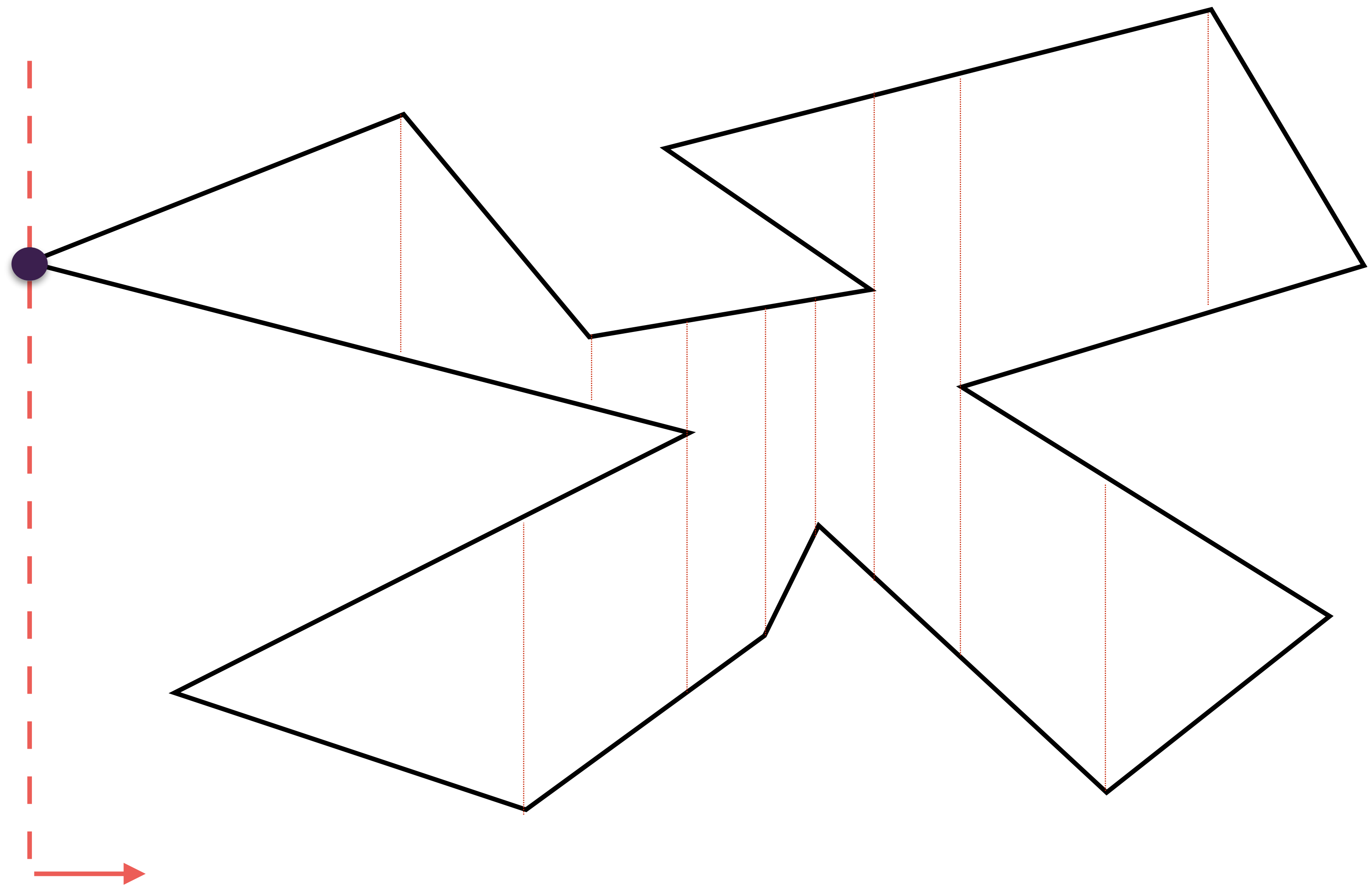
Computing the trapezoid partition in O(n lg n)

- Plane sweep



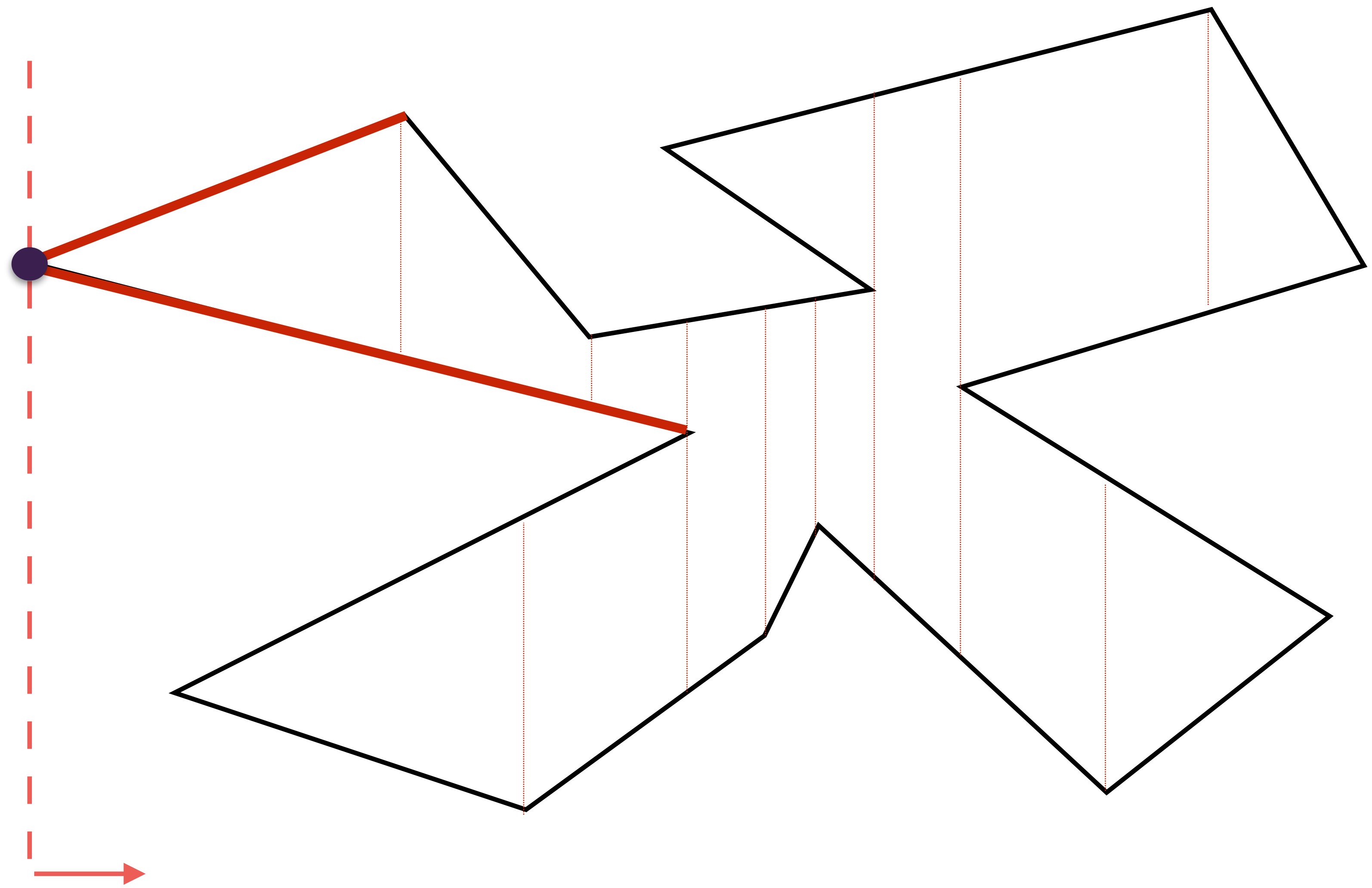
Computing the trapezoid partition in O(n lg n)

- Plane sweep



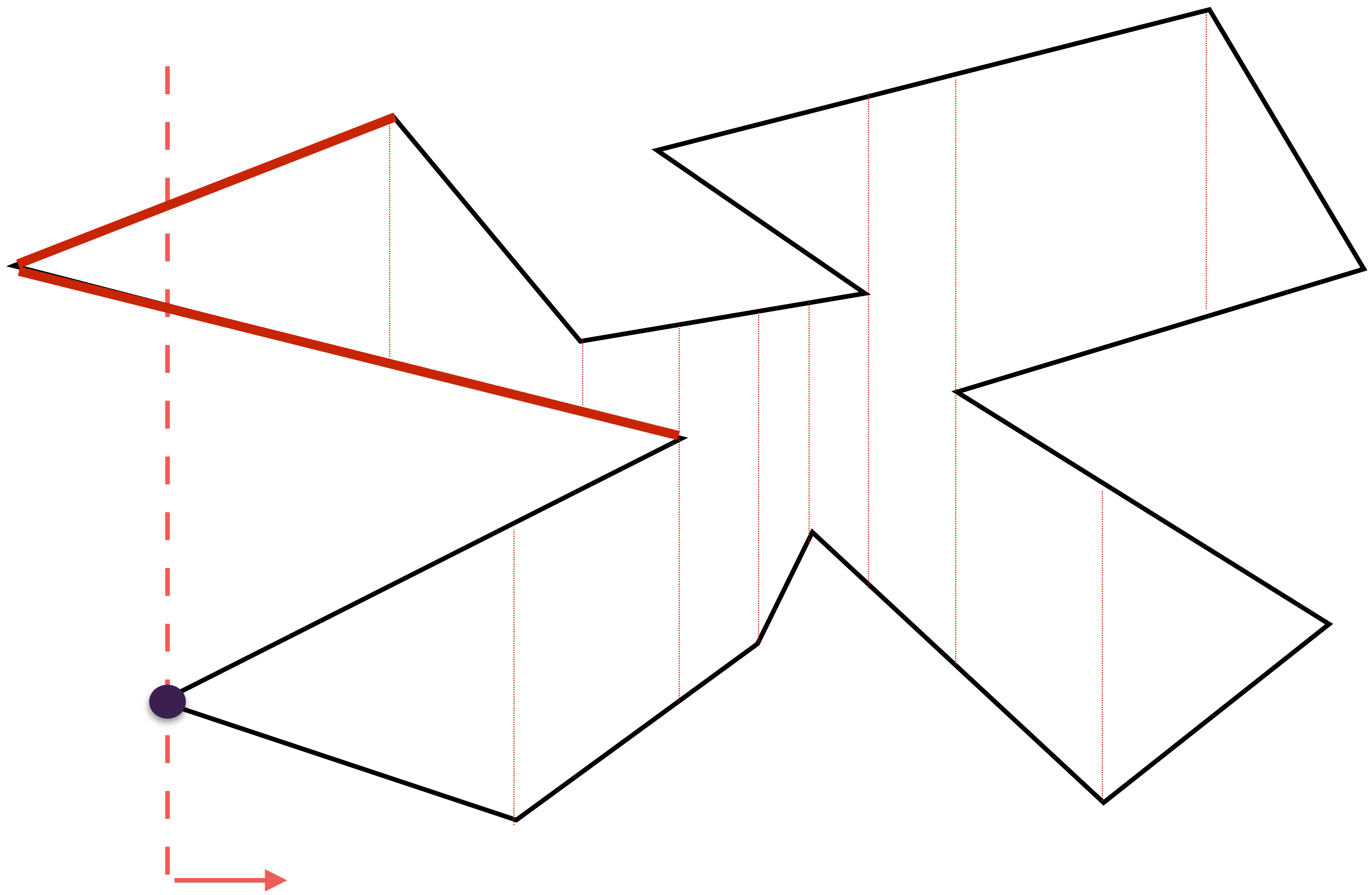
Computing the trapezoid partition in O(n lg n)

- Plane sweep



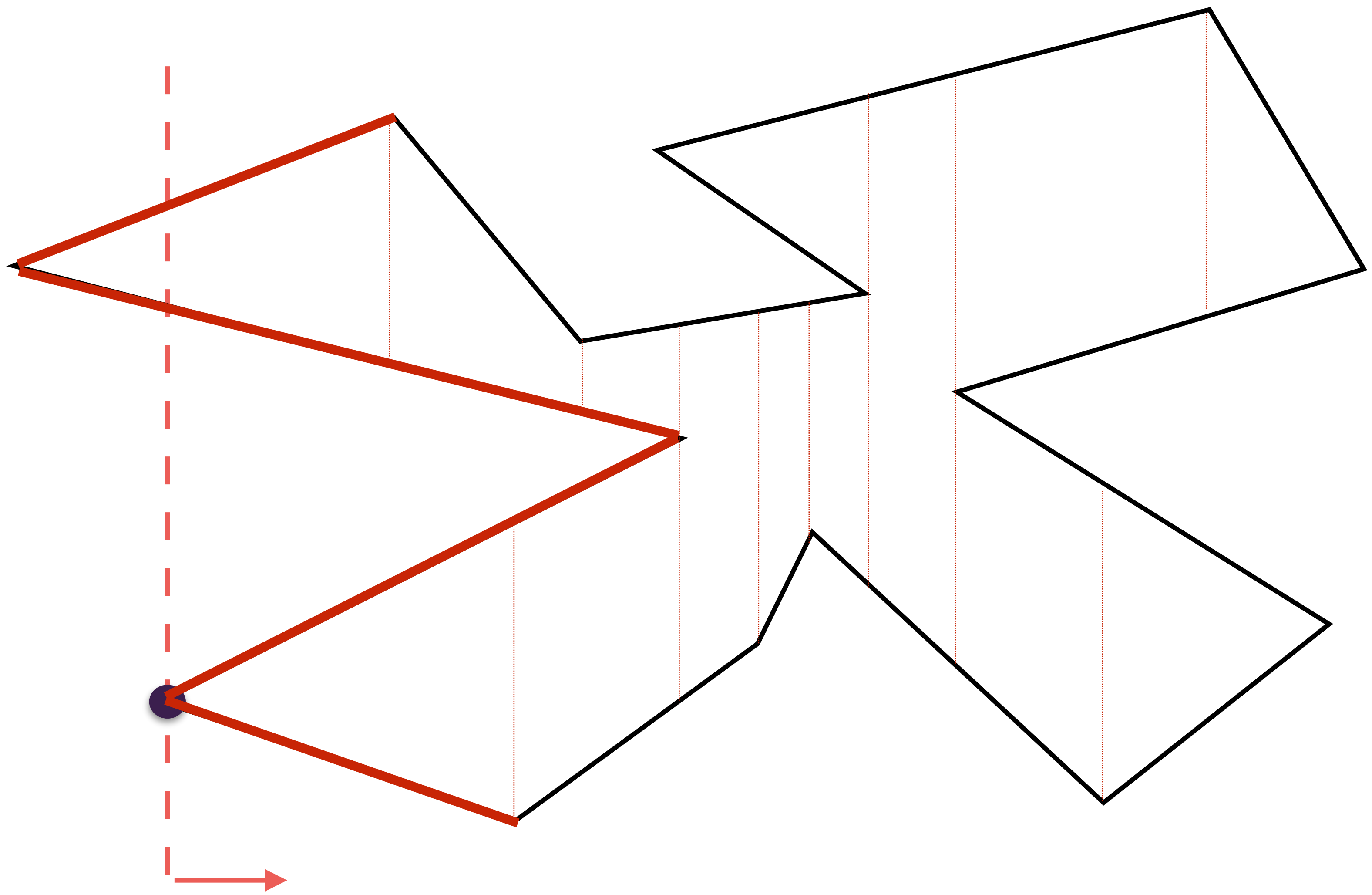
Computing the trapezoid partition in O(n lg n)

- Plane sweep



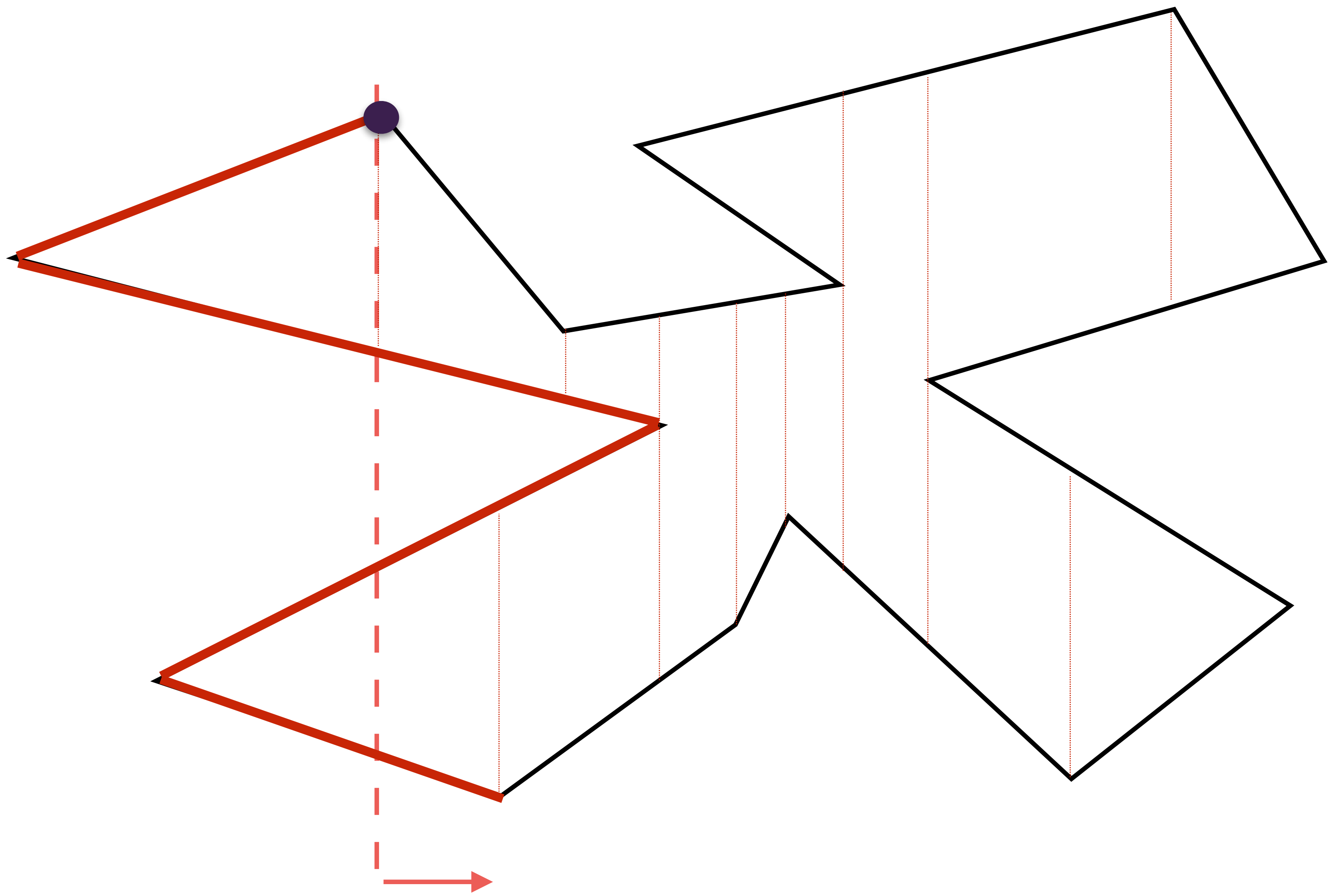
Computing the trapezoid partition in O(n lg n)

- Plane sweep



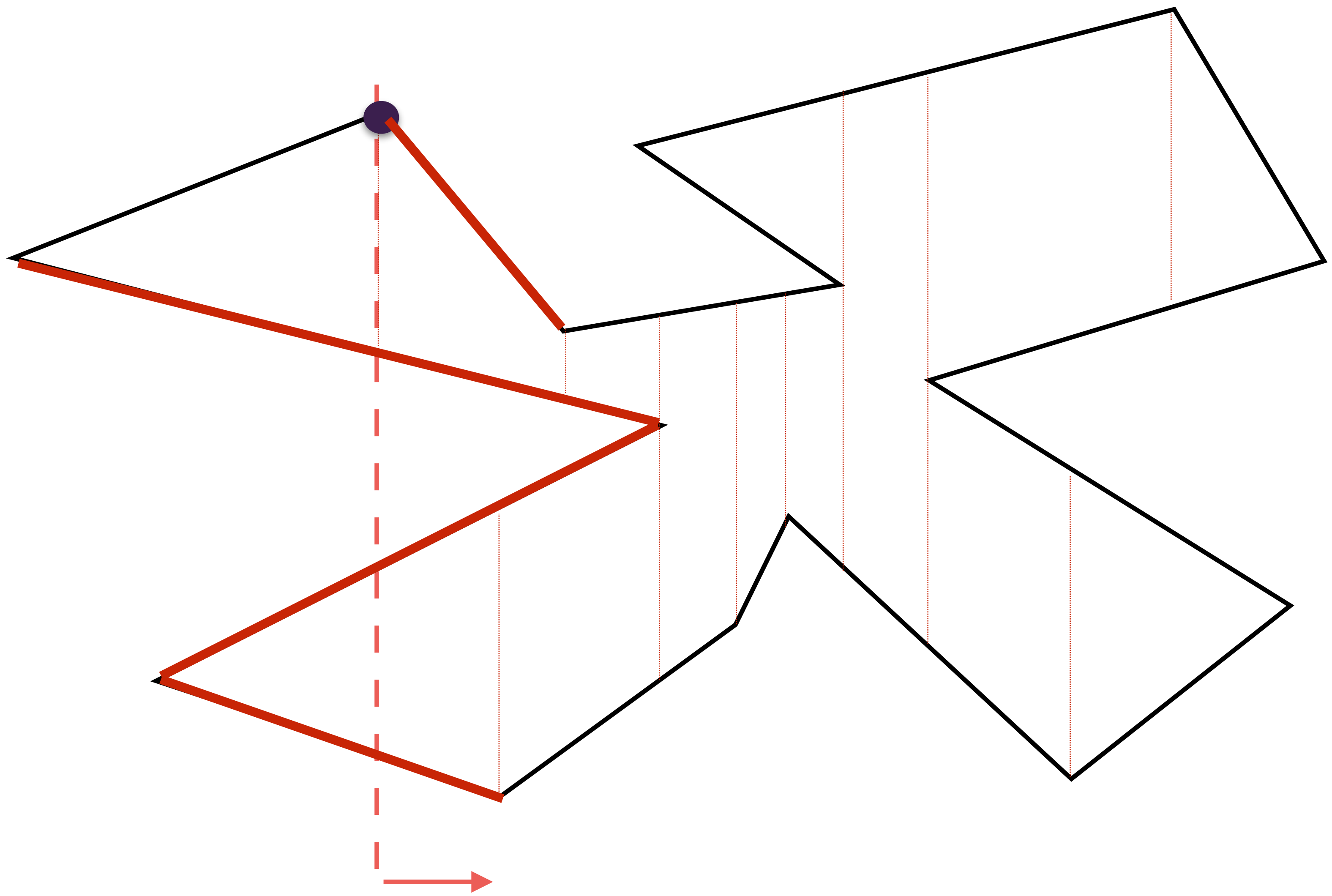
Computing the trapezoid partition in O(n lg n)

- Plane sweep



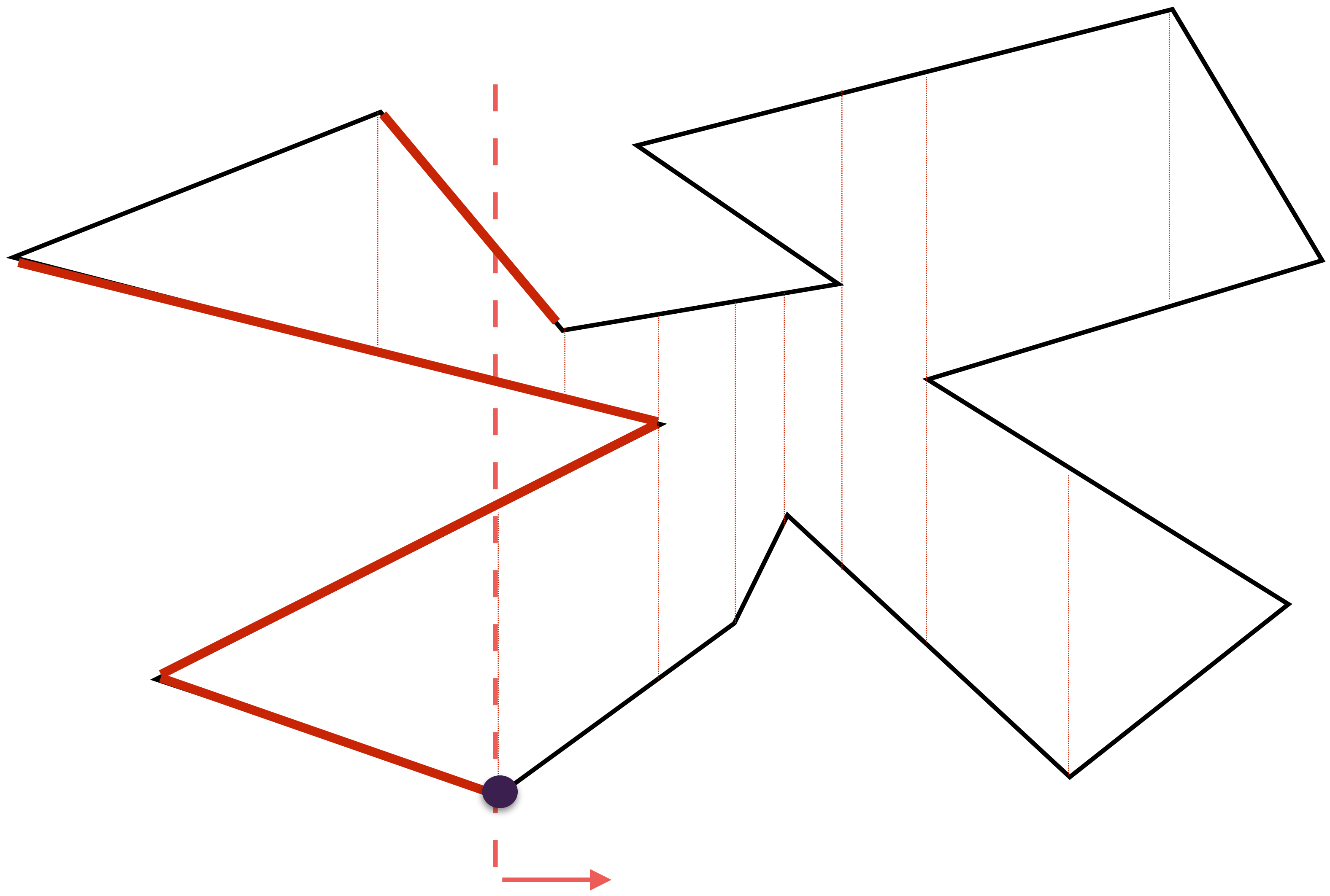
Computing the trapezoid partition in O(n lg n)

- Plane sweep



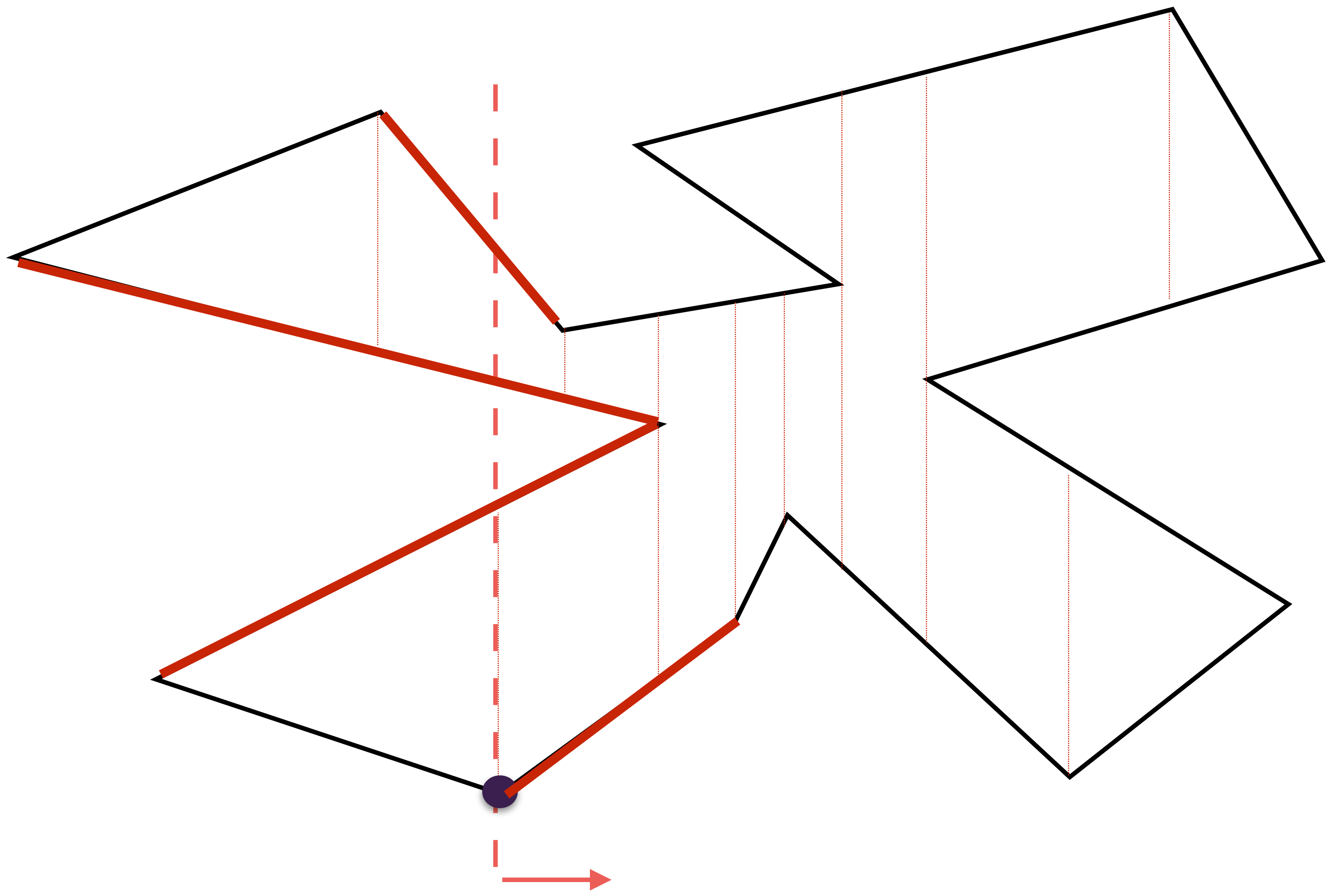
Computing the trapezoid partition in O(n lg n)

- Plane sweep



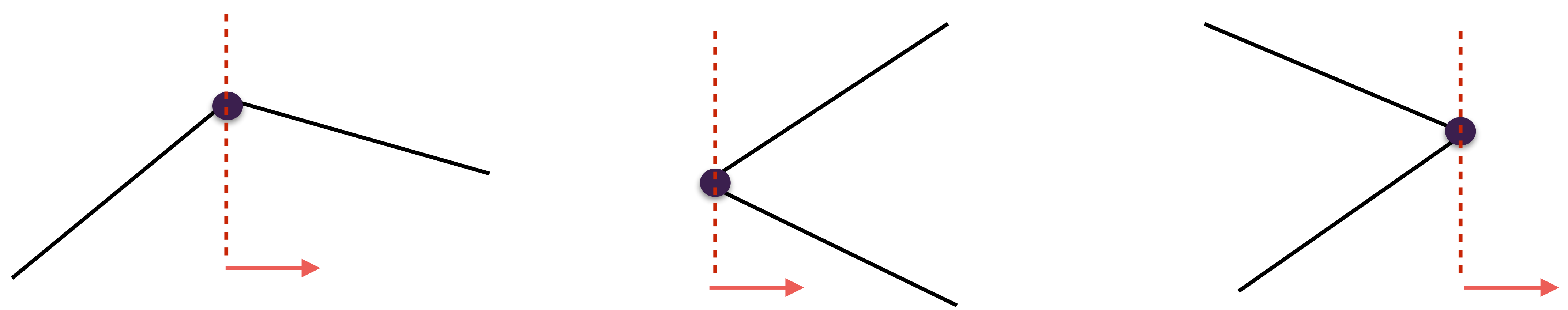
Computing the trapezoid partition in O(n lg n)

- Plane sweep



Computing the trapezoid partition in O(n lg n)

- Plane sweep
- Events: polygon vertices
- Status structure: edges that intersect current sweep line, in y-order
- Events:



How do you determine the trapezoids?

Computing the trapezoid partition in O(n lg n)

- Algorithm

History of Polygon Triangulation

- Early algorithms: $O(n^4)$, $O(n^3)$, $O(n^2)$
- First pseudo-linear algorithm: $O(n \lg n)$
- … Many papers with improved bounds
- Until finally Bernard Chazelle (Princeton) gave an $O(n)$ algorithm in 1991
  - https://www.cs.princeton.edu/~chazelle/pubs/polygon-triang.pdf
  - Ridiculously complicated!
  - $O(1)$ people actually understand it (and I'm not one of them)
- There is a randomized algorithm that runs in $O(n)$ expected