CSE520: Computational Geometry
Lecture 7
Polygons and Triangulations

Antoine Vigneron

Ulsan National Institute of Science and Technology

June 15, 2020

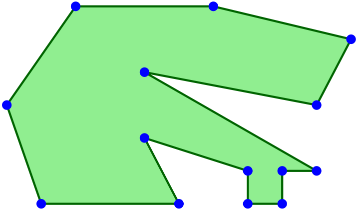
1. Introduction
2. Polygons and triangulations
3. Triangulating a monotone polygon
4. Partitioning a polygon into monotone pieces
5. Conclusion

Introduction

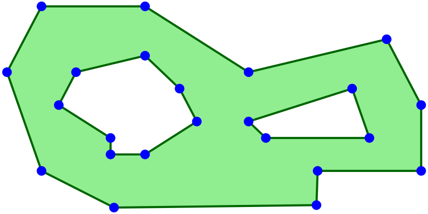
- This lecture is on polygons and triangulations.
- We will present an efficient algorithm for computing a triangulation of a polygon.
- Reference: Textbook Chapter 3.

Polygons

- A polygon is a face of a Planar Straight Line Graph.
- A *simple polygon* is the region enclosed by a simple (non-intersecting) polyline.

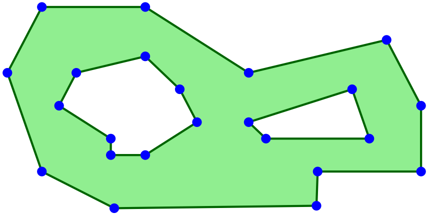


a simple polygon

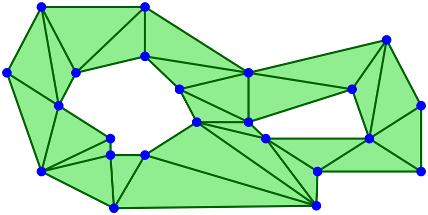


a polygon with 2 holes

Triangulations



a polygon P



a triangulation of P

Definition (Polygon triangulation)

A *Triangulation* of a polygon P is a partition of P into triangles whose vertices are the vertices of P.

- A polygon may have several triangulations.
- A triangulation is a planar straight line graph.

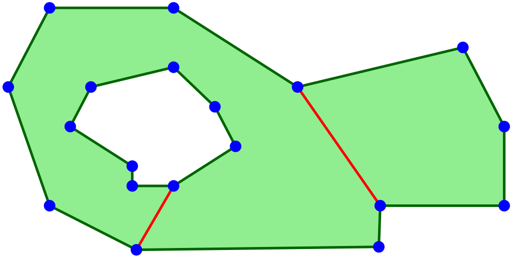
Applications

- Meshing $\Rightarrow$ scientific computing.
- Visibility problems.
- Graphics.
- Preprocessing step of many geometric algorithms.

Existence of a Triangulation

Definition

A *diagonal* of a polygon P is a line segment $\overline{pq}$ such that p and q are vertices of P and the interior of $\overline{pq}$ is in the interior of P.



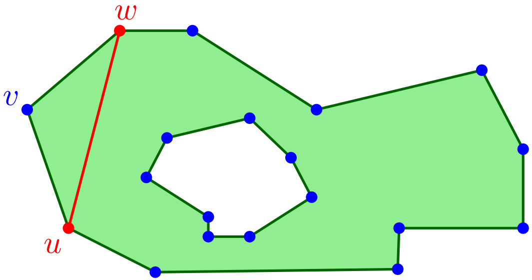
two diagonals

Lemma

Any polygon P with more than three vertices admits a diagonal.

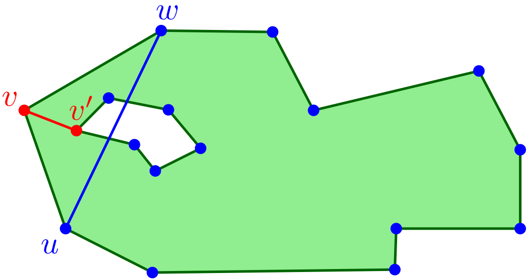
Proof (Lemma)

- Let v be the leftmost vertex of P.
- Let u and w be its neighbors.
- If $\overline{uw}$ is a diagonal we are done.



Proof (Lemma)

- If $\overline{uw}$ is not a diagonal, let v' be the vertex in triangle (u, v, w) that is farthest from $\overline{uw}$.



- Then $\overline{vv'}$ is a diagonal: if an edge was crossing it, one of its endpoints would be farther from $\overline{uw}$ and inside (u, v, w).

Existence of a Triangulation

Theorem

Any polygon P admits a triangulation.

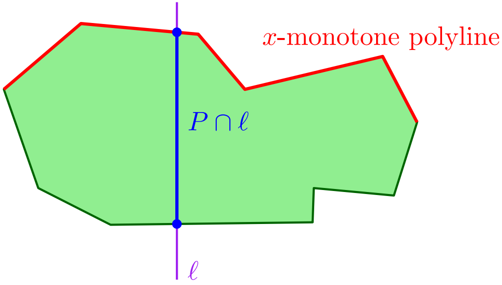
Proof:

- Subdivide P by adding diagonals.
- As long as a face of this subdivision has more than 3 vertices, we can add a new diagonal in this face.
- In the end, all faces of the subdivision have only 3 vertices.
- Hence it is a triangulation.

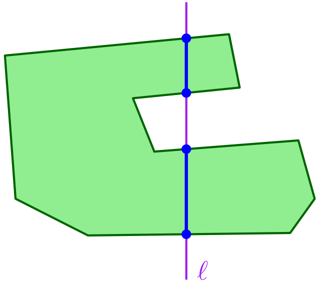
More Results

- Any triangulation of a simple polygon with n vertices has $n-2$ faces and $n-3$ diagonals.
- We can find a diagonal in $O(n)$ time.
- We can find a triangulation in $O(n^2)$ time.
- Is there a faster algorithm?
  - Yes, there is an optimal $O(n \log n)$ time and $O(n)$ space algorithm.
  - This is what we will see next.
- There is an $O(n)$ time algorithm for simple polygons.
  - It is a difficult result, not covered in this course.

Monotone Polygon



an x-monotone polygon



not x-monotone

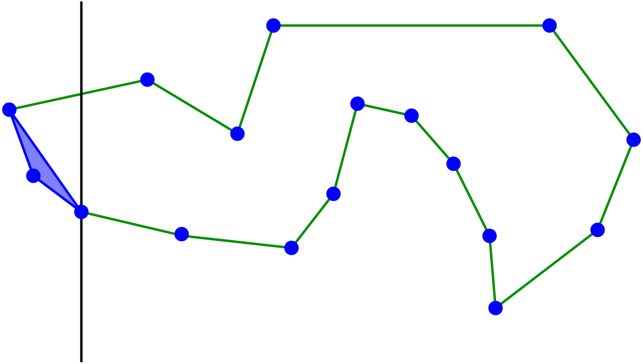
Definition (x-monotone polygon)

An *x-monotone polygon* is a polygon such that for any vertical line ℓ, the intersection $P \cap \ell$ is a line segment. Equivalently, it is a simple polygon whose boundary consists of two x-monotone polylines.

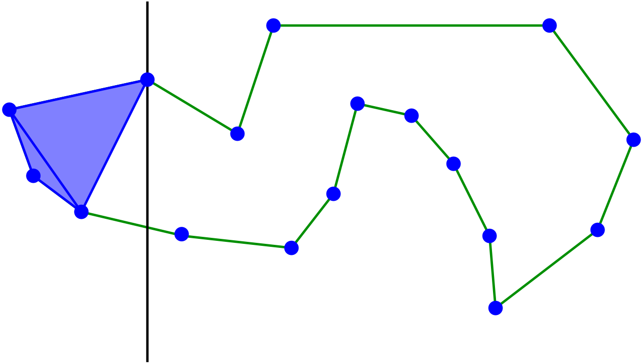
Algorithm for Triangulating a Monotone Polygon

- Plane sweep approach.
- The sweep line ℓ moves from left to right and stops at each vertex of P.
  - We can sort these vertices in $O(n \log n)$ time.
  - We can also do it in $O(n)$ time. How?

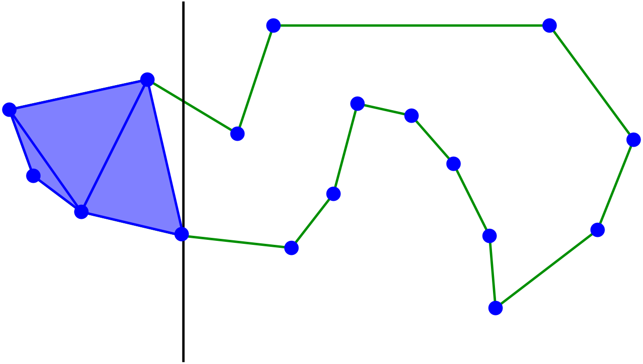
Example



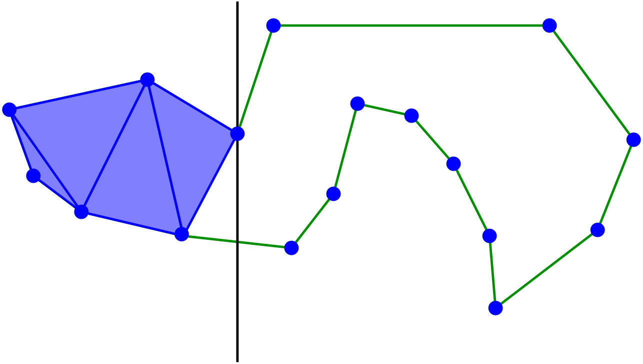
Example



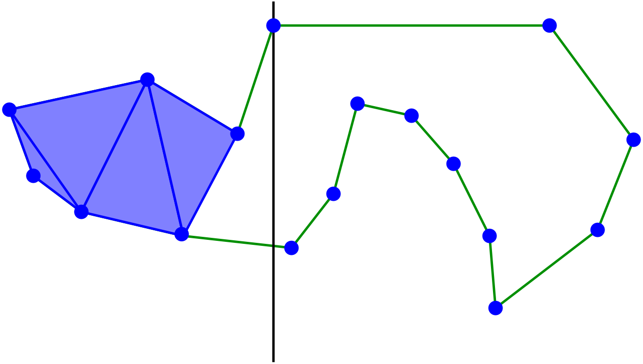
Example



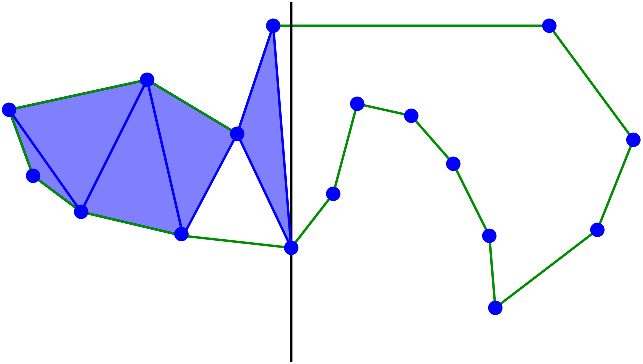
Example



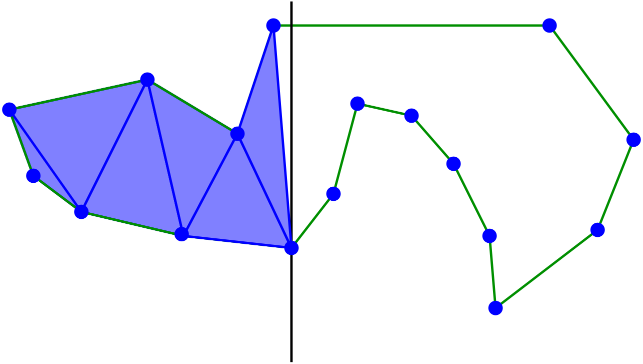
Example



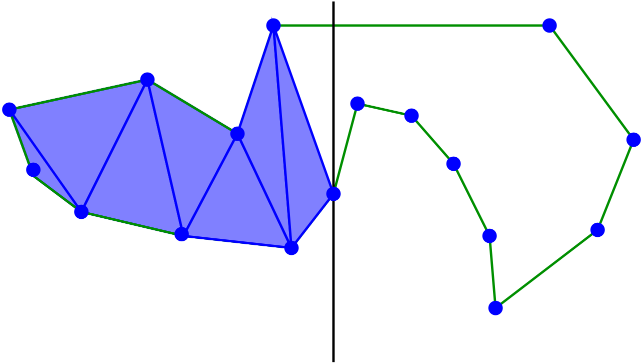
Example



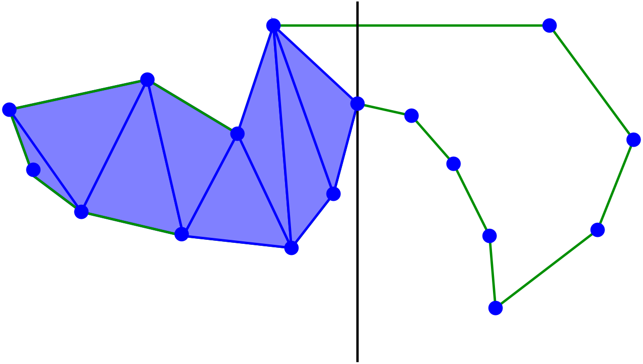
Example



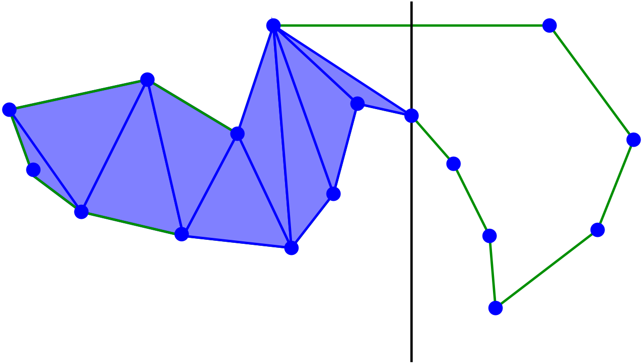
Example



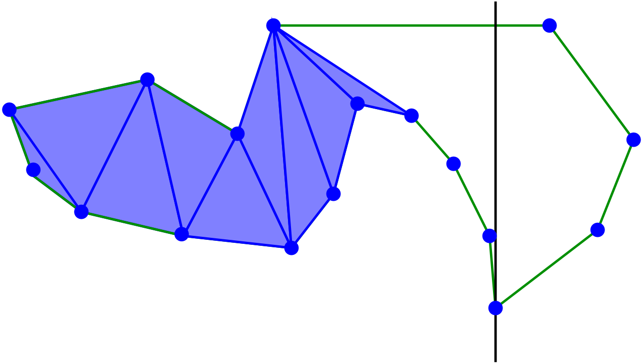
Example



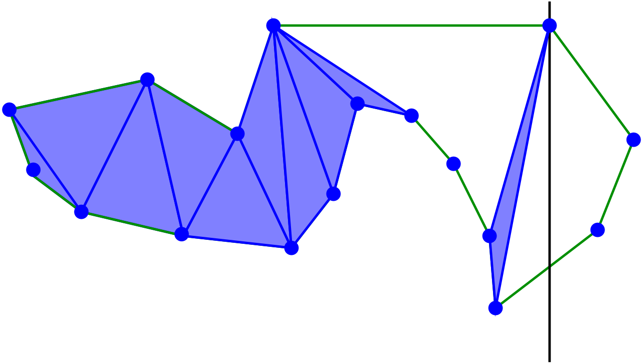
Example



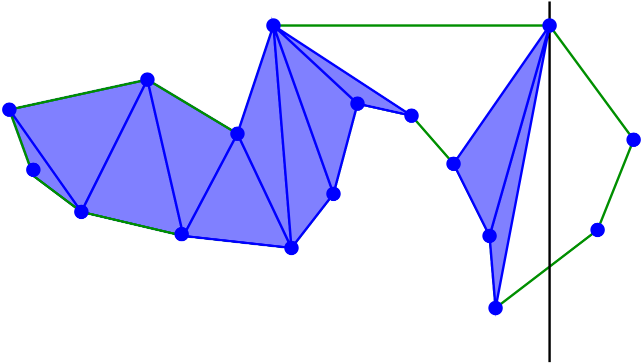
Example



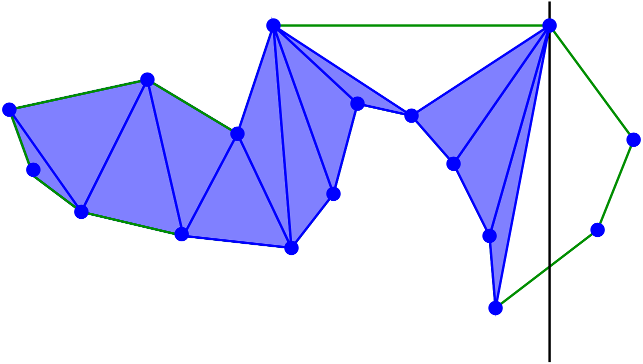
Example



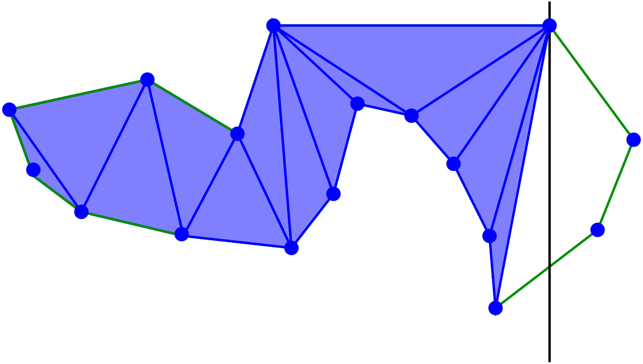
Example



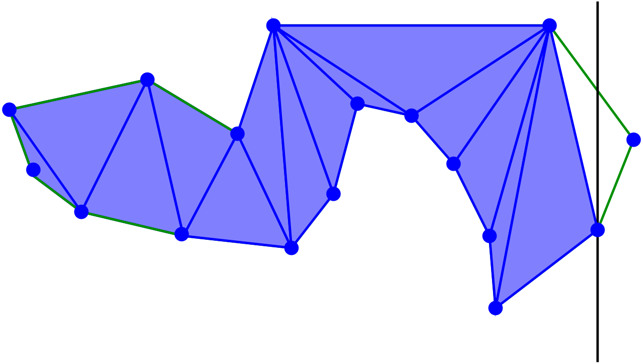
Example



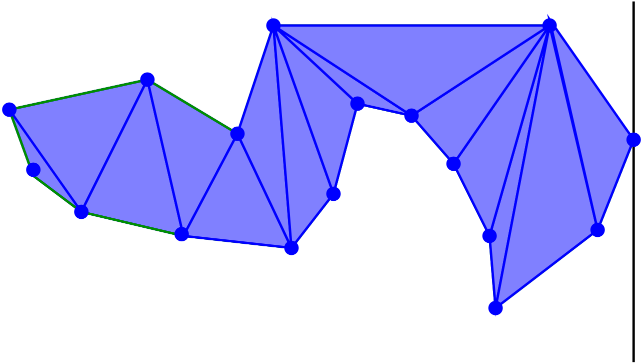
Example



Example

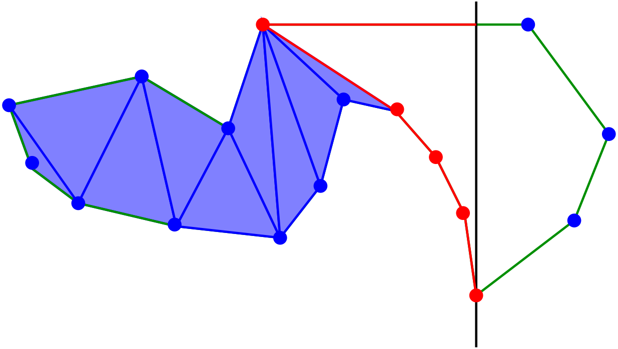


Example



Proof of Correctness

- Invariant:
  - The non-triangulated region to the left of the sweep line is delimited by an edge on one side and a reflex chain on the other side.

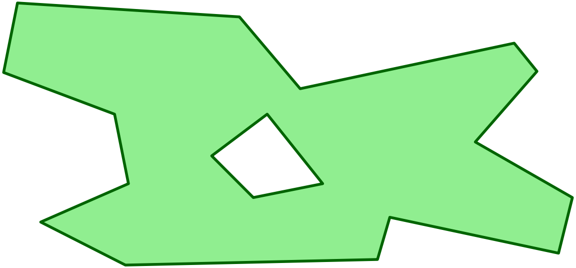


  - We can maintain this invariant. (See D. Mount notes).

Analysis

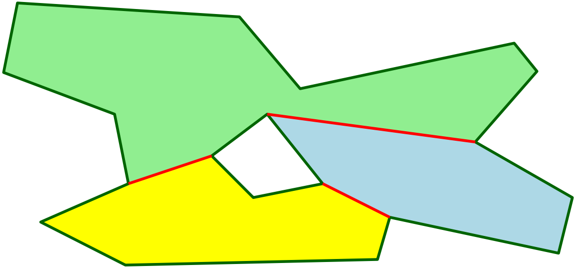
- Vertices can be sorted along the x-axis in $O(n)$ time.
- We maintain the reflex chain in a stack.
  - Push and pop in $O(1)$ time.
- Each vertex is pushed and popped at most once.
- This algorithm runs in optimal $\Theta(n)$ time.
- We can use Doubly Connected Edge Lists.

Partitioning into monotone pieces



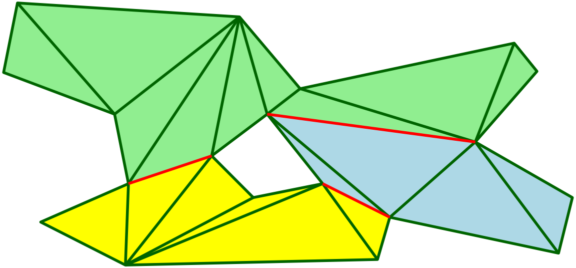
- We want to partition a polygon P into a collection of x-monotone polygons with same vertex set.
- After this, we can apply separately to each piece our algorithm for triangulating a monotone polygon.

Partitioning into monotone pieces



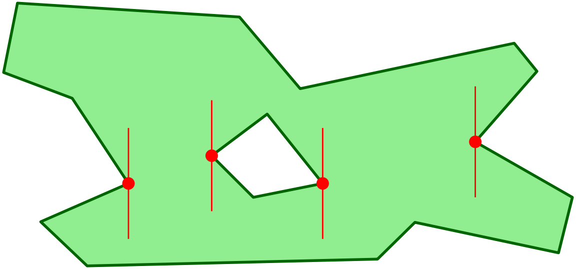
- We want to partition a polygon P into a collection of x-monotone polygons with same vertex set.
- After this, we can apply separately to each piece our algorithm for triangulating a monotone polygon.

Partitioning into monotone pieces



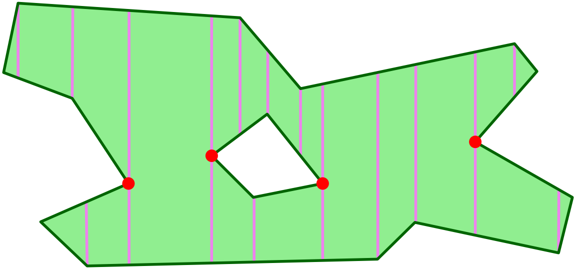
- We want to partition a polygon P into a collection of x-monotone polygons with same vertex set.
- After this, we can apply separately to each piece our algorithm for triangulating a monotone polygon.

Partitioning into monotone pieces



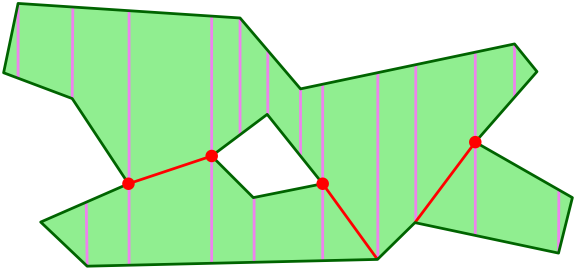
- A *turning point* is a vertex such that the vertical tangent at this vertex is locally inside P.

Partitioning into monotone pieces



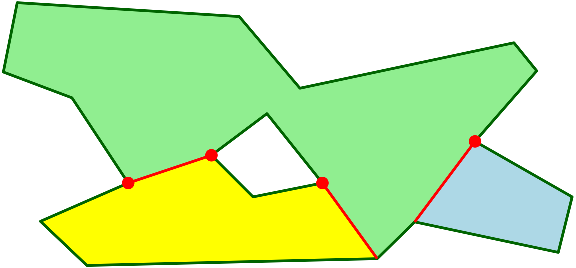
- We compute the trapezoidal map of P.
- What should we do now?

Partitioning into monotone pieces



- Each turning point lies on the interior of a vertical edge of a trapezoid.
- We connect this turning point to the other vertex of this trapezoid.

Partitioning into monotone pieces



- The resulting pieces have no turning point.
- Hence they are x-monotone.

Conclusion

Theorem

A polygon P with n vertices can be triangulated in $O(n \log n)$ time.

- We first compute the trapezoidal map of P. It takes $O(n \log n)$ time.
- Then we partition P into monotone pieces. It takes time $O(n)$ because there are $O(n)$ trapezoids and we can handle each trapezoid in $O(1)$ time.
- Finally we triangulate each monotone piece. In total, it takes $O(n)$ time because there are n vertices in total, so there are $O(n)$ edges in total (as the graph is planar). Therefore, the sum of the numbers of edges of all the pieces is $O(n)$, and hence the sum of the numbers of vertices is $O(n)$. As each piece is triangulated in linear time, the overall running time is $O(n)$.