TRIANGULATING POLYGONS

Vera Sacristán

Computational Geometry
Facultat d'Informàtica de Barcelona
Universitat Politcnica de Catalunya

TRIANGULATING POLYGONS

TRIANGULATING POLYGONS

A **polygon triangulation** is the decomposition of a polygon into triangles. This is done by inserting internal diagonals.

An **internal diagonal** is any segment...

- connecting two vertices of the polygon and
- completely enclosed in the polygon.

- intersección de los Δ's es vacía
- unión de los Δ's es el polígono
- vértices de Δ's = vértices del polígono.

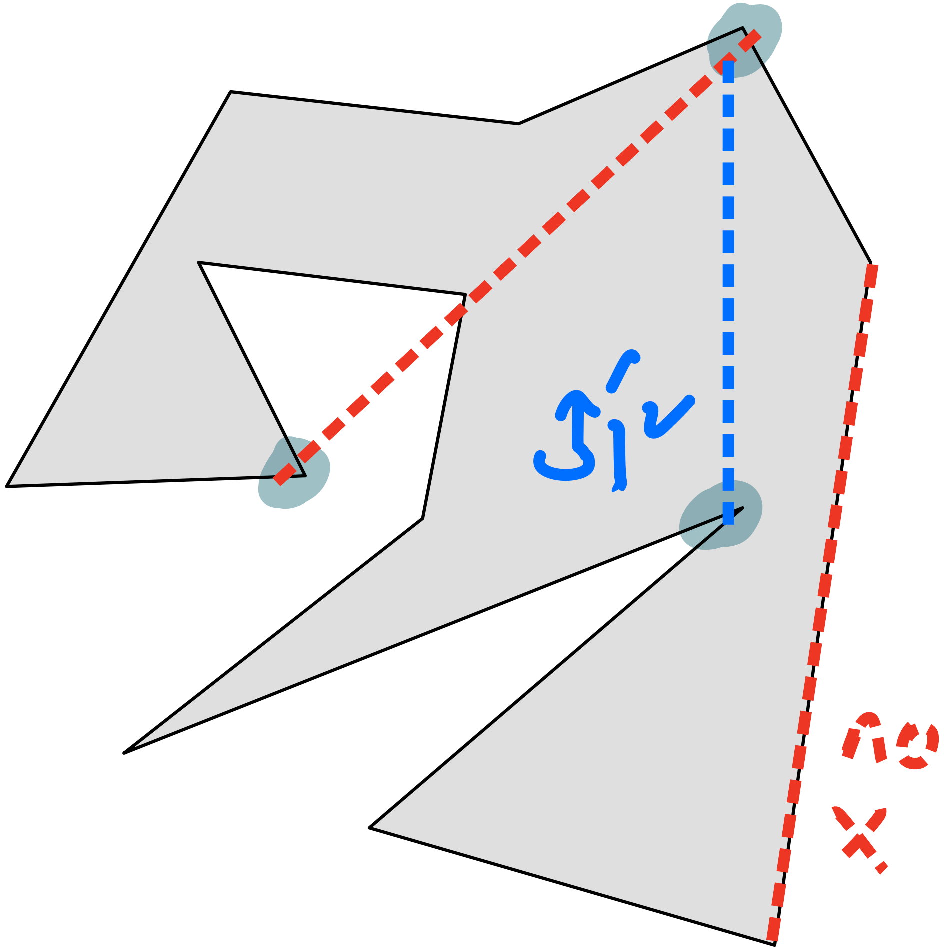
En el interior del polígono es una gráfica plana maximal, sus aristas son segmentos de recta

TRIANGULATING POLYGONS

A **polygon triangulation** is the decomposition of a polygon into triangles. This is done by inserting internal diagonals.

An **internal diagonal** is any segment...

- connecting two vertices of the polygon and
- completely enclosed in the polygon.

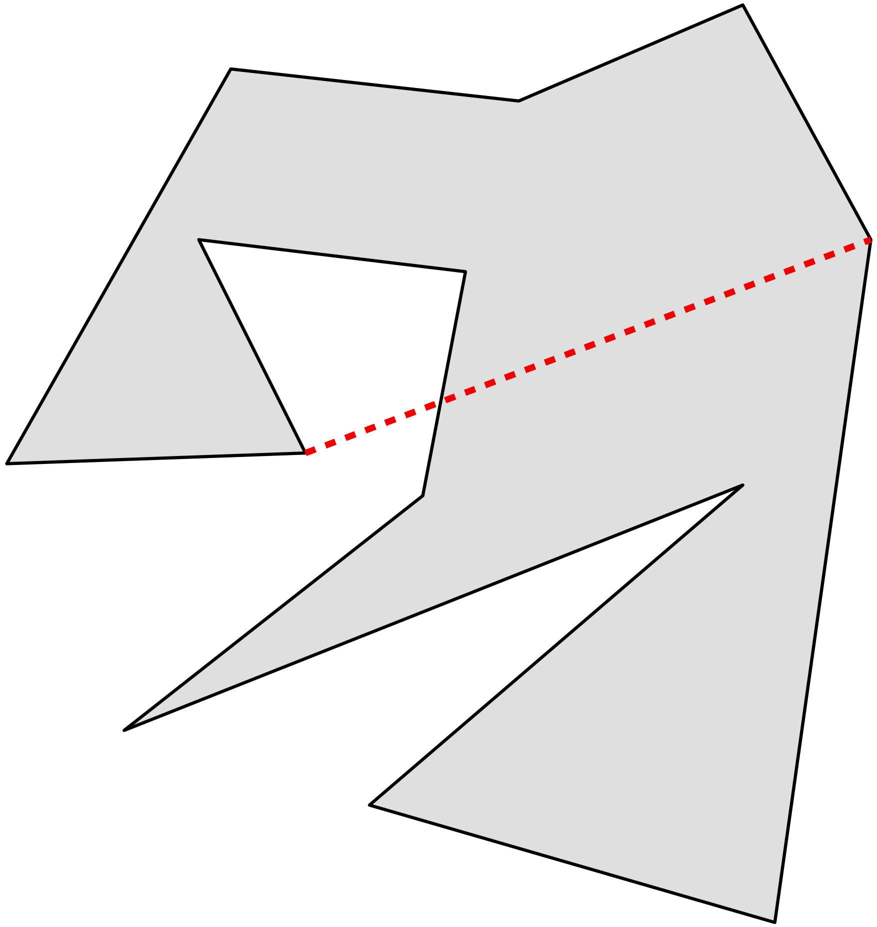


TRIANGULATING POLYGONS

A **polygon triangulation** is the decomposition of a polygon into triangles. This is done by inserting internal diagonals.

An **internal diagonal** is any segment...

- connecting two vertices of the polygon and
- completely enclosed in the polygon.

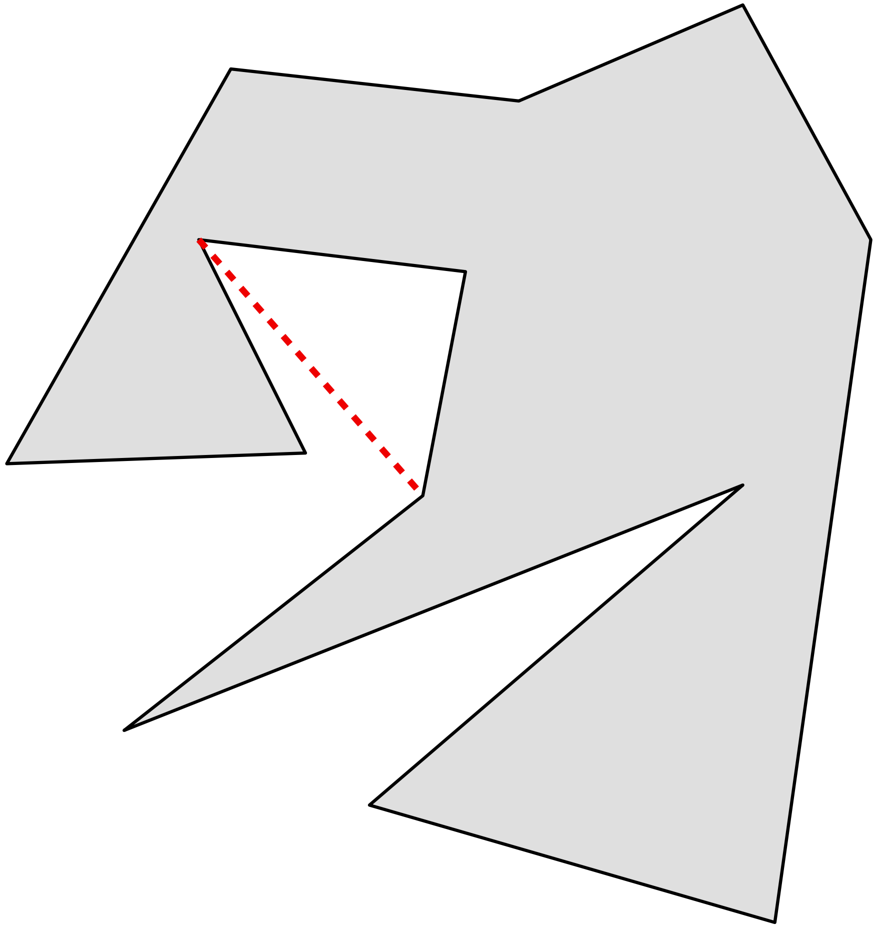


TRIANGULATING POLYGONS

A **polygon triangulation** is the decomposition of a polygon into triangles. This is done by inserting internal diagonals.

An **internal diagonal** is any segment...

- connecting two vertices of the polygon and
- completely enclosed in the polygon.



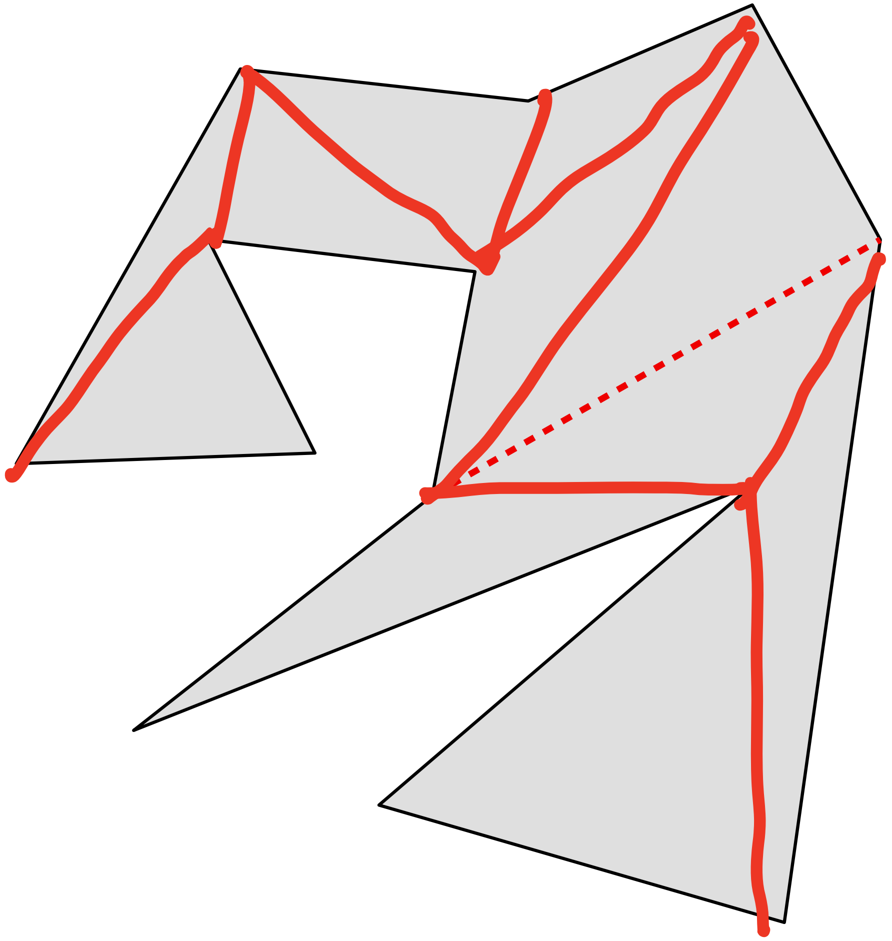
TRIANGULATING POLYGONS

A **polygon triangulation** is the decomposition of a polygon into triangles. This is done by inserting internal diagonals.

An **internal diagonal** is any segment...

- connecting two vertices of the polygon and
- completely enclosed in the polygon.

o No hay cruces.

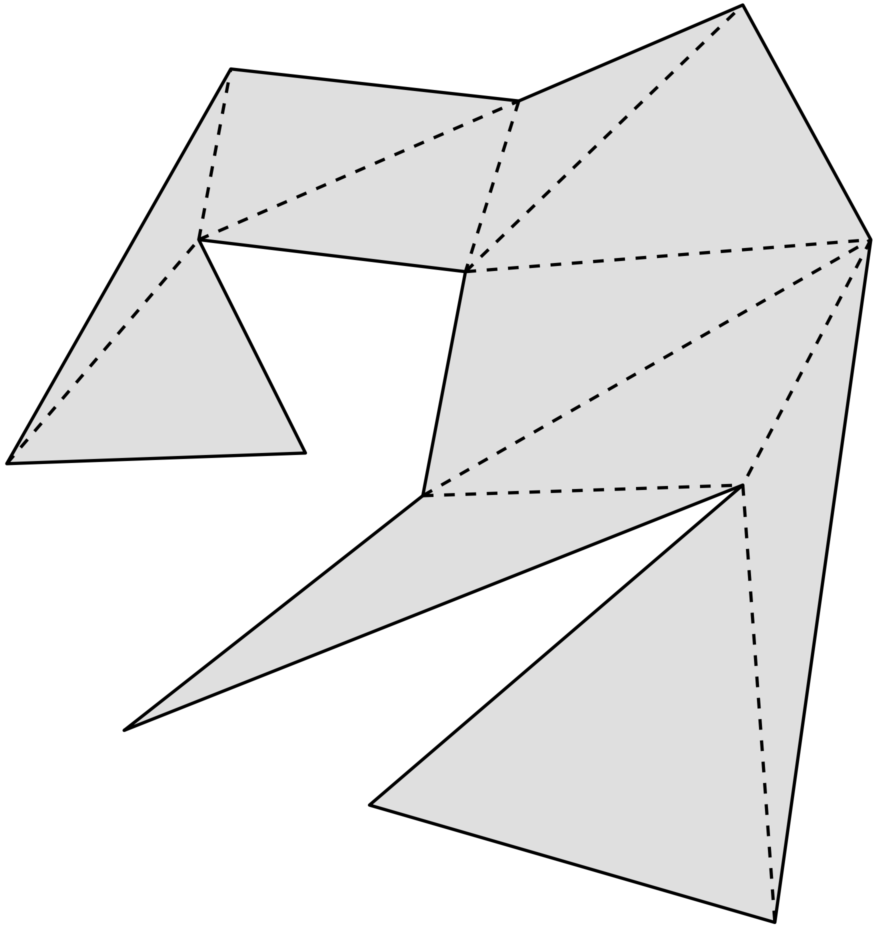


TRIANGULATING POLYGONS

A **polygon triangulation** is the decomposition of a polygon into triangles. This is done by inserting internal diagonals.

An **internal diagonal** is any segment...

- connecting two vertices of the polygon and
- completely enclosed in the polygon.

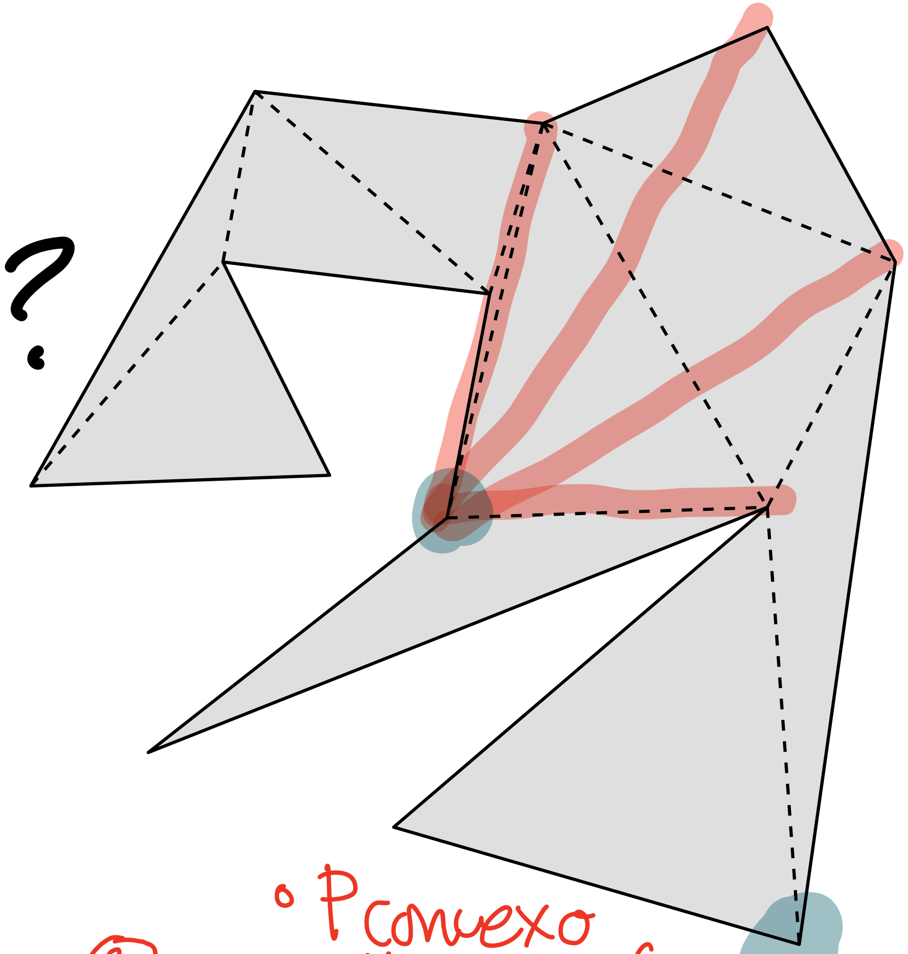


TRIANGULATING POLYGONS

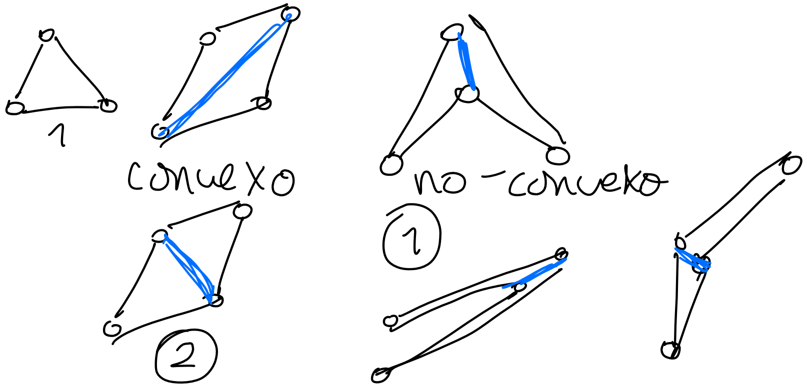
A **polygon triangulation** is the decomposition of a polygon into triangles. This is done by inserting internal diagonals.

An **internal diagonal** is any segment...

- connecting two vertices of the polygon and
- completely enclosed in the polygon.



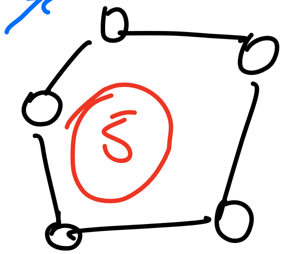
¿Cuántas Δ's tiene P?

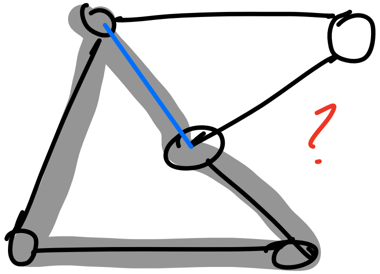


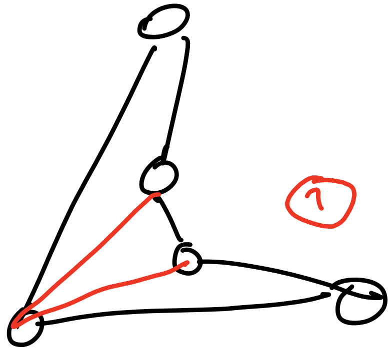
$T(n) = 1 + T(4)$

$n=5$

$\#$ diag.







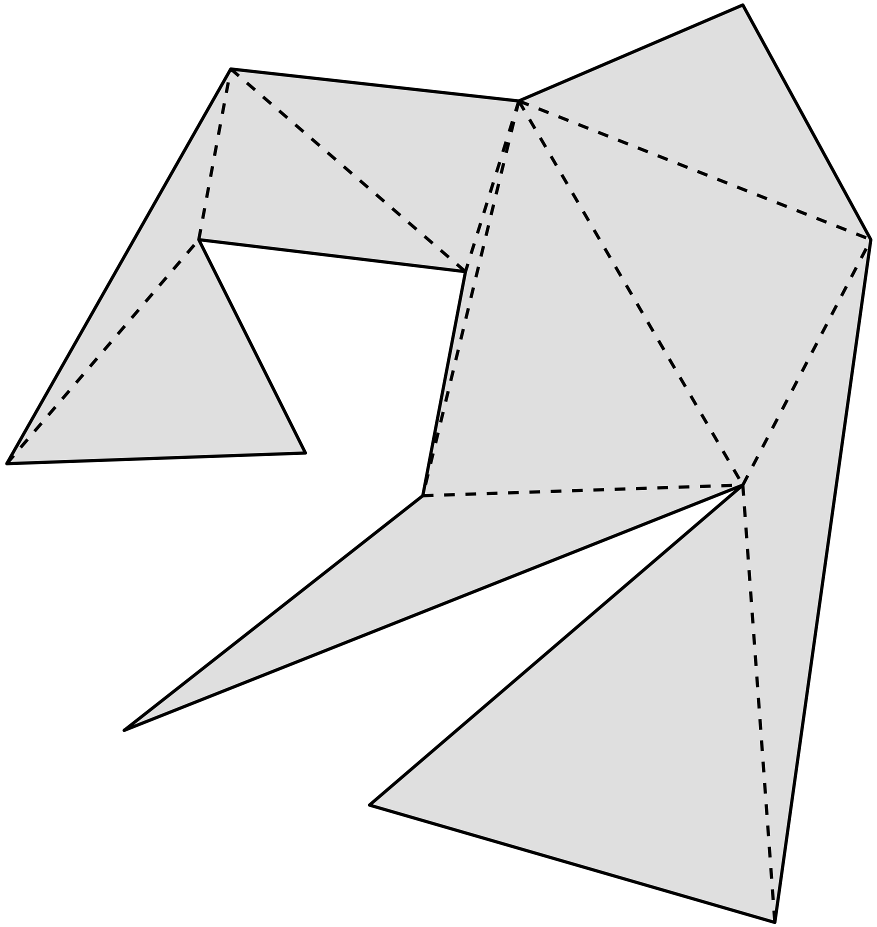
- P convexo
  $\#$Catalán → exp.

TRIANGULATING POLYGONS

A **polygon triangulation** is the decomposition of a polygon into triangles. This is done by inserting internal diagonals.

An **internal diagonal** is any segment...

- connecting two vertices of the polygon and
- completely enclosed in the polygon.



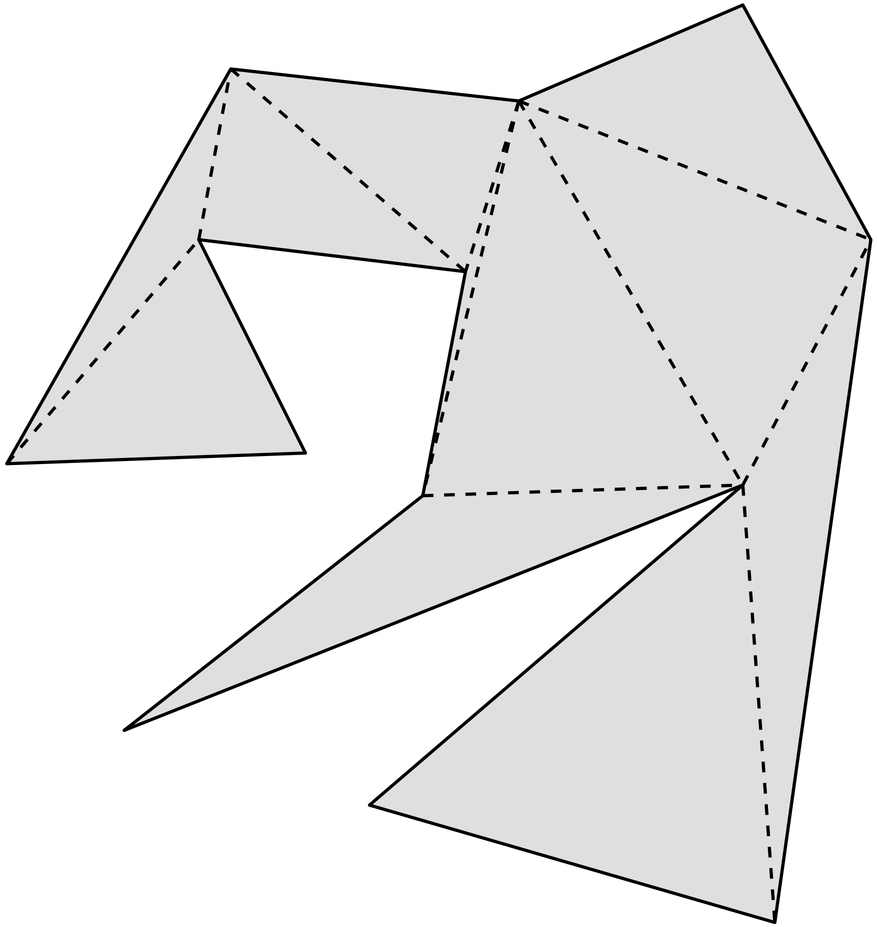
An application example:

TRIANGULATING POLYGONS

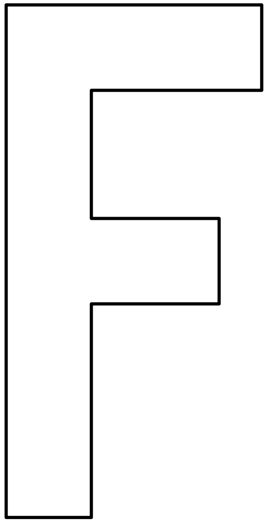
A **polygon triangulation** is the decomposition of a polygon into triangles. This is done by inserting internal diagonals.

An **internal diagonal** is any segment...

- connecting two vertices of the polygon and
- completely enclosed in the polygon.



An application example:

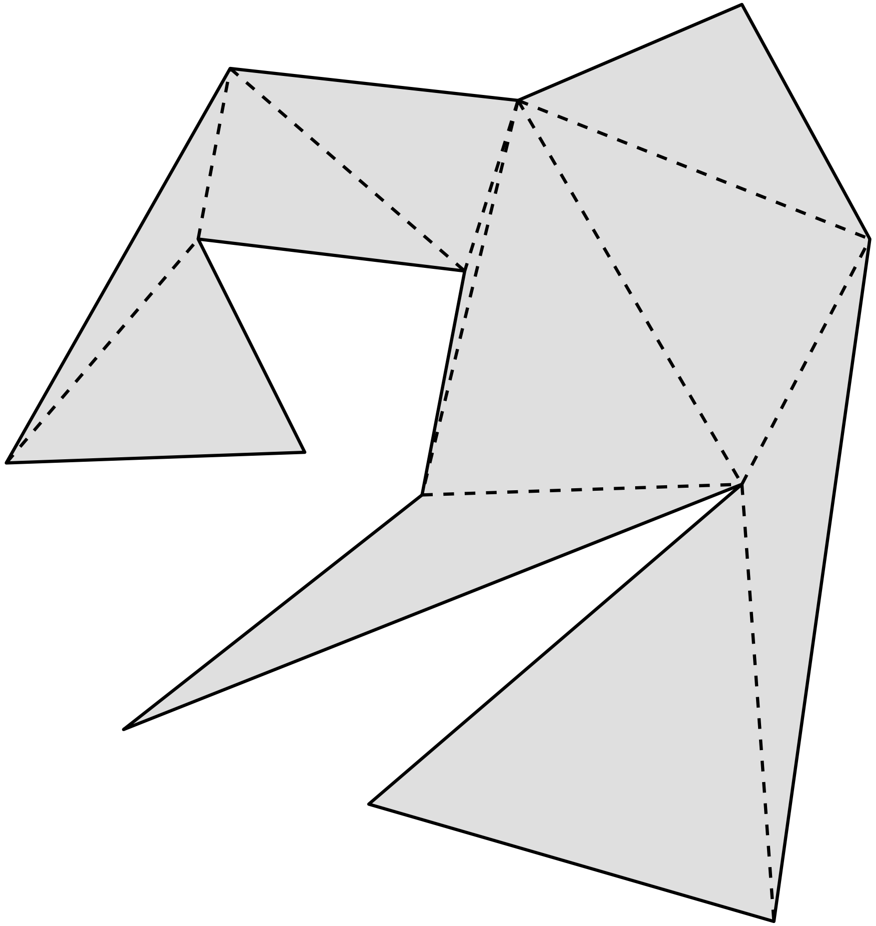


TRIANGULATING POLYGONS

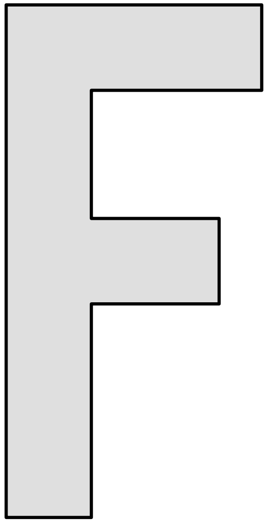
A **polygon triangulation** is the decomposition of a polygon into triangles. This is done by inserting internal diagonals.

An **internal diagonal** is any segment...

- connecting two vertices of the polygon and
- completely enclosed in the polygon.



An application example:

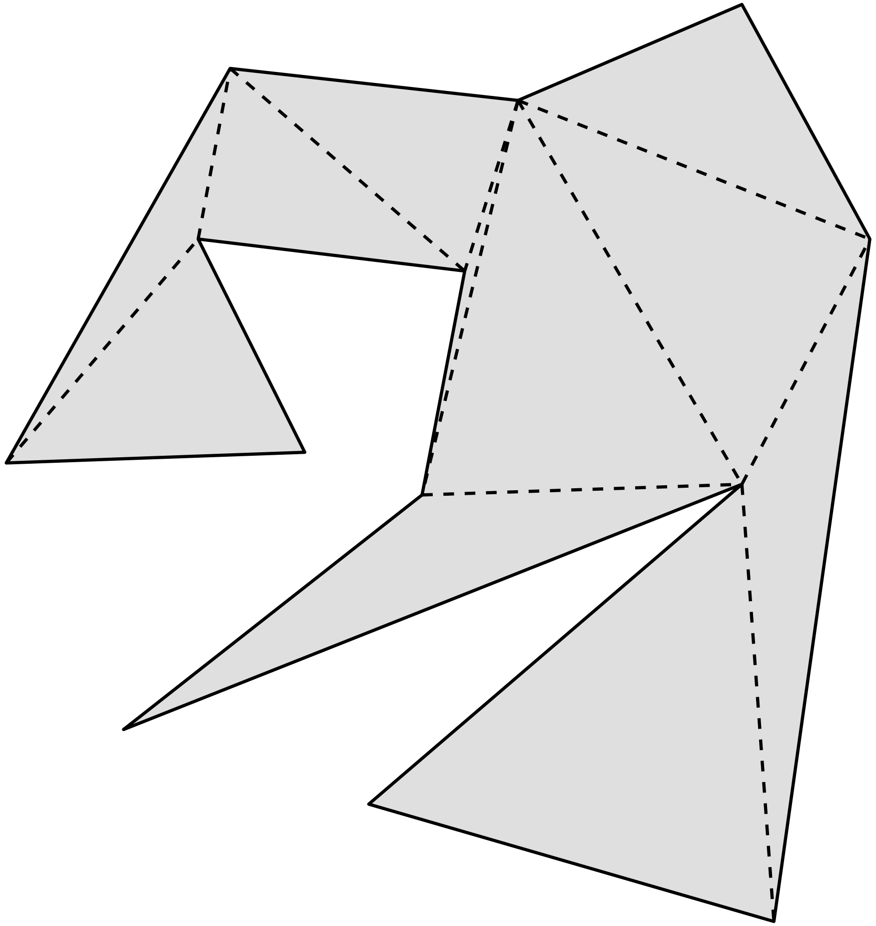


TRIANGULATING POLYGONS

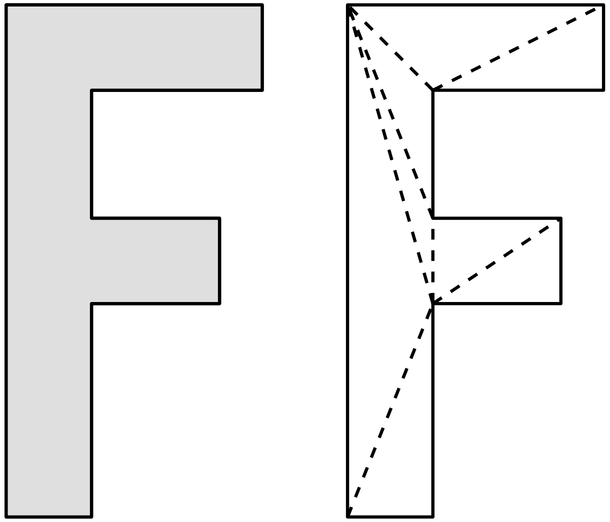
A **polygon triangulation** is the decomposition of a polygon into triangles. This is done by inserting internal diagonals.

An **internal diagonal** is any segment...

- connecting two vertices of the polygon and
- completely enclosed in the polygon.



An application example:

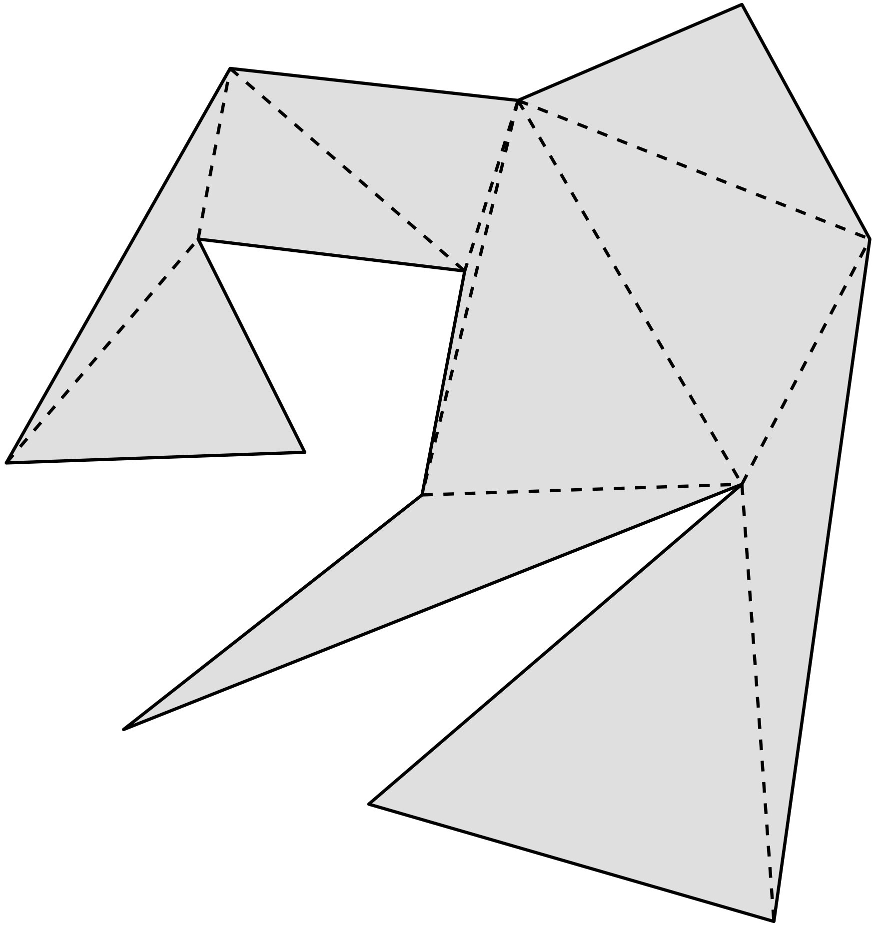


TRIANGULATING POLYGONS

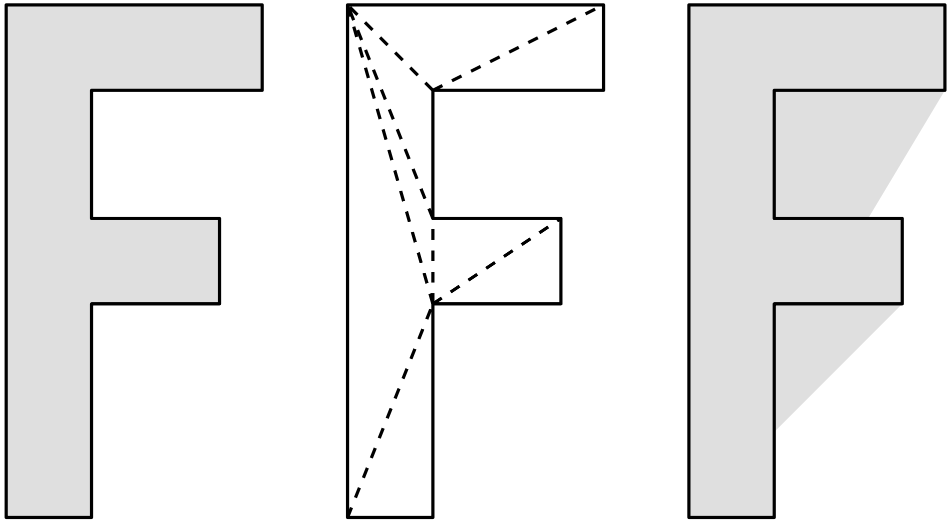
A **polygon triangulation** is the decomposition of a polygon into triangles. This is done by inserting internal diagonals.

An **internal diagonal** is any segment...

- connecting two vertices of the polygon and
- completely enclosed in the polygon.



An application example:

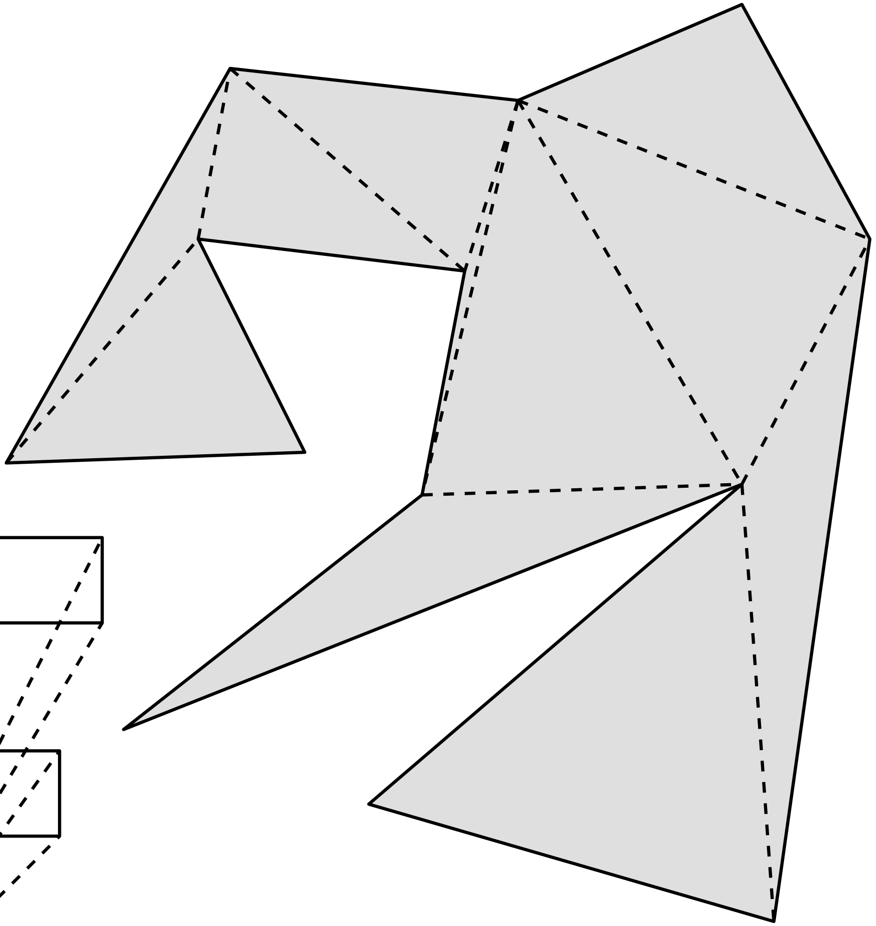


TRIANGULATING POLYGONS

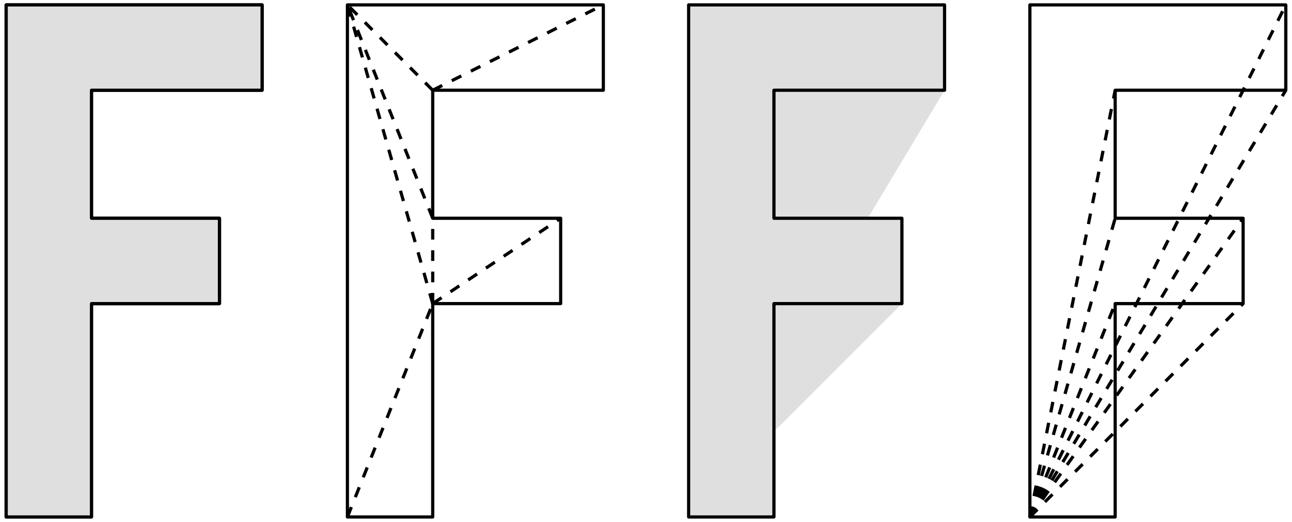
A **polygon triangulation** is the decomposition of a polygon into triangles. This is done by inserting internal diagonals.

An **internal diagonal** is any segment...

- connecting two vertices of the polygon and
- completely enclosed in the polygon.

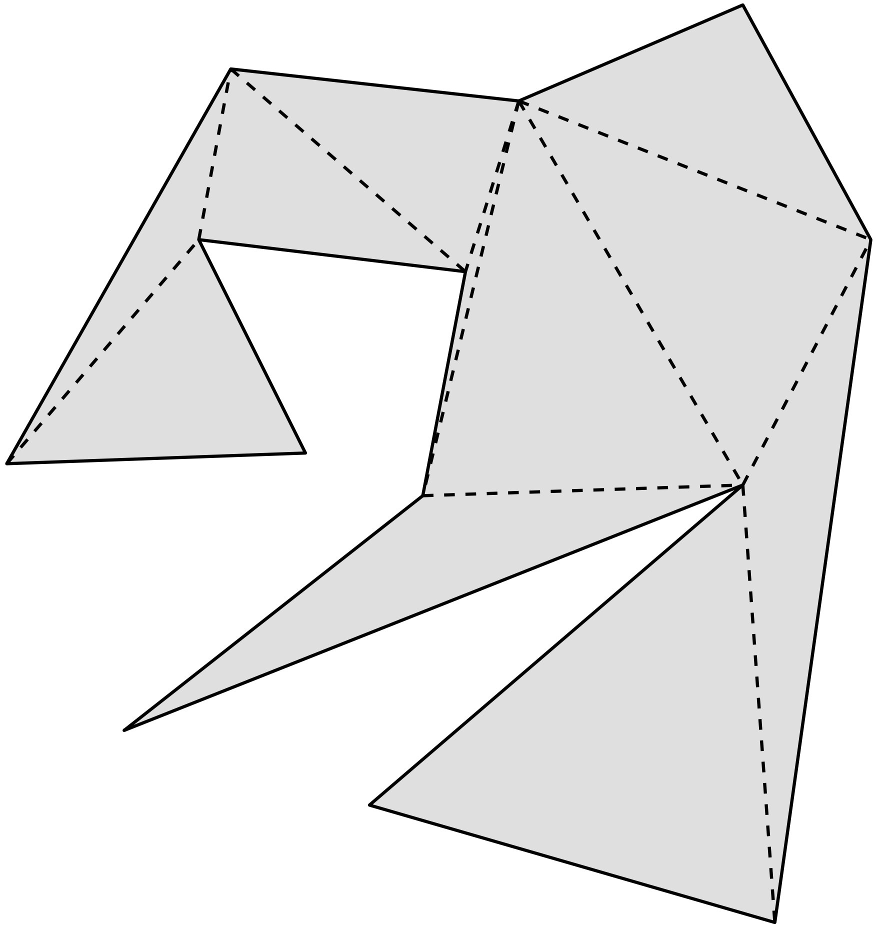


An application example:



TRIANGULATING POLYGONS

Index



TRIANGULATING POLYGONS

Every polygon admits a triangulation → descomposición en triángulos.

TRIANGULATING POLYGONS

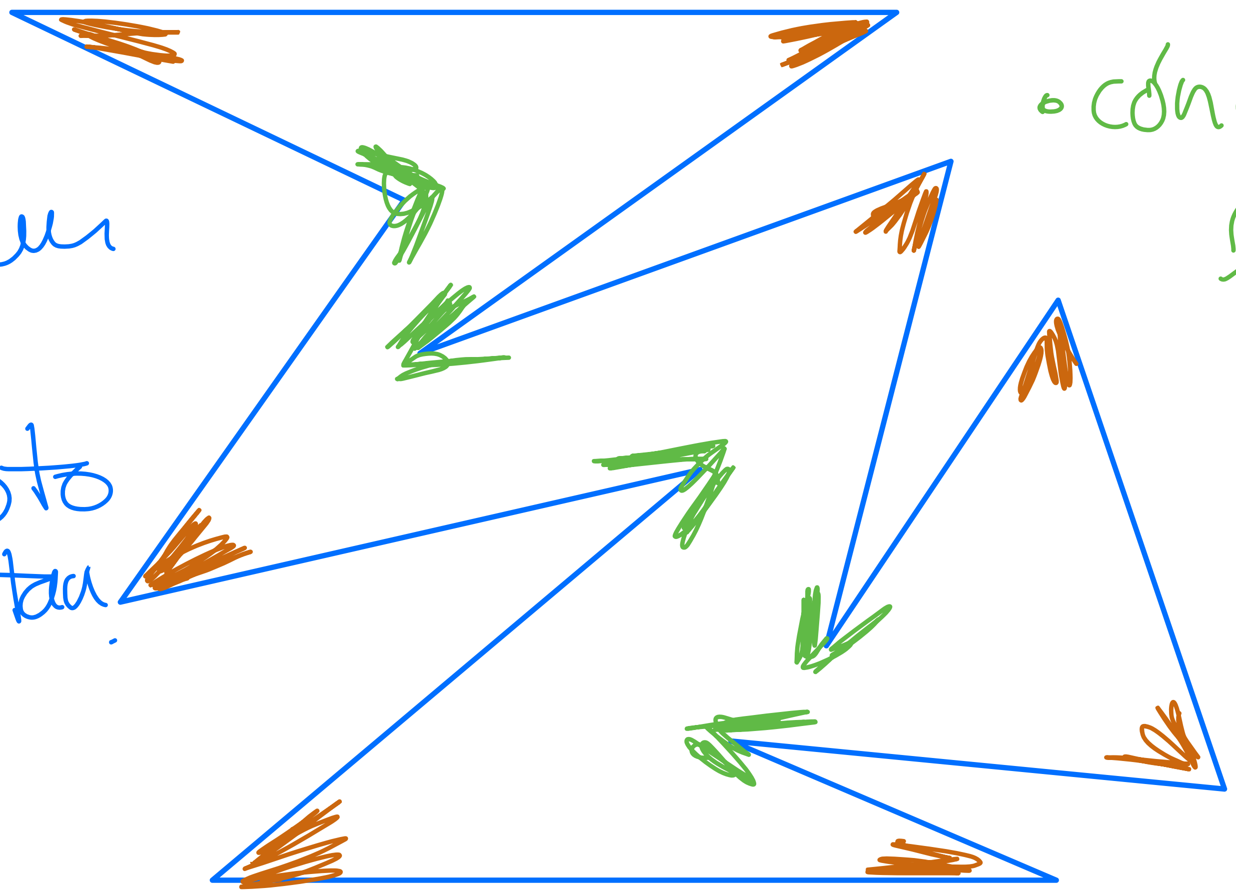
Every polygon admits a triangulation

Lemma 1. Every polygon has at least one convex vertex (actually, at least three).

- polígonos simples:

No tienen hoyos

No se auto intersectan.



menor de 180°

- cóncavo.

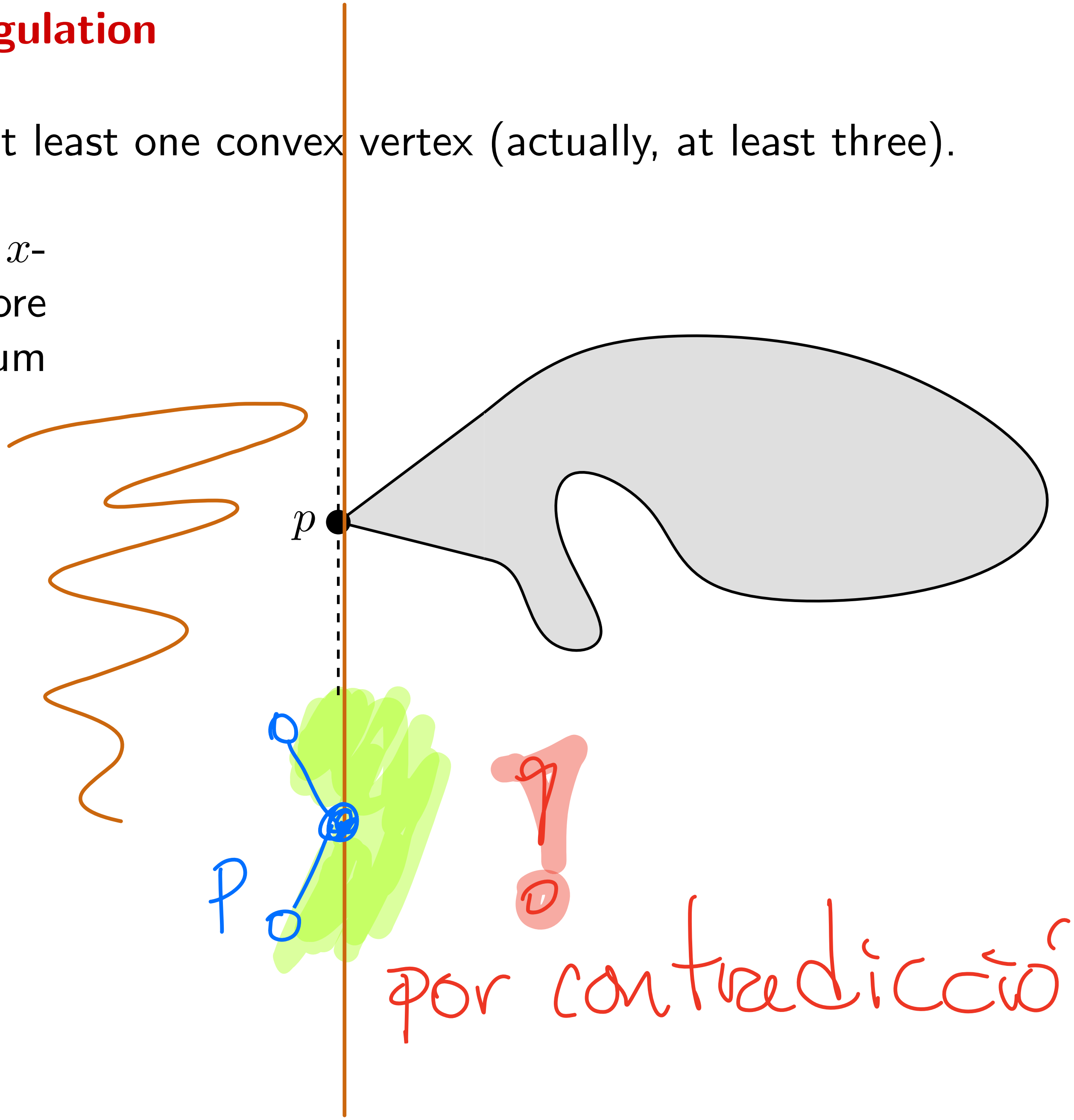
↗ > 180°

TRIANGULATING POLYGONS

Every polygon admits a triangulation

Lemma 1. Every polygon has at least one convex vertex (actually, at least three).

The vertex p with minimum x-coordinate (and, if there are more than one, the one with minimum y-coordinate) is convex.



vacío

?!

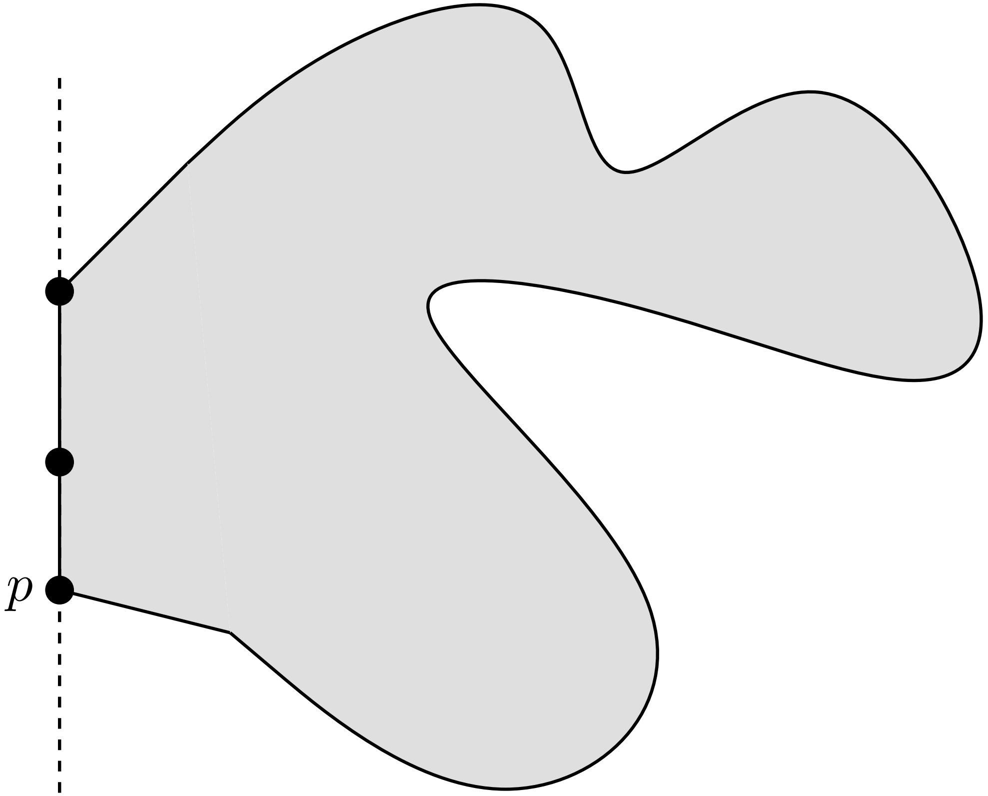
por contradicción.

TRIANGULATING POLYGONS

Every polygon admits a triangulation

Lemma 1. Every polygon has at least one convex vertex (actually, at least three).

The vertex p with minimum x-coordinate (and, if there are more than one, the one with minimum y-coordinate) is convex.



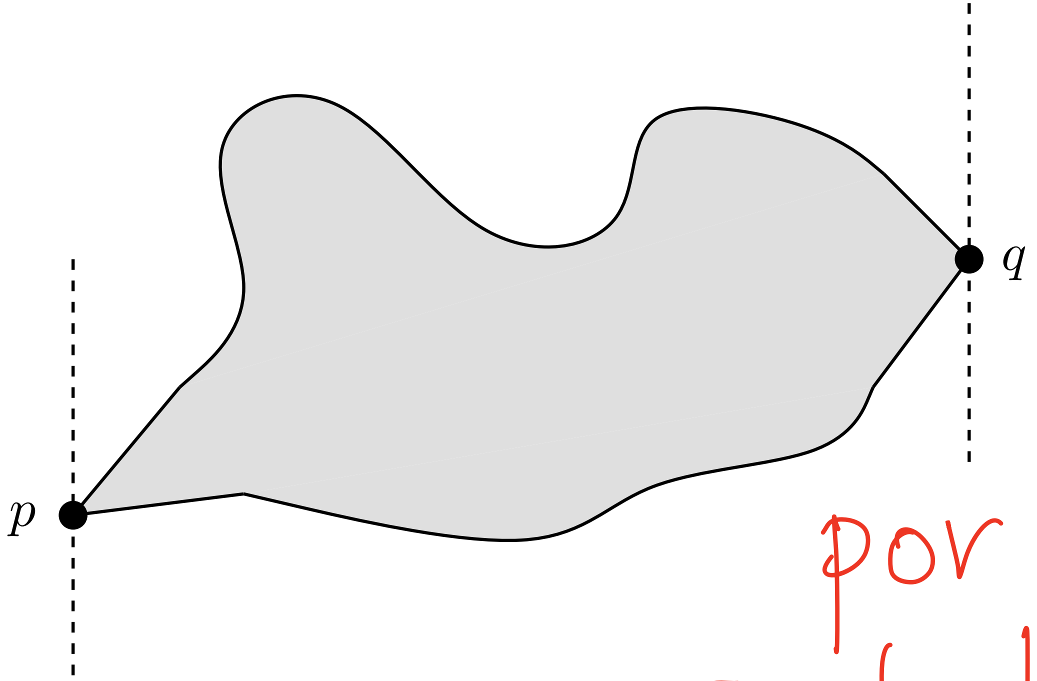
TRIANGULATING POLYGONS

Every polygon admits a triangulation

Lemma 1. Every polygon has at least one convex vertex (actually, at least three).

The vertex p with minimum x-coordinate (and, if there are more than one, the one with minimum y-coordinate) is convex.

So is the vertex q with maximum x-coordinate (which is different of the one with minimum one).



por contradicción,

TRIANGULATING POLYGONS

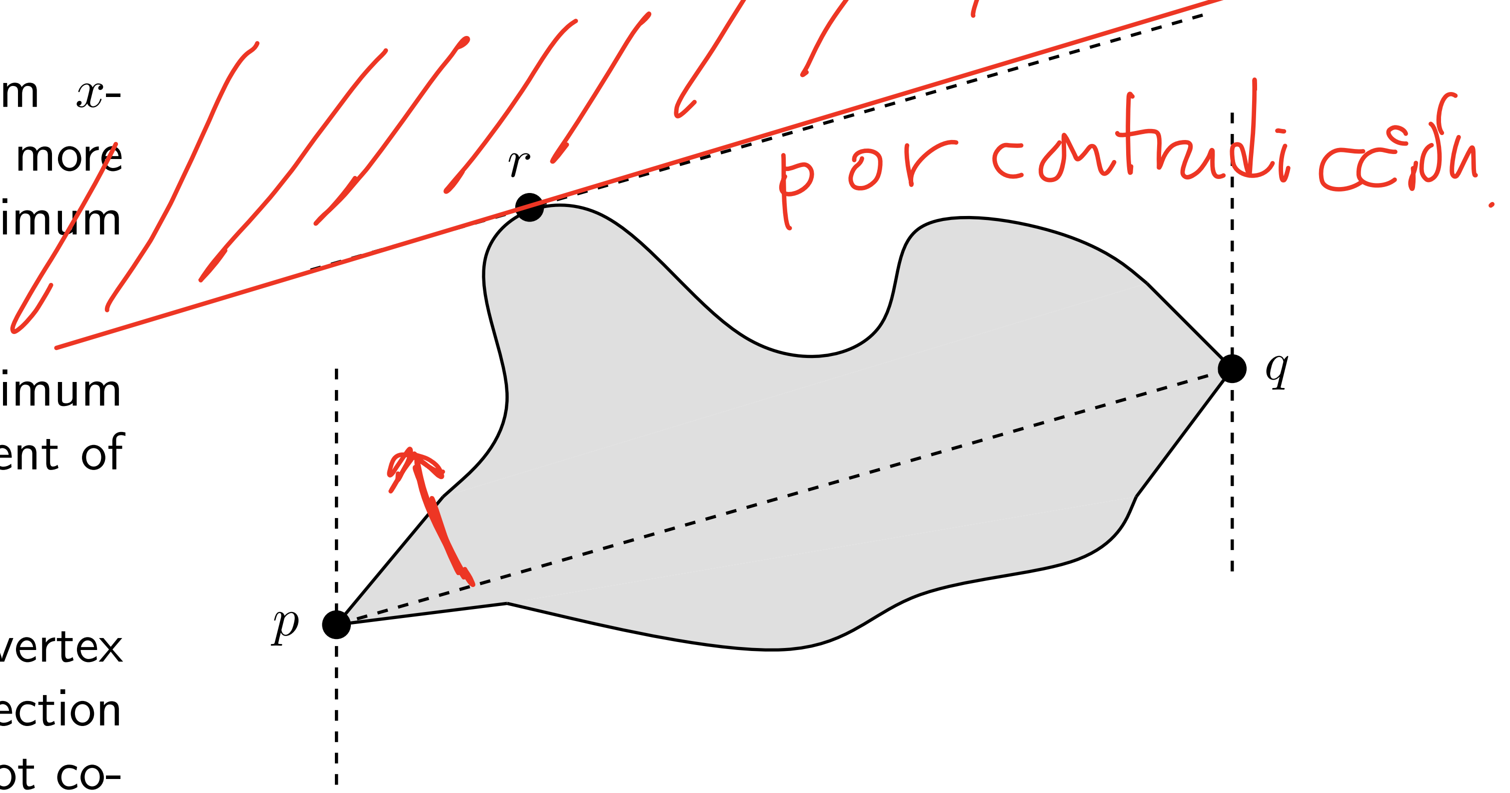
Every polygon admits a triangulation

Lemma 1. Every polygon has at least one convex vertex (actually, at least three).

The vertex p with minimum x-coordinate (and, if there are more than one, the one with minimum y-coordinate) is convex.

So is the vertex q with maximum x-coordinate (which is different of the one with minimum one).

Finally, there is at least one vertex which is extreme in the direction orthogonal to pq and does not coincide with any of the above. This third vertex r is necessarily convex.



por contradicción.

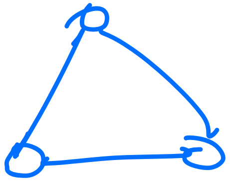
TRIANGULATING POLYGONS

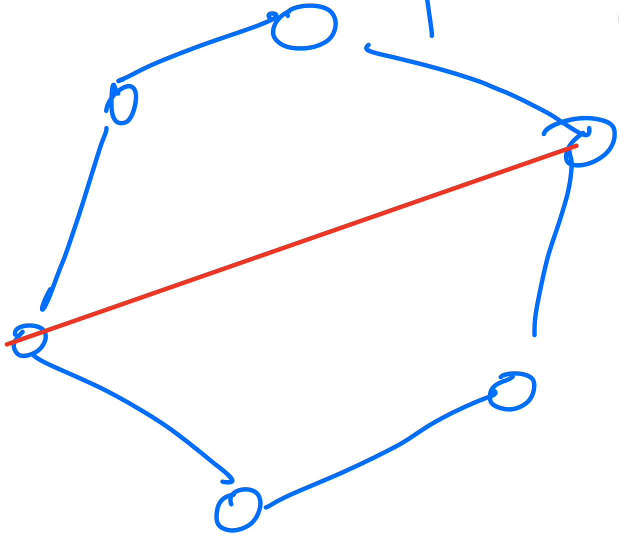
Every polygon admits a triangulation

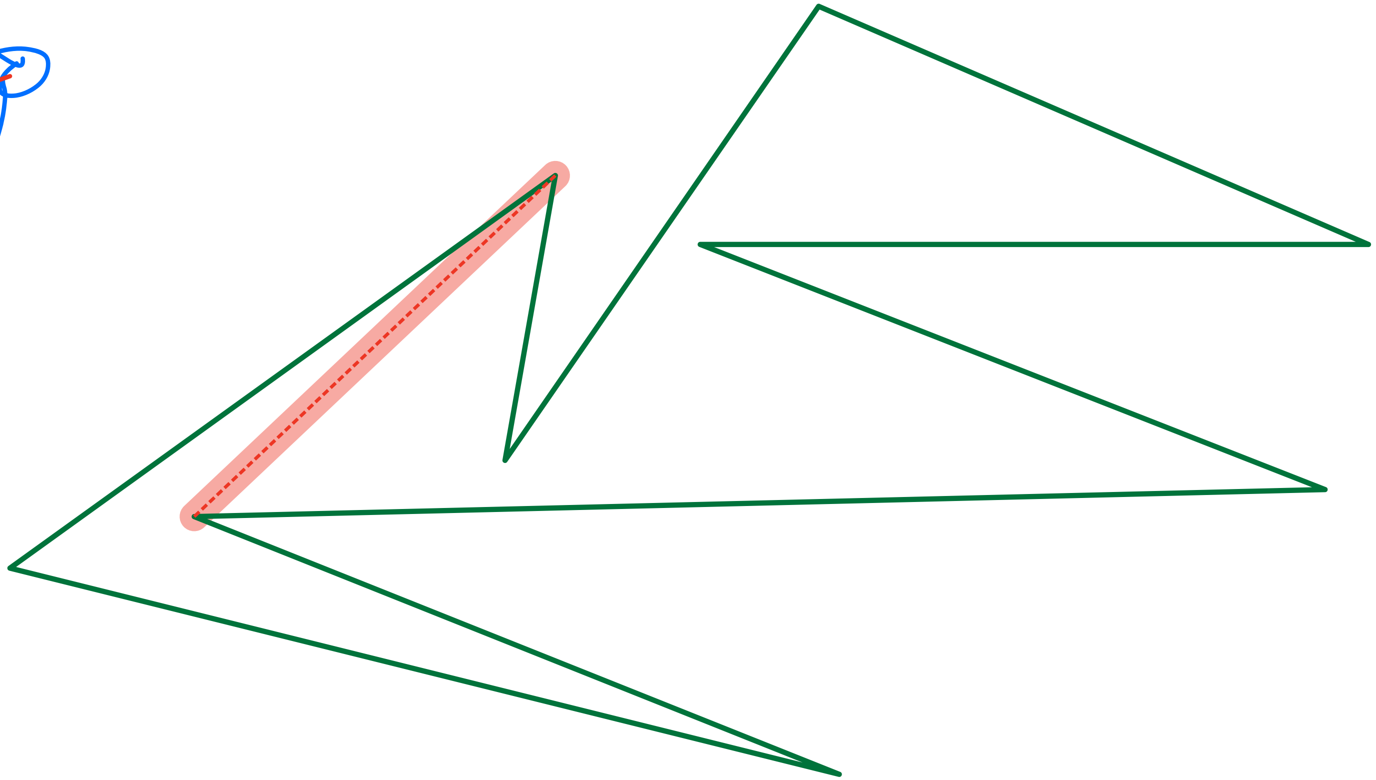
Lemma 1. Every polygon has at least one convex vertex (actually, at least three).

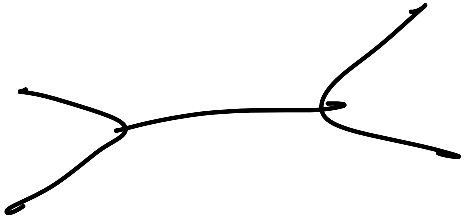
Lemma 2. Every n-gon with $n \geq 4$ has at least one internal diagonal.

polígono con n vértices.









TRIANGULATING POLYGONS

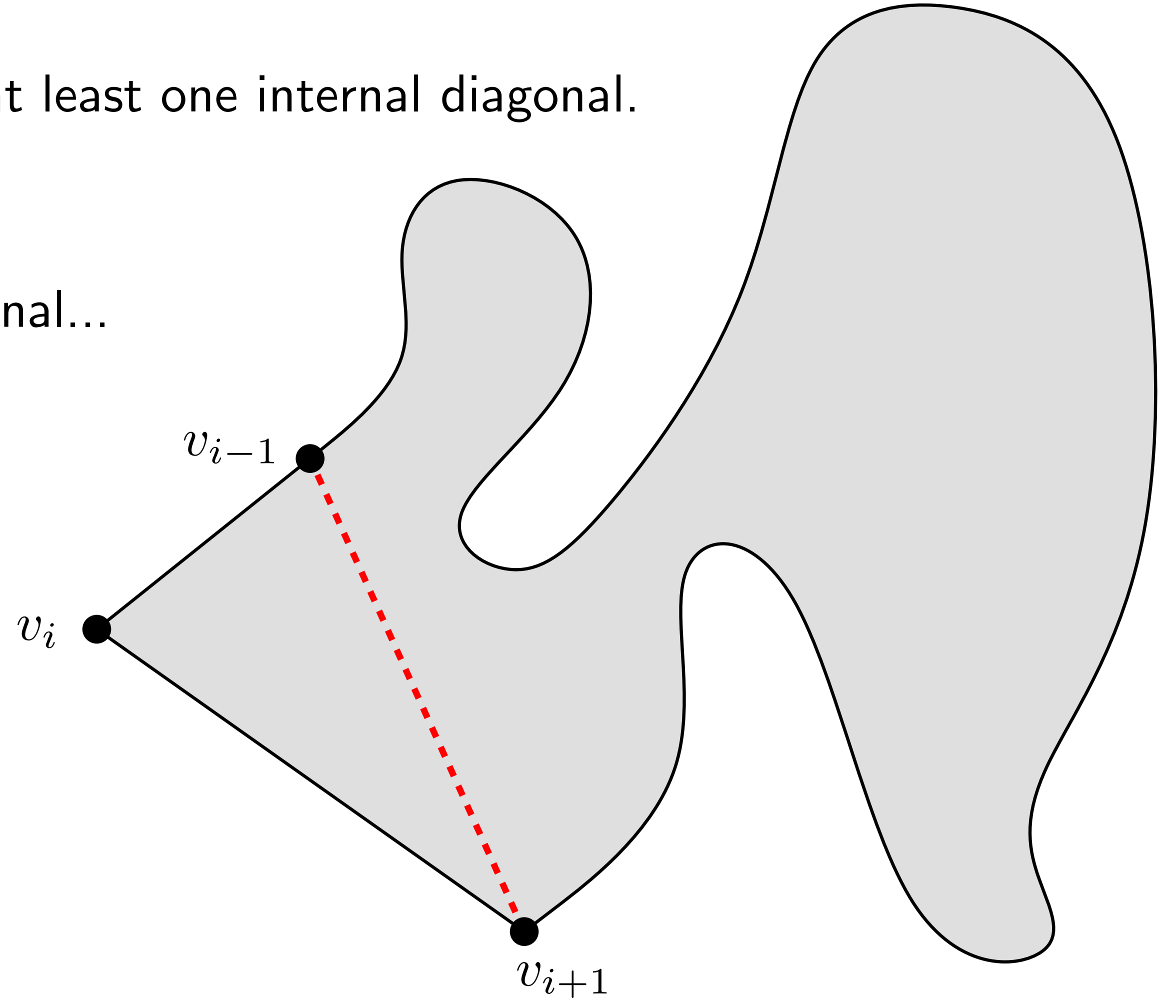
Every polygon admits a triangulation

Lemma 1. Every polygon has at least one convex vertex (actually, at least three).

Lemma 2. Every n-gon with $n \geq 4$ has at least one internal diagonal.

Let v_i be a convex vertex.

Then, either $v_{i-1}v_{i+1}$ is an internal diagonal...



TRIANGULATING POLYGONS

Every polygon admits a triangulation

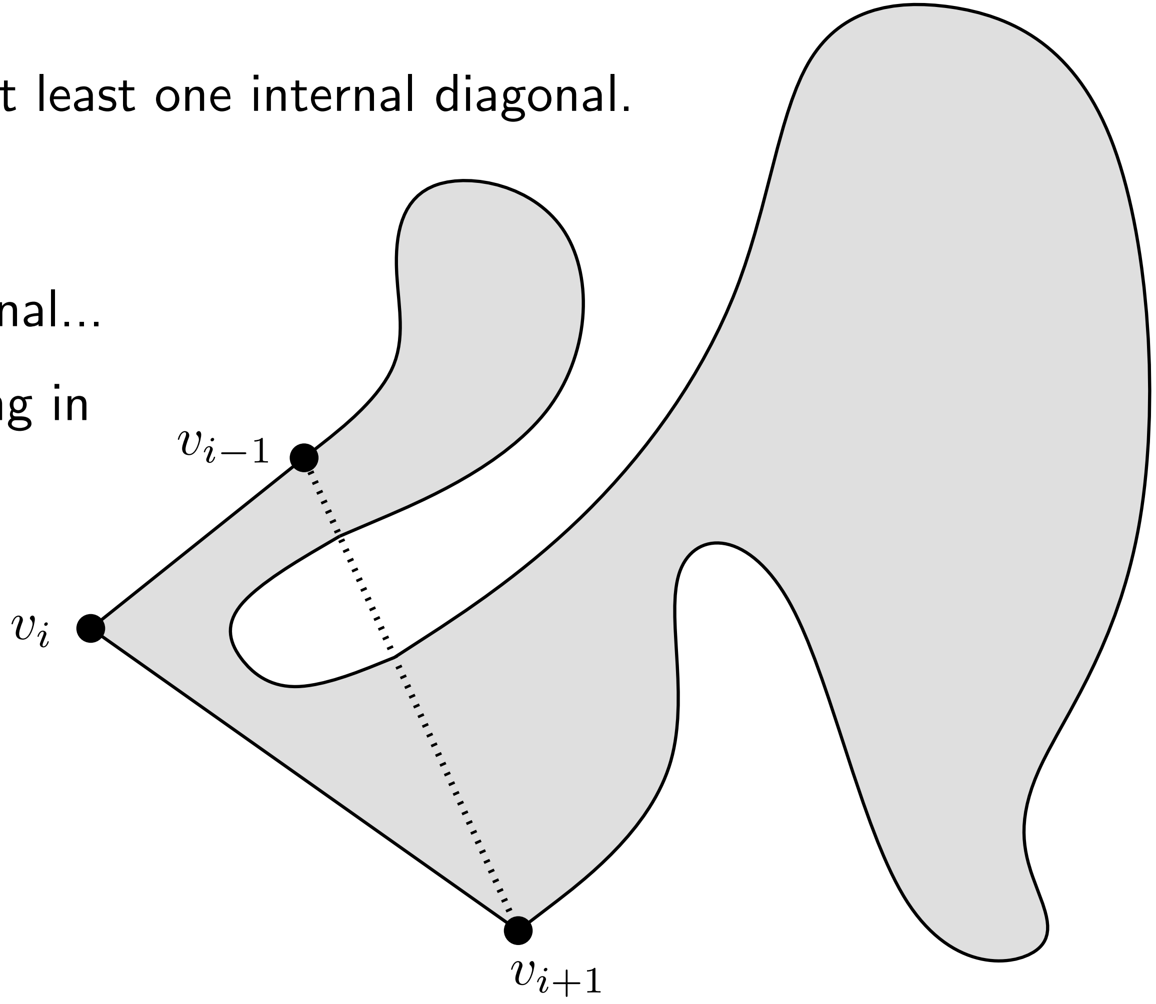
Lemma 1. Every polygon has at least one convex vertex (actually, at least three).

Lemma 2. Every n-gon with $n \geq 4$ has at least one internal diagonal.

Let v_i be a convex vertex.

Then, either $v_{i-1}v_{i+1}$ is an internal diagonal...

or there exists a vertex of the polygon lying in the triangle $v_{i-1}v_iv_{i+1}$.

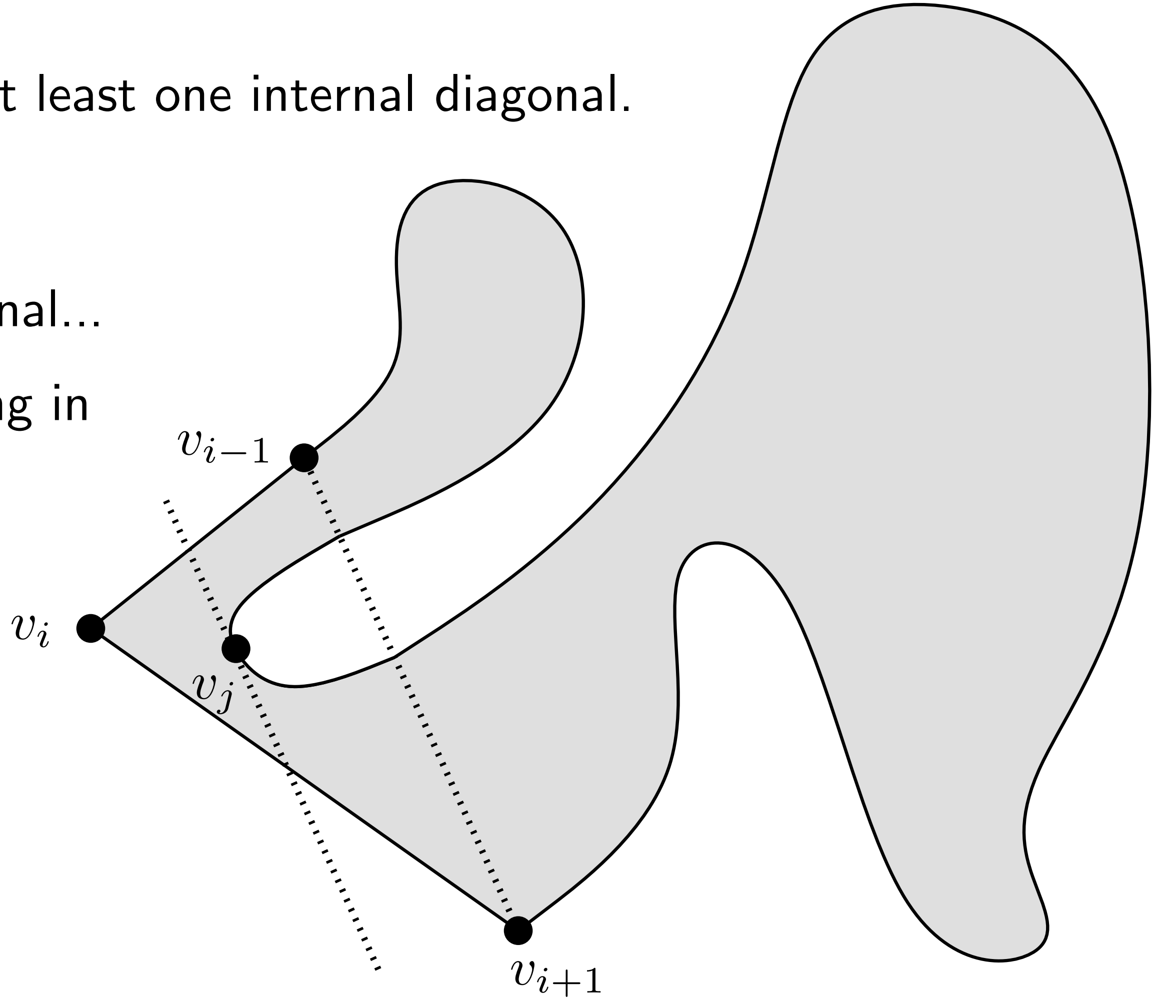


TRIANGULATING POLYGONS

Every polygon admits a triangulation

Lemma 1. Every polygon has at least one convex vertex (actually, at least three).

Lemma 2. Every n-gon with $n \geq 4$ has at least one internal diagonal.



Let v_i be a convex vertex.

Then, either $v_{i-1}v_{i+1}$ is an internal diagonal...

or there exists a vertex of the polygon lying in the triangle $v_{i-1}v_iv_{i+1}$.

In this case, among all the vertices lying in the triangle, let v_j be the farthest one from the segment $v_{i-1}v_{i+1}$. Then v_iv_j is an internal diagonal (it can not be intersected by any edge of the polygon).

TRIANGULATING POLYGONS

Every polygon admits a triangulation

Lemma 1. Every polygon has at least one convex vertex (actually, at least three).

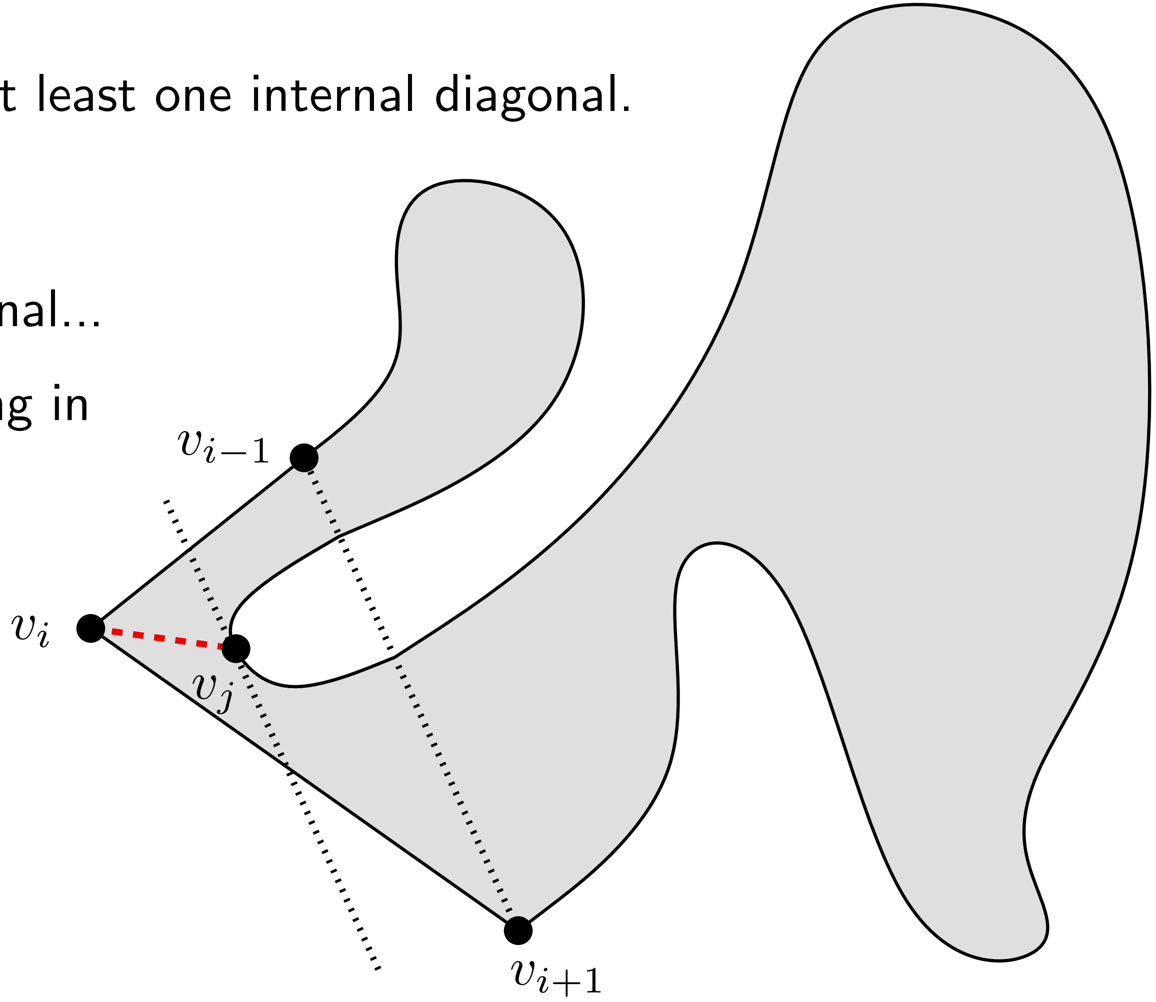
Lemma 2. Every n-gon with $n \geq 4$ has at least one internal diagonal.

Let v_i be a convex vertex.

Then, either $v_{i-1}v_{i+1}$ is an internal diagonal...

or there exists a vertex of the polygon lying in the triangle $v_{i-1}v_iv_{i+1}$.

In this case, among all the vertices lying in the triangle, let v_j be the farthest one from the segment $v_{i-1}v_{i+1}$. Then v_iv_j is an internal diagonal (it can not be intersected by any edge of the polygon).



TRIANGULATING POLYGONS

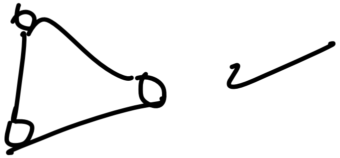
Every polygon admits a triangulation

Lemma 1. Every polygon has at least one convex vertex (actually, at least three).

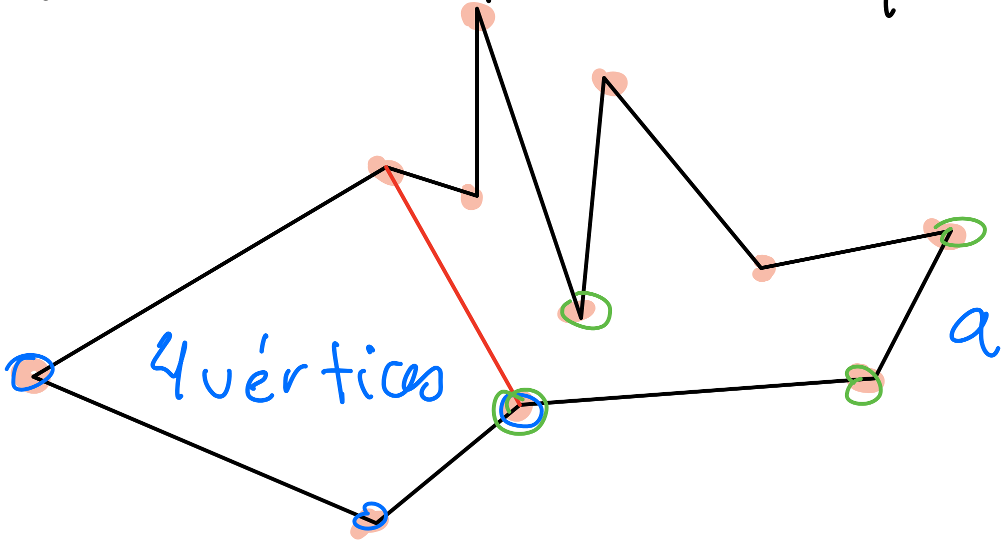
Lemma 2. Every n-gon with $n \geq 4$ has at least one internal diagonal.

Corollary. Every polygon can be triangulated. (By induction.) → sobre n.

Caso base $n=3$.



Para $n \geq 3$ $P = P_1 \cup P_2$



$n=11$.

4vértices

9vértices.

TRIANGULATING POLYGONS

Properties of the triangulations of polygons

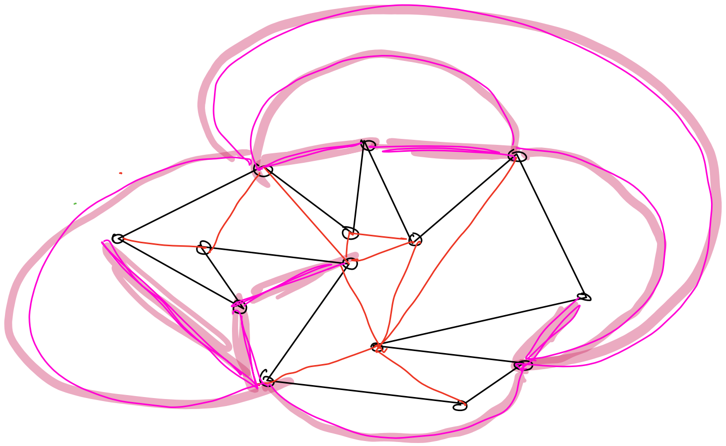
TRIANGULATING POLYGONS

Properties of the triangulations of polygons

Let P be a simple n-gon.

Property 1. Every triangulation of P has $n-3$ diagonals.

$V - E + F = 2$

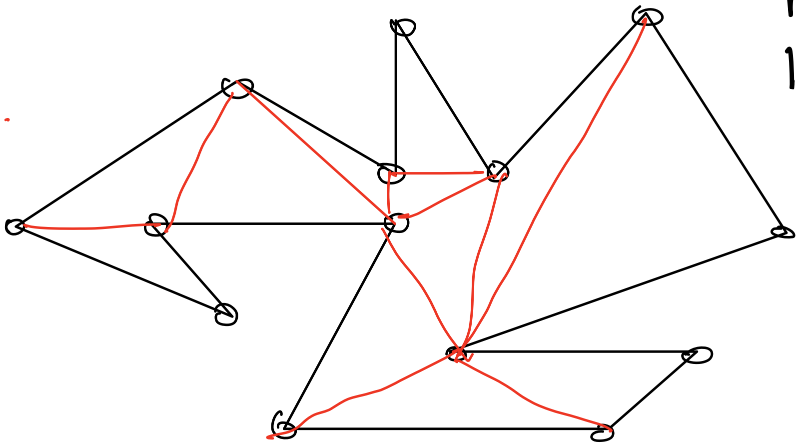


aristas = 14

$n = 14$

11 diagonals

12 Δ's.



¿ "Identidad de Euler"?

TRIANGULATING POLYGONS

Properties of the triangulations of polygons

Let P be a simple n-gon.

Property 1. Every triangulation of P has $n-3$ diagonals.

Proof by induction.

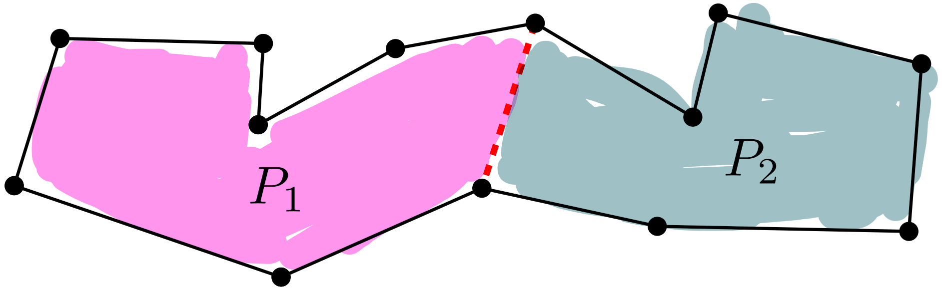
Base case: When $n=3$, the number of diagonals is $d=0=n-3$.

Inductive step: Consider a diagonal of a triangulation T of P, decomposing P into two subpolygons: a $(k+1)$-gon P_1 and an $(n-k+1)$-gon P_2. By inductive hypothesis, the number of diagonals of the triangulations induced by T in P_1 and P_2 are:

$$d_1 = k+1-3,$$
$$d_2 = n-k+1-3,$$

therefore, $d = d_1 + d_2 + 1 = k+1-3+n-k+1-3+1 = n-3$. ✓

k+1
vértices.



n- k+1
vértices.

TRIANGULATING POLYGONS

Properties of the triangulations of polygons

Let P be a simple n-gon.

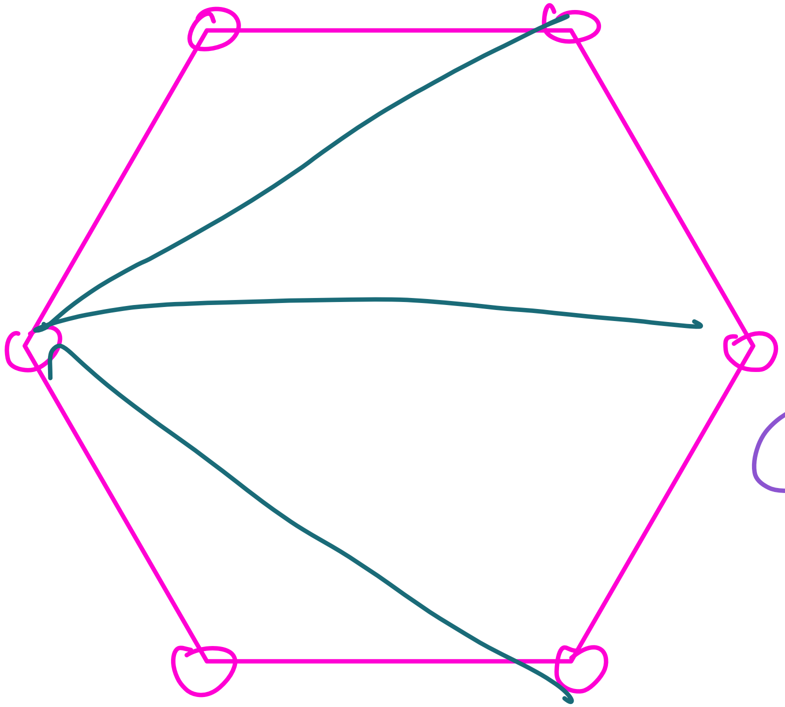
Property 1. Every triangulation of P has $n-3$ diagonals.

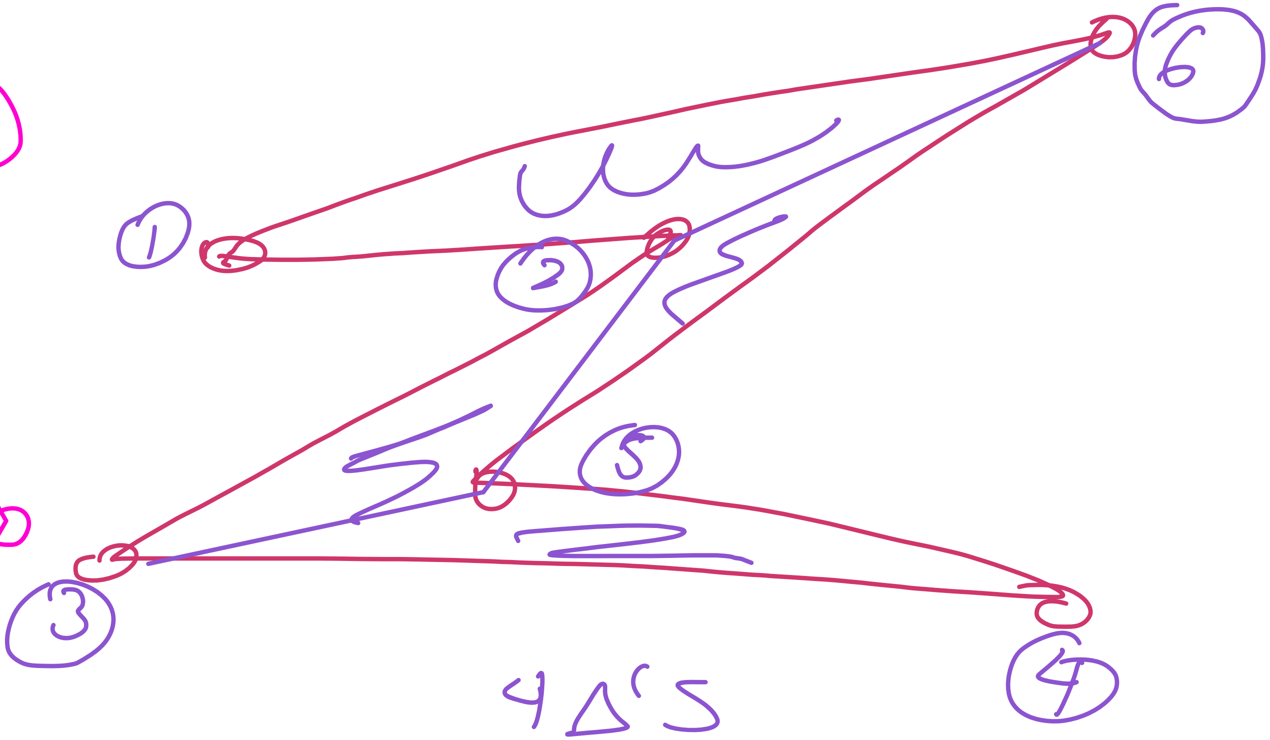
Property 2. Every triangulation of P has $n-2$ triangles.

$\underline{n=6}$

n

$n+(n-3)$





4 Δ's

TRIANGULATING POLYGONS

Properties of the triangulations of polygons

Let P be a simple n-gon.

Property 1. Every triangulation of P has $n-3$ diagonals.

Property 2. Every triangulation of P has $n-2$ triangles.

Teorema. (Identidad de Euler).

Para cada gráfica conexa aplanable con V vértices, E aristas y F caras
Ocurre que:

$$V-E+F=2.$$

Teorema: Si G es una gráfica aplanable con $V \geq 3$ y E aristas, entonces

$$E \leq 3V-6.$$

Lema. El # de diagonales exteriores en una gráfica plana maximal es $n-3$.

TRIANGULATING POLYGONS

Properties of the triangulations of polygons

Let P be a simple n-gon.

Property 1. Every triangulation of P has $n - 3$ diagonals.

Property 2. Every triangulation of P has $n - 2$ triangles.

Again, the proof is by induction.

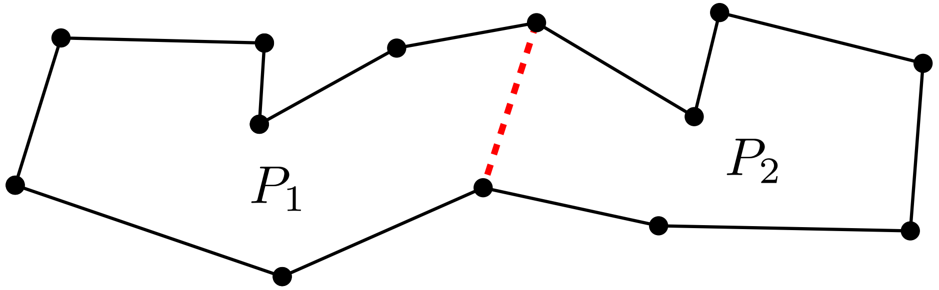
Base case: When $n = 3$, the number of triangles is $t = 1 = n - 2$.

Inductive step: With the same conditions of the previous proof,

$$\begin{aligned} t_1 &= k + 1 - 2, \\ t_2 &= n - k + 1 - 2, \end{aligned}$$

hence,

$$t = t_1 + t_2 = k + 1 - 2 + n - k + 1 - 2 = n - 2.$$



TRIANGULATING POLYGONS

Properties of the triangulations of polygons

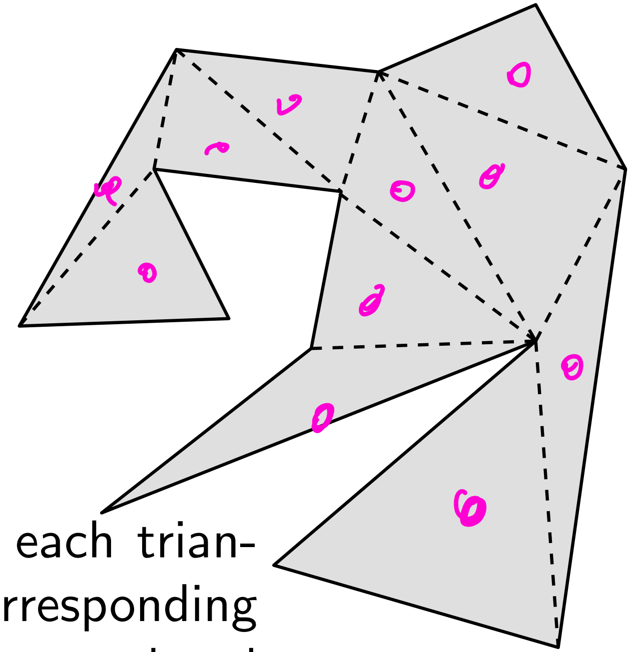
Let P be a simple n-gon.

Property 1. Every triangulation of P has $n-3$ diagonals.

Property 2. Every triangulation of P has $n-2$ triangles.

Property 3. The dual graph of any triangulation of P is a tree.

TRIANGULATING POLYGONS



Properties of the triangulations of polygons

Let P be a simple n-gon.

Property 1. Every triangulation of P has $n-3$ diagonals.

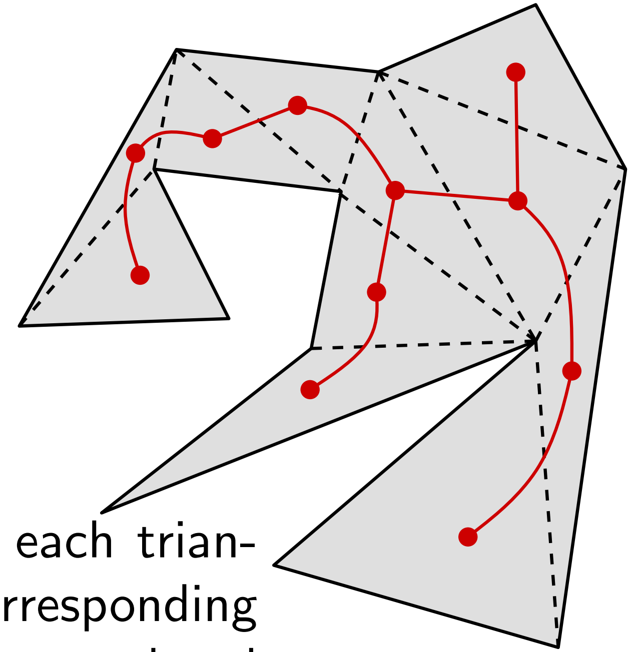
Property 2. Every triangulation of P has $n-2$ triangles.

Property 3. The dual graph of any triangulation of P is a tree.

Given a triangulation of P, its dual graph has one vertex for each triangle, and one edge connecting two vertices whenever their corresponding triangles are adjacent. We want to prove that this graph is connected and acyclic.

TRIANGULATING POLYGONS

Properties of the triangulations of polygons



Let P be a simple n-gon.

Property 1. Every triangulation of P has $n-3$ diagonals.

Property 2. Every triangulation of P has $n-2$ triangles.

Property 3. The dual graph of any triangulation of P is a tree.

Given a triangulation of P, its dual graph has one vertex for each triangle, and one edge connecting two vertices whenever their corresponding triangles are adjacent. We want to prove that this graph is connected and acyclic.

$V =$ un vértice por cada triangulo

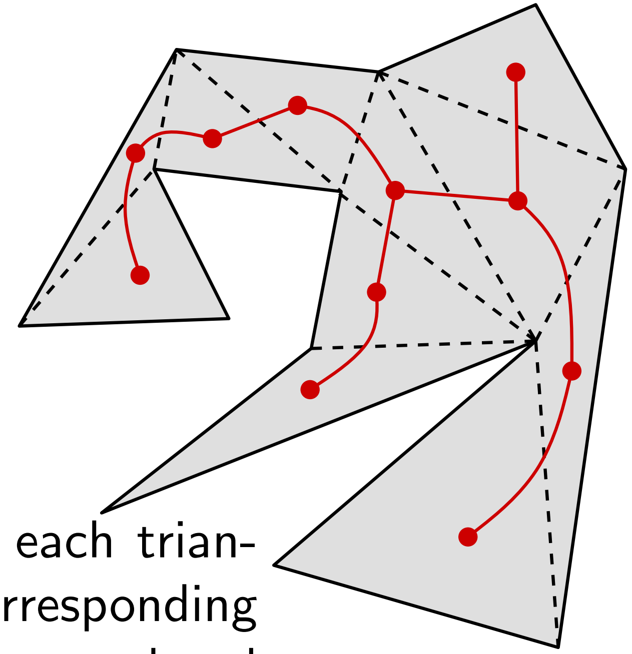
$E = \{ (v_i, v_j)$ | el triángulo correspond. a v_i y el △ corresp. a v_j tienen una diagonal en común?

$|V| = n-2$; $|E| = n-3$

① conexa
② acíclica. } para demostrar que es un árbol.

TRIANGULATING POLYGONS

Properties of the triangulations of polygons



Let P be a simple n-gon.

Property 1. Every triangulation of P has $n-3$ diagonals.

Property 2. Every triangulation of P has $n-2$ triangles.

Property 3. The dual graph of any triangulation of P is a tree.

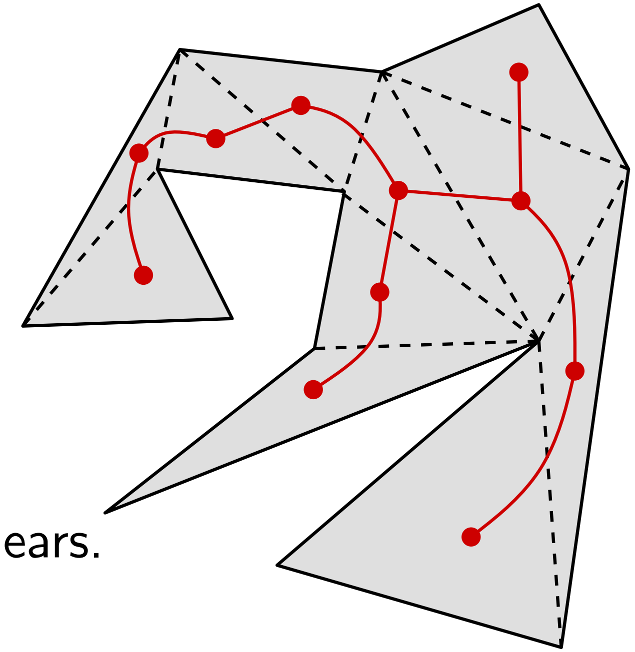
Given a triangulation of P, its dual graph has one vertex for each triangle, and one edge connecting two vertices whenever their corresponding triangles are adjacent. We want to prove that this graph is connected and acyclic.

The graph is trivially connected.

About the acyclicity: Notice that each edge of the dual graph "separates" the two endpoints of the internal diagonal of P shared by the two adjacent triangles. If the graph had a cycle, it would enclose the endpoint(s) of the diagonals intersected by the cycle and, therefore, it would enclose points belonging to the boundary of the polygon, contradicting the hypothesis that P is simple and without holes.

TRIANGULATING POLYGONS

Properties of the triangulations of polygons



Let P be a simple n-gon.

Property 1. Every triangulation of P has $n-3$ diagonals.

Property 2. Every triangulation of P has $n-2$ triangles.

Property 3. The dual graph of any triangulation of P is a tree.

Corollary. Every n-gon with $n \geq 4$ has at least two non-adjacent ears.

ALGORITHMS FOR POLYGON TRIANGULATION

TRIANGULATING POLYGONS

Tringulating a polygon by subtracting ears

TRIANGULATING POLYGONS

Tringulating a polygon by subtracting ears

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

TRIANGULATING POLYGONS

Tringulating a polygon by subtracting ears

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure

1. Sequentially explore the vertices until you find an ear
2. Crop it
3. Proceed recursively

TRIANGULATING POLYGONS

Tringulating a polygon by subtracting ears

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure

1. Sequentially explore the vertices until you find an ear
2. Crop it
3. Proceed recursively

Running time

TRIANGULATING POLYGONS

Tringulating a polygon by subtracting ears

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure

1. Sequentially explore the vertices until you find an ear
2. Crop it
3. Proceed recursively

Running time

Detecting whether a vertex is convex: $O(1)$.

TRIANGULATING POLYGONS

Tringulating a polygon by subtracting ears

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure

1. Sequentially explore the vertices until you find an ear
2. Crop it
3. Proceed recursively

Running time

Detecting whether a vertex is convex: $O(1)$.

Detecting whether a convex vertex is an ear: $O(n)$.

TRIANGULATING POLYGONS

Tringulating a polygon by subtracting ears

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure

1. Sequentially explore the vertices until you find an ear
2. Crop it
3. Proceed recursively

Running time

Detecting whether a vertex is convex: $O(1)$.
Detecting whether a convex vertex is an ear: $O(n)$.
Finding an ear: $O(n^2)$.

TRIANGULATING POLYGONS

Tringulating a polygon by subtracting ears

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure

1. Sequentially explore the vertices until you find an ear
2. Crop it
3. Proceed recursively

Running time

Detecting whether a vertex is convex: $O(1)$.
Detecting whether a convex vertex is an ear: $O(n)$.
Finding an ear: $O(n^2)$.

Overall running time:

$$T(n) = O(n^2) + O((n-1)^2) + O((n-2)^2) + \cdots + O(1) = O(n^3).$$

TRIANGULATING POLYGONS

Tringulating a polygon by subtracting ears

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Improved procedure

Initialization

1. Detect all convex vertices
2. Detect all ears

Next step

1. Crop an ear
2. Update the information of the convex vertices
3. Update the information of the ears

TRIANGULATING POLYGONS

Tringulating a polygon by subtracting ears

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Improved procedure	**Running time**	
Initialization		
1. Detect all convex vertices	$O(n)$	Only once
2. Detect all ears	$O(n^2)$	
Next step		
1. Crop an ear	$O(1)$	$O(n)$ times
2. Update the information of the convex vertices	$O(1)$	
3. Update the information of the ears	$O(n)$	

TRIANGULATING POLYGONS

Tringulating a polygon by subtracting ears

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, v_iv_j, determining a triangulation of P.

Improved procedure	**Running time**	
Initialization		
1. Detect all convex vertices	$O(n)$	Only once
2. Detect all ears	$O(n^2)$	
Next step		
1. Crop an ear	$O(1)$	
2. Update the information of the convex vertices	$O(1)$	$O(n)$ times
3. Update the information of the ears	$O(n)$	

Running time: $O(n^2)$

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, v_iv_j, determining a triangulation of P.

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \dots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

Test. How to decide whether a given segment $v_i v_j$ is an internal diagonal?

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

Test. How to decide whether a given segment $v_i v_j$ is an internal diagonal?

Is it a diagonal?

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \dots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

Test. How to decide whether a given segment $v_i v_j$ is an internal diagonal?

Is it a diagonal?

Check $v_i v_j$ against all segments $v_k v_{k+1}$ for intersection.

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \dots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

Test. How to decide whether a given segment $v_i v_j$ is an internal diagonal?

Is it a diagonal?

Check $v_i v_j$ against all segments $v_k v_{k+1}$ for intersection.

Is it internal?

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, v_iv_j, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

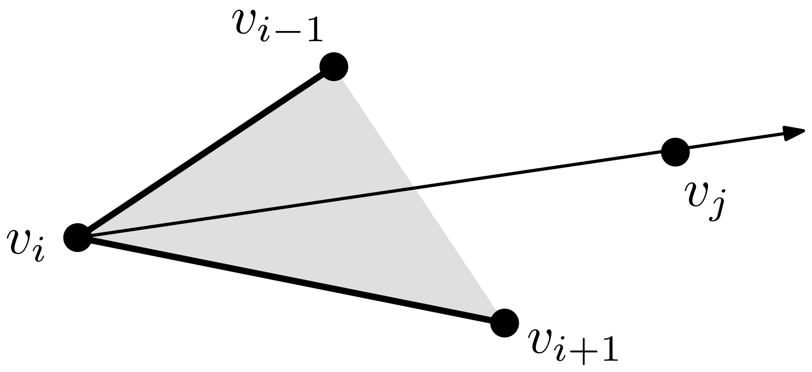
Test. How to decide whether a given segment v_iv_j is an internal diagonal?

Is it a diagonal?

Check v_iv_j against all segments v_kv_{k+1} for intersection.

Is it internal?

If v_i is convex, the oriented line $\overrightarrow{v_iv_j}$ should leave v_{i-1} to its left and v_{i+1} to its right.



TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

Test. How to decide whether a given segment $v_i v_j$ is an internal diagonal?

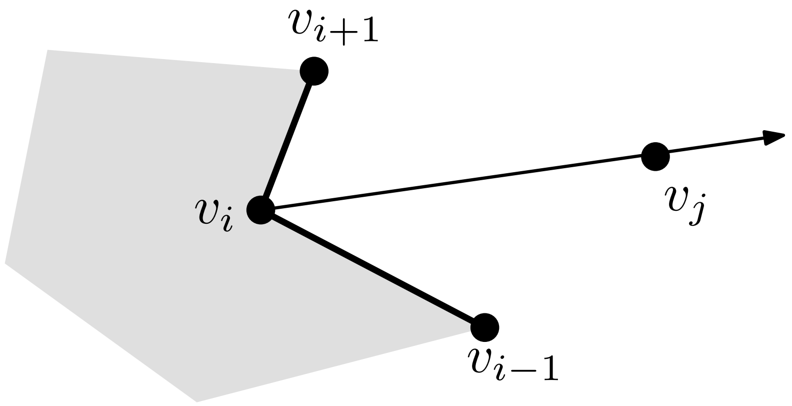
Is it a diagonal?

Check $v_i v_j$ against all segments $v_k v_{k+1}$ for intersection.

Is it internal?

If v_i is convex, the oriented line $\overrightarrow{v_i v_j}$ should leave v_{i-1} to its left and v_{i+1} to its right.

If v_i is reflex, the oriented line $\overrightarrow{v_i v_j}$ should not leave v_{i-1} to its right and v_{i+1} to its left.



TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

Running time

Test. How to decide whether a given segment $v_i v_j$ is an internal diagonal?

Is it a diagonal?

Check $v_i v_j$ against all segments $v_k v_{k+1}$ for intersection.

Is it internal?

If v_i is convex, the oriented line $\overrightarrow{v_i v_j}$ should leave v_{i-1} to its left and v_{i+1} to its right.

If v_i is reflex, the oriented line $\overrightarrow{v_i v_j}$ should not leave v_{i-1} to its right and v_{i+1} to its left.

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

Running time

Test. How to decide whether a given segment $v_i v_j$ is an internal diagonal?

$O(n)$

Is it a diagonal?

Check $v_i v_j$ against all segments $v_k v_{k+1}$ for intersection.

Is it internal?

If v_i is convex, the oriented line $\overrightarrow{v_i v_j}$ should leave v_{i-1} to its left and v_{i+1} to its right.

If v_i is reflex, the oriented line $\overrightarrow{v_i v_j}$ should not leave v_{i-1} to its right and v_{i+1} to its left.

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

Running time

Test. How to decide whether a given segment $v_i v_j$ is an internal diagonal? $O(n)$

Search. How to find an internal diagonal?

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

Running time

Test. How to decide whether a given segment $v_i v_j$ is an internal diagonal? $O(n)$

Search. How to find an internal diagonal?

Brute-force solution:

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

Running time

Test. How to decide whether a given segment $v_i v_j$ is an internal diagonal? $O(n)$

Search. How to find an internal diagonal?

Brute-force solution:

Apply the test to each candidate segment.

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

	Running time
Test. How to decide whether a given segment $v_i v_j$ is an internal diagonal?	$O(n)$
Search. How to find an internal diagonal?	
Brute-force solution:	
Apply the test to each candidate segment.	$O(n^3)$

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, v_iv_j, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

Running time

Test. How to decide whether a given segment v_iv_j is an internal diagonal? $O(n)$

Search. How to find an internal diagonal?

Brute-force solution:

Apply the test to each candidate segment. $O(n^3)$

Testing each candidate takes $O(n)$ time, and there are $\binom{n}{2}$ of them.

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

Running time

Test. How to decide whether a given segment $v_i v_j$ is an internal diagonal? $O(n)$

Search. How to find an internal diagonal?

Brute-force solution:

Apply the test to each candidate segment. $O(n^3)$

Applying previous results:

1. Find a convex vertex, v_i.
2. Detect whether $v_{i-1}v_{i+1}$ is an internal diagonal.
3. If so, report it.
   Else, find the farthest v_k from the segment $v_{i-1}v_{i+1}$, lying in the triangle $v_{i-1}v_i v_{i+1}$.

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, v_iv_j, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

Running time

Test. How to decide whether a given segment v_iv_j is an internal diagonal? $O(n)$

Search. How to find an internal diagonal?

Brute-force solution:

Apply the test to each candidate segment. $O(n^3)$

Applying previous results: $O(n)$

1. Find a convex vertex, v_i.
2. Detect whether $v_{i-1}v_{i+1}$ is an internal diagonal.
3. If so, report it.
   Else, find the farthest v_k from the segment $v_{i-1}v_{i+1}$, lying in the triangle $v_{i-1}v_iv_{i+1}$.

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, v_iv_j, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

	Running time
Test. How to decide whether a given segment v_iv_j is an internal diagonal?	$O(n)$
Search. How to find an internal diagonal?	$O(n)$

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, v_iv_j, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

	Running time
Test. How to decide whether a given segment v_iv_j is an internal diagonal?	$O(n)$
Search. How to find an internal diagonal?	$O(n)$
Partition. How to partition the polygon into two subpolygons?	

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, $v_i v_j$, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

	Running time
Test. How to decide whether a given segment $v_i v_j$ is an internal diagonal?	$O(n)$
Search. How to find an internal diagonal?	$O(n)$

Partition. How to partition the polygon into two subpolygons?

From the diagonal found, create the sorted list of the vertices of the two subpolygons.

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \dots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, v_iv_j, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

	Running time
Test. How to decide whether a given segment v_iv_j is an internal diagonal?	$O(n)$
Search. How to find an internal diagonal?	$O(n)$
Partition. How to partition the polygon into two subpolygons?	$O(n)$

From the diagonal found, create the sorted list of the vertices of the two subpolygons.

TRIANGULATING POLYGONS

Tringulating a polygon by inserting diagonals

Input: $v_1, \ldots, v_n$, sorted list of the vertices of a simple polygon P.

Output: List of internal diagonals of P, v_iv_j, determining a triangulation of P.

Procedure:

1. Find an internal diagonal
2. Decompose the polygon into two subpolygons
3. Proceed recursively

	Running time
Test. How to decide whether a given segment v_iv_j is an internal diagonal?	$O(n)$
Search. How to find an internal diagonal?	$O(n)$
Partition. How to partition the polygon into two subpolygons?	$O(n)$

Total running time of the algorithm: $O(n^2)$

It finds $n-3$ diagonals and each one is found in $O(n)$ time.

TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

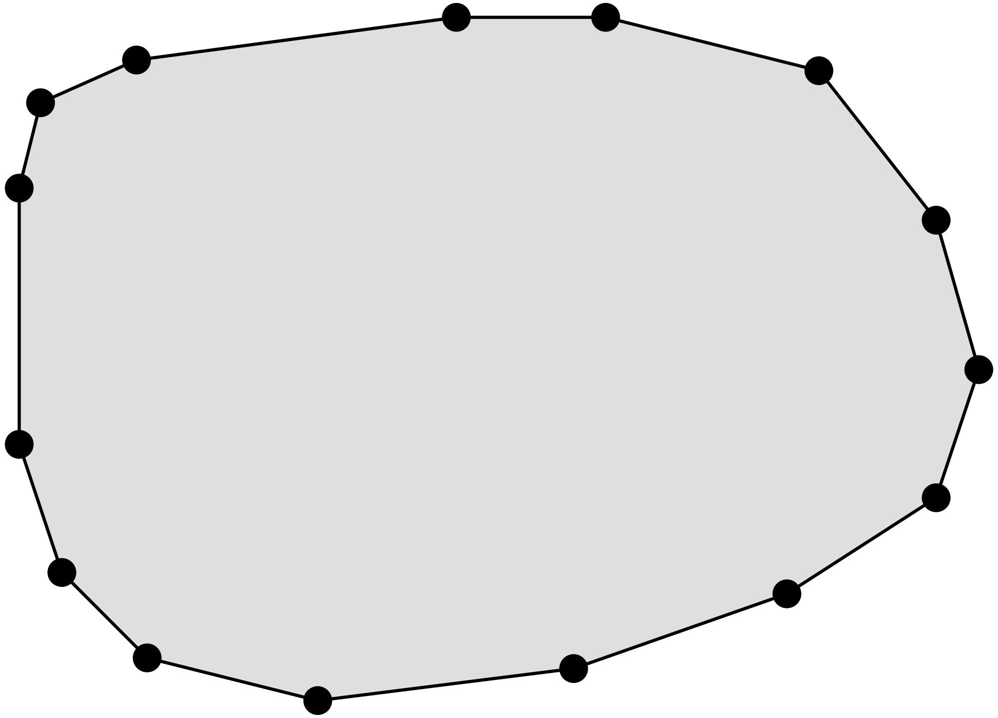
Trivially done in $O(n)$ time.

TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.

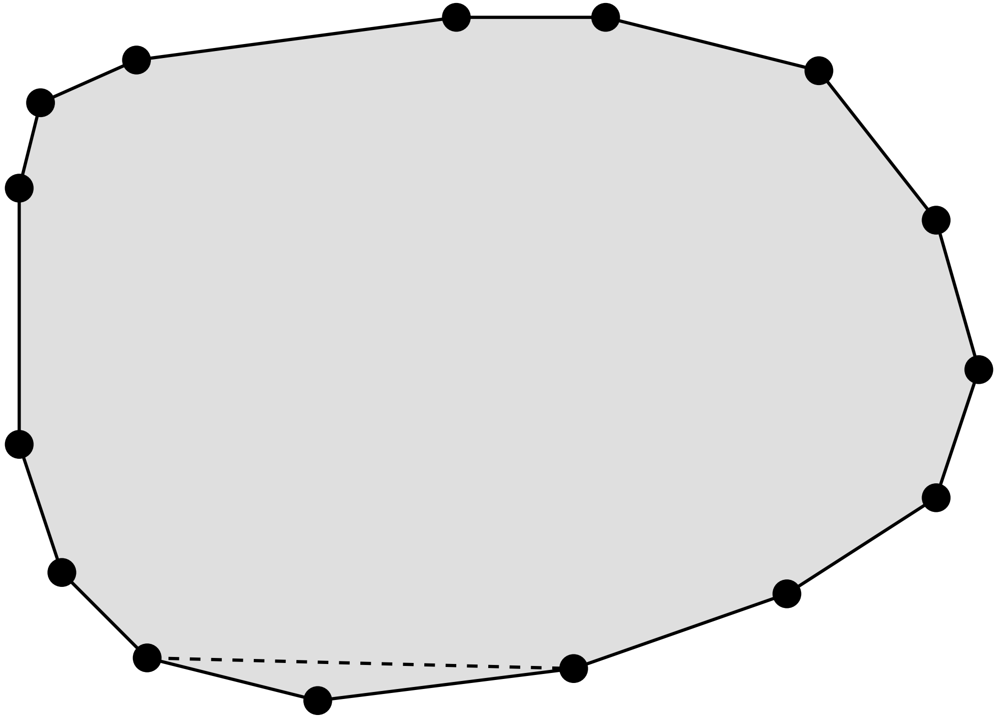


TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.

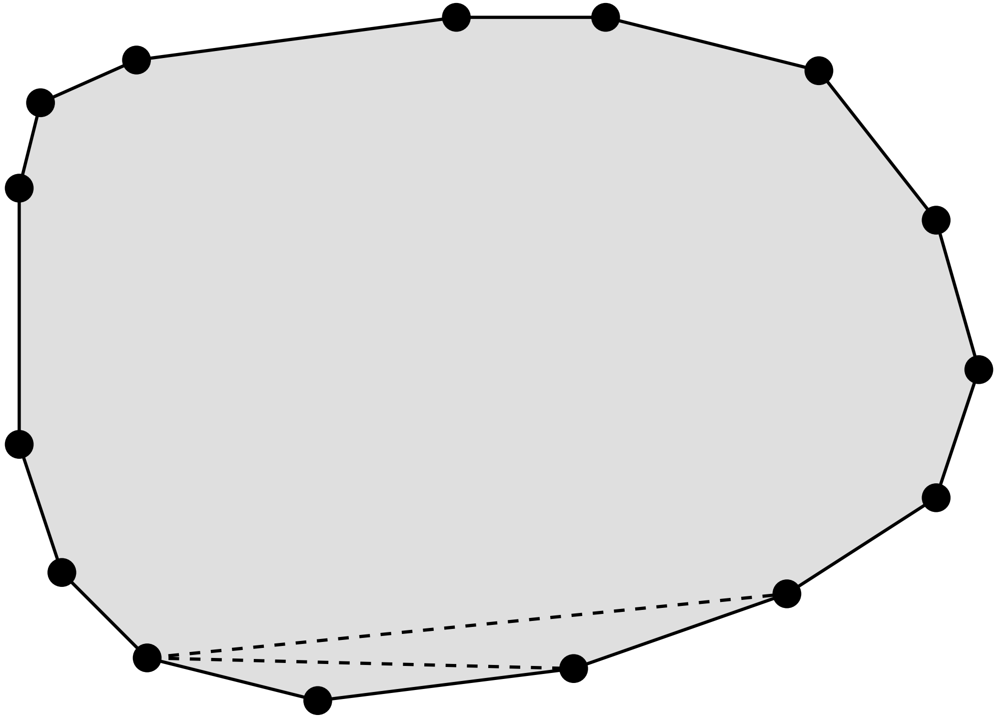


TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.

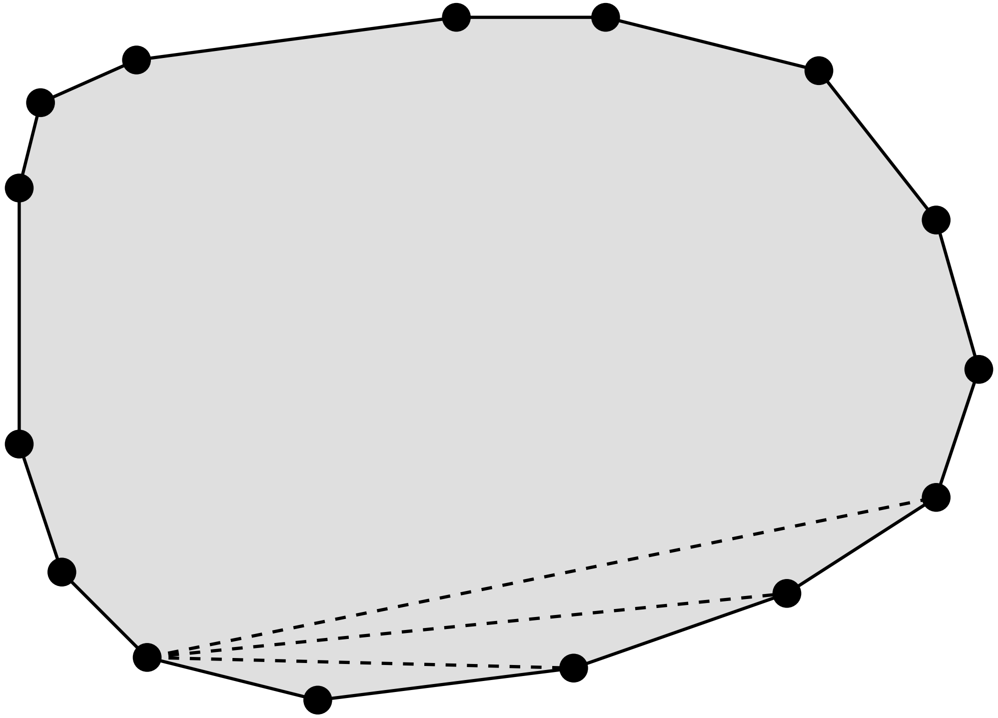


TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.

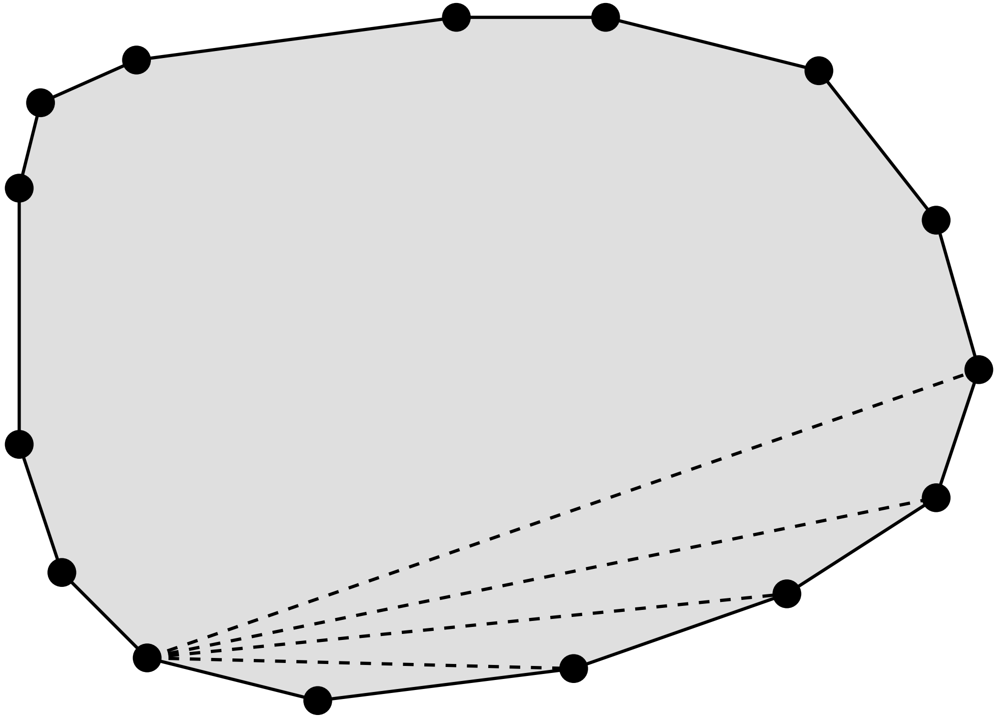


TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.

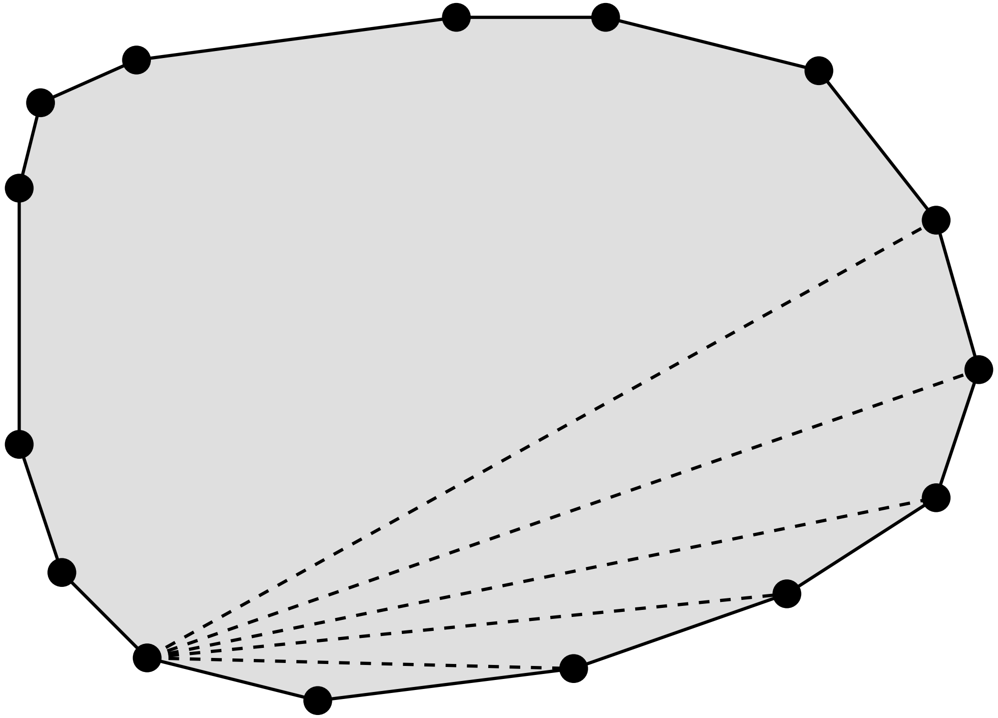


TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.

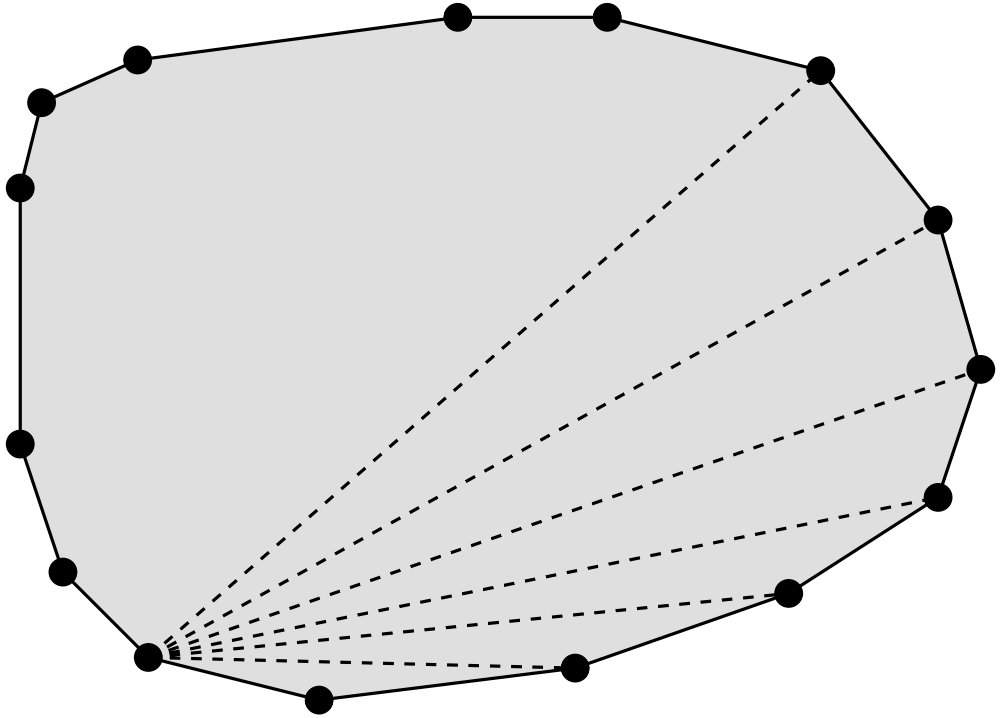


TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.

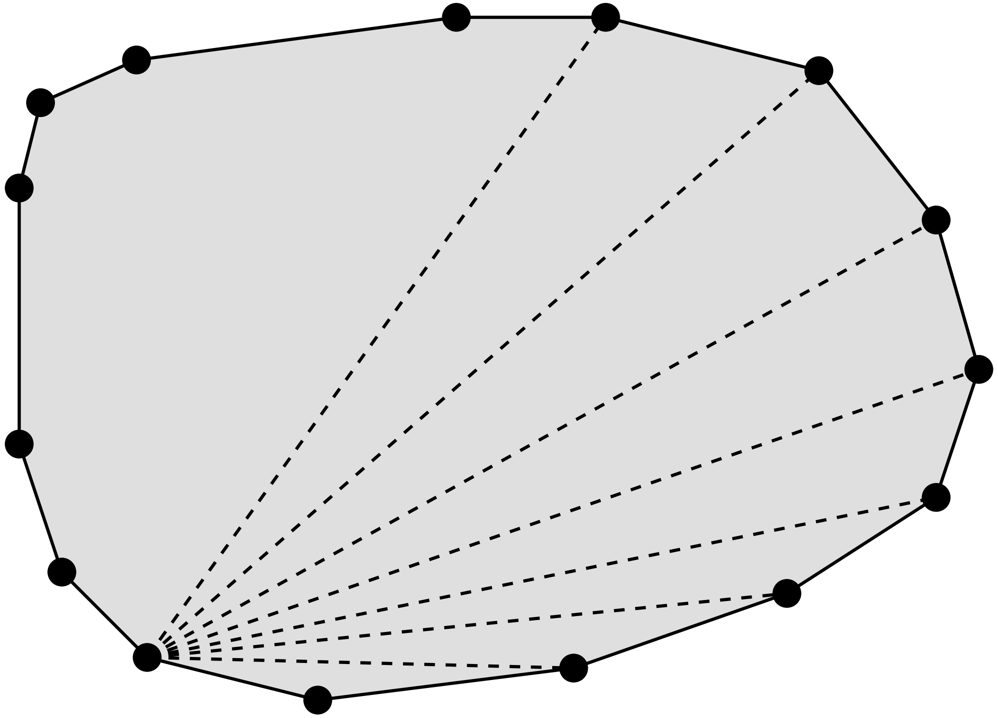


TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.

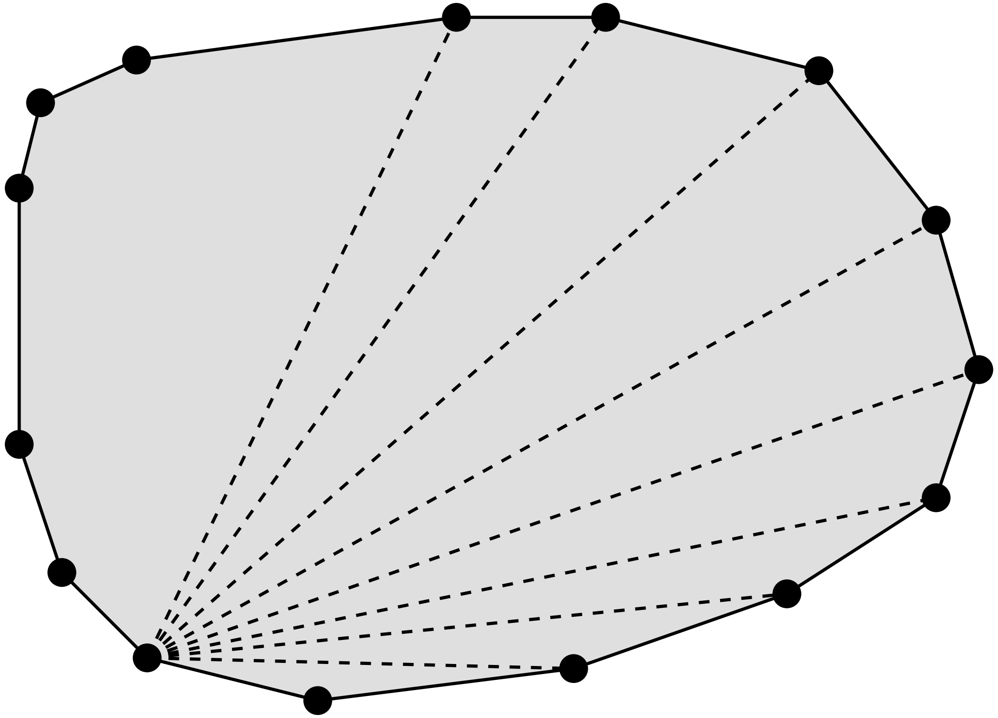


TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.

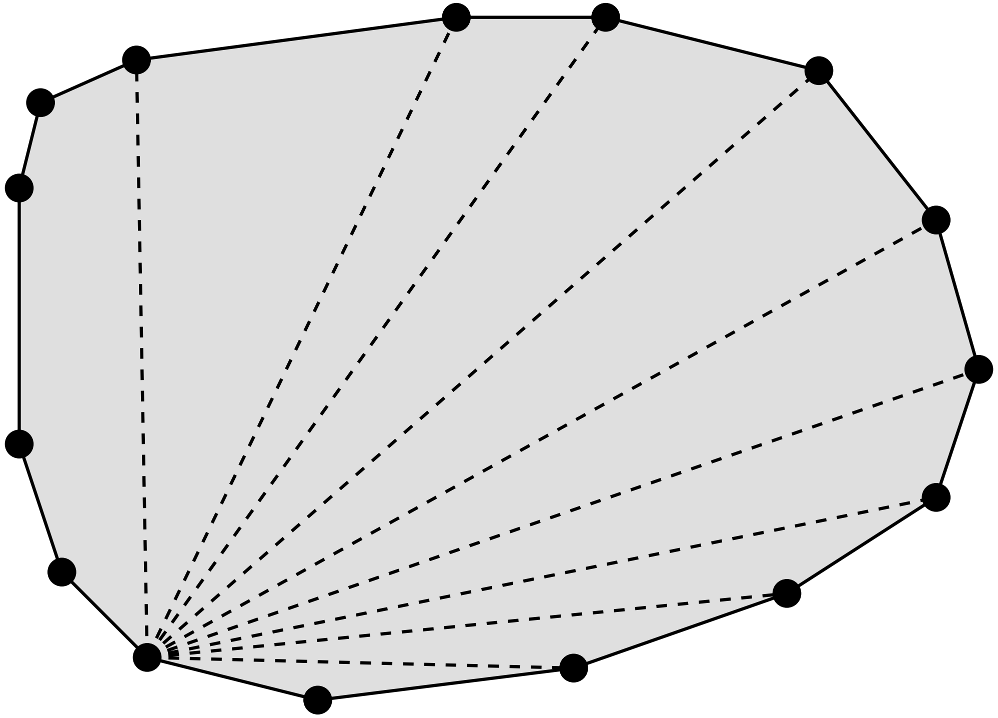


TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.

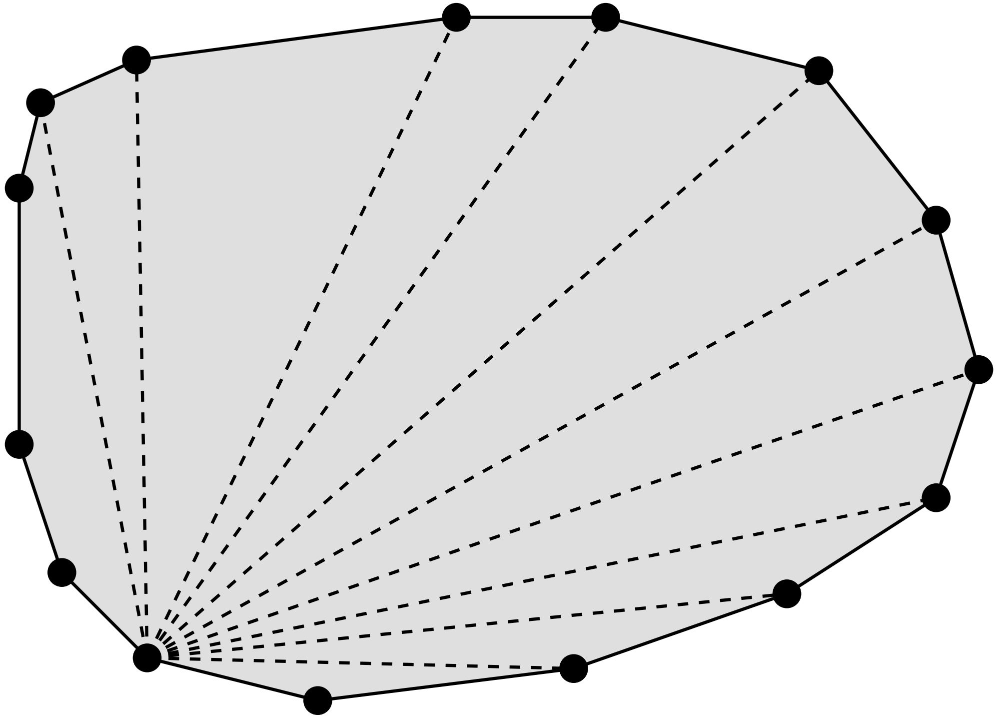


TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.

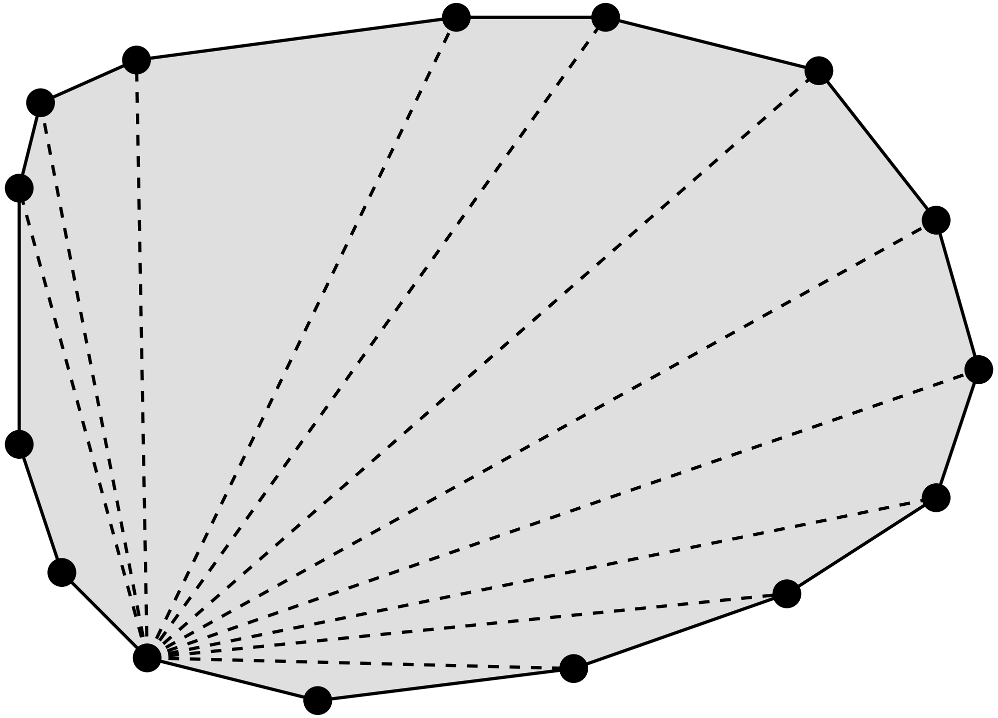


TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.

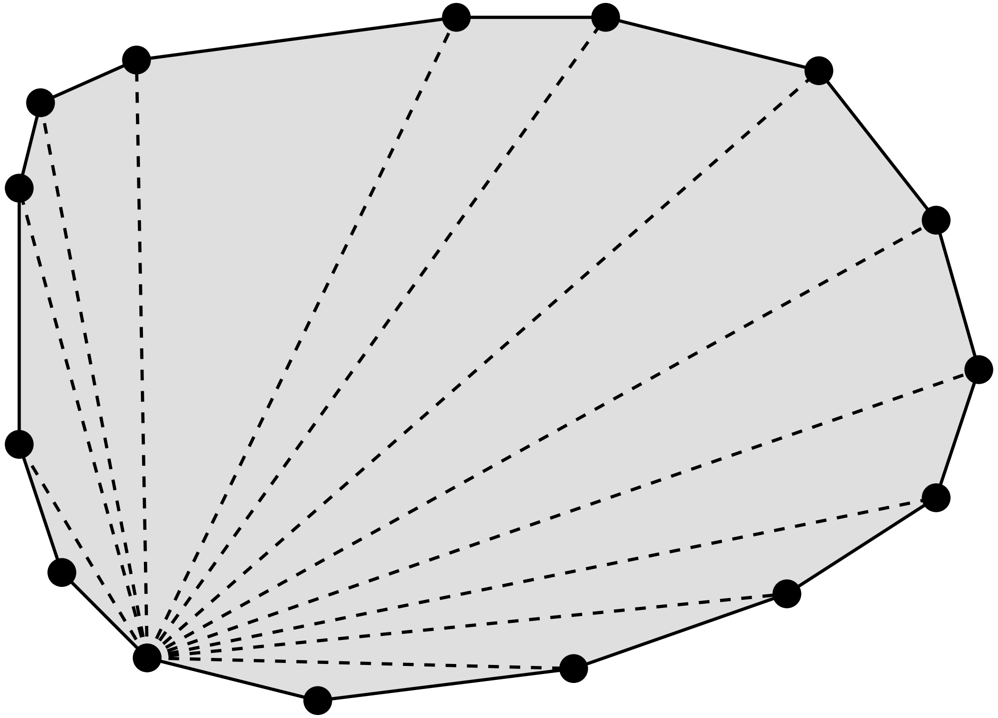


TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.



TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.

Triangulating a star-shaped polygon

Can be done in $O(n)$ time. Posed as problem.

TRIANGULATING POLYGONS

Is it possible to triangulate a polygon more efficiently?

Triangulating a convex polygon

Trivially done in $O(n)$ time.

Triangulating a star-shaped polygon

Can be done in $O(n)$ time. Posed as problem.

Triangulating a monotone polygon

It can also be done in $O(n)$ time. In the following we will see how.

TRIANGULATING POLYGONS

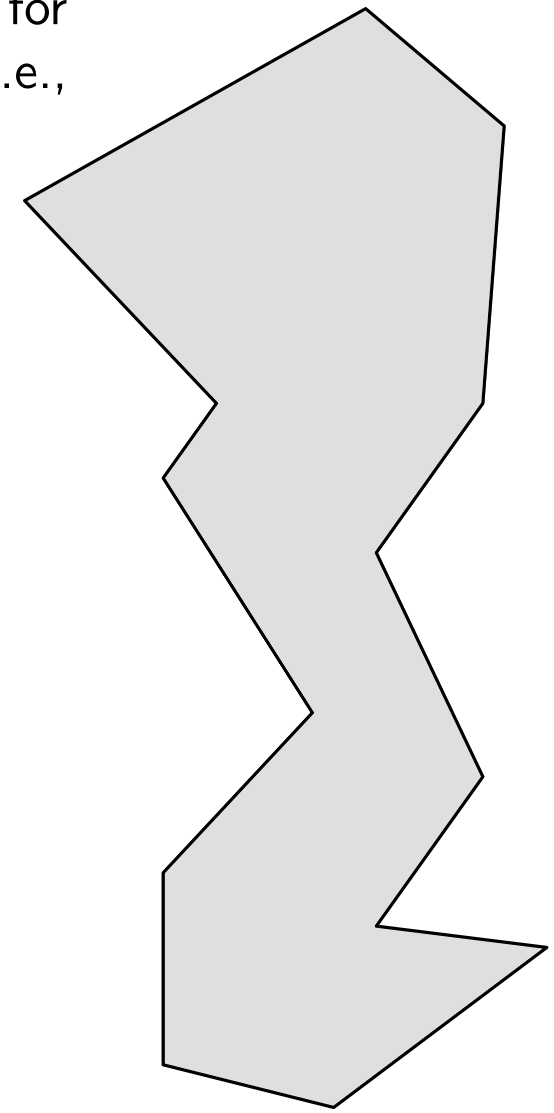
Monotone polygon

A polygon P is called **monotone** with respect to a direction r if, for every line r' orthogonal to r, the intersection $P \cap r'$ is connected (i.e., it is a segment, a point or the empty set).

TRIANGULATING POLYGONS

Monotone polygon

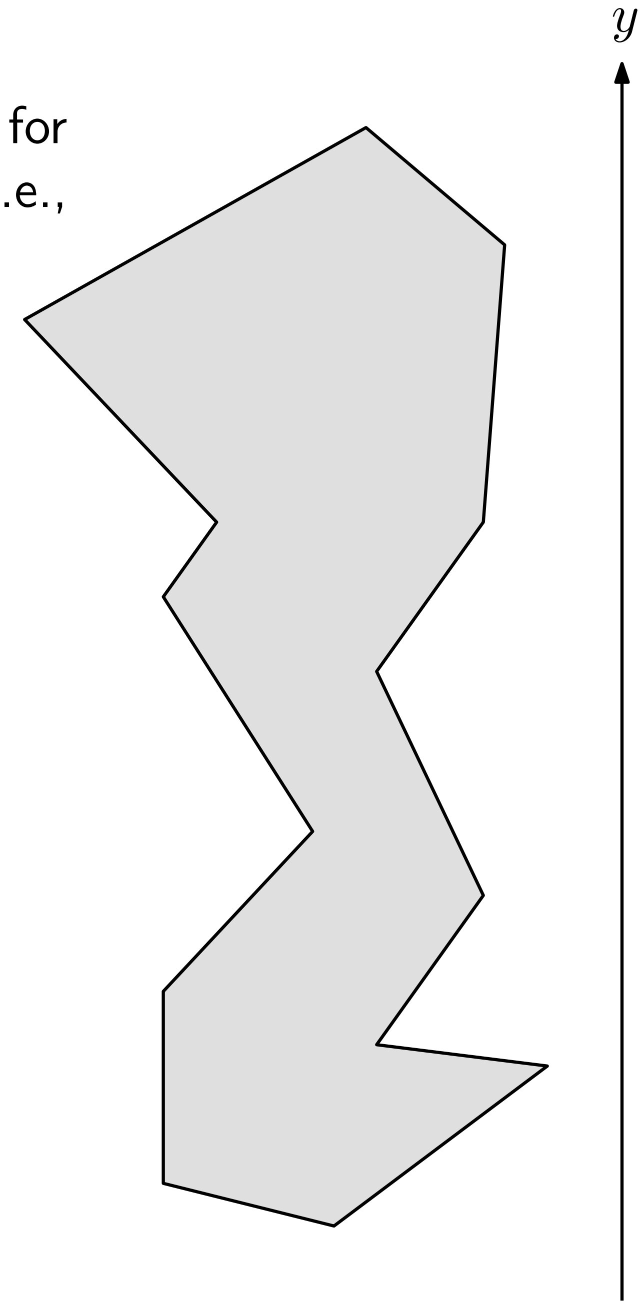
A polygon P is called **monotone** with respect to a direction r if, for every line r' orthogonal to r, the intersection $P \cap r'$ is connected (i.e., it is a segment, a point or the empty set).



TRIANGULATING POLYGONS

Monotone polygon

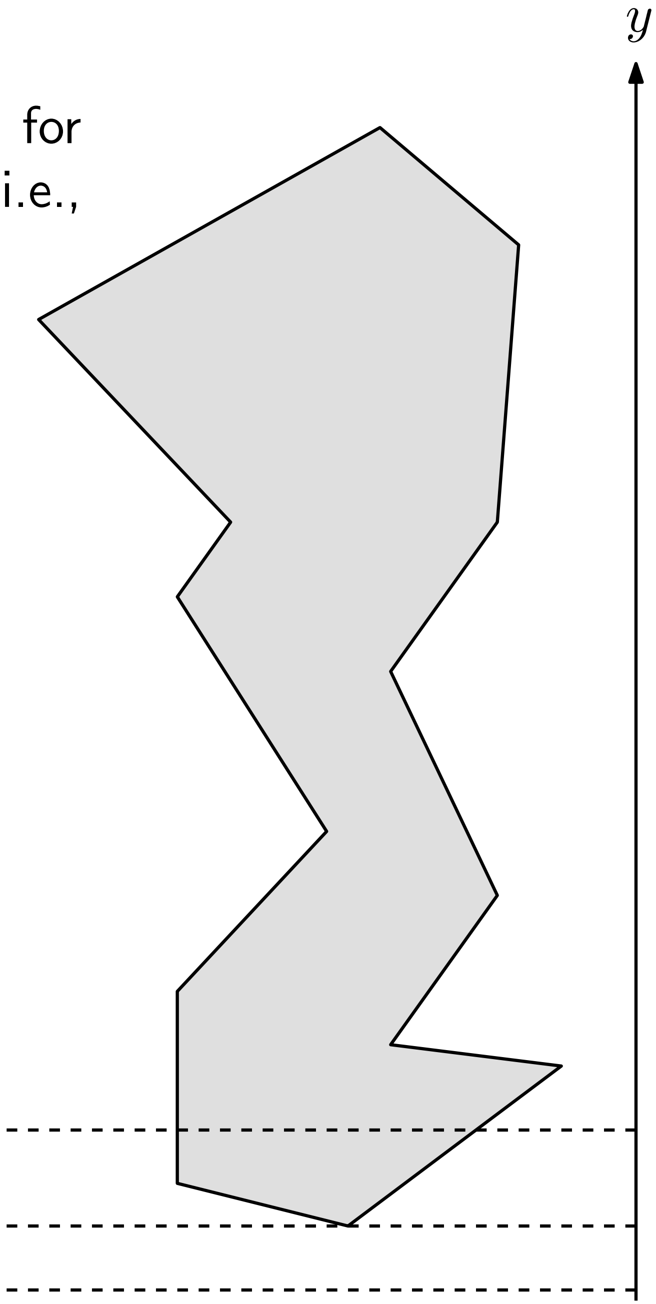
A polygon P is called **monotone** with respect to a direction r if, for every line r' orthogonal to r, the intersection $P \cap r'$ is connected (i.e., it is a segment, a point or the empty set).



TRIANGULATING POLYGONS

Monotone polygon

A polygon P is called **monotone** with respect to a direction r if, for every line r' orthogonal to r, the intersection $P \cap r'$ is connected (i.e., it is a segment, a point or the empty set).



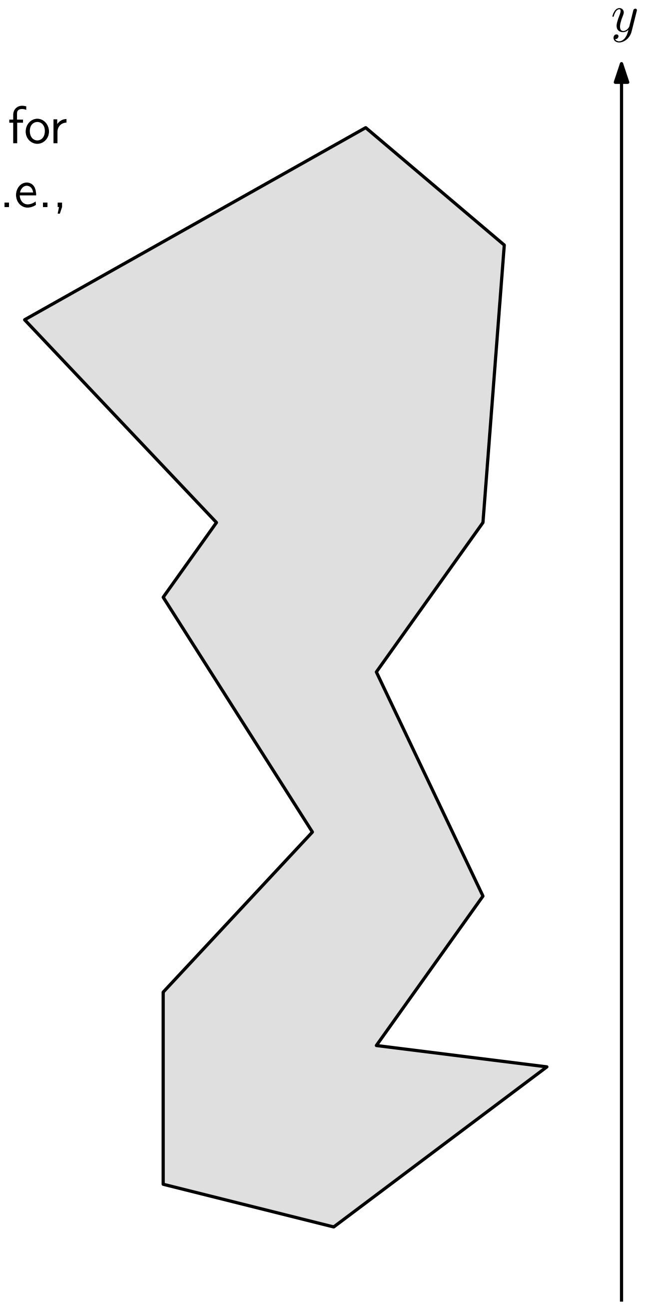
TRIANGULATING POLYGONS

Monotone polygon

A polygon P is called **monotone** with respect to a direction r if, for every line r' orthogonal to r, the intersection $P \cap r'$ is connected (i.e., it is a segment, a point or the empty set).

Local characterization

A polygon is y-monotone if and only if it does not have any cusp.



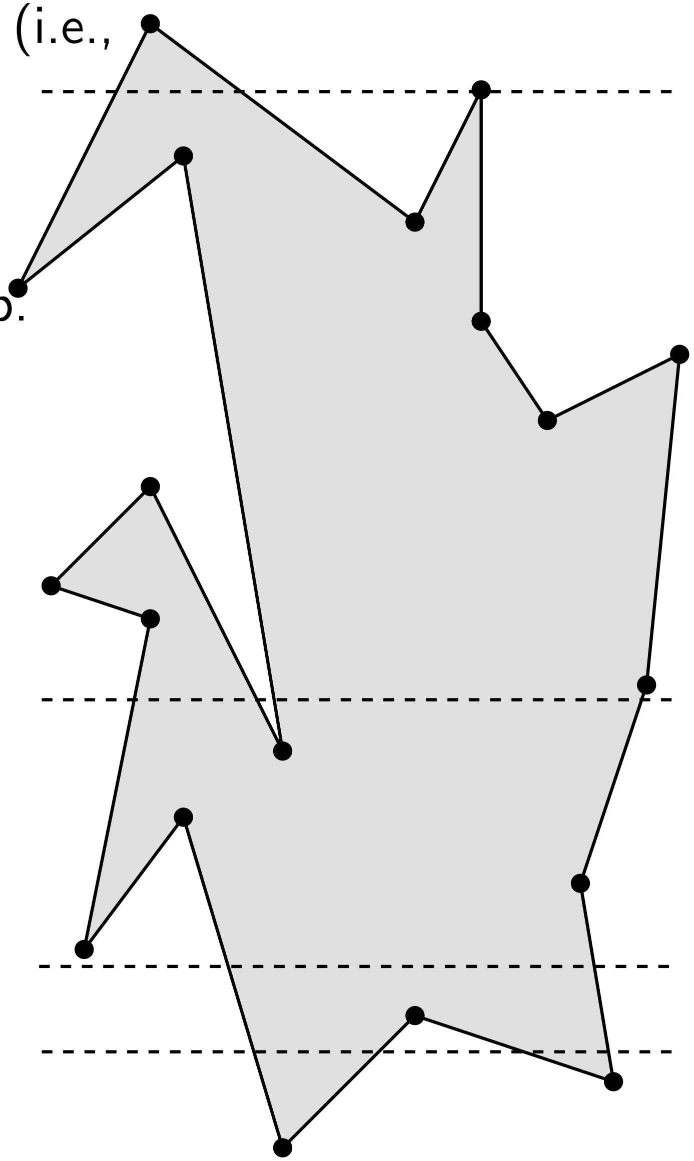
TRIANGULATING POLYGONS

Monotone polygon

A polygon P is called **monotone** with respect to a direction r if, for every line r' orthogonal to r, the intersection $P \cap r'$ is connected (i.e., it is a segment, a point or the empty set).

Local characterization

A polygon is y-monotone if and only if it does not have any cusp.



TRIANGULATING POLYGONS

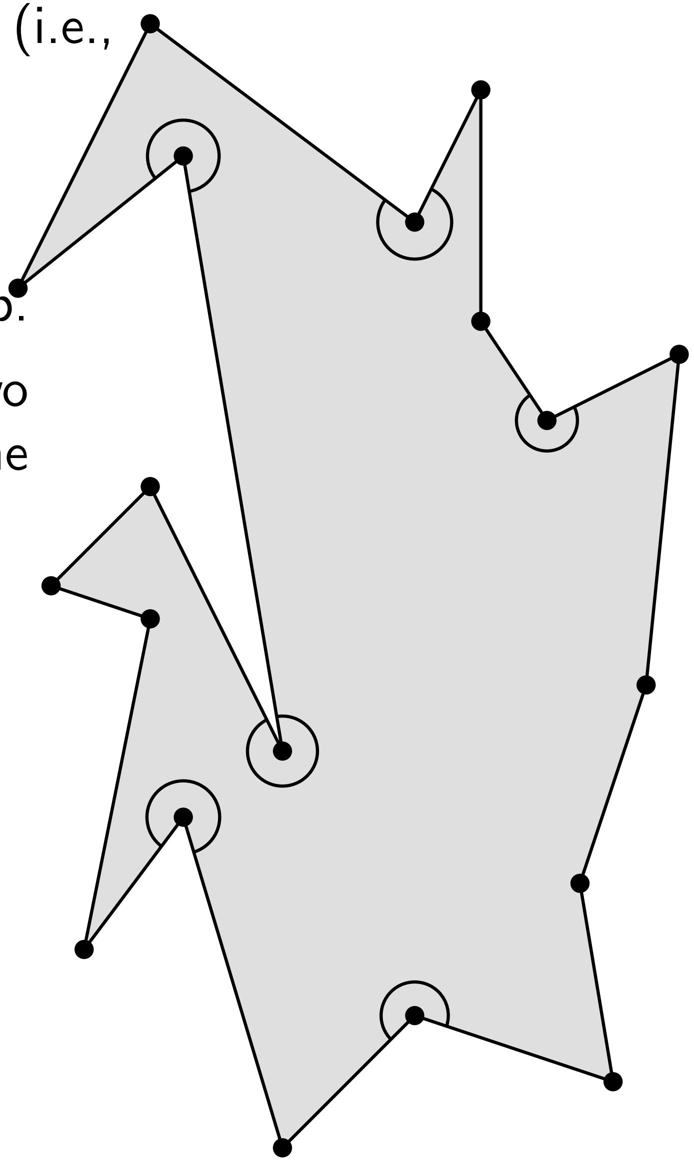
Monotone polygon

A polygon P is called **monotone** with respect to a direction r if, for every line r' orthogonal to r, the intersection $P \cap r'$ is connected (i.e., it is a segment, a point or the empty set).

Local characterization

A polygon is y-monotone if and only if it does not have any cusp.

> A **cusp** is a reflex vertex v of the polygon such that its two incident edges both lie to the same side of the horizontal line through v.



TRIANGULATING POLYGONS

Monotone polygon

A polygon P is called **monotone** with respect to a direction r if, for every line r' orthogonal to r, the intersection $P \cap r'$ is connected (i.e., it is a segment, a point or the empty set).

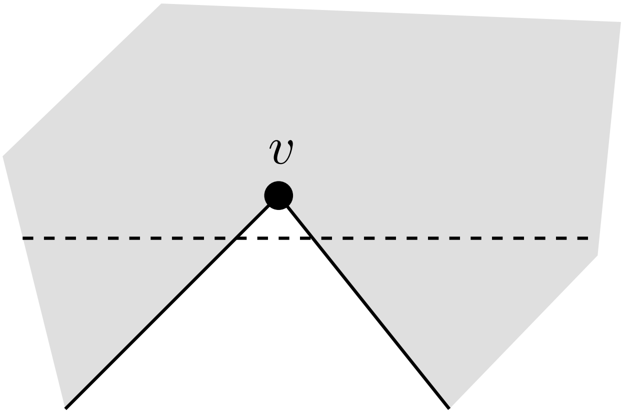
Local characterization

A polygon is y-monotone if and only if it does not have any cusp.

> A **cusp** is a reflex vertex v of the polygon such that its two incident edges both lie to the same side of the horizontal line through v.

Proof:

If the polygon has a local maximum cusp v, an infinitesimal downwards translation of the horizontal line through v would intersect the polygon in at least two connected components.



TRIANGULATING POLYGONS

Monotone polygon

A polygon P is called **monotone** with respect to a direction r if, for every line r' orthogonal to r, the intersection $P \cap r'$ is connected (i.e., it is a segment, a point or the empty set).

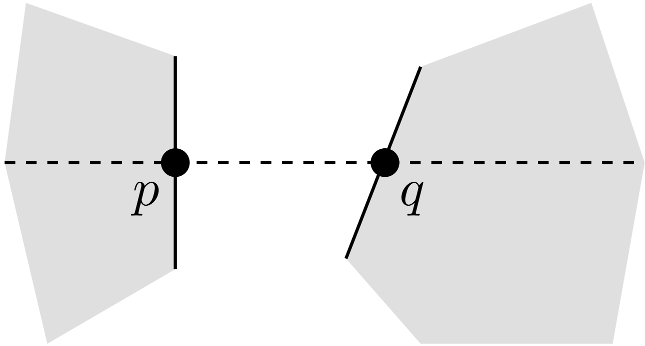
Local characterization

A polygon is y-monotone if and only if it does not have any cusp.

> A **cusp** is a reflex vertex v of the polygon such that its two incident edges both lie to the same side of the horizontal line through v.

Proof:

If the polygon has a local maximum cusp v, an infinitesimal downwards translation of the horizontal line through v would intersect the polygon in at least two connected components.



If the polygon is not y-monotone, let r be a horizontal line intersecting the polygon in two or more connected components. Consider two consecutive components, with facing endpoints p and q as in the figure. The polygon boundary needs to connect p and q. No matter whether it goes above or below the horizontal line, it will have a cusp.

TRIANGULATING POLYGONS

Monotone polygon

A polygon P is called **monotone** with respect to a direction r if, for every line r' orthogonal to r, the intersection $P \cap r'$ is connected (i.e., it is a segment, a point or the empty set).

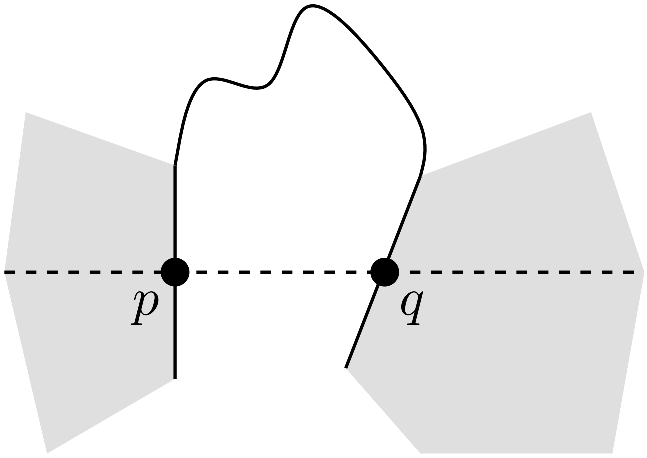
Local characterization

A polygon is y-monotone if and only if it does not have any cusp.

> A **cusp** is a reflex vertex v of the polygon such that its two incident edges both lie to the same side of the horizontal line through v.

Proof:

If the polygon has a local maximum cusp v, an infinitesimal downwards translation of the horizontal line through v would intersect the polygon in at least two connected components.



If the polygon is not y-monotone, let r be a horizontal line intersecting the polygon in two or more connected components. Consider two consecutive components, with facing endpoints p and q as in the figure. The polygon boundary needs to connect p and q. No matter whether it goes above or below the horizontal line, it will have a cusp.

TRIANGULATING POLYGONS

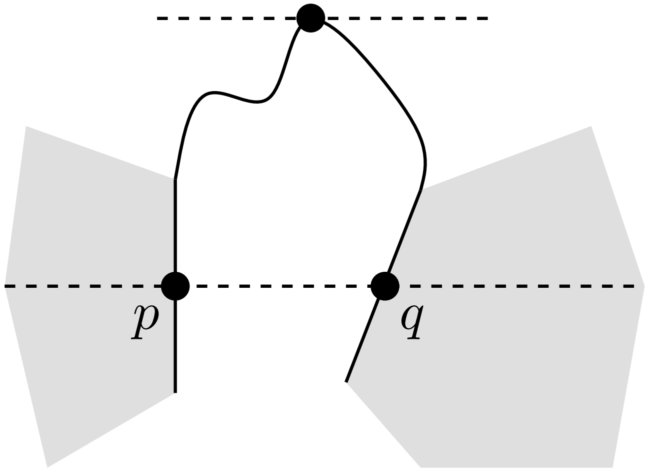
Monotone polygon

A polygon P is called **monotone** with respect to a direction r if, for every line r' orthogonal to r, the intersection $P \cap r'$ is connected (i.e., it is a segment, a point or the empty set).

Local characterization

A polygon is y-monotone if and only if it does not have any cusp.

> A **cusp** is a reflex vertex v of the polygon such that its two incident edges both lie to the same side of the horizontal line through v.



Proof:

If the polygon has a local maximum cusp v, an infinitesimal downwards translation of the horizontal line through v would intersect the polygon in at least two connected components.

If the polygon is not y-monotone, let r be a horizontal line intersecting the polygon in two or more connected components. Consider two consecutive components, with facing endpoints p and q as in the figure. The polygon boundary needs to connect p and q. No matter whether it goes above or below the horizontal line, it will have a cusp.

TRIANGULATING POLYGONS

Monotone polygon

A polygon P is called **monotone** with respect to a direction r if, for every line r' orthogonal to r, the intersection $P \cap r'$ is connected (i.e., it is a segment, a point or the empty set).

Local characterization

A polygon is y-monotone if and only if it does not have any cusp.

> A **cusp** is a reflex vertex v of the polygon such that its two incident edges both lie to the same side of the horizontal line through v.

TRIANGULATING POLYGONS

Monotone polygon

A polygon P is called **monotone** with respect to a direction r if, for every line r' orthogonal to r, the intersection $P \cap r'$ is connected (i.e., it is a segment, a point or the empty set).

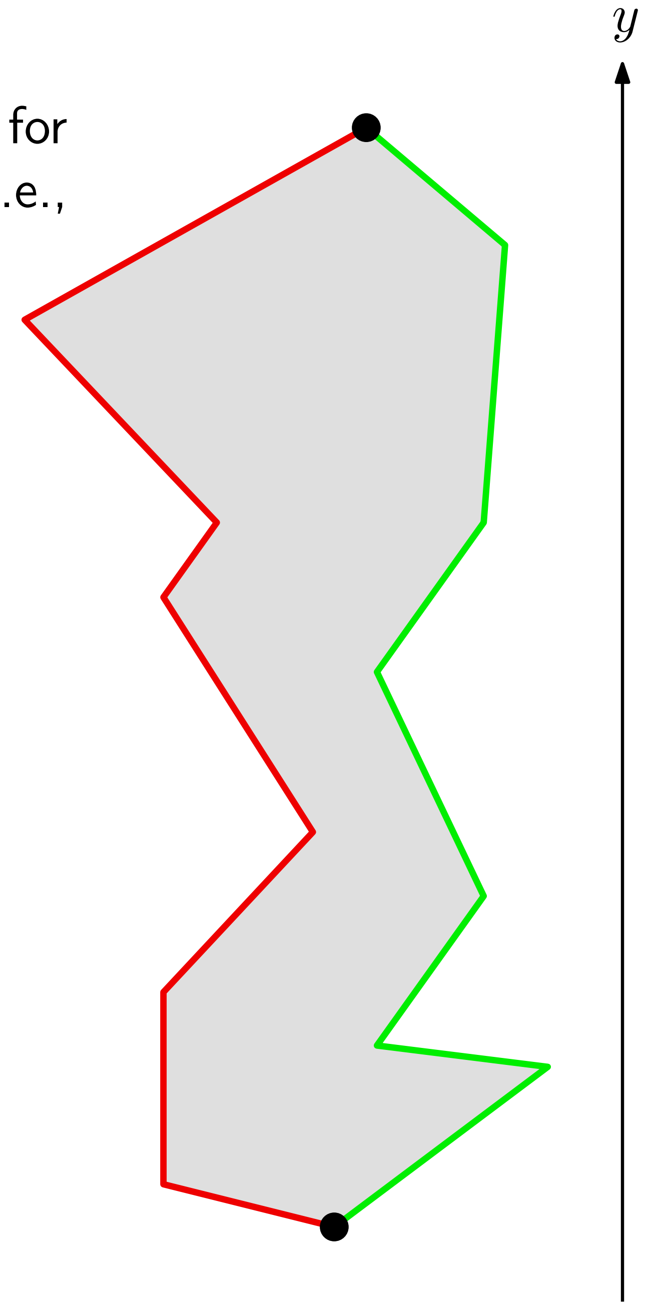
Local characterization

A polygon is y-monotone if and only if it does not have any cusp.

> A **cusp** is a reflex vertex v of the polygon such that its two incident edges both lie to the same side of the horizontal line through v.

Corollary

If a polygon is y-monotone, then it can be decomposed into two y-monotone non intersecting chains sharing their endpoints.



TRIANGULATING POLYGONS

Triangulating a monotone polygon

TRIANGULATING POLYGONS

Triangulating a monotone polygon

The vertices of the polygon P are processed by decreasing order of their y-coordinate.

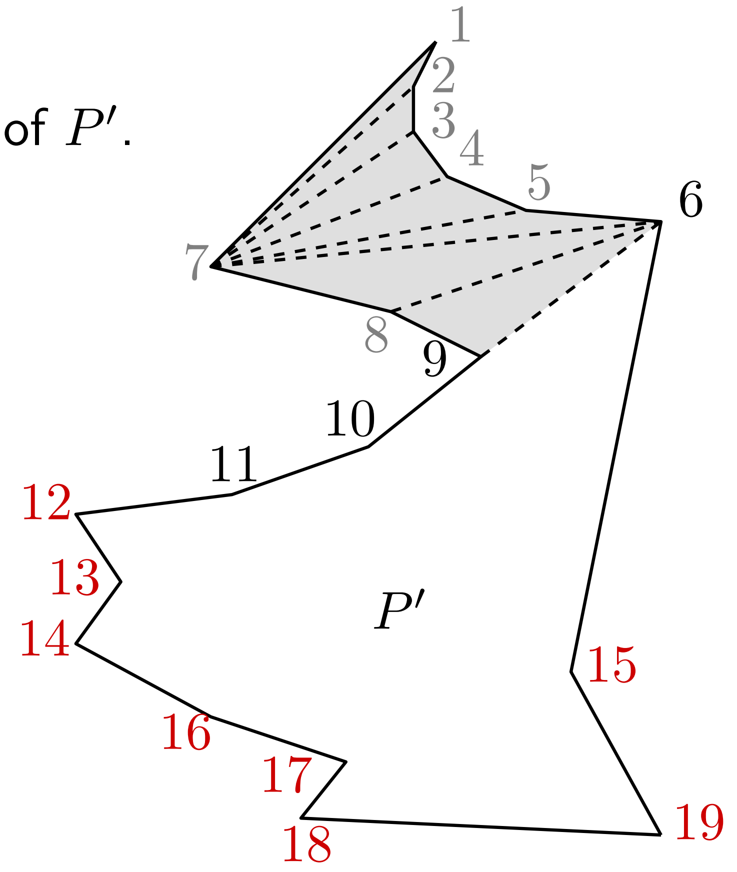
TRIANGULATING POLYGONS

Triangulating a monotone polygon

The vertices of the polygon P are processed by decreasing order of their y-coordinate.

During the process a queue Q is used to store the vertices that have already been visited but are still needed in order to generate the triangulation. Characteristics of Q:

- The topmost (i.e., largest y-coordinate) vertex in Q, is a convex vertex of the subpolygon P' still to be triangulated.
- All the remaining vertices in Q are reflex.
- All the vertices in Q belong to the same monotone chain of P'.



TRIANGULATING POLYGONS

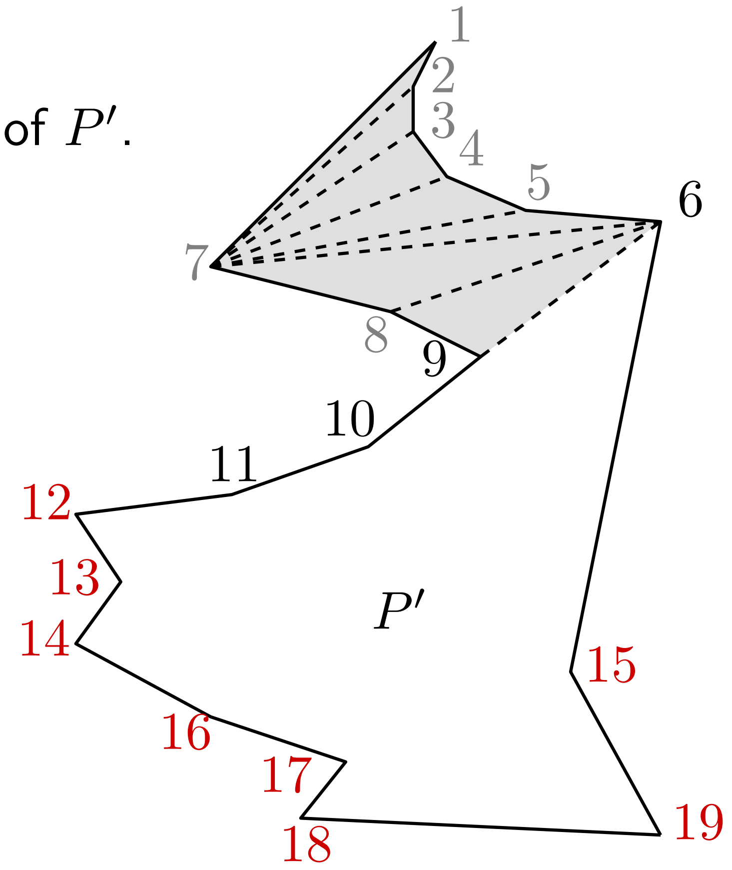
Triangulating a monotone polygon

The vertices of the polygon P are processed by decreasing order of their y-coordinate.

During the process a queue Q is used to store the vertices that have already been visited but are still needed in order to generate the triangulation. Characteristics of Q:

- The topmost (i.e., largest y-coordinate) vertex in Q, is a convex vertex of the subpolygon P' still to be triangulated.
- All the remaining vertices in Q are reflex.
- All the vertices in Q belong to the same monotone chain of P'.

Processing a vertex v_i:



TRIANGULATING POLYGONS

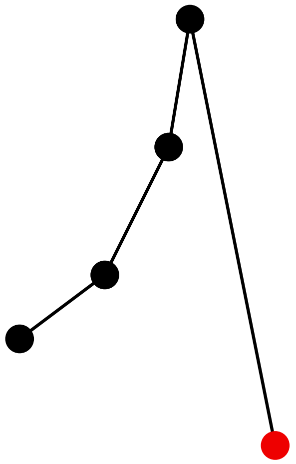
Triangulating a monotone polygon

The vertices of the polygon P are processed by decreasing order of their y-coordinate.

During the process a queue Q is used to store the vertices that have already been visited but are still needed in order to generate the triangulation. Characteristics of Q:
 - The topmost (i.e., largest y-coordinate) vertex in Q, is a convex vertex of the subpolygon P' still to be triangulated.
 - All the remaining vertices in Q are reflex.
 - All the vertices in Q belong to the same monotone chain of P'.

Processing a vertex v_i:

- If v_i belongs to the opposite chain, report the diagonals connecting v_i to every vertex of Q and delete them all from Q, except the last one. Add v_i to Q.



TRIANGULATING POLYGONS

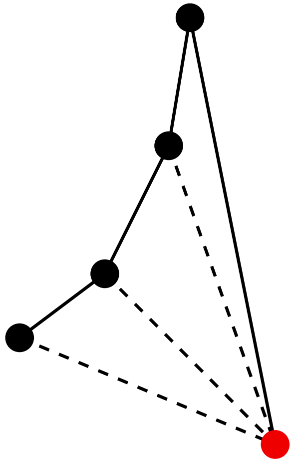
Triangulating a monotone polygon

The vertices of the polygon P are processed by decreasing order of their y-coordinate.

During the process a queue Q is used to store the vertices that have already been visited but are still needed in order to generate the triangulation. Characteristics of Q:
- The topmost (i.e., largest y-coordinate) vertex in Q, is a convex vertex of the subpolygon P' still to be triangulated.
- All the remaining vertices in Q are reflex.
- All the vertices in Q belong to the same monotone chain of P'.

Processing a vertex v_i:

- If v_i belongs to the opposite chain, report the diagonals connecting v_i to every vertex of Q and delete them all from Q, except the last one. Add v_i to Q.



TRIANGULATING POLYGONS

Triangulating a monotone polygon

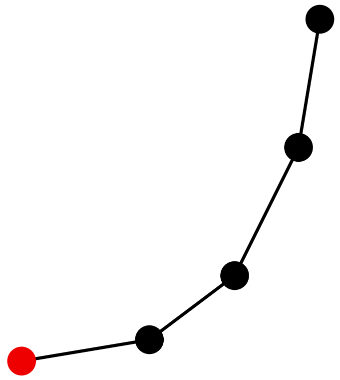
The vertices of the polygon P are processed by decreasing order of their y-coordinate.

During the process a queue Q is used to store the vertices that have already been visited but are still needed in order to generate the triangulation. Characteristics of Q:

- The topmost (i.e., largest y-coordinate) vertex in Q, is a convex vertex of the subpolygon P' still to be triangulated.
- All the remaining vertices in Q are reflex.
- All the vertices in Q belong to the same monotone chain of P'.

Processing a vertex v_i:

- If v_i belongs to the opposite chain, report the diagonals connecting v_i to every vertex of Q and delete them all from Q, except the last one. Add v_i to Q.
- If v_i belongs to the same chain and produces a reflex turn, add v_i to Q.



TRIANGULATING POLYGONS

Triangulating a monotone polygon

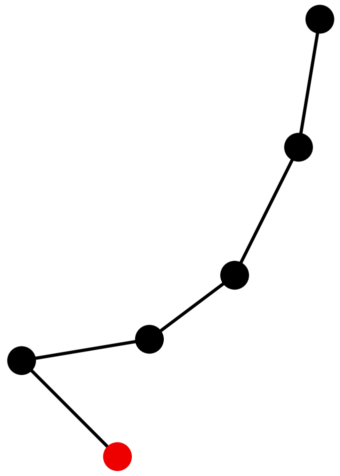
The vertices of the polygon P are processed by decreasing order of their y-coordinate.

During the process a queue Q is used to store the vertices that have already been visited but are still needed in order to generate the triangulation. Characteristics of Q:

- The topmost (i.e., largest y-coordinate) vertex in Q, is a convex vertex of the subpolygon P' still to be triangulated.
- All the remaining vertices in Q are reflex.
- All the vertices in Q belong to the same monotone chain of P'.

Processing a vertex v_i:

- If v_i belongs to the opposite chain, report the diagonals connecting v_i to every vertex of Q and delete them all from Q, except the last one. Add v_i to Q.
- If v_i belongs to the same chain and produces a reflex turn, add v_i to Q.
- If v_i belongs to the same chain and produces a convex turn, report the diagonal connecting v_i to the penultimate element of Q, delete the last element of Q and process v_i again.



TRIANGULATING POLYGONS

Triangulating a monotone polygon

The vertices of the polygon P are processed by decreasing order of their y-coordinate.

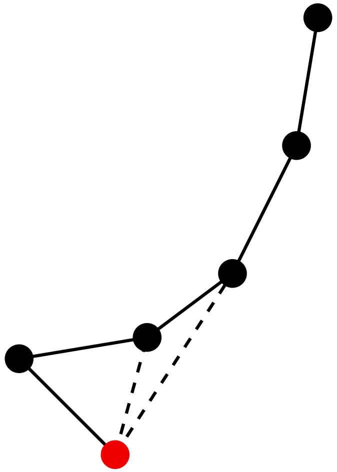
During the process a queue Q is used to store the vertices that have already been visited but are still needed in order to generate the triangulation. Characteristics of Q:

- The topmost (i.e., largest y-coordinate) vertex in Q, is a convex vertex of the subpolygon P' still to be triangulated.

- All the remaining vertices in Q are reflex.

- All the vertices in Q belong to the same monotone chain of P'.

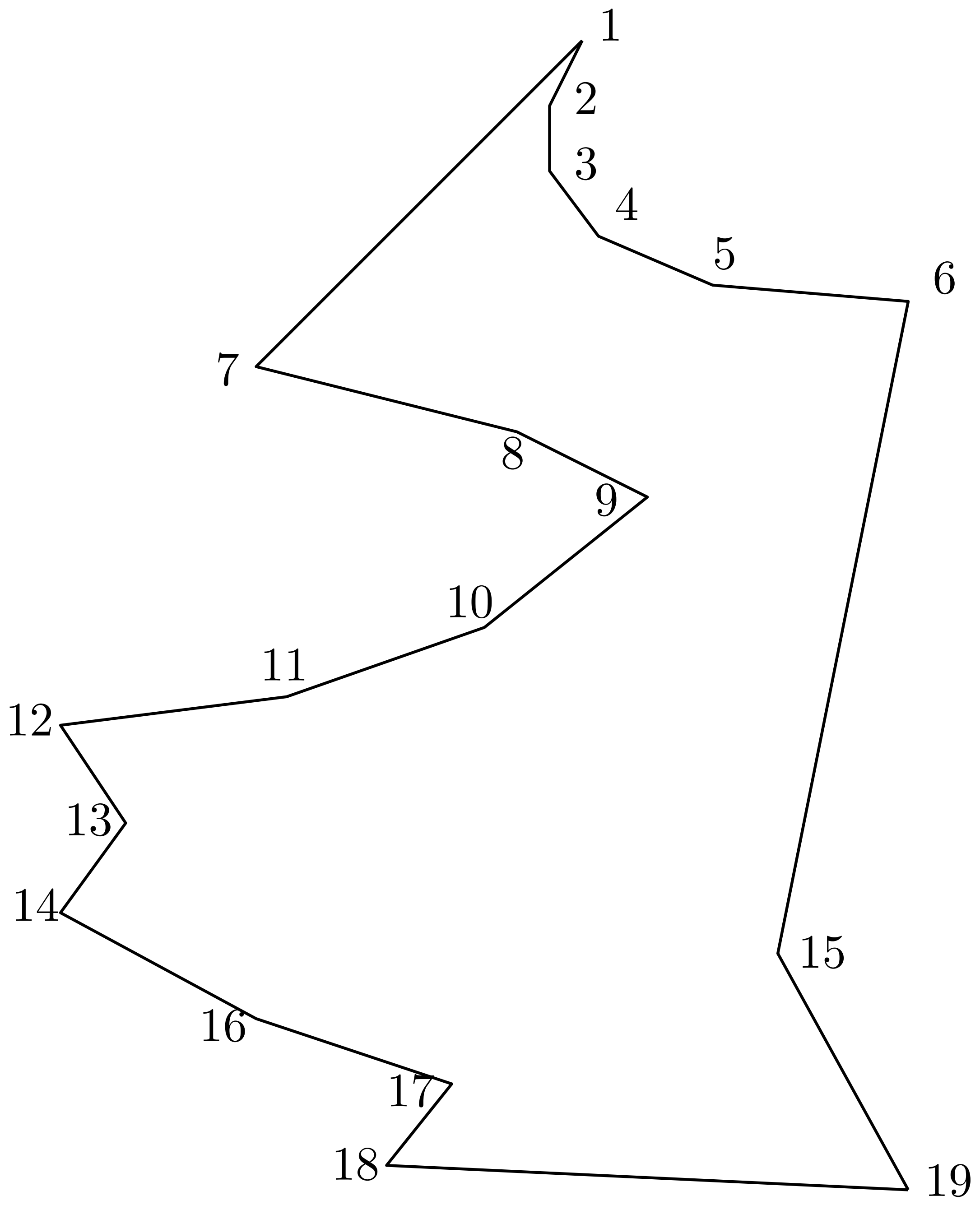
Processing a vertex v_i:



- If v_i belongs to the opposite chain, report the diagonals connecting v_i to every vertex of Q and delete them all from Q, except the last one. Add v_i to Q.
- If v_i belongs to the same chain and produces a reflex turn, add v_i to Q.
- If v_i belongs to the same chain and produces a convex turn, report the diagonal connecting v_i to the penultimate element of Q, delete the last element of Q and process v_i again.

TRIANGULATING POLYGONS

Triangulating a monotone polygon



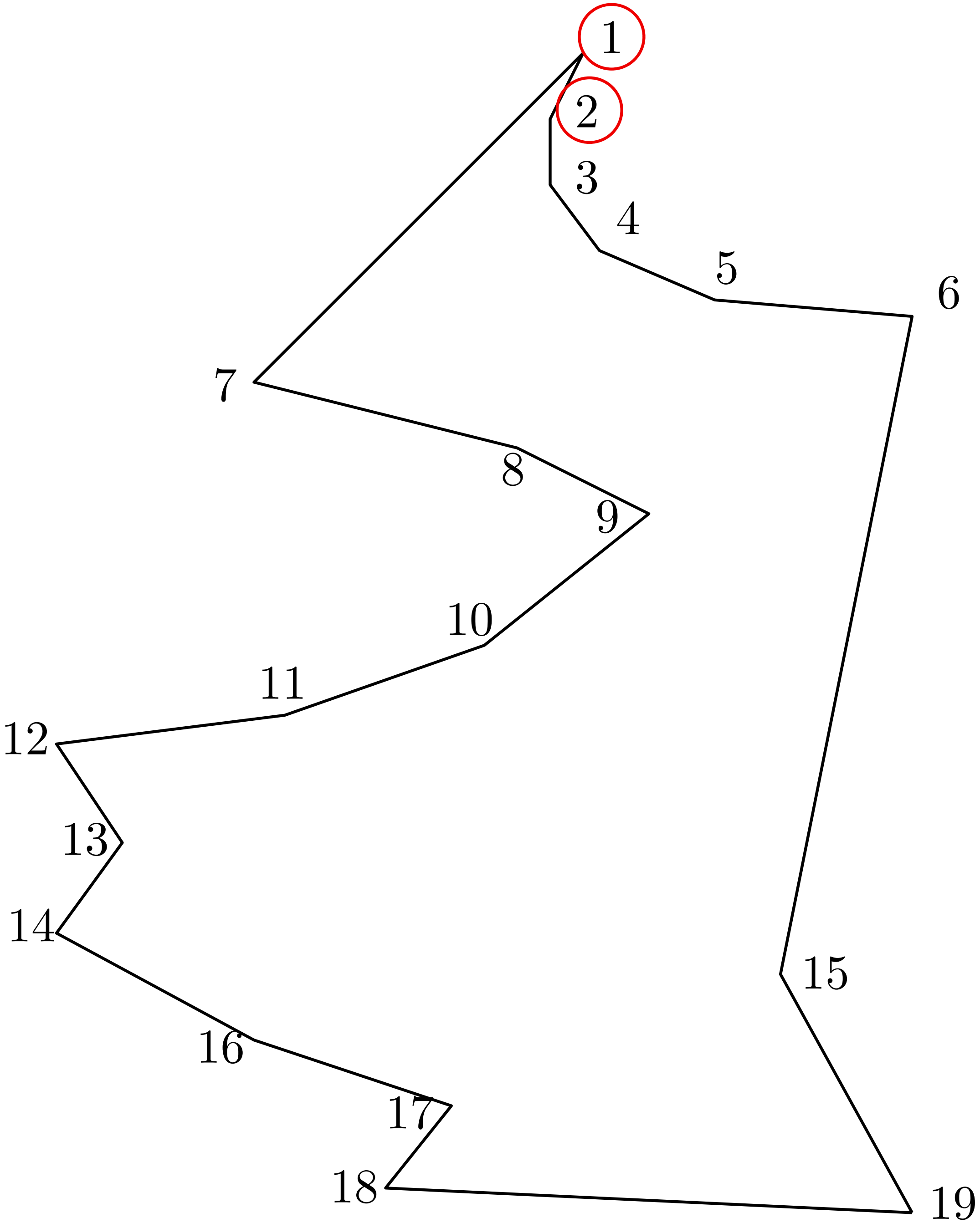
TRIANGULATING POLYGONS

Triangulating a monotone polygon

Start

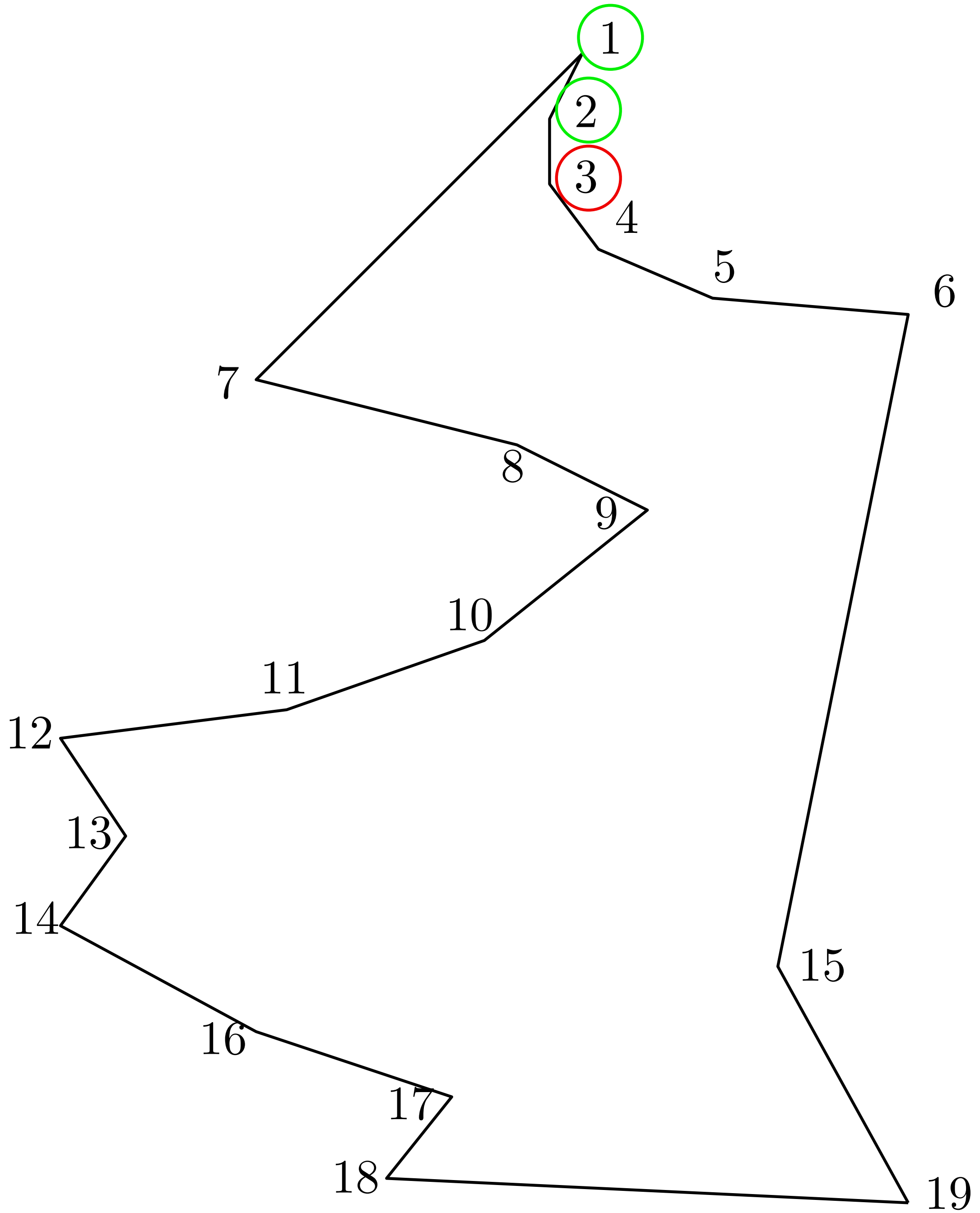
Queue state

1, 2



TRIANGULATING POLYGONS

Triangulating a monotone polygon



Current vertex: 3

Add

Queue state:

1, 2, 3

TRIANGULATING POLYGONS

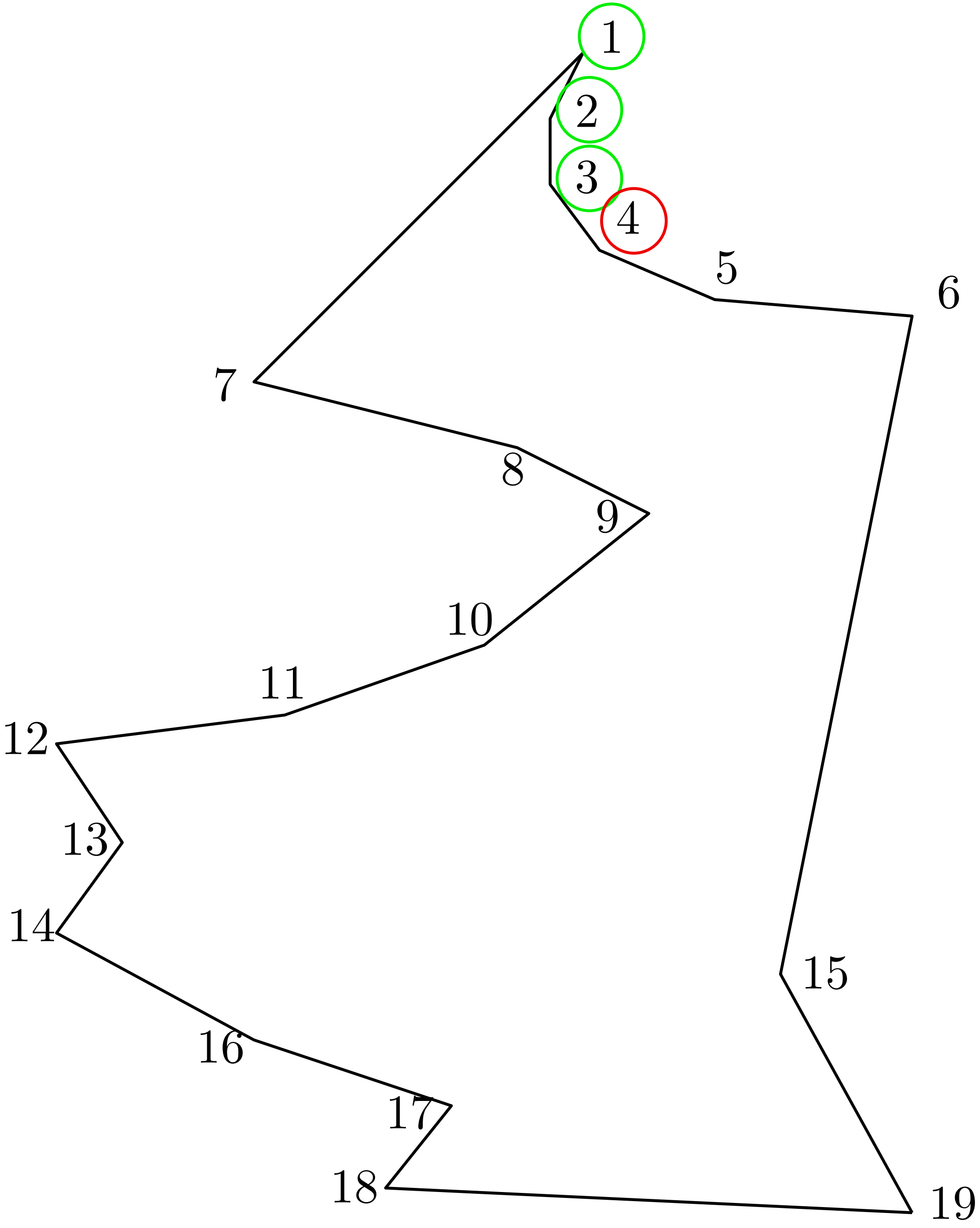
Triangulating a monotone polygon

Current vertex: 4

Add

Queue state:

1, 2, 3, 4



TRIANGULATING POLYGONS

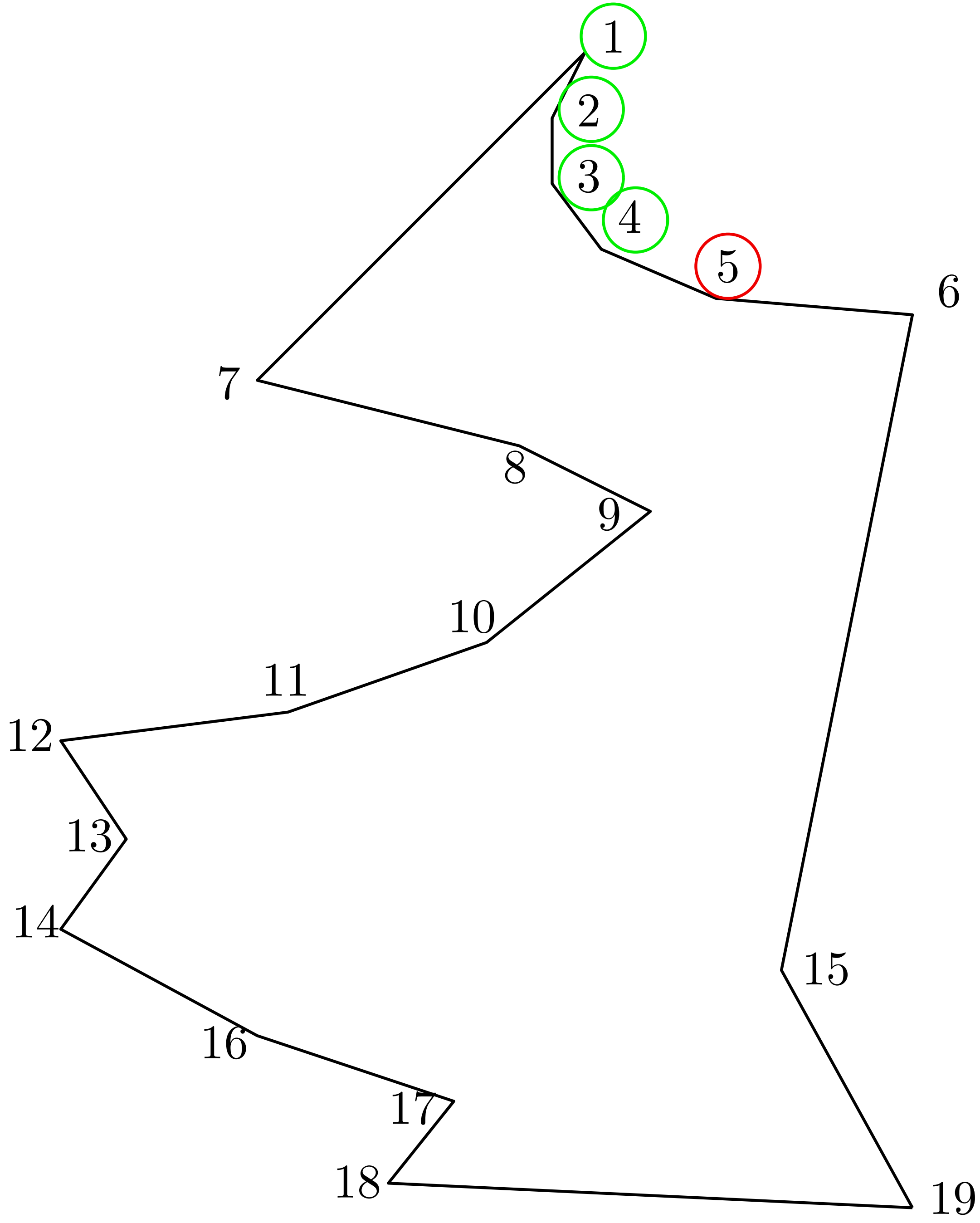
Triangulating a monotone polygon

Current vertex: 5

Add

Queue state:

1, 2, 3, 4, 5



TRIANGULATING POLYGONS

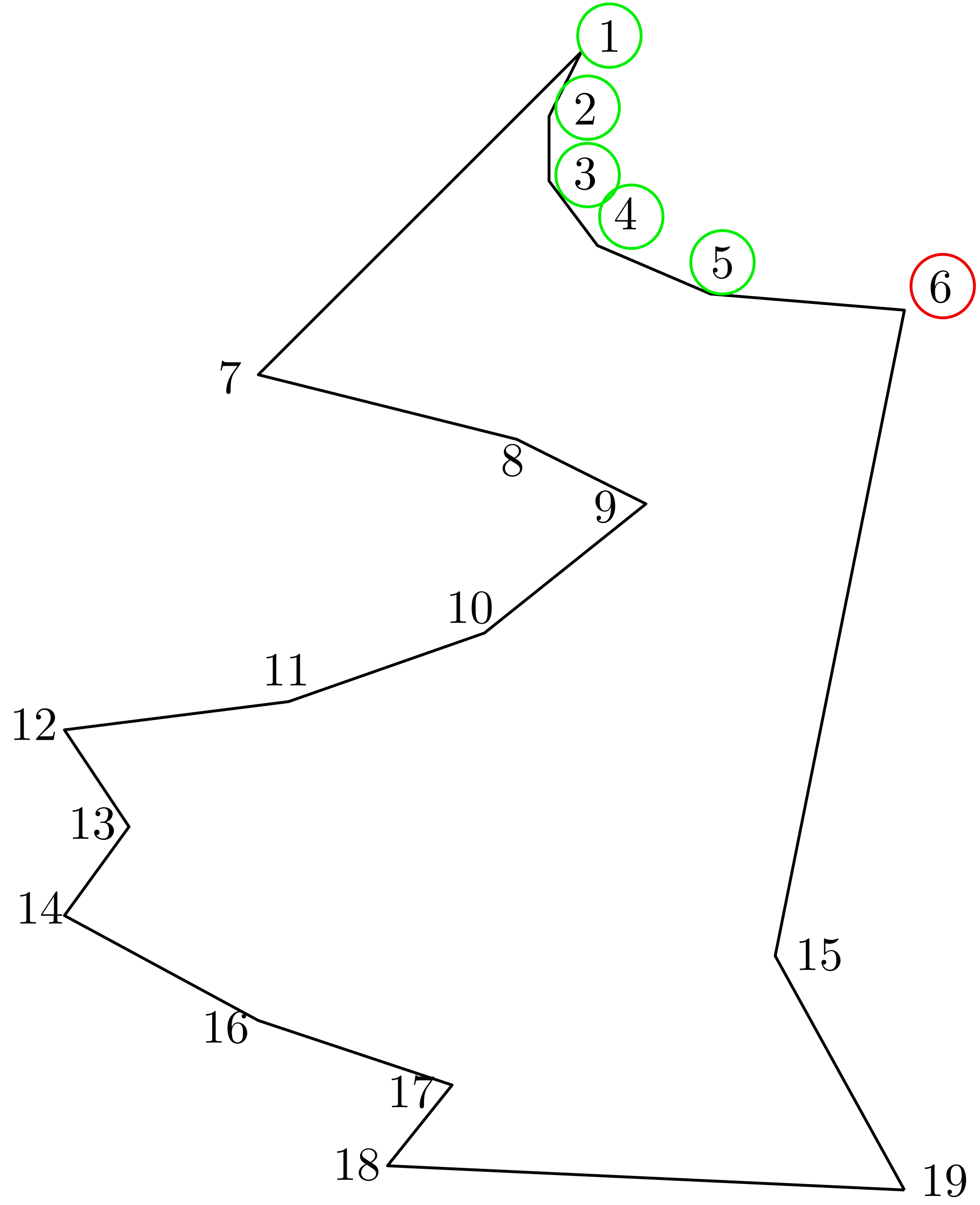
Triangulating a monotone polygon

Current vertex: 6

Add

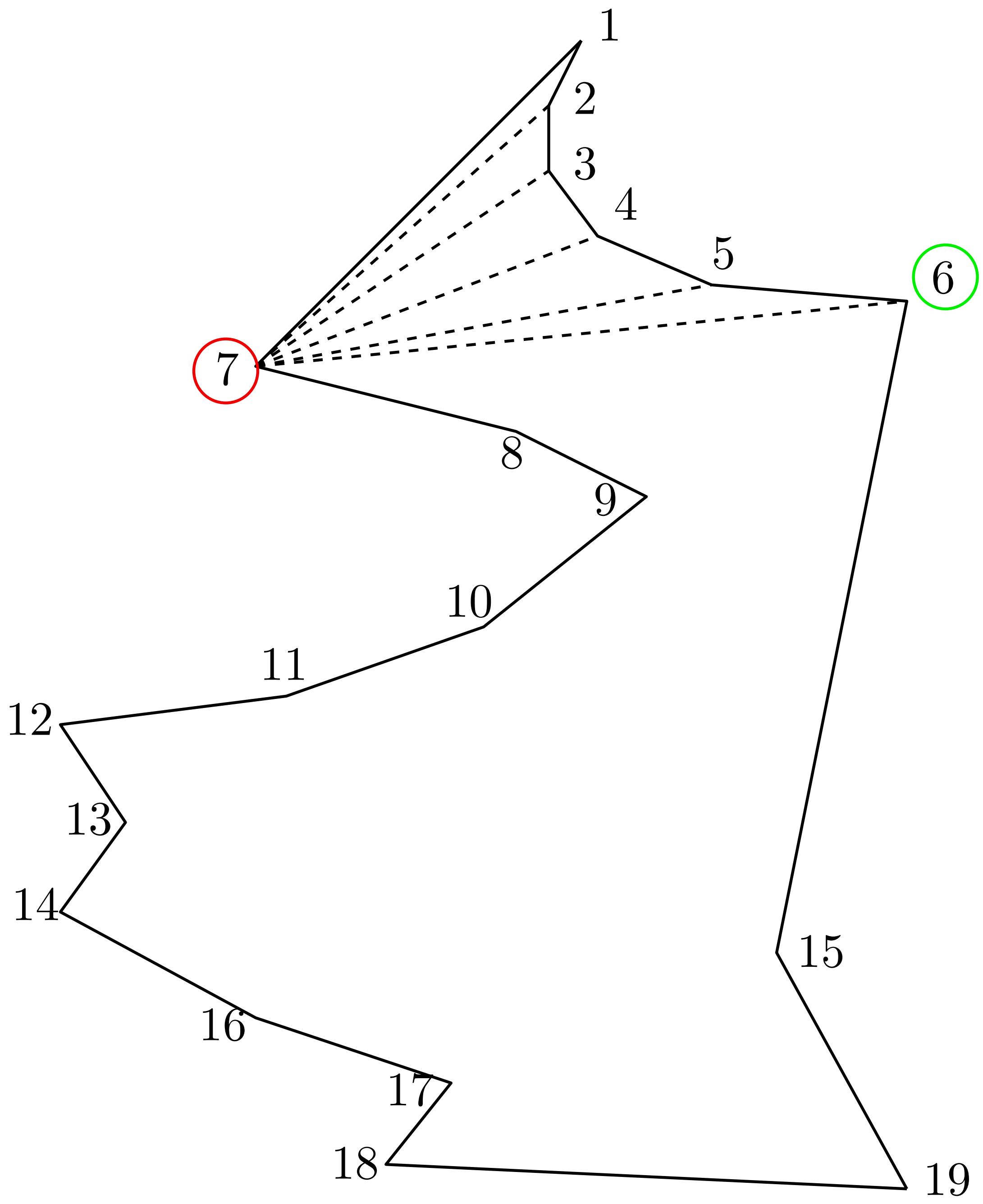
Queue state:

1, 2, 3, 4, 5, 6



TRIANGULATING POLYGONS

Triangulating a monotone polygon



Current vertex: 7

Opposite chain

Queue state:

6, 7

TRIANGULATING POLYGONS

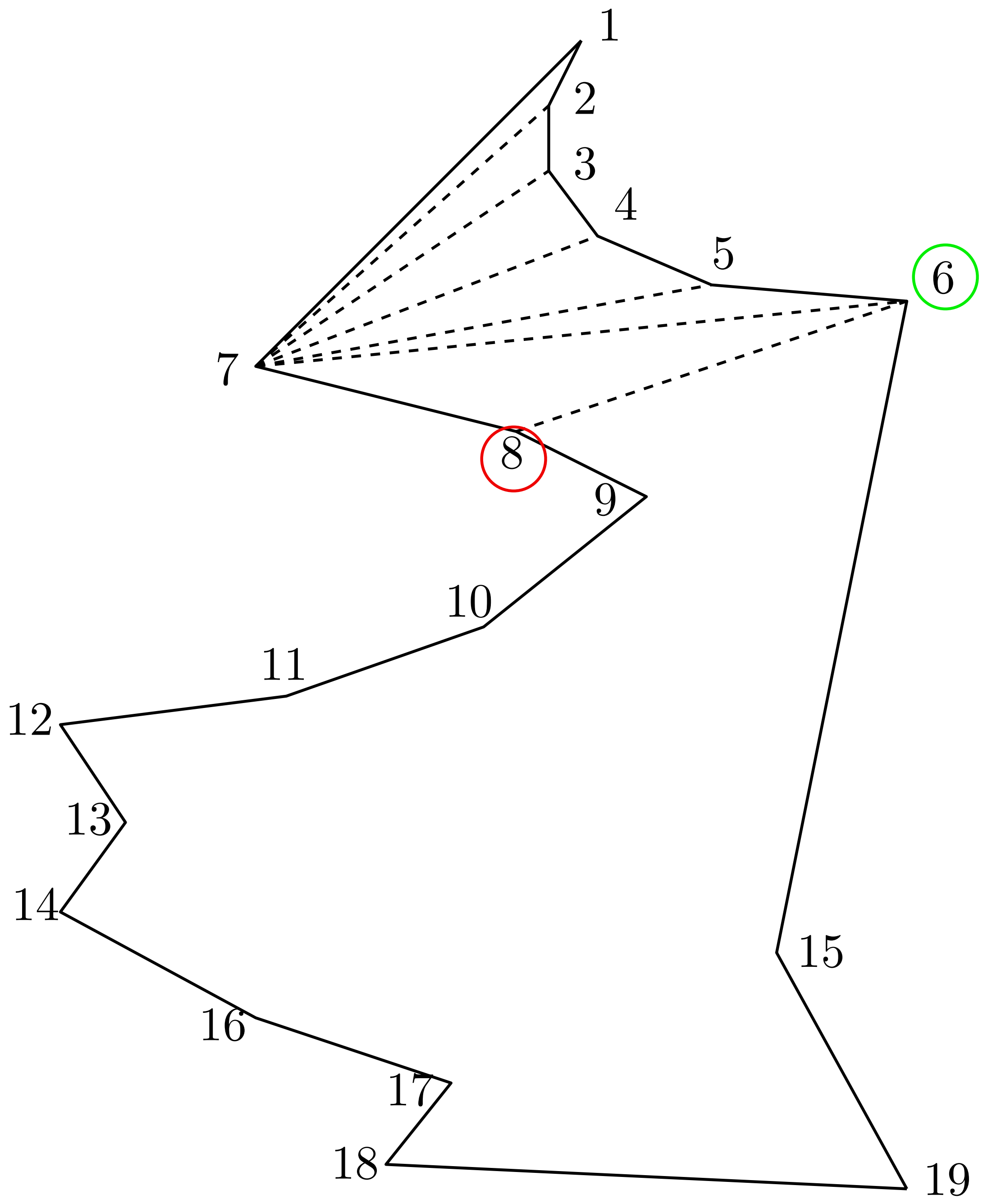
Triangulating a monotone polygon

Current vertex: 8

Ear

Queue state:

6, 8



TRIANGULATING POLYGONS

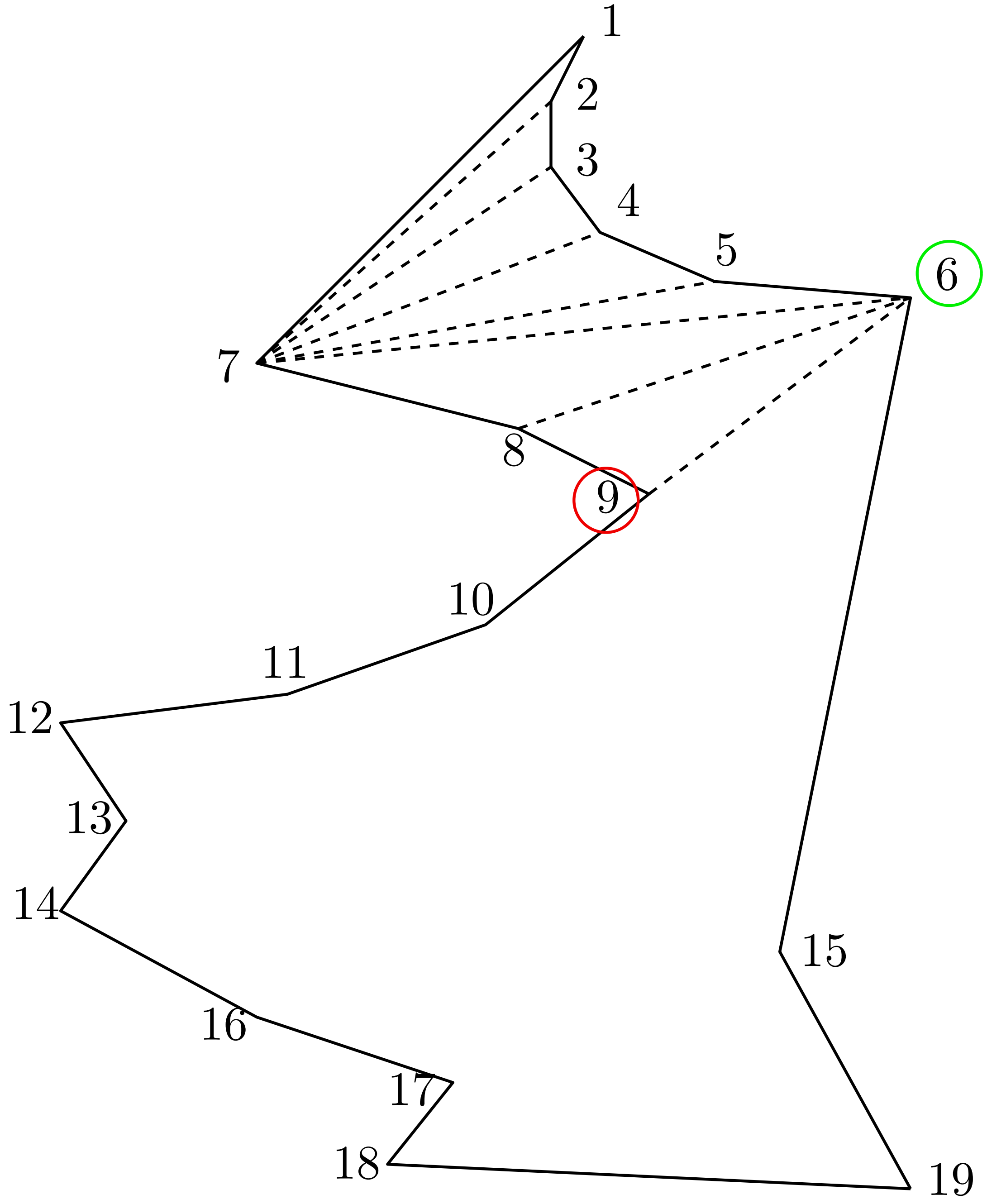
Triangulating a monotone polygon

Current vertex: 9

Ear

Queue state:

6, 9



TRIANGULATING POLYGONS

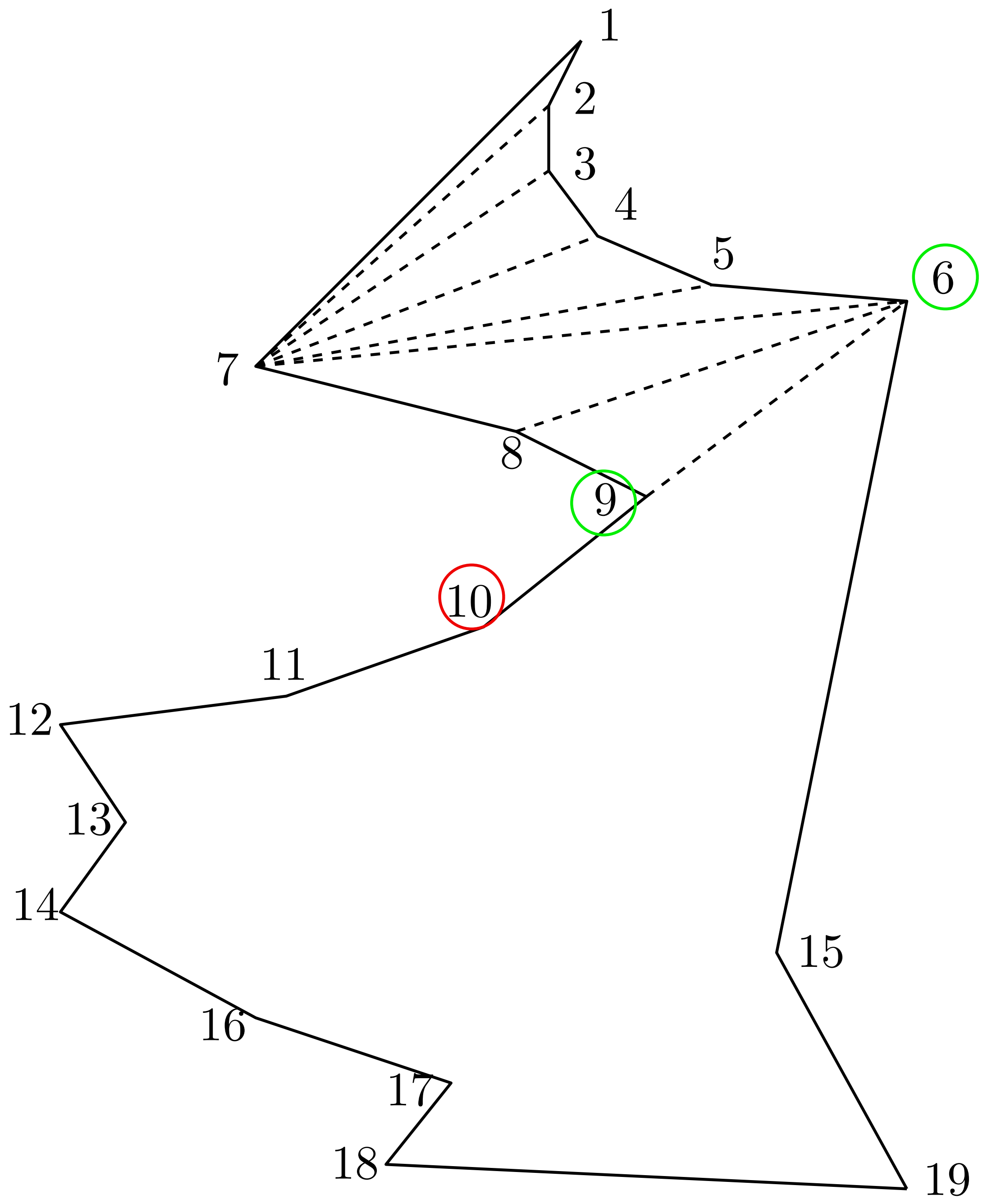
Triangulating a monotone polygon

Current vertex: 10

Add

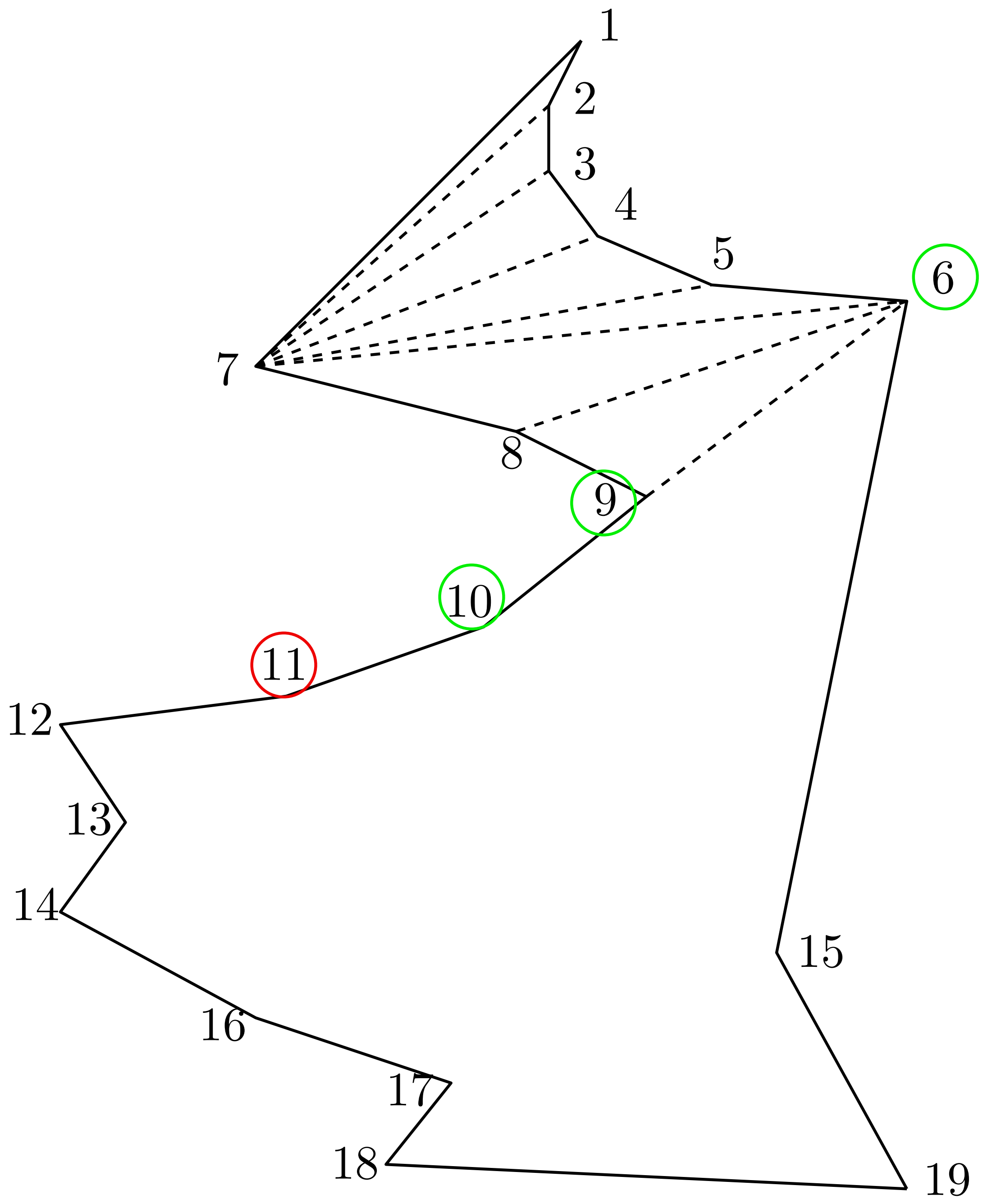
Queue state:

6, 9, 10



TRIANGULATING POLYGONS

Triangulating a monotone polygon



Current vertex: 11

Add

Queue state:

6, 9, 10, 11

TRIANGULATING POLYGONS

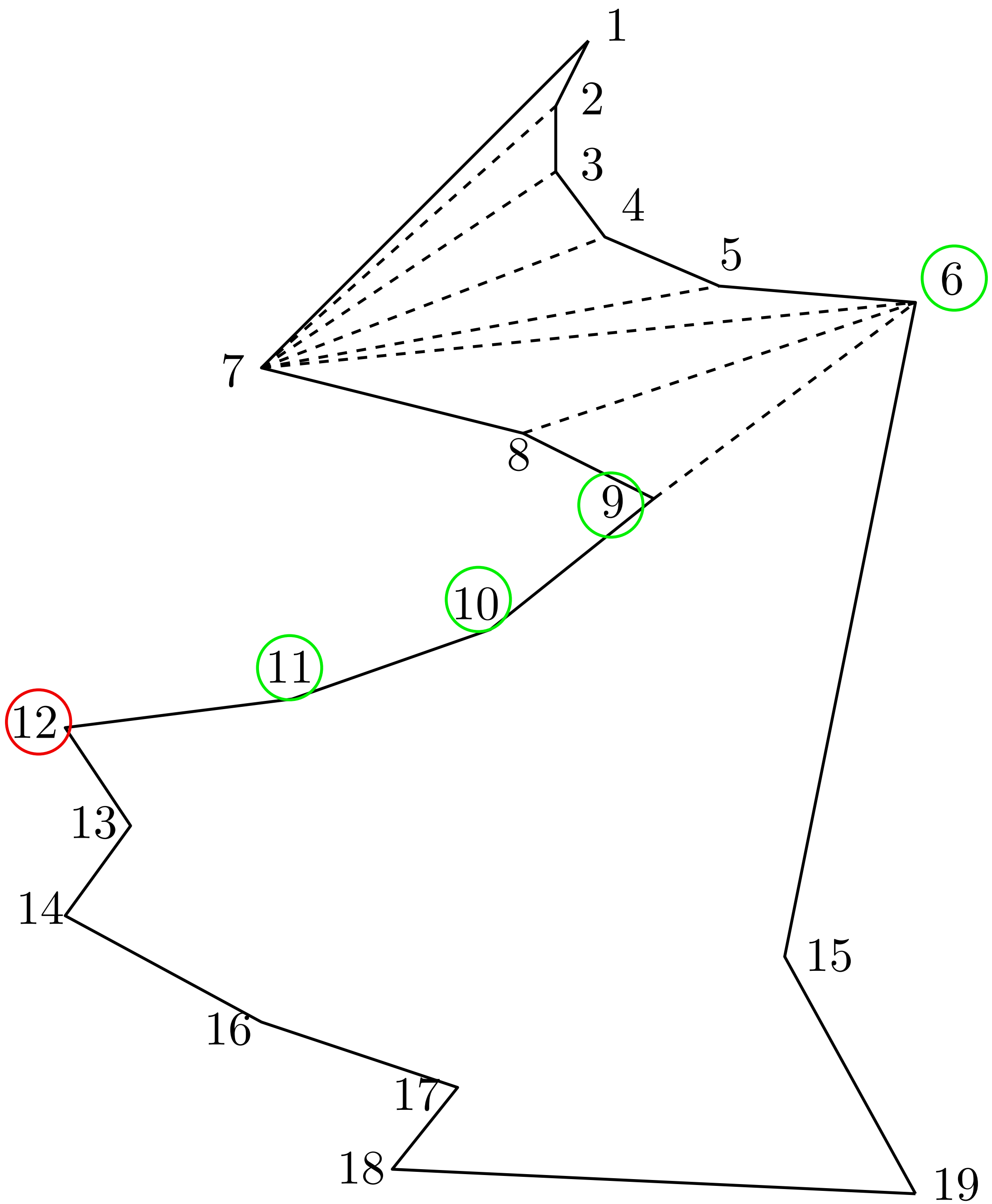
Triangulating a monotone polygon

Current vertex: 12

Add

Queue state:

6, 9, 10, 11, 12



TRIANGULATING POLYGONS

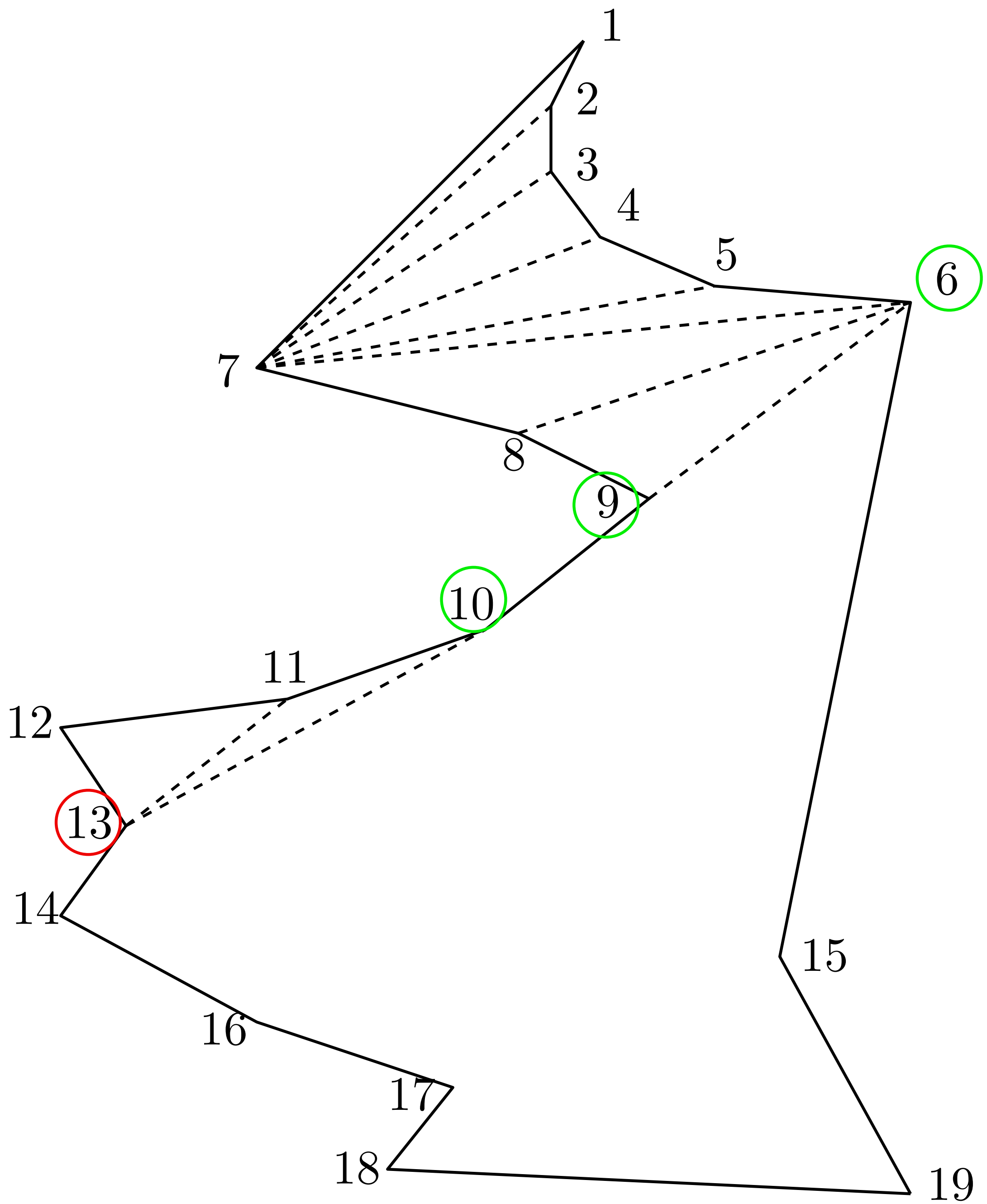
Triangulating a monotone polygon

Current vertex: 13

Ear

Queue state:

6, 9, 10, 13



TRIANGULATING POLYGONS

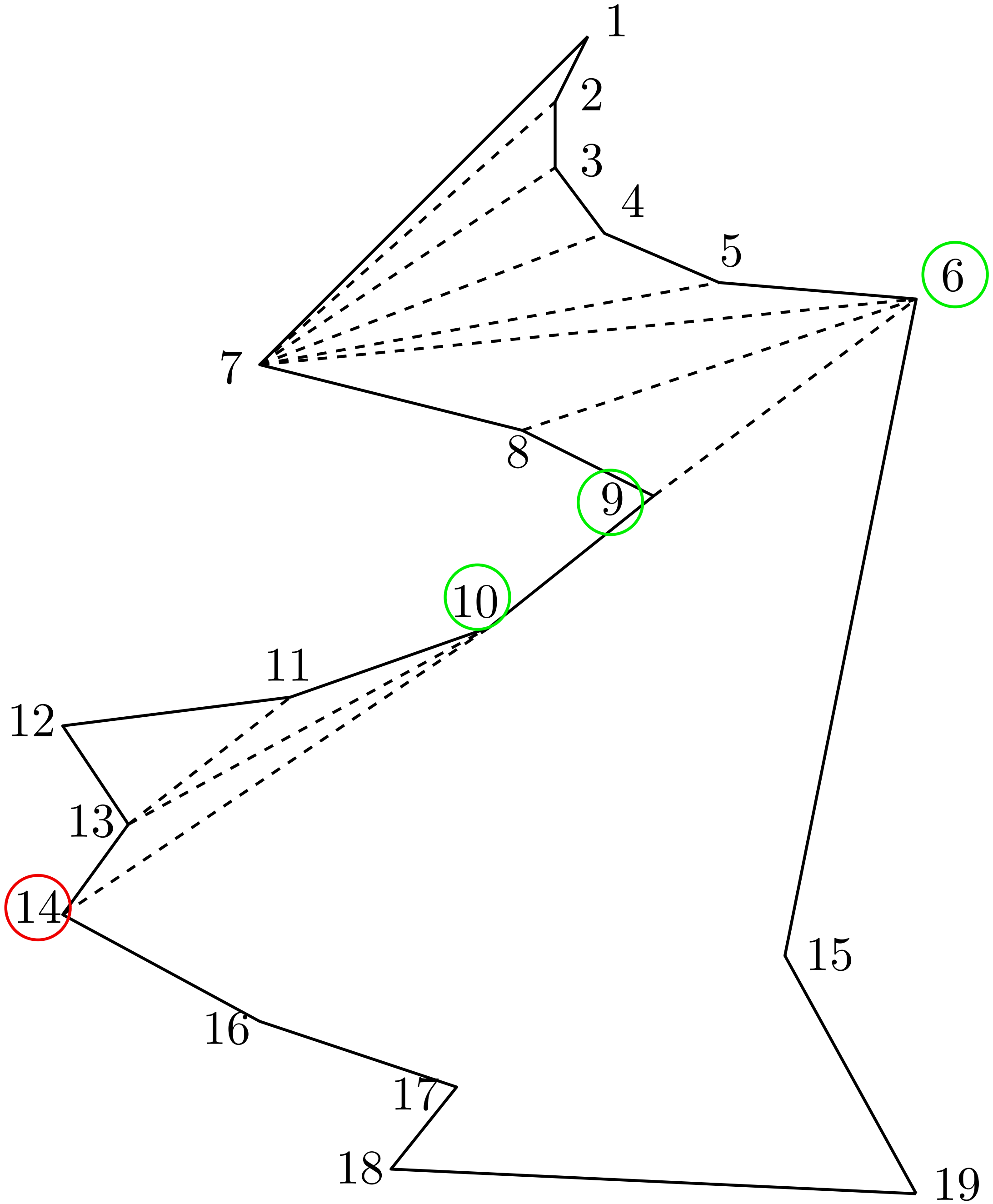
Triangulating a monotone polygon

Current vertex: 14

Ear

Queue state:

6, 9, 10, 14



TRIANGULATING POLYGONS

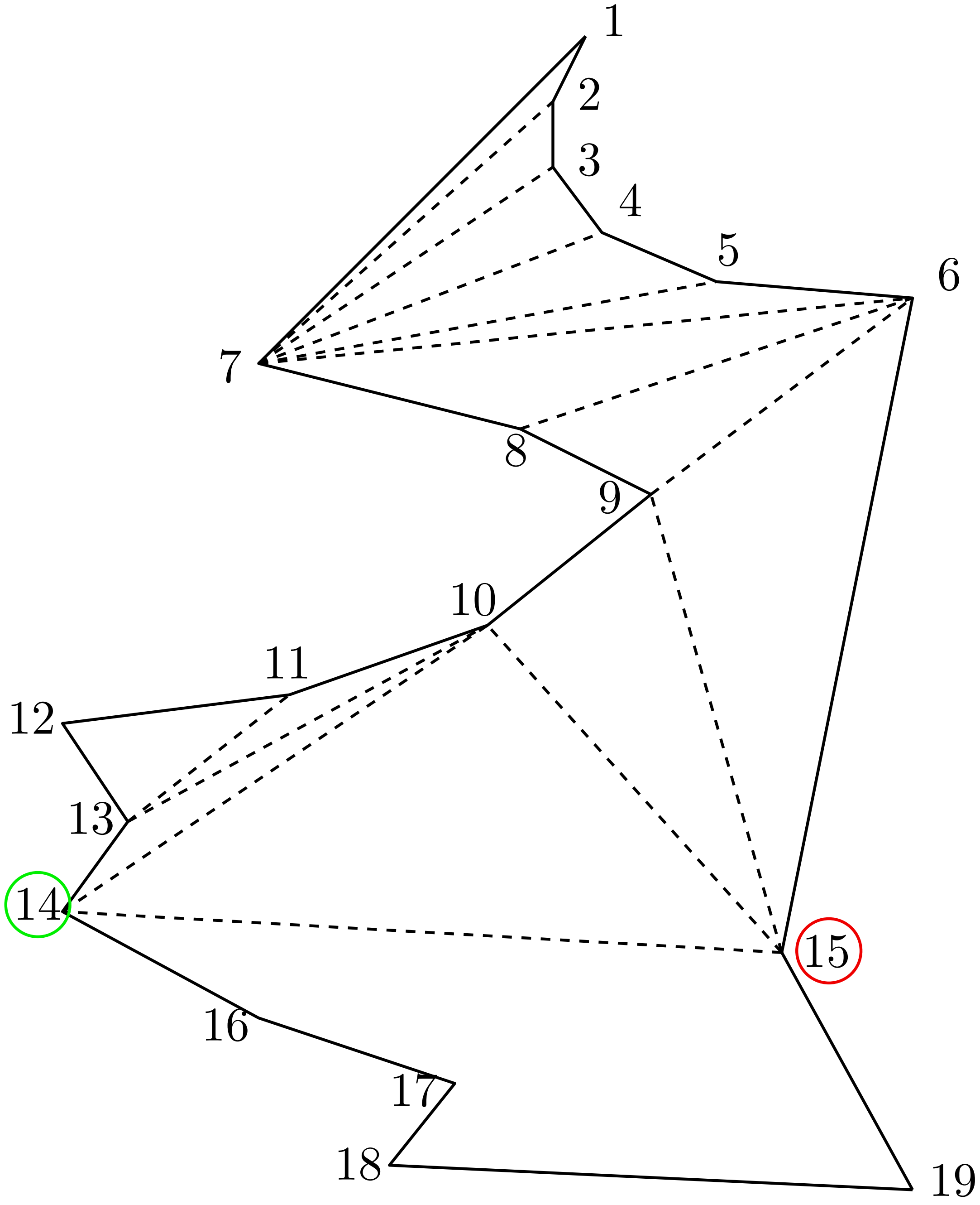
Triangulating a monotone polygon

Current vertex: 15

Opposite chain

Queue state:

14, 15



TRIANGULATING POLYGONS

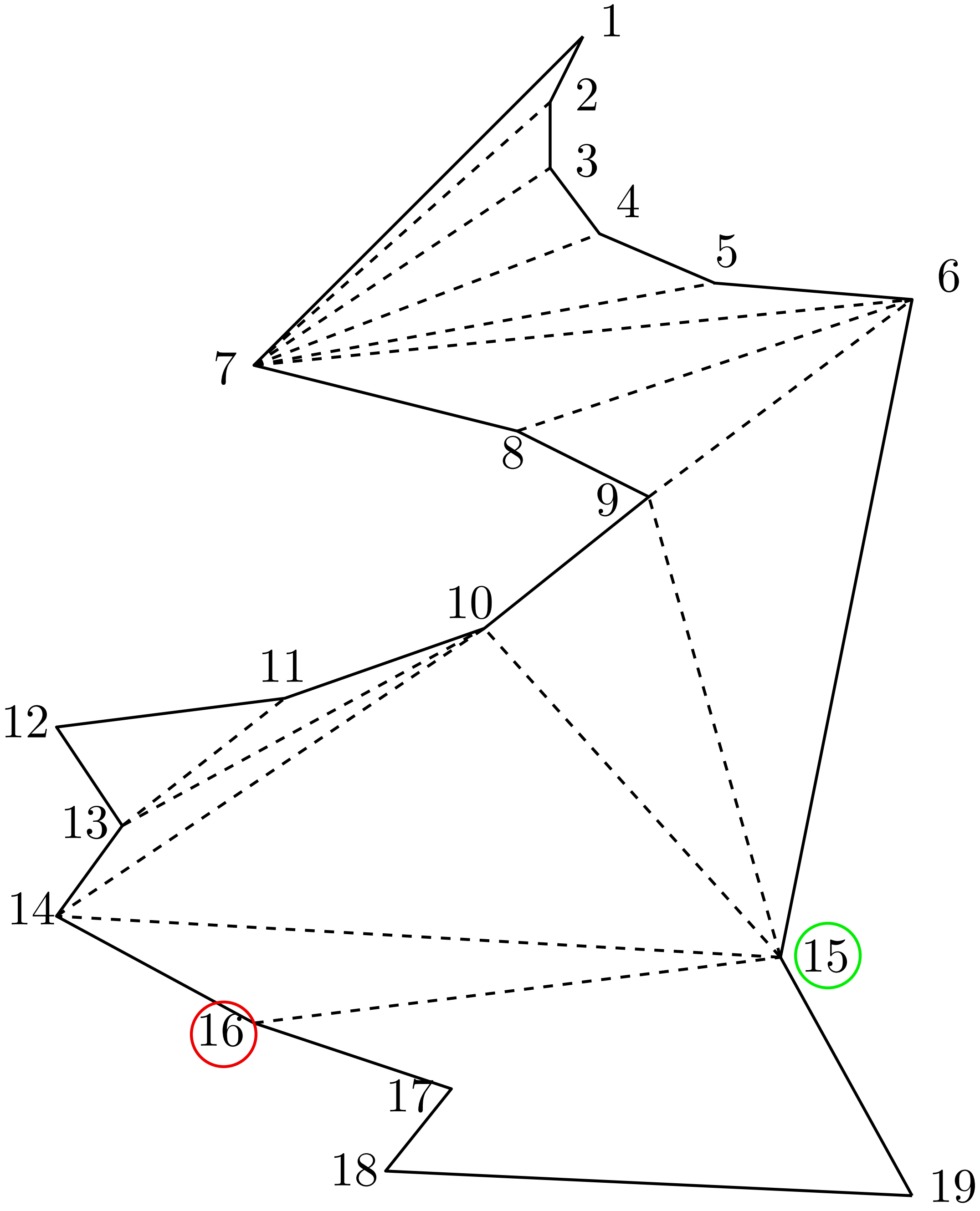
Triangulating a monotone polygon

Current vertex: 16

Opposite chain

Queue state:

15, 16



TRIANGULATING POLYGONS

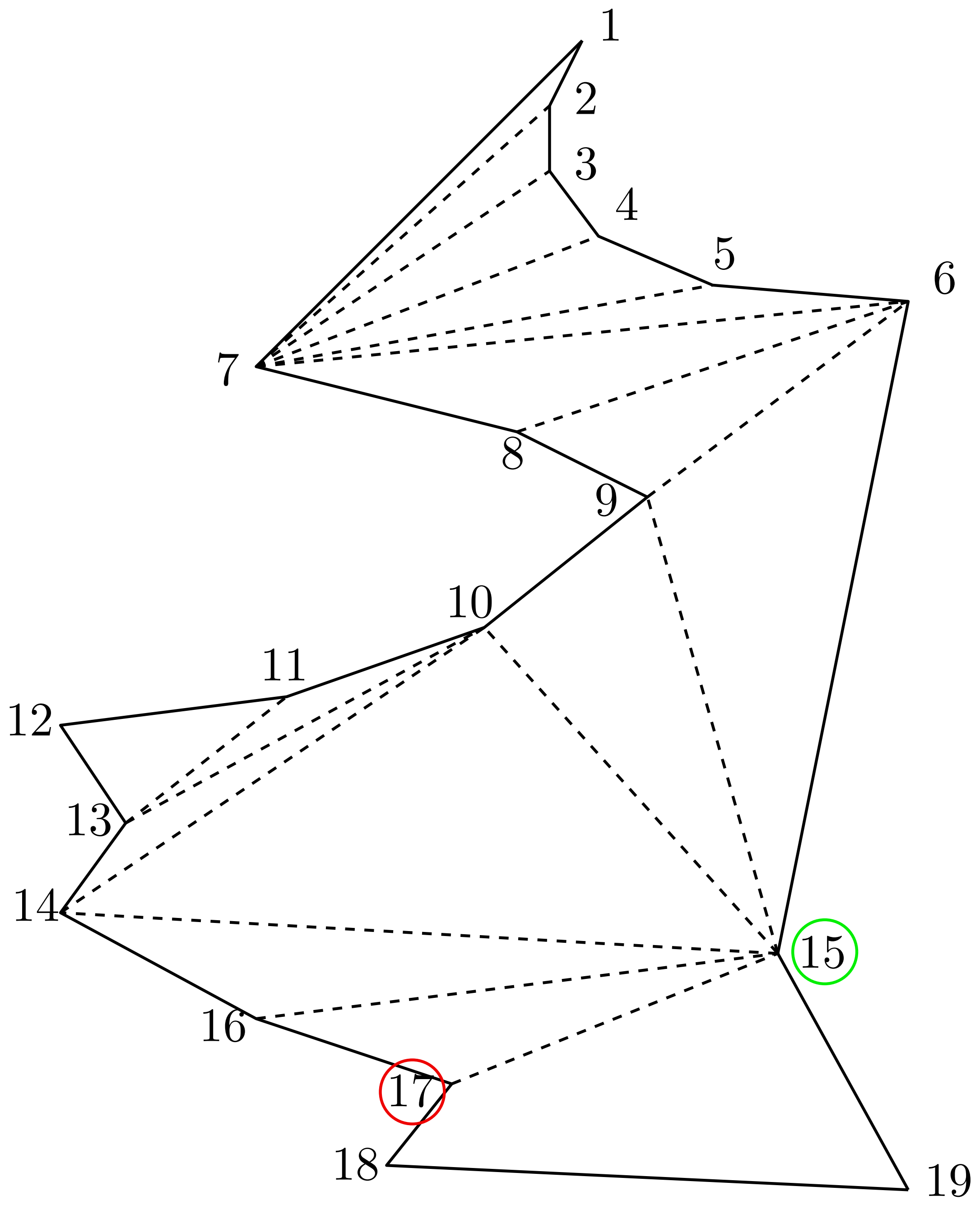
Triangulating a monotone polygon

Current vertex: 17

Ear

Queue state:

15, 17



TRIANGULATING POLYGONS

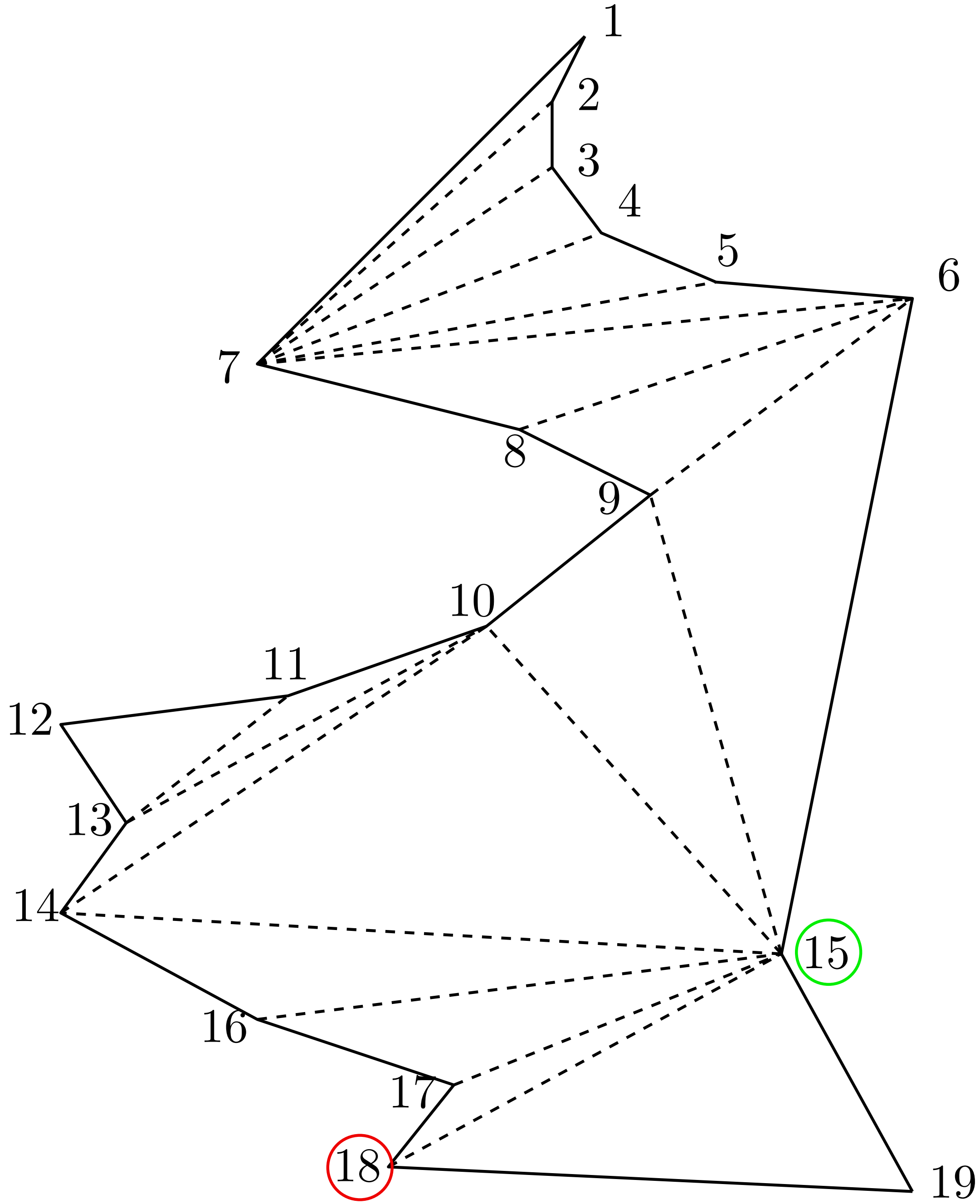
Triangulating a monotone polygon

Current vertex: 18

Ear

Queue state:

15, 18



TRIANGULATING POLYGONS

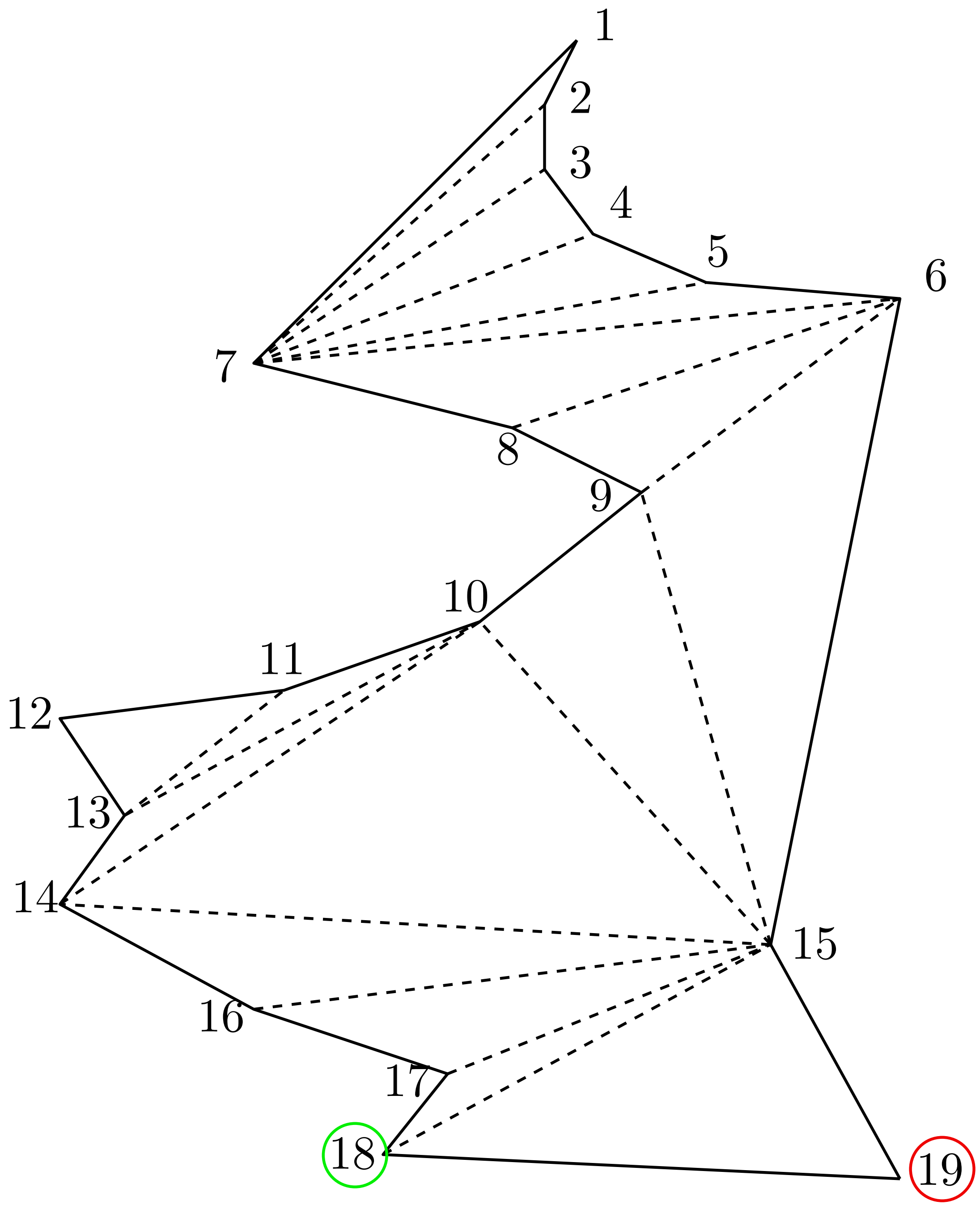
Triangulating a monotone polygon

Current vertex: 19

Opposite chain

Queue state:

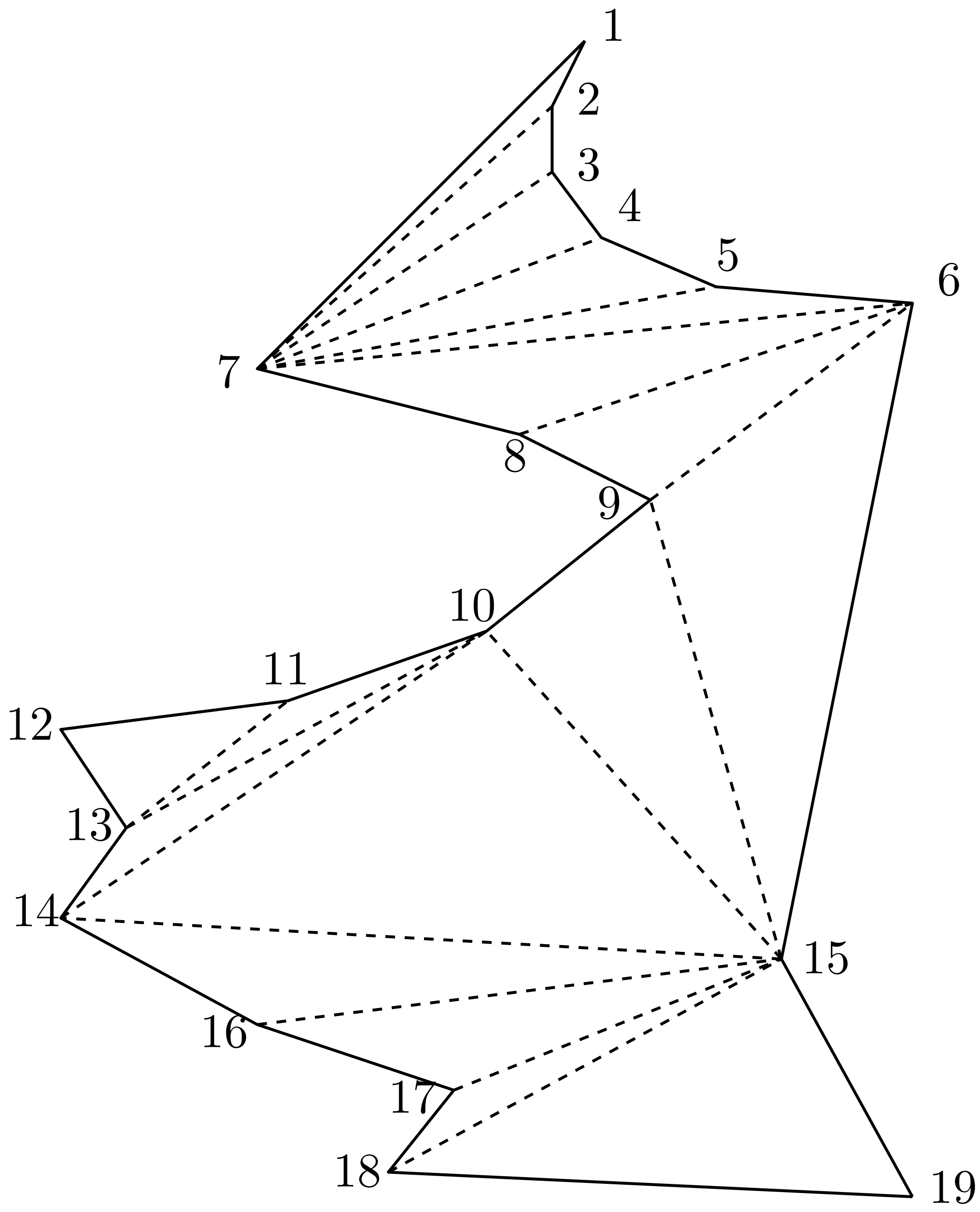
18, 19



TRIANGULATING POLYGONS

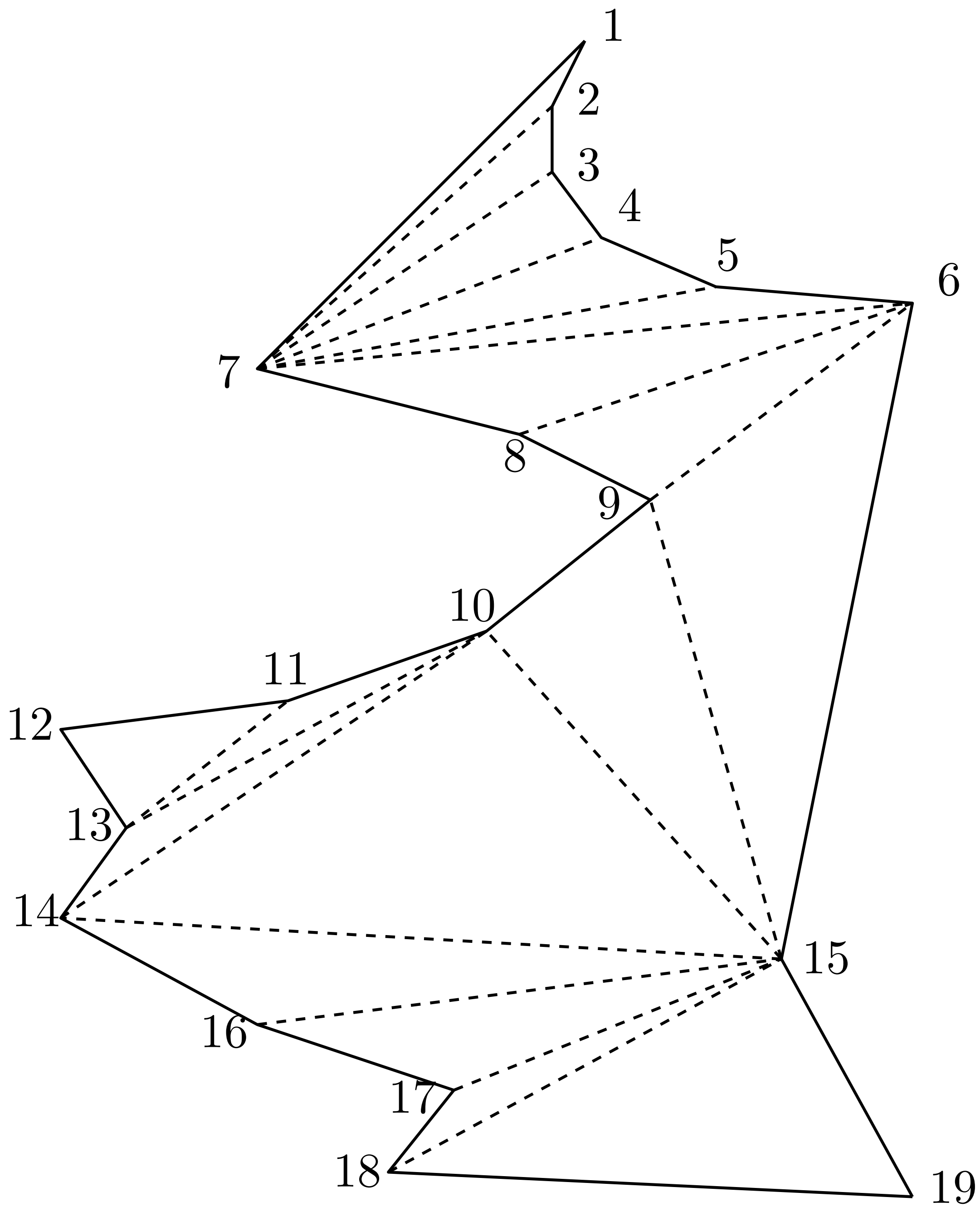
Triangulating a monotone polygon

End



TRIANGULATING POLYGONS

Triangulating a monotone polygon



Running time: $O(n)$

Each vertex is removed from the queue Q in $O(1)$ time.

TRIANGULATING POLYGONS

Summarizing

Running time for triangulating a polygon:

- $O(n^2)$ by subtracting ears
- $O(n^2)$ by inserting diagonals

If the polygon is convex:

- $O(n)$ trivially

If the polygon is monotone:

- $O(n)$ scanning the monotone chains in order

TRIANGULATING POLYGONS

Summarizing

Running time for triangulating a polygon:

- $O(n^2)$ by subtracting ears
- $O(n^2)$ by inserting diagonals

If the polygon is convex:

- $O(n)$ trivially

If the polygon is monotone:

- $O(n)$ scanning the monotone chains in order

Is it possible to be more efficient more general polygons?

TRIANGULATING POLYGONS

Summarizing

Running time for triangulating a polygon:

- $O(n^2)$ by subtracting ears
- $O(n^2)$ by inserting diagonals

If the polygon is convex:

- $O(n)$ trivially

If the polygon is monotone:

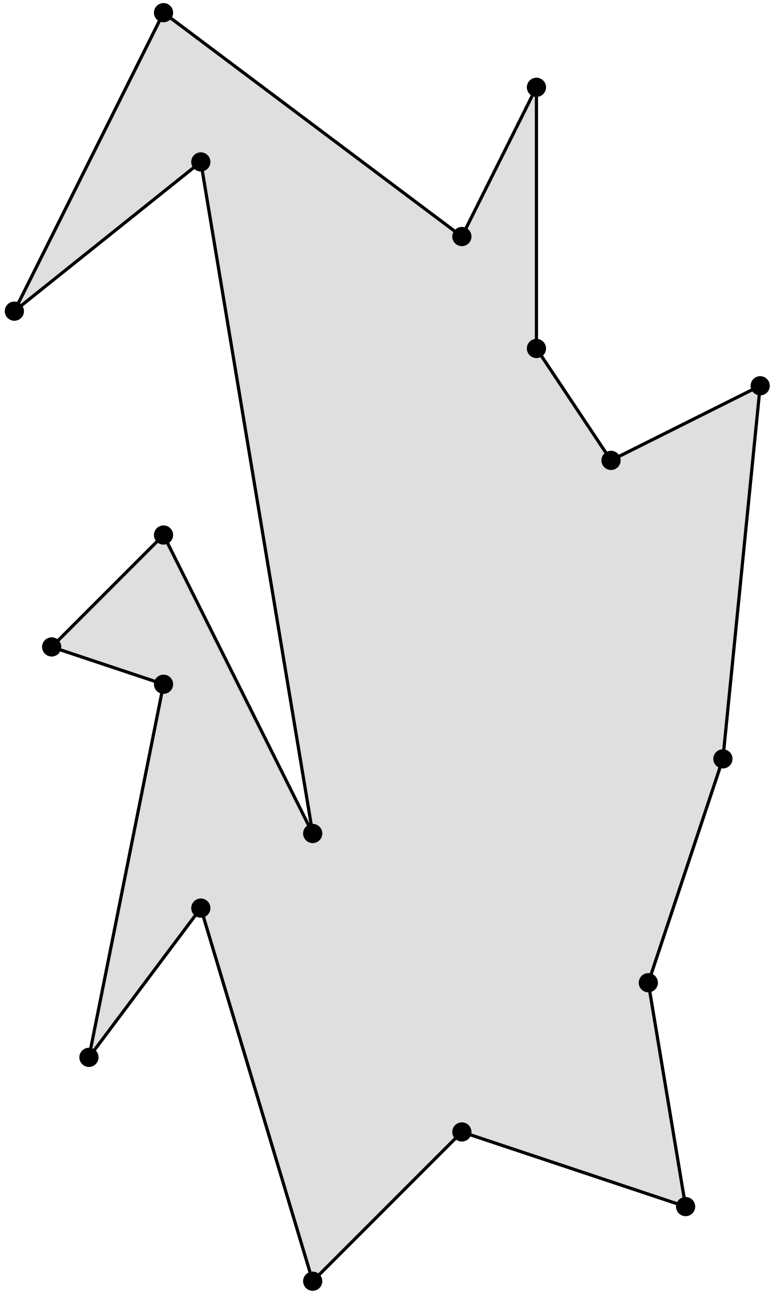
- $O(n)$ scanning the monotone chains in order

Is it possible to be more efficient more general polygons? Yes!

1. Decompose the polygon into monotone subpolygons
2. Triangulate the monotone subpolygons

TRIANGULATING POLYGONS

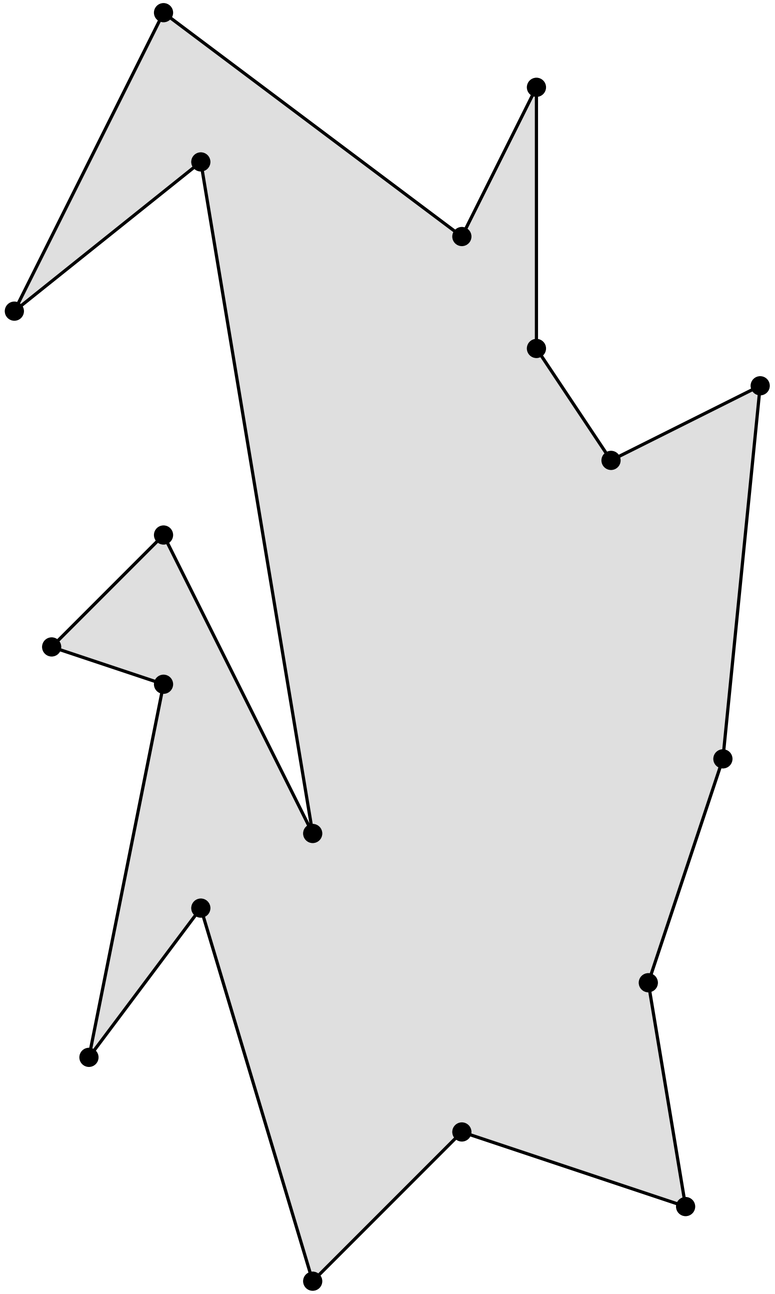
Monotone partition



TRIANGULATING POLYGONS

Monotone partition

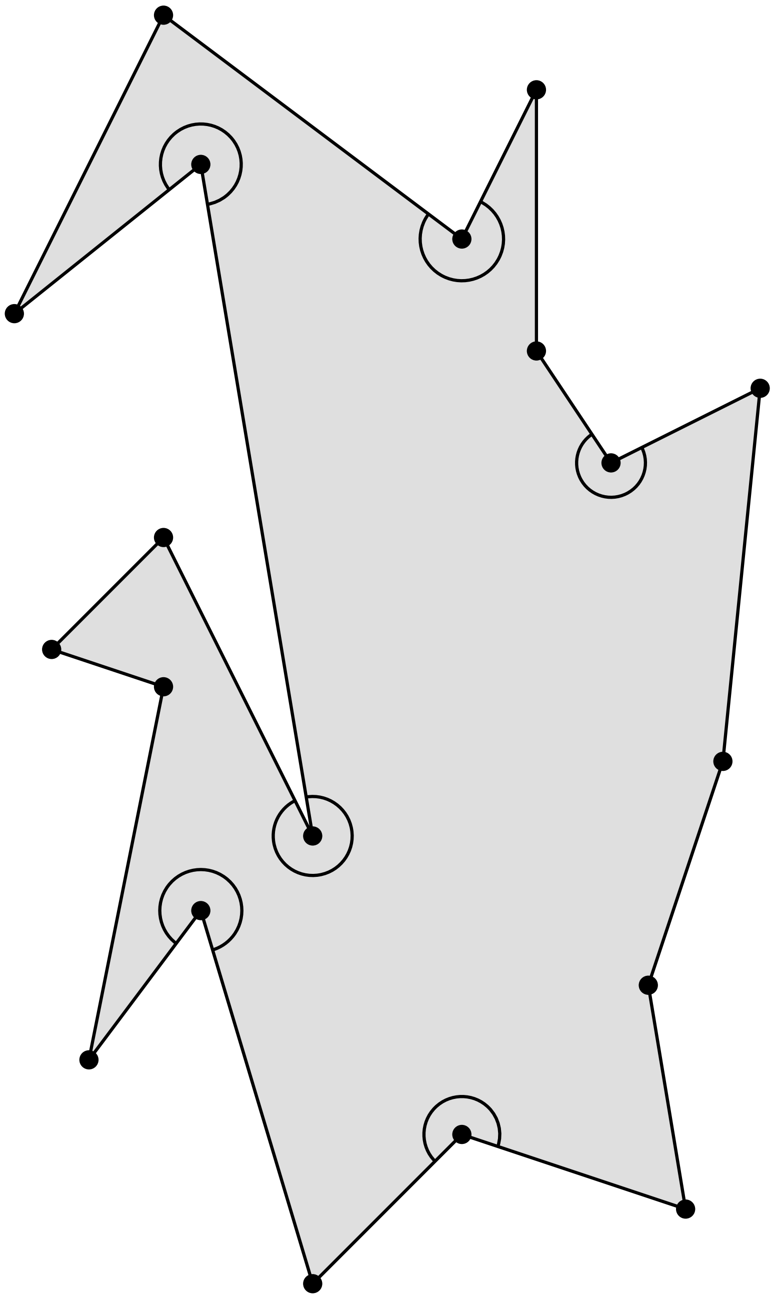
In order to create a monotone partition of a polygon, all cusps need to be "broken" by internal diagonals.



TRIANGULATING POLYGONS

Monotone partition

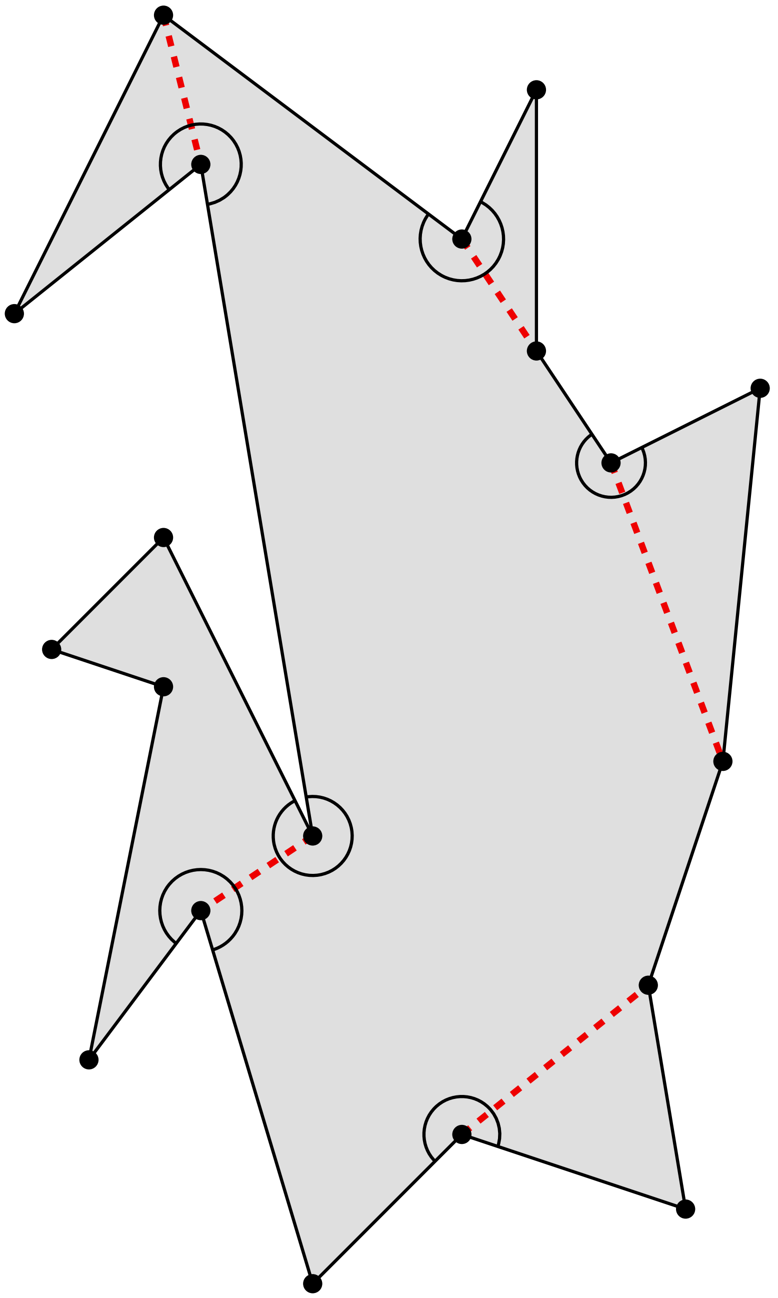
In order to create a monotone partition of a polygon, all cusps need to be "broken" by internal diagonals.



TRIANGULATING POLYGONS

Monotone partition

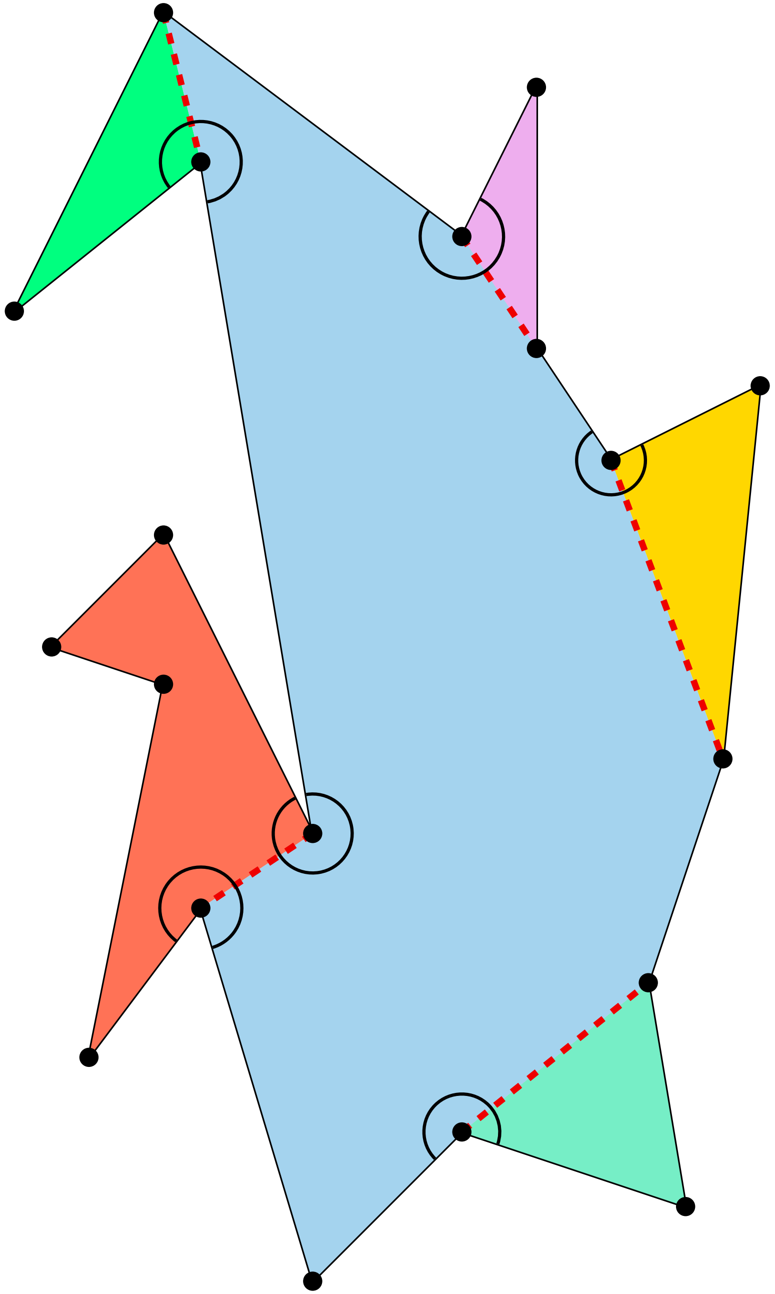
In order to create a monotone partition of a polygon, all cusps need to be “broken” by internal diagonals.



TRIANGULATING POLYGONS

Monotone partition

In order to create a monotone partition of a polygon, all cusps need to be “broken” by internal diagonals.

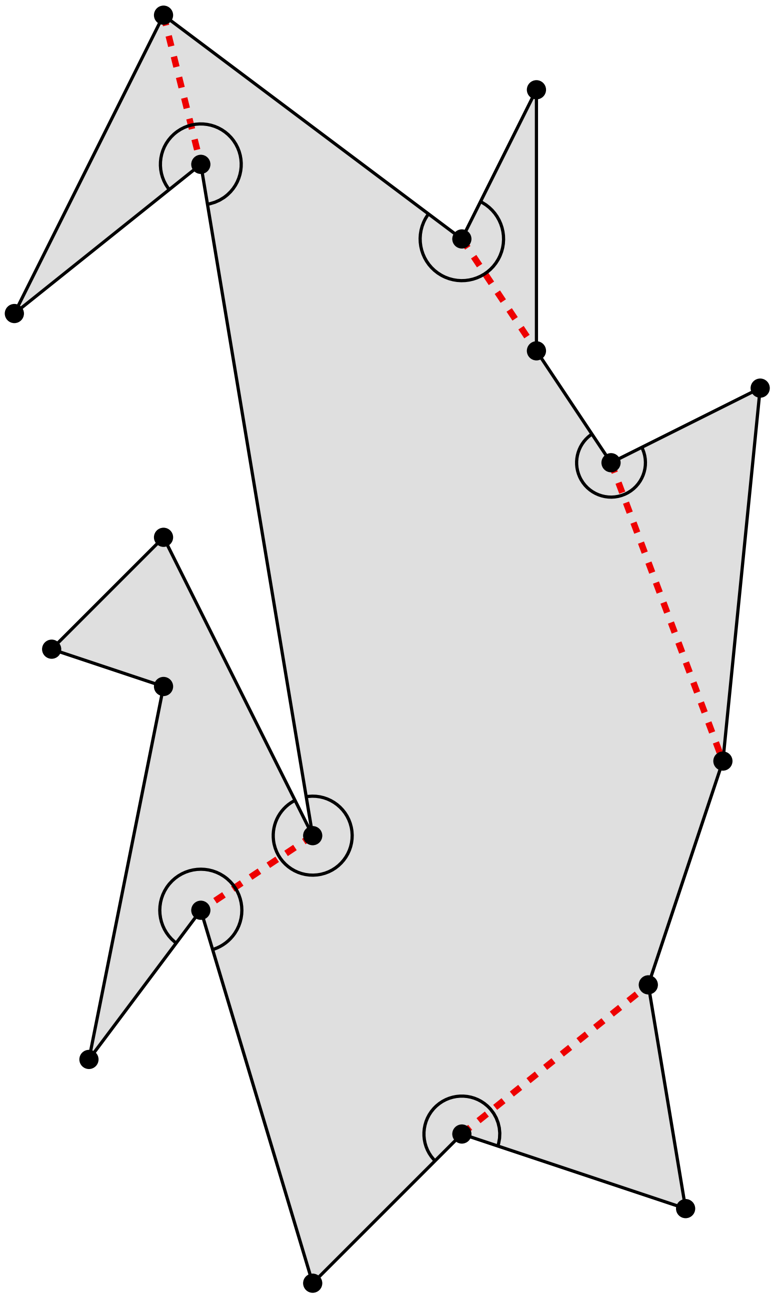


TRIANGULATING POLYGONS

Monotone partition

In order to create a monotone partition of a polygon, all cusps need to be "broken" by internal diagonals.

This can be done starting from a **trapezoidal decomposition** of the polygon.

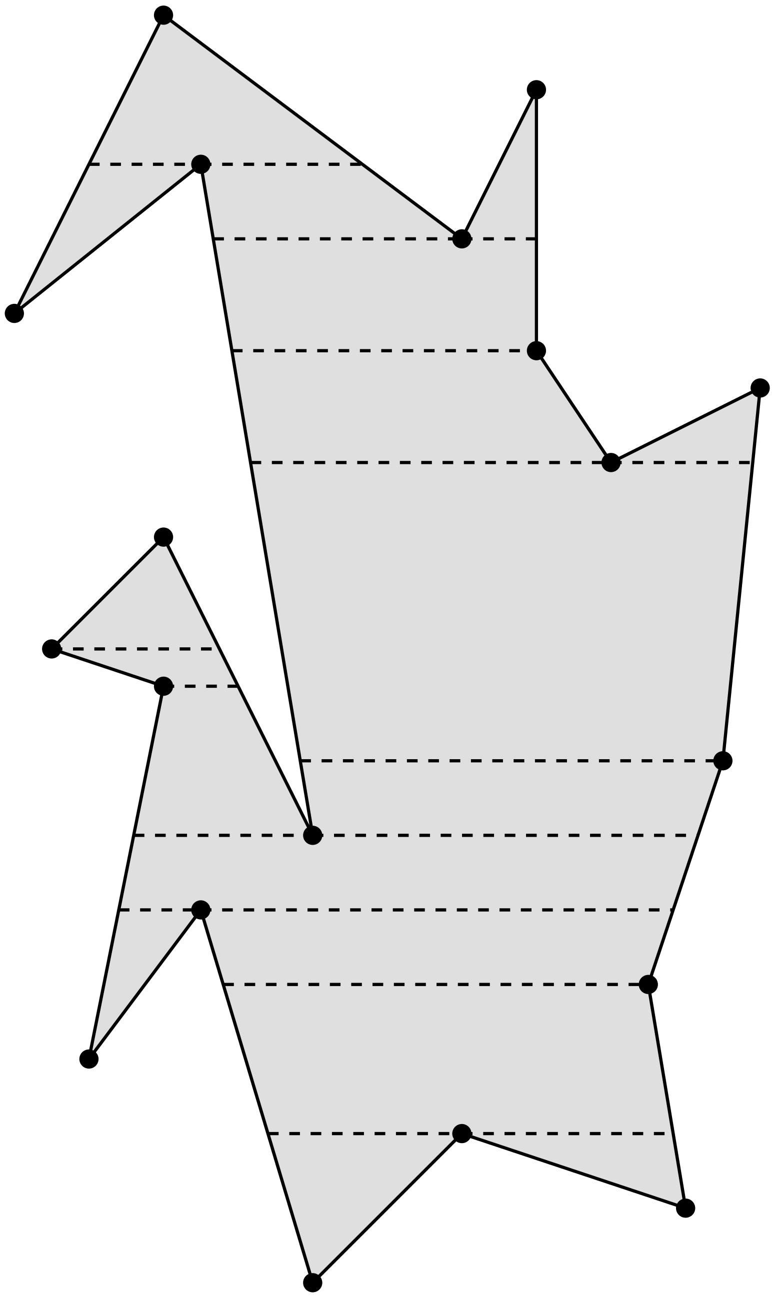


TRIANGULATING POLYGONS

Monotone partition

In order to create a monotone partition of a polygon, all cusps need to be “broken” by internal diagonals.

This can be done starting from a **trapezoidal decomposition** of the polygon.



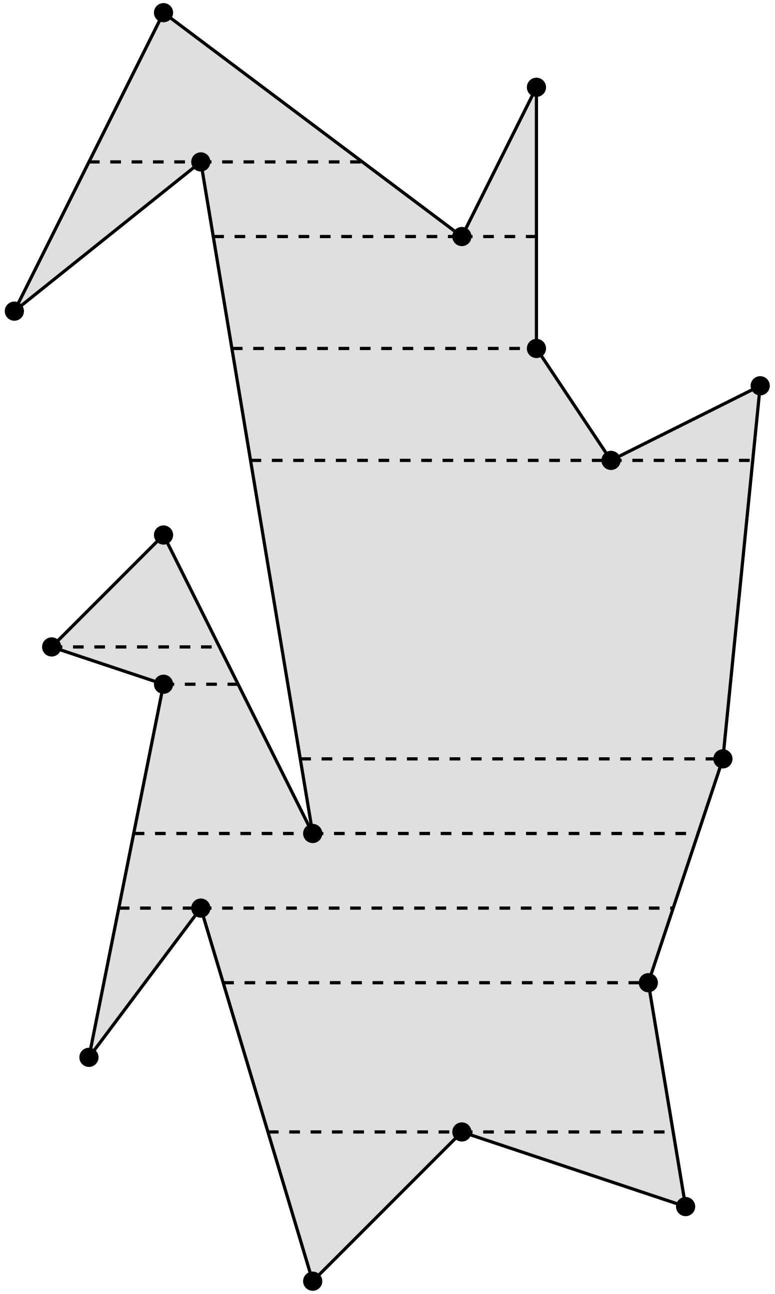
TRIANGULATING POLYGONS

Monotone partition

In order to create a monotone partition of a polygon, all cusps need to be “broken” by internal diagonals.

This can be done starting from a **trapezoidal decomposition** of the polygon.

Connect each cusp with the opposite vertex in its trapezoid (the upper trapezoid, if the cusp is a local maximum, the lower one, if it is a local minimum).



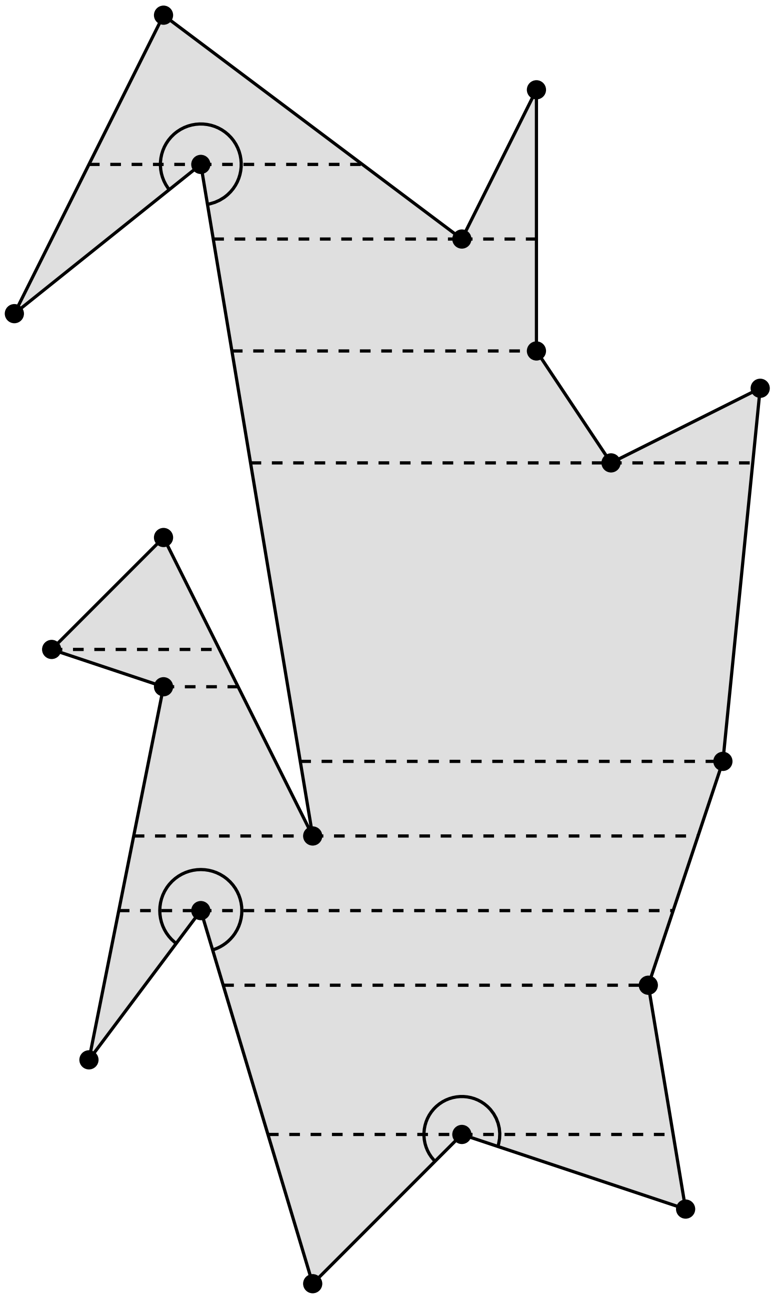
TRIANGULATING POLYGONS

Monotone partition

In order to create a monotone partition of a polygon, all cusps need to be “broken” by internal diagonals.

This can be done starting from a **trapezoidal decomposition** of the polygon.

Connect each cusp with the opposite vertex in its trapezoid (the upper trapezoid, if the cusp is a local maximum, the lower one, if it is a local minimum).



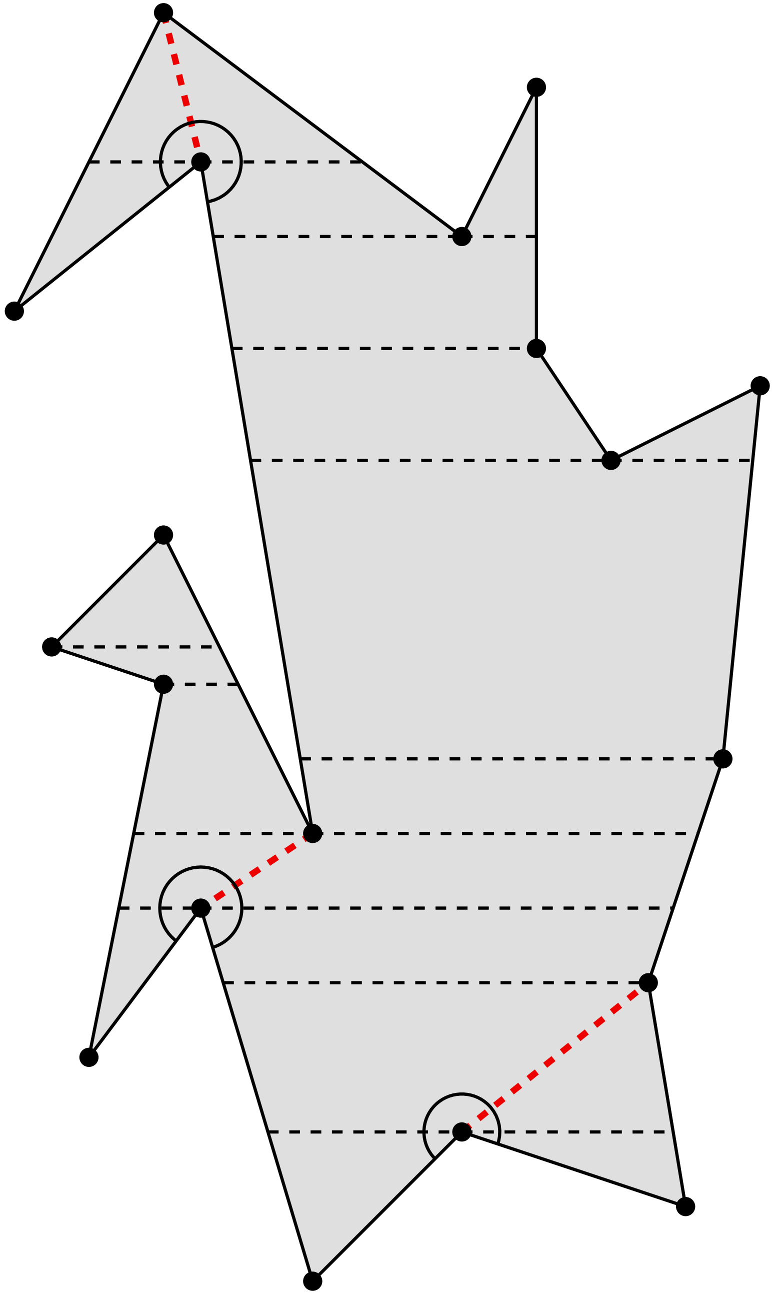
TRIANGULATING POLYGONS

Monotone partition

In order to create a monotone partition of a polygon, all cusps need to be “broken” by internal diagonals.

This can be done starting from a **trapezoidal decomposition** of the polygon.

Connect each cusp with the opposite vertex in its trapezoid (the upper trapezoid, if the cusp is a local maximum, the lower one, if it is a local minimum).



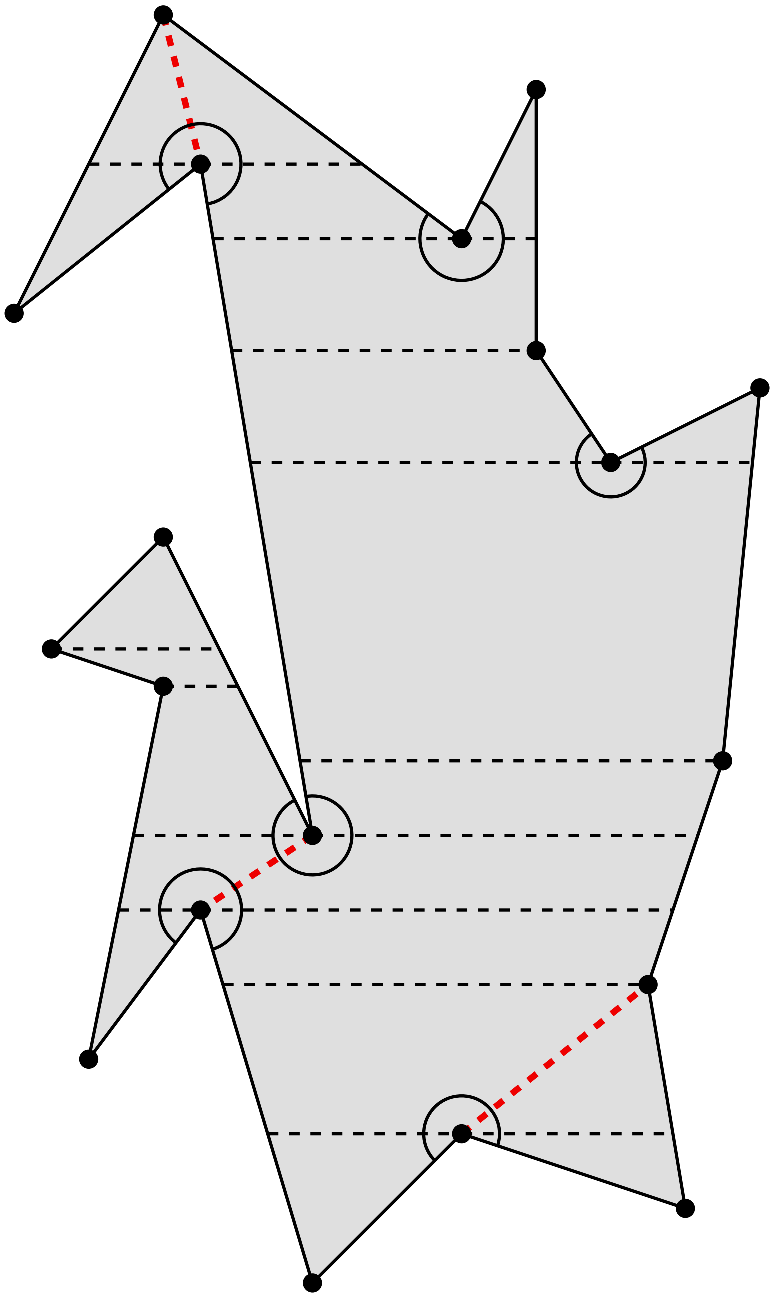
TRIANGULATING POLYGONS

Monotone partition

In order to create a monotone partition of a polygon, all cusps need to be “broken” by internal diagonals.

This can be done starting from a **trapezoidal decomposition** of the polygon.

Connect each cusp with the opposite vertex in its trapezoid (the upper trapezoid, if the cusp is a local maximum, the lower one, if it is a local minimum).



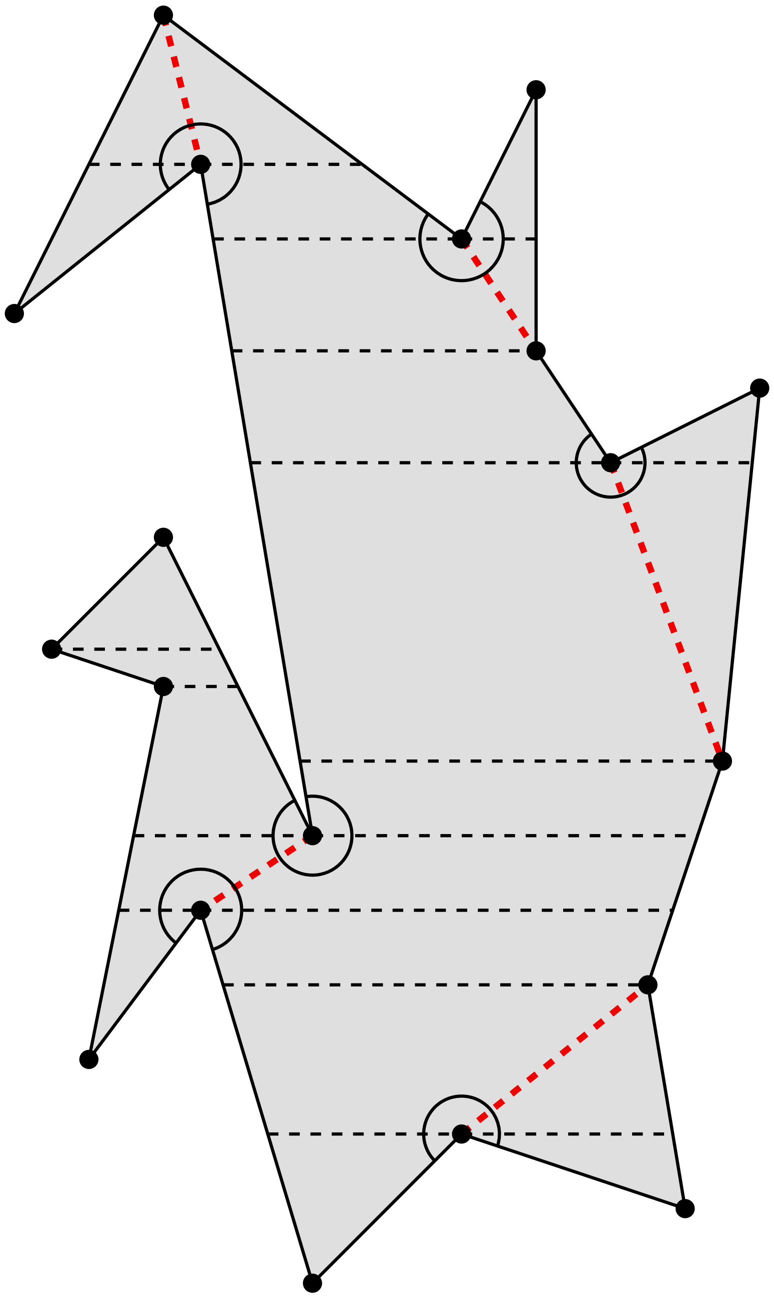
TRIANGULATING POLYGONS

Monotone partition

In order to create a monotone partition of a polygon, all cusps need to be “broken” by internal diagonals.

This can be done starting from a **trapezoidal decomposition** of the polygon.

Connect each cusp with the opposite vertex in its trapezoid (the upper trapezoid, if the cusp is a local maximum, the lower one, if it is a local minimum).



TRIANGULATING POLYGONS

Monotone partition

In order to create a monotone partition of a polygon, all cusps need to be "broken" by internal diagonals.

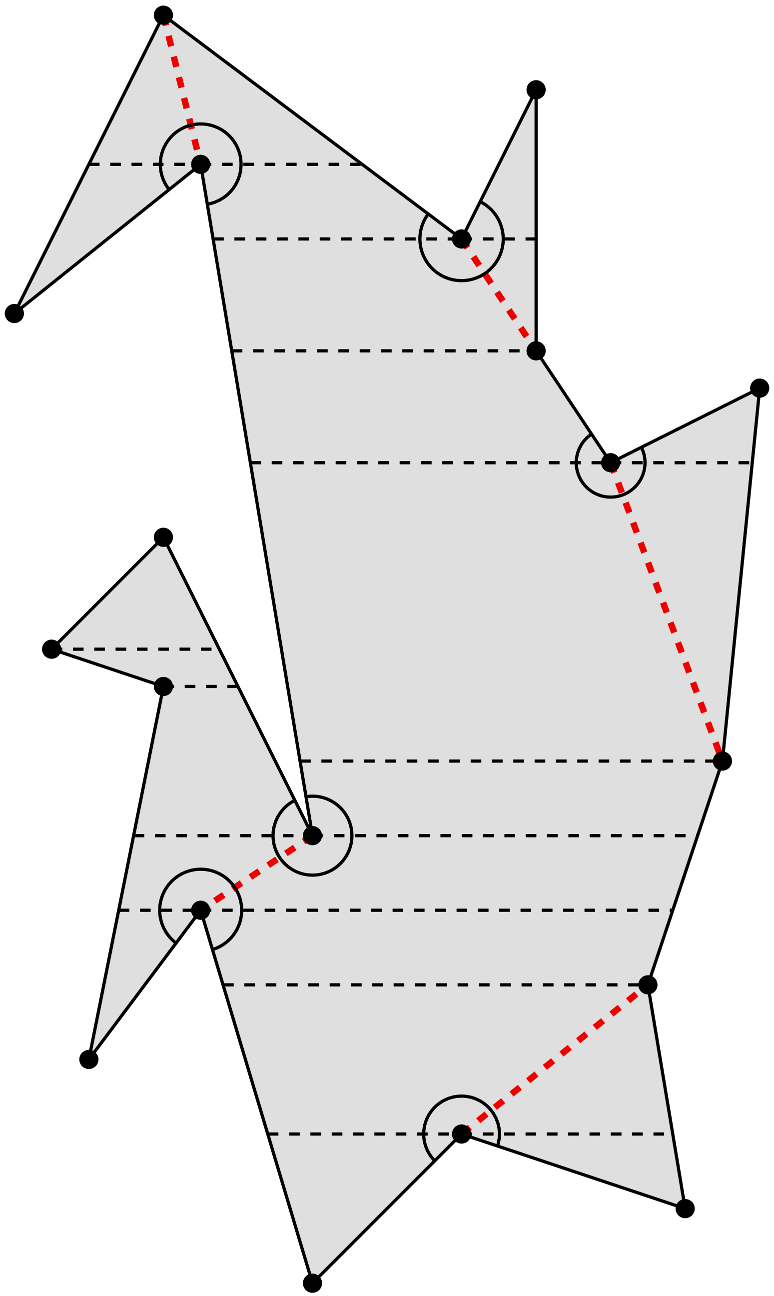
This can be done starting from a **trapezoidal decomposition** of the polygon.

Connect each cusp with the opposite vertex in its trapezoid (the upper trapezoid, if the cusp is a local maximum, the lower one, if it is a local minimum).

This gives rise to a correct algorithm:

- The diagonals do not intersect, because they belong to different trapezoids.
- The polygon ends up decomposed into monotone subpolygons.

Sweep line algorithm



TRIANGULATING POLYGONS

Monotone partition

Sweep line algorithm

TRIANGULATING POLYGONS

Monotone partition

Sweep line algorithm

A straight line (horitzontal, in this case) scans the object (the polygon) and allows detecting and constructing the desired elements (cusps, trapezoids, diagonals), leaving the problem solved behind it. The sweeping process is discretized.

TRIANGULATING POLYGONS

Monotone partition

Sweep line algorithm

A straight line (horitzontal, in this case) scans the object (the polygon) and allows detecting and constructing the desired elements (cusps, trapezoids, diagonals), leaving the problem solved behind it. The sweeping process is discretized.

Essential elements of a sweep line algorithm:

- Events queue

  Priority queue keeping the information of the algorithm stops. In our problem, the events will be the vertices of the polygon, sorted by their y-coordinate, all known in advance.

- Sweep line

  Data structure storing the information of the portion of the object intersected by the sweep line. It gets updated at each event. In our problem, it will contain the information of the edges of the polygon intersected by the sweep line, sorted by abscissa, as well as the list of active trapezoids and ther upper vertex.

Monotone partition

Updating the sweep line:

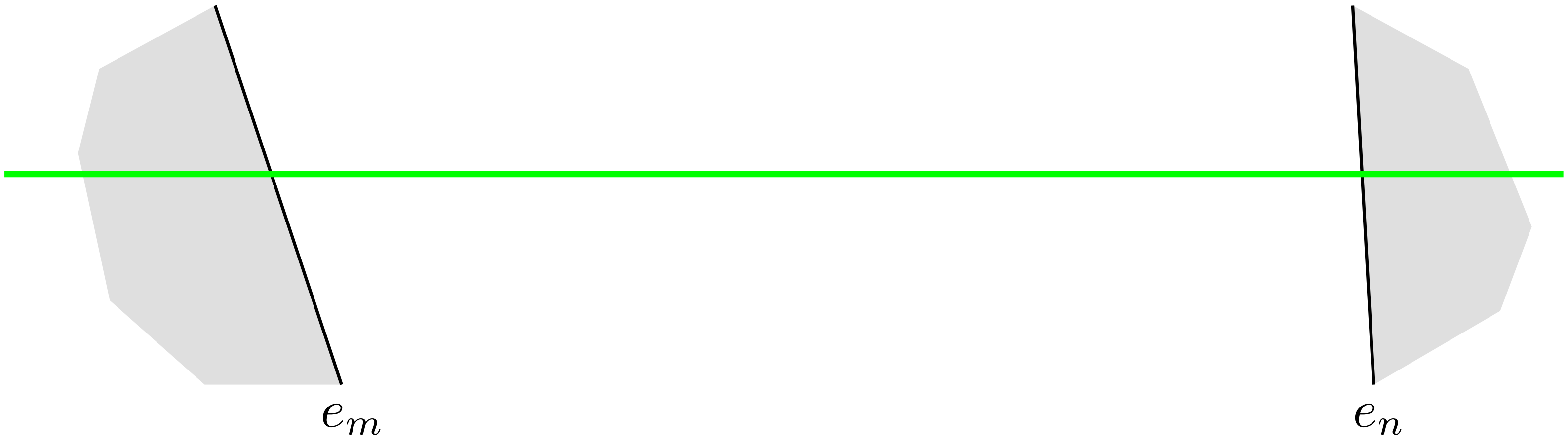
- Passing vertex: replace
- Local minimum cusp: delete
- Local maximum cusp: insert
- Initial vertex: insert
- Final vertex: delete

TRIANGULATING POLYGONS

Monotone partition

Updating the sweep line:

- Passing vertex: replace
- Local minimum cusp: delete
- Local maximum cusp: insert
- Initial vertex: insert
- Final vertex: delete

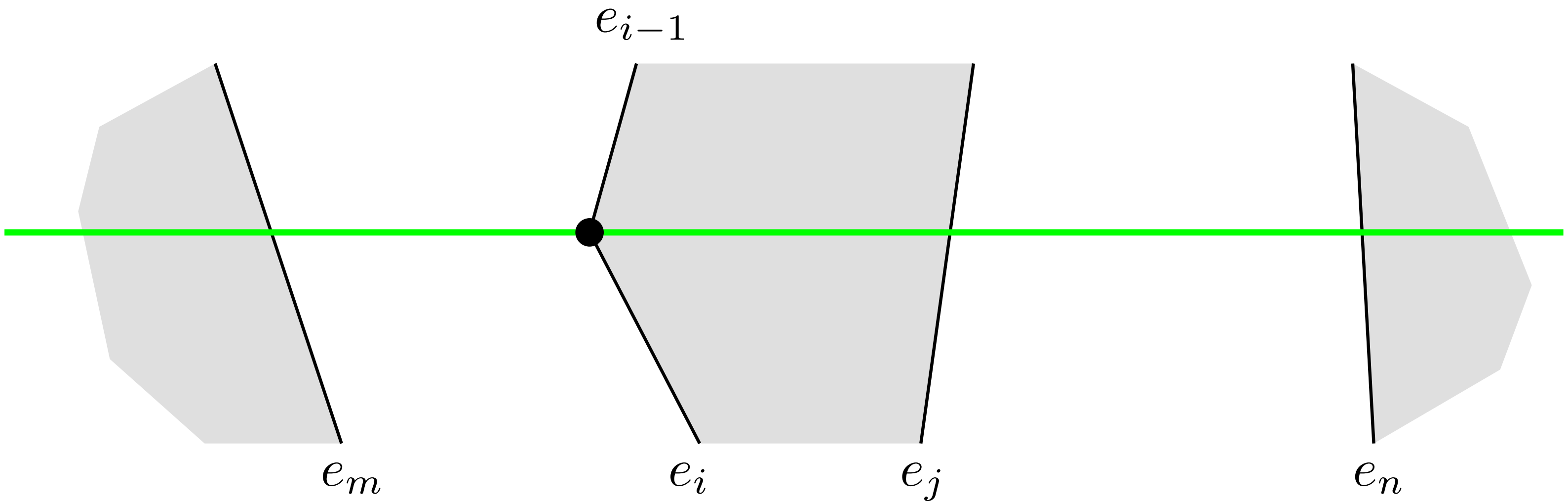


TRIANGULATING POLYGONS

Monotone partition

Updating the sweep line:

- Passing vertex: replace
- Local minimum cusp: delete
- Local maximum cusp: insert
- Initial vertex: insert
- Final vertex: delete

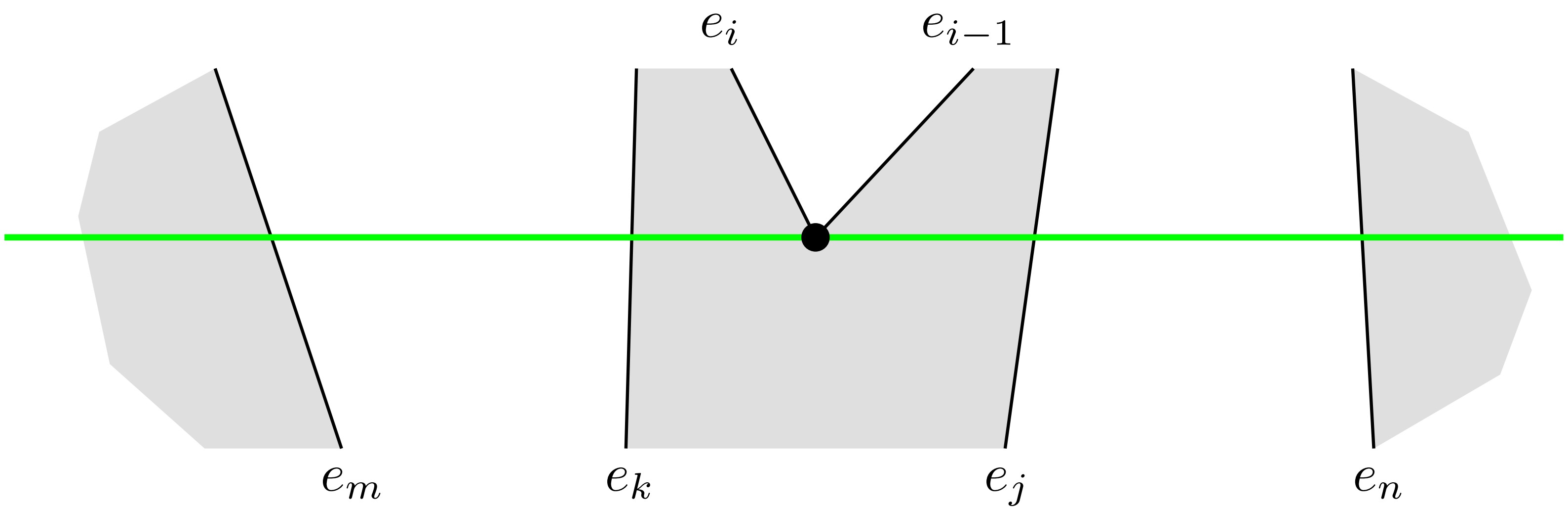


TRIANGULATING POLYGONS

Monotone partition

Updating the sweep line:

- Passing vertex: replace
- Local minimum cusp: delete
- Local maximum cusp: insert
- Initial vertex: insert
- Final vertex: delete

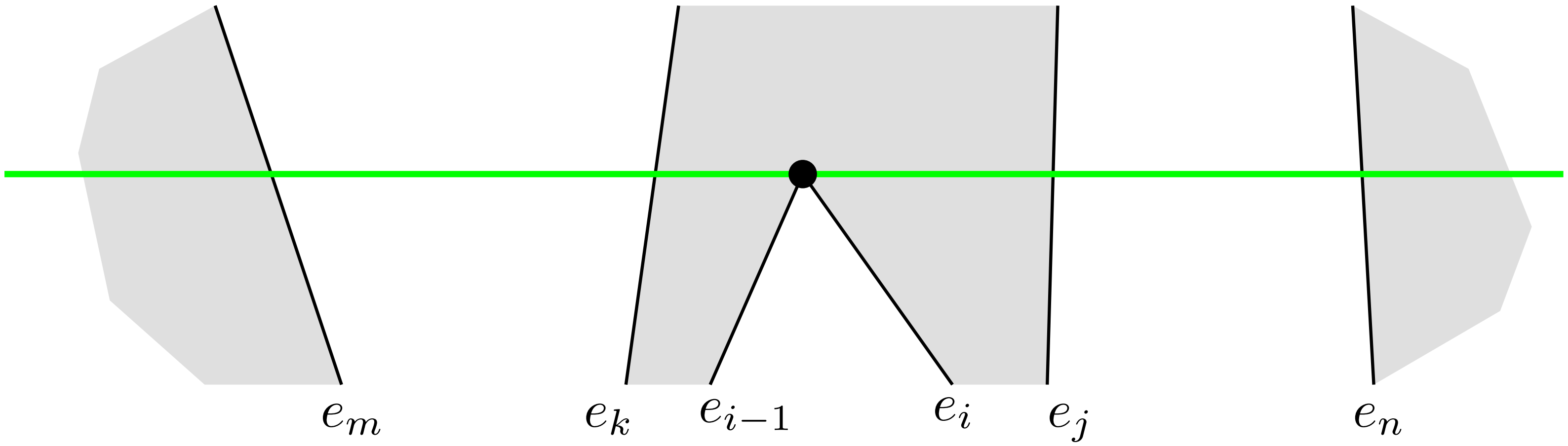


TRIANGULATING POLYGONS

Monotone partition

Updating the sweep line:

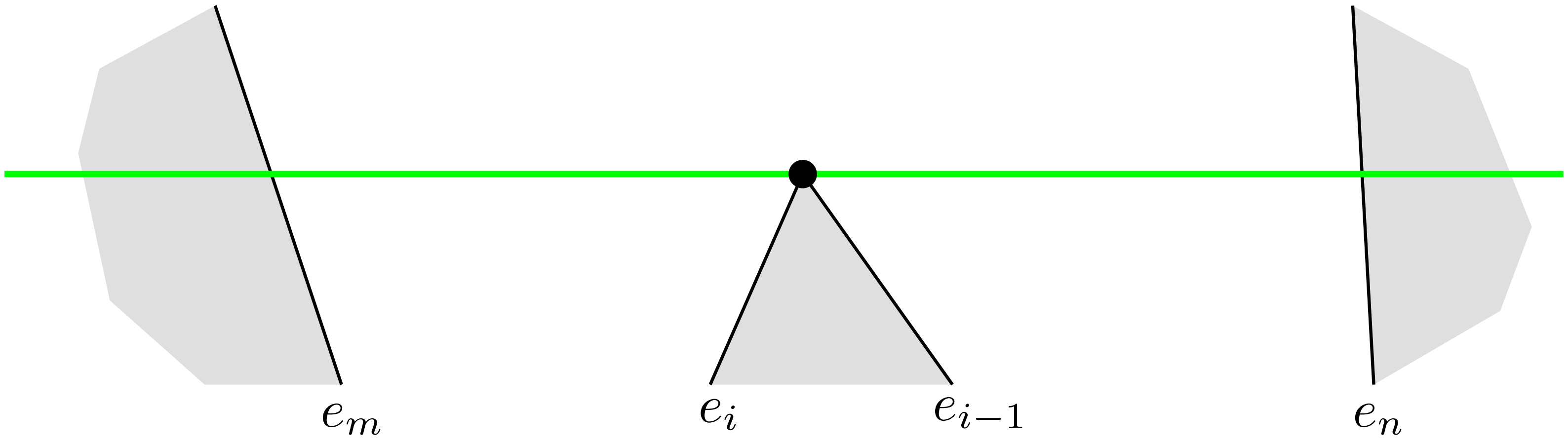
- Passing vertex: replace
- Local minimum cusp: delete
- Local maximum cusp: insert
- Initial vertex: insert
- Final vertex: delete



Monotone partition

Updating the sweep line:

- Passing vertex: replace
- Local minimum cusp: delete
- Local maximum cusp: insert
- Initial vertex: insert
- Final vertex: delete

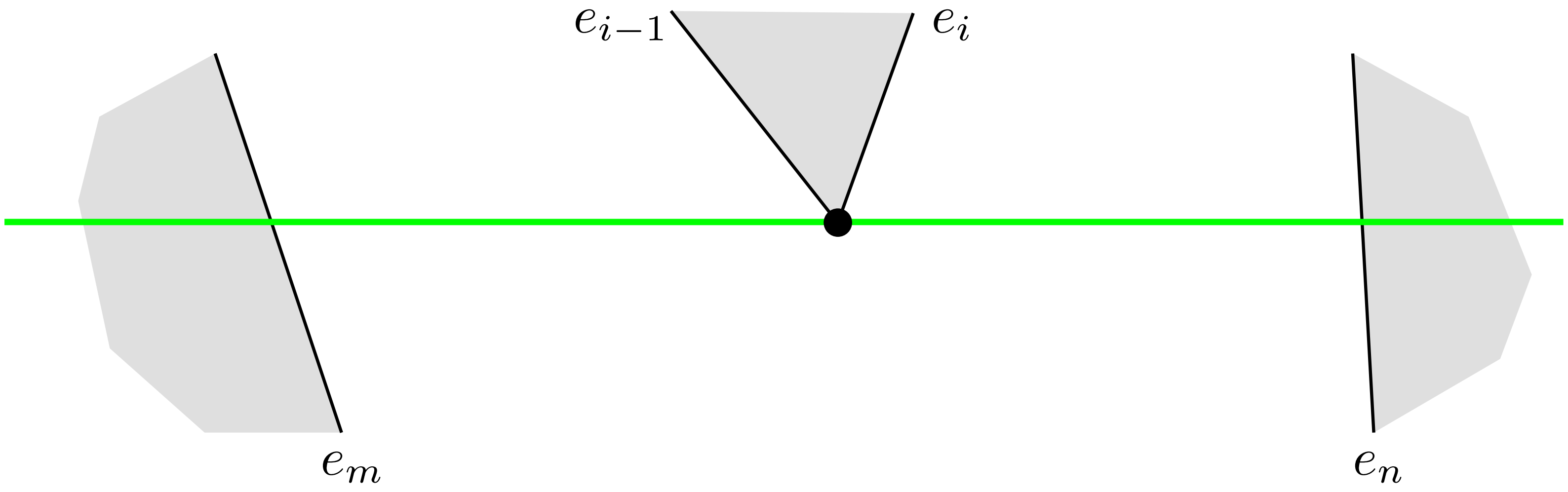


TRIANGULATING POLYGONS

Monotone partition

Updating the sweep line:

- Passing vertex: replace
- Local minimum cusp: delete
- Local maximum cusp: insert
- Initial vertex: insert
- Final vertex: delete



Monotone partition

Updating the sweep line:

- Passing vertex: replace
- Local minimum cusp: delete
- Local maximum cusp: insert
- Initial vertex: insert
- Final vertex: delete

In addition, eventually update the information of the upper vertex of the starting trapezoid.

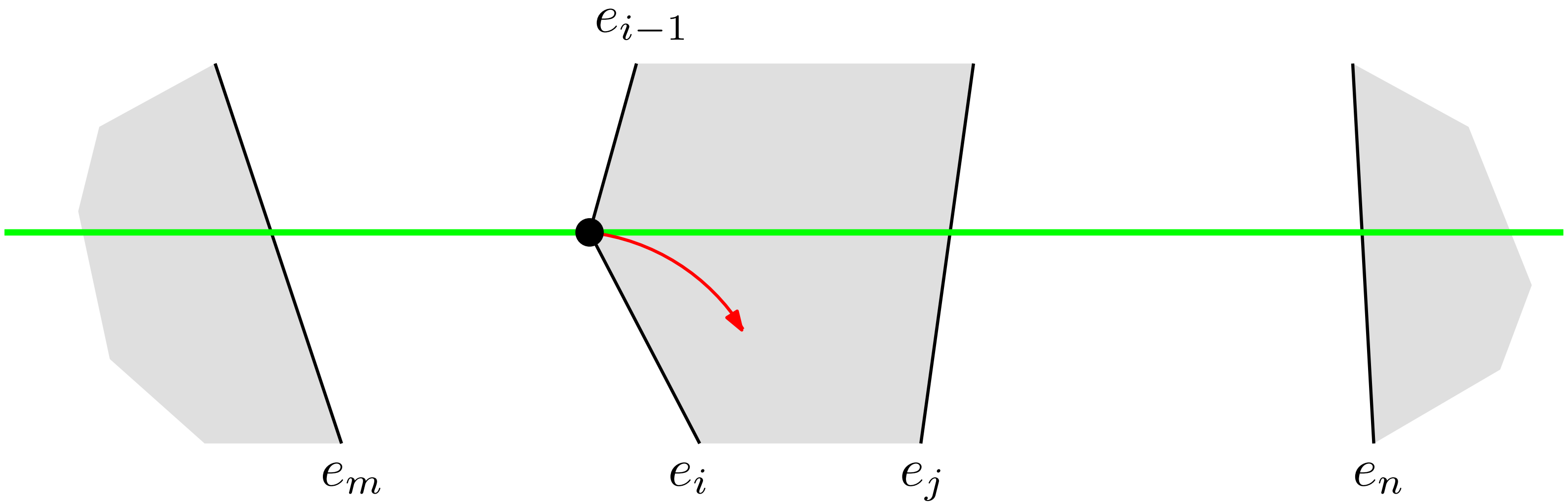
TRIANGULATING POLYGONS

Monotone partition

Updating the sweep line:

- Passing vertex: replace
- Local minimum cusp: delete
- Local maximum cusp: insert
- Initial vertex: insert
- Final vertex: delete

In addition, eventually update the information of the upper vertex of the starting trapezoid.



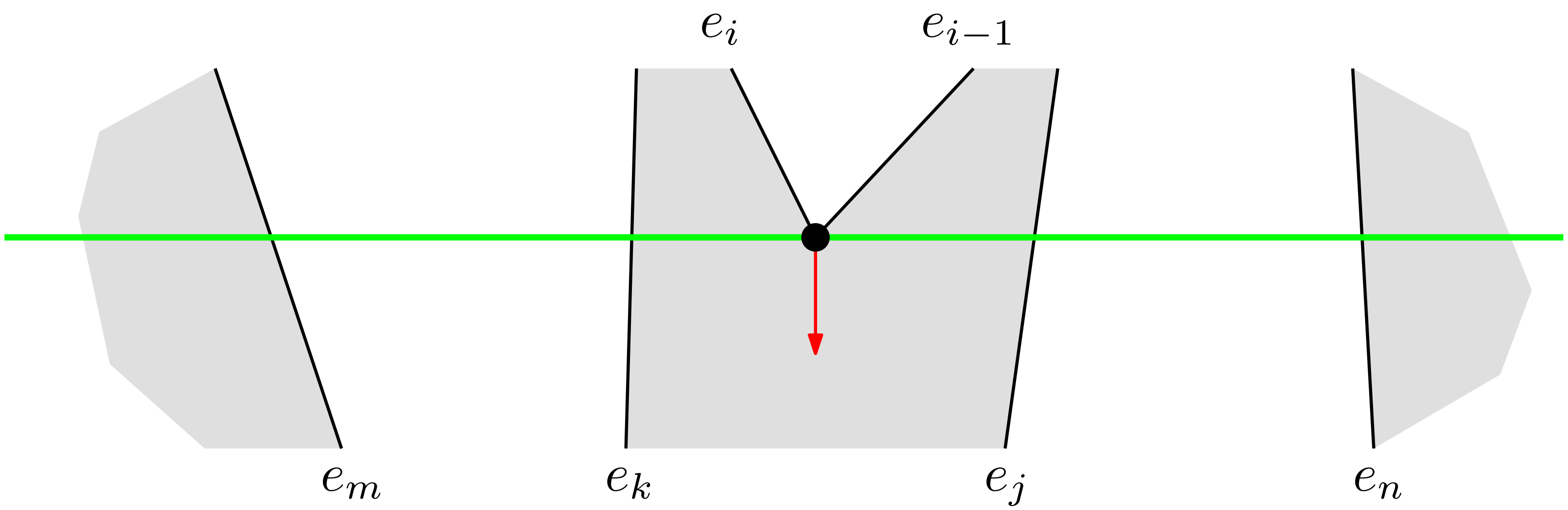
TRIANGULATING POLYGONS

Monotone partition

Updating the sweep line:

- Passing vertex: replace
- Local minimum cusp: delete
- Local maximum cusp: insert
- Initial vertex: insert
- Final vertex: delete

In addition, eventually update the information of the upper vertex of the starting trapezoid.



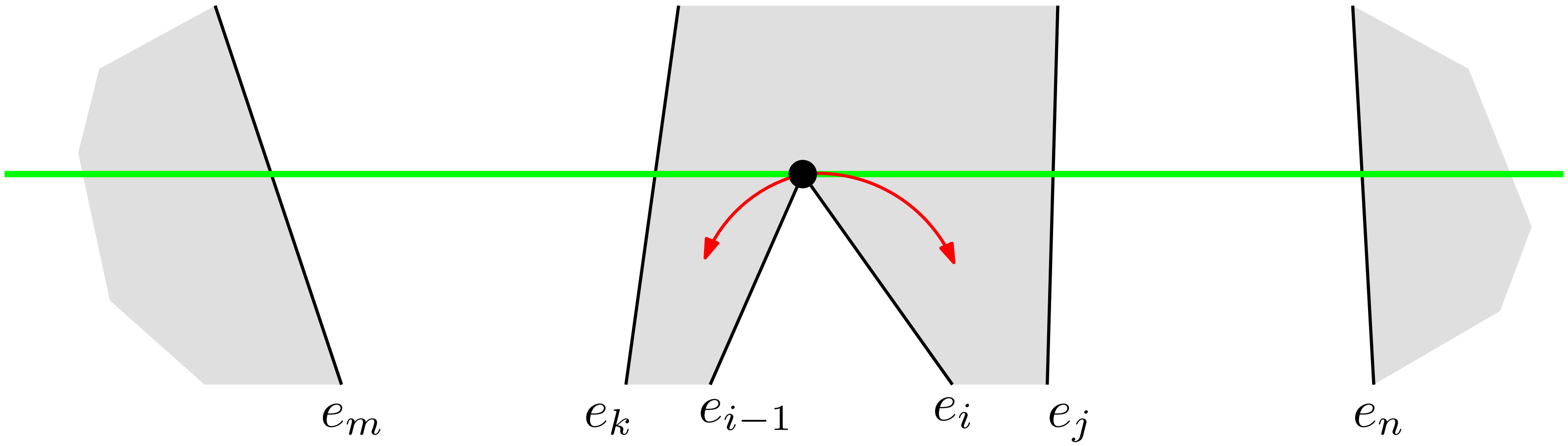
TRIANGULATING POLYGONS

Monotone partition

Updating the sweep line:

- Passing vertex: replace
- Local minimum cusp: delete
- Local maximum cusp: insert
- Initial vertex: insert
- Final vertex: delete

In addition, eventually update the information of the upper vertex of the starting trapezoid.



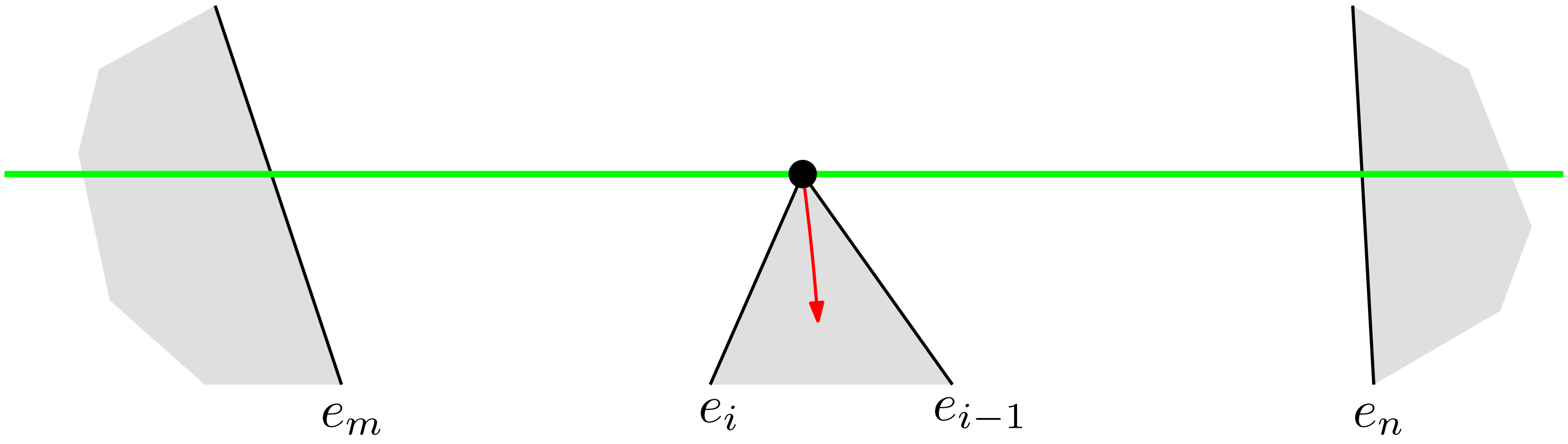
TRIANGULATING POLYGONS

Monotone partition

Updating the sweep line:

- Passing vertex: replace
- Local minimum cusp: delete
- Local maximum cusp: insert
- Initial vertex: insert
- Final vertex: delete

In addition, eventually update the information of the upper vertex of the starting trapezoid.



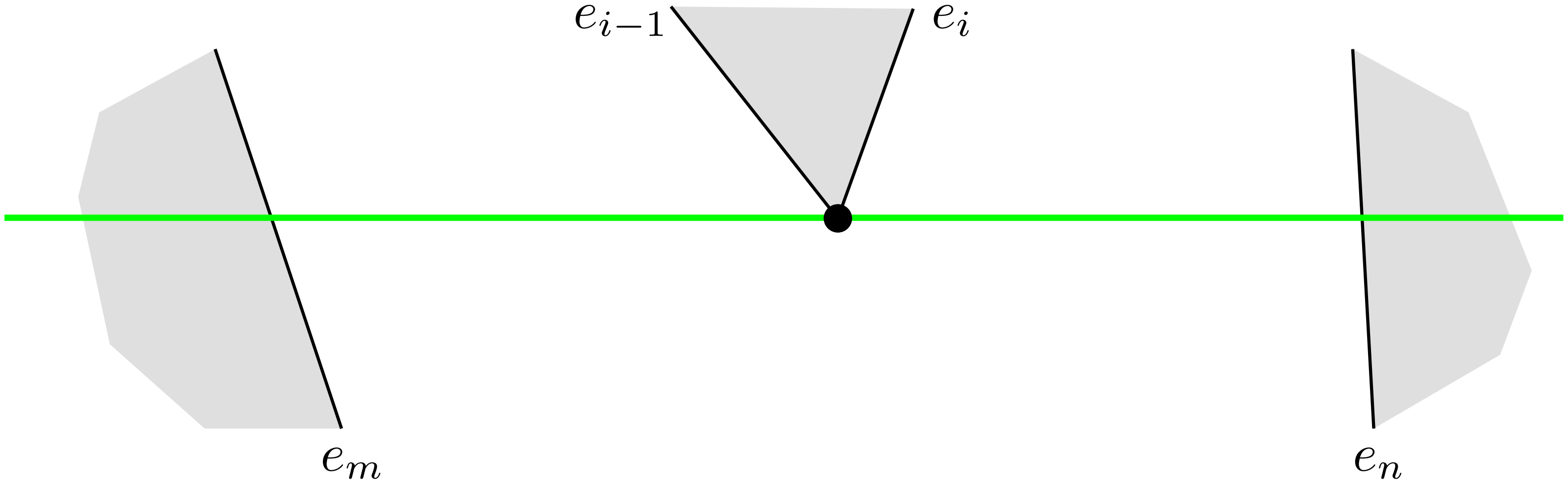
TRIANGULATING POLYGONS

Monotone partition

Updating the sweep line:

- Passing vertex: replace
- Local minimum cusp: delete
- Local maximum cusp: insert
- Initial vertex: insert
- Final vertex: delete

In addition, eventually update the information of the upper vertex of the starting trapezoid.



TRIANGULATING POLYGONS

Monotone partition

As for diagonals:

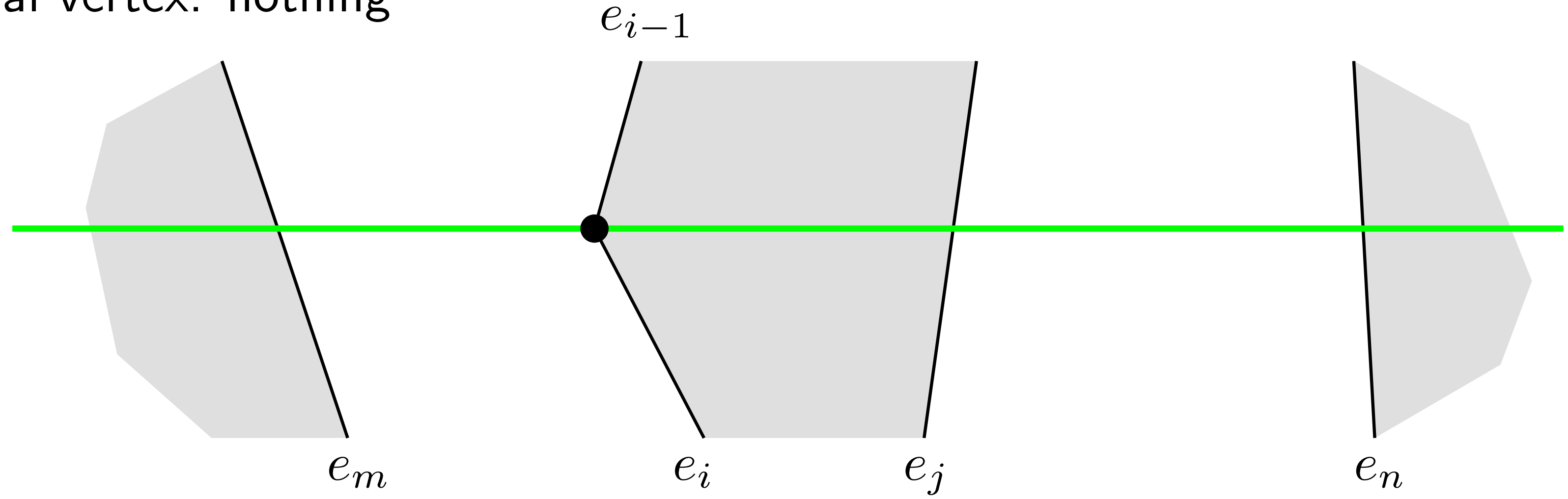
- Passing vertex: nothing
- Local minimum cusp: wait until it can be connect to the final vertex of the starting trapezoid
- Local maximum cusp: connect it with the initial vertex of the ending trapezoid
- Initial vertex: nothing
- Final vertex: nothing

TRIANGULATING POLYGONS

Monotone partition

As for diagonals:

- Passing vertex: nothing
- Local minimum cusp: wait until it can be connect to the final vertex of the starting trapezoid
- Local maximum cusp: connect it with the initial vertex of the ending trapezoid
- Initial vertex: nothing
- Final vertex: nothing

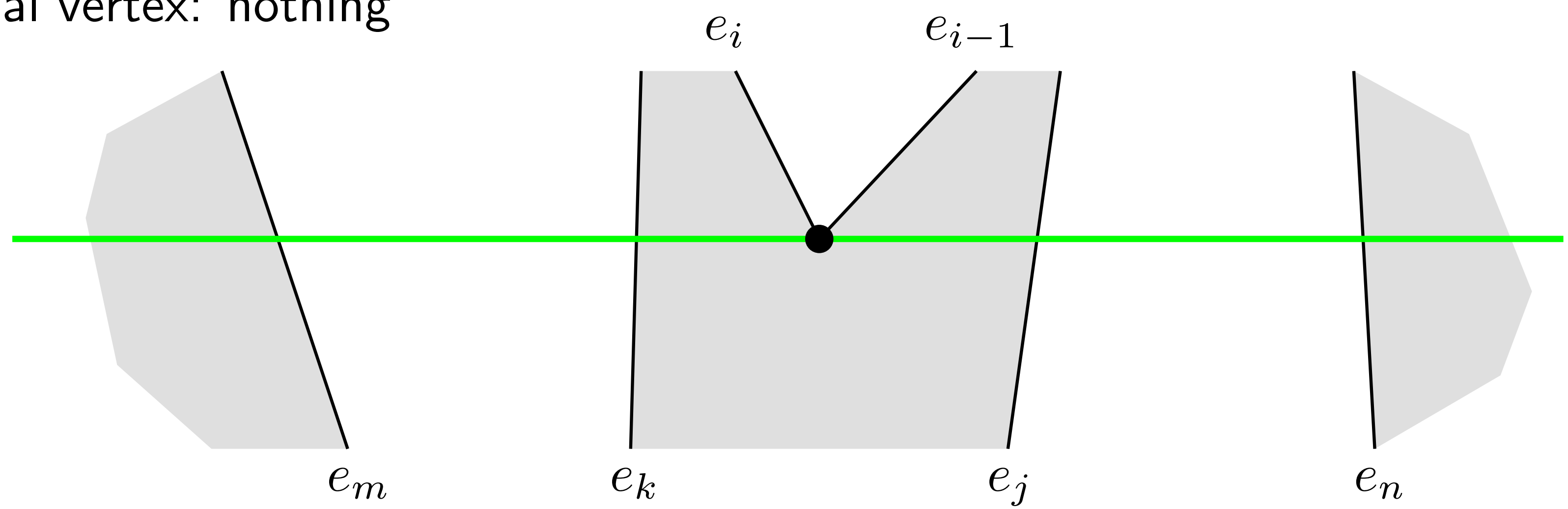


TRIANGULATING POLYGONS

Monotone partition

As for diagonals:

- Passing vertex: nothing
- Local minimum cusp: wait until it can be connect to the final vertex of the starting trapezoid
- Local maximum cusp: connect it with the initial vertex of the ending trapezoid
- Initial vertex: nothing
- Final vertex: nothing

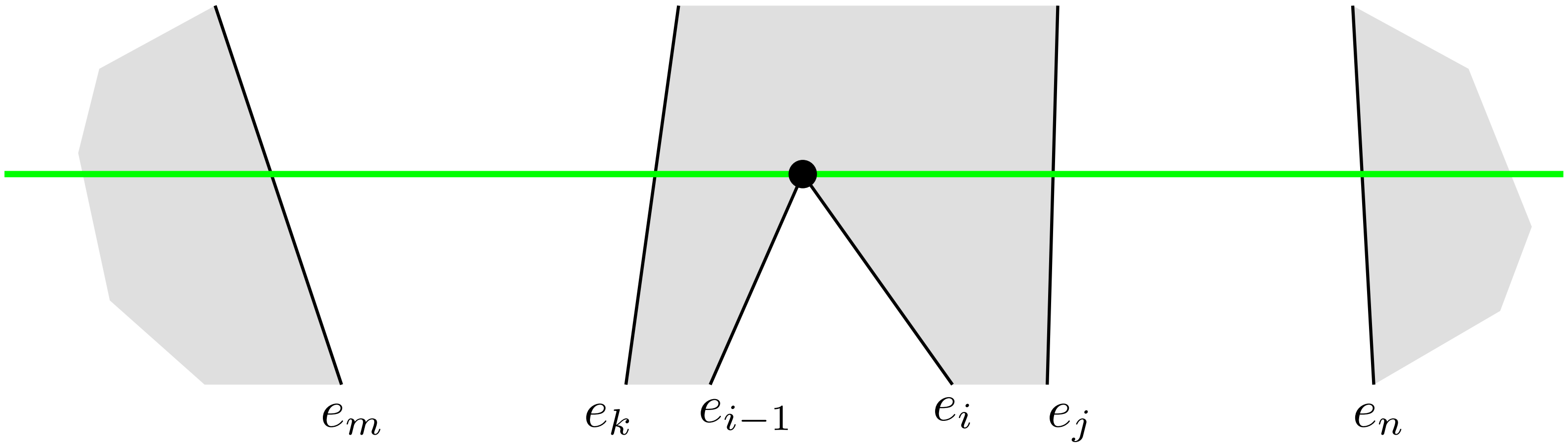


TRIANGULATING POLYGONS

Monotone partition

As for diagonals:

- Passing vertex: nothing
- Local minimum cusp: wait until it can be connect to the final vertex of the starting trapezoid
- Local maximum cusp: connect it with the initial vertex of the ending trapezoid
- Initial vertex: nothing
- Final vertex: nothing

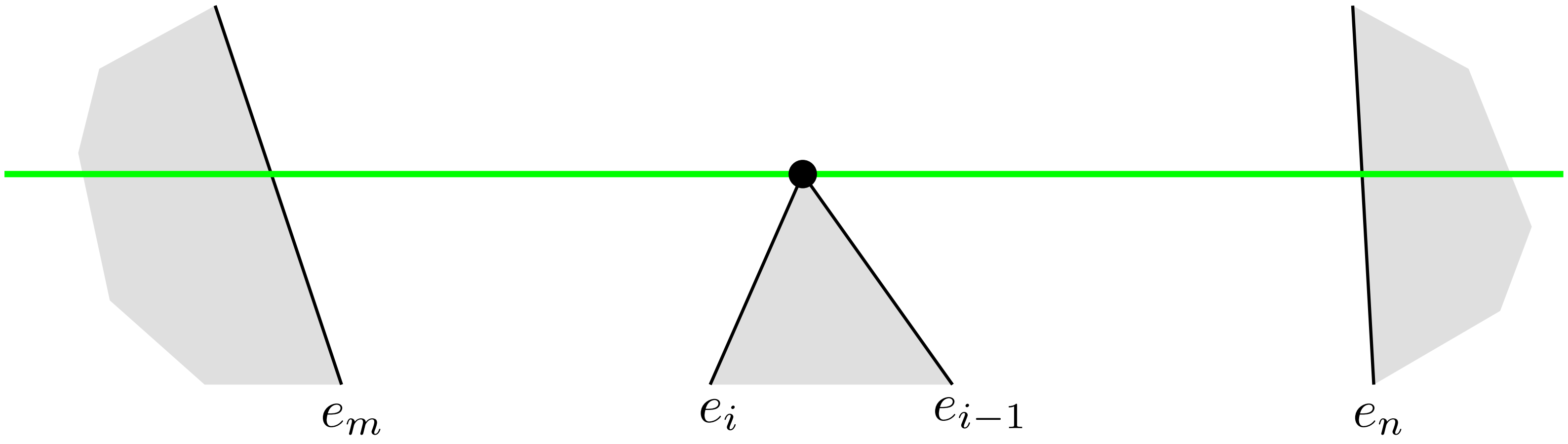


TRIANGULATING POLYGONS

Monotone partition

As for diagonals:

- Passing vertex: nothing
- Local minimum cusp: wait until it can be connect to the final vertex of the starting trapezoid
- Local maximum cusp: connect it with the initial vertex of the ending trapezoid
- Initial vertex: nothing
- Final vertex: nothing

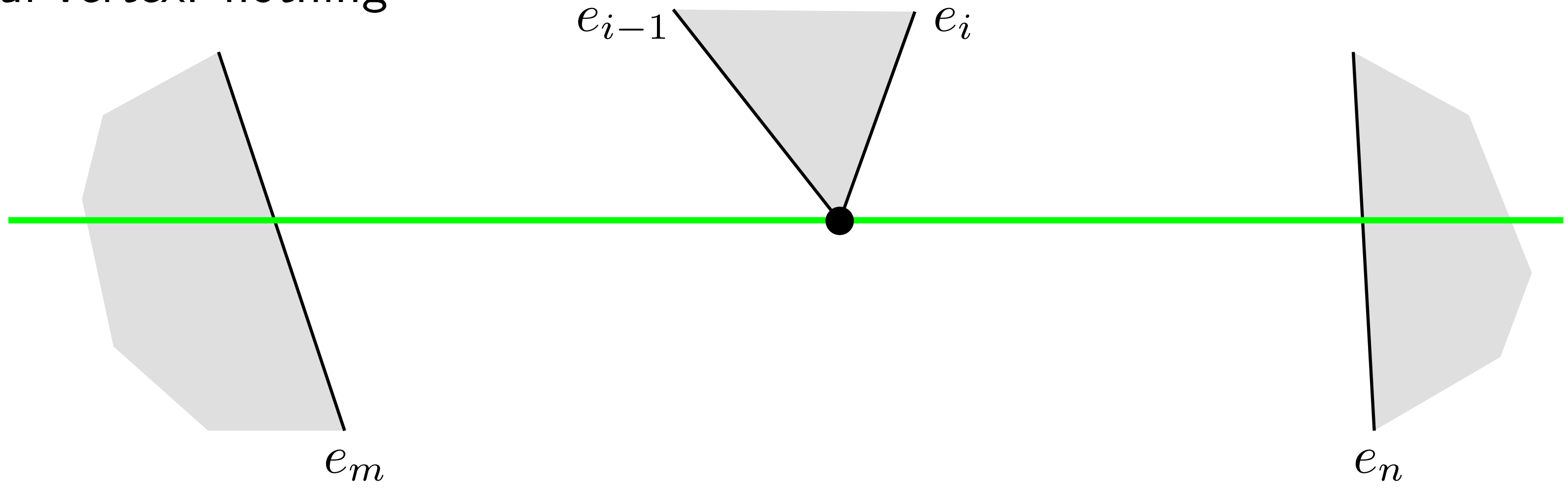


TRIANGULATING POLYGONS

Monotone partition

As for diagonals:

- Passing vertex: nothing
- Local minimum cusp: wait until it can be connect to the final vertex of the starting trapezoid
- Local maximum cusp: connect it with the initial vertex of the ending trapezoid
- Initial vertex: nothing
- Final vertex: nothing



TRIANGULATING POLYGONS

Monotone partition

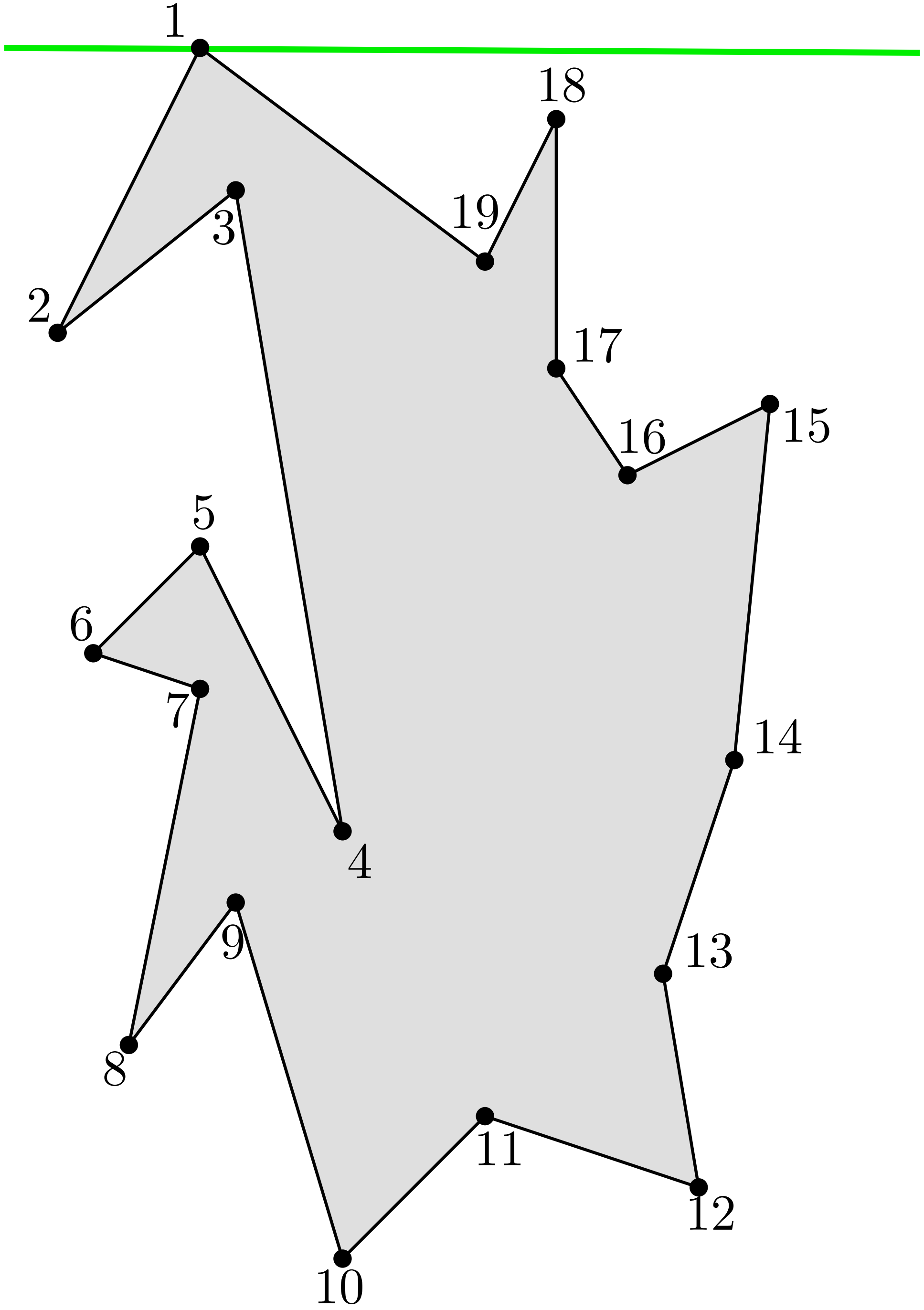
vertex

1

sweep line

$v1$

$e1, e19$



TRIANGULATING POLYGONS

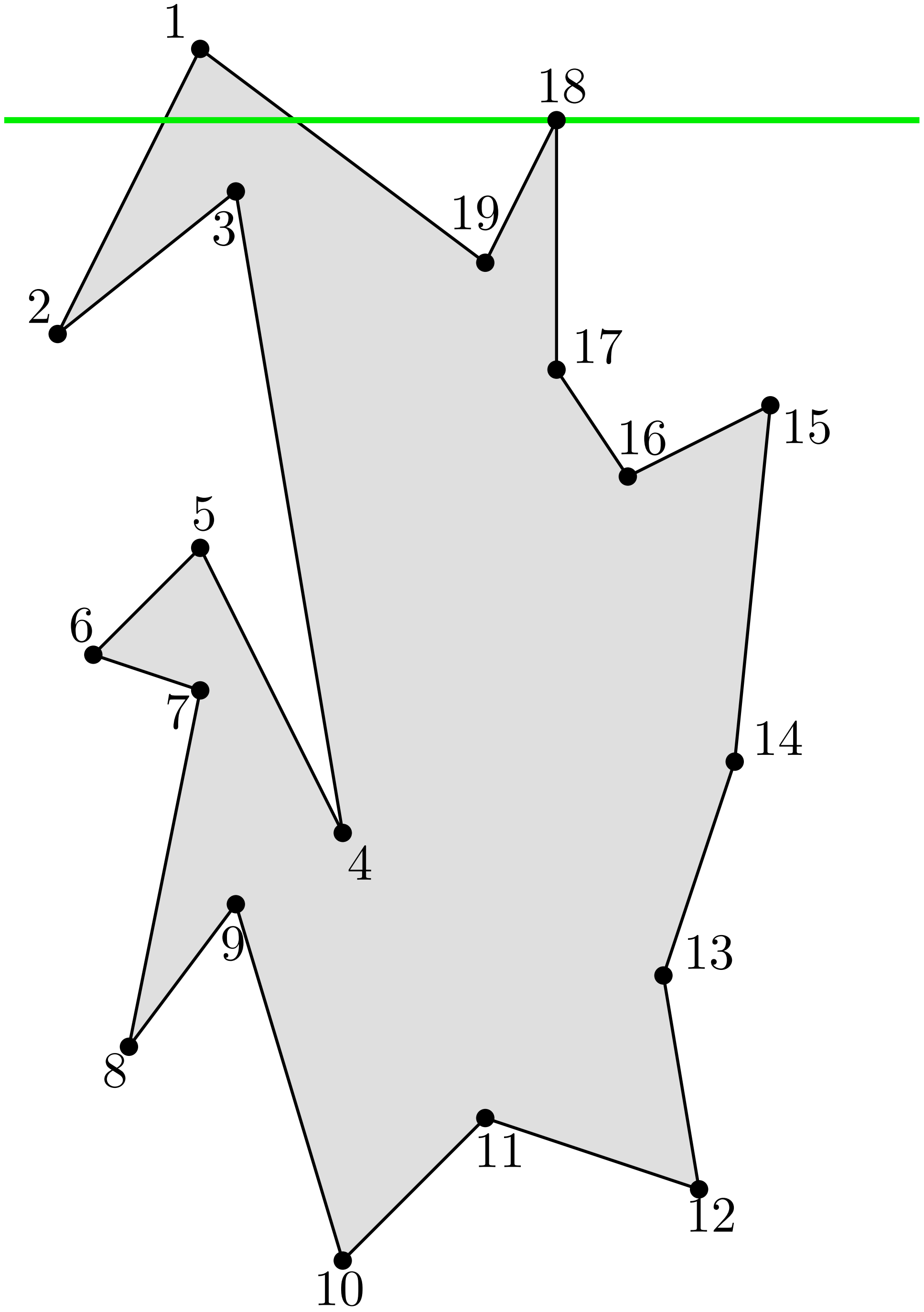
Monotone partition

vertex

18

sweep line

$\frac{v1}{e1, e19}$ $\frac{v18}{e18, e17}$

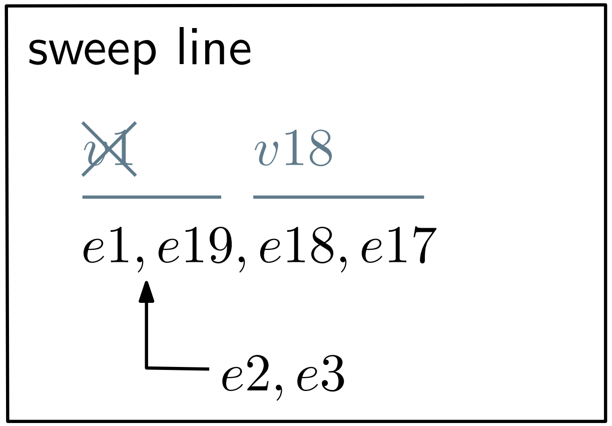


TRIANGULATING POLYGONS

Monotone partition

vertex

3

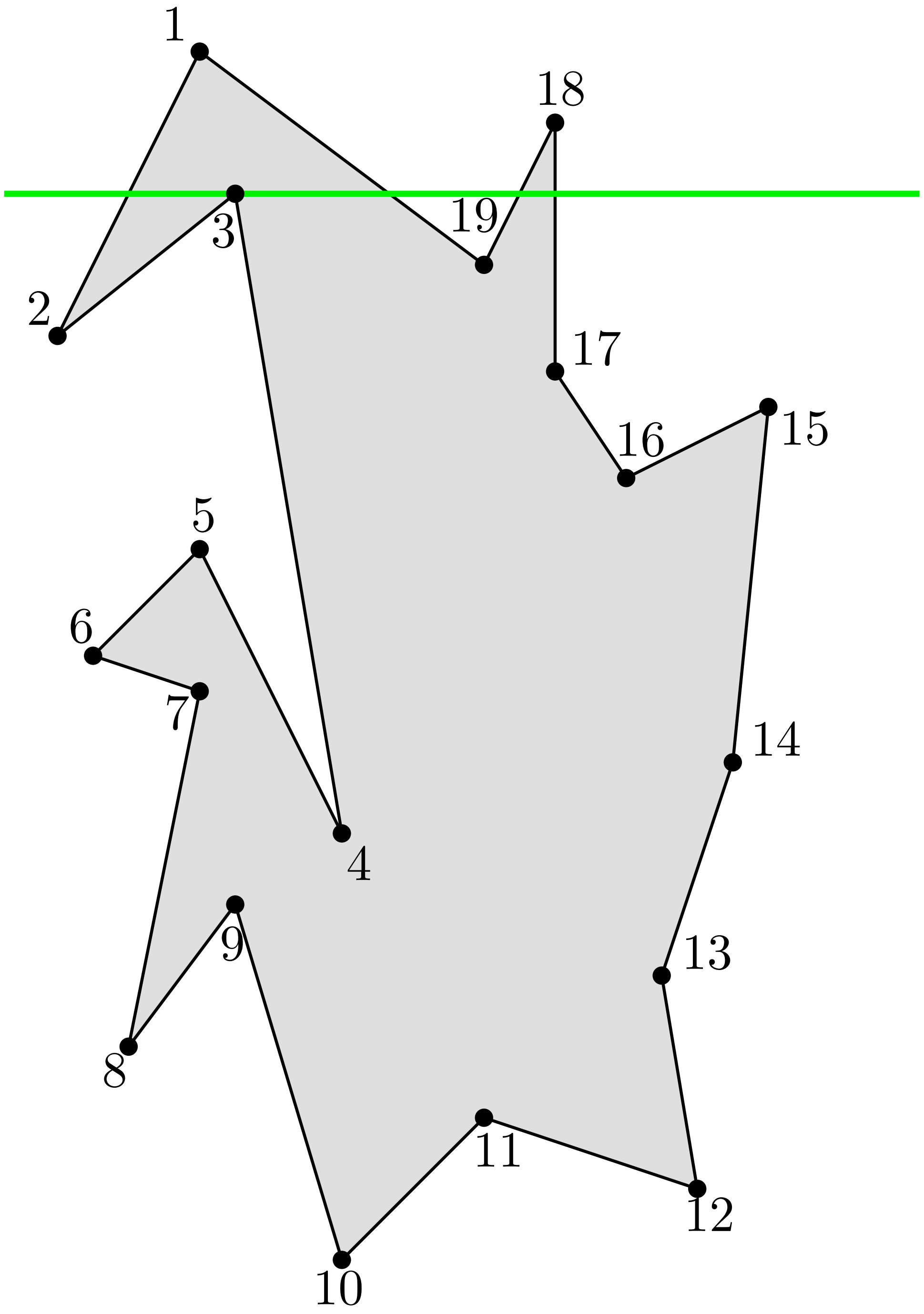


sweep line

~~$v1$~~ $v18$

$e1, e19, e18, e17$

$e2, e3$

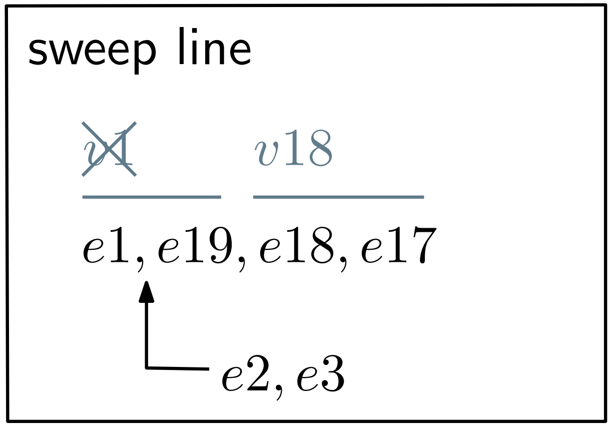


TRIANGULATING POLYGONS

Monotone partition

vertex

3



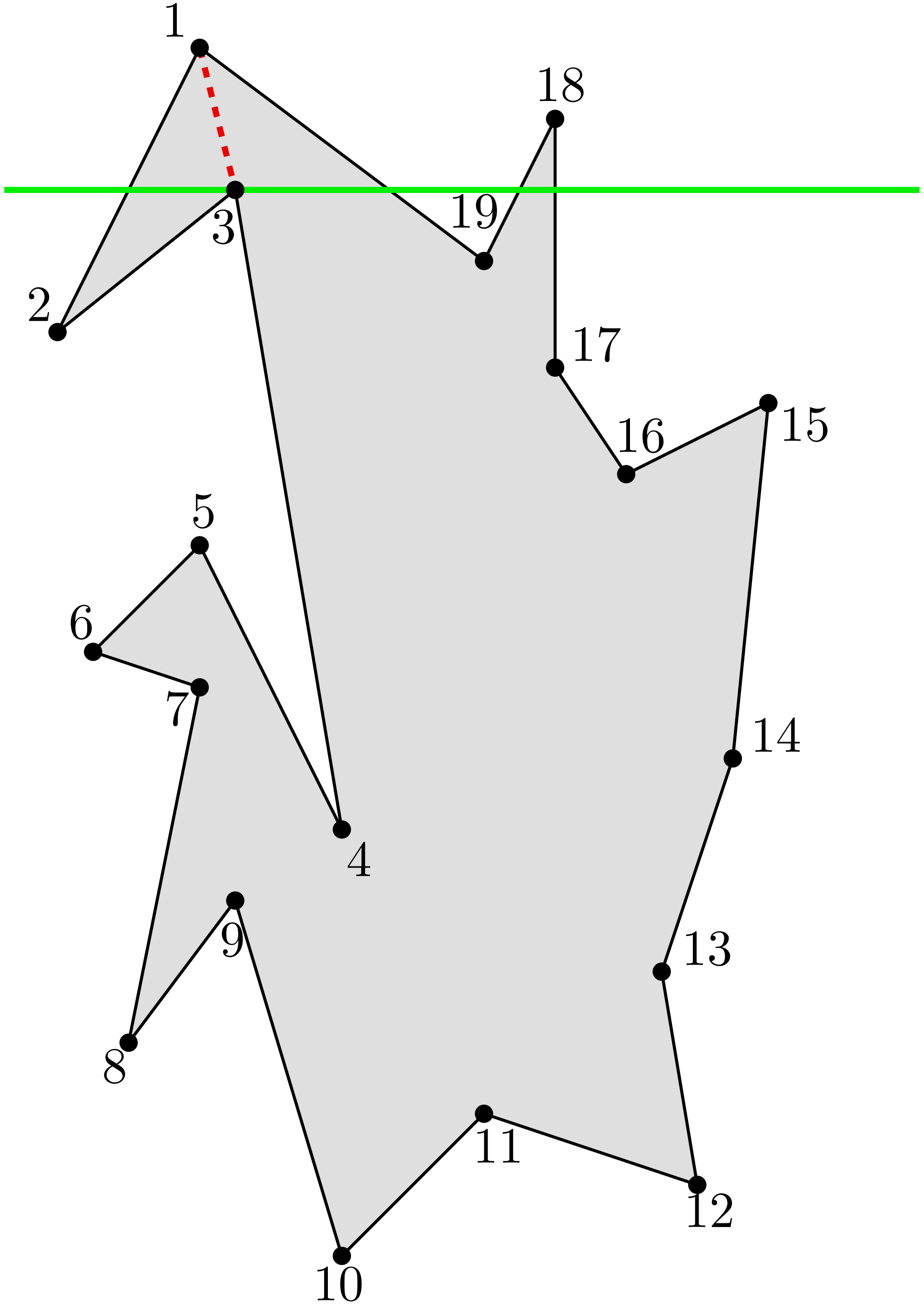
sweep line

$\cancel{v1}$ $v18$

$e1, e19, e18, e17$

$e2, e3$

Diagonal $v1 - v3$



TRIANGULATING POLYGONS

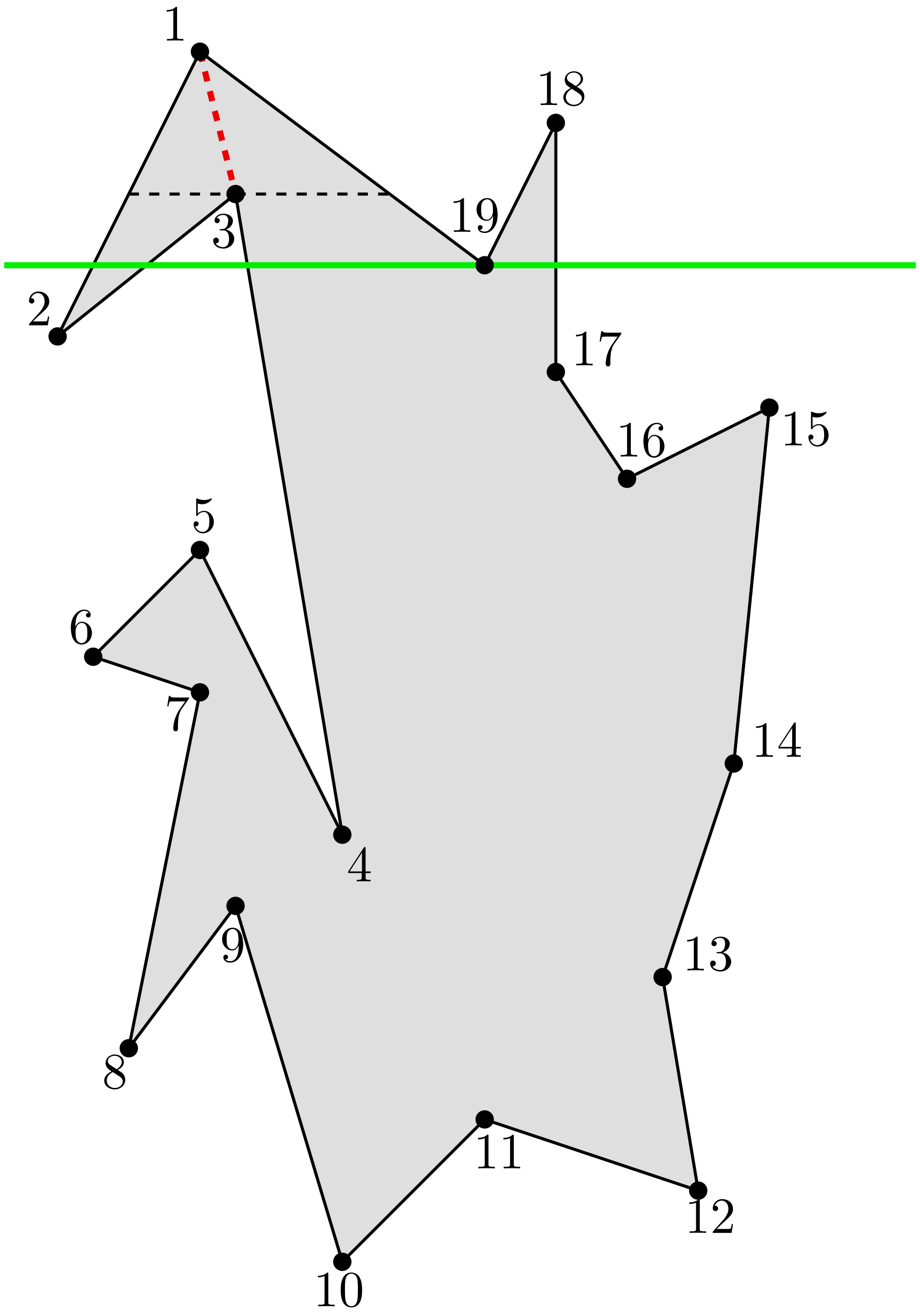
Monotone partition

vertex

19

sweep line

$$\overset{v3}{\overline{e1, e2}}, \overset{\cancel{v3}}{\overline{e3, \cancel{e19}}}, \overset{\cancel{v18}}{\overline{\cancel{e18}, e17}}$$



TRIANGULATING POLYGONS

Monotone partition

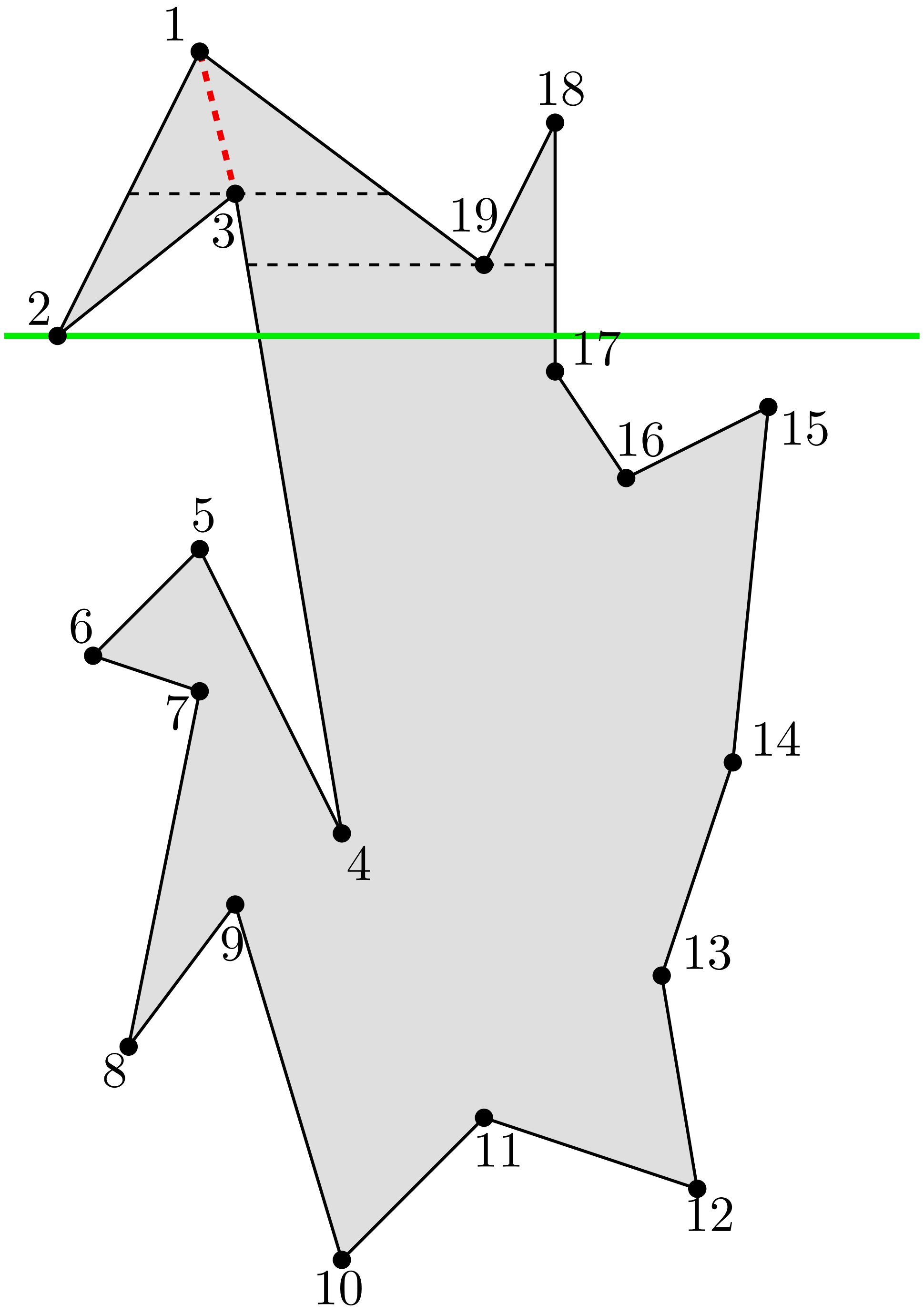
vertex

2

sweep line

$v3$ $v19$

$\cancel{e1}, \cancel{e2}, e3, e17$



TRIANGULATING POLYGONS

Monotone partition

vertex

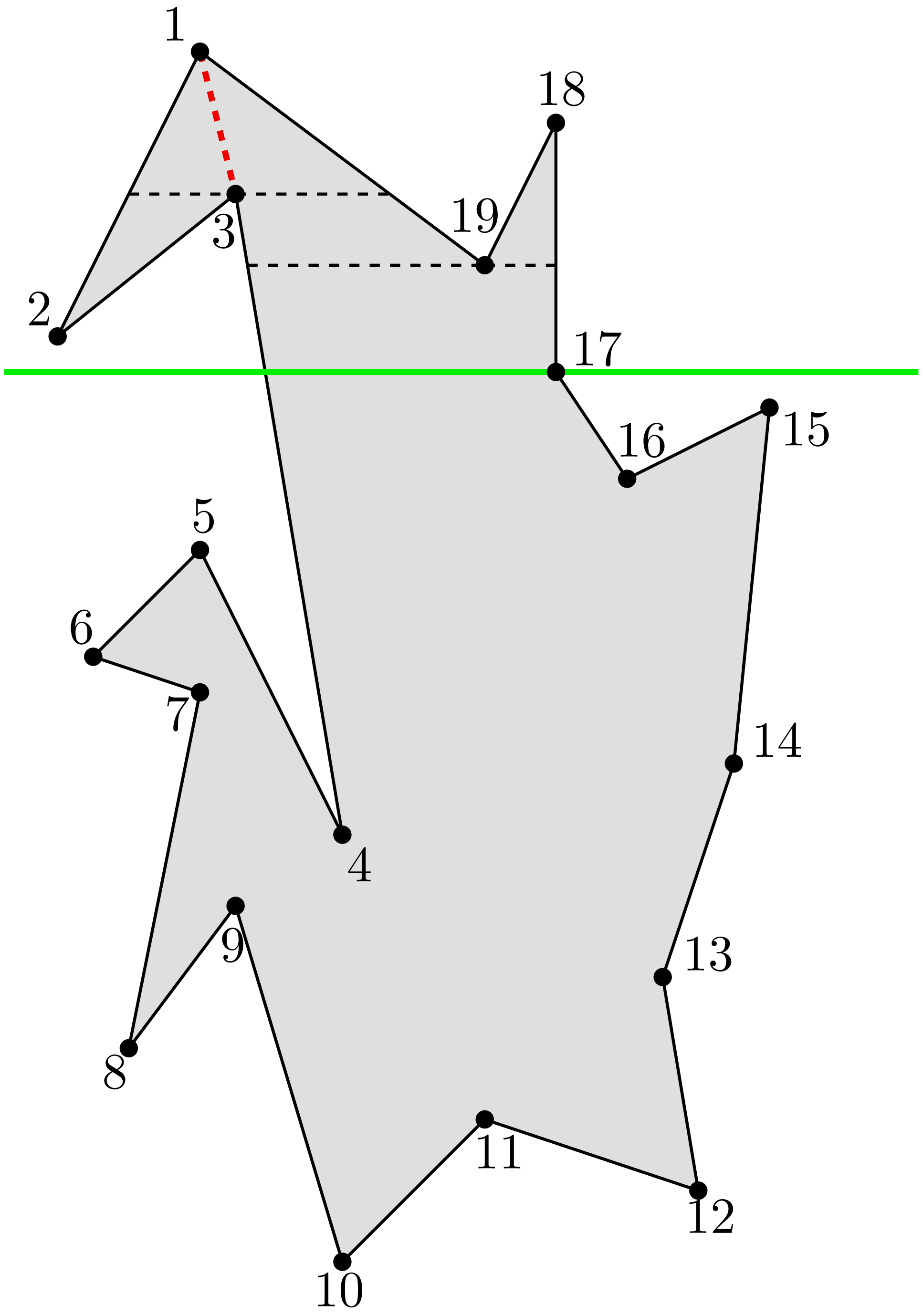
17

sweep line

$\cancel{v19}$

$e3, \cancel{e17}$

$e16$



TRIANGULATING POLYGONS

Monotone partition

vertex

17

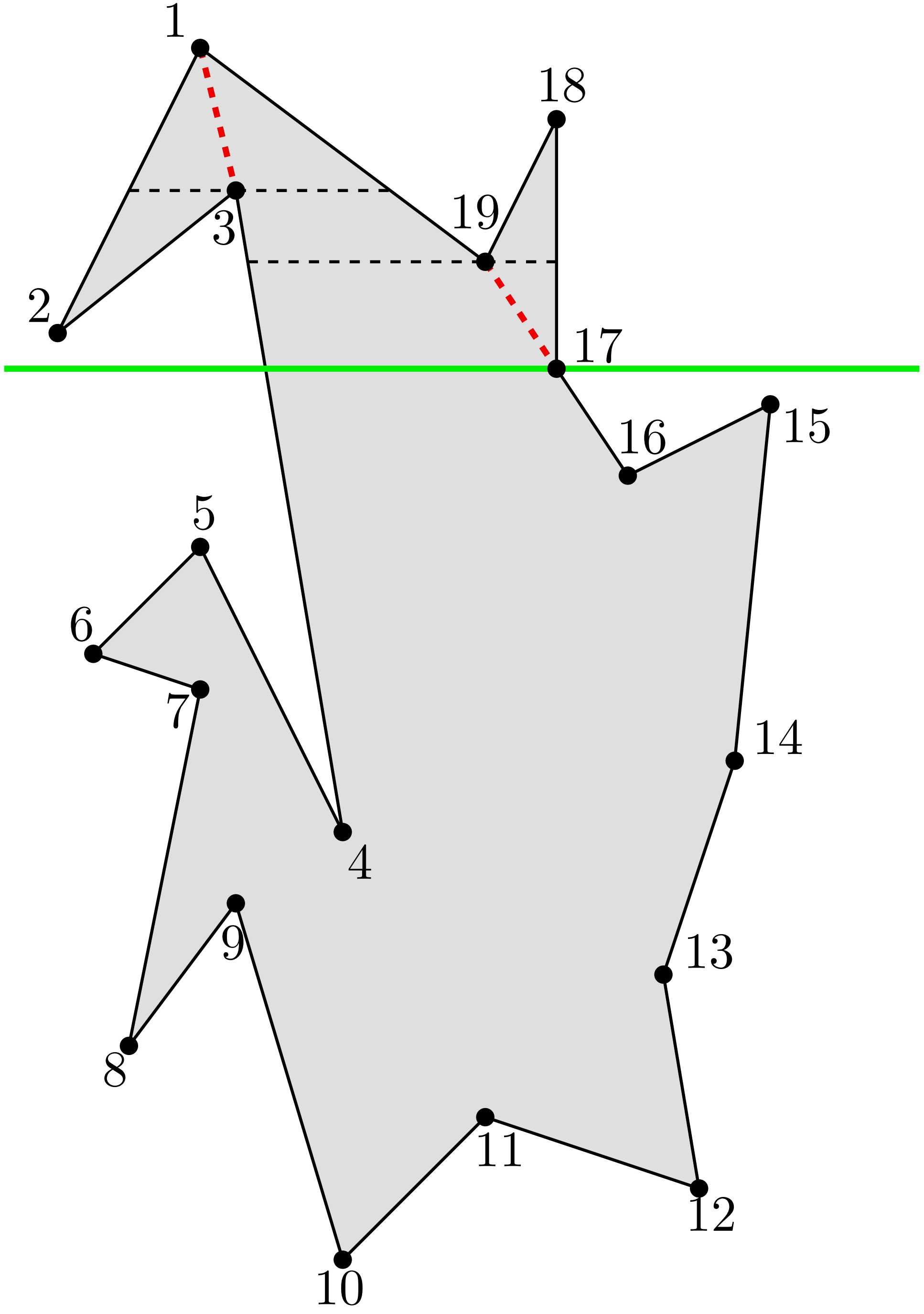
sweep line

$\cancel{v19}$

$e3, e\cancel{17}$

$\uparrow$ $e16$

Diagonal $v19 - v17$



TRIANGULATING POLYGONS

Monotone partition

vertex

15

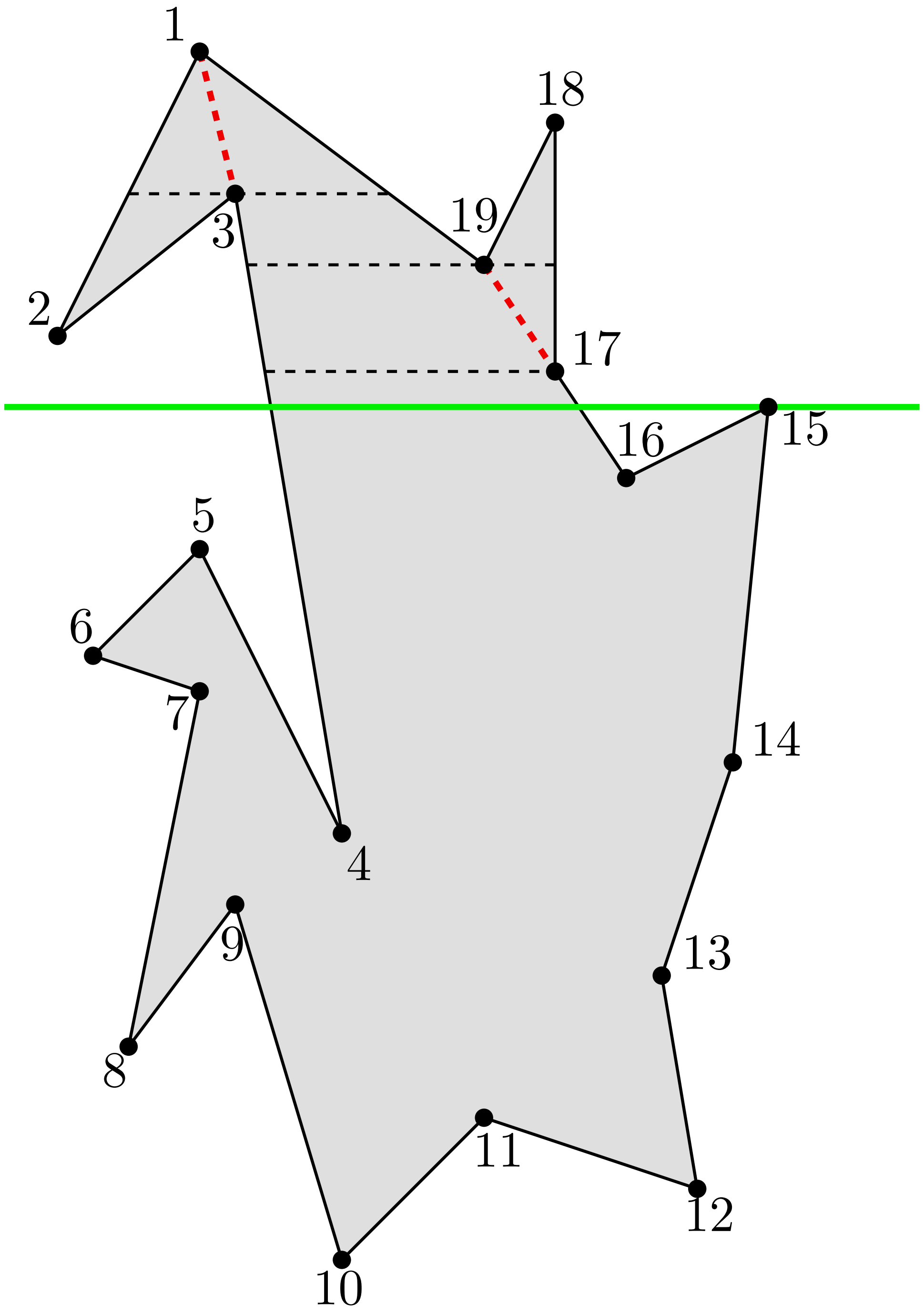
sweep line

$v17$

$e3, e16$

$v15$

$e15, e14$



TRIANGULATING POLYGONS

Monotone partition

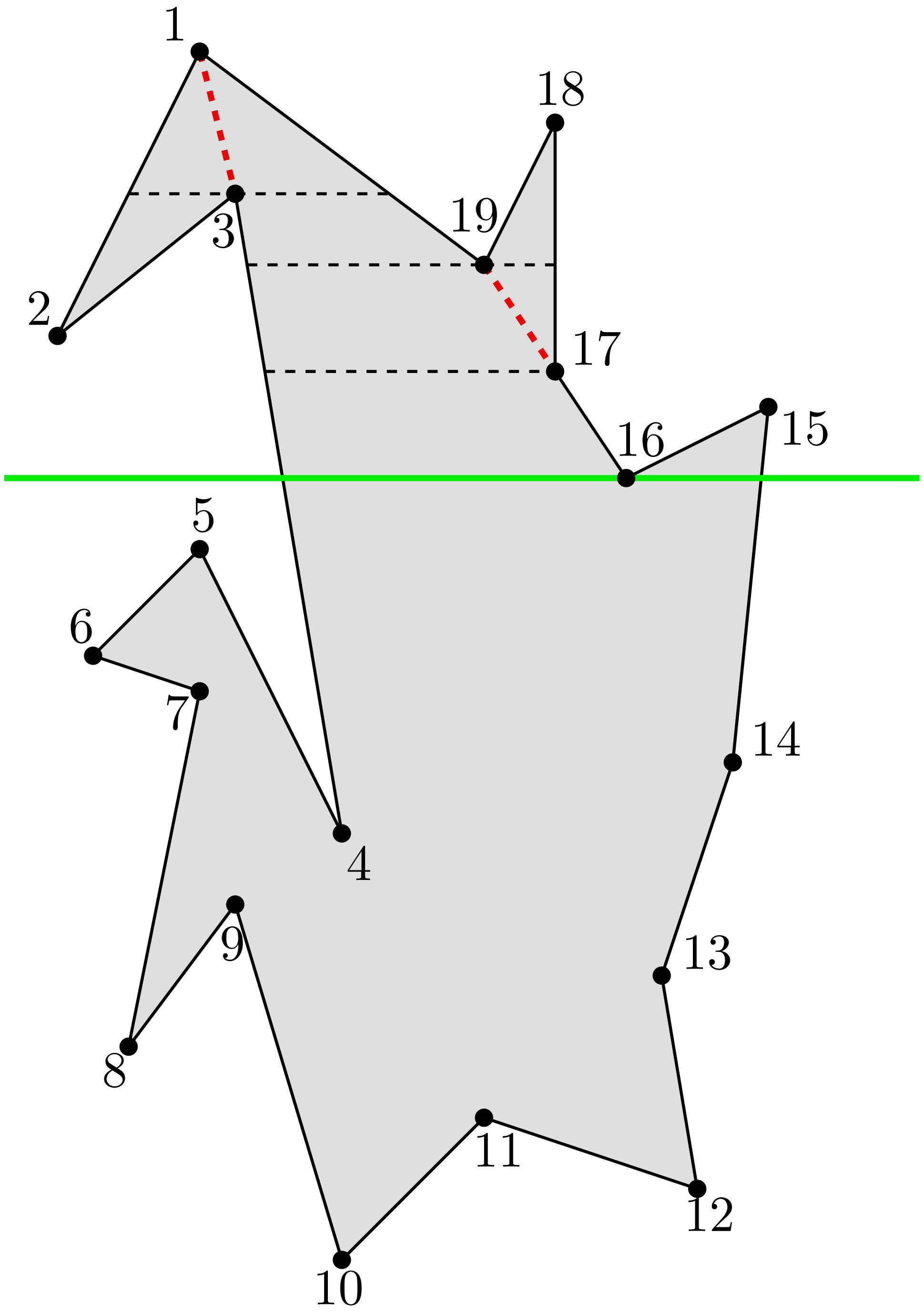
vertex

16

sweep line

$\xcancel{v17}$ $\xcancel{v15}$

$e3, \xcancel{e16}, \xcancel{e15}, e14$



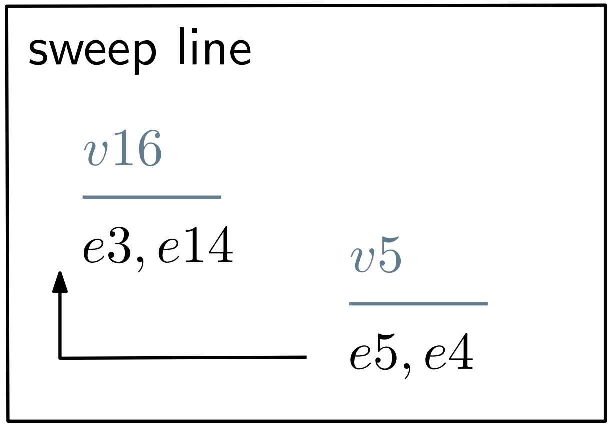
TRIANGULATING POLYGONS

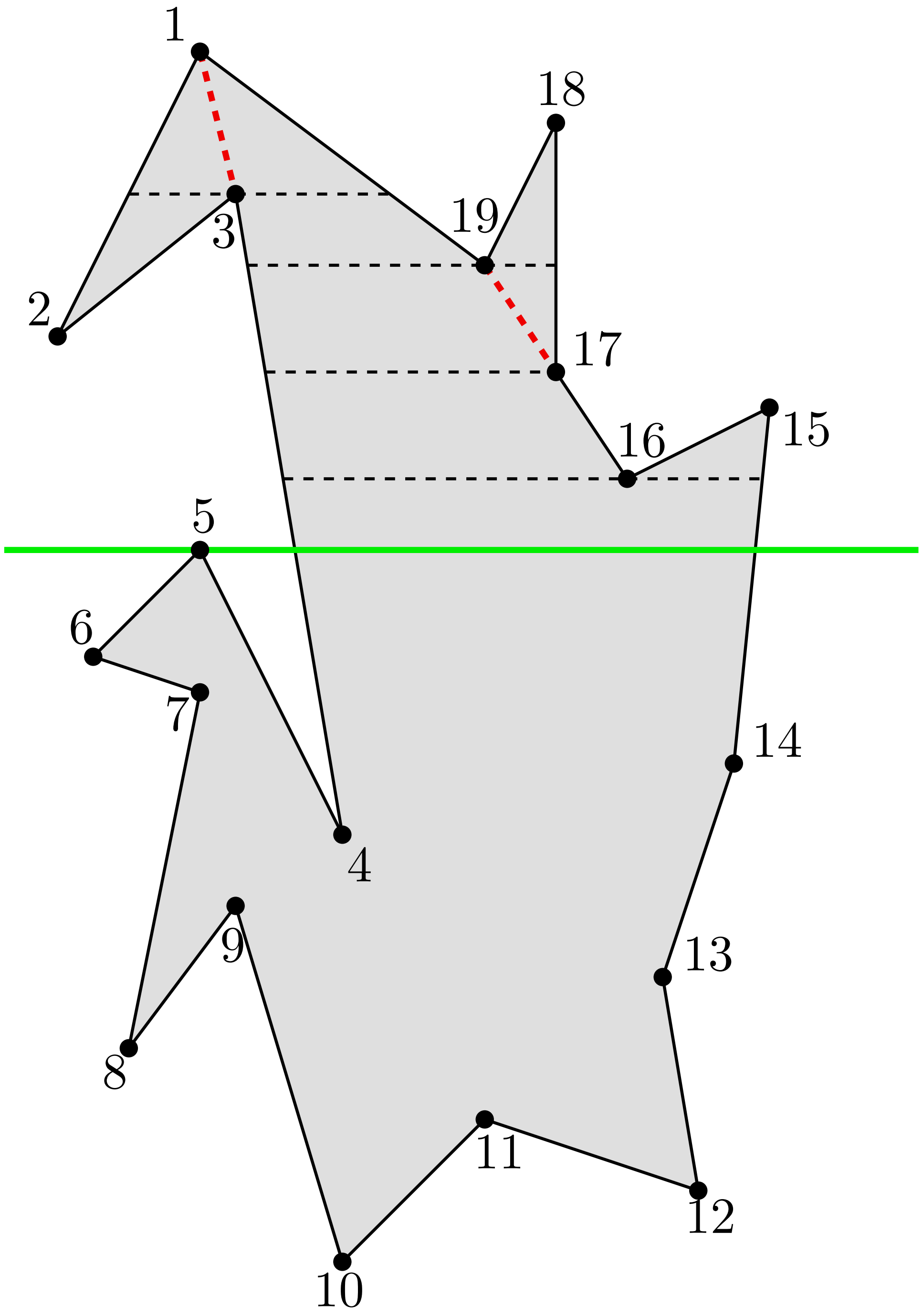
Monotone partition

vertex

5

sweep line





TRIANGULATING POLYGONS

Monotone partition

vertex

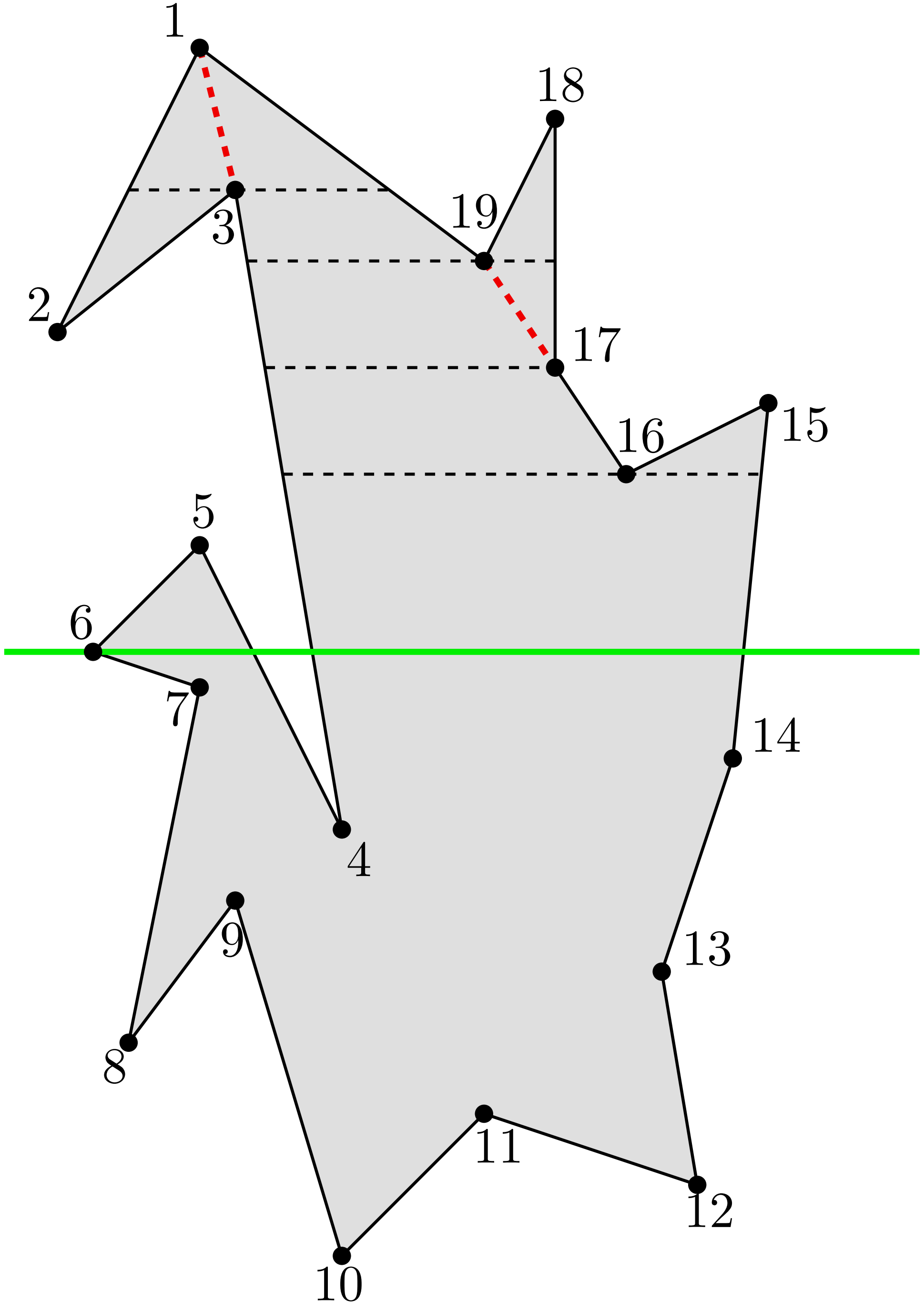
6

sweep line

$\cancel{v5}$ $v16$

$\cancel{e5}, e4, e3, e14$

$e6$



TRIANGULATING POLYGONS

Monotone partition

vertex

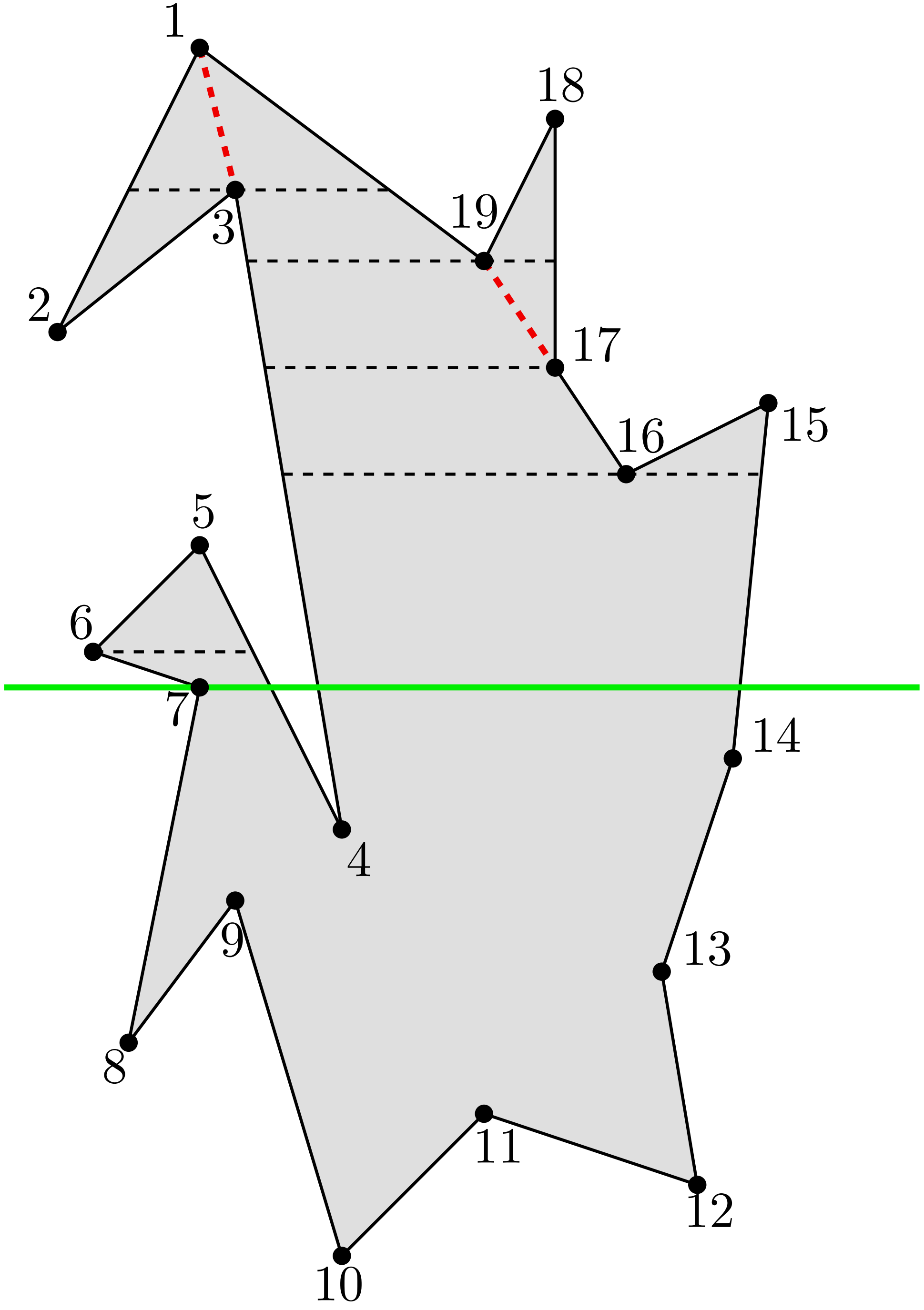
7

sweep line

$\cancel{v6}$ $v16$

$\cancel{e6}, e4, e3, e14$

$e7$



TRIANGULATING POLYGONS

Monotone partition

vertex

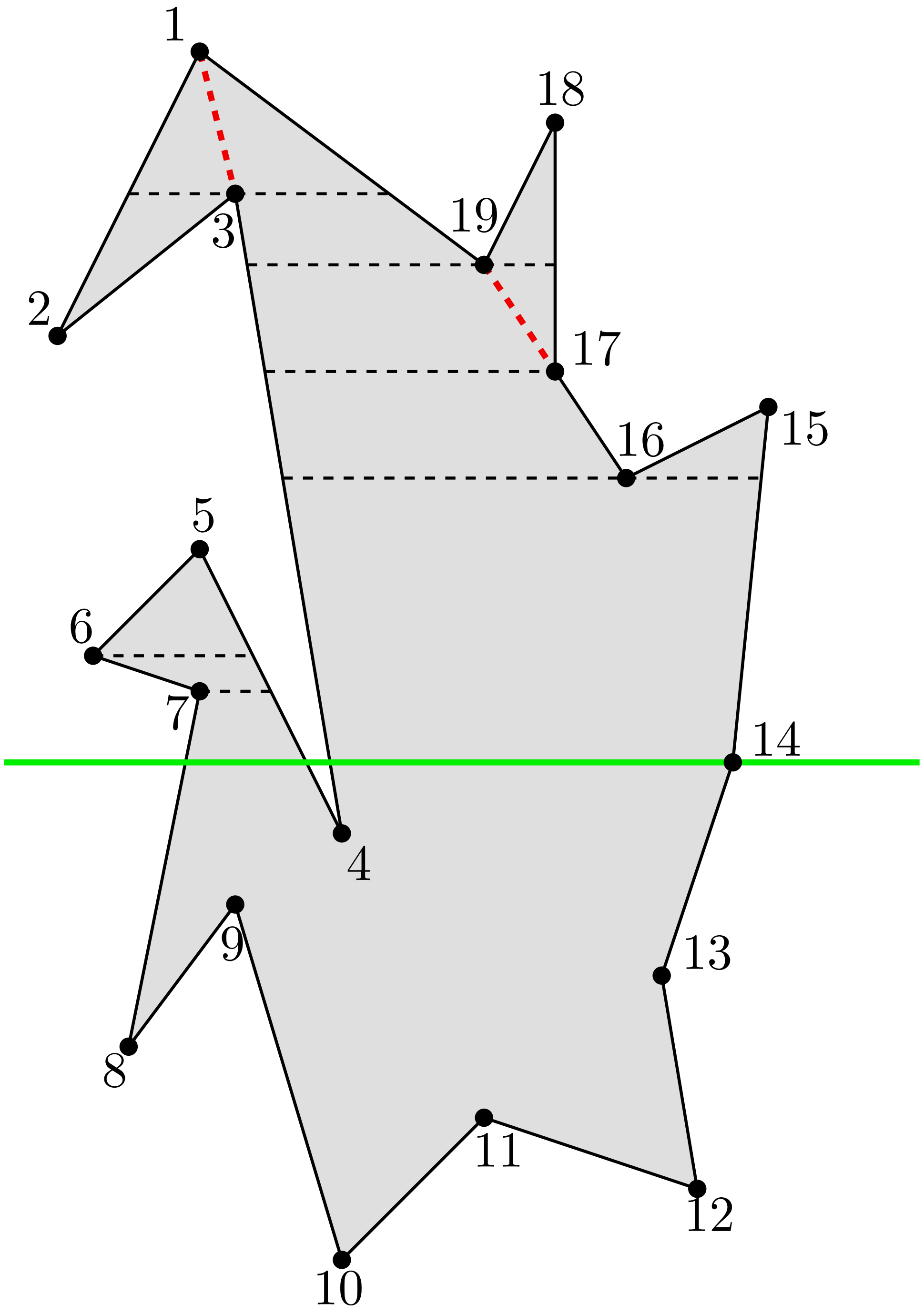
14

sweep line

$v7$ ~~$v16$~~

$e7, e4, e3,$ ~~$e14$~~

↑ $e13$



TRIANGULATING POLYGONS

Monotone partition

vertex

14

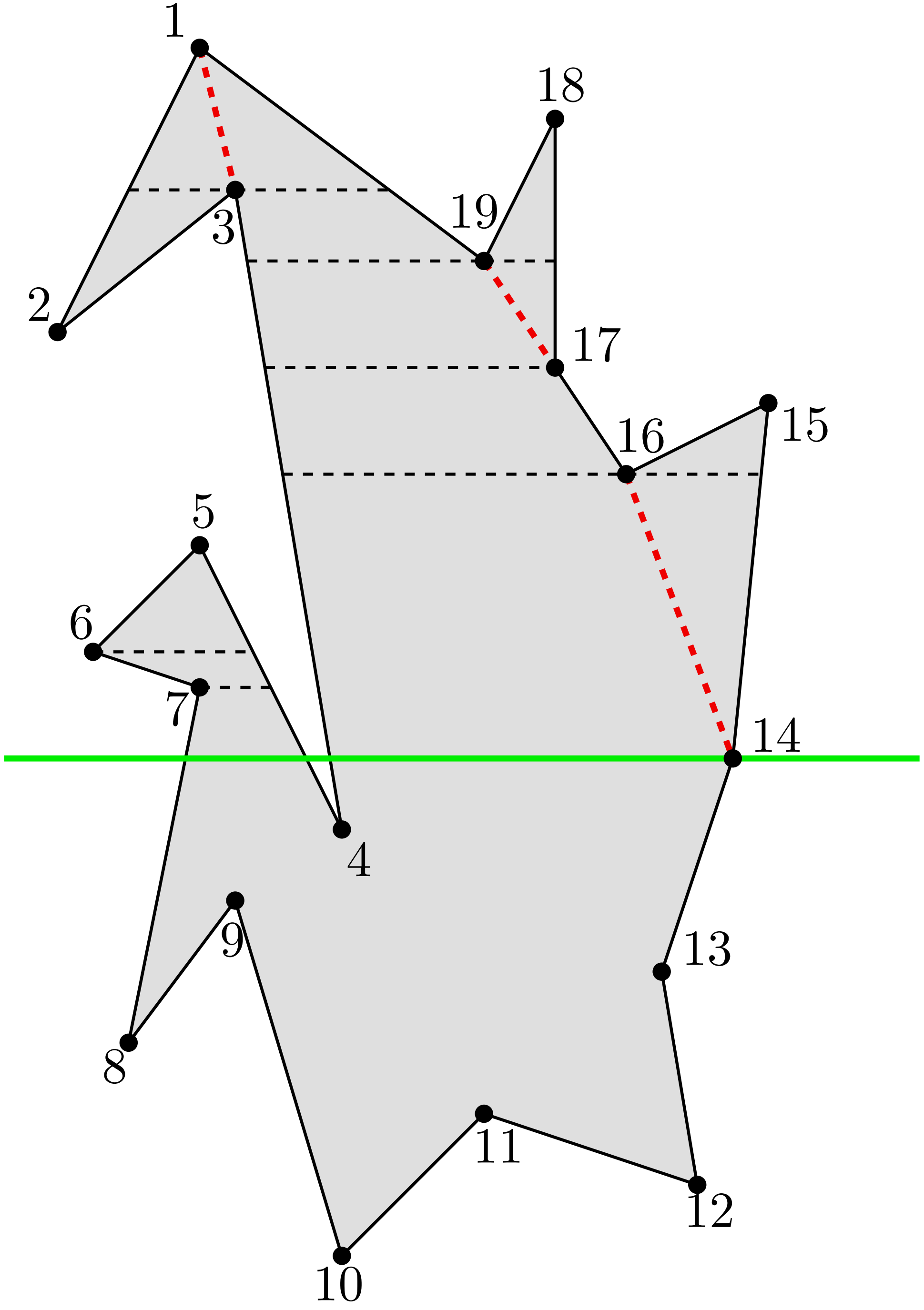
sweep line

$v7$ $v\cancel{1}6$

$e7, e4, e3, e\cancel{1}4$

↑ $e13$

Diagonal $v16 - v14$



TRIANGULATING POLYGONS

Monotone partition

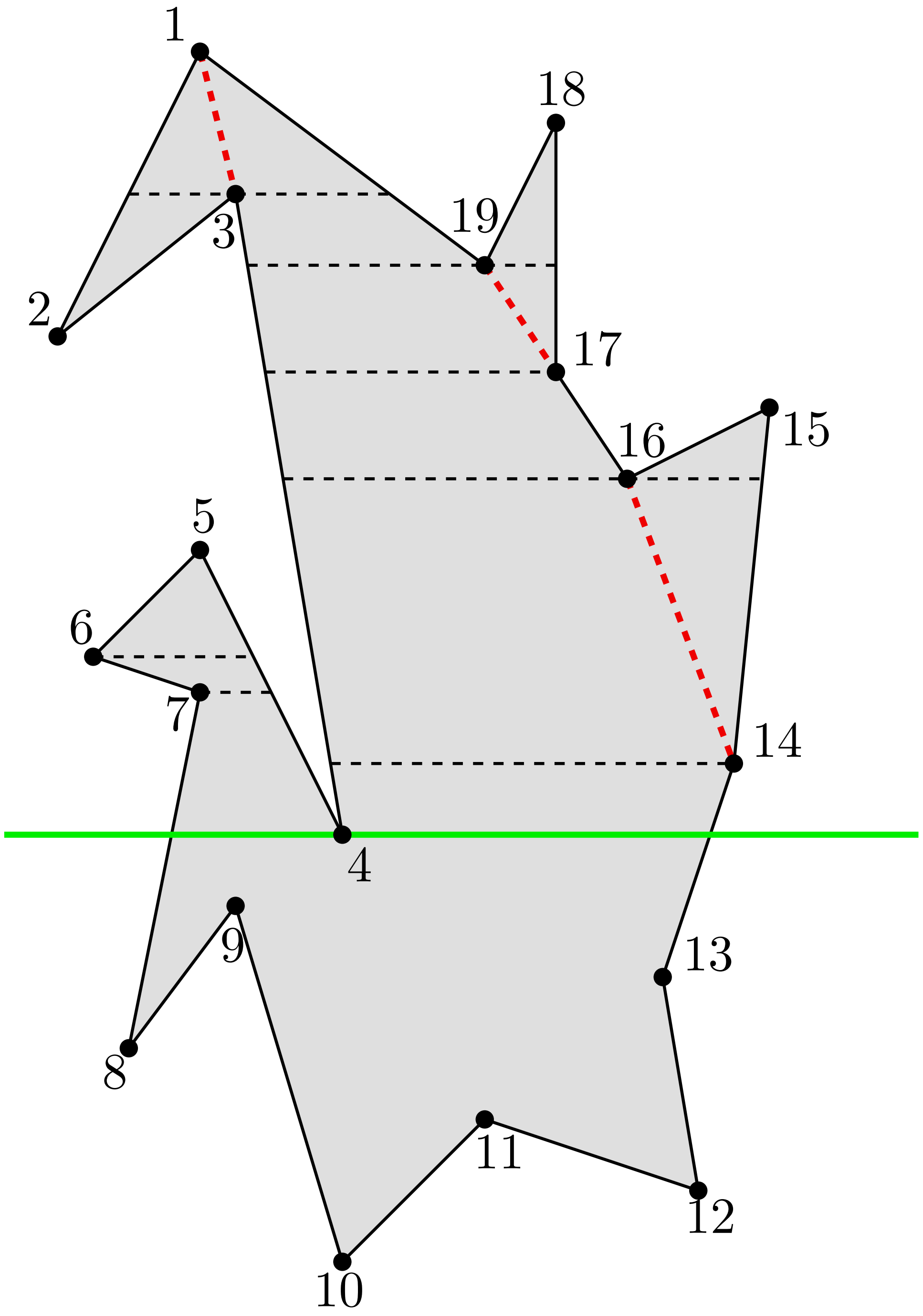
vertex

4

sweep line

$\cancel{v7}$ $\cancel{v1}4$

$e7, \cancel{e4}, \cancel{e3}, e13$



TRIANGULATING POLYGONS

Monotone partition

vertex

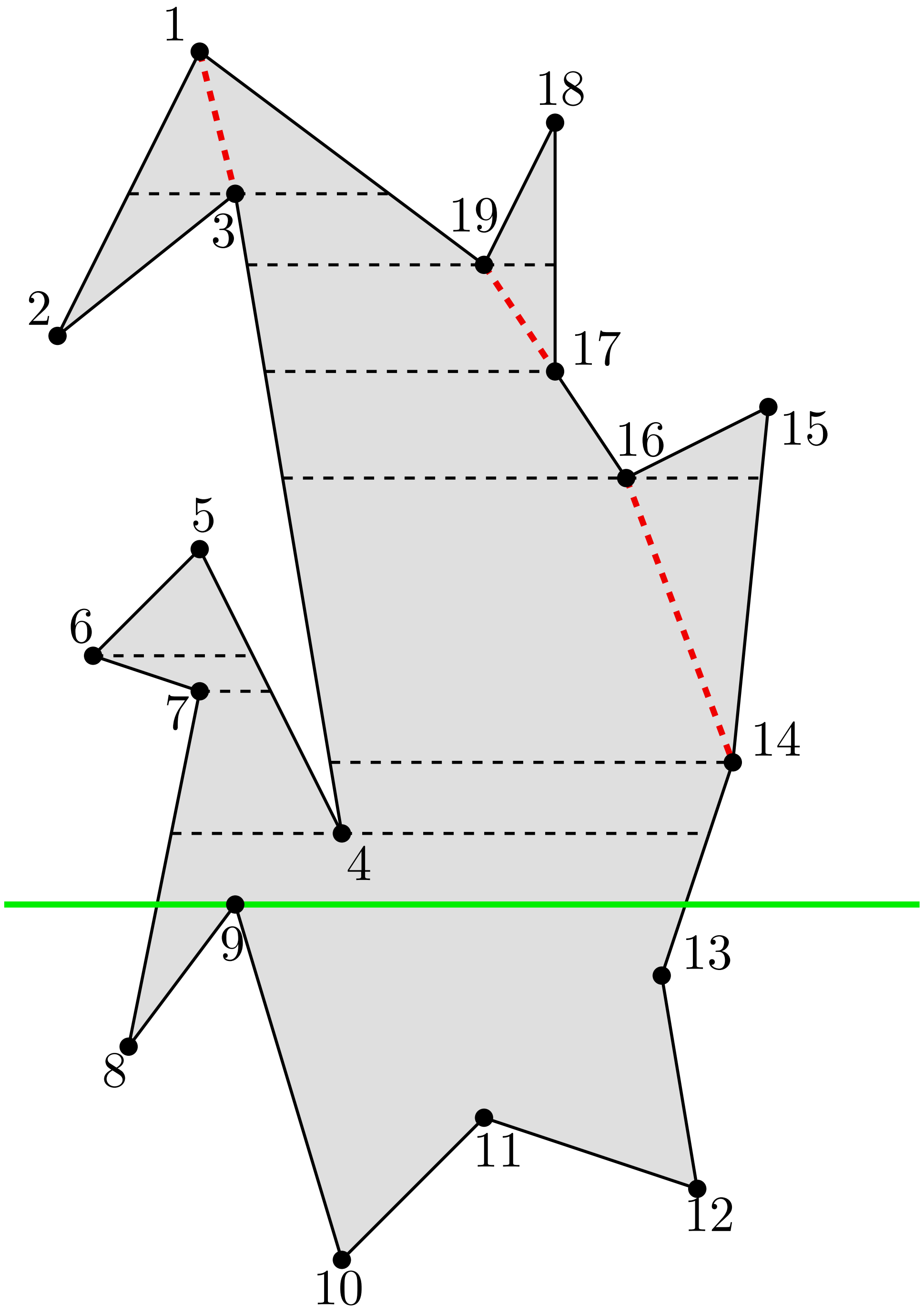
9

sweep line

~~$v4$~~

$e7, e13$

$e8, e9$



TRIANGULATING POLYGONS

Monotone partition

vertex

9

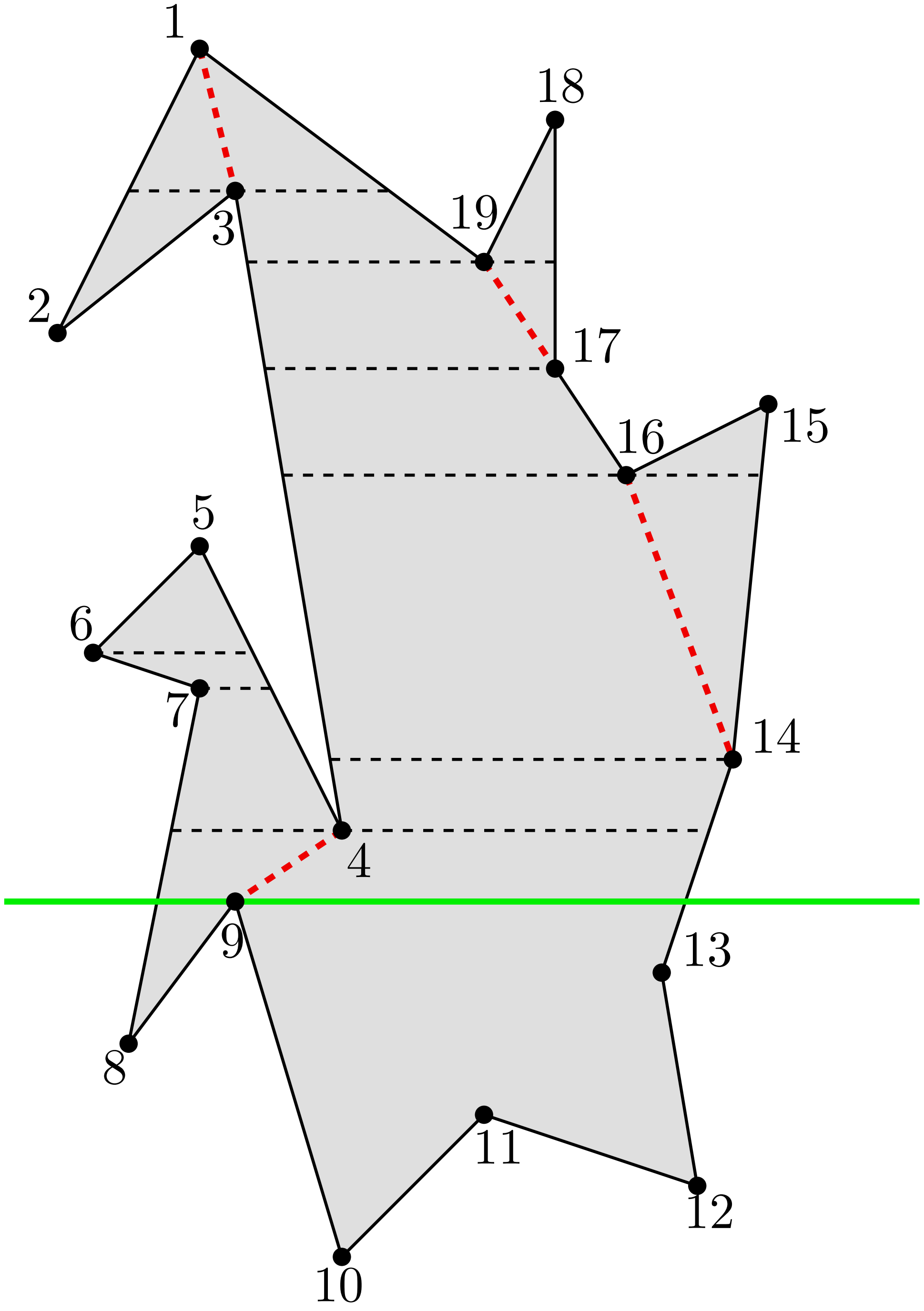
sweep line

~~$v4$~~

$e7, e13$

$e8, e9$

Diagonal $v4 - v9$



TRIANGULATING POLYGONS

Monotone partition

vertex

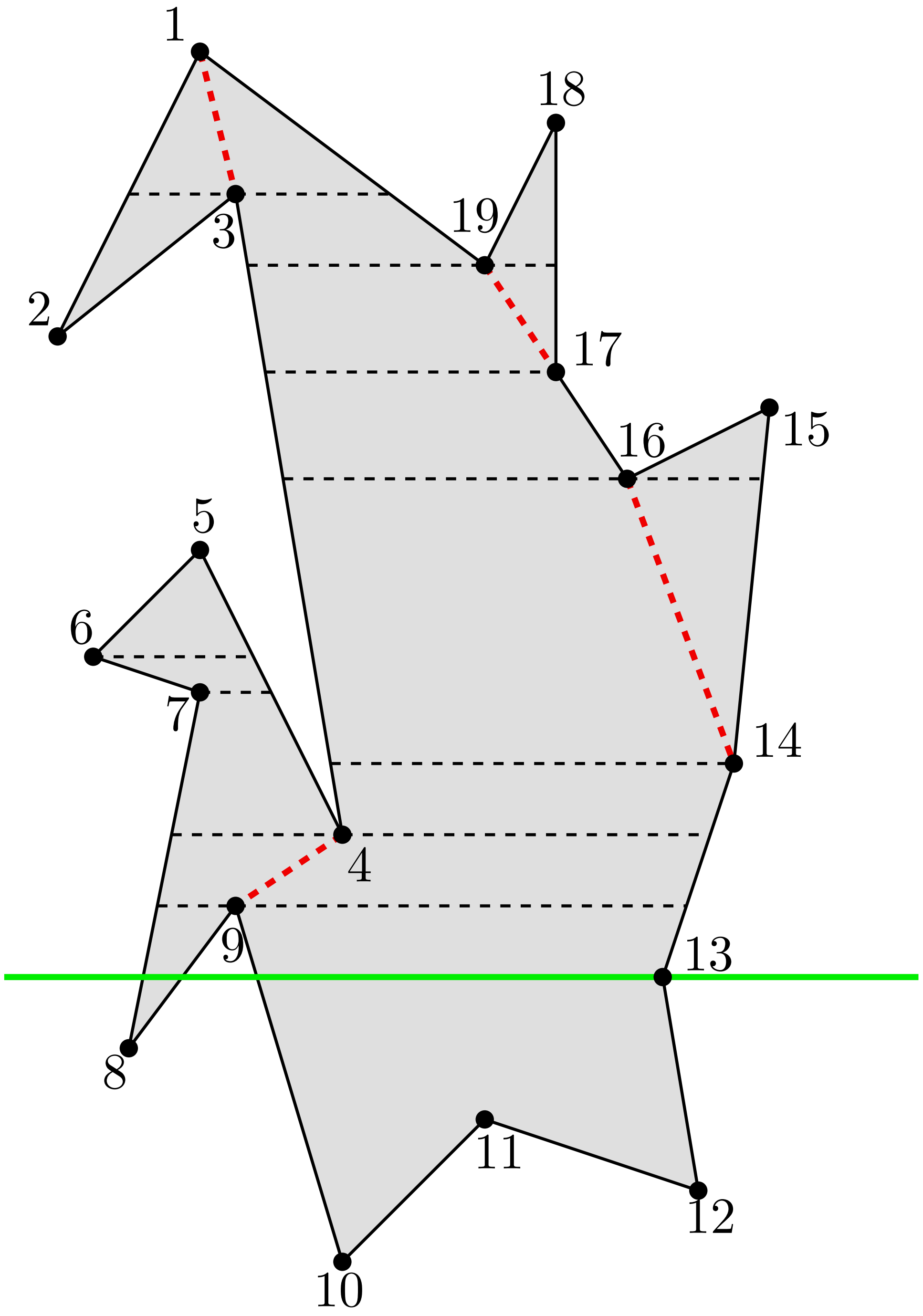
13

sweep line

$v9$ $\cancel{v9}$

$e7, e8, \cancel{e9}, e13$

$\uparrow$ $e12$



TRIANGULATING POLYGONS

Monotone partition

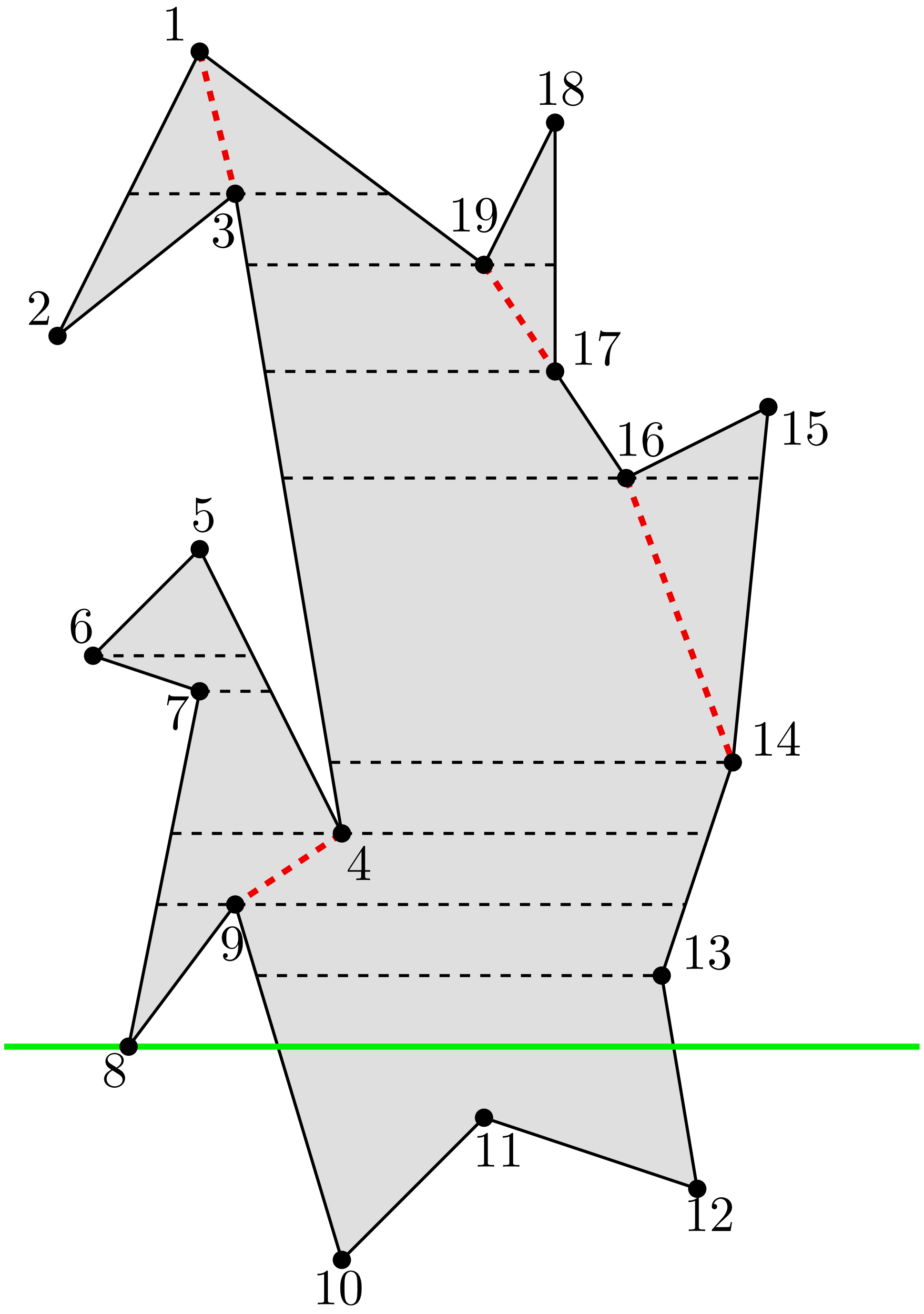
vertex

8

sweep line

$v9$ $v13$

$\cancel{e7}, \cancel{e8}, e9, e12$



TRIANGULATING POLYGONS

Monotone partition

vertex

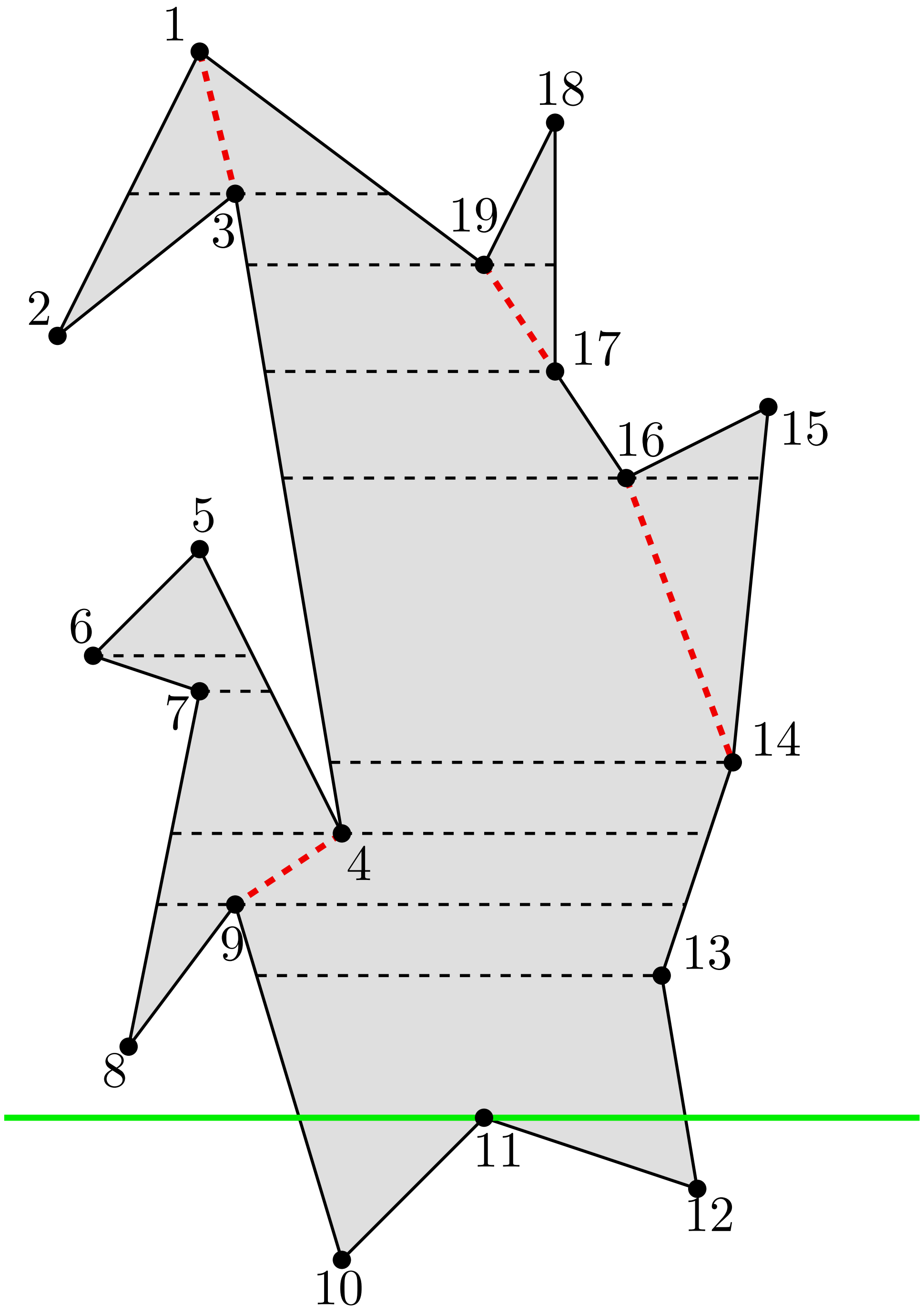
11

sweep line

$\cancel{v13}$

$e9, e12$

$\uparrow$ $e10, e11$



TRIANGULATING POLYGONS

Monotone partition

vertex

11

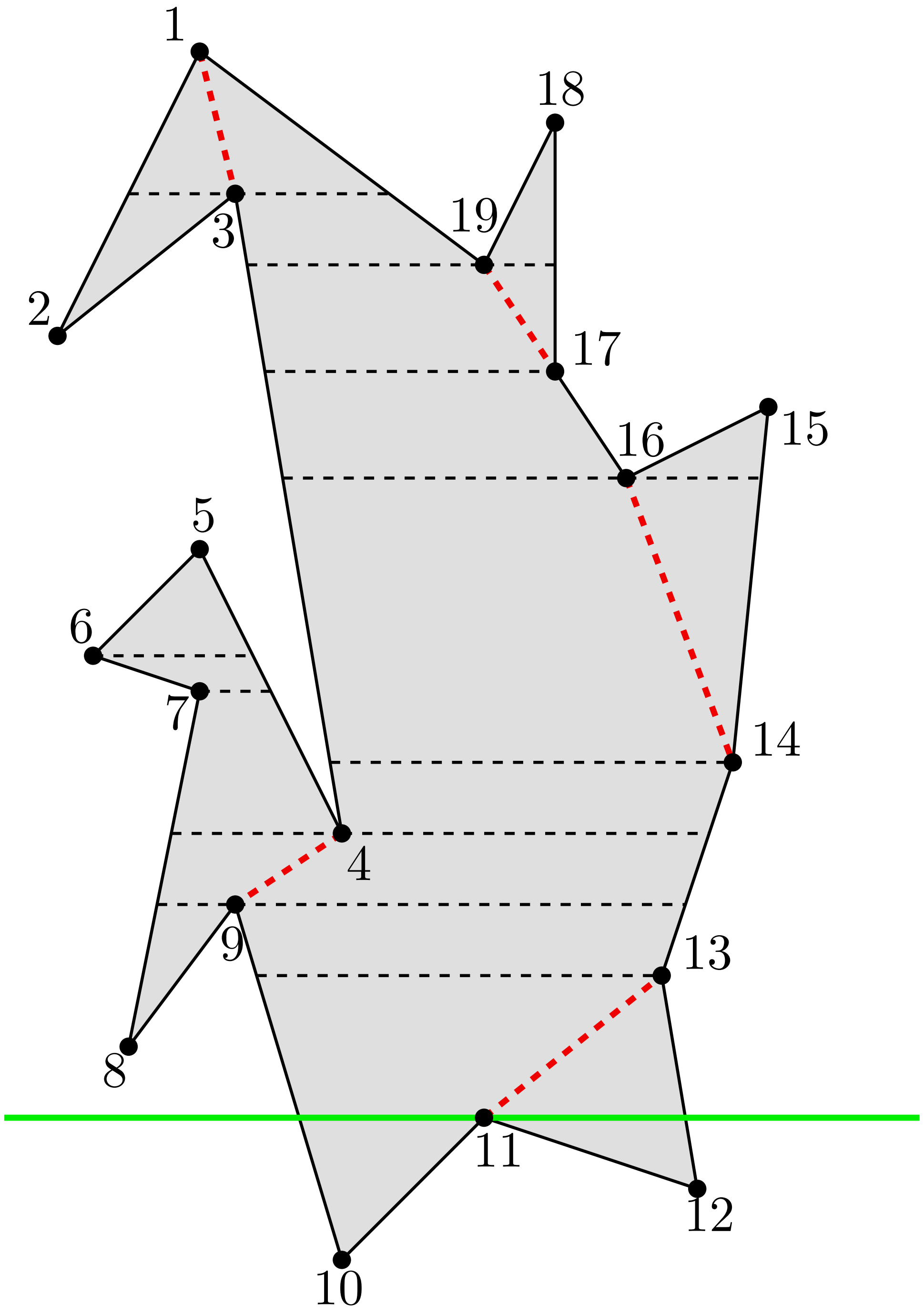
sweep line

~~$v13$~~

$e9, e12$

$e10, e11$

Diagonal $v13 - v11$



TRIANGULATING POLYGONS

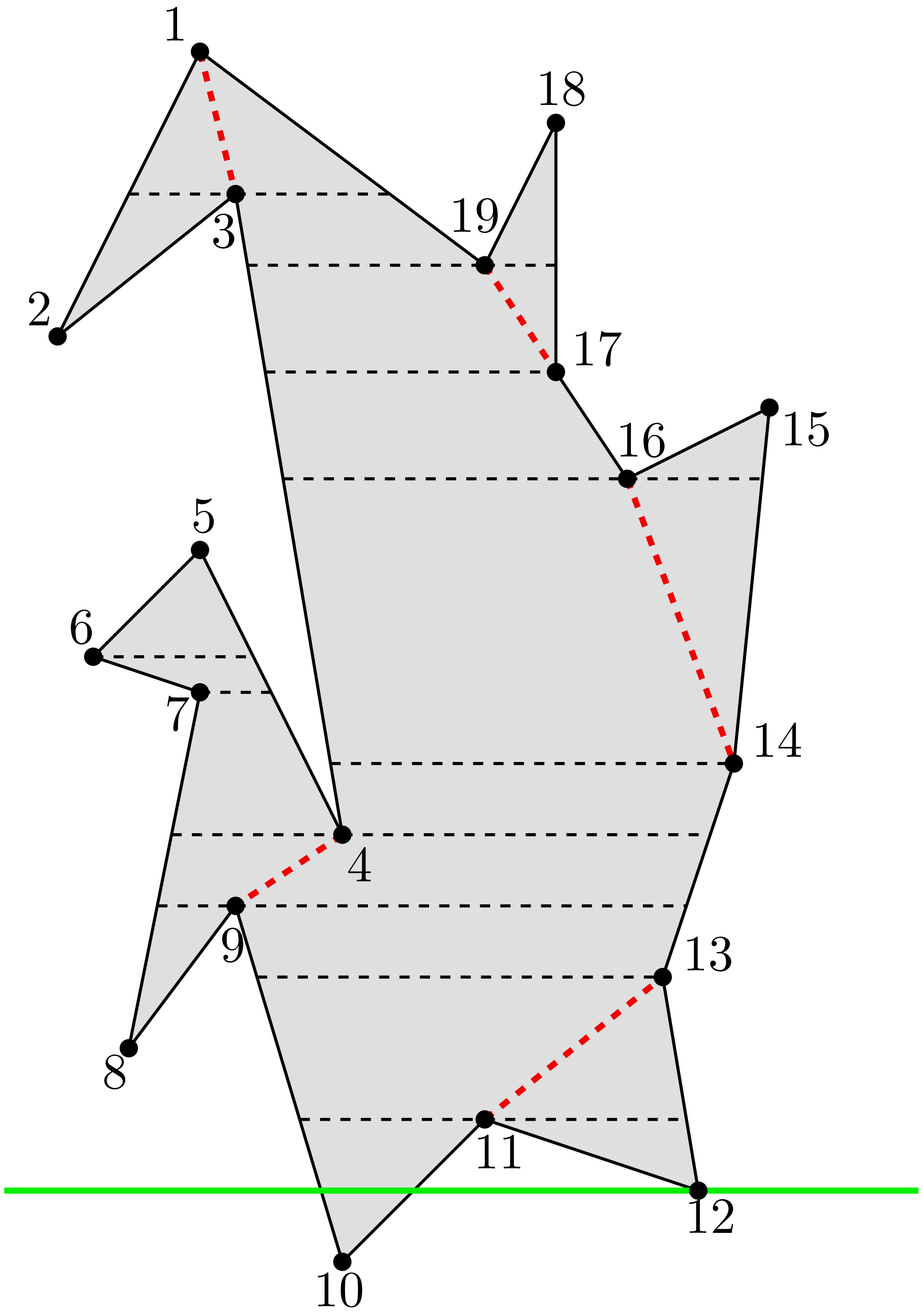
Monotone partition

vertex

12

sweep line

$\overset{v11}{\overline{e9, e10}}, \overset{v11}{\overline{\cancel{e11}, \cancel{e12}}}$



TRIANGULATING POLYGONS

Monotone partition

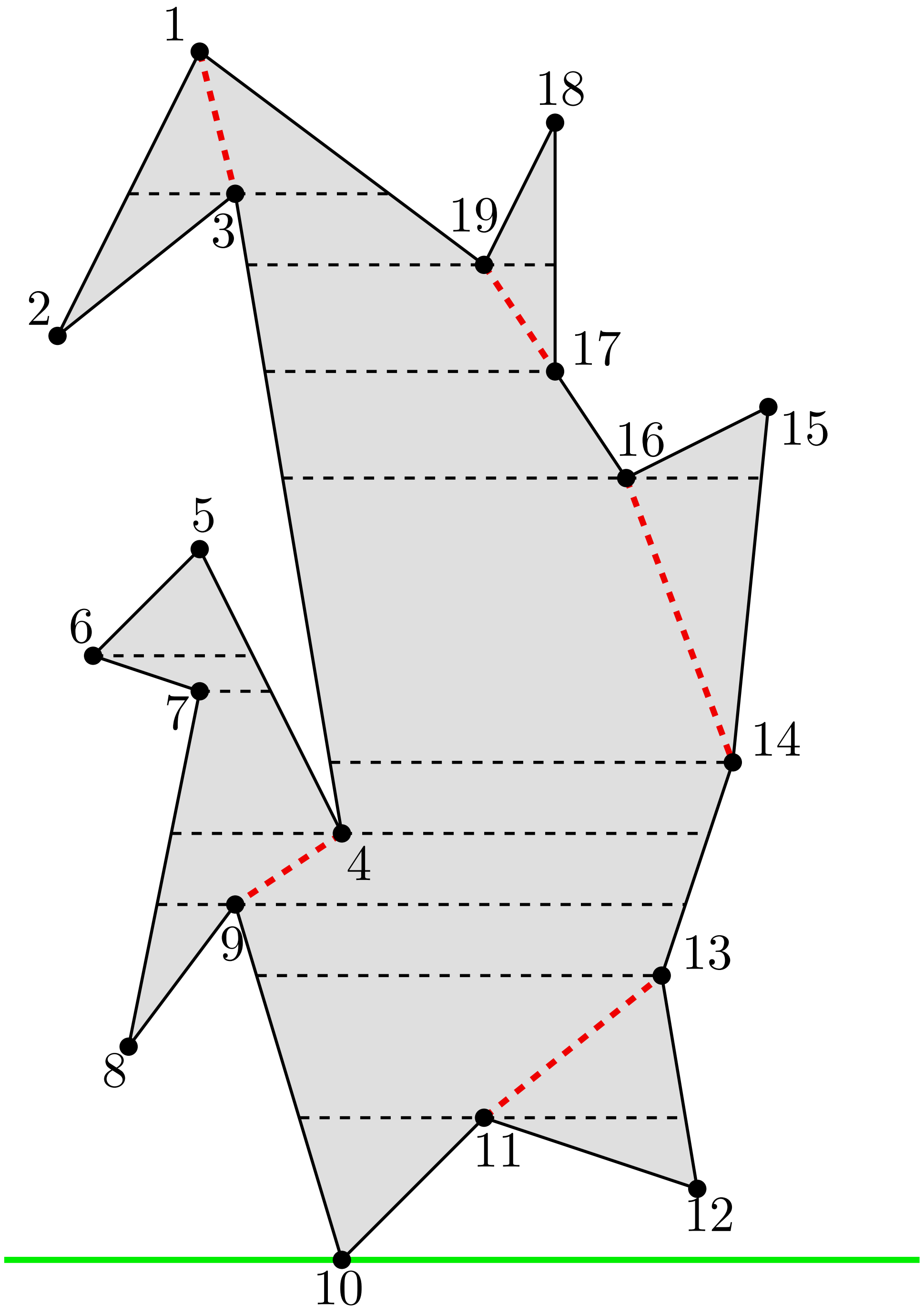
vertex

10

sweep line

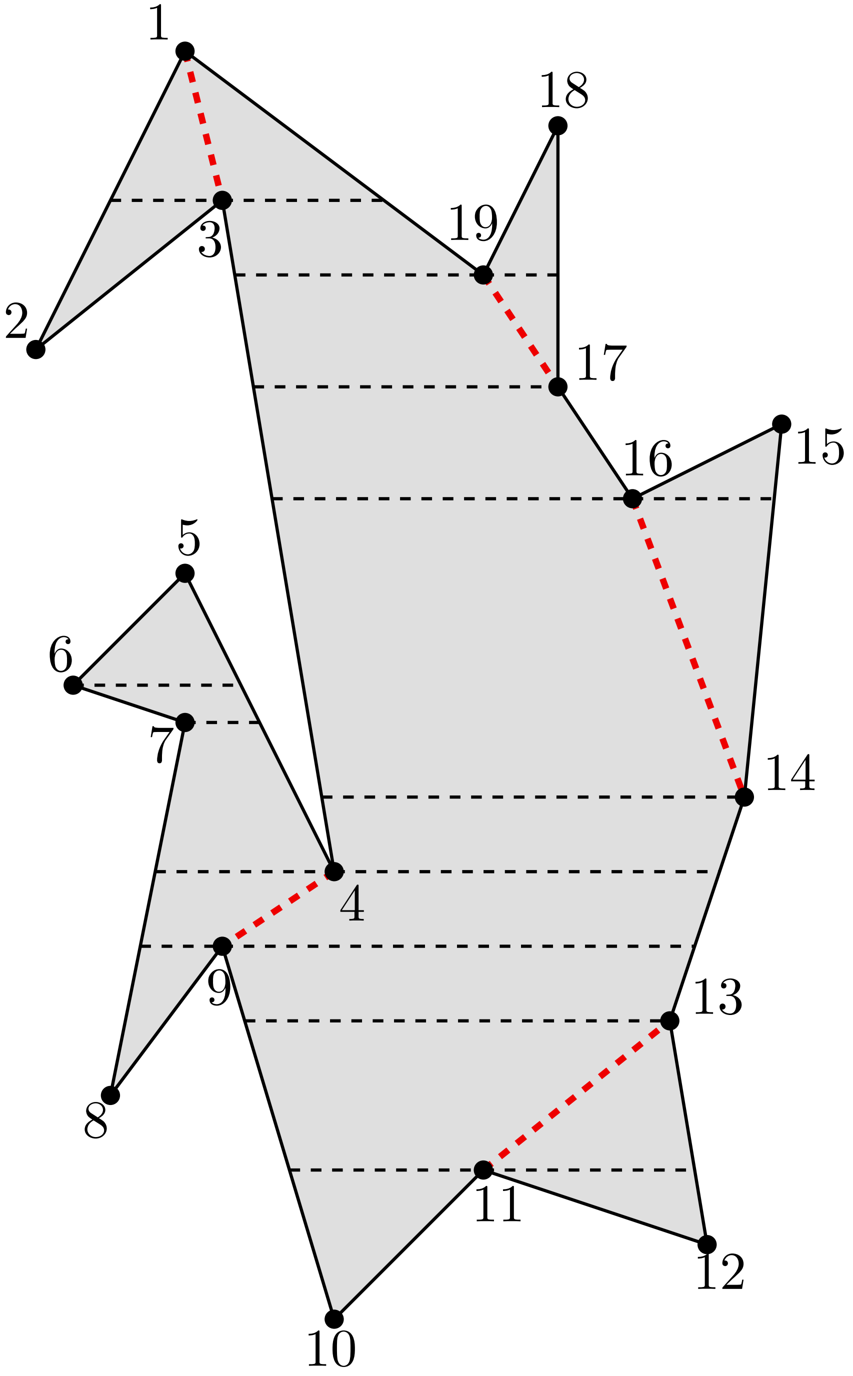
$v11$

~~$e9$~~, ~~$e10$~~



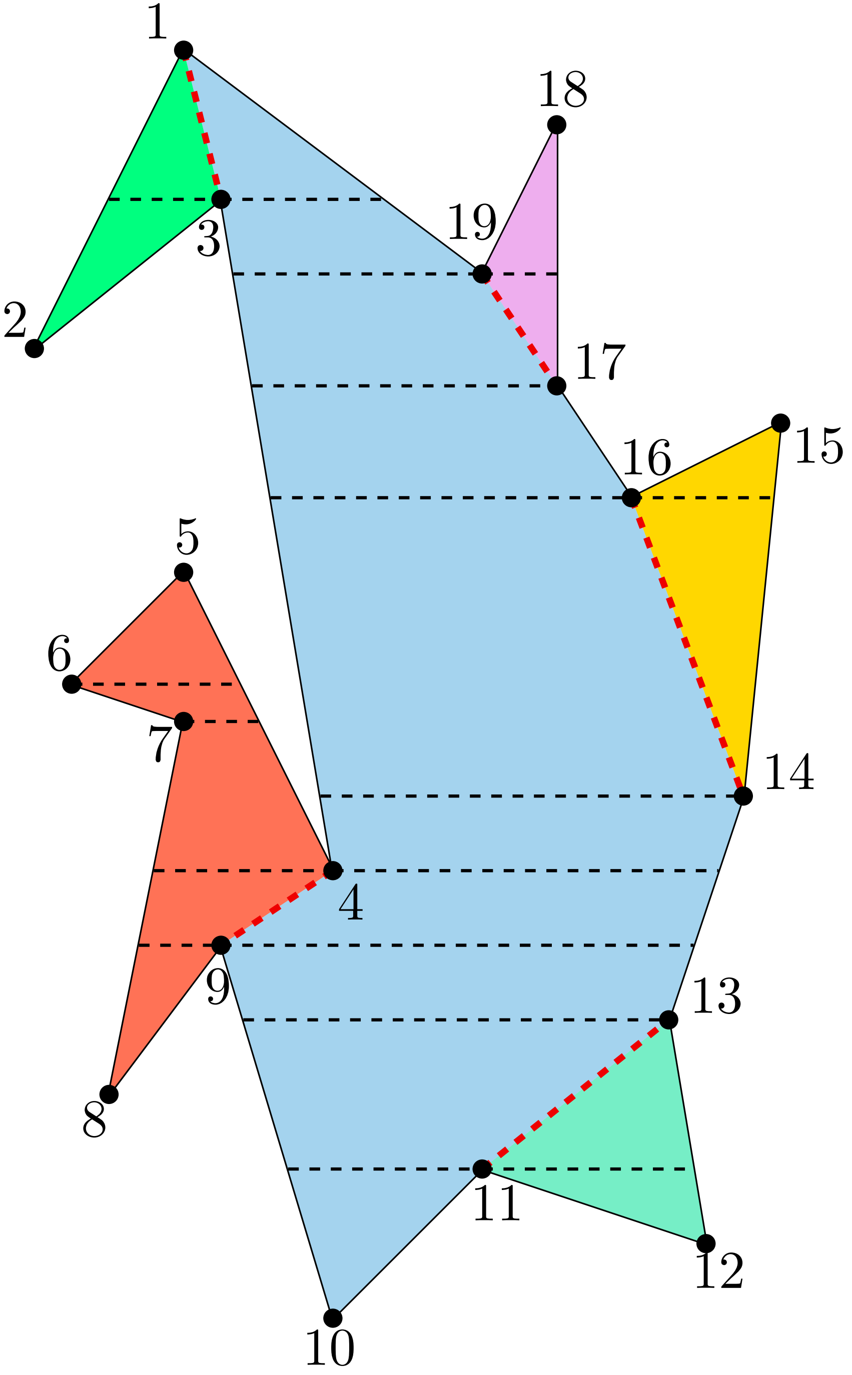
TRIANGULATING POLYGONS

Monotone partition



TRIANGULATING POLYGONS

Monotone partition



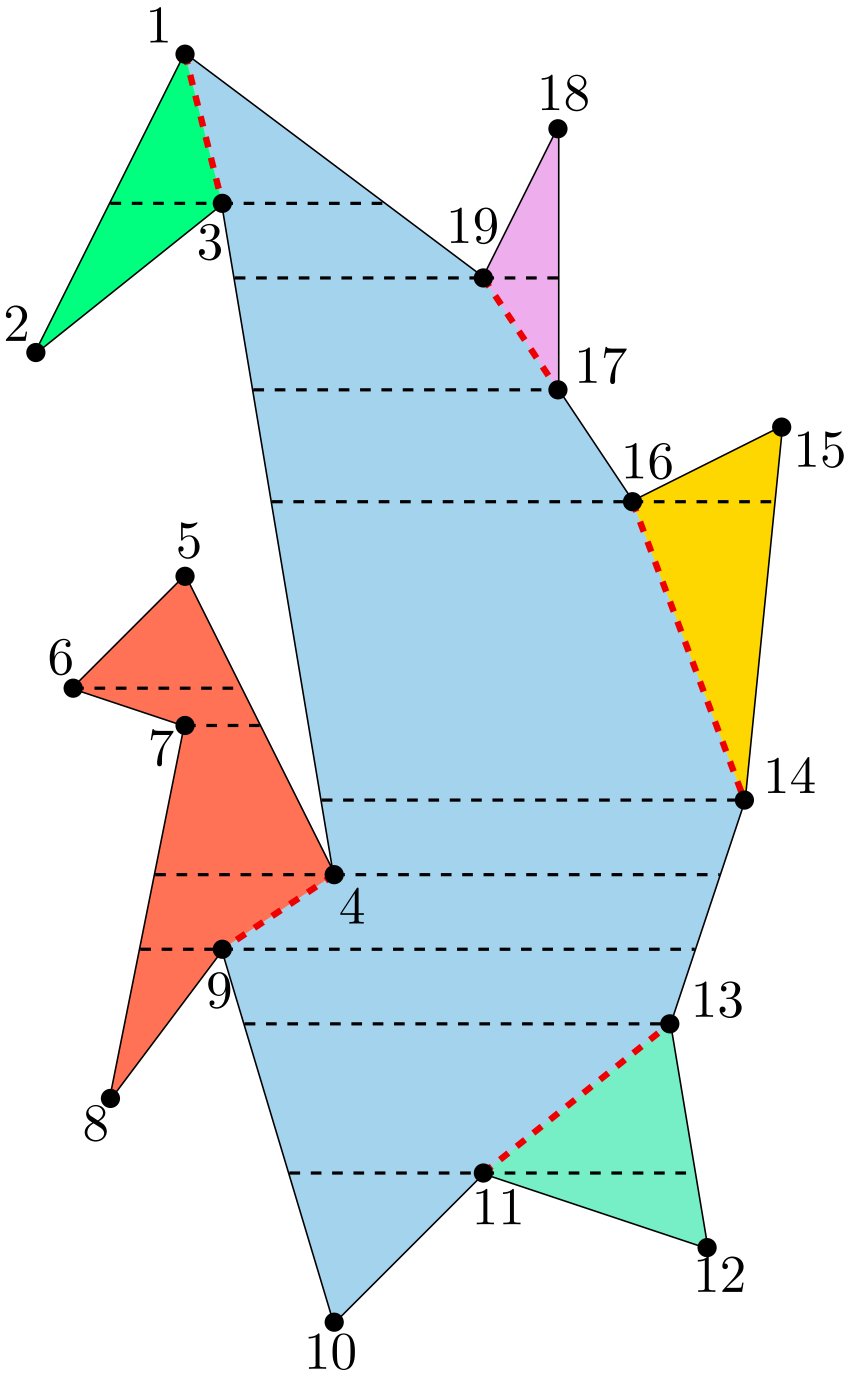
TRIANGULATING POLYGONS

Monotone partition

Running time:

- Sorting the vertices in the event queue: $O(n \log n)$ time.
- On each event, update sweep line: replace, insert or delete vertices or edges in $O(\log n)$ time each.
- There are n events.

The algorithm runs in $O(n \log n)$ time.



Summarizing

Running time of polygon triangulation:

- $O(n^2)$ by substracting ears
- $O(n^2)$ by inserting diagonals
- $O(n \log n)$ by:
  1. Decomposing the polygon into monotone subpolygons in $O(n \log n)$ time
  2. Triangulating each monotone subpolygon in $O(n)$ time

TRIANGULATING POLYGONS

Summarizing

Running time of polygon triangulation:

- $O(n^2)$ by substracting ears
- $O(n^2)$ by inserting diagonals
- $O(n \log n)$ by:
    1. Decomposing the polygon into monotone subpolygons in $O(n \log n)$ time
    2. Triangulating each monotone subpolygon in $O(n)$ time

Is it possible to triangulate a polygon in $o(n \log n)$ time?

TRIANGULATING POLYGONS

Summarizing

Running time of polygon triangulation:

- $O(n^2)$ by substracting ears
- $O(n^2)$ by inserting diagonals
- $O(n \log n)$ by:
  1. Decomposing the polygon into monotone subpolygons in $O(n \log n)$ time
  2. Triangulating each monotone subpolygon in $O(n)$ time

Is it possible to triangulate a polygon in $o(n \log n)$ time?

> Yes.
> There exists an algorithm to triangulate an n-gon in $O(n)$ time, but it is too complicated and, in practice, it is not used.

STORING THE POLYGON TRIANGULATION

Storing the polygon triangulation

Possible options, advantages and disadvantages

Storing the polygon triangulation

Possible options, advantages and disadvantages

Storing the list of all the diagonals of the triangulation

Storing the polygon triangulation

Possible options, advantages and disadvantages

Storing the list of all the diagonals of the triangulation

- **Advantage:** small memory usage.
- **Disadvantage:** it suffices to draw the triangulation, but it does not contain the proximity information. For example, finding the triangles incident to a given diagonal, or finding the neighbors of a given triangle are expensive computations.

Storing the polygon triangulation

Possible options, advantages and disadvantages

Storing the list of all the diagonals of the triangulation

> **Advantage:** small memory usage.
>
> **Disadvantage:** it suffices to draw the triangulation, but it does not contain the proximity information. For example, finding the triangles incident to a given diagonal, or finding the neighbors of a given triangle are expensive computations.

For each triangle, storing the sorted list of its vertices and edges, as well as the sorted list of its neighbors.

Storing the polygon triangulation

Possible options, advantages and disadvantages

Storing the list of all the diagonals of the triangulation

Advantage: small memory usage.

Disadvantage: it suffices to draw the triangulation, but it does not contain the proximity information. For example, finding the triangles incident to a given diagonal, or finding the neighbors of a given triangle are expensive computations.

For each triangle, storing the sorted list of its vertices and edges, as well as the sorted list of its neighbors.

Advantage: allows to quickly recover neighborhood information.

Disadvantage: the stored data is redundant and it uses more space than required.

Storing the polygon triangulation

Possible options, advantages and disadvantages

Storing the list of all the diagonals of the triangulation

Advantage: small memory usage.

Disadvantage: it suffices to draw the triangulation, but it does not contain the proximity information. For example, finding the triangles incident to a given diagonal, or finding the neighbors of a given triangle are expensive computations.

For each triangle, storing the sorted list of its vertices and edges, as well as the sorted list of its neighbors.

Advantage: allows to quickly recover neighborhood information.

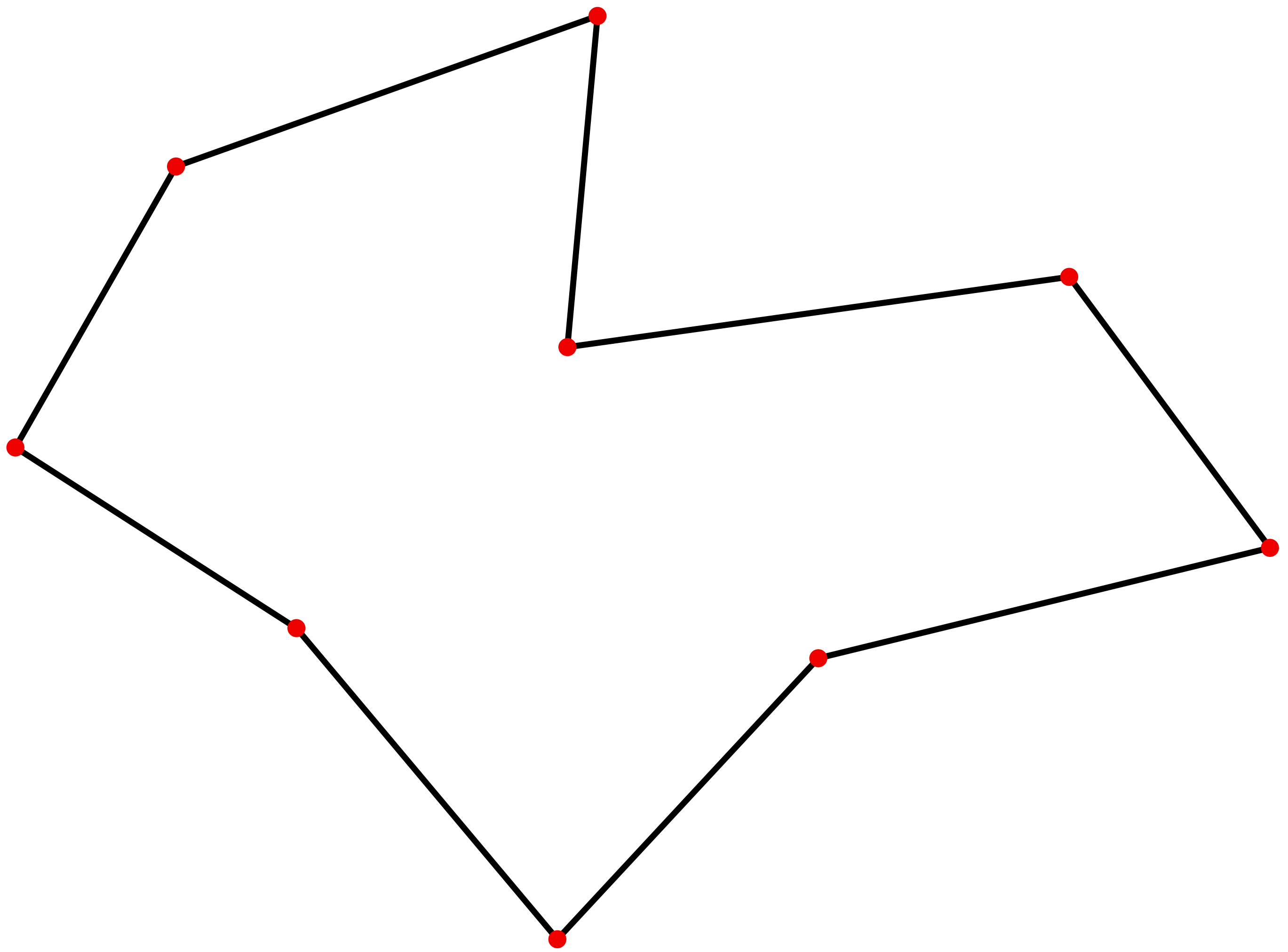
Disadvantage: the stored data is redundant and it uses more space than required.

The data structure which is most frequently used to store a triangulation is the DCEL (doubly connected edge list).

The DCEL is also used to store plane partitions, polyhedra, meshes, etc.

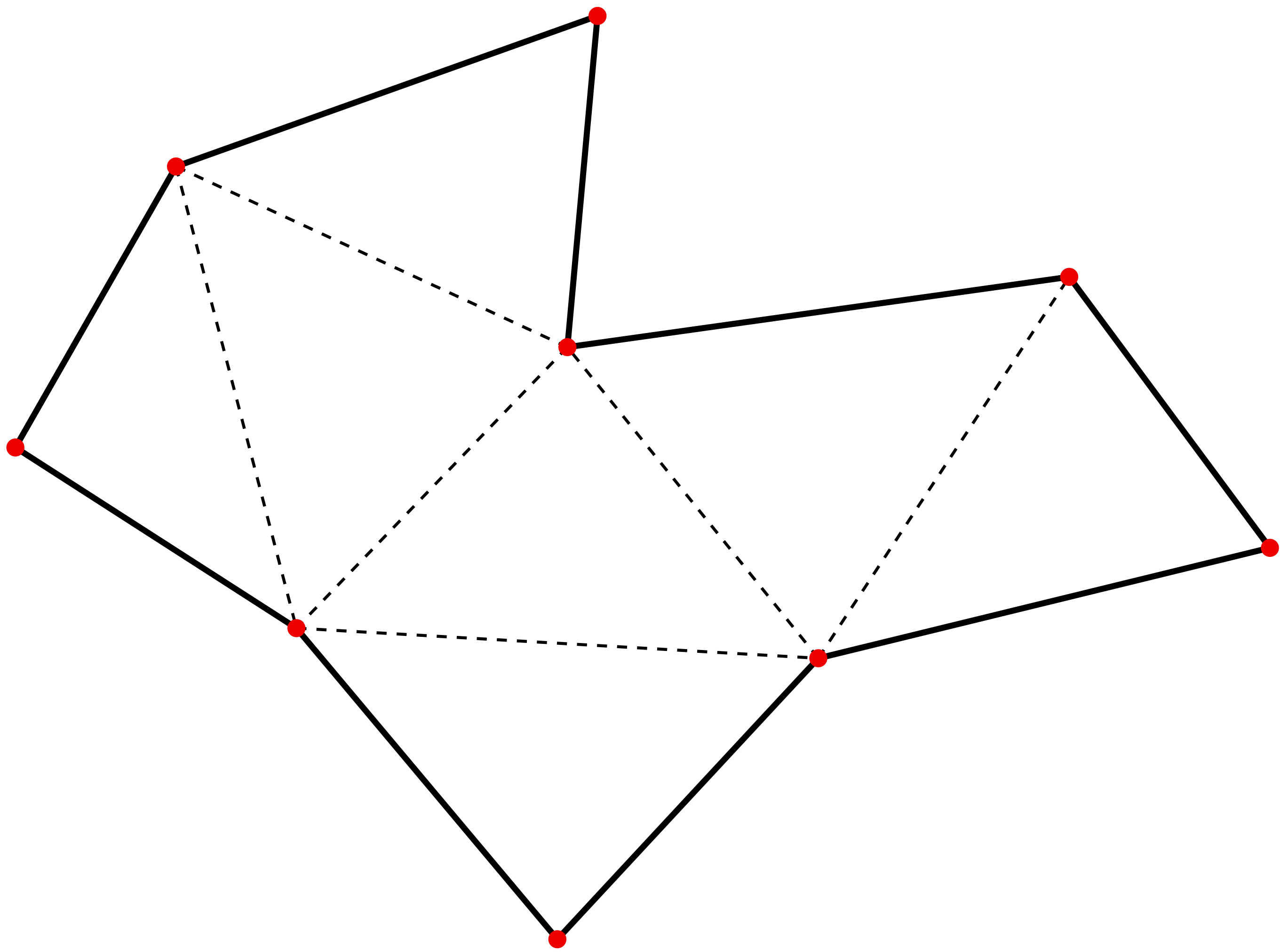
Storing the polygon triangulation

DCEL



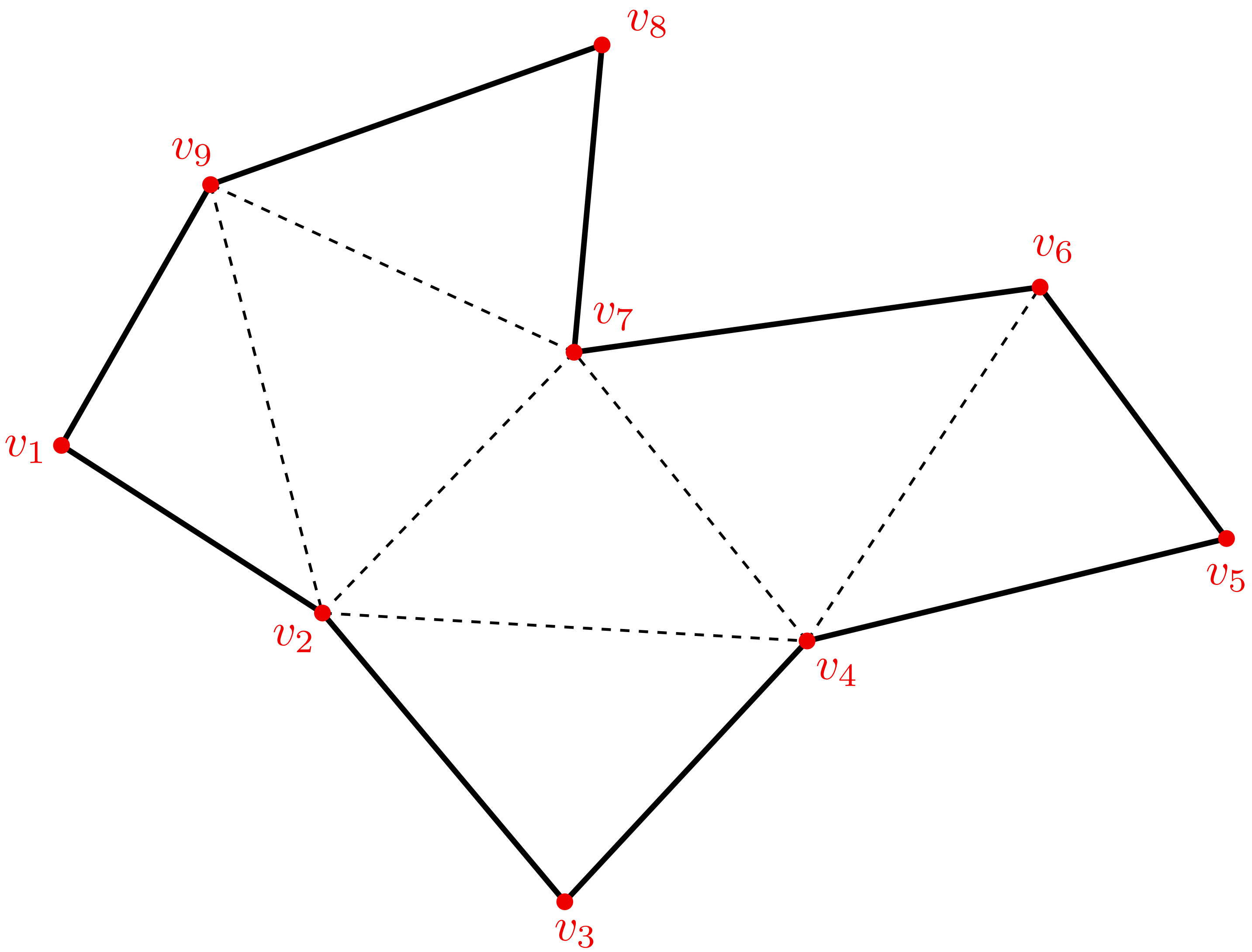
Storing the polygon triangulation

DCEL



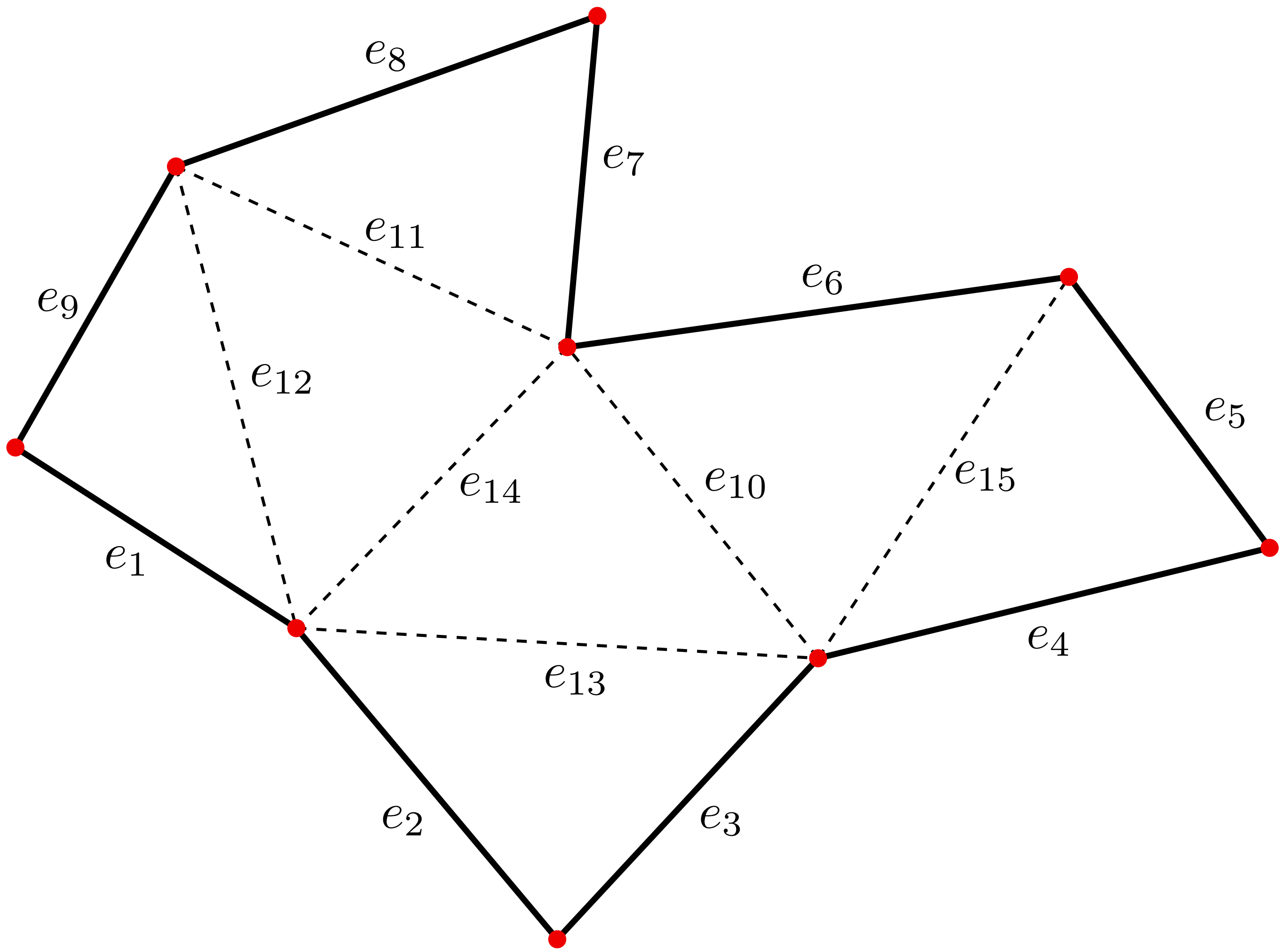
Storing the polygon triangulation

DCEL



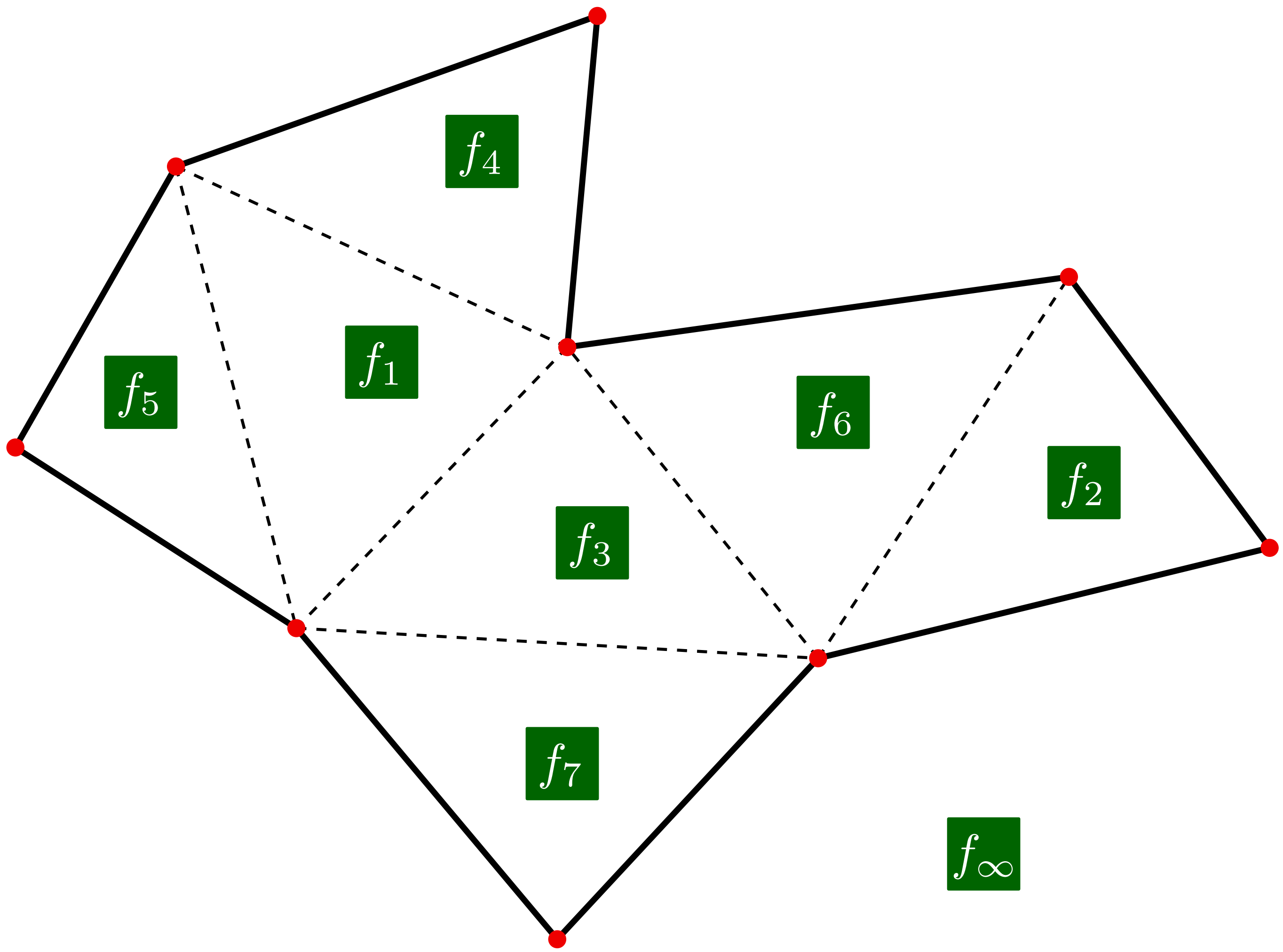
Storing the polygon triangulation

DCEL



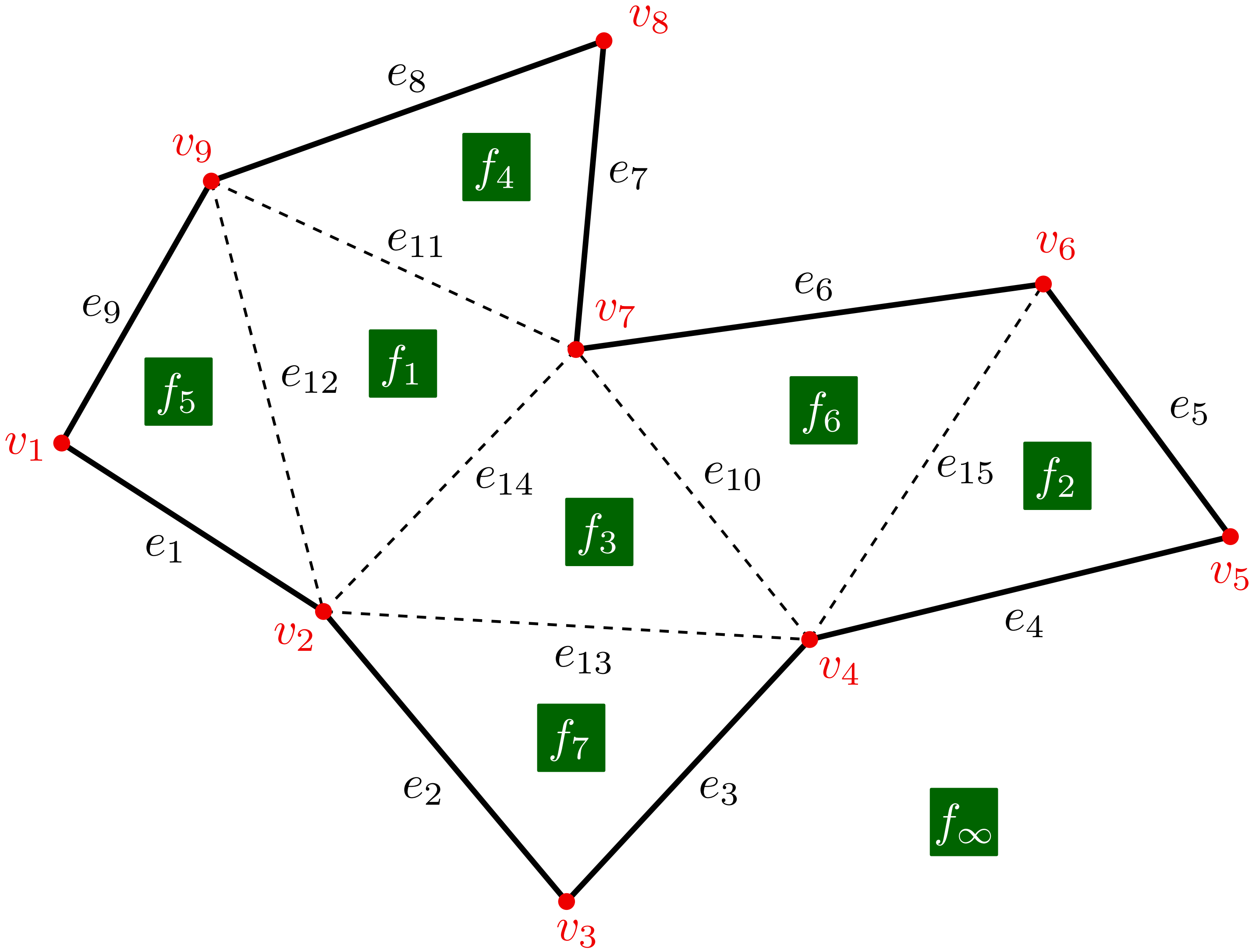
Storing the polygon triangulation

DCEL



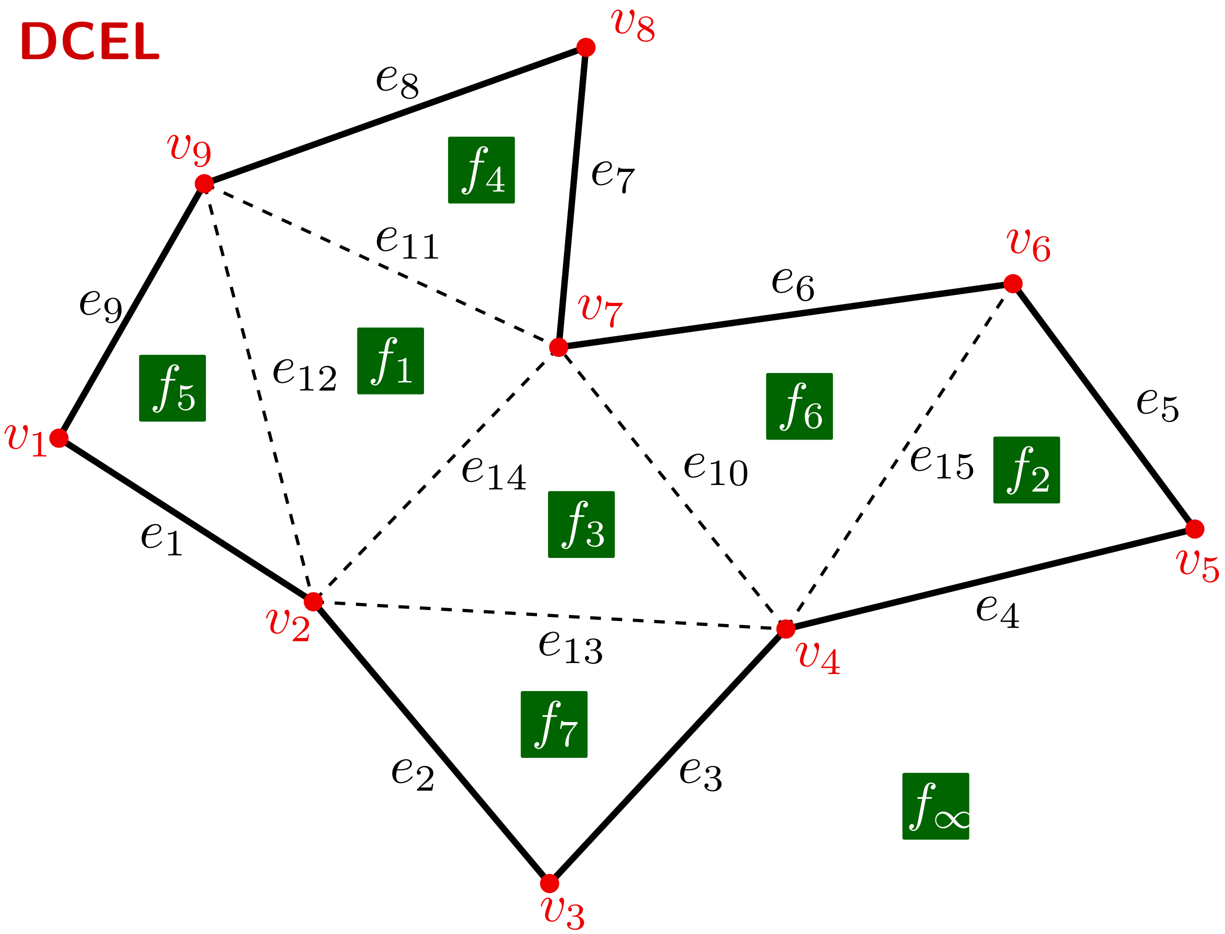
Storing the polygon triangulation

DCEL



Storing the polygon triangulation

DCEL

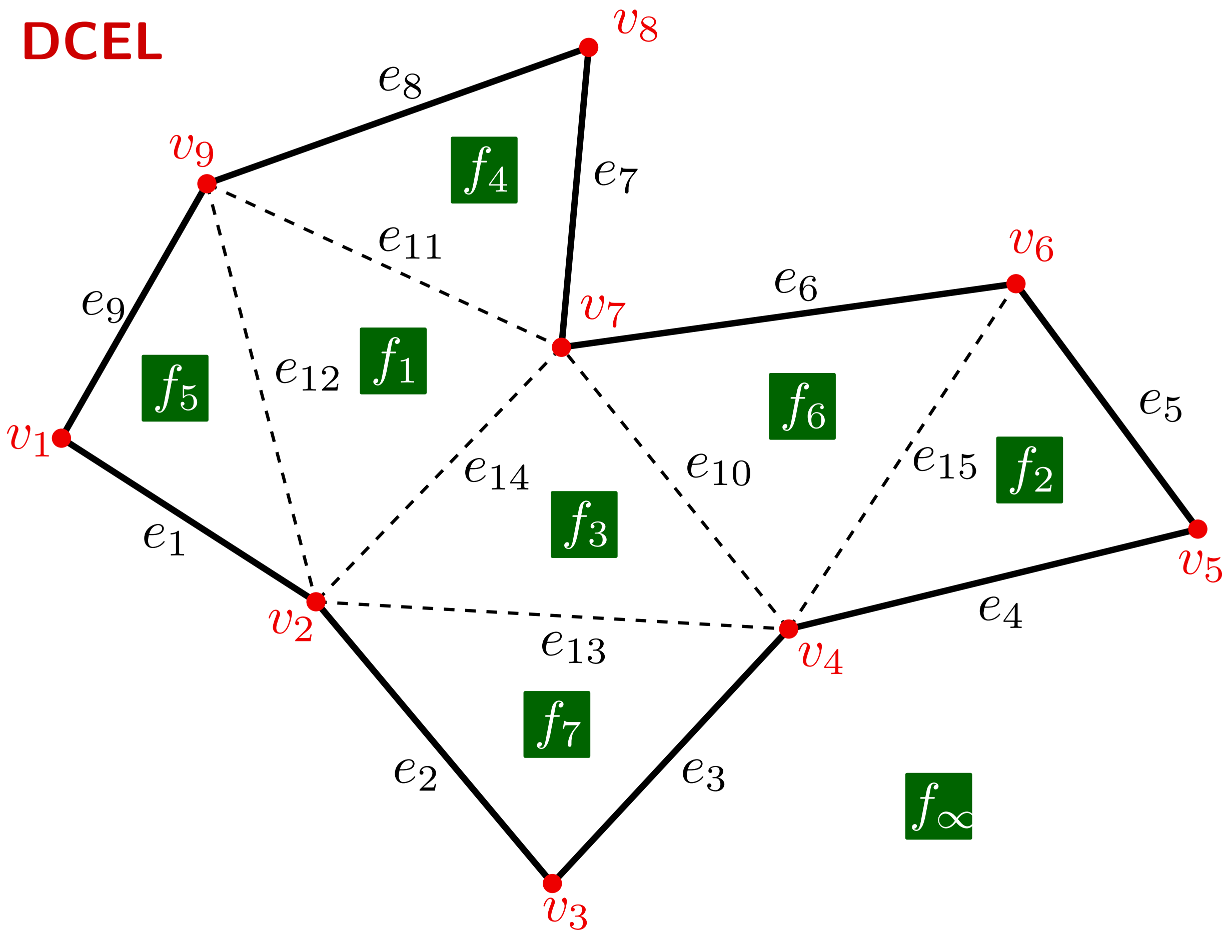


Storing the polygon triangulation

Table of vertices

v	x	y	e
1	x_1	y_1	1
2	x_2	y_2	1
3	x_3	y_3	2
4	x_4	y_4	10
5	x_5	y_5	4
6	x_6	y_6	6
7	x_7	y_7	10
8	x_8	y_8	8
9	x_9	y_9	9

DCEL

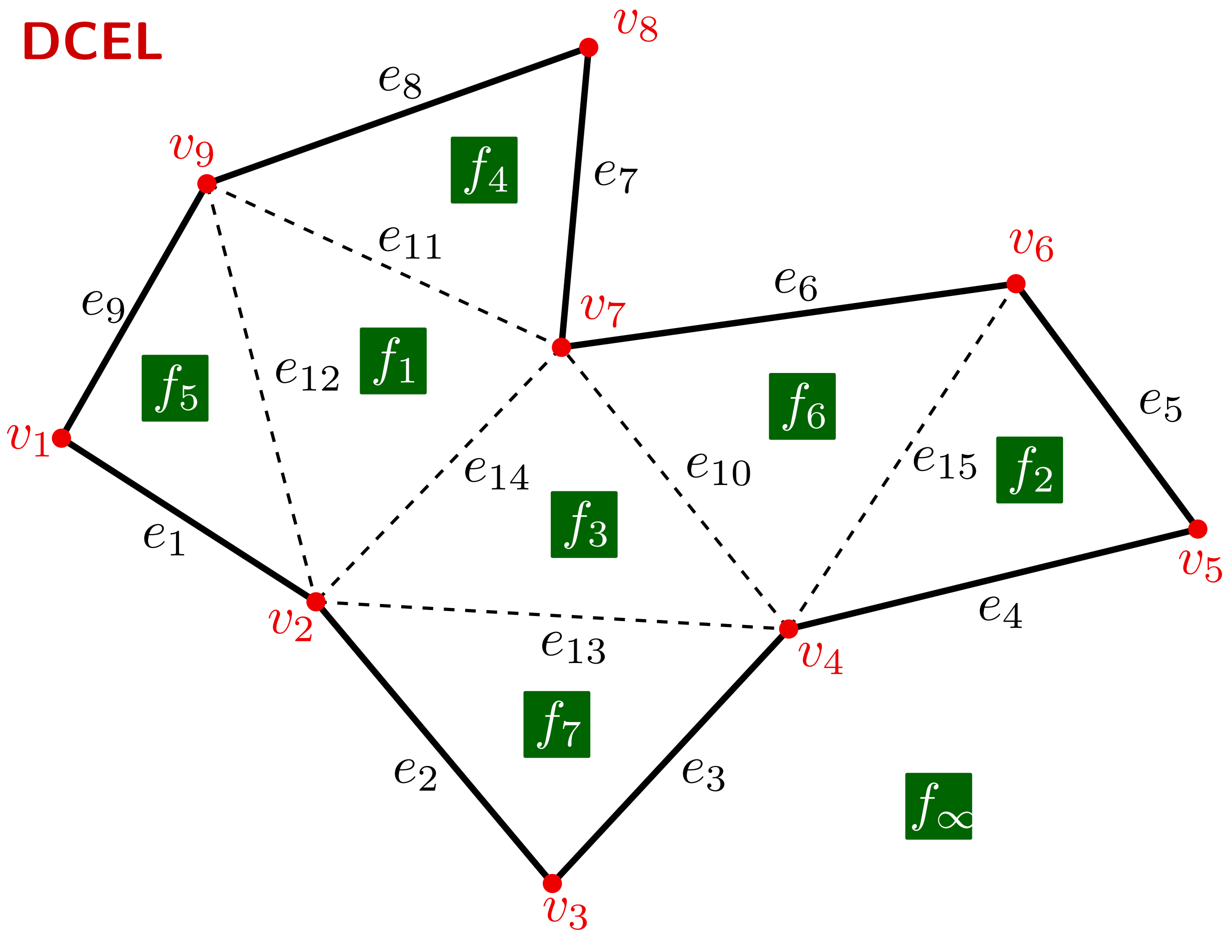


Storing the polygon triangulation

Table of faces

f	e
1	11
2	4
3	10
4	11
5	1
6	6
7	2
∞	9

DCEL

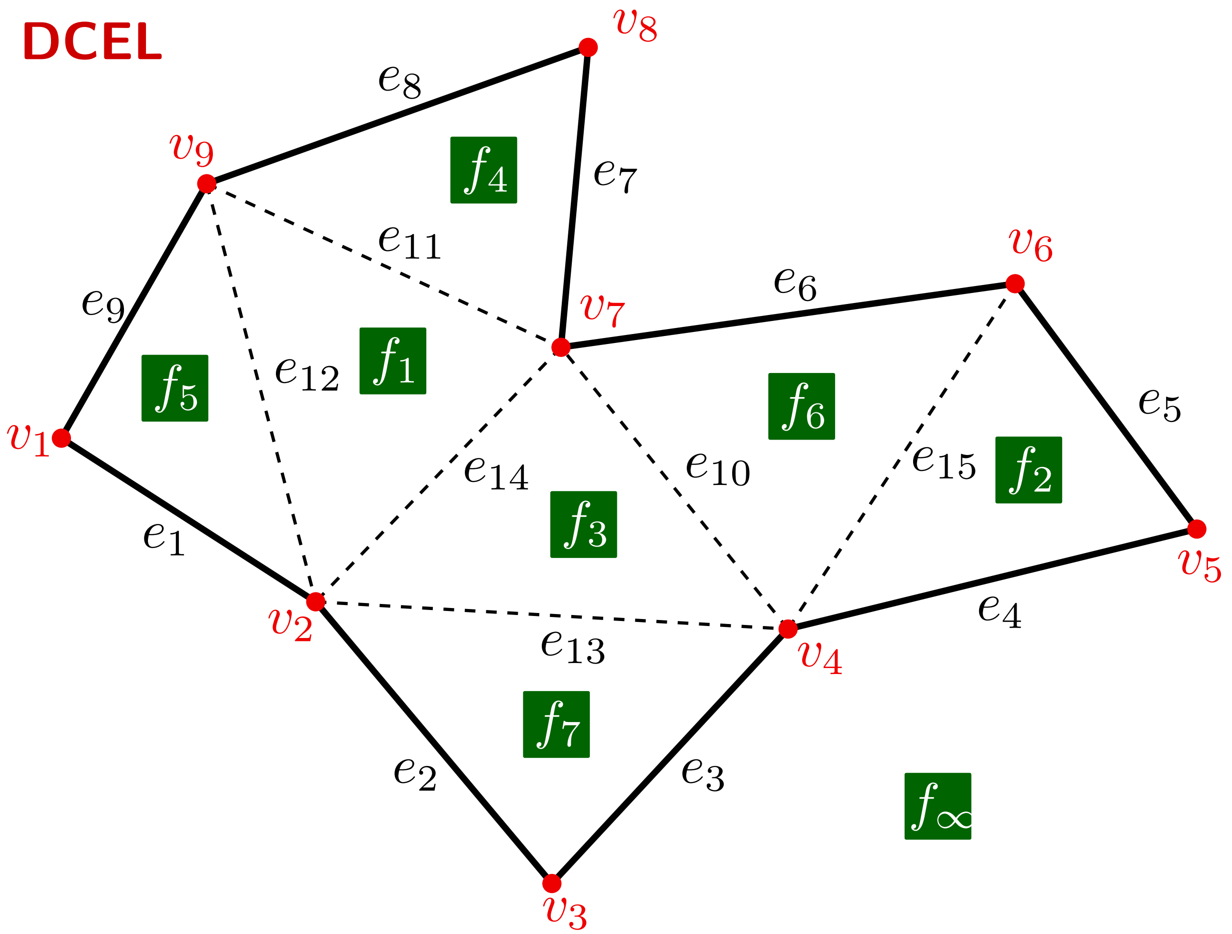


Storing the polygon triangulation

DCEL

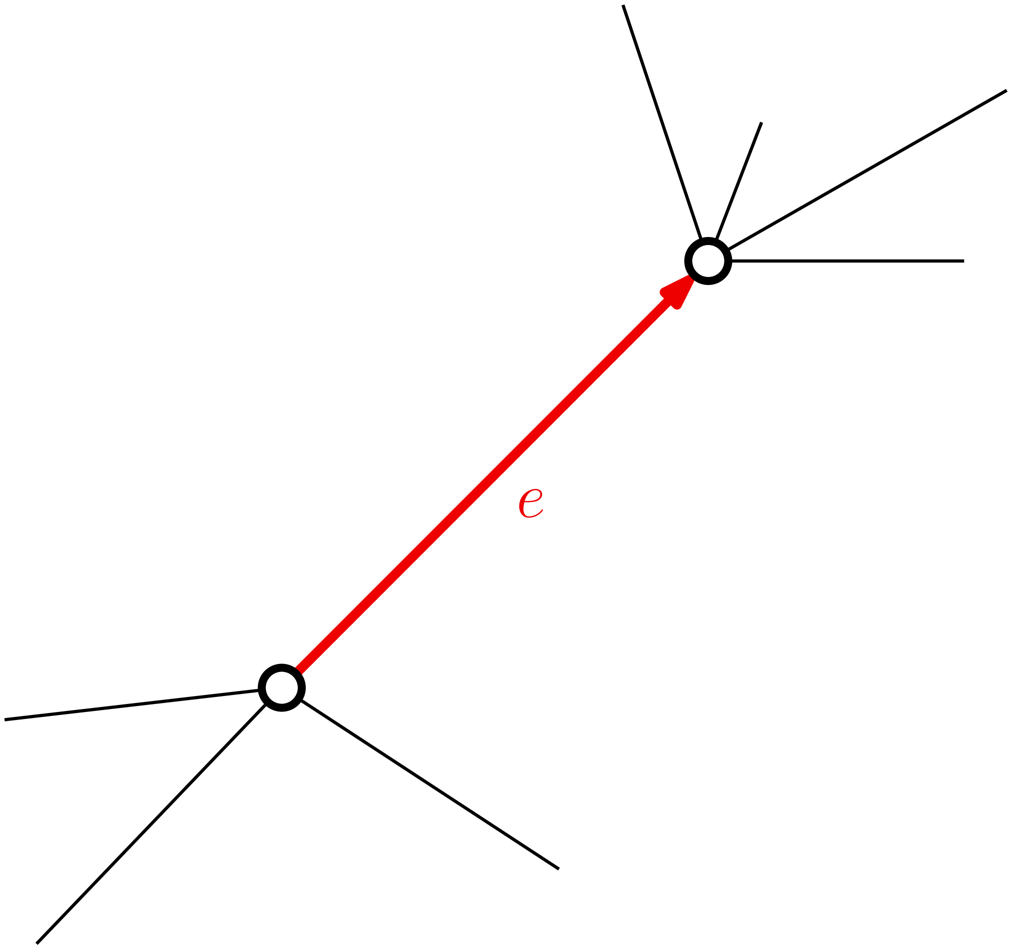
e	v_B	v_E	f_L	f_R	e_P	e_N
1	1	2	5	∞	9	2
2	2	3	7	∞	13	3
3	4	3	∞	7	4	2
4	4	5	2	∞	15	5
5	5	6	2	∞	4	6
6	6	7	6	∞	15	7
7	7	8	4	∞	11	8
8	8	9	4	∞	7	9
9	9	1	5	∞	12	1
10	4	7	3	6	13	6
11	9	7	4	1	8	14
12	2	9	5	1	1	11
13	2	4	3	7	14	3
14	2	7	1	3	12	10
15	4	6	6	2	10	5

DCEL



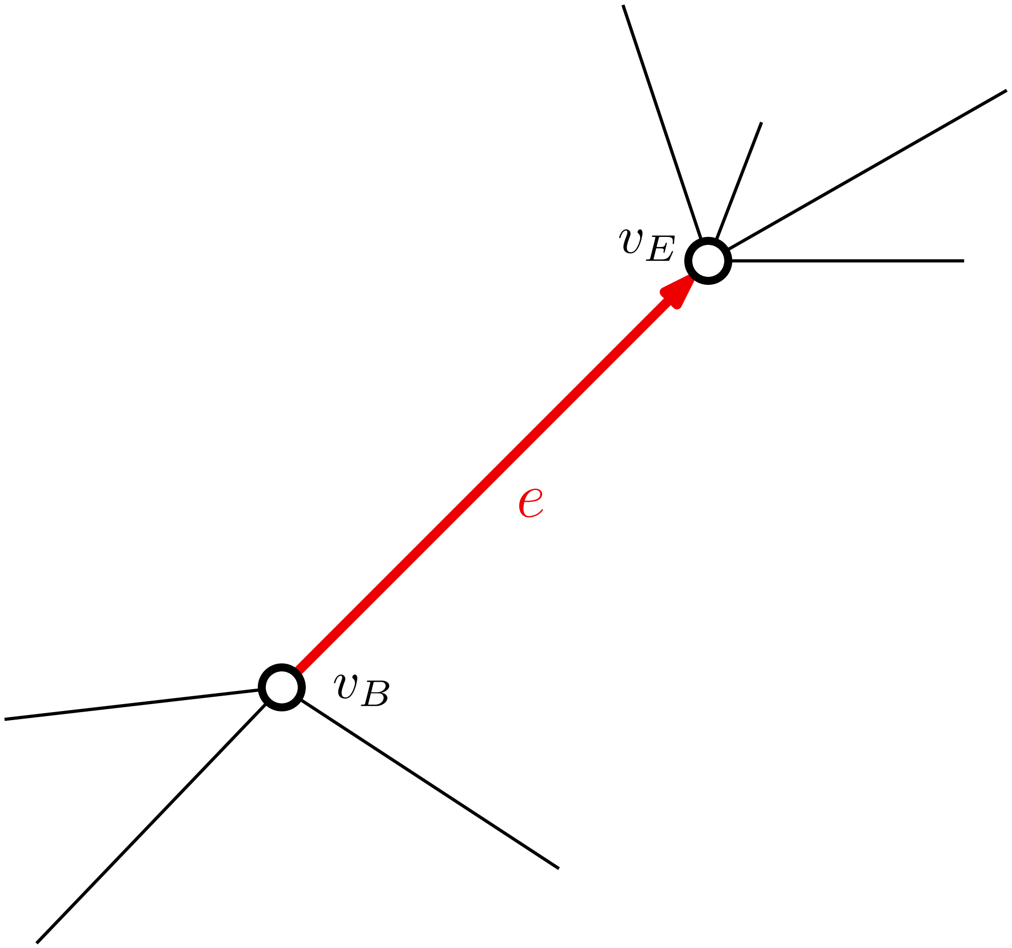
Storing the polygon triangulation

DCEL



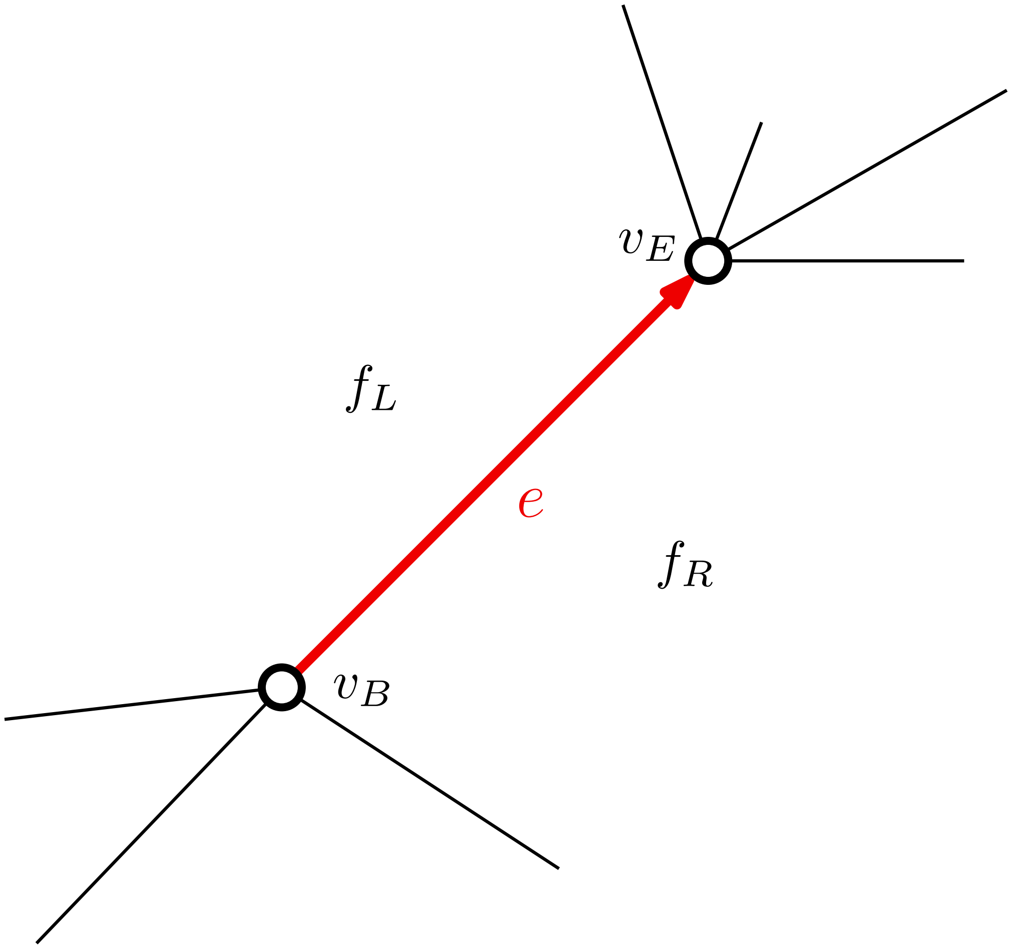
Storing the polygon triangulation

DCEL



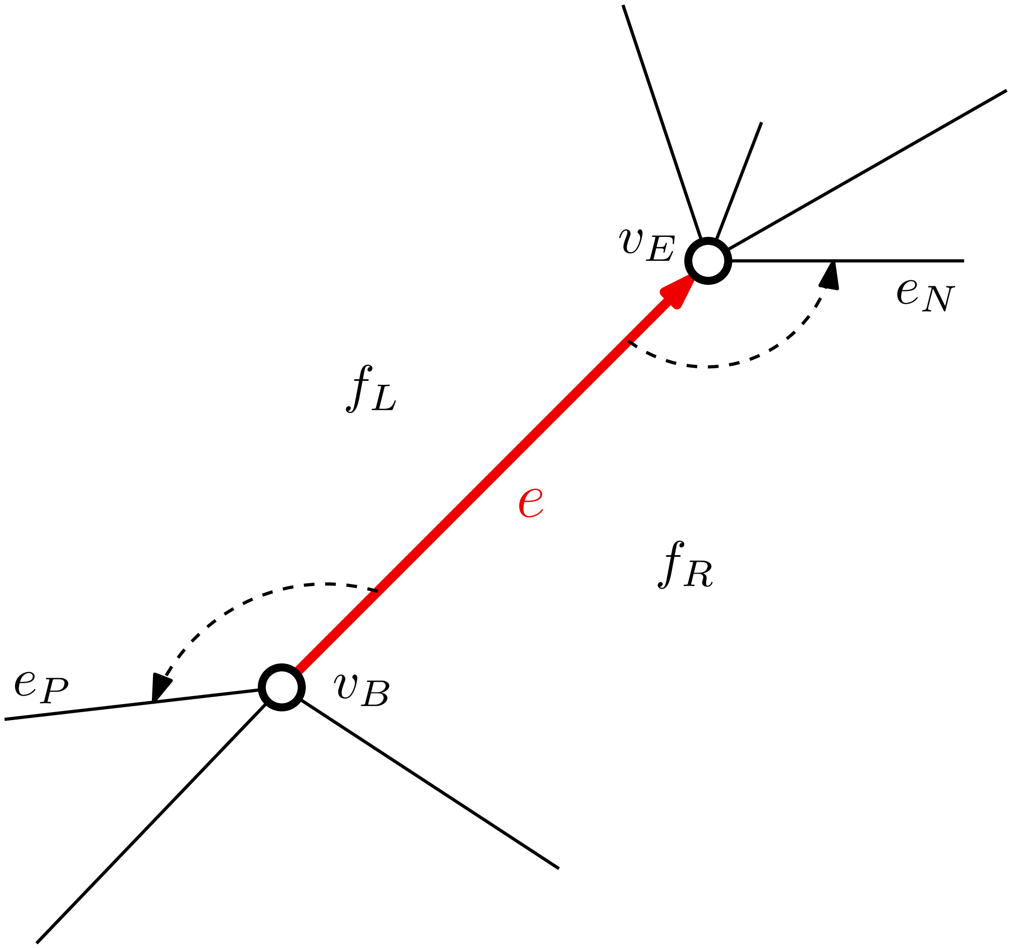
Storing the polygon triangulation

DCEL



Storing the polygon triangulation

DCEL

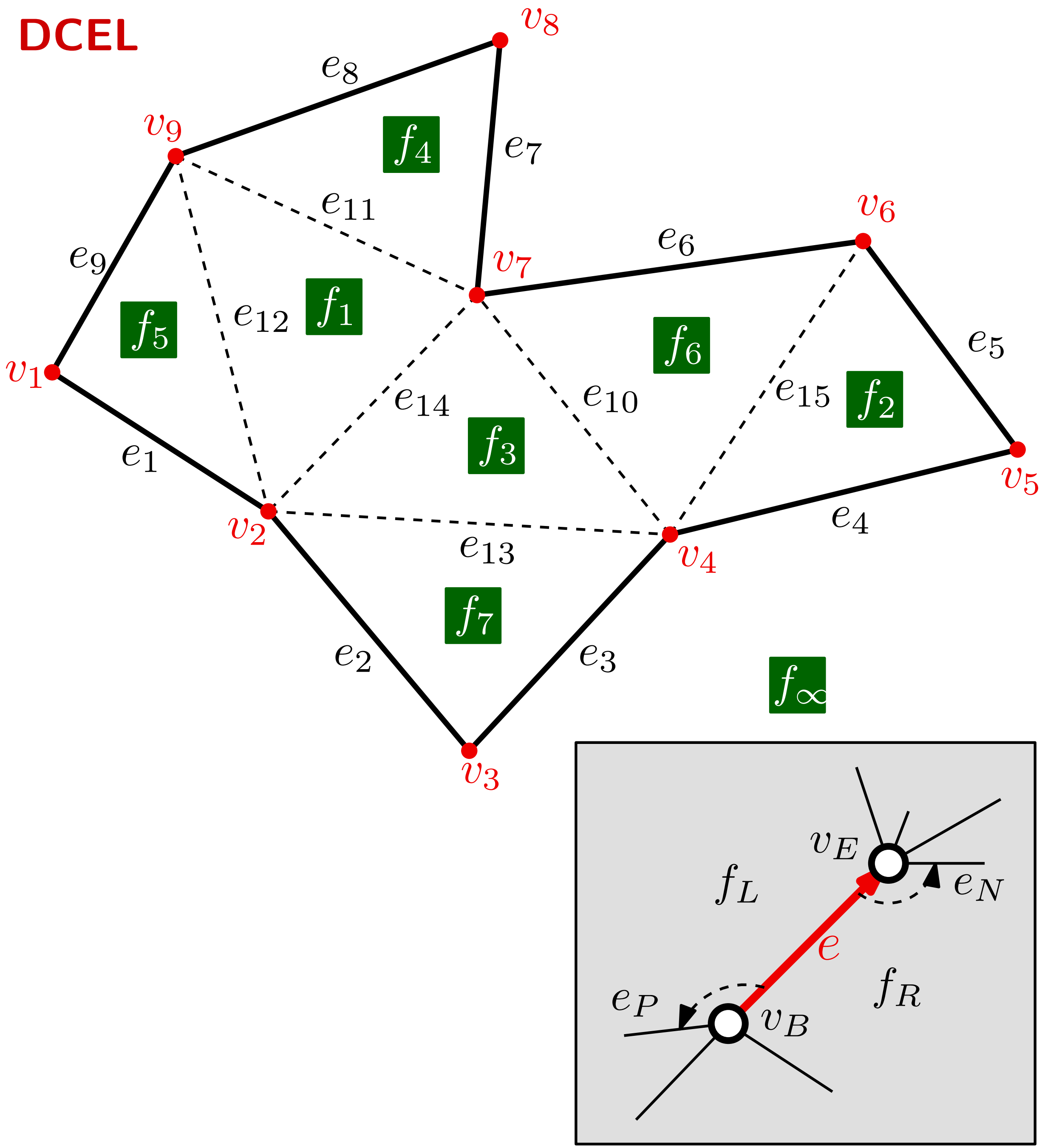


Storing the polygon triangulation

DCEL

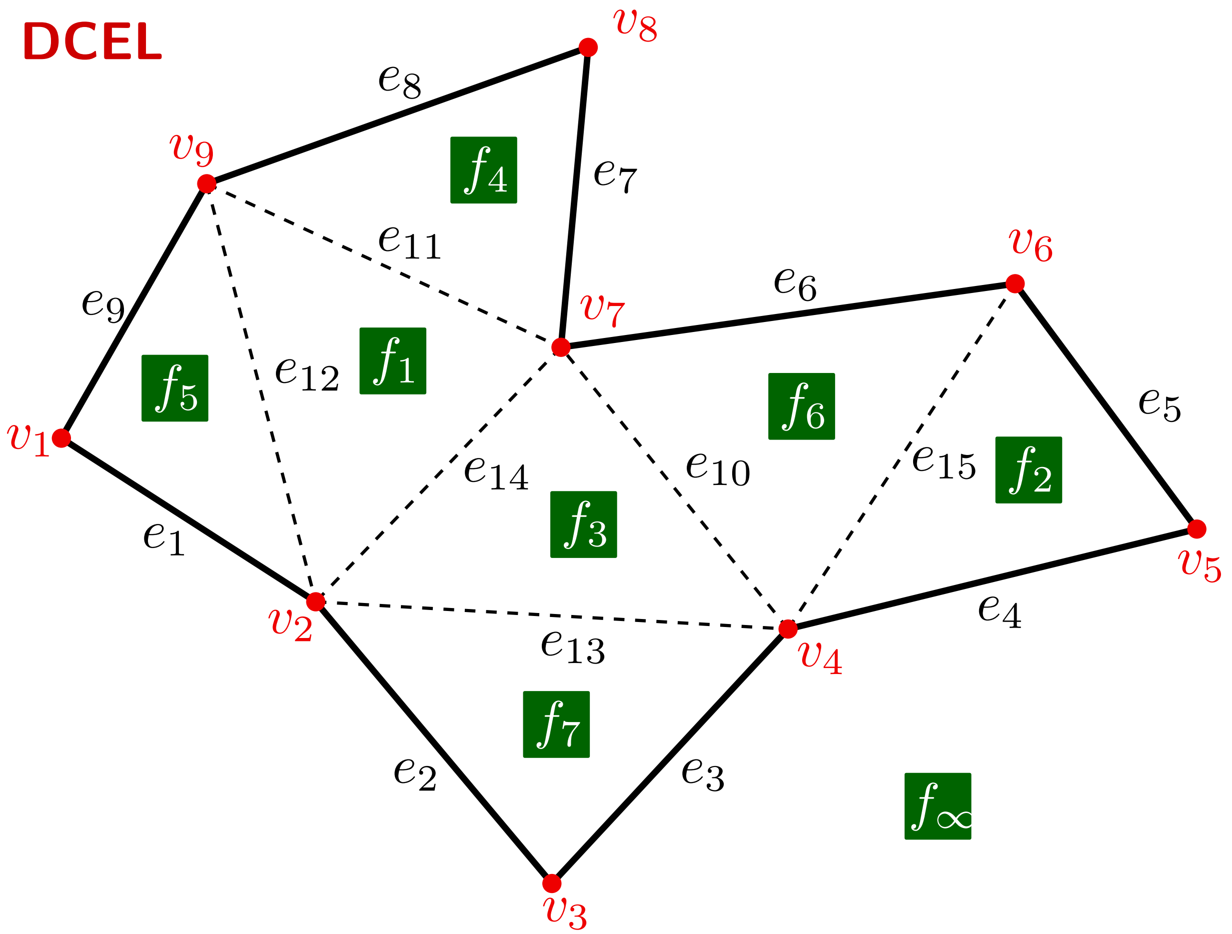
e	v_B	v_E	f_L	f_R	e_P	e_N
1	1	2	5	∞	9	2
2	2	3	7	∞	13	3
3	4	3	∞	7	4	2
4	4	5	2	∞	15	5
5	5	6	2	∞	4	6
6	6	7	6	∞	15	7
7	7	8	4	∞	11	8
8	8	9	4	∞	7	9
9	9	1	5	∞	12	1
10	4	7	3	6	13	6
11	9	7	4	1	8	14
12	2	9	5	1	1	11
13	2	4	3	7	14	3
14	2	7	1	3	12	10
15	4	6	6	2	10	5

DCEL



Storing the polygon triangulation

DCEL



Storage space

- For each face:

  1 pointer

- For each vertex:

  2 coordinates + 1 pointer

- For each edge:

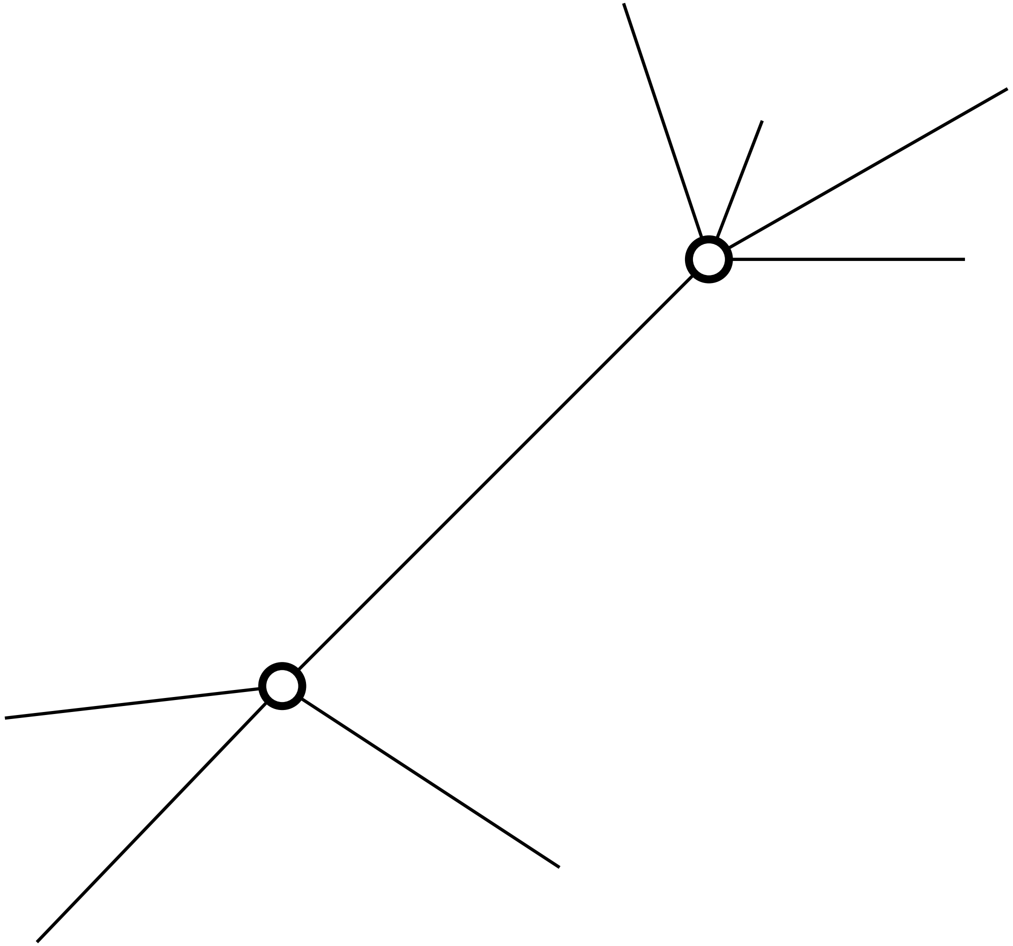
  6 pointers

In total, the storage space is $O(n)$.

Storing the polygon triangulation

DCEL

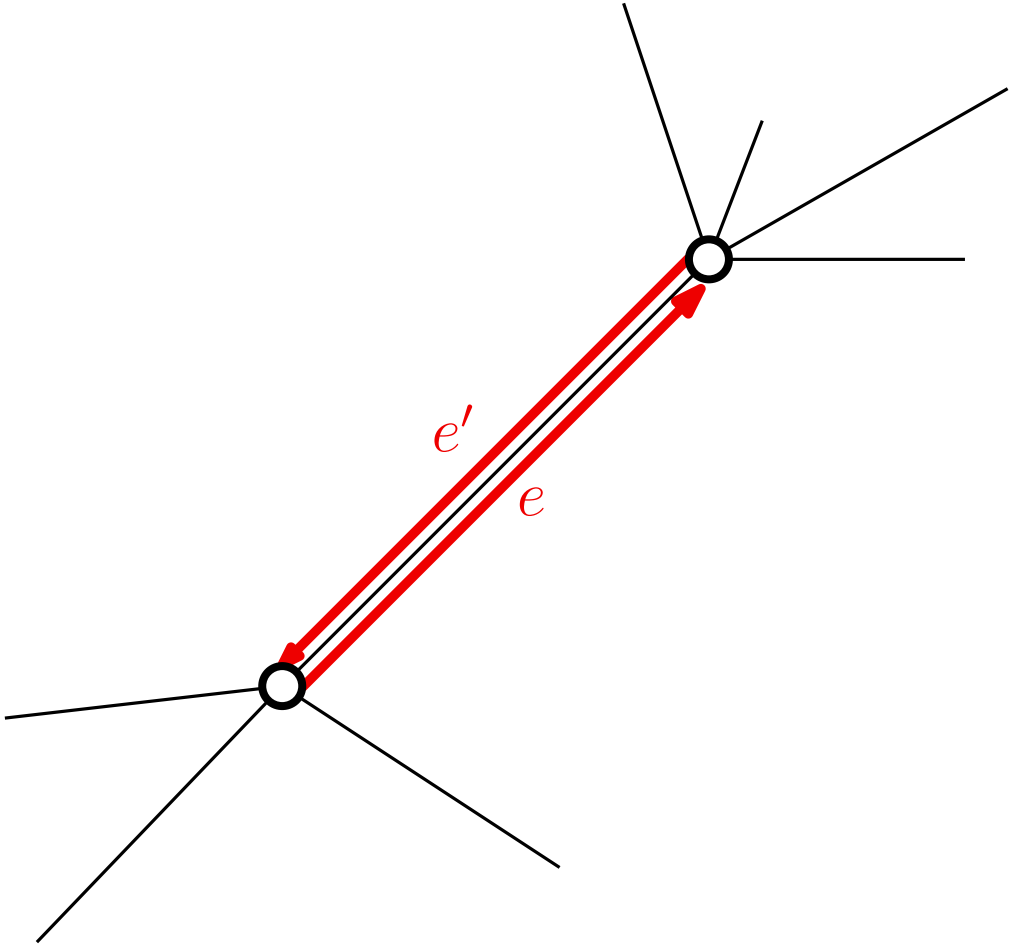
There are other DCEL variants, as for example:



Storing the polygon triangulation

DCEL

There are other DCEL variants, as for example:

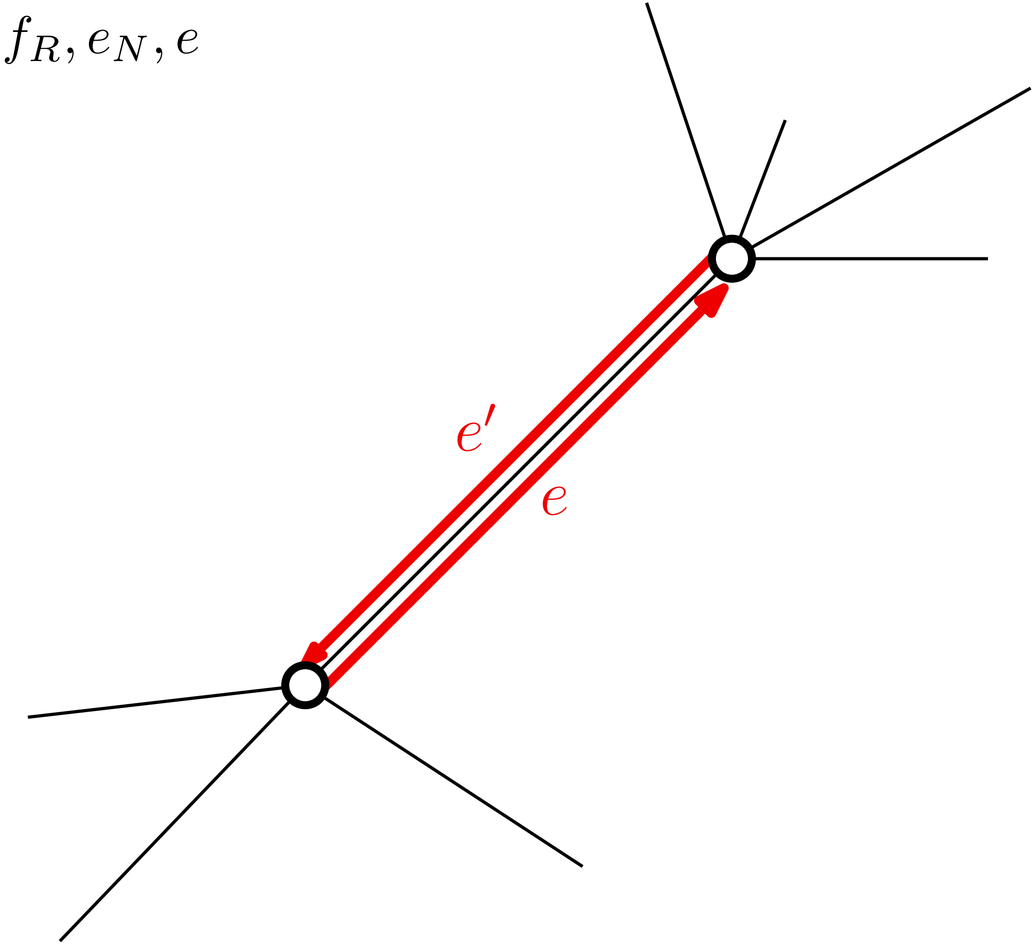


Storing the polygon triangulation

DCEL

There are other DCEL variants, as for example:

$$e \longrightarrow v_B, f_R, e_N, e'$$
$$e' \longrightarrow v_B, f_R, e_N, e$$

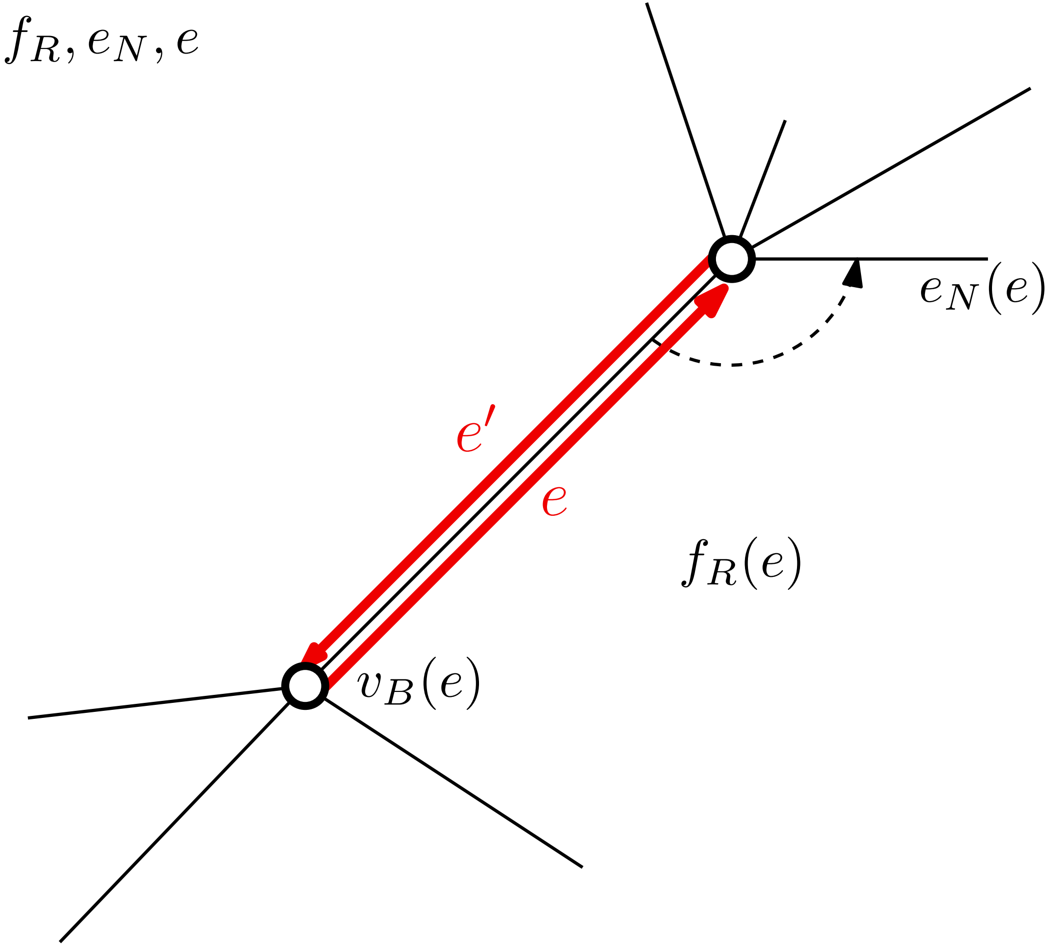


Storing the polygon triangulation

DCEL

There are other DCEL variants, as for example:

$$e \longrightarrow v_B, f_R, e_N, e'$$
$$e' \longrightarrow v_B, f_R, e_N, e$$

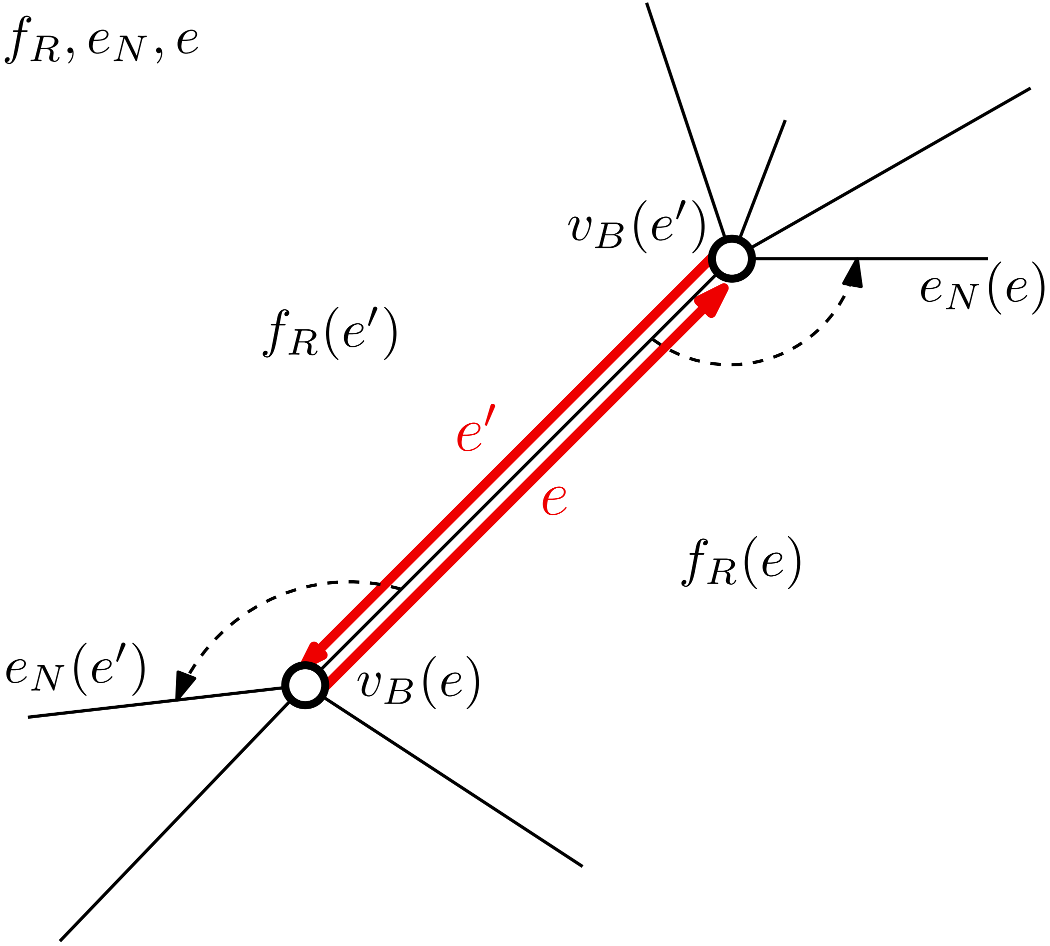


Storing the polygon triangulation

DCEL

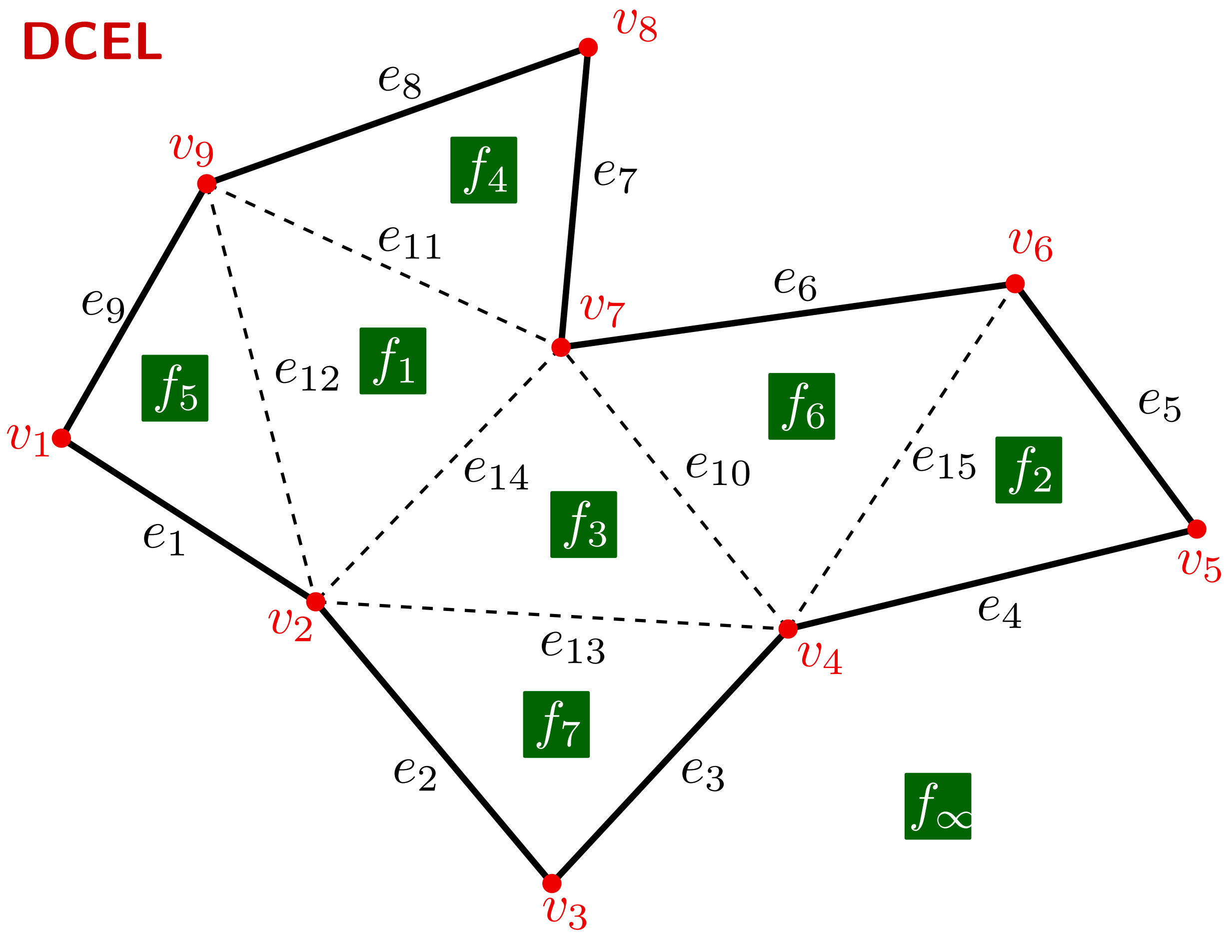
There are other DCEL variants, as for example:

$$e \longrightarrow v_B, f_R, e_N, e'$$
$$e' \longrightarrow v_B, f_R, e_N, e$$



Storing the polygon triangulation

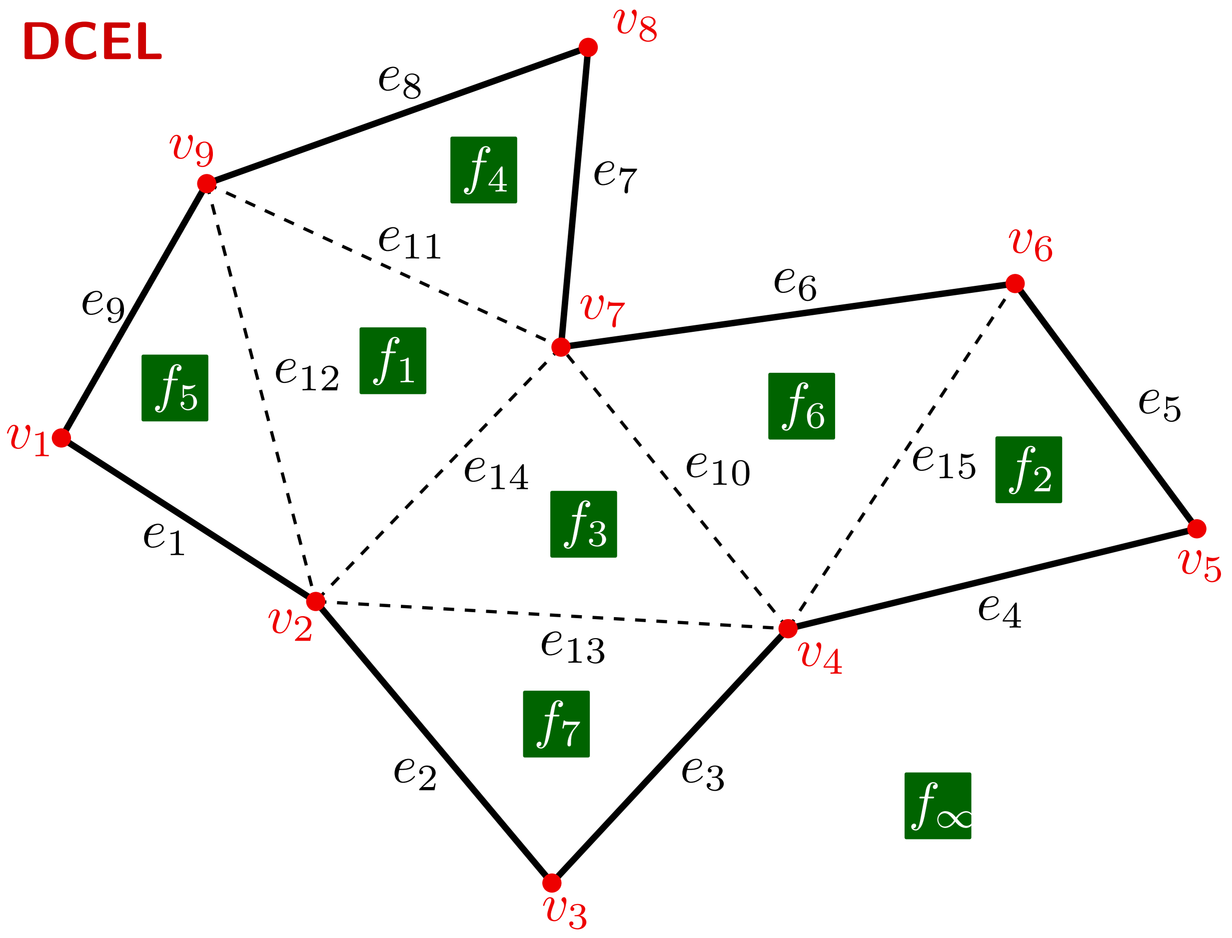
DCEL



Storing the polygon triangulation

e	v_B	v_E	f_L	f_R	e_P	e_N
1	1	2	5	∞	9	2
2	2	3	7	∞	13	3
3	4	3	∞	7	4	2
4	4	5	2	∞	15	5
5	5	6	2	∞	4	6
6	6	7	6	∞	15	7
7	7	8	4	∞	11	8
8	8	9	4	∞	7	9
9	9	1	5	∞	12	1
10	4	7	3	6	13	6
11	9	7	4	1	8	14
12	2	9	5	1	1	11
13	2	4	3	7	14	3
14	2	7	1	3	12	10
15	4	6	6	2	10	5

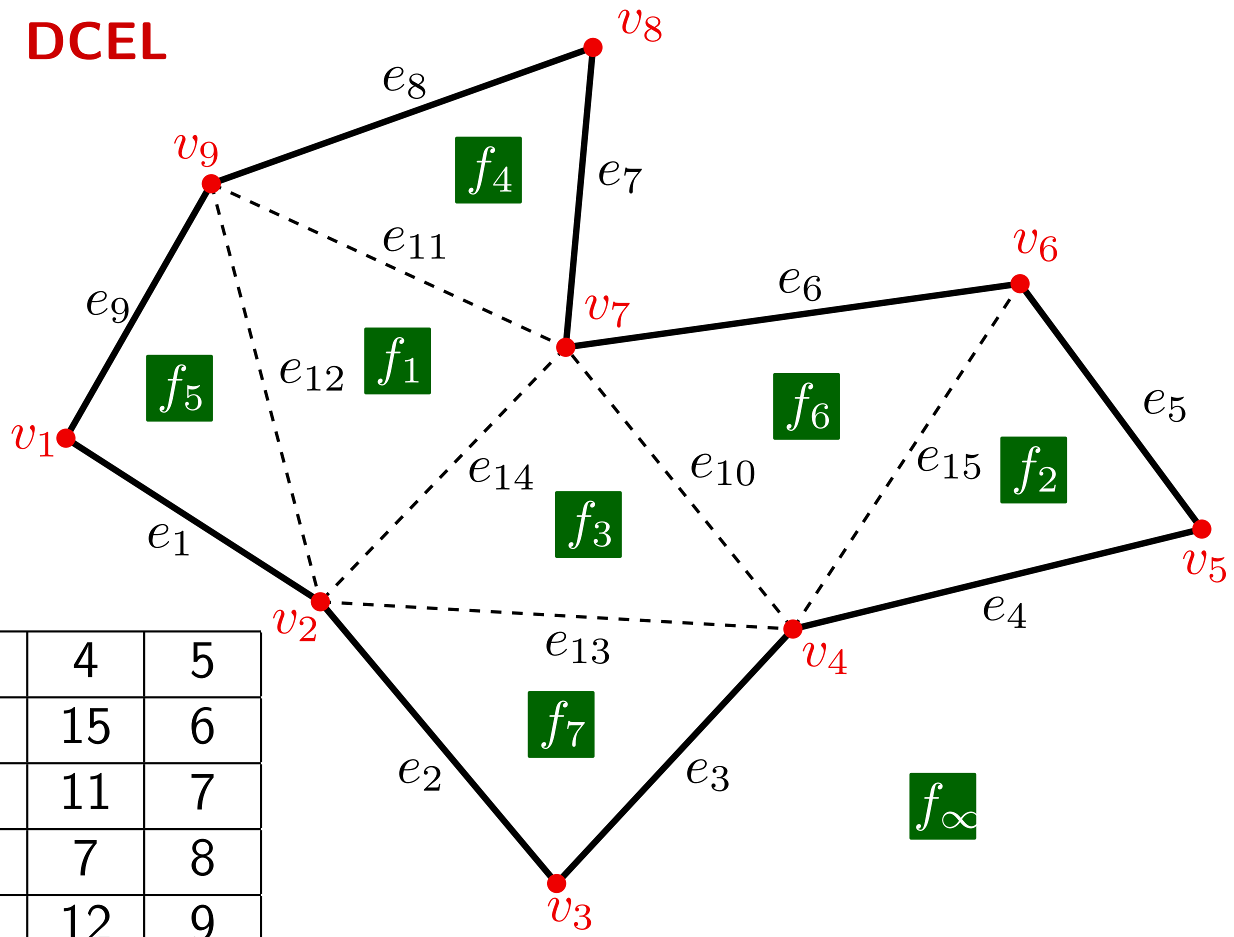
DCEL



Storing the polygon triangulation

e	v_B	f_R	e_N	e'
1	1	∞	2	1'
2	2	∞	3	2'
3	4	7	2	3'
4	4	∞	5	4'
5	5	∞	6	5'
6	6	∞	7	6'
7	7	∞	8	7'
8	8	∞	9	8'
9	9	∞	1	9'
10	4	6	6	10'
11	9	1	14	11'
12	2	1	11	12'
13	2	7	3	13'
14	2	3	10	14'
15	4	2	5	15'
1'	2	5	9	1
2'	3	7	13	2
3'	3	∞	4	3
4'	5	2	15	4
5'	6	2	4	5
6'	7	6	15	6
7'	8	4	11	7
8'	9	4	7	8
9'	1	5	12	9
10'	7	3	13	10
11'	7	4	8	11
12'	9	5	1	12
13'	4	3	14	13
14'	7	1	12	14
15'	6	6	10	15

DCEL



Storing the polygon triangulation

How to build the DCEL

Storing the polygon triangulation

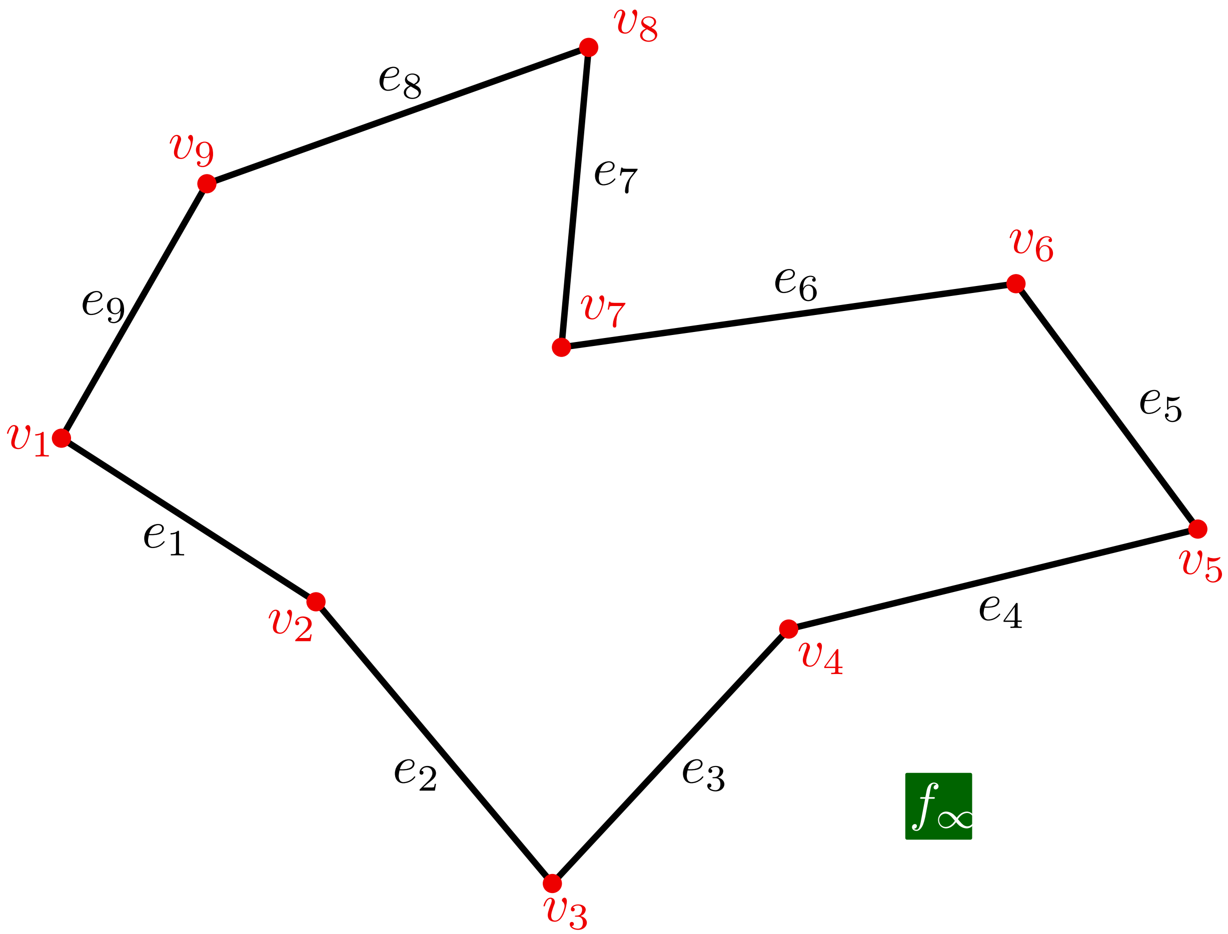
How to build the DCEL

Algorithm 1: substracting ears

Storing the polygon triangulation

How to build the DCEL

Algorithm 1: substracting ears

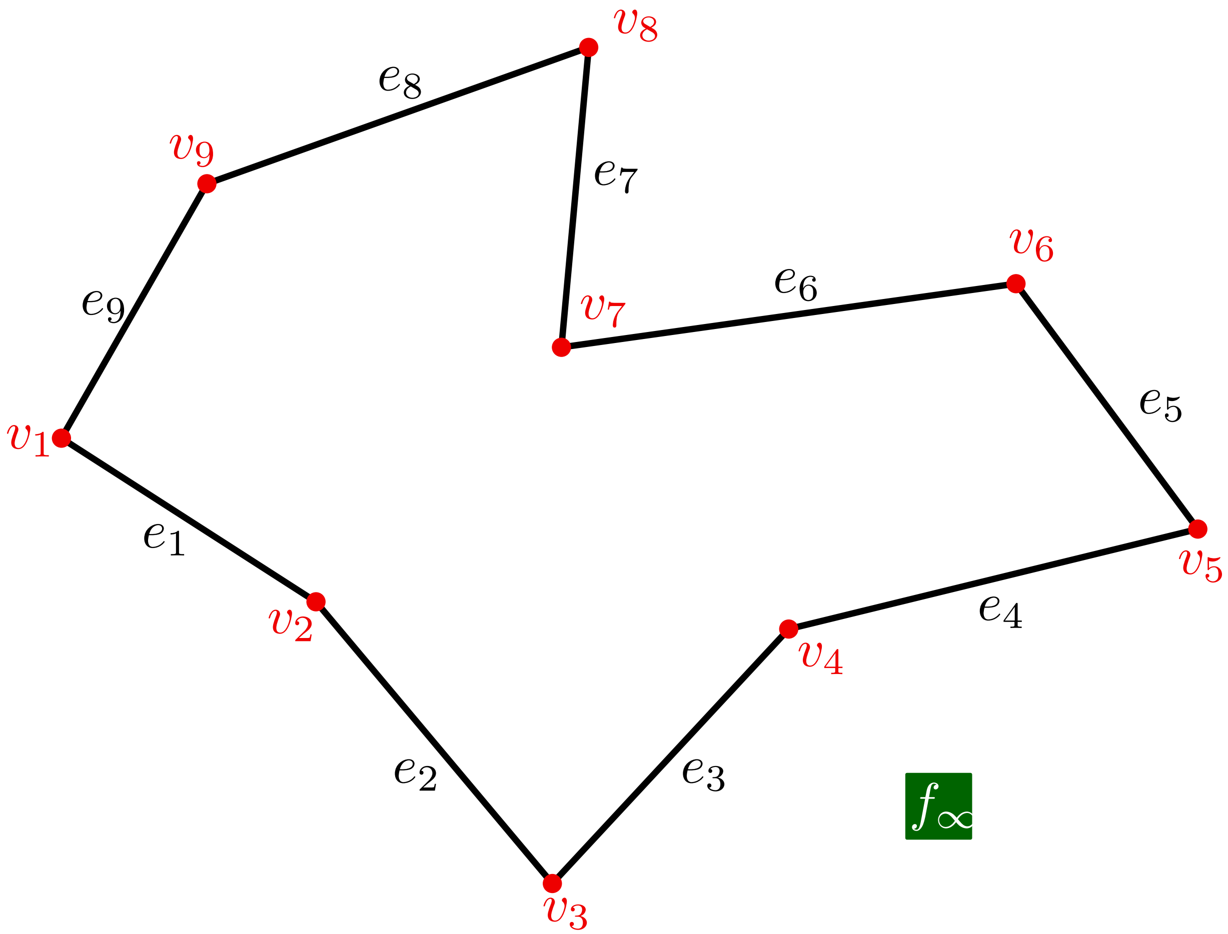


Storing the polygon triangulation

How to build the DCEL

Algorithm 1: substracting ears

Initialize



Storing the polygon triangulation

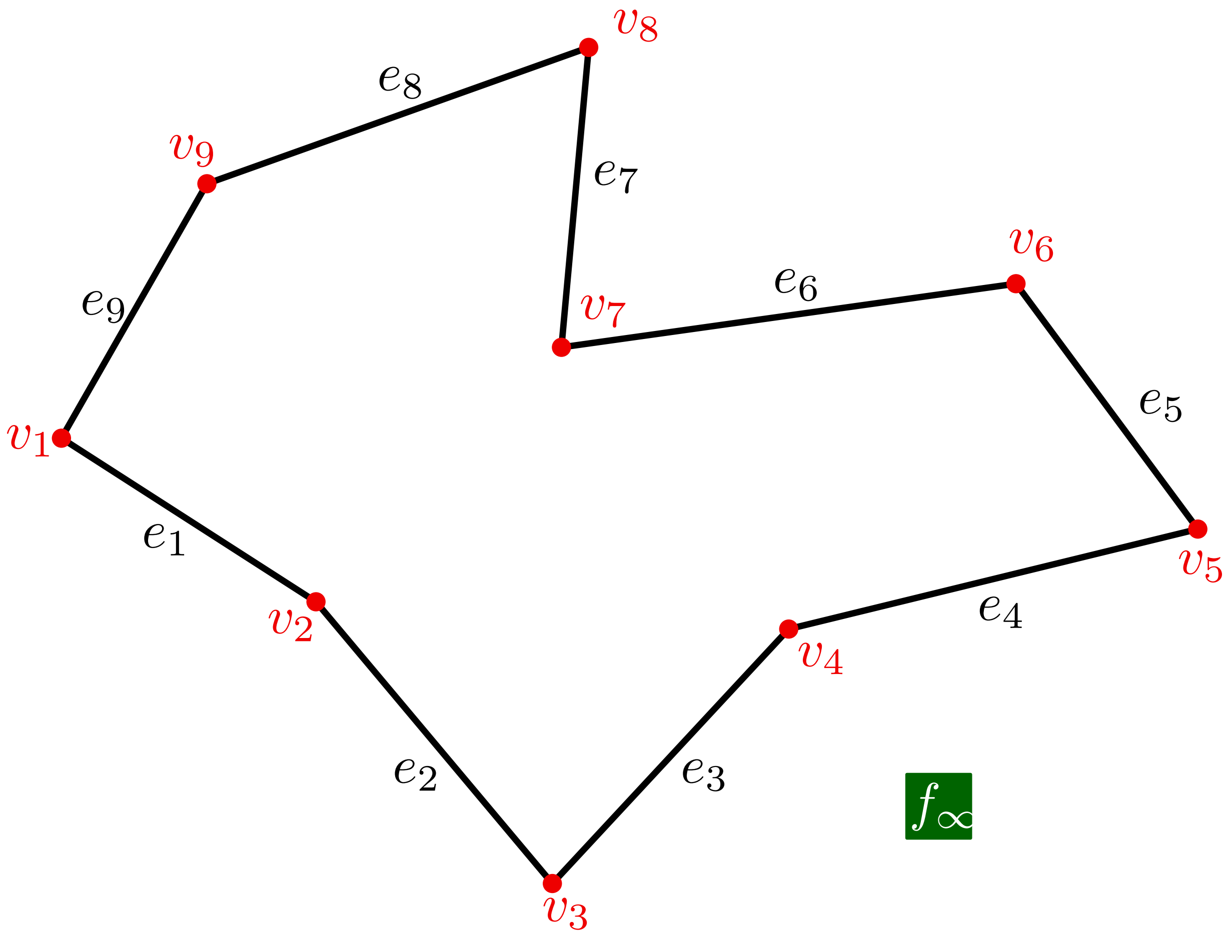
How to build the DCEL

Algorithm 1: substracting ears

Initialize

Table of vertices

v	x	y	e
1	x_1	y_1	1
2	x_2	y_2	2
3	x_3	y_3	3
4	x_4	y_4	4
5	x_5	y_5	5
6	x_6	y_6	6
7	x_7	y_7	7
8	x_8	y_8	8
9	x_9	y_9	9



Storing the polygon triangulation

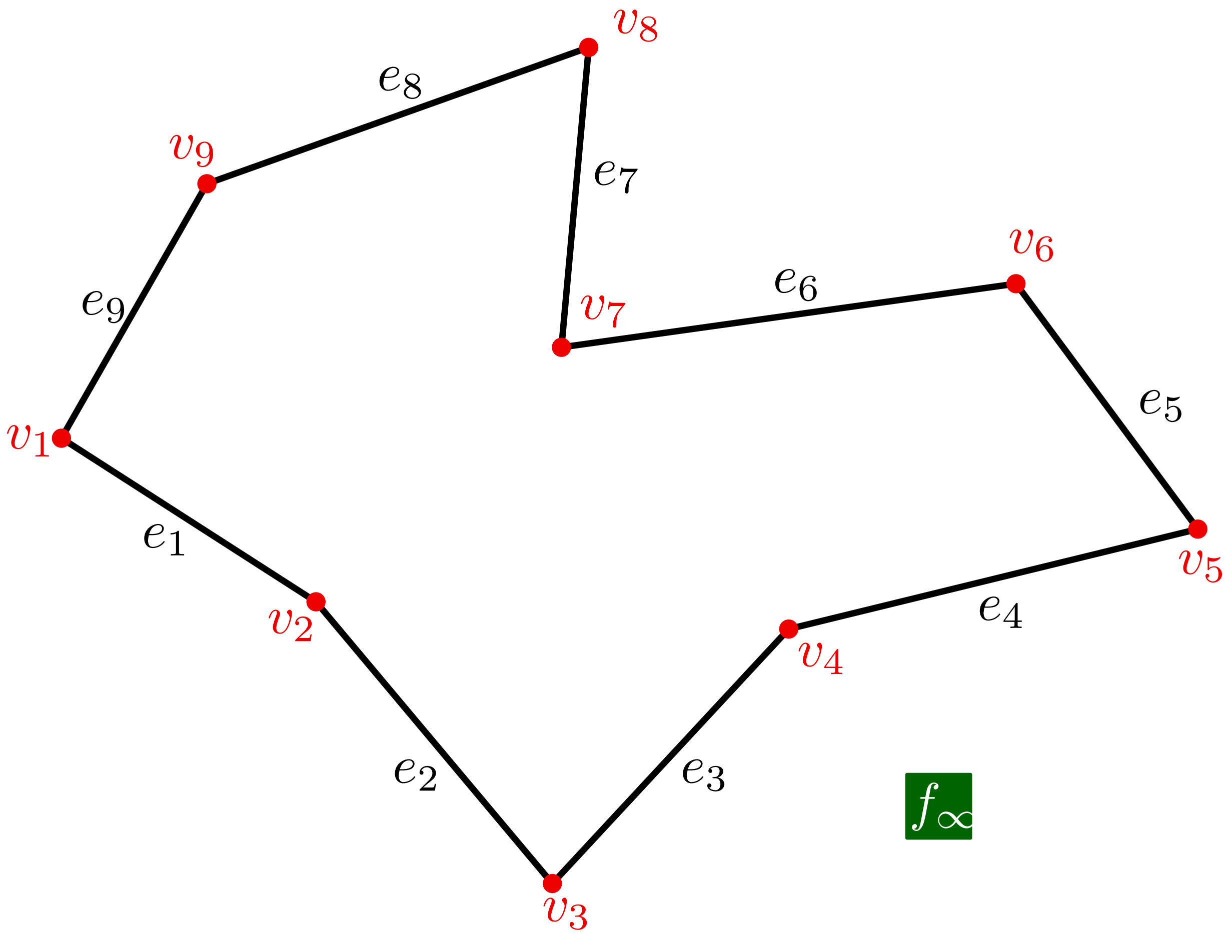
How to build the DCEL

Algorithm 1: substracting ears

Initialize

Table of faces

f	e
∞	9



Storing the polygon triangulation

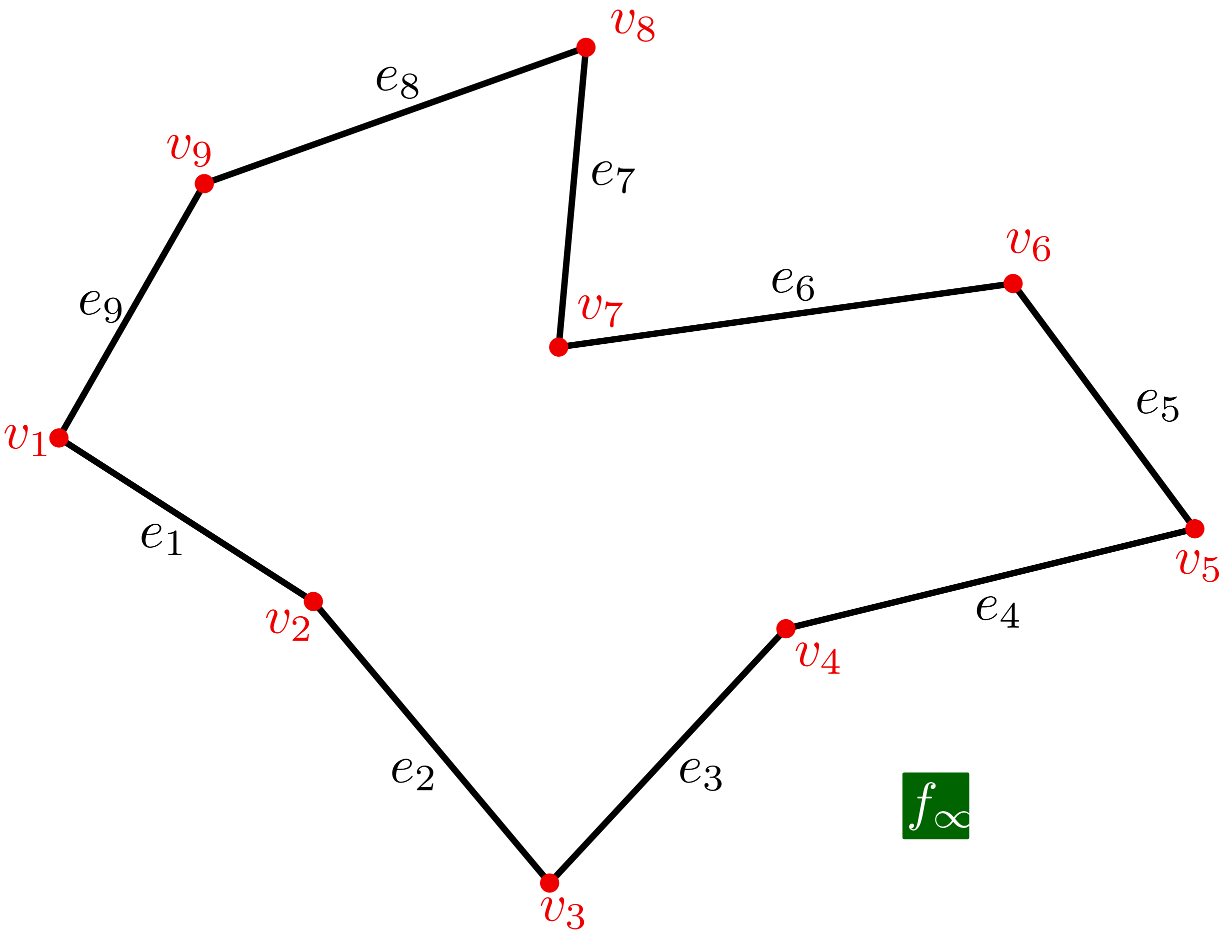
How to build the DCEL

Algorithm 1: substracting ears

Initialize

DCEL

e	v_B	v_E	f_L	f_R	e_P	e_N
1	1	2		∞		2
2	2	3		∞		3
3	3	4		∞		4
4	4	5		∞		5
5	5	6		∞		6
6	6	7		∞		7
7	7	8		∞		8
8	8	9		∞		9
9	9	1		∞		1



Storing the polygon triangulation

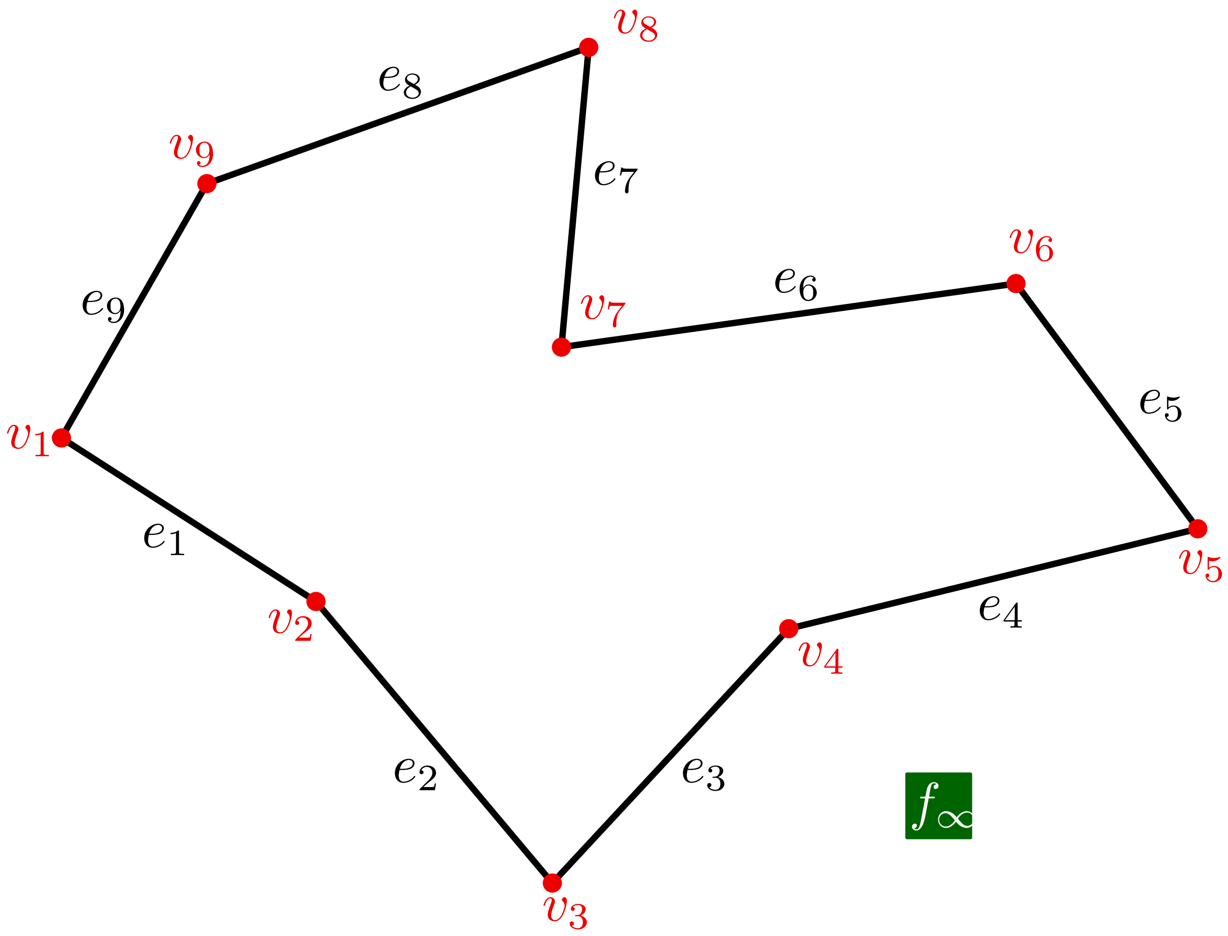
How to build the DCEL

Algorithm 1: substracting ears

Initialize

DCEL

e	v_B	v_E	f_L	f_R	e_P	e_N
1	1	2		∞		2
2	2	3		∞		3
3	4	3	∞		4	
4	4	5		∞		5
5	5	6		∞		6
6	6	7		∞		7
7	7	8		∞		8
8	8	9		∞		9
9	9	1		∞		1

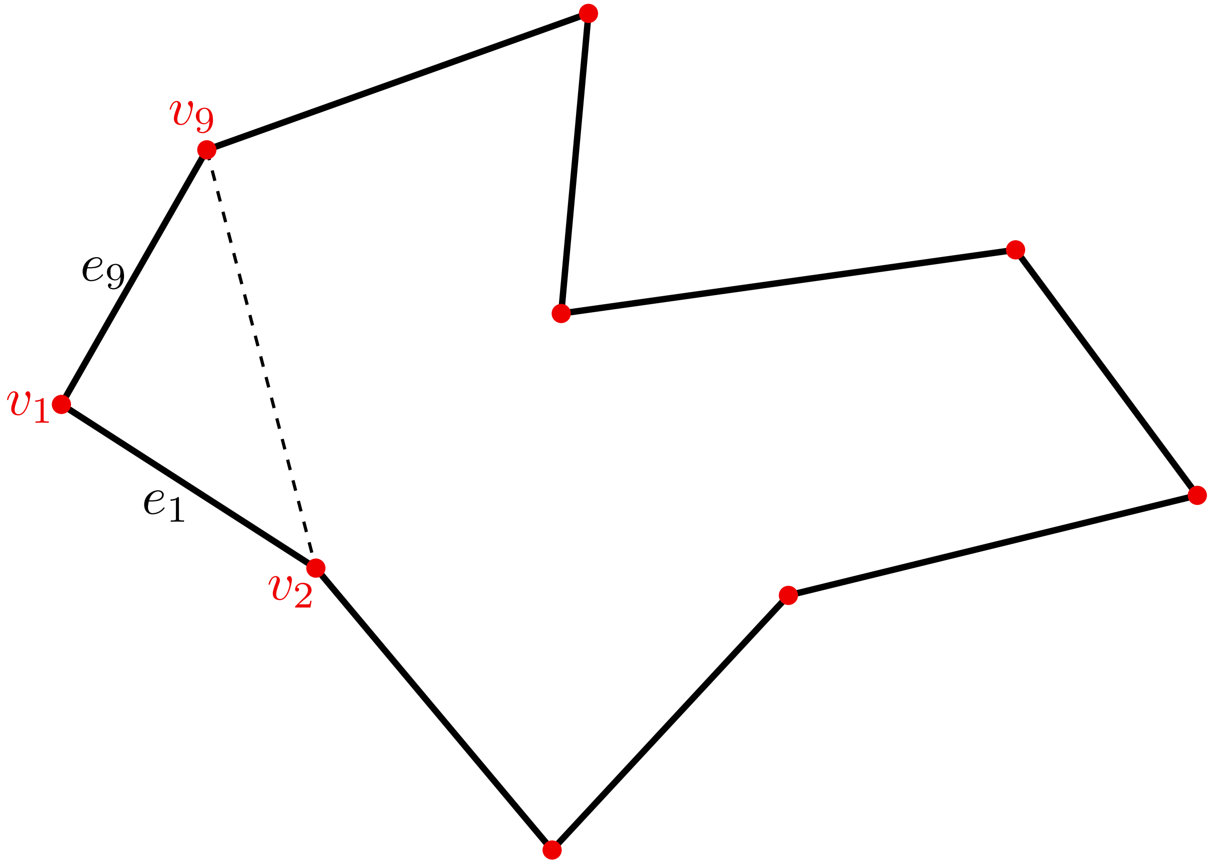


Storing the polygon triangulation

How to build the DCEL

Algorithm 1: substracting ears

Advance

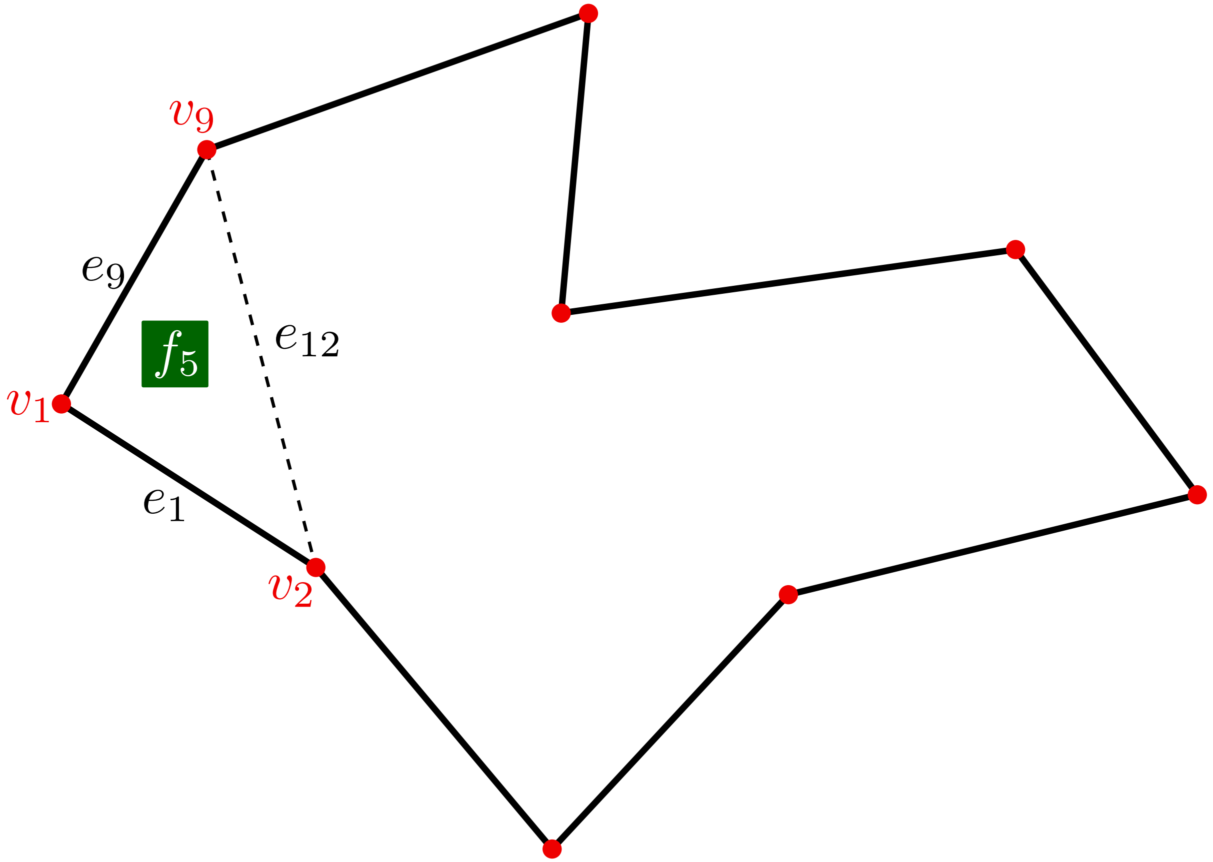


Storing the polygon triangulation

How to build the DCEL

Algorithm 1: substracting ears

Advance



Storing the polygon triangulation

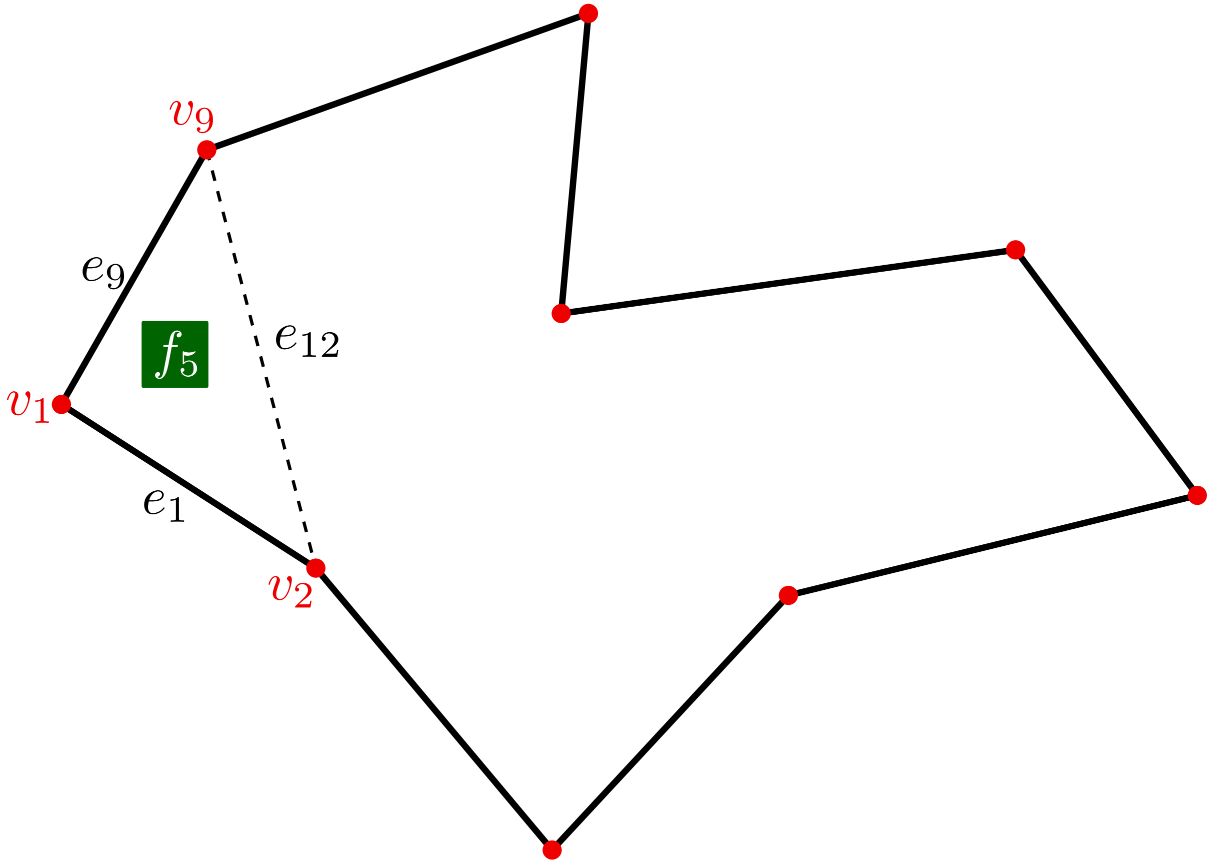
How to build the DCEL

Algorithm 1: substracting ears

Advance

Table of faces

f	e
5	9



Storing the polygon triangulation

How to build the DCEL

Algorithm 1: substracting ears

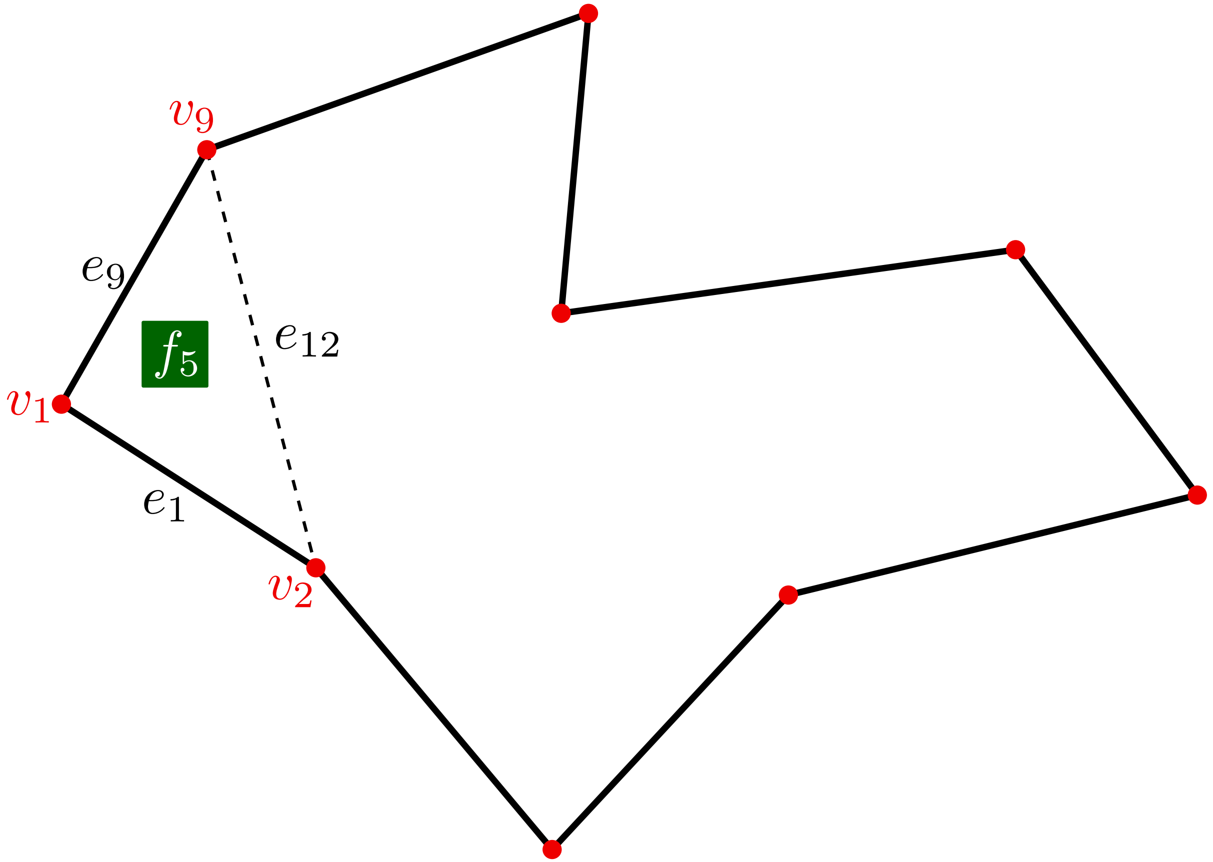
Advance

Table of faces

f	e
5	9

DCEL

e	v_B	v_E	f_L	f_R	e_P	e_N
1	1	2	5	∞	9	2
9	9	1	5	∞	12	1
12	2	9	5		1	



Storing the polygon triangulation

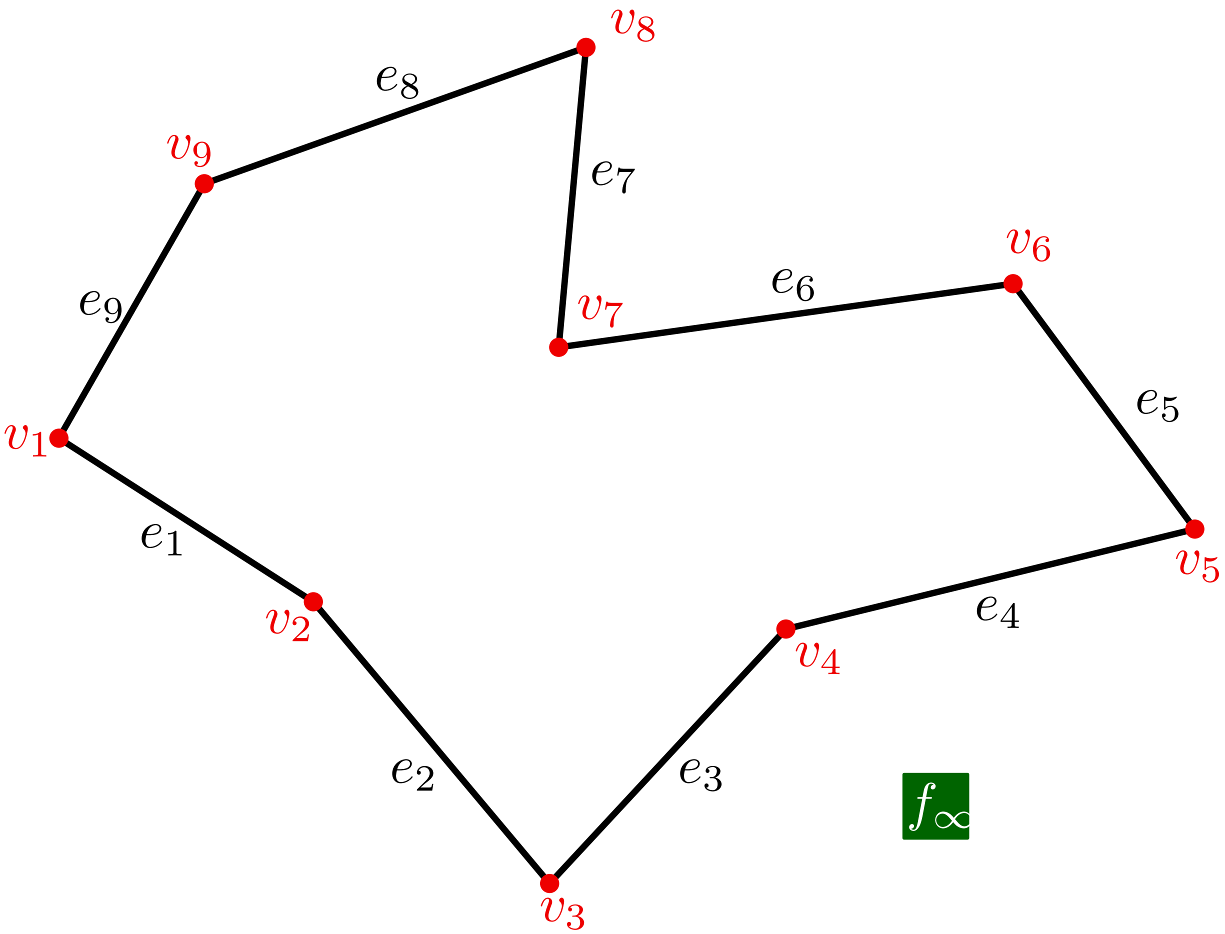
How to build the DCEL

Algorithm 2: inserting diagonals

Storing the polygon triangulation

How to build the DCEL

Algorithm 2: inserting diagonals

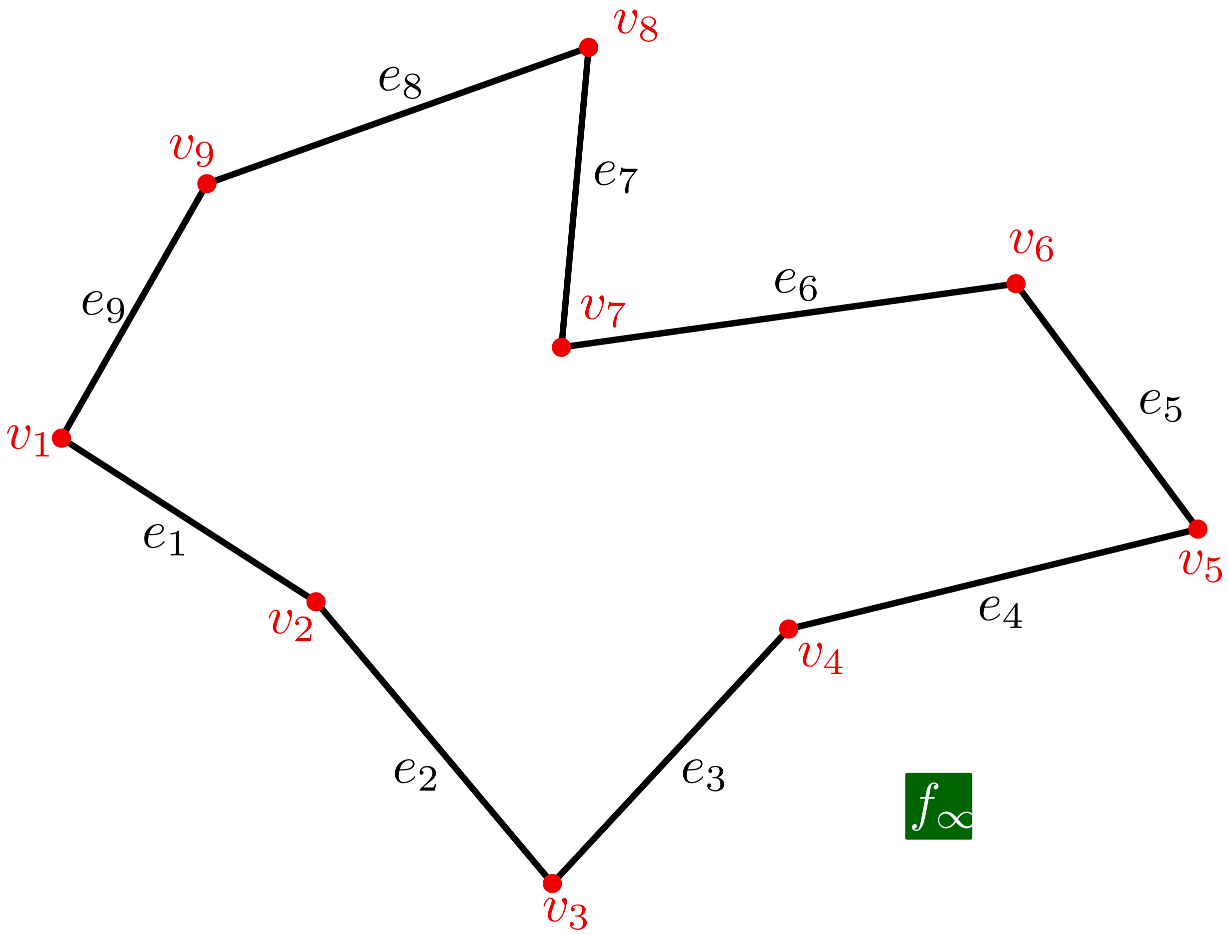


Storing the polygon triangulation

How to build the DCEL

Algorithm 2: inserting diagonals

Initialize



Storing the polygon triangulation

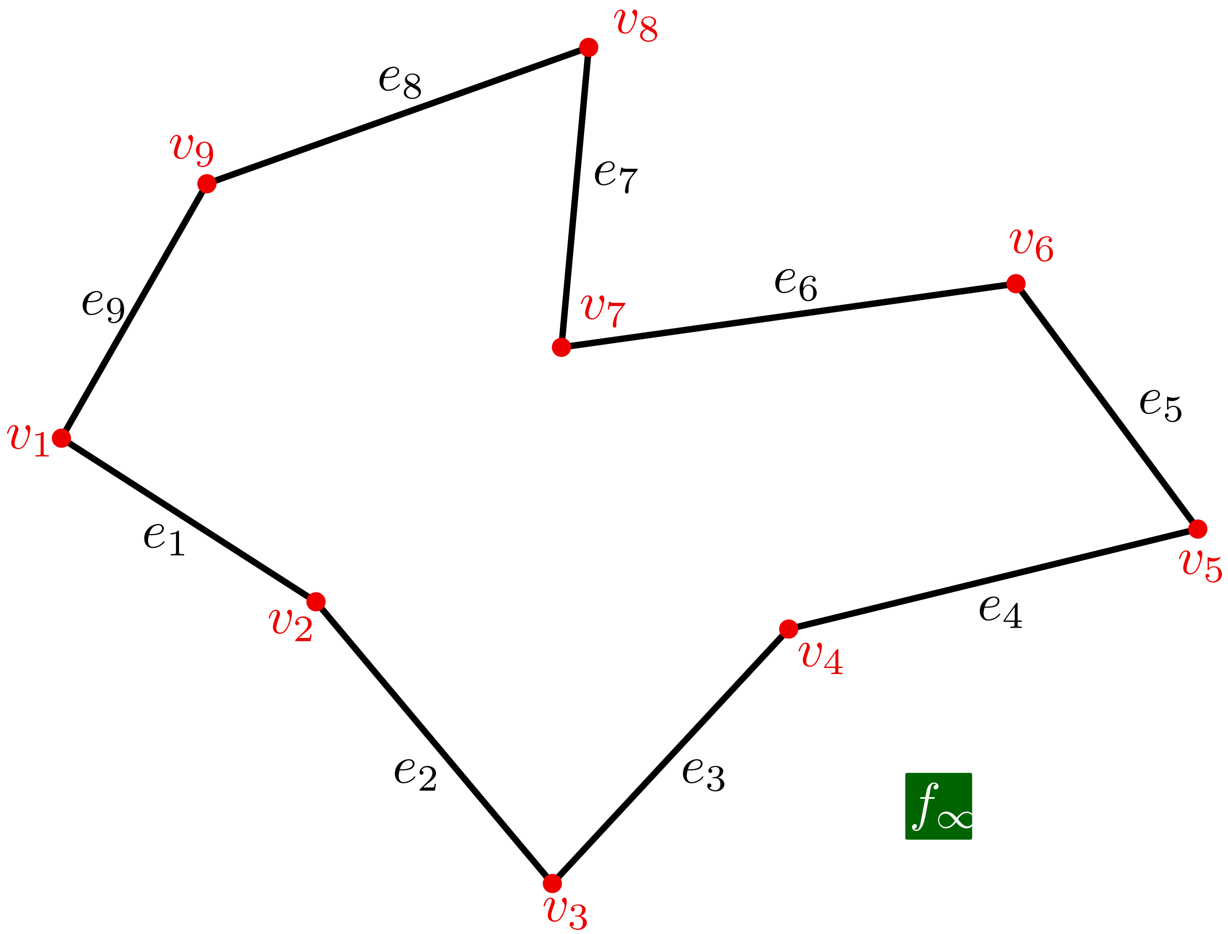
How to build the DCEL

Algorithm 2: inserting diagonals

Initialize

Table of vertices

v	x	y	e
1	x_1	y_1	1
2	x_2	y_2	2
3	x_3	y_3	3
4	x_4	y_4	4
5	x_5	y_5	5
6	x_6	y_6	6
7	x_7	y_7	7
8	x_8	y_8	8
9	x_9	y_9	9



Storing the polygon triangulation

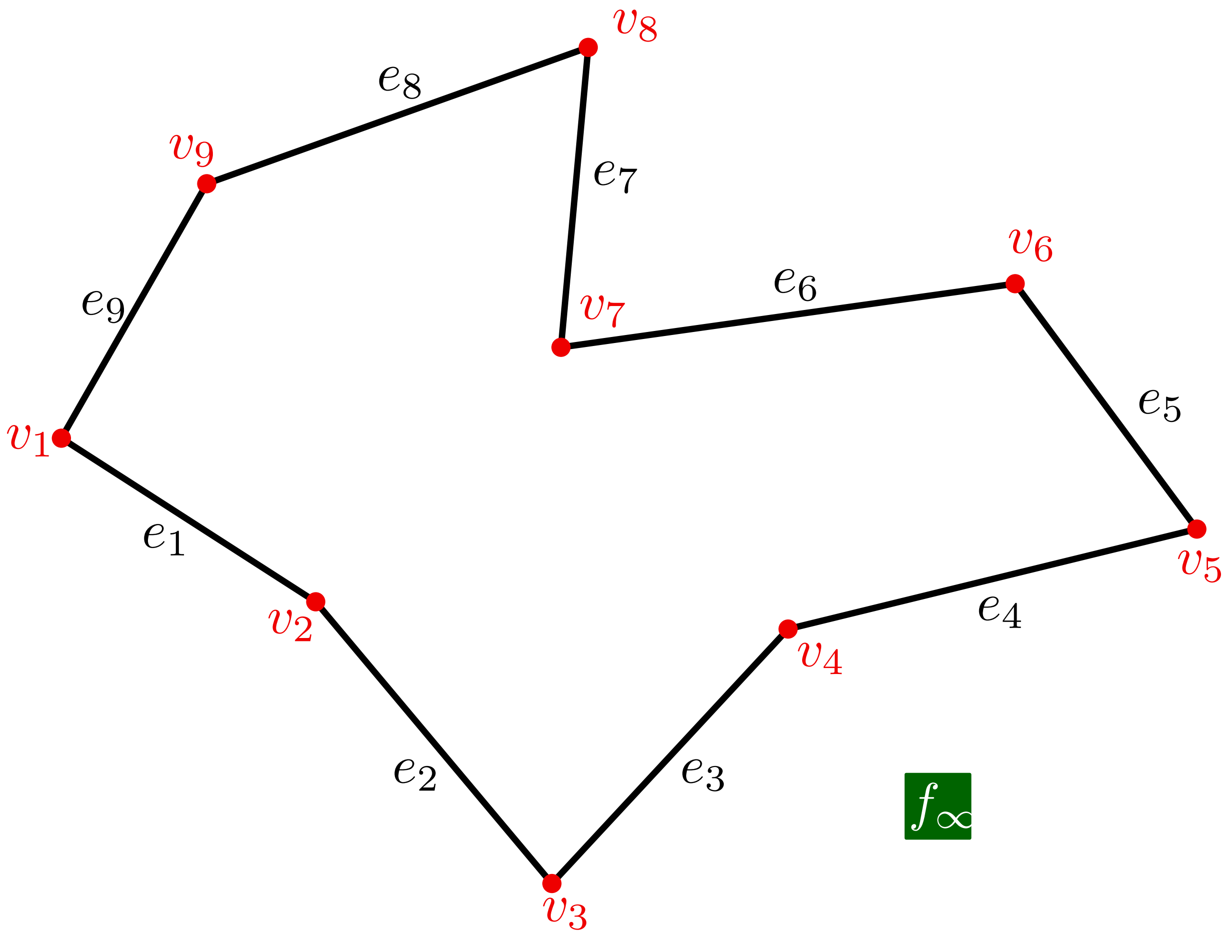
How to build the DCEL

Algorithm 2: inserting diagonals

Initialize

Table of faces

f	e
∞	9

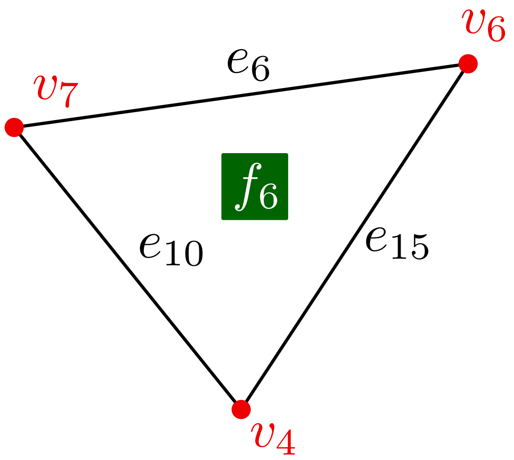


Storing the polygon triangulation

How to build the DCEL

Algorithm 2: inserting diagonals

Base step



Storing the polygon triangulation

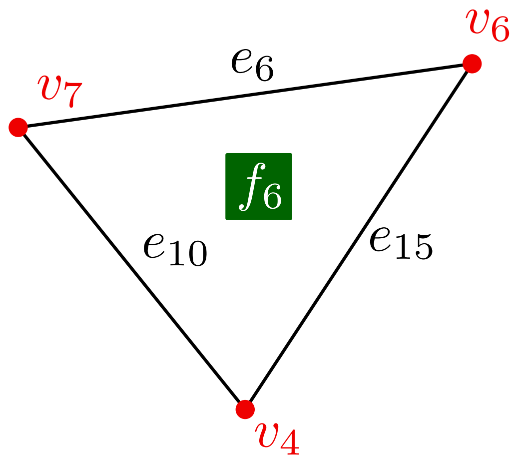
How to build the DCEL

Algorithm 2: inserting diagonals

Base step

Table of faces

f	e
6	10



Storing the polygon triangulation

How to build the DCEL

Algorithm 2: inserting diagonals

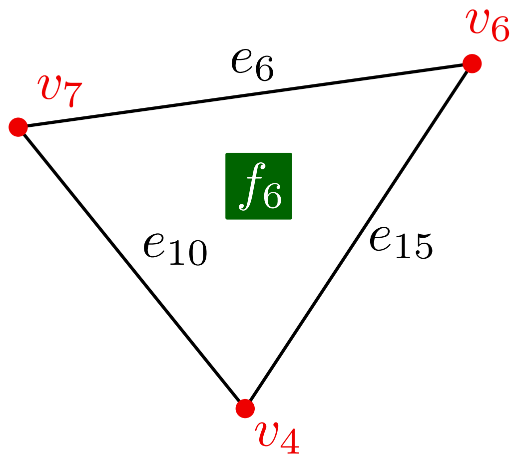
Base step

Table of faces

f	e
6	10

DCEL

e	v_B	v_E	f_L	f_R	e_P	e_N
6	6	7	6	∞	15	10
10	7	4	6	∞	6	15
15	4	6	6	∞	10	6

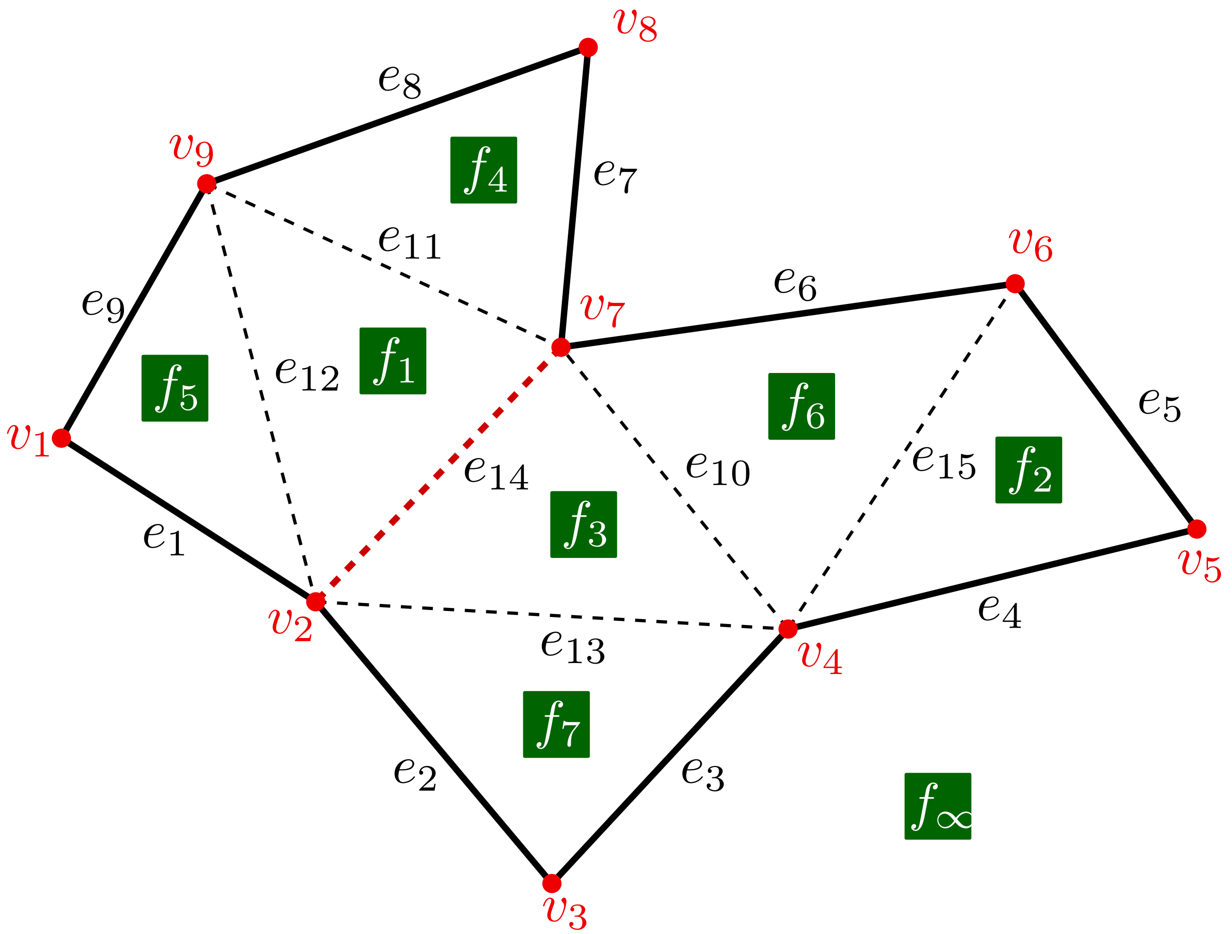


Storing the polygon triangulation

How to build the DCEL

Algorithm 2: inserting diagonals

Merge step



Storing the polygon triangulation

How to build the DCEL

Algorithm 2: inserting diagonals

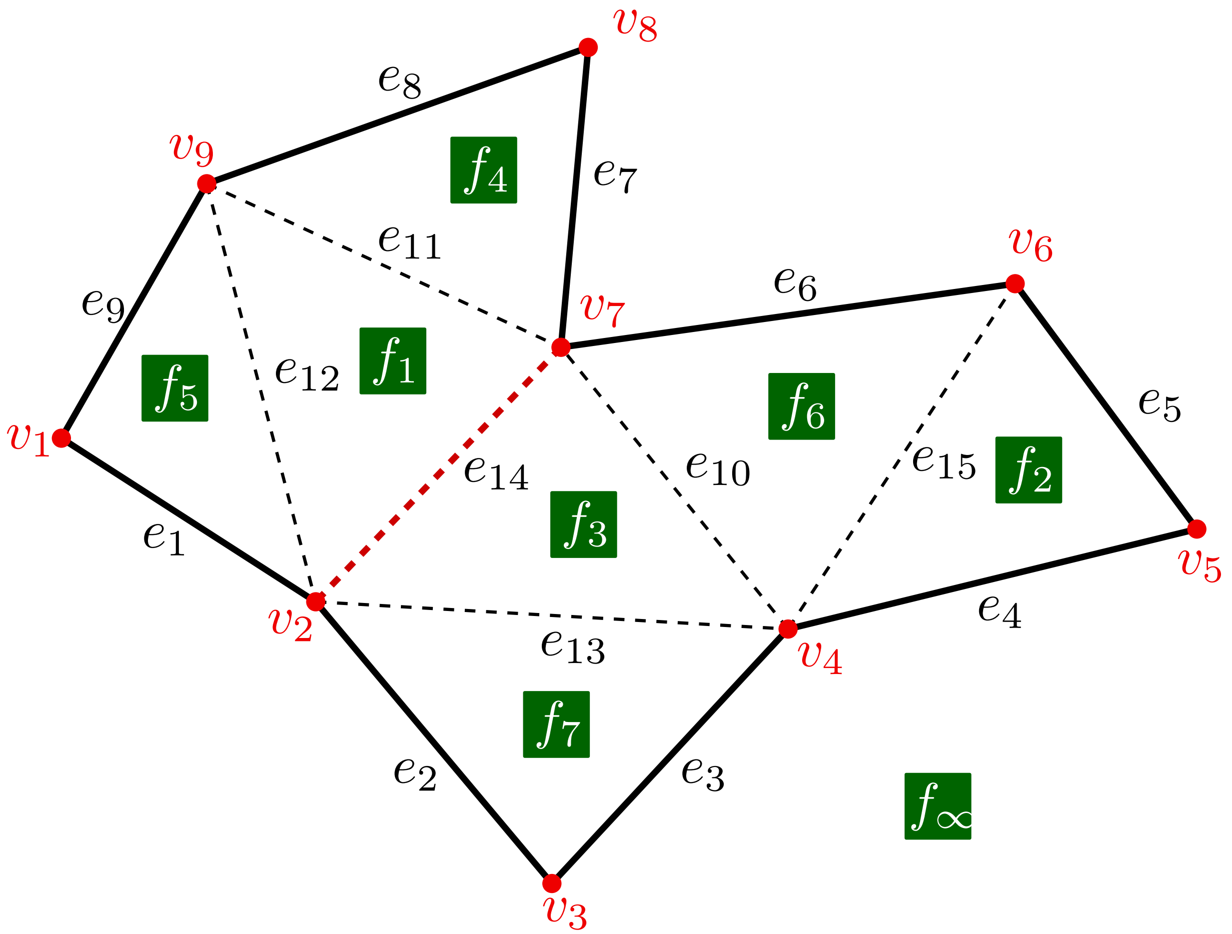
Merge step

DCEL 1

e	v_B	v_E	f_L	f_R	e_P	e_N
1	1	2	5	∞	9	14
14	2	7	1	∞	12	7

DCEL 2

e	v_B	v_E	f_L	f_R	e_P	e_N
6	6	7	6	∞	15	14
14	2	7	∞	3	2	10



Storing the polygon triangulation

How to build the DCEL

Algorithm 2: inserting diagonals

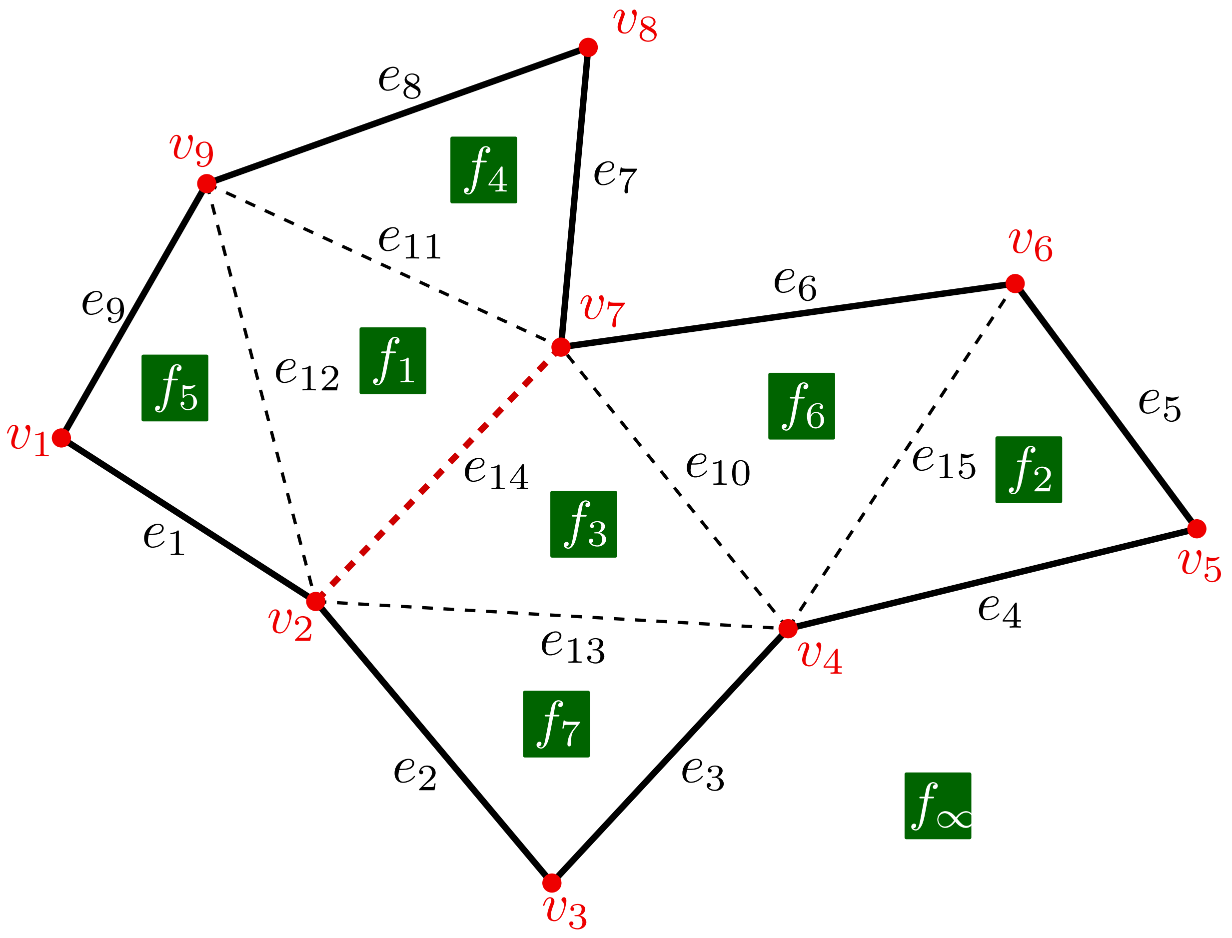
Merge step

DCEL 1

e	v_B	v_E	f_L	f_R	e_P	e_N
1	1	2	5	∞	9	~~14~~
14	2	7	1	~~∞~~	12	~~7~~

DCEL 2

e	v_B	v_E	f_L	f_R	e_P	e_N
6	6	7	6	∞	15	~~14~~
14	2	7	~~∞~~	3	~~2~~	10



Storing the polygon triangulation

How to build the DCEL

Algorithm 2: inserting diagonals

Merge step

DCEL 1

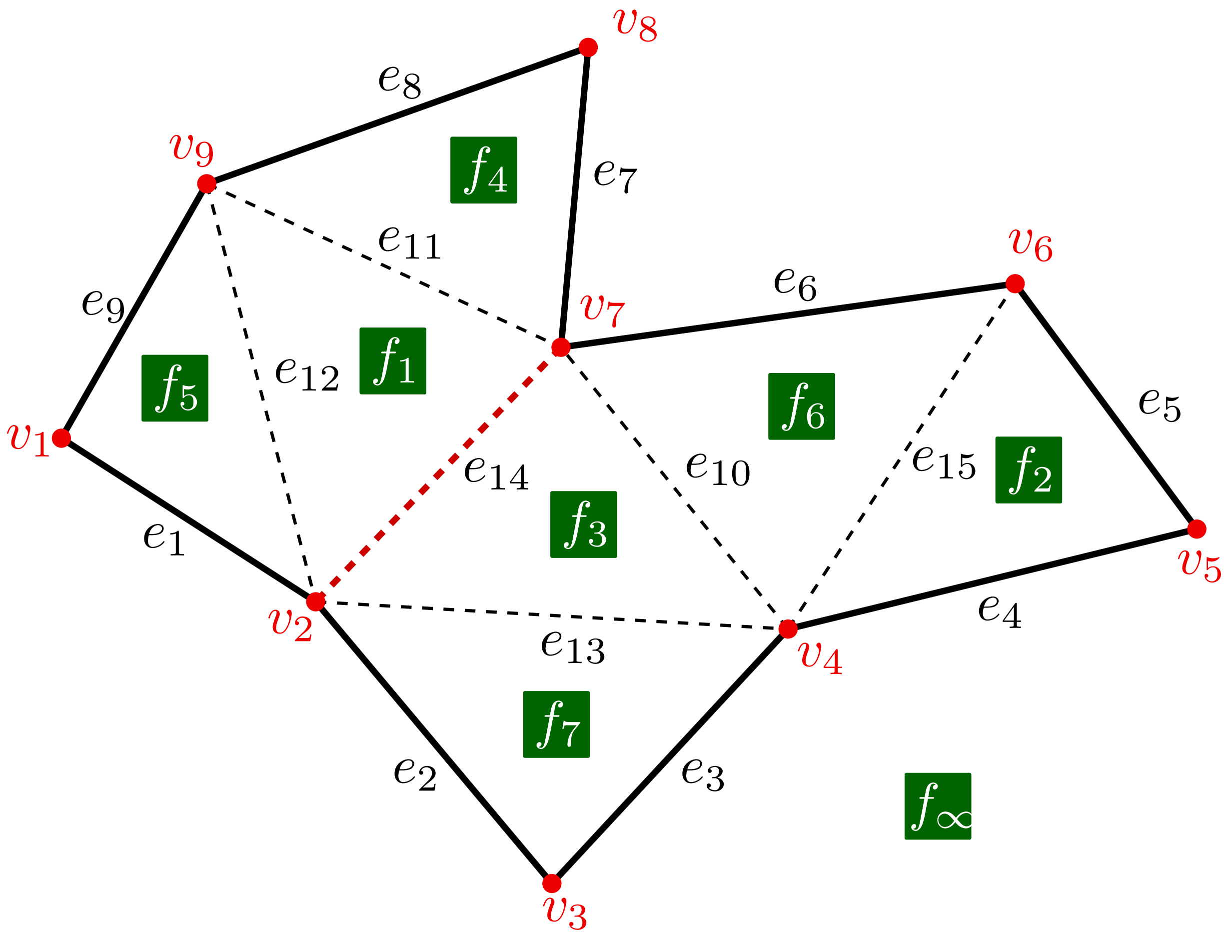
e	v_B	v_E	f_L	f_R	e_P	e_N
1	1	2	5	∞	9	~~14~~
14	2	7	1	~~∞~~	12	~~7~~

DCEL 2

e	v_B	v_E	f_L	f_R	e_P	e_N
6	6	7	6	∞	15	~~14~~
14	2	7	~~∞~~	3	~~2~~	10

Merged DCEL

e	v_B	v_E	f_L	f_R	e_P	e_N
1	1	2	5	∞	9	2
6	6	7	6	∞	15	7
14	2	7	1	3	12	10



Storing the polygon triangulation

How to build the DCEL

Algorithm 3:

Storing the polygon triangulation

How to build the DCEL

Algorithm 3:

1. Decompose into monotone polygons
2. Triangulate monotone pieces

Storing the polygon triangulation

How to build the DCEL

Algorithm 3:

1. Decompose into monotone polygons
2. Triangulate monotone pieces

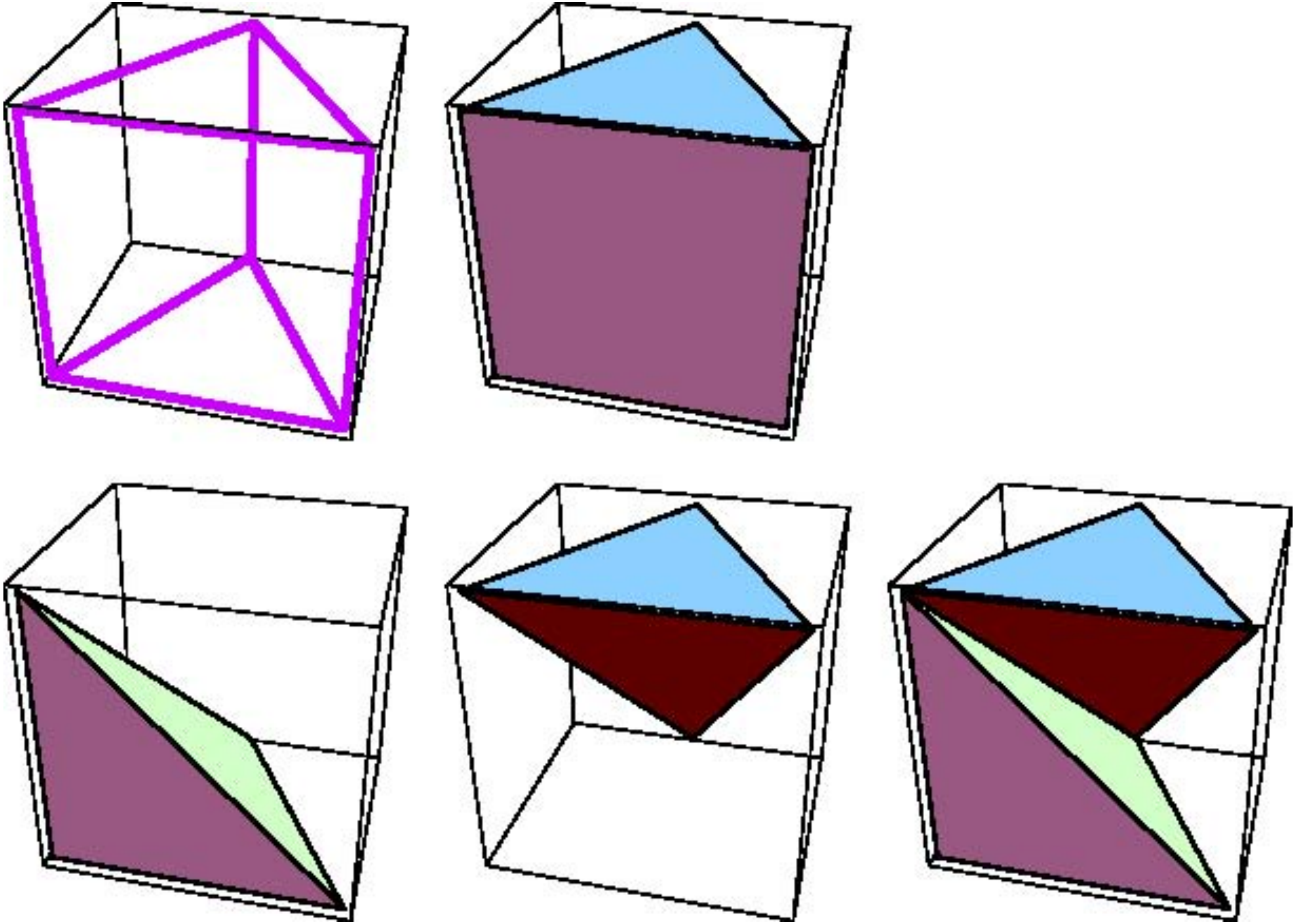
Computing the DCEL is done by combining previous strategies:

- Separating ears for triangulating each monotone subpolygon.
- Merging DCELs for putting together the monotone pieces.

WHAT HAPPENS IN 3D?

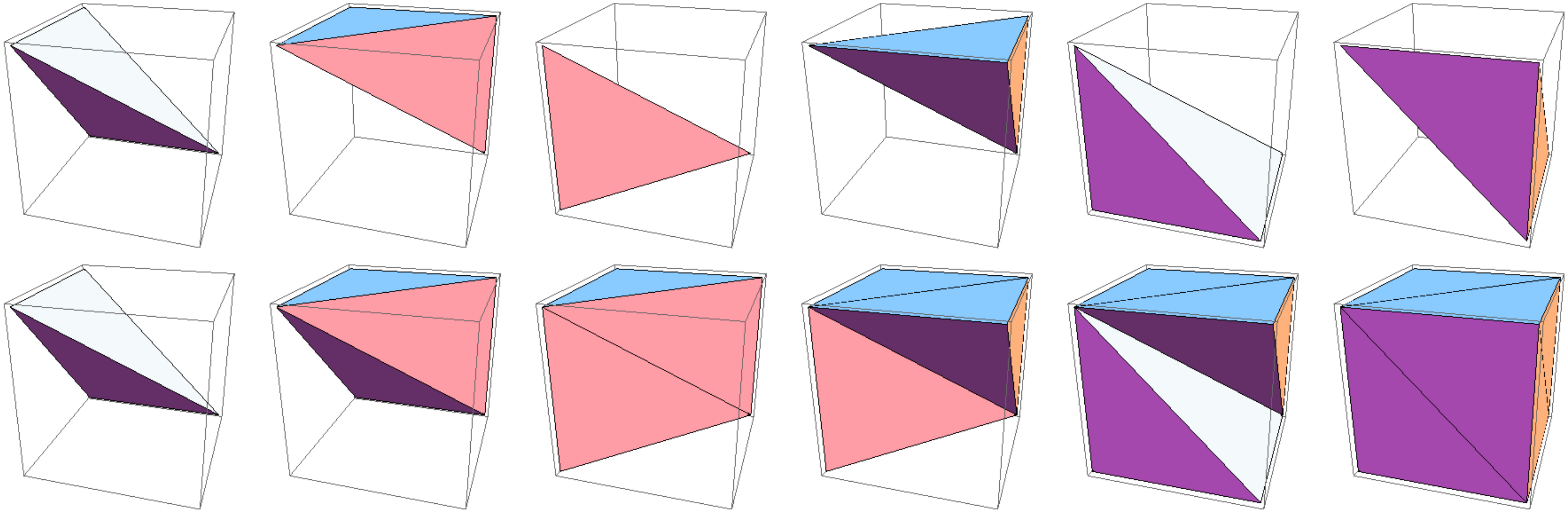
WHAT HAPPENS IN 3D?

A polyhedron that can be *tetrahedralized*:



WHAT HAPPENS IN 3D?

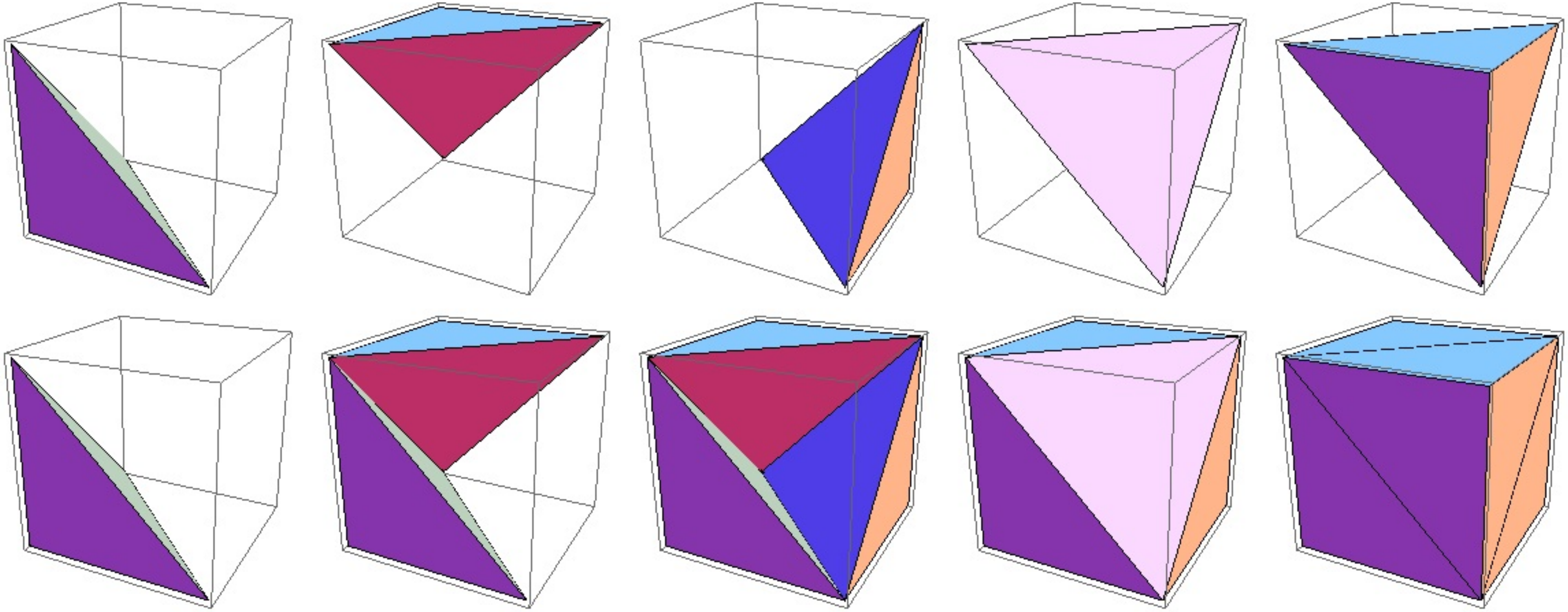
A cube can be decomposed into 6 tetrahedra...



WHAT HAPPENS IN 3D?

A cube can be decomposed into 6 tetrahedra...

but also into 5!



WHAT HAPPENS IN 3D?

A polyhedron that cannot be tetrahedralized:

Schönhardt polyhedron

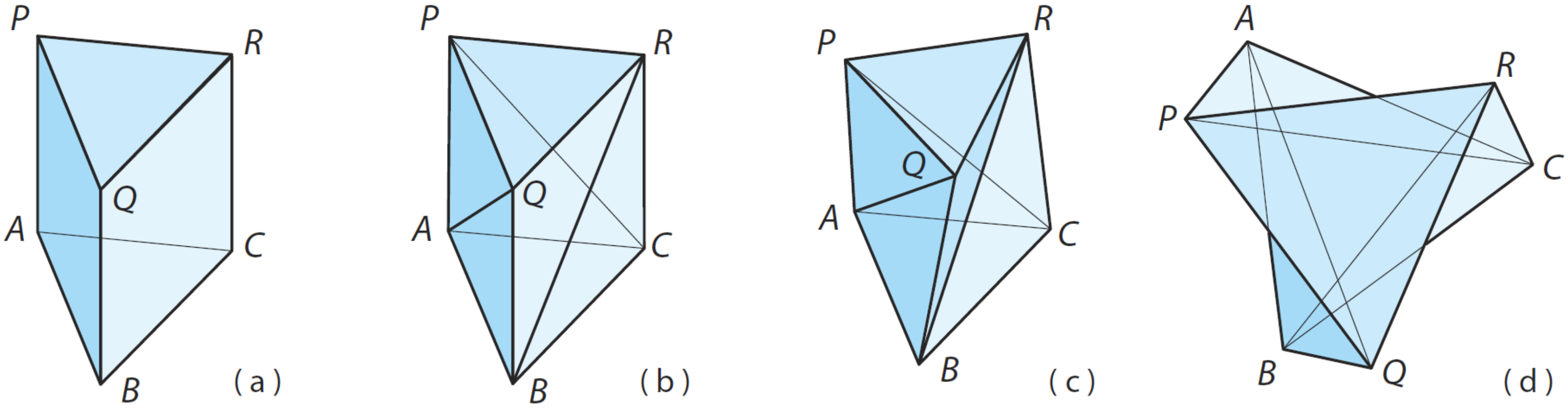


Figure from the book by Devadoss and O'Rourke

Smallest polyhedron that cannot be tetrahedralized

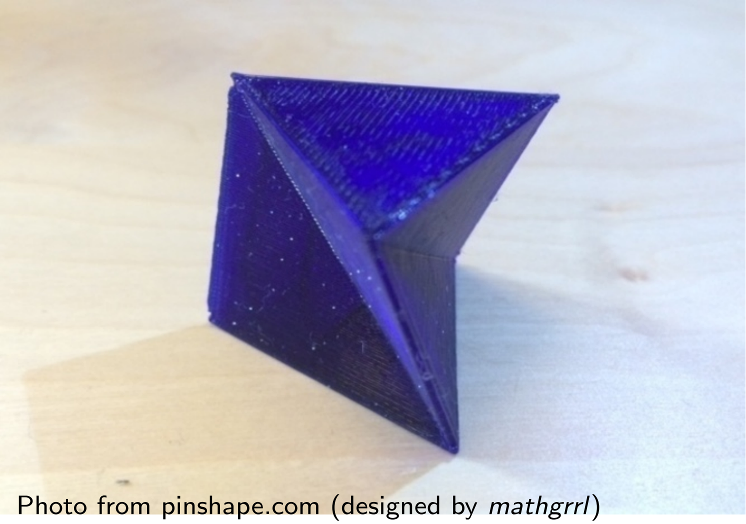


Photo from pinshape.com (designed by *mathgrrl*)

TRIANGULATING POLYGONS

TO LEARN MORE

- J. O'Rourke, **Computational Geometry in C (2nd ed.)**, Cambridge University Press, 1998.
- M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, **Computational Geometry: Algorithms and Applications (3rd rev. ed.)**, Springer, 2008.

TRIANGULATING POLYGONS

TO LEARN MORE

- J. O'Rourke, **Computational Geometry in C (2nd ed.)**, Cambridge University Press, 1998.
- M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, **Computational Geometry: Algorithms and Applications (3rd rev. ed.)**, Springer, 2008.

A NICE APPLICATION

The art gallery theorem

- J. O'Rourke, **Art Gallery Theorems and Algorithms**, Oxford University Press, 1987. http://maven.smith.edu/∼orourke/books/ArtGalleryTheorems/art.html