ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

The Minimum Weight Triangulation Problem with Few Inner Points

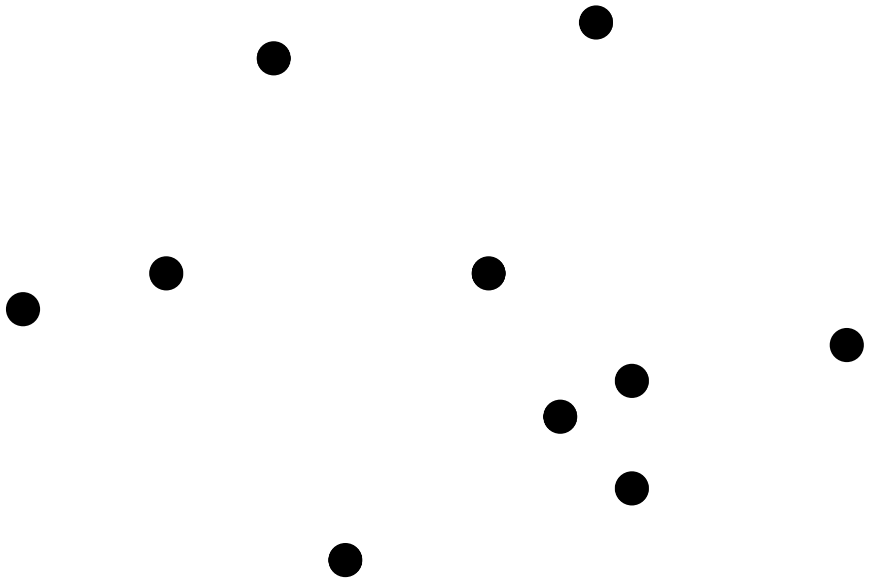
Michael Hoffmann ETH Zurich
Yoshio Okamoto ETH Zurich

September 16, 2004
International Workshop on Exact and Parameterized Computation (IWPEC)
Radisson SAS Royal Hotel, Bergen, Norway

Triangulation

S a set of points in the plane, no three points lying on a line

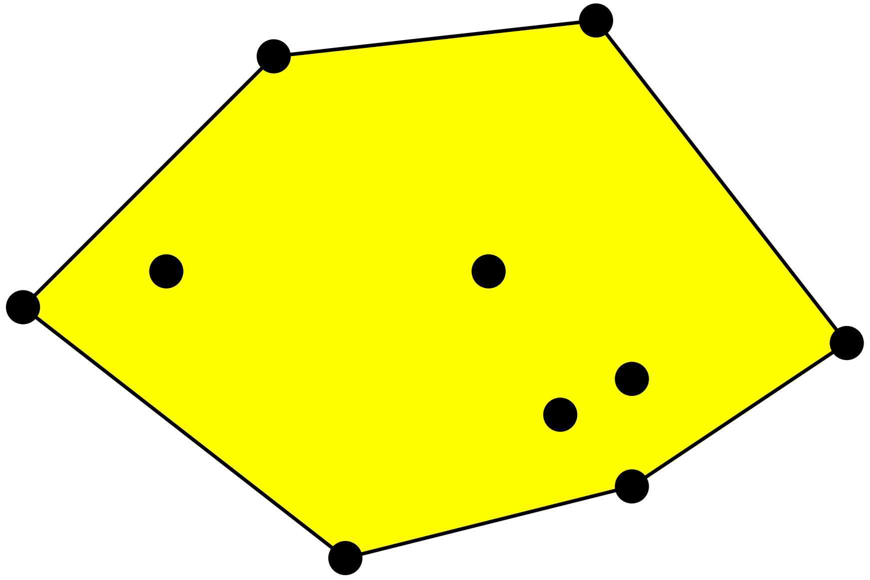
A **triangulation** of S:



Triangulation

S a set of points in the plane, no three points lying on a line

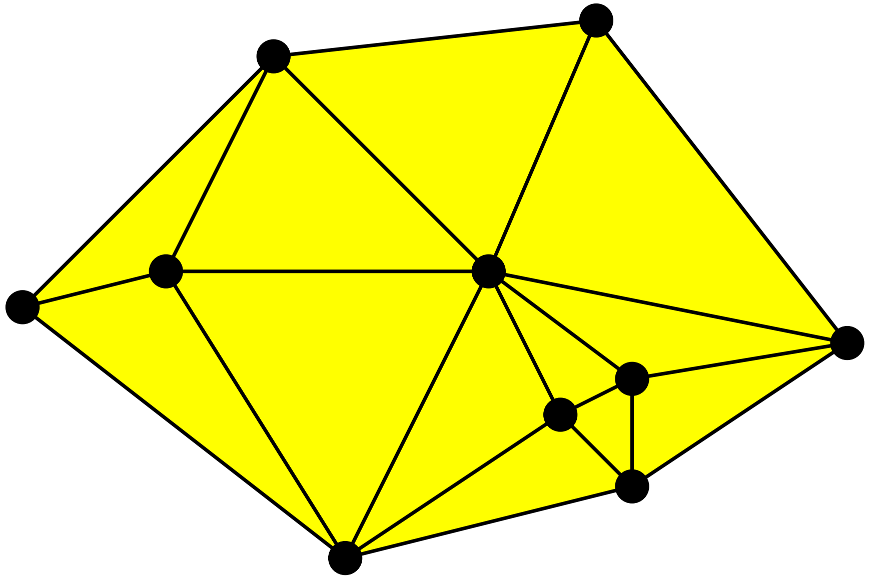
A **triangulation** of S:



Triangulation

S a set of points in the plane, no three points lying on a line

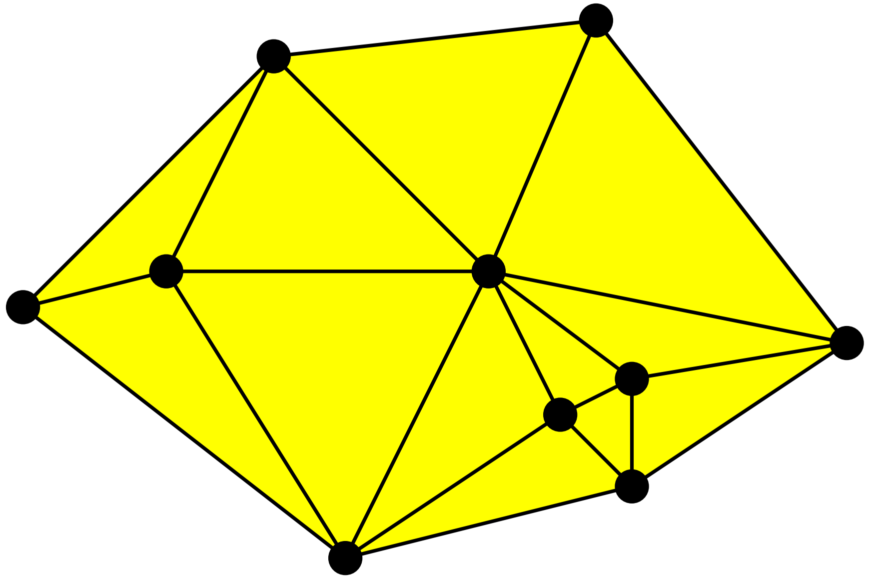
A **triangulation** of S:



Triangulation

S a set of points in the plane, no three points lying on a line

A **triangulation** of S:



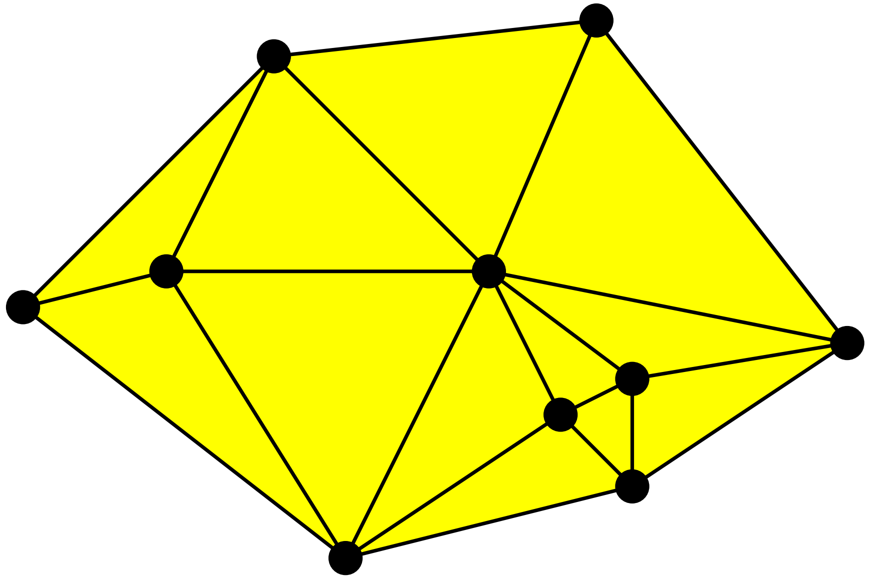
a subdivision of $\mathrm{conv}(S)$ into triangles s.t.
edges: straight line segments connecting points from S and
triangles: with no point from S in their interiors

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Minimum weight triangulation

S a set of points in the plane, no three points lying on a line

Weight of a triangulation of S:

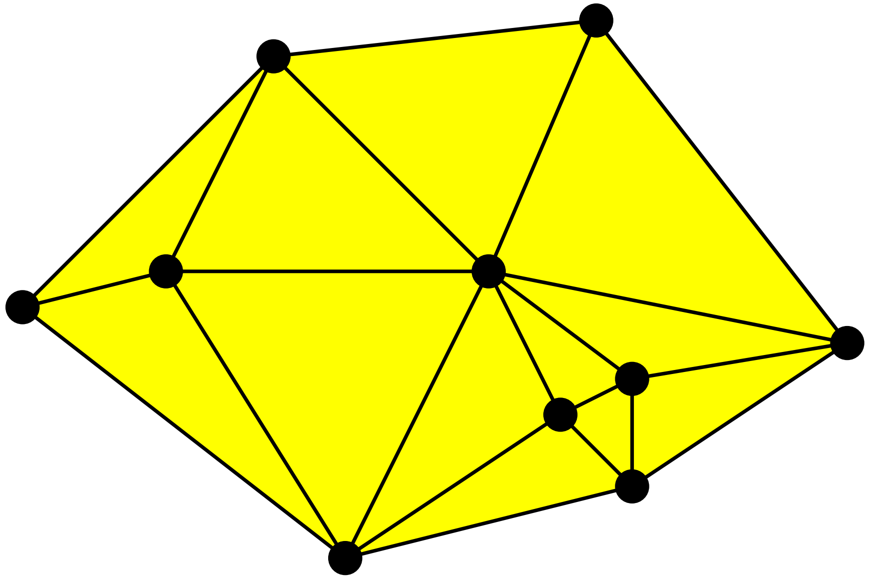


sum of the lengths of the edges

Minimum weight triangulation

S a set of points in the plane, no three points lying on a line

Weight of a triangulation of S:

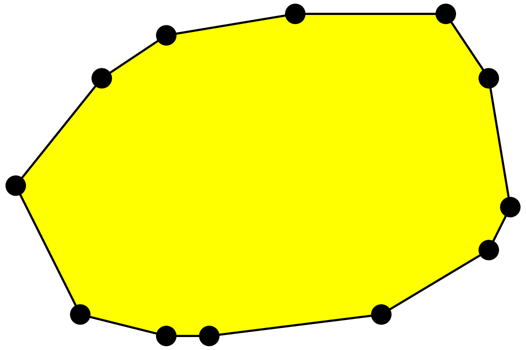


sum of the lengths of the edges

Minimum weight triangulation:
a triangulation with minimum weight

The minimum weight triangulation problem

- **In general**, not known to be
  - solvable in polynomial time, or
  - NP-hard.
- **If the points are in convex position,** can be solved in $O(n^3)$ time

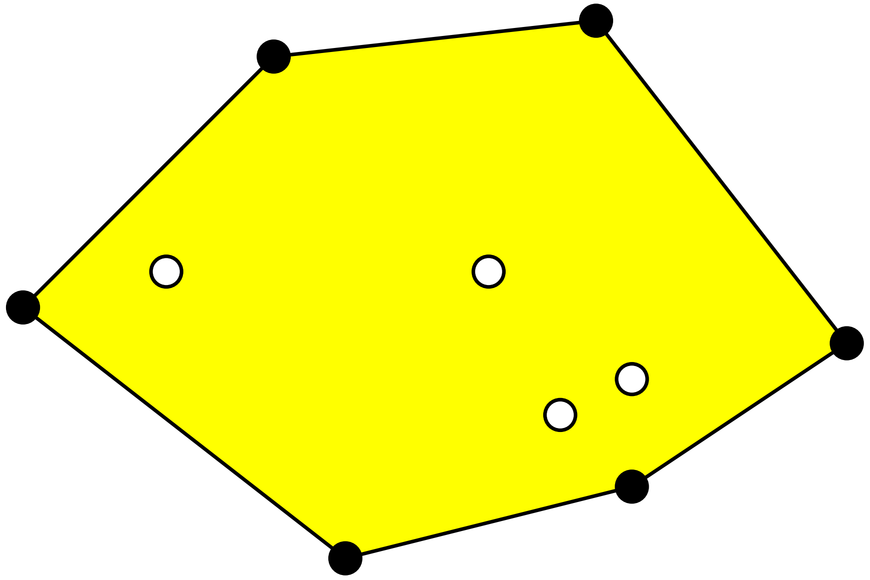


n the number of points

Distance-from-Triviality approach

Guo, Hüffner & Niedermeier (prev. talk)

“Distance from Triviality” = Number of inner points



The less inner points ⟷ The easier instance

Cf. Deĭneko, Hoffmann, Okamoto & Woeginger ’04:
TSP with few inner points

Result

Develop an algorithm to solve the minimum weight triangulation problem in $O(6^k n^5 \log n)$ time.

n	the total number of points
k	the number of inner points

Result

Develop an algorithm to solve the minimum weight triangulation problem in $O(6^k n^5 \log n)$ time.

n the total number of points
k the number of inner points

Consequences:

The minimum weight triangulation problem

- is **fixed parameter tractable** (FPT) w.r.t. the number of inner points.
- can be solved in **polynomial time if $k = O(\log n)$**.

Divide & Conquer and Dynamic Programming

Overview of the rest of my talk:

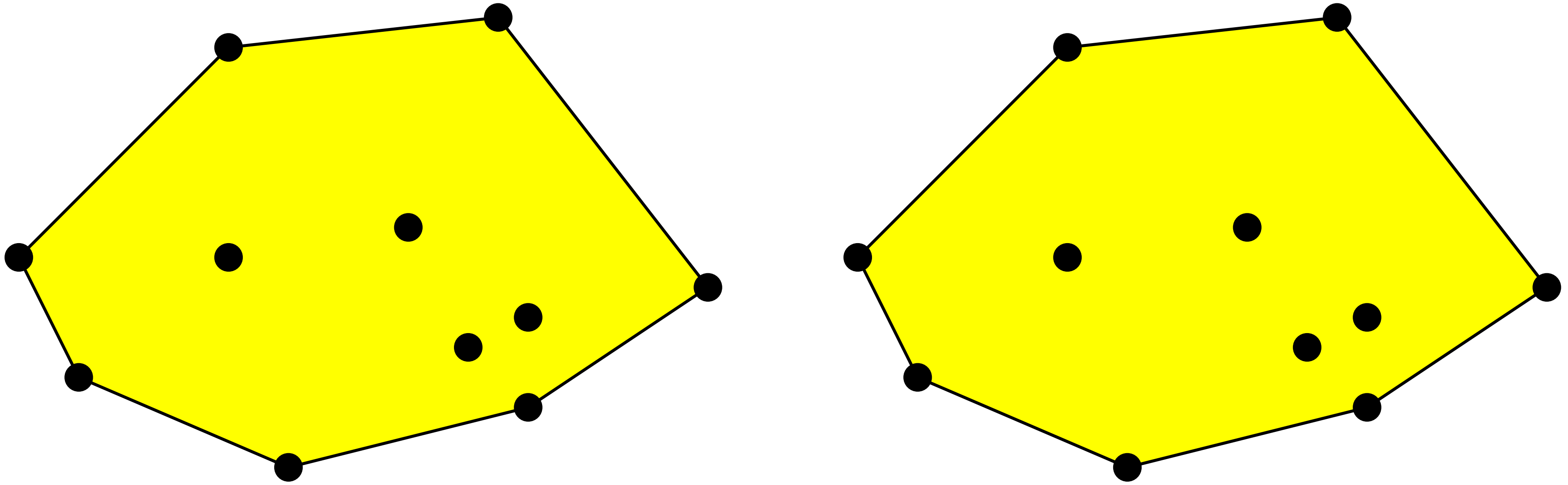
- Basic property
- Decompositions
- Dynamic Programming

Basic Property

S a point set, $\mathcal{T}$ a triangulation of S, p the leftmost pt of S

Obs 1 One of the following two happens.

- p forms a triangle together with its neighbors on the bd.
- $\exists$ an x-monotone inner path from p.

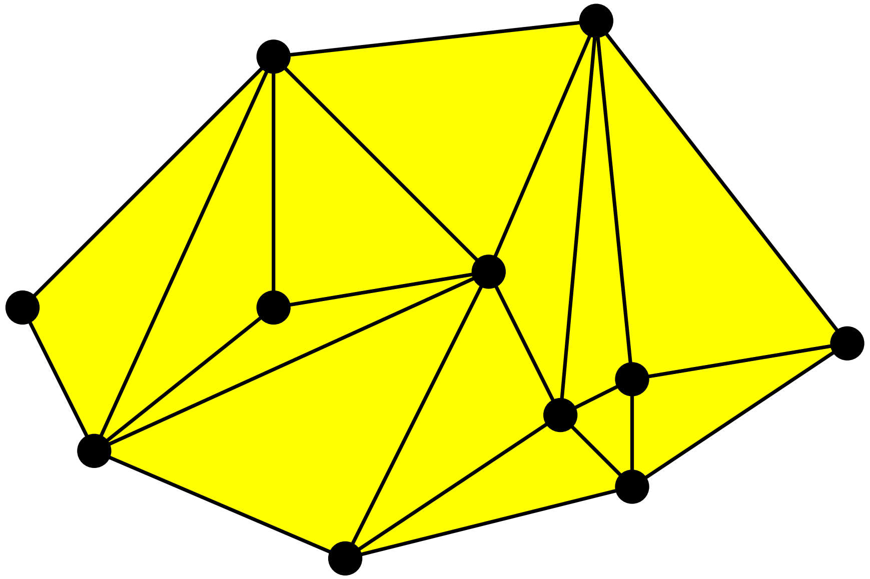


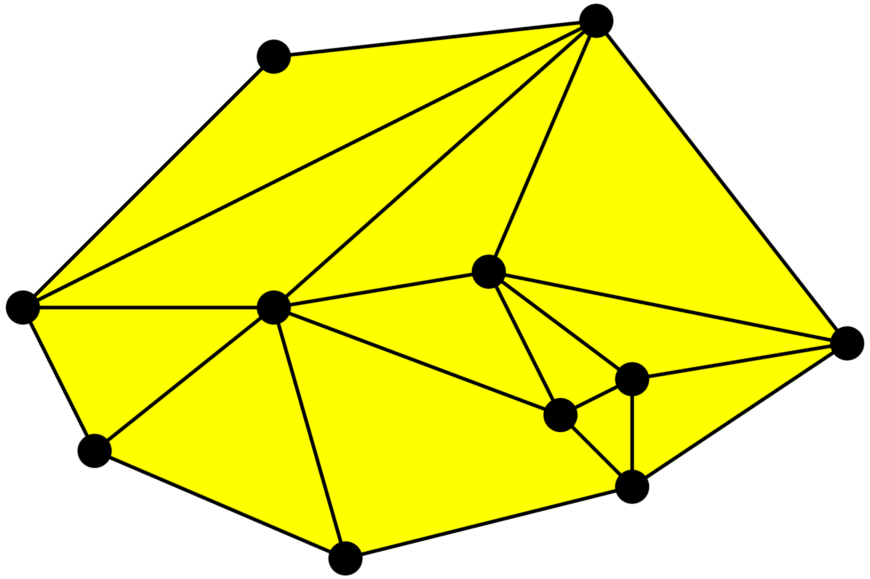
Basic Property

S a point set, $\mathcal{T}$ a triangulation of S, p the leftmost pt of S

Obs 1 One of the following two happens.

- p forms a triangle together with its neighbors on the bd.
- $\exists$ an x-monotone inner path from p.



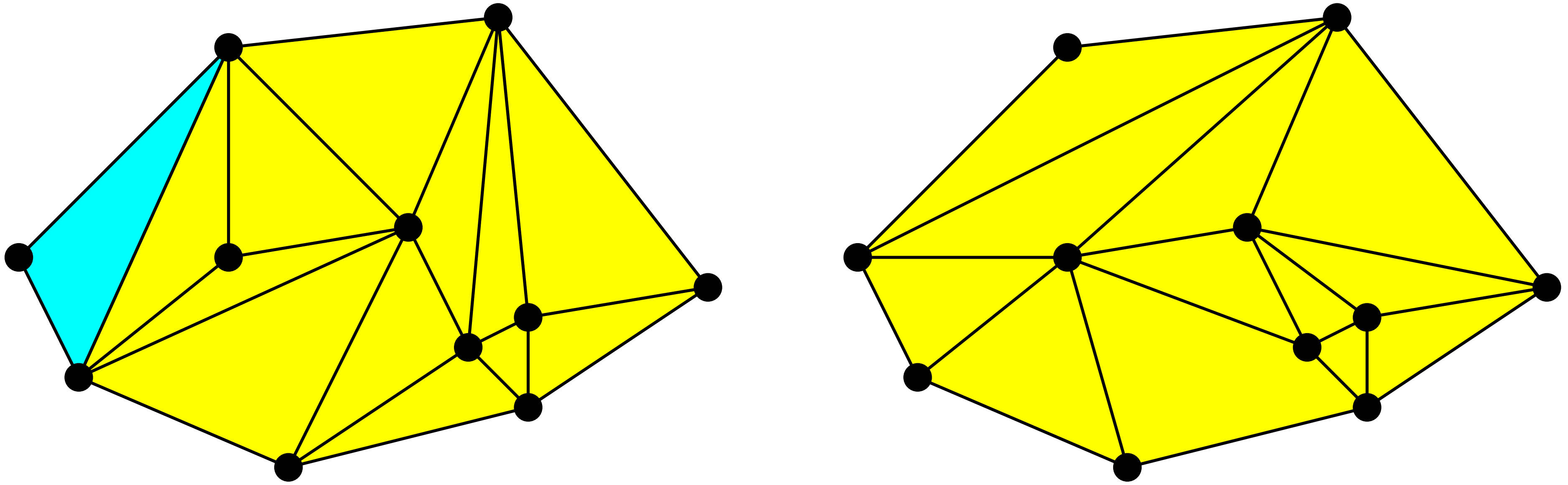


Basic Property

S a point set, $\mathcal{T}$ a triangulation of S, p the leftmost pt of S

Obs 1 One of the following two happens.

- p forms a triangle together with its neighbors on the bd.
- $\exists$ an x-monotone inner path from p.

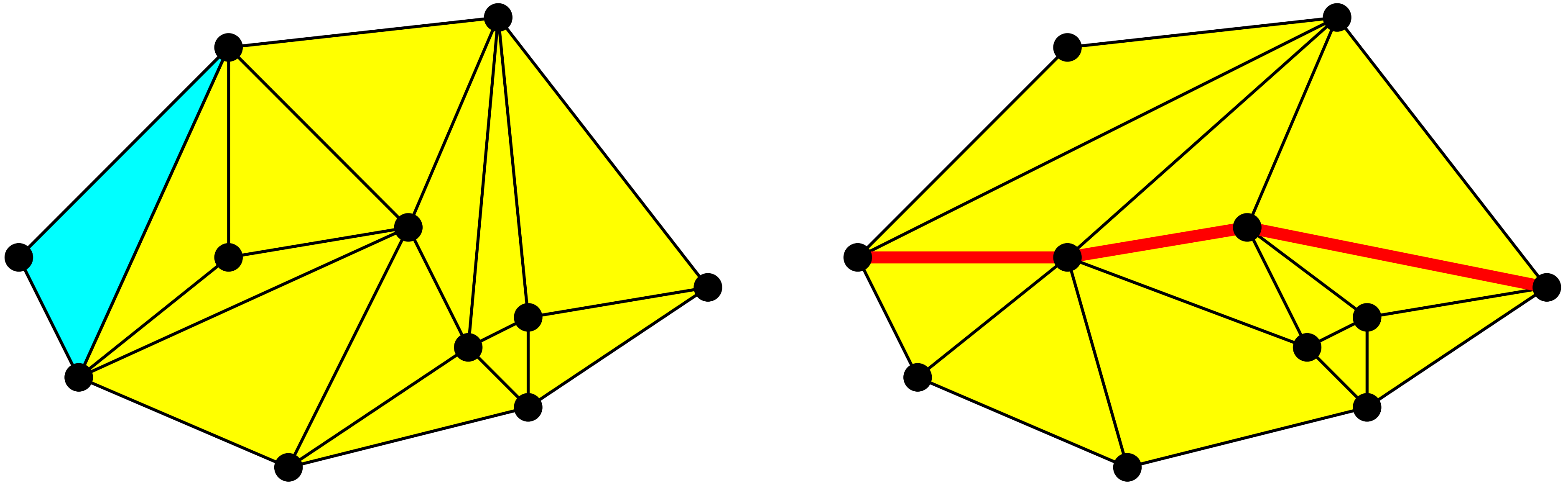


Basic Property

S a point set, $\mathcal{T}$ a triangulation of S, p the leftmost pt of S

Obs 1 One of the following two happens.

- p forms a triangle together with its neighbors on the bd.
- $\exists$ an x-monotone inner path from p.

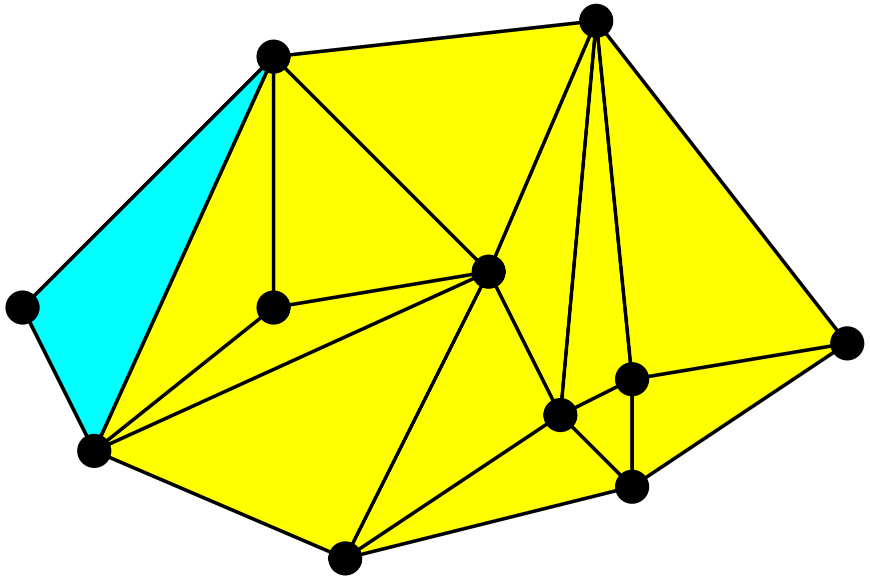


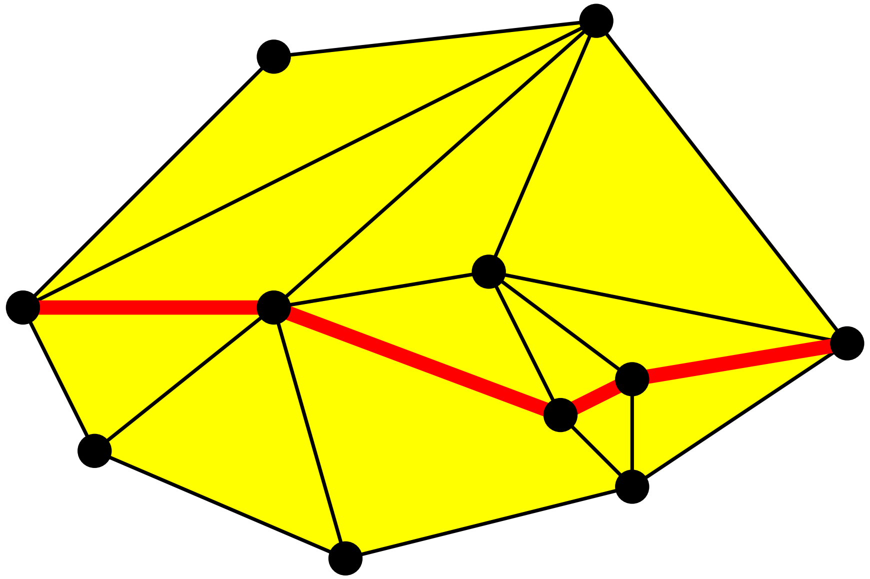
Basic Property

S a point set, $\mathcal{T}$ a triangulation of S, p the leftmost pt of S

Obs 1 One of the following two happens.

- p forms a triangle together with its neighbors on the bd.
- $\exists$ an x-monotone inner path from p.



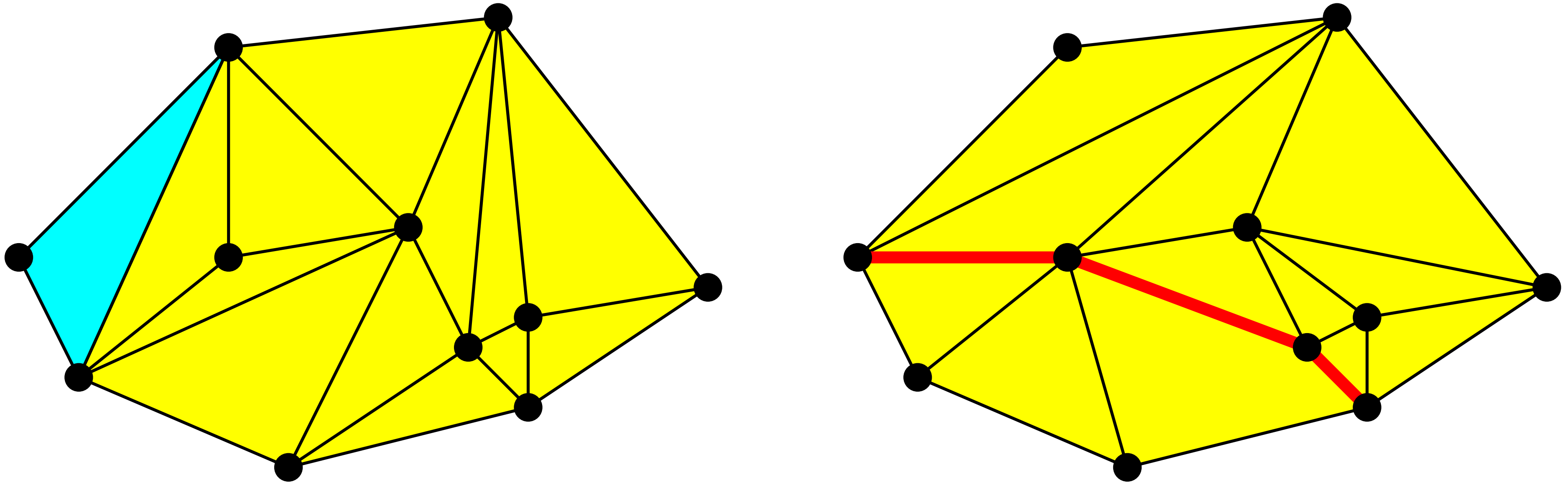


Basic Property

S a point set, $\mathcal{T}$ a triangulation of S, p the leftmost pt of S

Obs 1 One of the following two happens.

- p forms a triangle together with its neighbors on the bd.
- $\exists$ an x-monotone inner path from p.

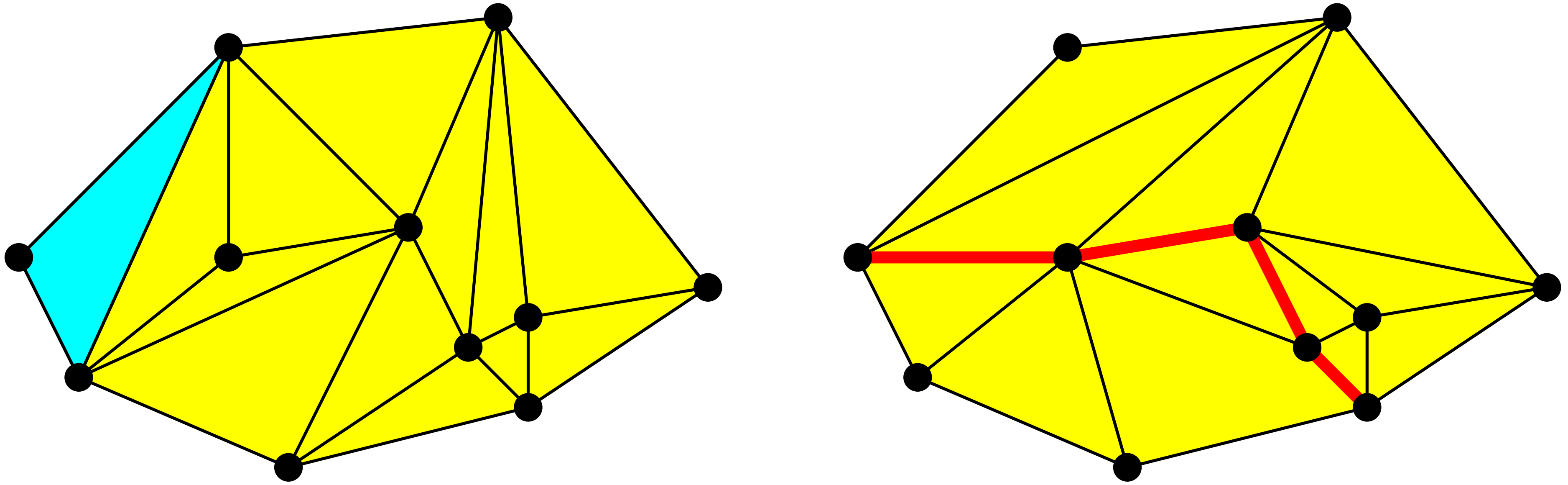


Basic Property

S a point set, $\mathcal{T}$ a triangulation of S, p the leftmost pt of S

Obs 1 One of the following two happens.

- p forms a triangle together with its neighbors on the bd.
- $\exists$ an x-monotone inner path from p.

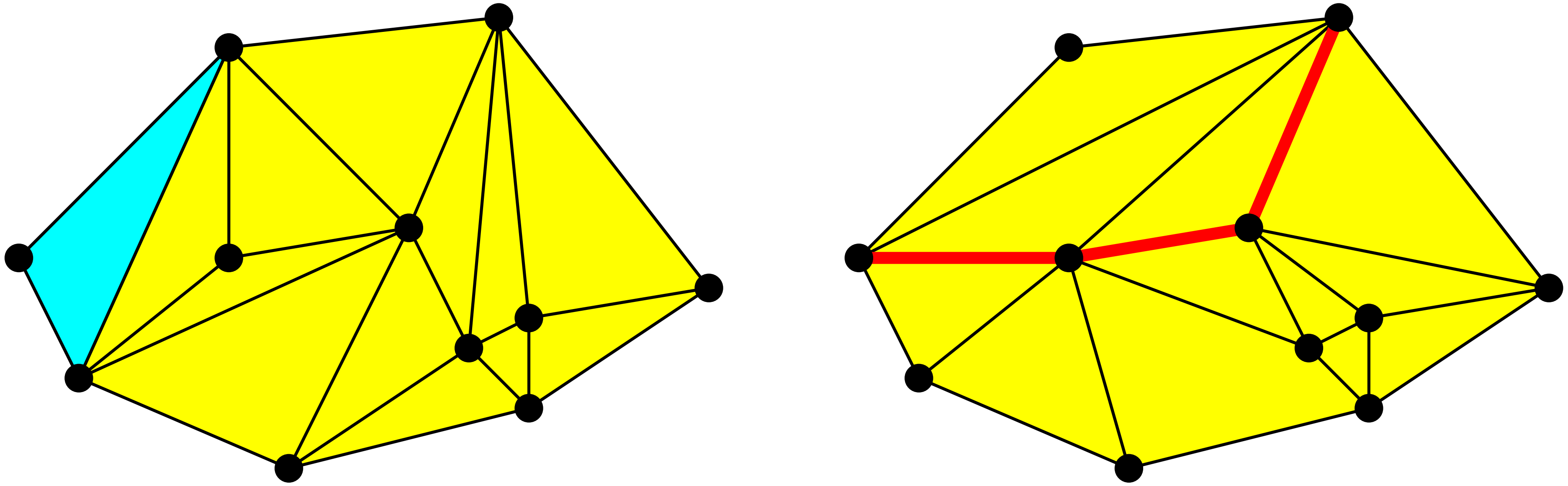


Basic Property

S a point set, $\mathcal{T}$ a triangulation of S, p the leftmost pt of S

Obs 1 One of the following two happens.

- p forms a triangle together with its neighbors on the bd.
- $\exists$ an x-monotone inner path from p.

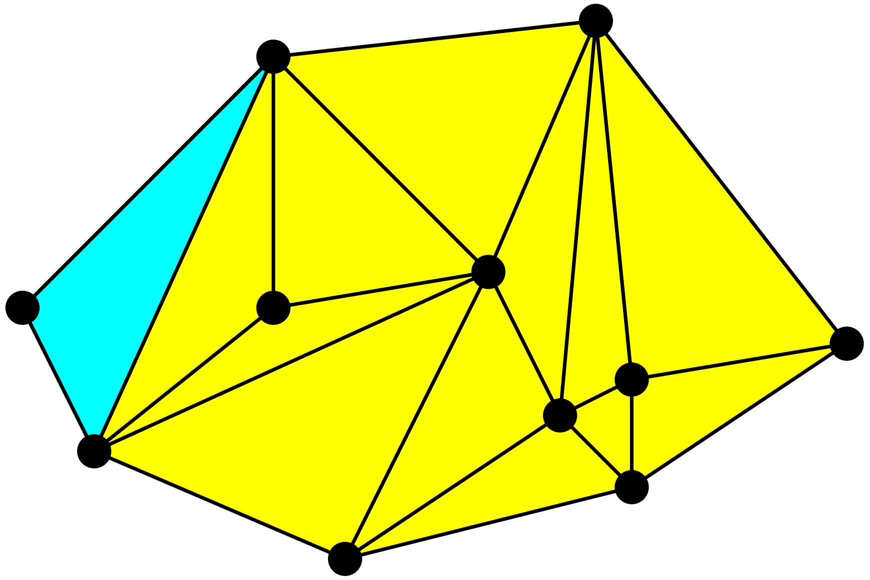


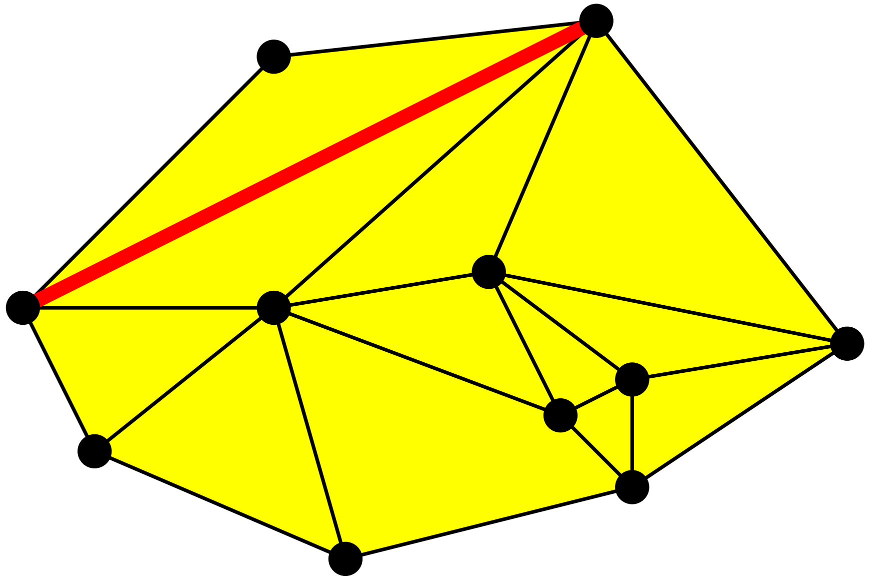
Basic Property

S a point set, $\mathcal{T}$ a triangulation of S, p the leftmost pt of S

Obs 1 One of the following two happens.

- p forms a triangle together with its neighbors on the bd.
- $\exists$ an x-monotone inner path from p.





Strategy

S a point set, $\mathcal{T}$ a triangulation of S, p the leftmost pt of S

Obs 1 One of the following two happens.

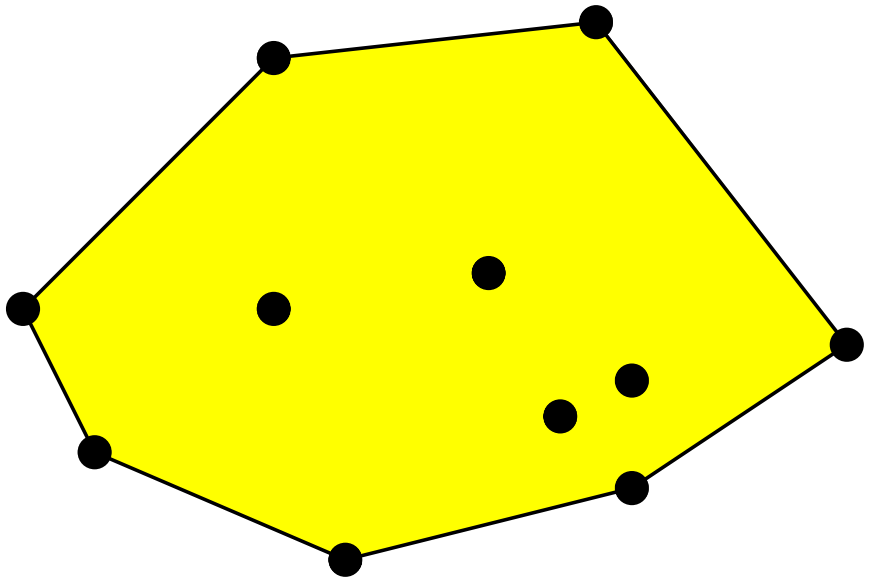
- ◆ p forms a triangle together with its neighbors on the bd.
- ◆ $\exists$ an x-monotone inner path from p.

In particular, this holds for min weight triangulations.

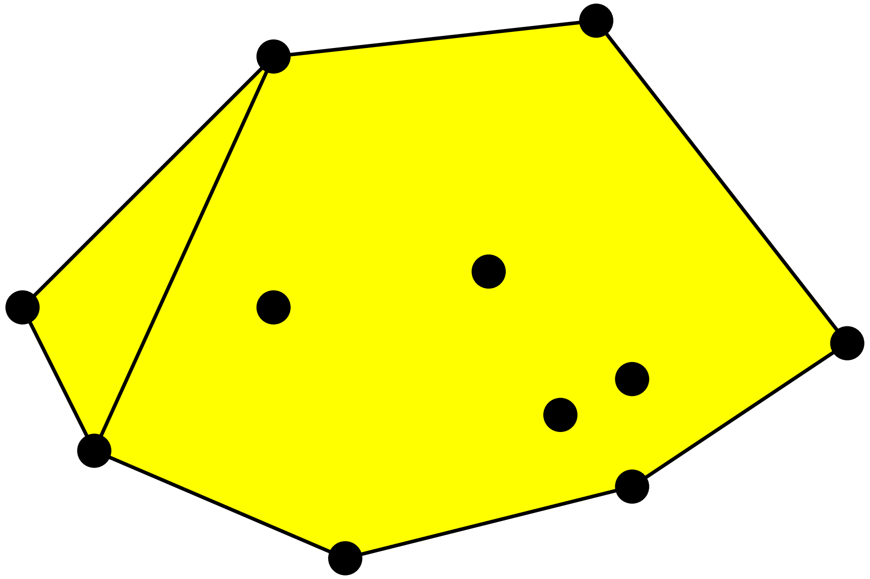
$$\Downarrow$$

Try all possible cases!!

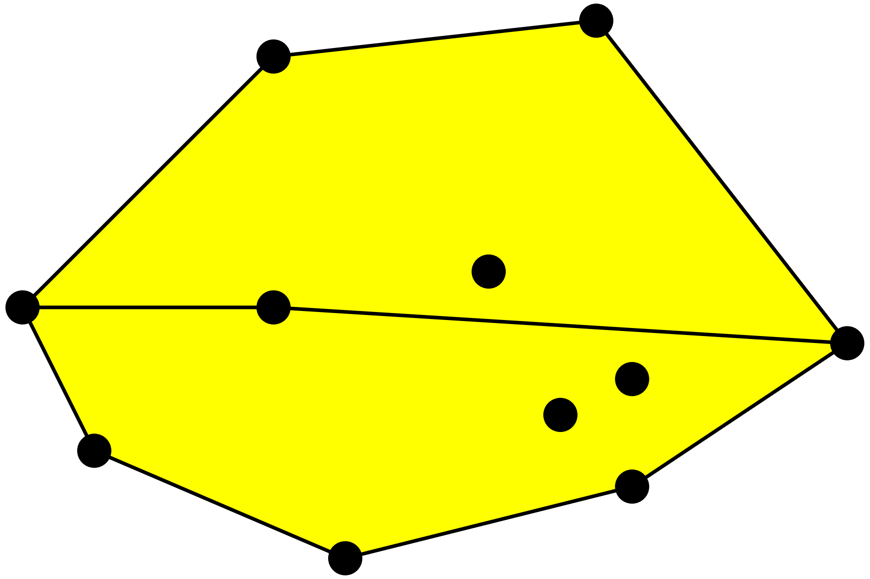
Given a point set...



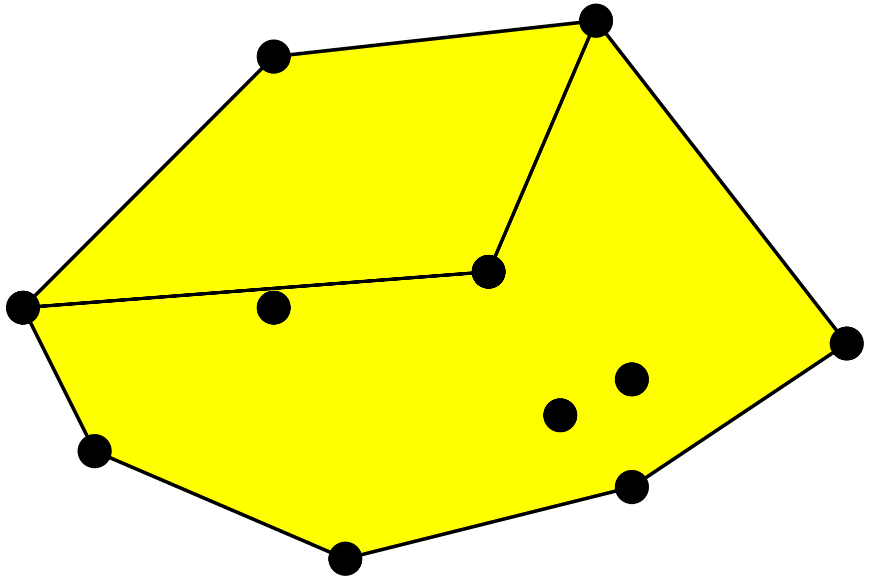
In a triangulation, this happens?



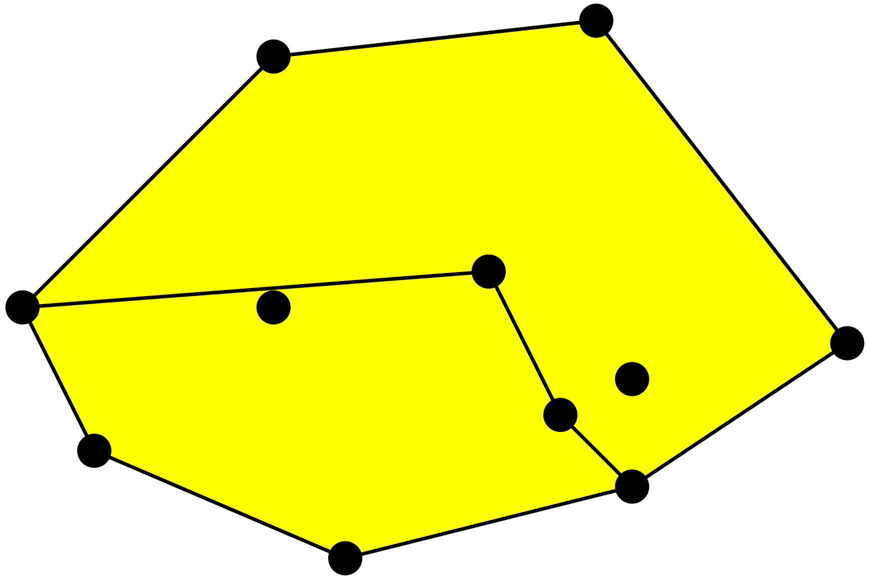
In a triangulation, this happens?



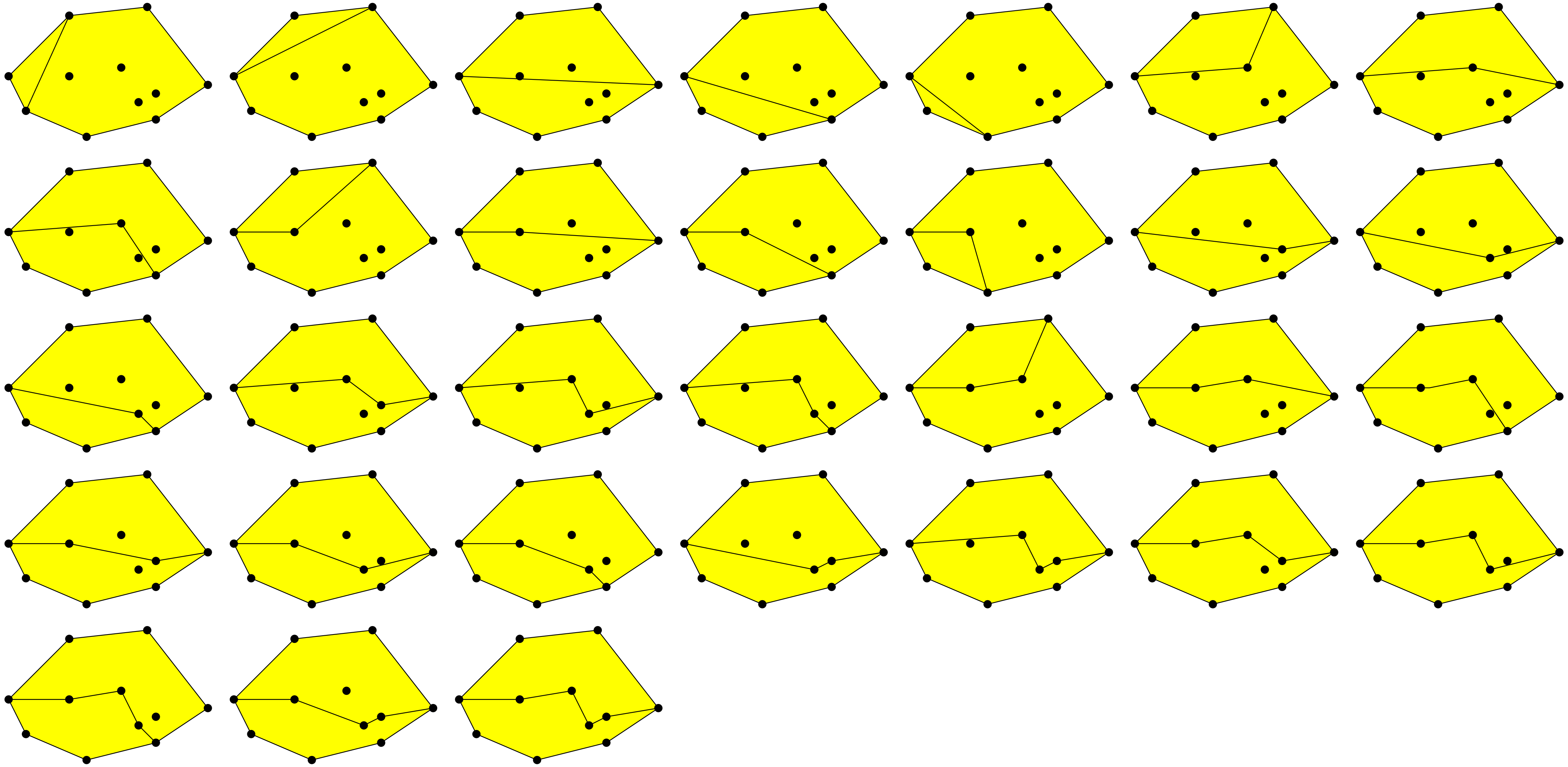
In a triangulation, this happens?



In a triangulation, this happens?



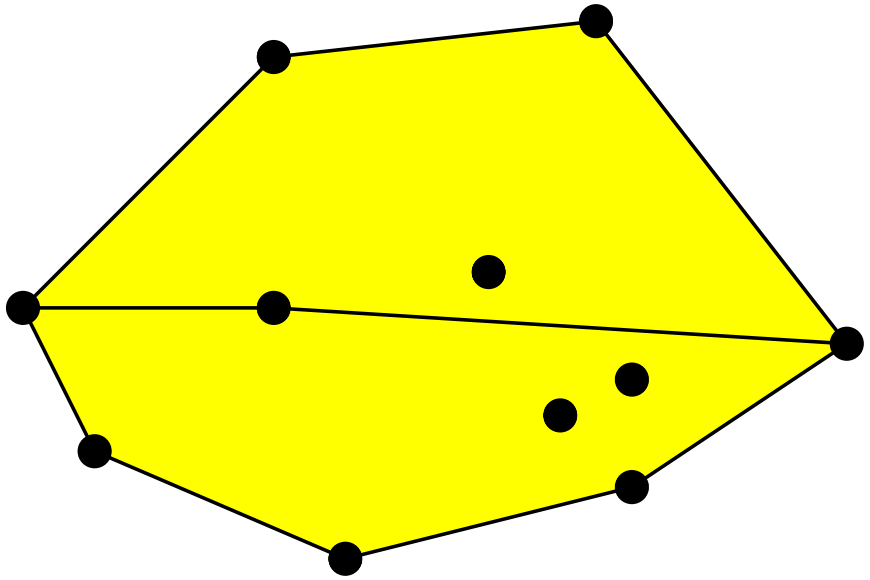
Try all possible cases!!



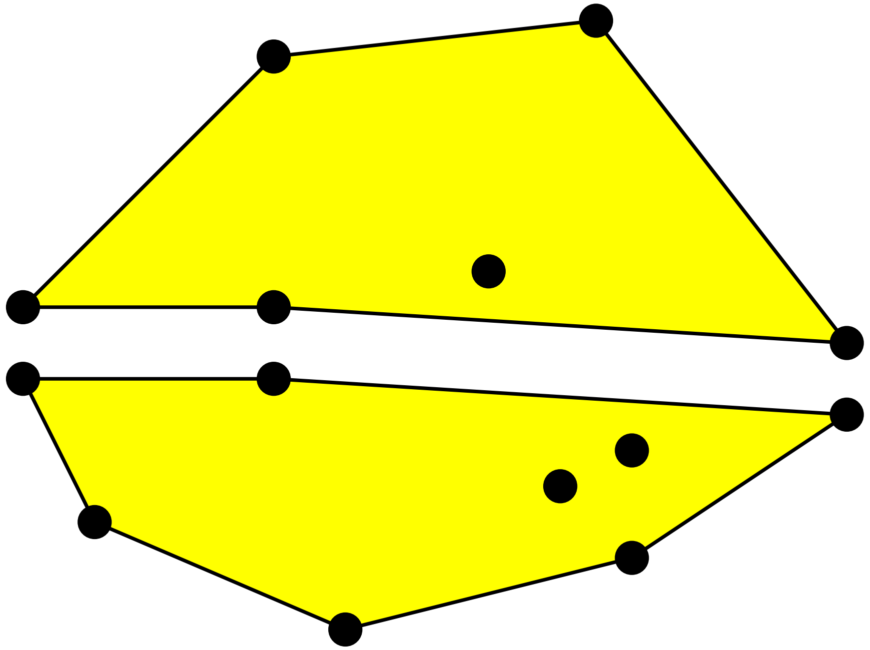
A min weight triangulation fits at least one of these cases.

 # possible cases $\leq 1 + 2^k n$.

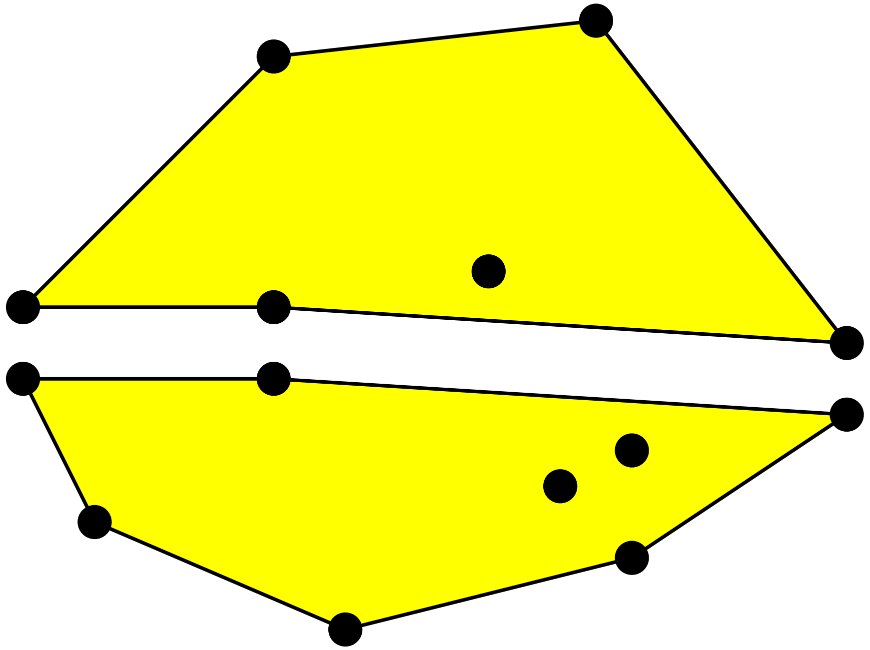
Look at one of the cases...



Divide into two pieces!!

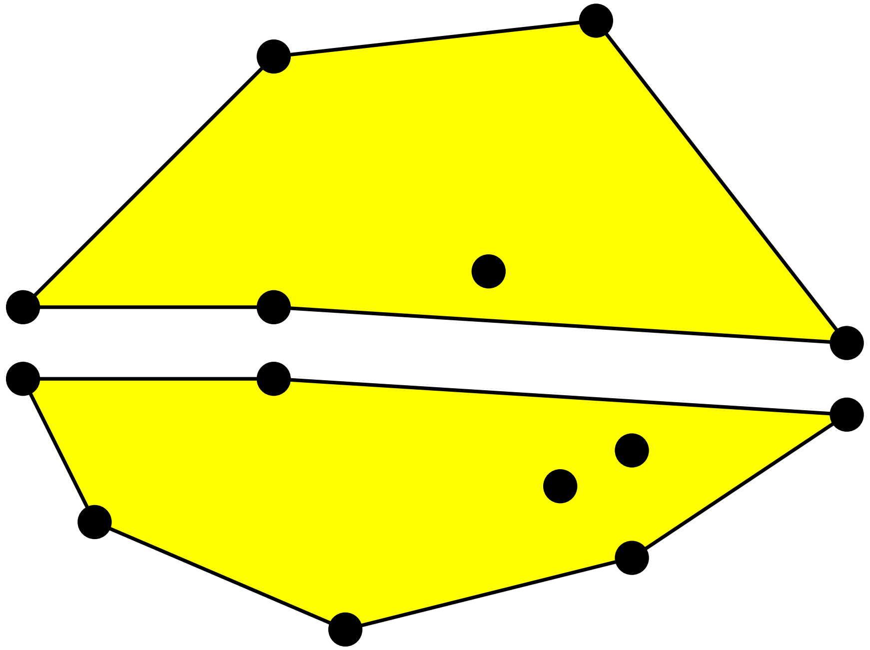


Divide into two pieces!!



⇒ **Recursively solve on the smaller polygons!!**

Divide into two pieces!!



⇒ **Recursively solve on the smaller polygons!!**

Two Problems:

- Smaller polygons can be non-convex.
- Top-down recursion will not give FPT.

Two problems to resolve

Two Problems:

- Smaller polygons can be non-convex.

- Top-down recursion will not give FPT.

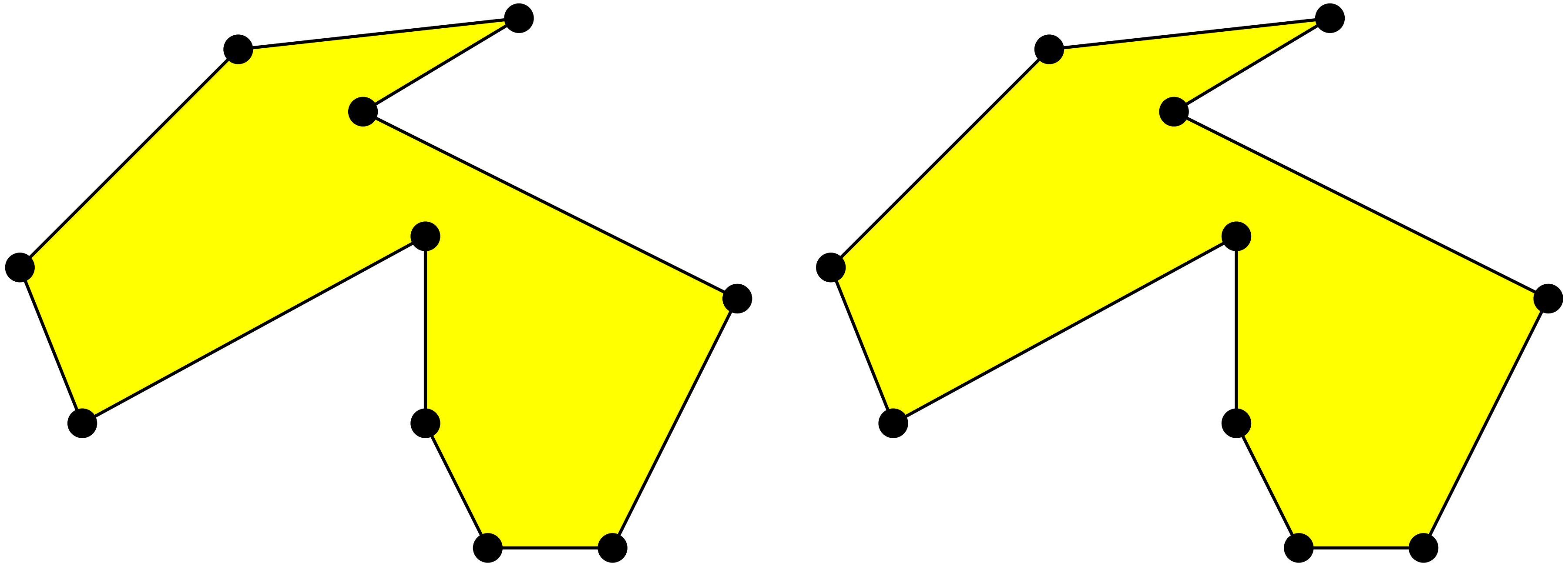
Two problems to resolve

Two Problems:

- Smaller polygons can be non-convex.
  Generalize to simple polygons!!
- Top-down recursion will not give FPT.

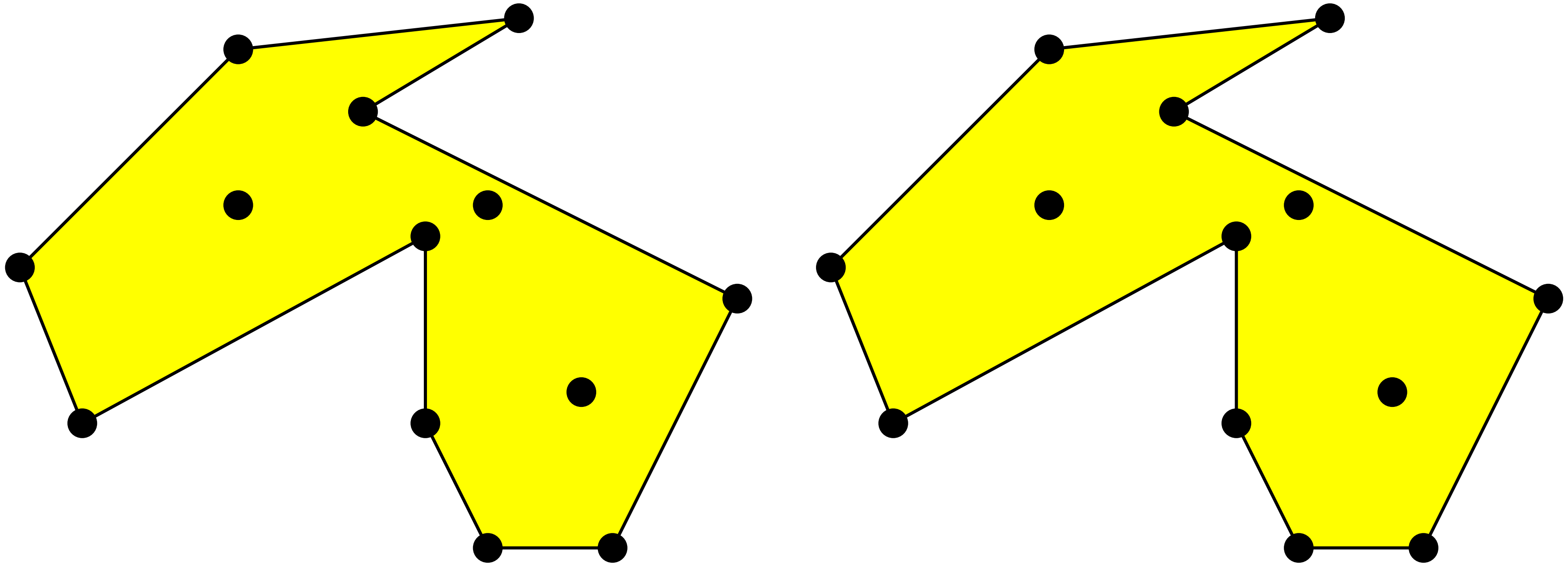
Obs 1 also holds for simple polygons

P a simple polygon



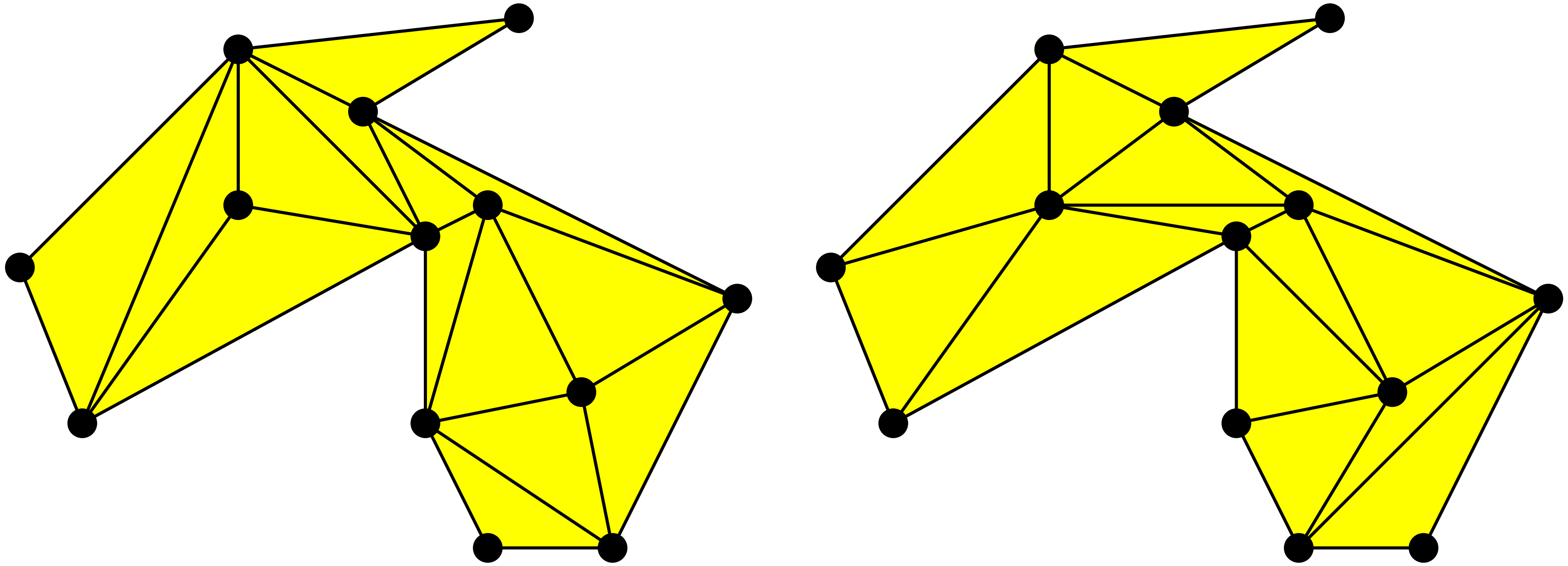
Obs 1 also holds for simple polygons

P a simple polygon, S a point set inside P,



Obs 1 also holds for simple polygons

P a simple polygon, S a point set inside P,
$\mathcal{T}$ a triangulation of (P, S), p the leftmost point of P

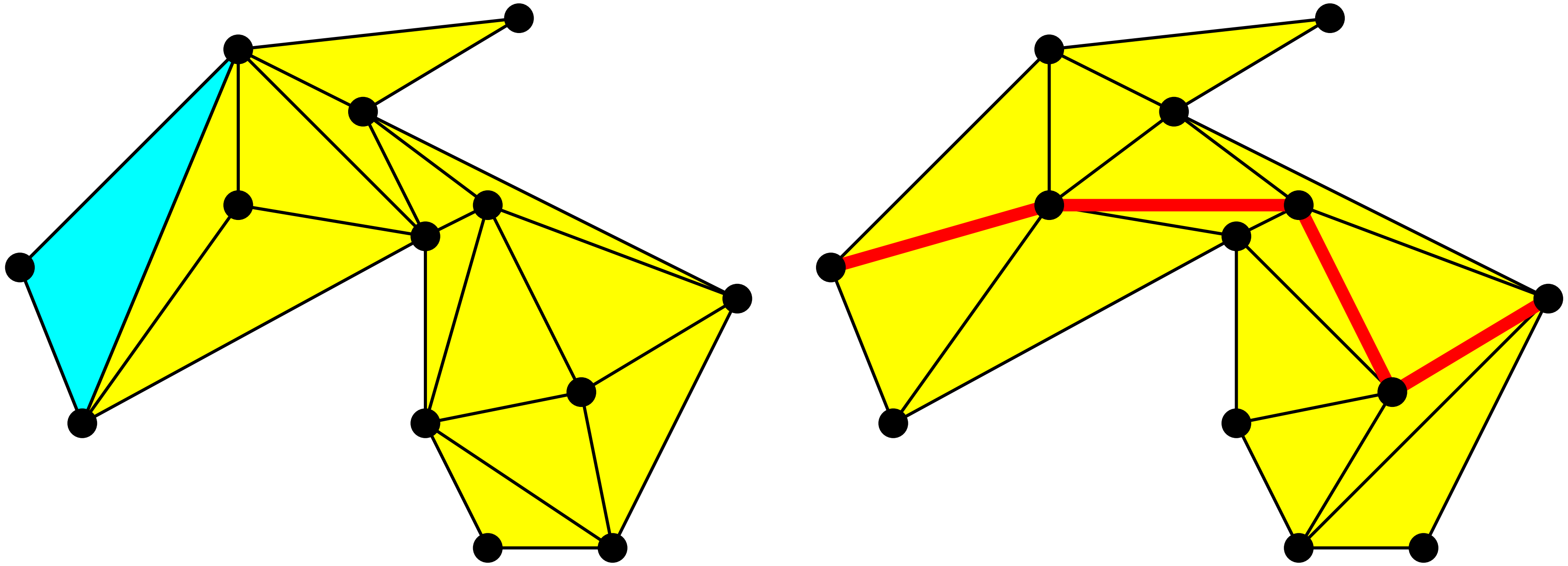


Obs 1 also holds for simple polygons

P a simple polygon, S a point set inside P,
$\mathcal{T}$ a triangulation of (P, S), p the leftmost point of P

Obs 1' One of the following two happens.

- p forms a triangle together with its neighbors on the bd.
- $\exists$ an x-monotone inner path from p.



Two problems to resolve

Two Problems:

- Smaller polygons can be non-convex.
  Generalize to simple polygons!!
- Top-down recursion will not give FPT.

Two problems to resolve

Two Problems:

- Smaller polygons can be non-convex.
  Generalize to simple polygons!!
- Top-down recursion will not give FPT.
  Solve in a bottom-up manner by DP!!

Outline of the algorithm

(1) Enumerate all small polygons appearing in the subdivisions.

(2) Determine which polygons arise from which polygons

(3) Solve the recursion in a bottom-up manner by DP.

Two problems to resolve

Two Problems:

- Smaller polygons can be non-convex.
  Generalize to simple polygons!!
- Top-down recursion will not give FPT.
  Solve in a bottom-up manner by DP!!

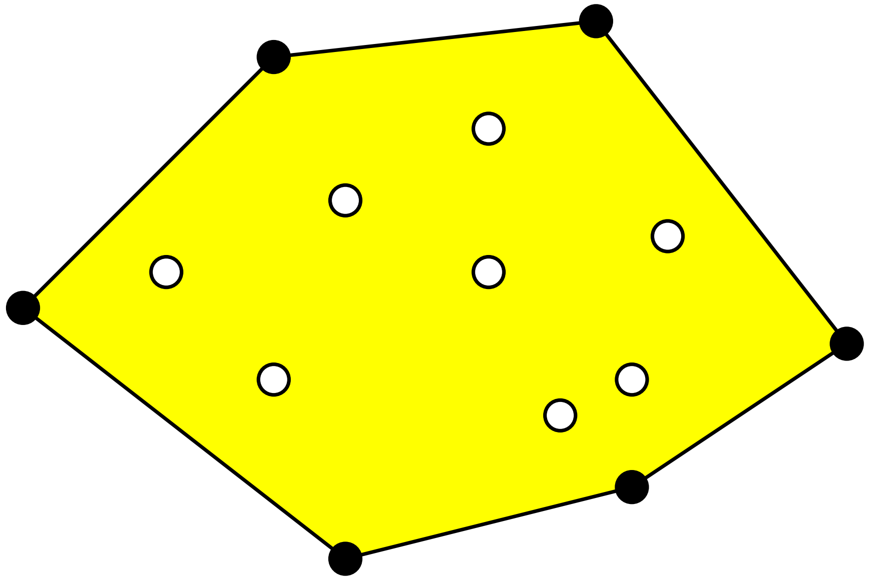
Two problems to resolve

Two Problems:

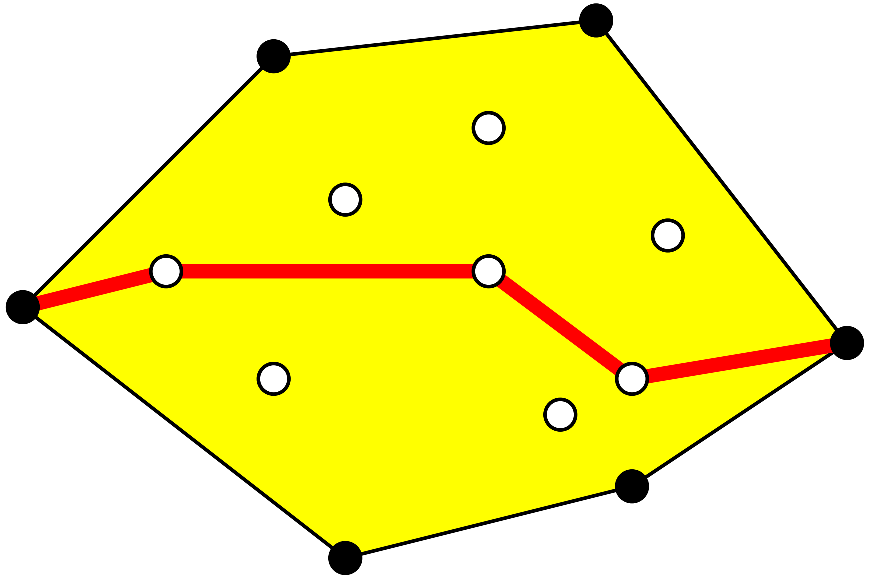
- Smaller polygons can be non-convex.
  Generalize to simple polygons!!
- Top-down recursion will not give FPT.
  Solve in a bottom-up manner by DP!!
  Need to control the types of polygons appearing in subdivisions.

Introduce three types of polygons ...

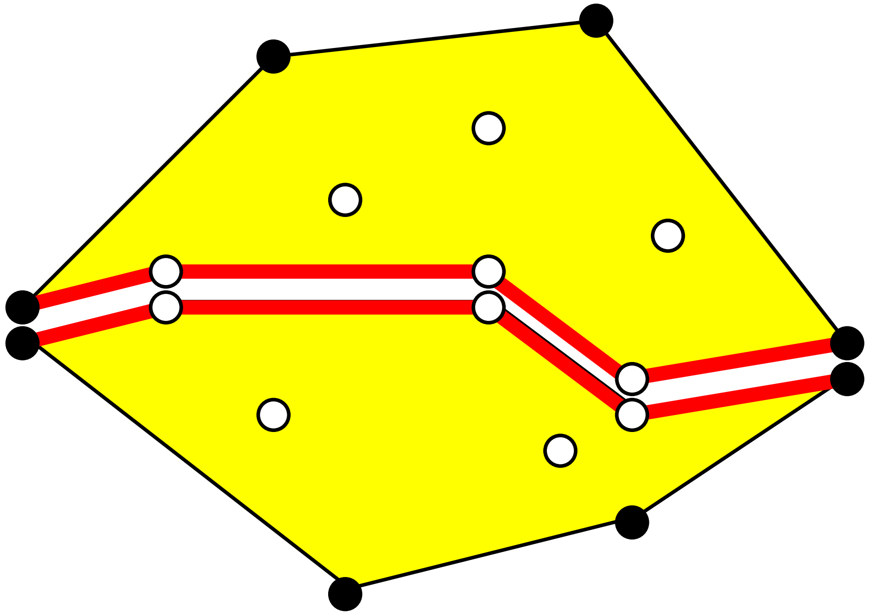
Again, given a point set...



Take an x-monotone inner path...



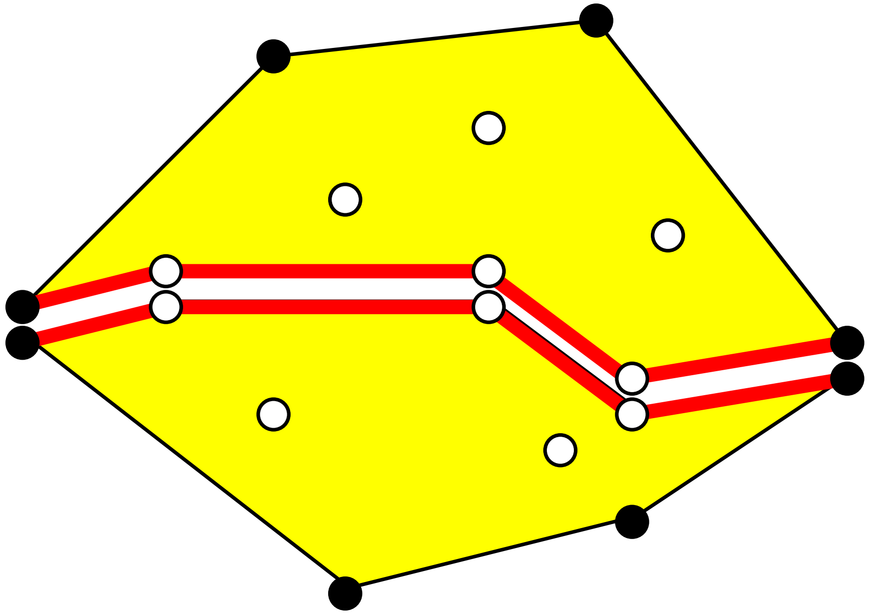
Then, subdivide



Type-1 polygons

A **type-1 polygon**:

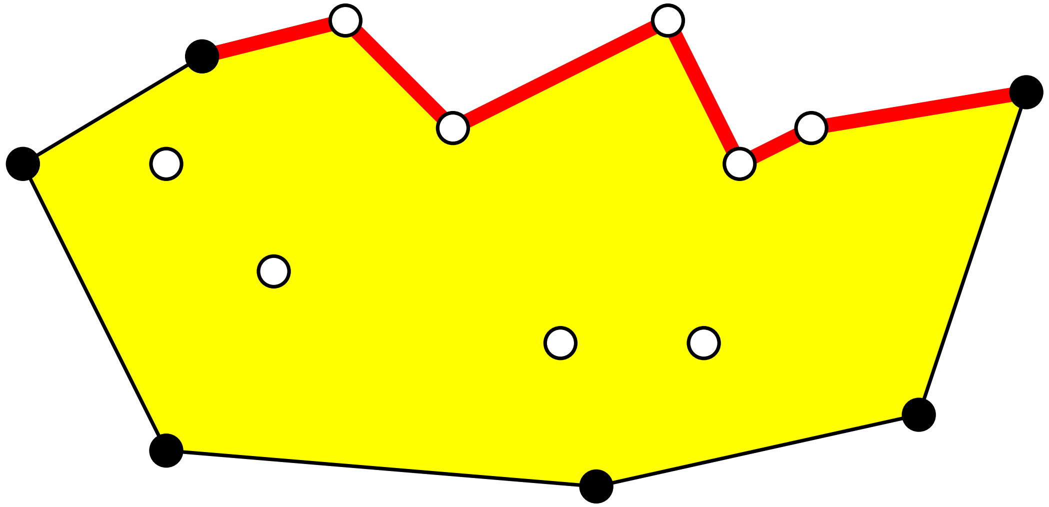
the boundary consists of
one x-monotone inner path of the orig polygon and
one boundary chain of the orig polygon.



Type-1 polygons

A **type-1 polygon**:

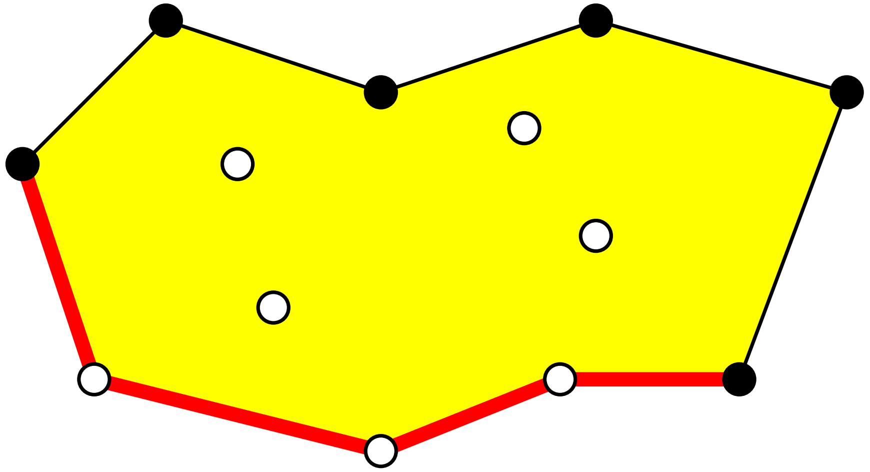
the boundary consists of
one x-monotone inner path of the orig polygon and
one boundary chain of the orig polygon.



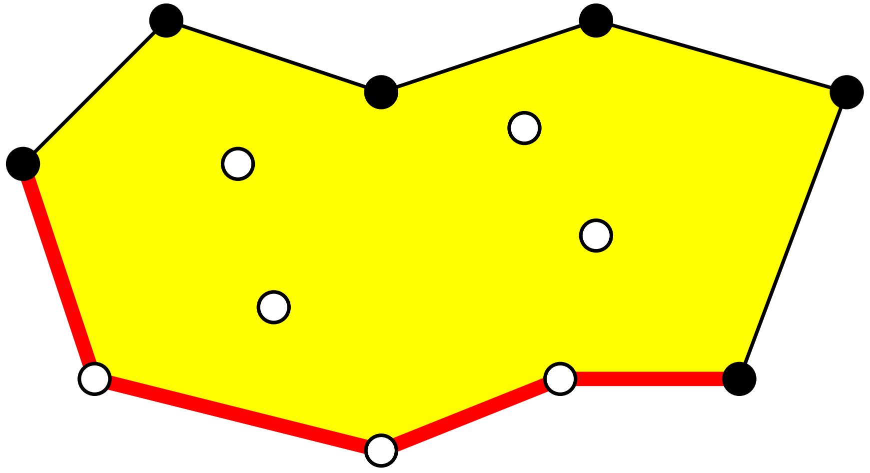
Type-1 polygons

A **type-1 polygon**:

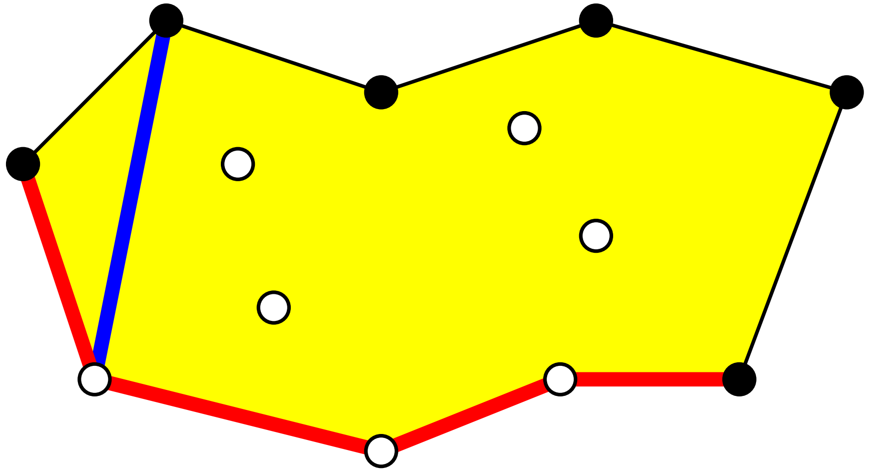
the boundary consists of
one x-monotone inner path of the orig polygon and
one boundary chain of the orig polygon.



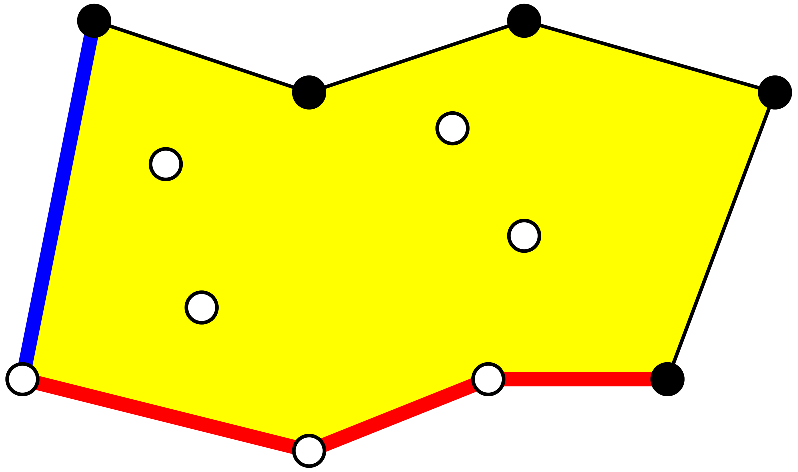
Take this polygon



Subdivide this way



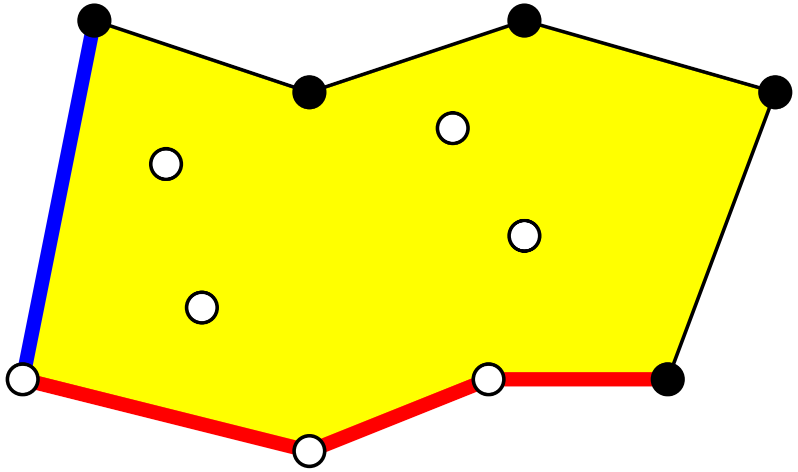
Look at this part



Type-2 polygons

A **type-2 polygon**:

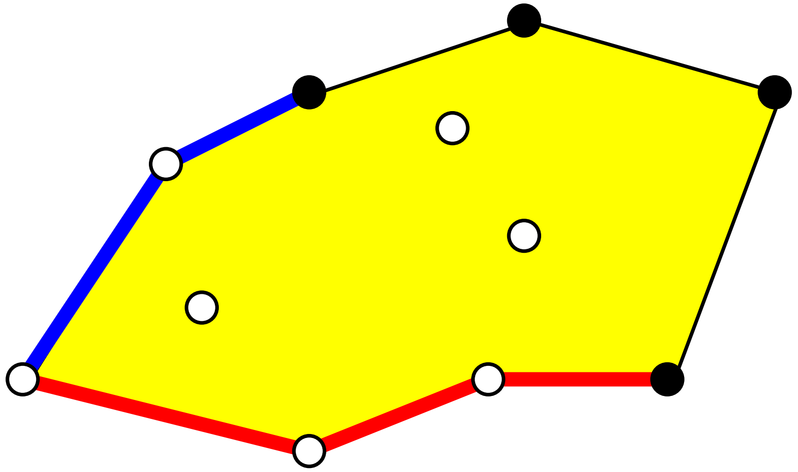
the boundary consists of
two x-monotone inner paths of the orig polygon and
one boundary chain of the orig polygon.



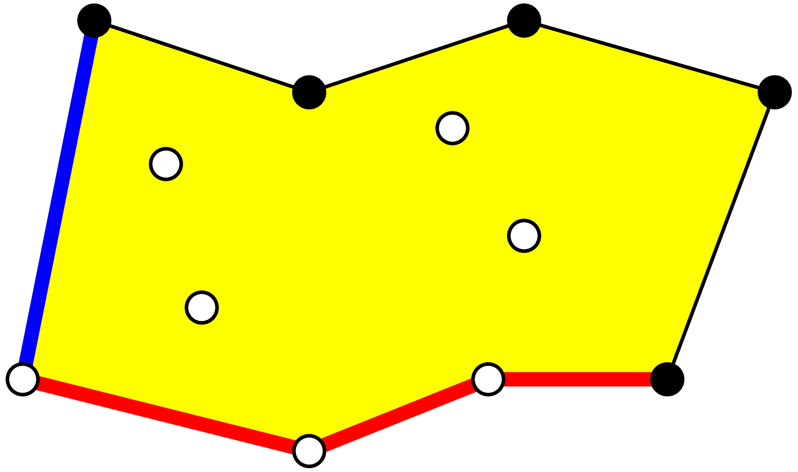
Type-2 polygons

A **type-2 polygon**:

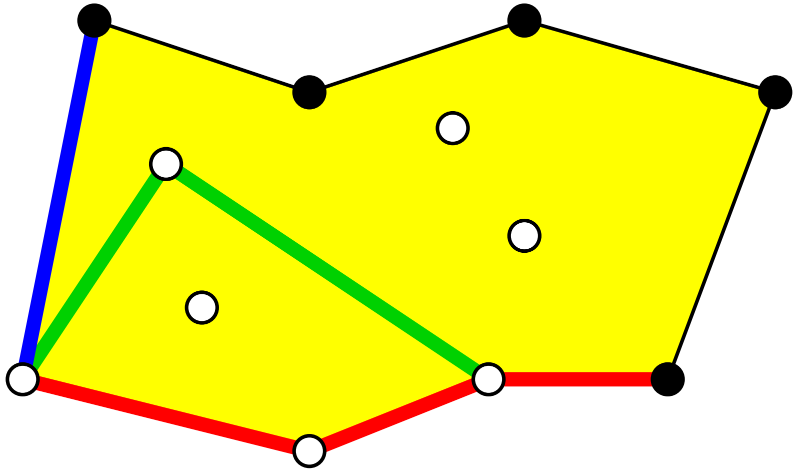
the boundary consists of
two x-monotone inner paths of the orig polygon and
one boundary chain of the orig polygon.



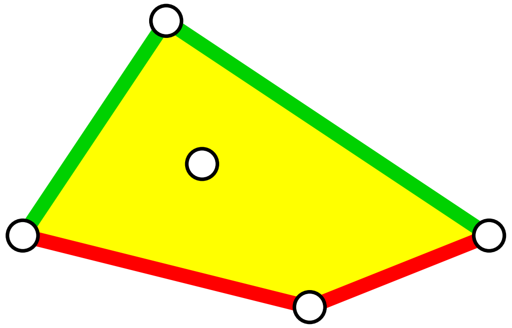
Look at this polygon



Take this inner path



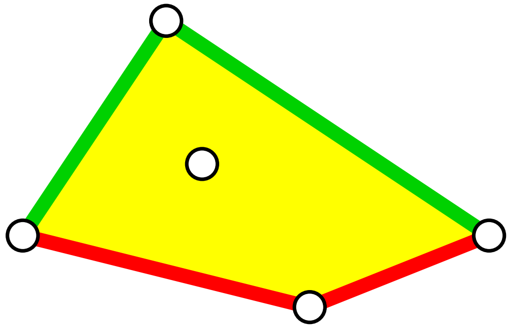
Then subdivide



Type-3 polygons

A **type-3 polygon**:

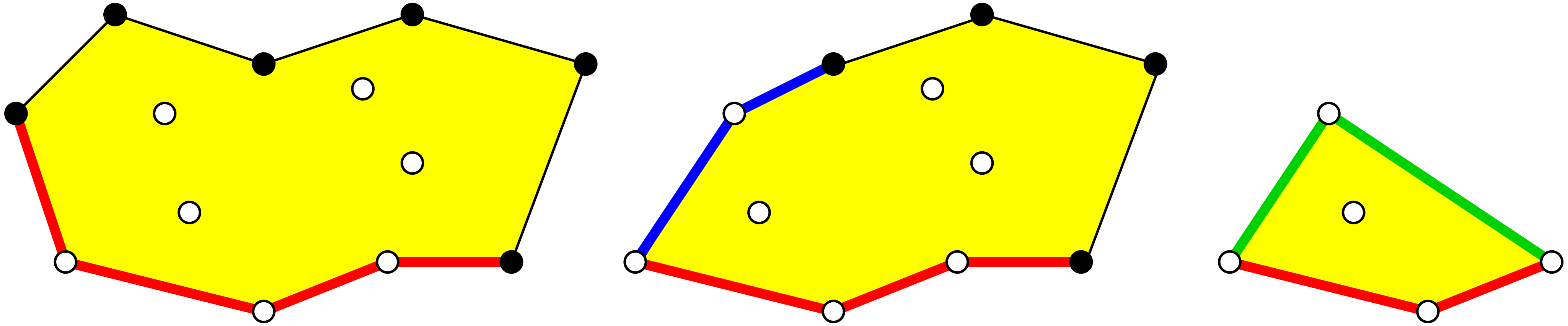
the boundary consists of
two x-monotone inner paths of the orig polygon.



Important lemma

By **subdividing polygons carefully**, we can show that

> **Small polygons appearing in the subdivisions are limited to type-1, 2, 3 polygons.**



Outline of the algorithm

(1) Enumerate all small polygons appearing in the subdivisions.

(2) Determine which polygons arise from which polygons

(3) Solve the recursion in a bottom-up manner by DP.

Outline of the algorithm

(1) **Enumerate all small polygons appearing in the subdivisions.**

- # of small polygons we enumerate $= O(3^k n^3)$.
- Can be done in $O(3^k n^4 \log n)$ time.

(2) **Determine which polygons arise from which polygons**

(3) **Solve the recursion in a bottom-up manner by DP.**

Outline of the algorithm

(1) **Enumerate all small polygons appearing in the subdivisions.**

- # of small polygons we enumerate $= O(3^k n^3)$.
- Can be done in $O(3^k n^4 \log n)$ time.

(2) **Determine which polygons arise from which polygons**

- Can be done in $O(6^k n^5 \log n)$ time.

(3) **Solve the recursion in a bottom-up manner by DP.**

Outline of the algorithm

(1) **Enumerate all small polygons appearing in the subdivisions.**

- # of small polygons we enumerate $= O(3^k n^3)$.
- Can be done in $O(3^k n^4 \log n)$ time.

(2) **Determine which polygons arise from which polygons**

- Can be done in $O(6^k n^5 \log n)$ time.

(3) **Solve the recursion in a bottom-up manner by DP.**

- Can be done in $O(6^k n^4)$ time.

Outline of the algorithm

(1) Enumerate all small polygons appearing in the subdivisions.

- \# of small polygons we enumerate $= O(3^k n^3)$.
- Can be done in $O(3^k n^4 \log n)$ time.

(2) Determine which polygons arise from which polygons

- Can be done in $O(6^k n^5 \log n)$ time.

(3) Solve the recursion in a bottom-up manner by DP.

- Can be done in $O(6^k n^4)$ time.

In total: **runs in $O(6^k n^5 \log n)$ time.**

Result (again)

The minimum weight triangulation problem can be solved in $O(6^k n^5 \log n)$ time.

n	the total number of points
k	the number of inner points

Consequences:

The minimum weight triangulation problem

- is **fixed parameter tractable** (FPT) w.r.t. the number of inner points.
- can be solved in **polynomial time if $k = O(\log n)$**.

Variation

The maximum weight triangulation problem can be solved in $O(6^k n^5 \log n)$ time.

n	the total number of points
k	the number of inner points

Consequences:

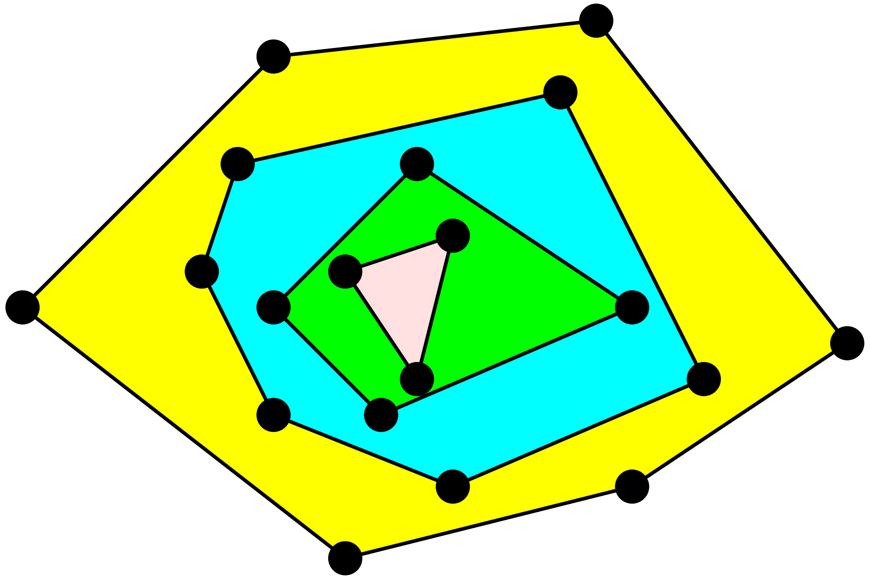
The maximum weight triangulation problem

- is **fixed parameter tractable** (FPT) w.r.t. the number of inner points.
- can be solved in **polynomial time** **if $k = O(\log n)$**.

Related past work

- **Not known to be NP-hard**
- **Approximation algorithms**
  - $O(1)$-approximation (Levcopoulos & Krznaric '98)
- **Exact algorithms** without run-time analysis
  - Integer prog. (Kyoda, Imai, Takeuchi & Tajima '97)
  - “Paths of a triangulation” (Aichholzer '99)
- **Parameterization**
  - Nested convex hulls (Anagnostou & Corneil '93)

Nested convex hulls



k the number of “nests”

A min weight triangulation can be found in $O(n^{3k+1})$ time. (Anagnostou & Corneil ’93)

Thank you

Tusen takk.