

Preprocessing Imprecise Points and Splitting Triangulations

Maarten Löffler
Utrecht
University
the
Netherlands

Joseph S. B.
Mitchell
SUNY at
Stony Brook
United States

Marc van
Kreveld
Utrecht
University
the
Netherlands

Let's start by
defining
imprecise
points.

Computational
geometry deals
with problems
on *precisely*
specified *points*
in \mathbb{R}^2 .



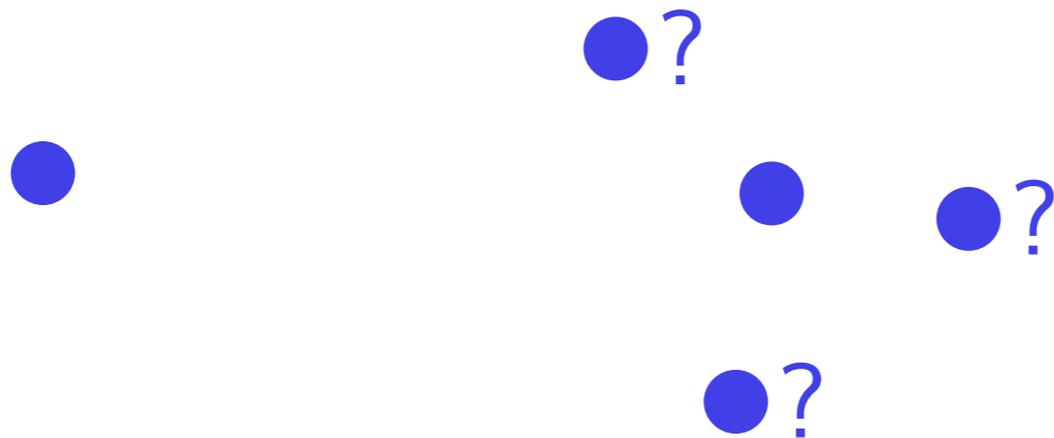
Computational geometry deals with problems on *precisely* specified *points* in \mathbb{R}^2 .

However, in many practical applications, locations of input points are *not* precise.



Computational geometry deals with problems on *precisely* specified *points* in \mathbb{R}^2 .

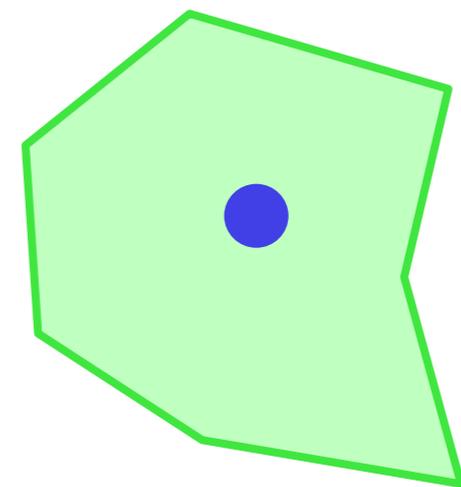
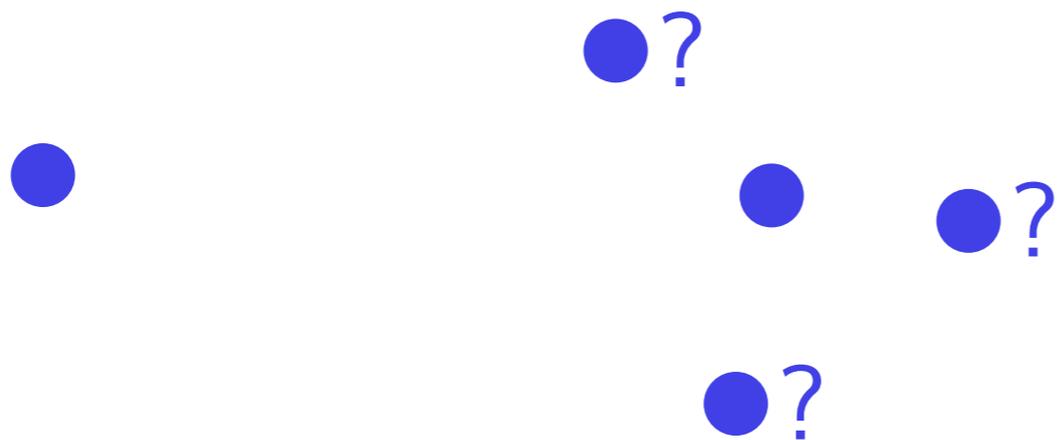
However, in many practical applications, locations of input points are *not* precise.



Computational geometry deals with problems on *precisely* specified *points* in \mathbb{R}^2 .

However, in many practical applications, locations of input points are *not* precise.

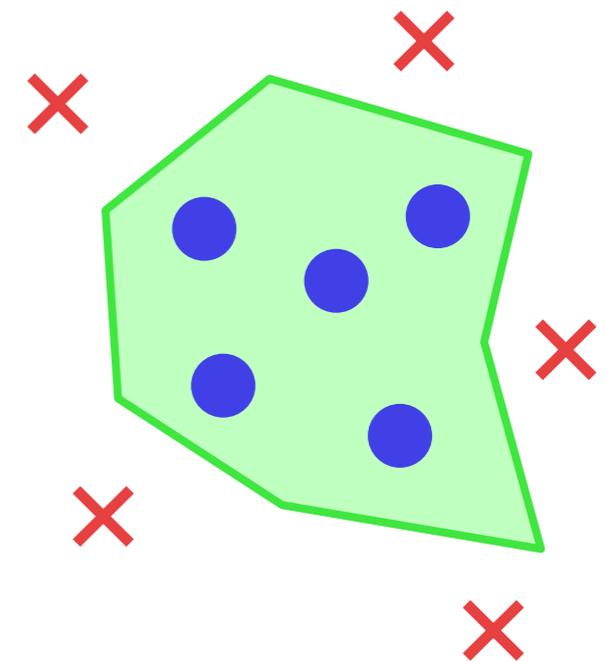
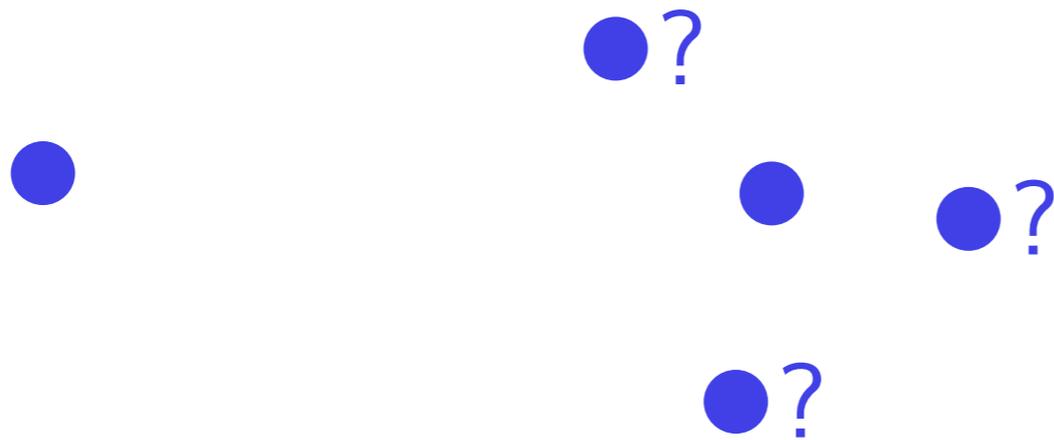
Often, a bound is available: a point is at least certain to be in a given *region*.



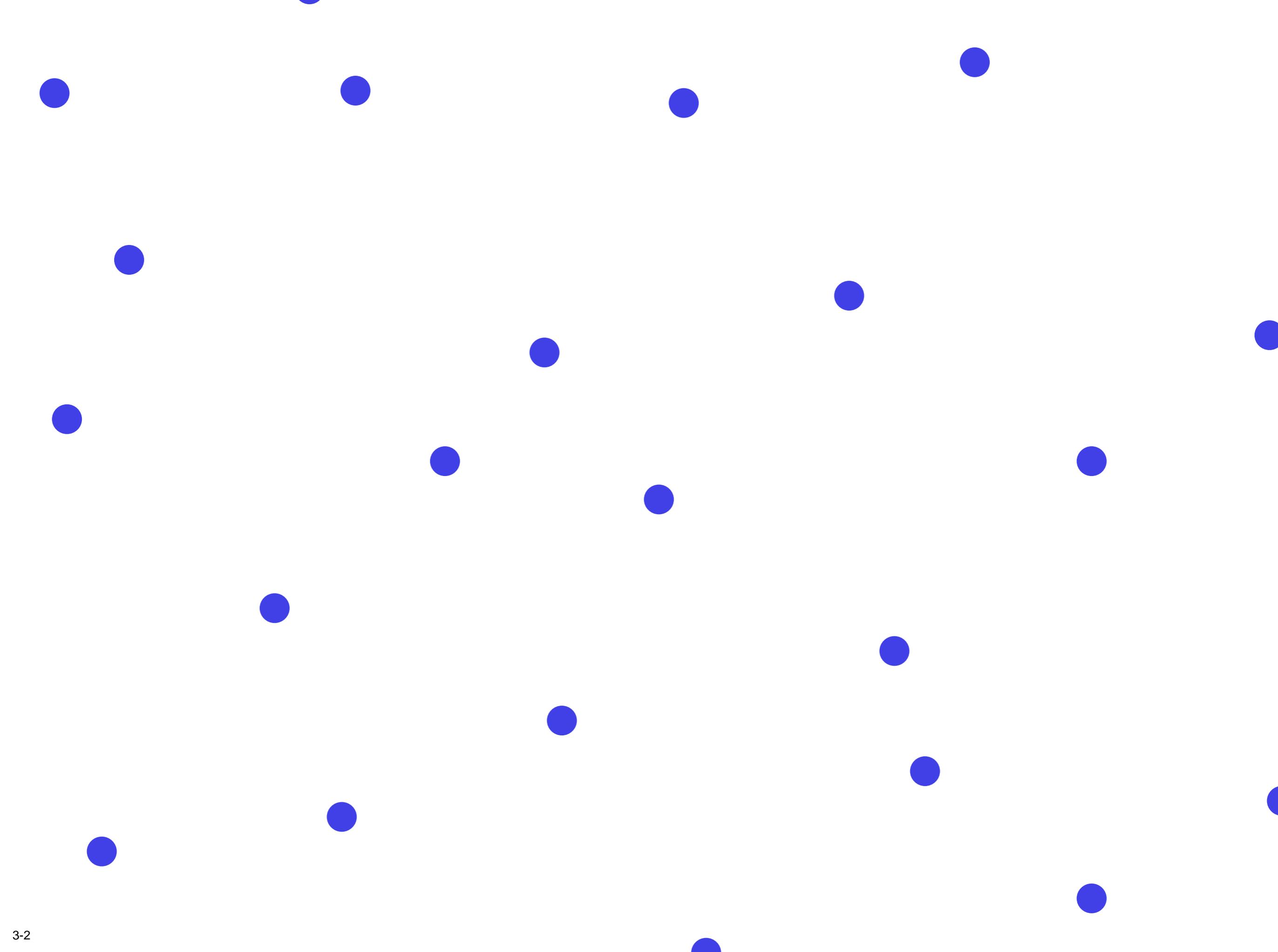
Computational geometry deals with problems on *precisely* specified *points* in \mathbb{R}^2 .

However, in many practical applications, locations of input points are *not* precise.

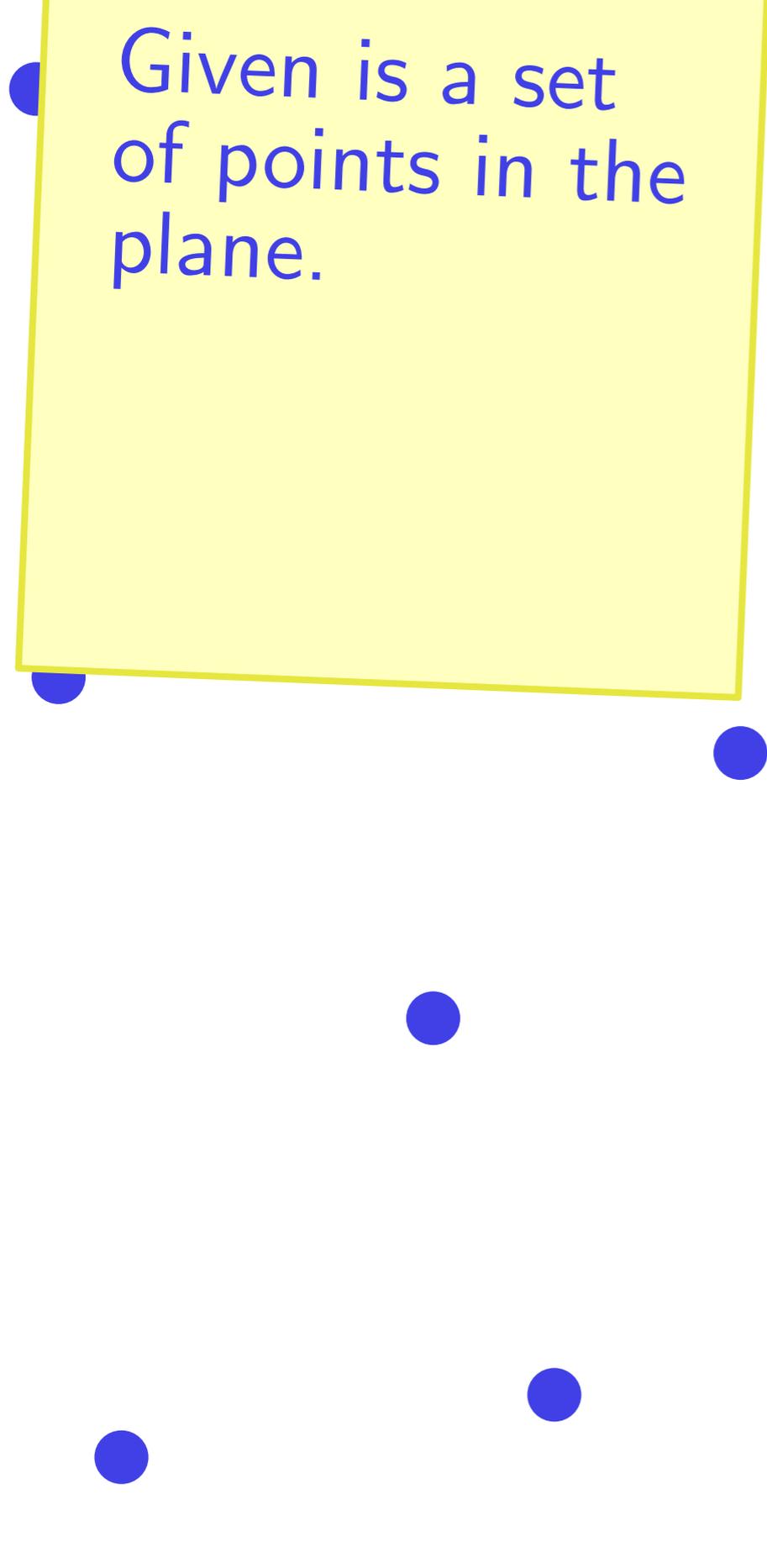
Often, a bound is available: a point is at least certain to be in a given *region*.



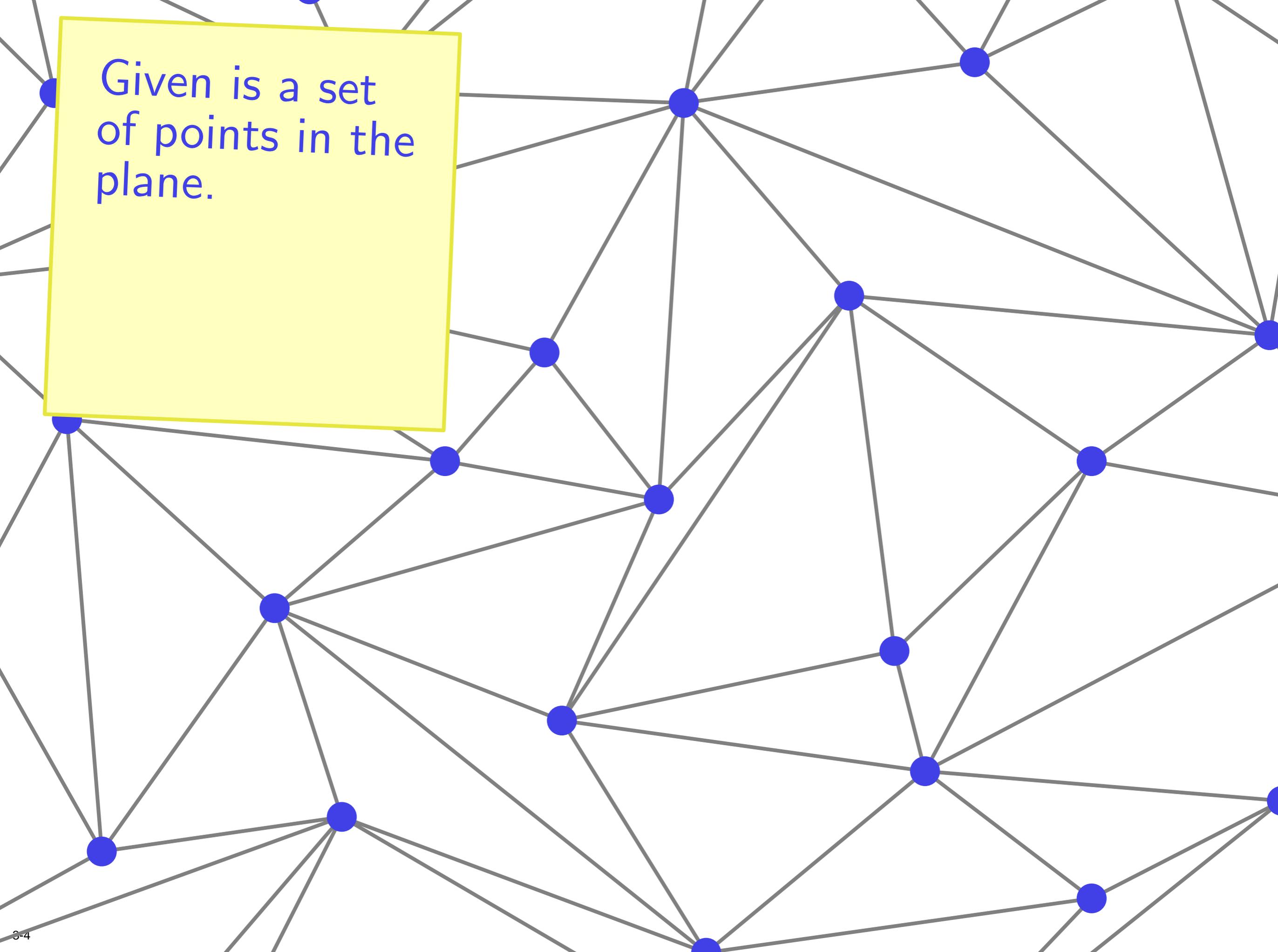
Then, what is
a
triangulation?

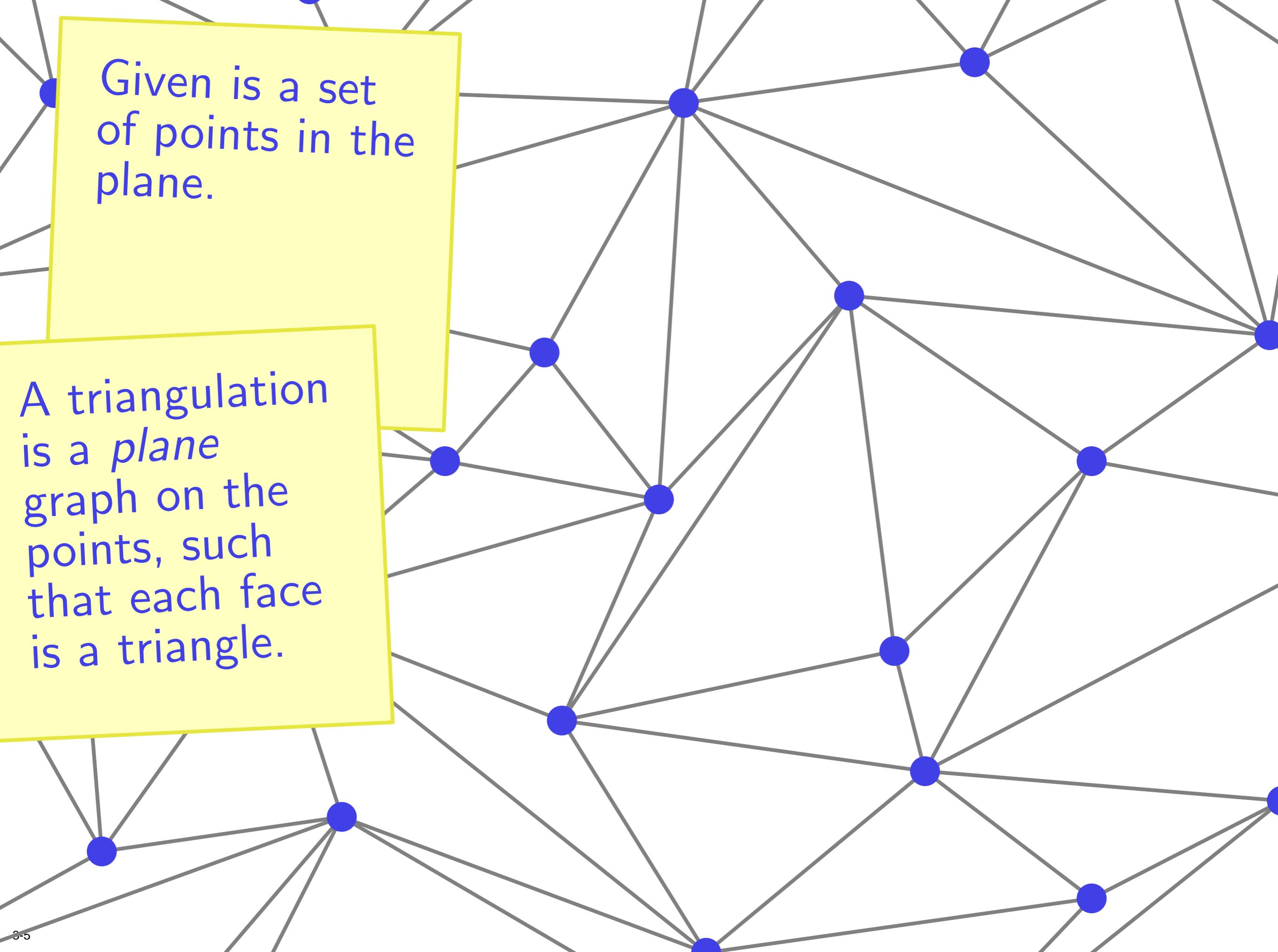


Given is a set
of points in the
plane.



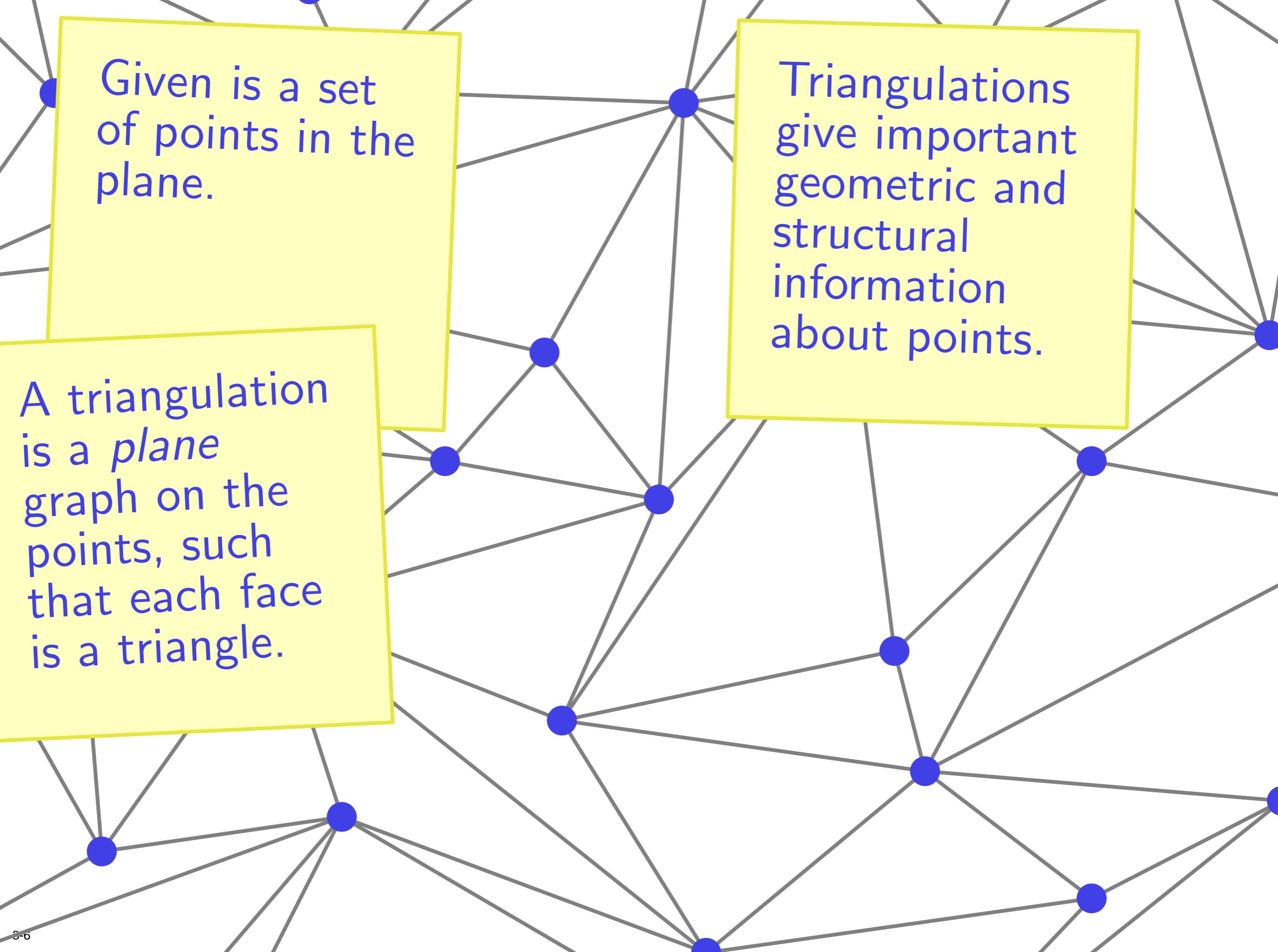
Given is a set
of points in the
plane.





Given is a set of points in the plane.

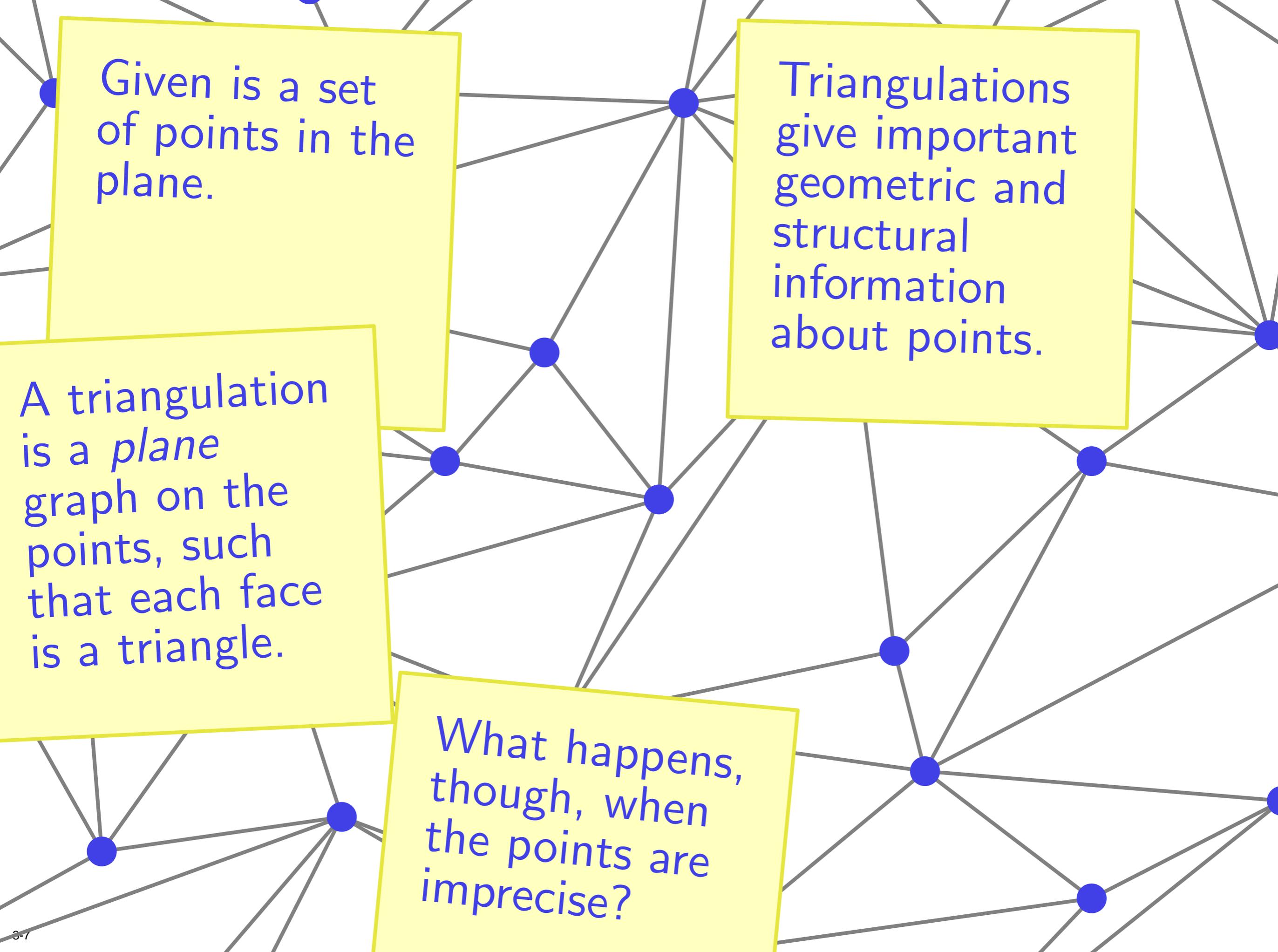
A triangulation is a *plane* graph on the points, such that each face is a triangle.



Given is a set of points in the plane.

Triangulations give important geometric and structural information about points.

A triangulation is a *plane* graph on the points, such that each face is a triangle.

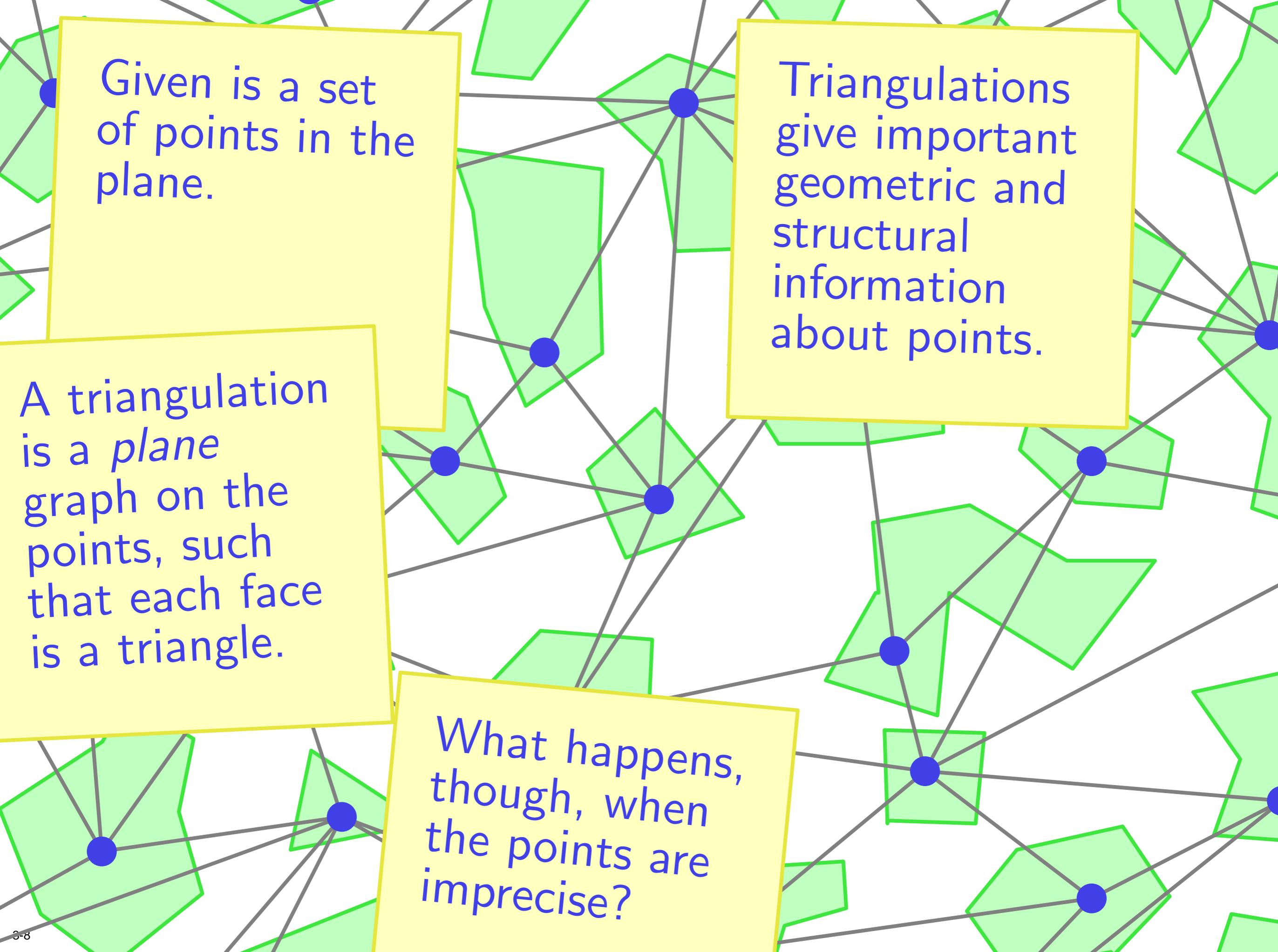


Given is a set of points in the plane.

Triangulations give important geometric and structural information about points.

A triangulation is a *plane* graph on the points, such that each face is a triangle.

What happens, though, when the points are imprecise?

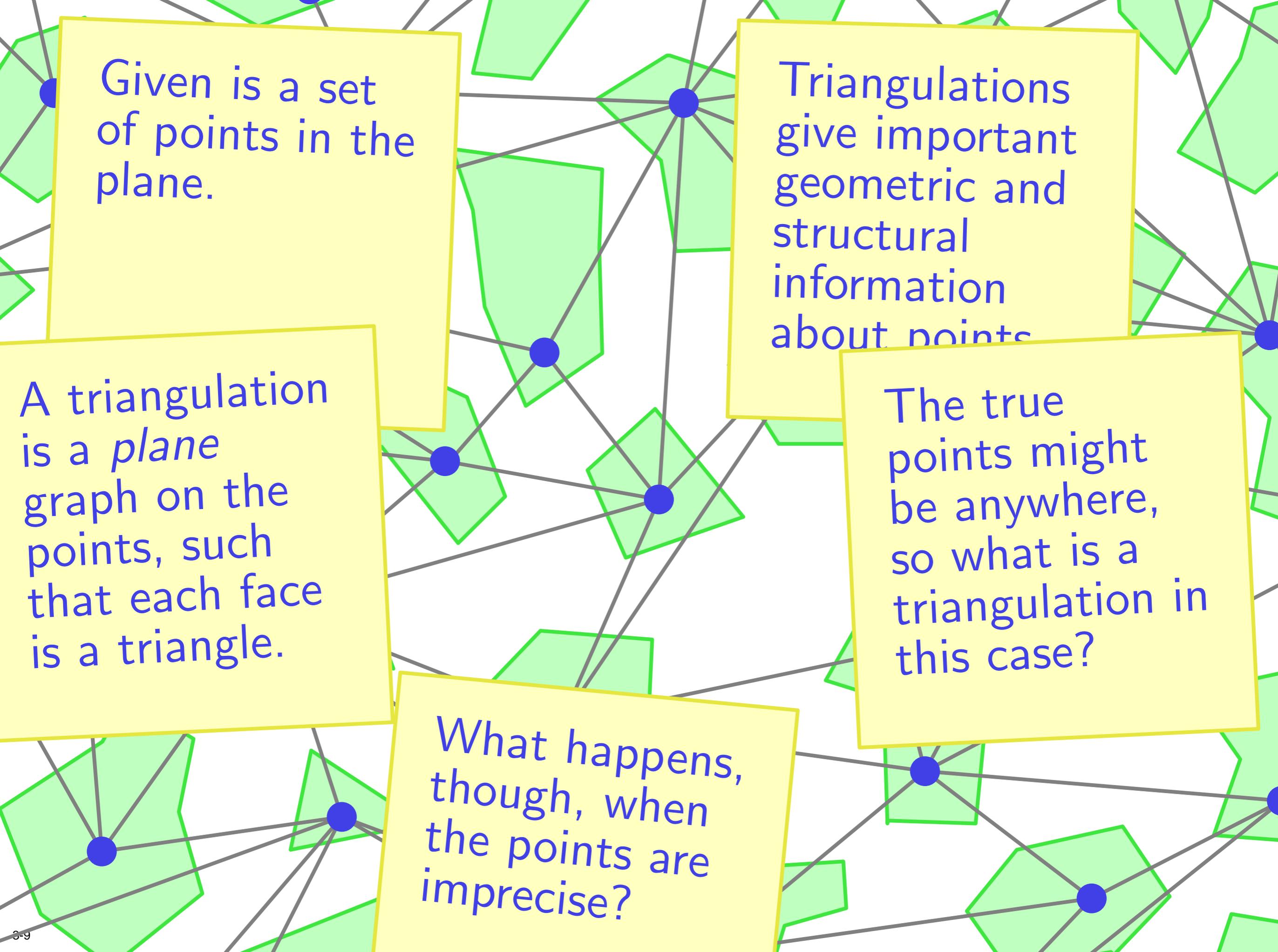


Given is a set of points in the plane.

Triangulations give important geometric and structural information about points.

A triangulation is a *plane* graph on the points, such that each face is a triangle.

What happens, though, when the points are imprecise?



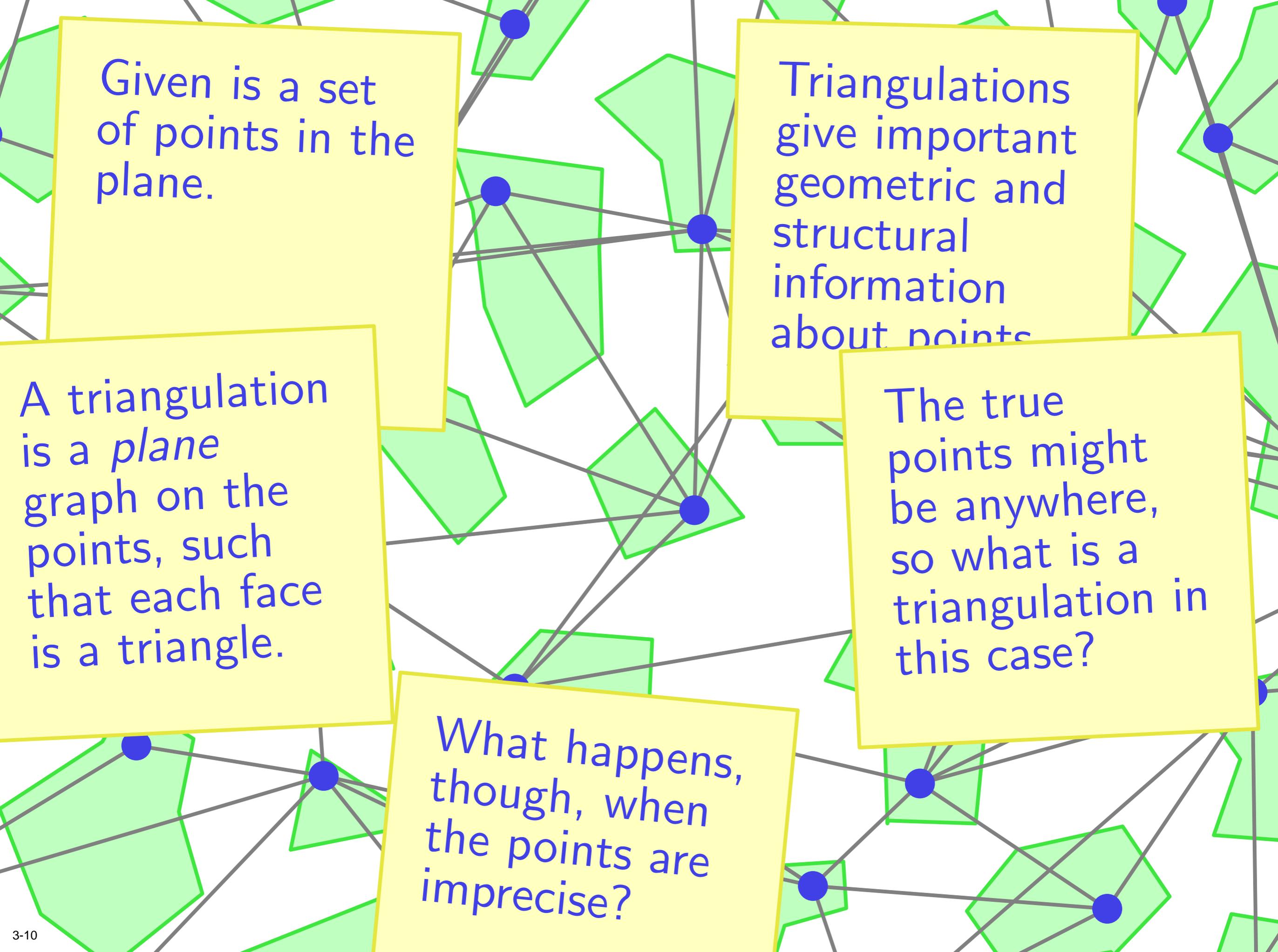
Given is a set of points in the plane.

Triangulations give important geometric and structural information about points.

A triangulation is a *plane* graph on the points, such that each face is a triangle.

The true points might be anywhere, so what is a triangulation in this case?

What happens, though, when the points are imprecise?



Given is a set of points in the plane.

Triangulations give important geometric and structural information about points.

A triangulation is a *plane* graph on the points, such that each face is a triangle.

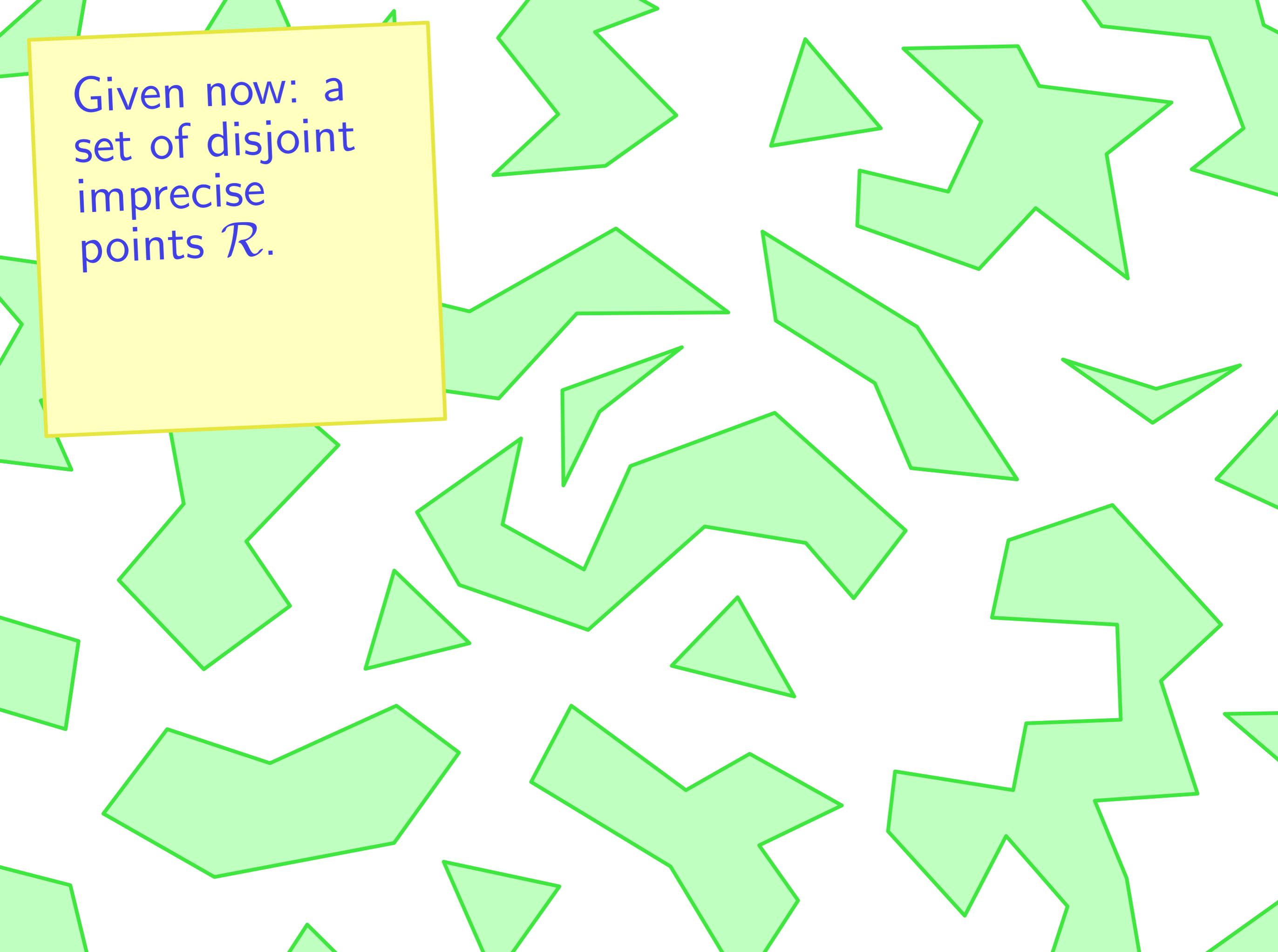
The true points might be anywhere, so what is a triangulation in this case?

What happens, though, when the points are imprecise?

Let's define a
problem
statement.

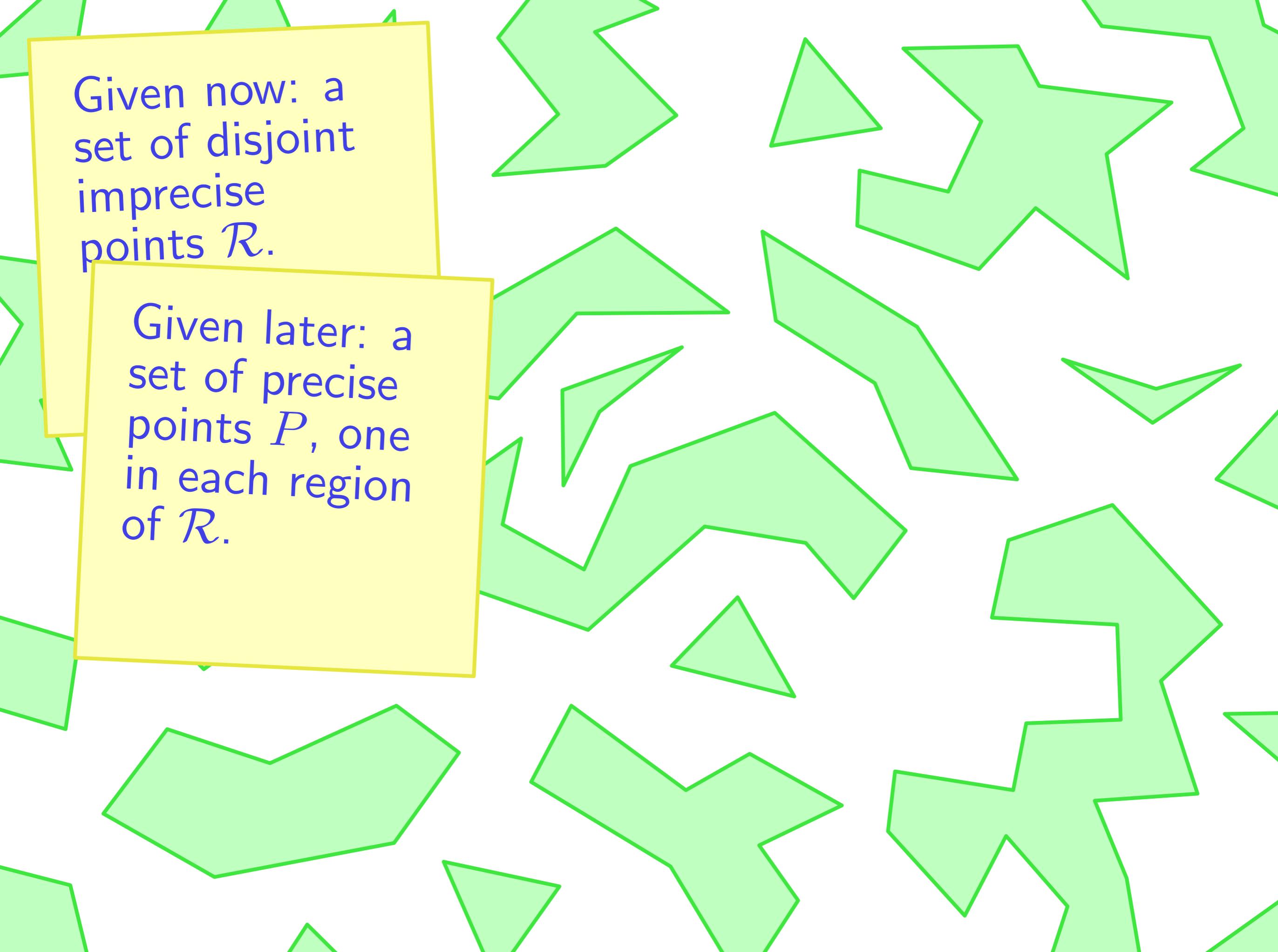
Given now: a
set of disjoint
imprecise
points \mathcal{R} .

Given now: a
set of disjoint
imprecise
points \mathcal{R} .



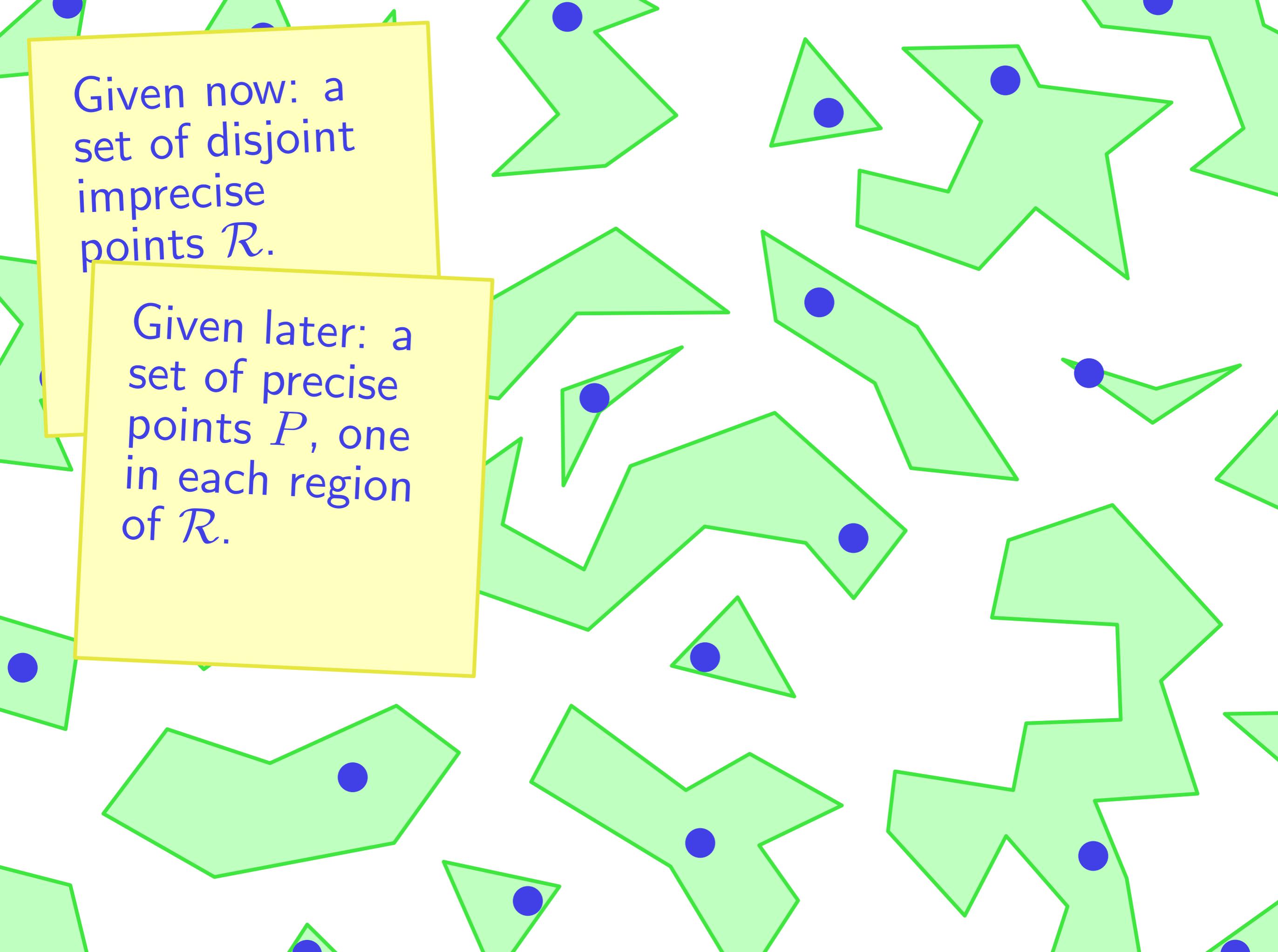
Given now: a set of disjoint imprecise points \mathcal{R} .

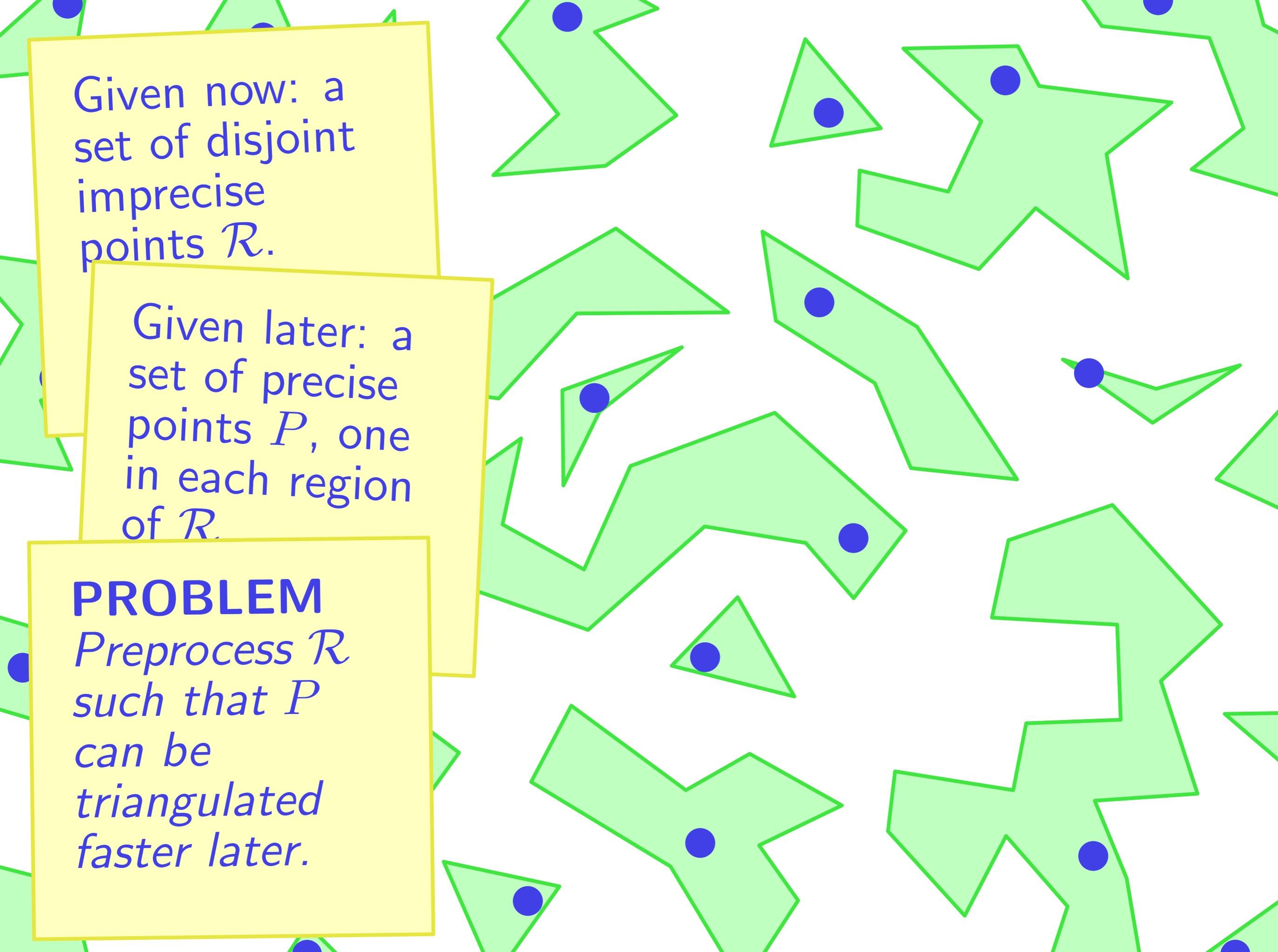
Given later: a set of precise points P , one in each region of \mathcal{R} .



Given now: a set of disjoint imprecise points \mathcal{R} .

Given later: a set of precise points P , one in each region of \mathcal{R} .



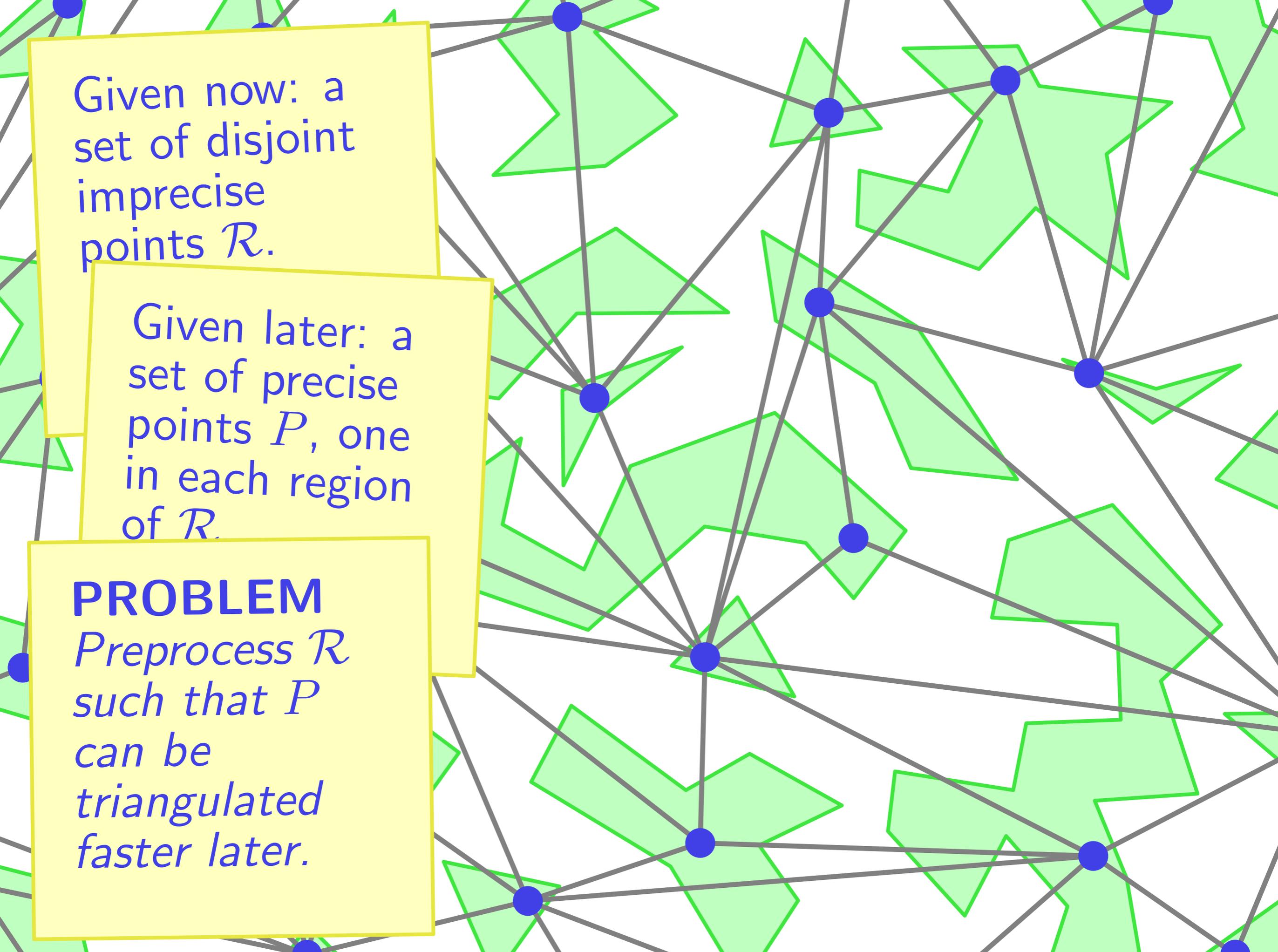
The background of the slide features a repeating pattern of light green, irregular polygons with dark green outlines. Each polygon contains a solid blue circular dot in its center. The polygons are scattered across the white background, creating a textured, abstract pattern.

Given now: a set of disjoint imprecise points \mathcal{R} .

Given later: a set of precise points P , one in each region of \mathcal{R} .

PROBLEM

Preprocess \mathcal{R} such that P can be triangulated faster later.

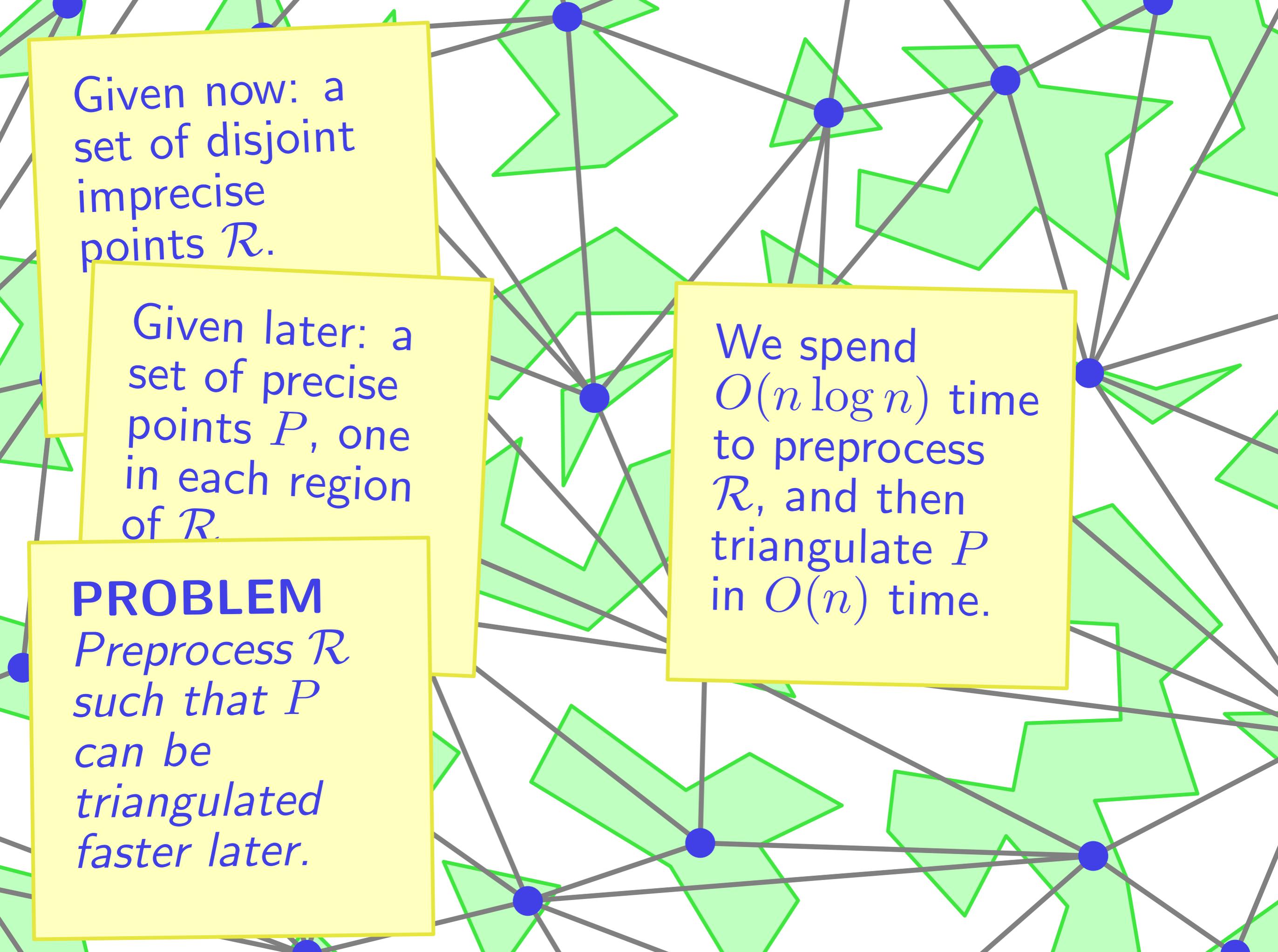


Given now: a set of disjoint imprecise points \mathcal{R} .

Given later: a set of precise points P , one in each region of \mathcal{R} .

PROBLEM

Preprocess \mathcal{R} such that P can be triangulated faster later.



Given now: a set of disjoint imprecise points \mathcal{R} .

Given later: a set of precise points P , one in each region of \mathcal{R} .

We spend $O(n \log n)$ time to preprocess \mathcal{R} , and then triangulate P in $O(n)$ time.

PROBLEM

Preprocess \mathcal{R} such that P can be triangulated faster later.

A few similar results have recently been obtained.

Given a set of disjoint discs, preprocess them to triangulate the points later.

Given a set of disjoint discs, preprocess them to triangulate the points later.

Preprocessing in $O(n \log n)$, reconstruction in $O(n)$.

[Held & Mitchell, 2008]

Given a set of disjoint discs, preprocess them to triangulate the points later.

Preprocessing in $O(n \log n)$, reconstruction in $O(n)$.

[Held & Mitchell, 2008]

Given a set of disjoint discs, preprocessing them for the *Delaunay* triangulation.

Given a set of disjoint discs, preprocess them to triangulate the points later.

Preprocessing in $O(n \log n)$, reconstruction in $O(n)$.

[Held & Mitchell, 2008]

Given a set of disjoint discs, preprocessing them for the *Delaunay* triangulation.

Preprocessing in $O(n \log n)$, reconstruction in $O(n)$.

[Löffler & Snoeyink, 2008]

Given a set of disjoint discs, preprocess them to triangulate the points later.

Given a set of disjoint discs, preprocessing them for the *Delaunay* triangulation.

Our improvement: not just discs, but *general* regions.

Preprocessing in $O(n \log n)$, reconstruction in $O(n)$.

[Held & Mitchell, 2008]

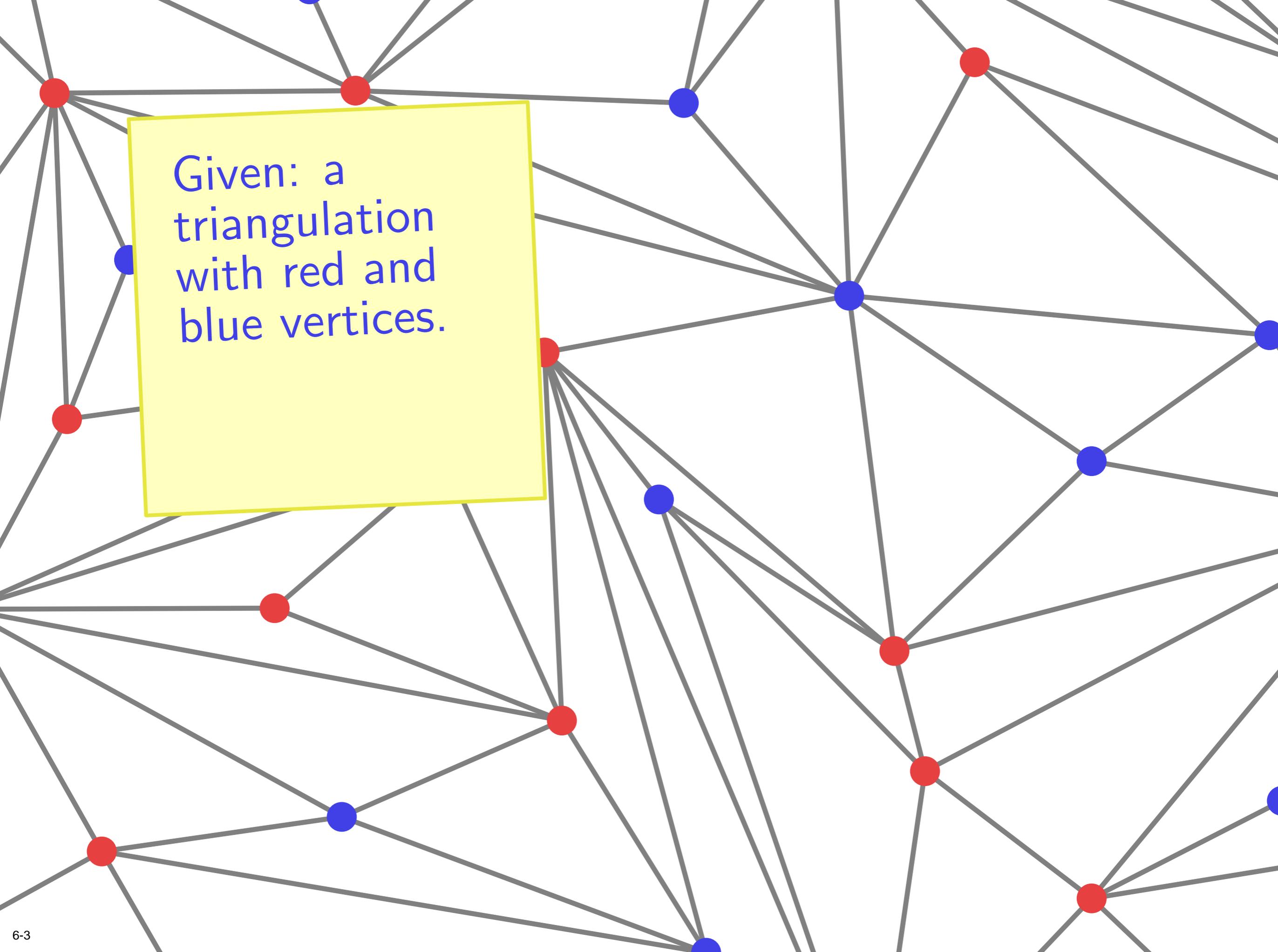
Preprocessing in $O(n \log n)$, reconstruction in $O(n)$.

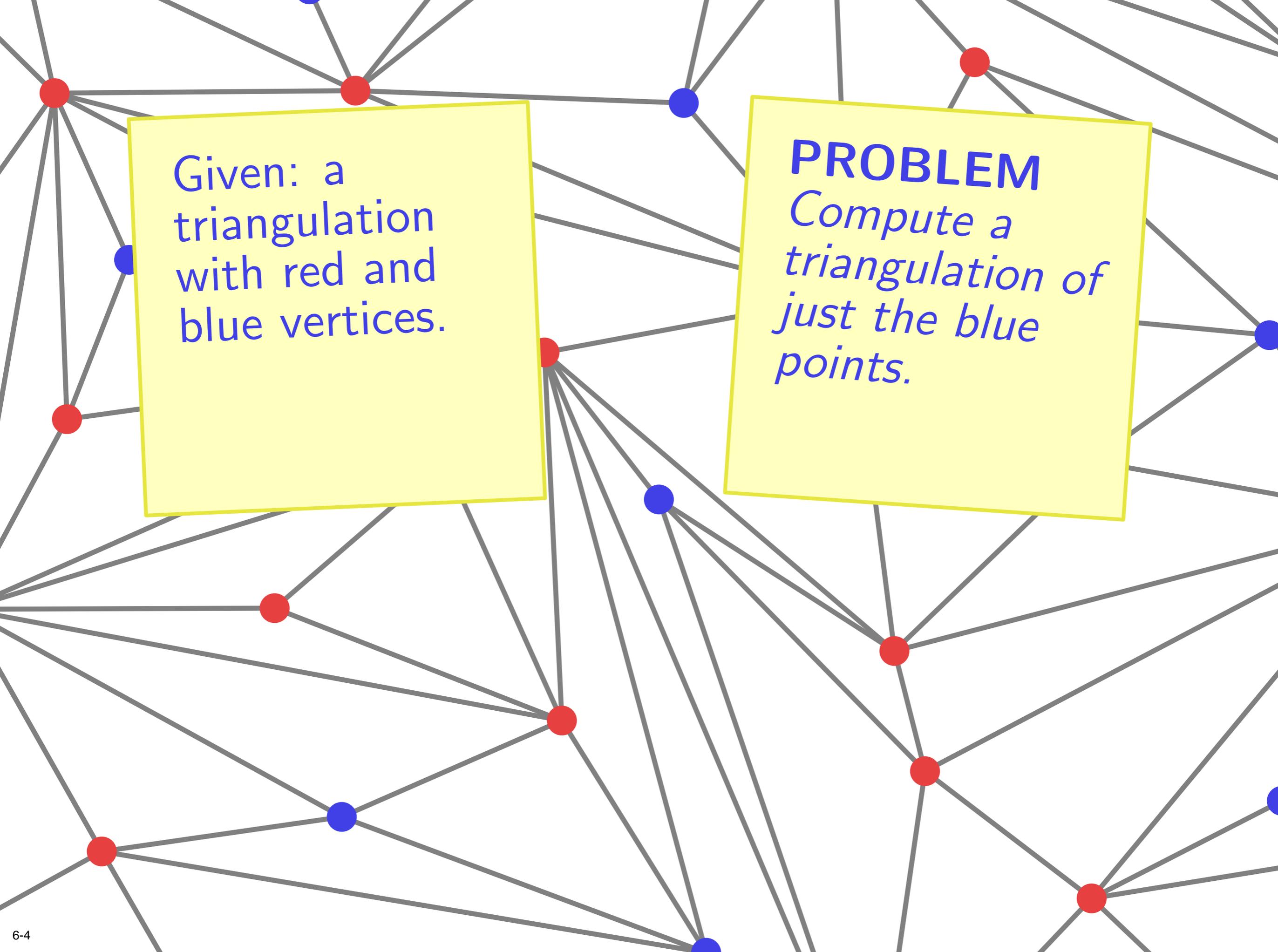
[Löffler & Snoeyink, 2008]

As the main
tool in our
solution, we
solve a different
problem.

Given: a
triangulation
with red and
blue vertices.

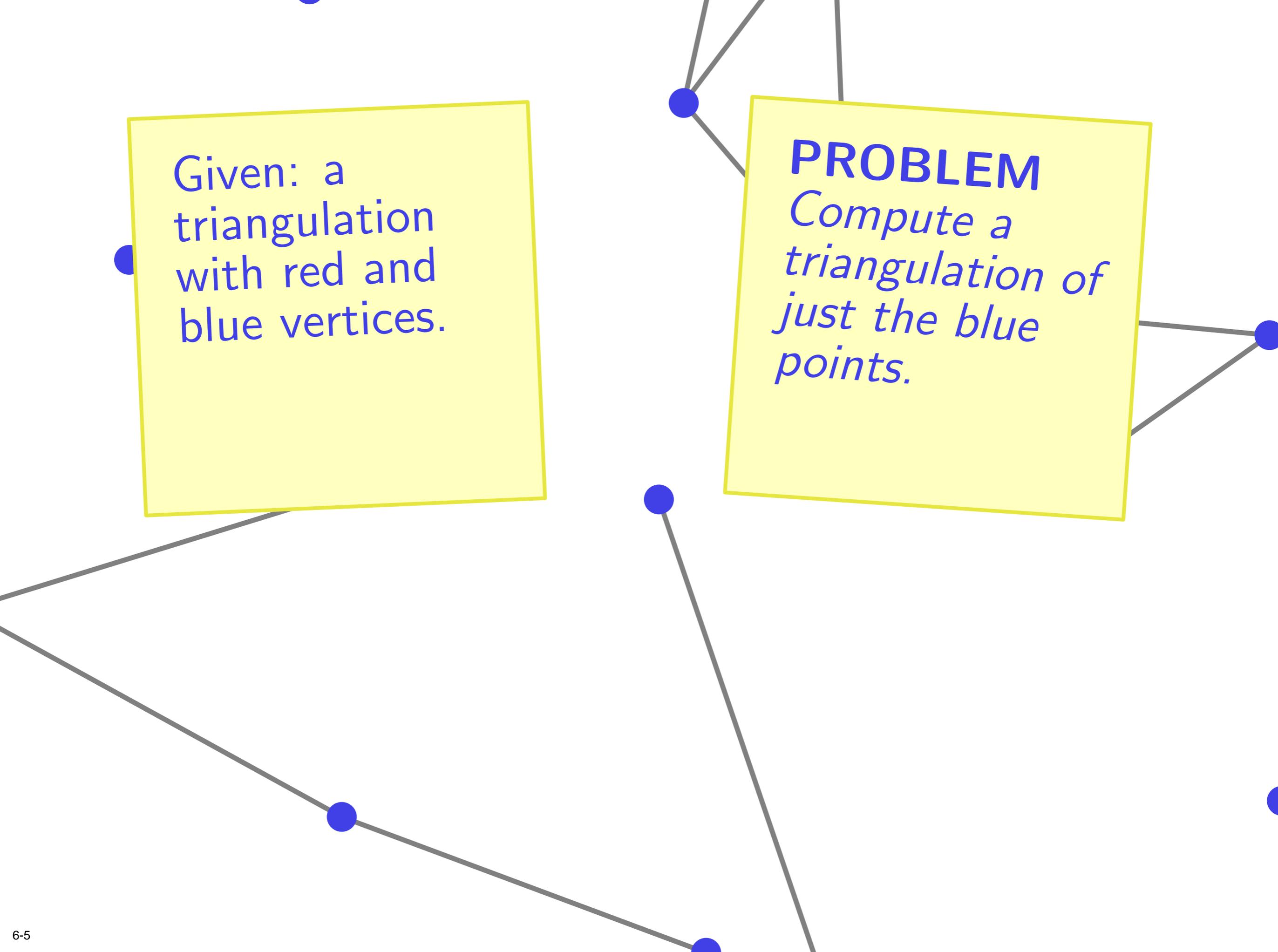
Given: a triangulation with red and blue vertices.





Given: a
triangulation
with red and
blue vertices.

PROBLEM
*Compute a
triangulation of
just the blue
points.*

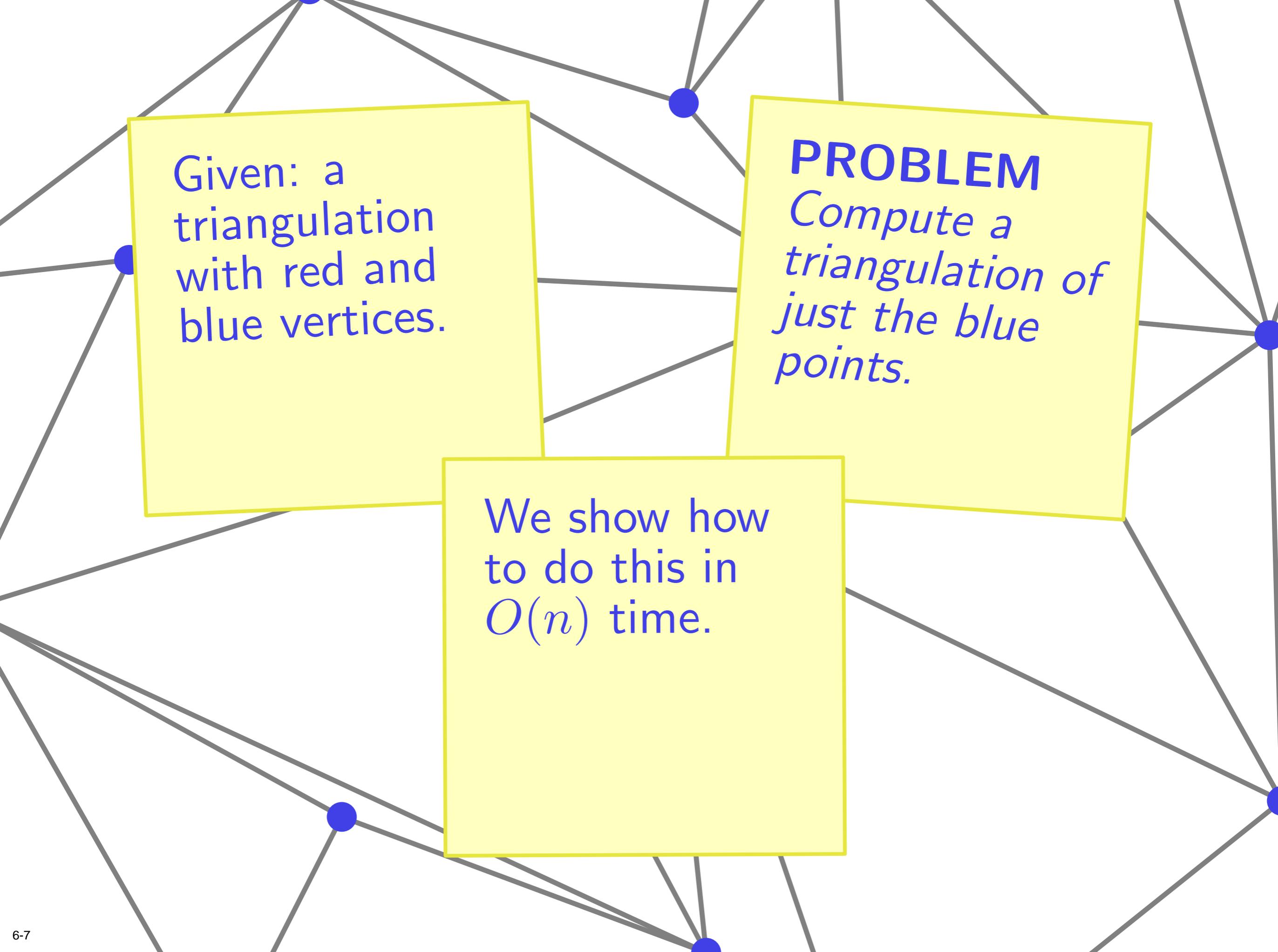


Given: a
triangulation
with red and
blue vertices.

PROBLEM
*Compute a
triangulation of
just the blue
points.*

Given: a
triangulation
with red and
blue vertices.

PROBLEM
*Compute a
triangulation of
just the blue
points.*



Given: a triangulation with red and blue vertices.

PROBLEM
Compute a triangulation of just the blue points.

We show how to do this in $O(n)$ time.

Idea: remove
constant
degree red
vertices one by
one and
retriangulate.

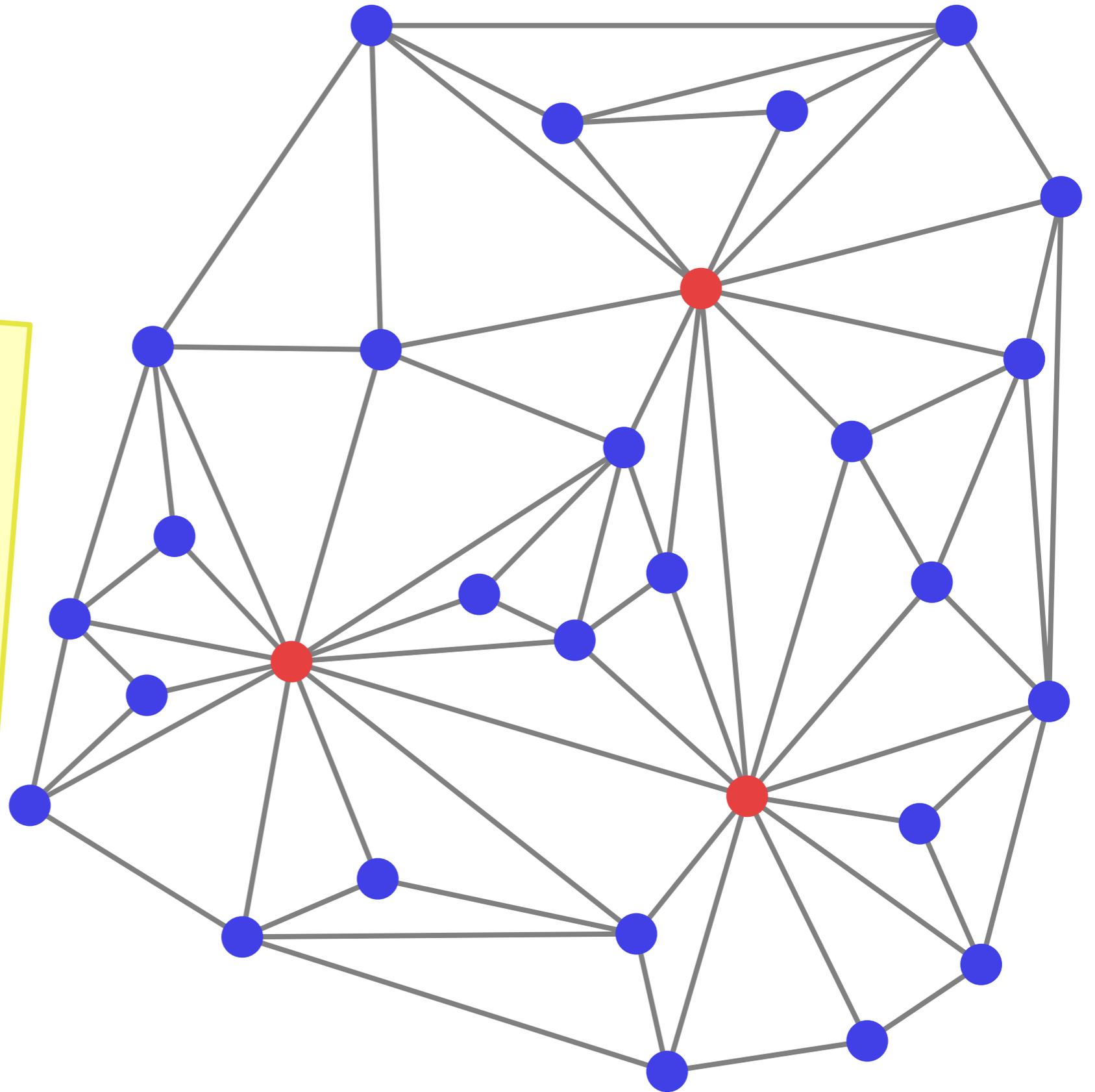
There is always
a vertex of
constant
degree.

There is always
a vertex of
constant
degree.

Slight problem:
they might all
be blue.

There is always
a vertex of
constant
degree.

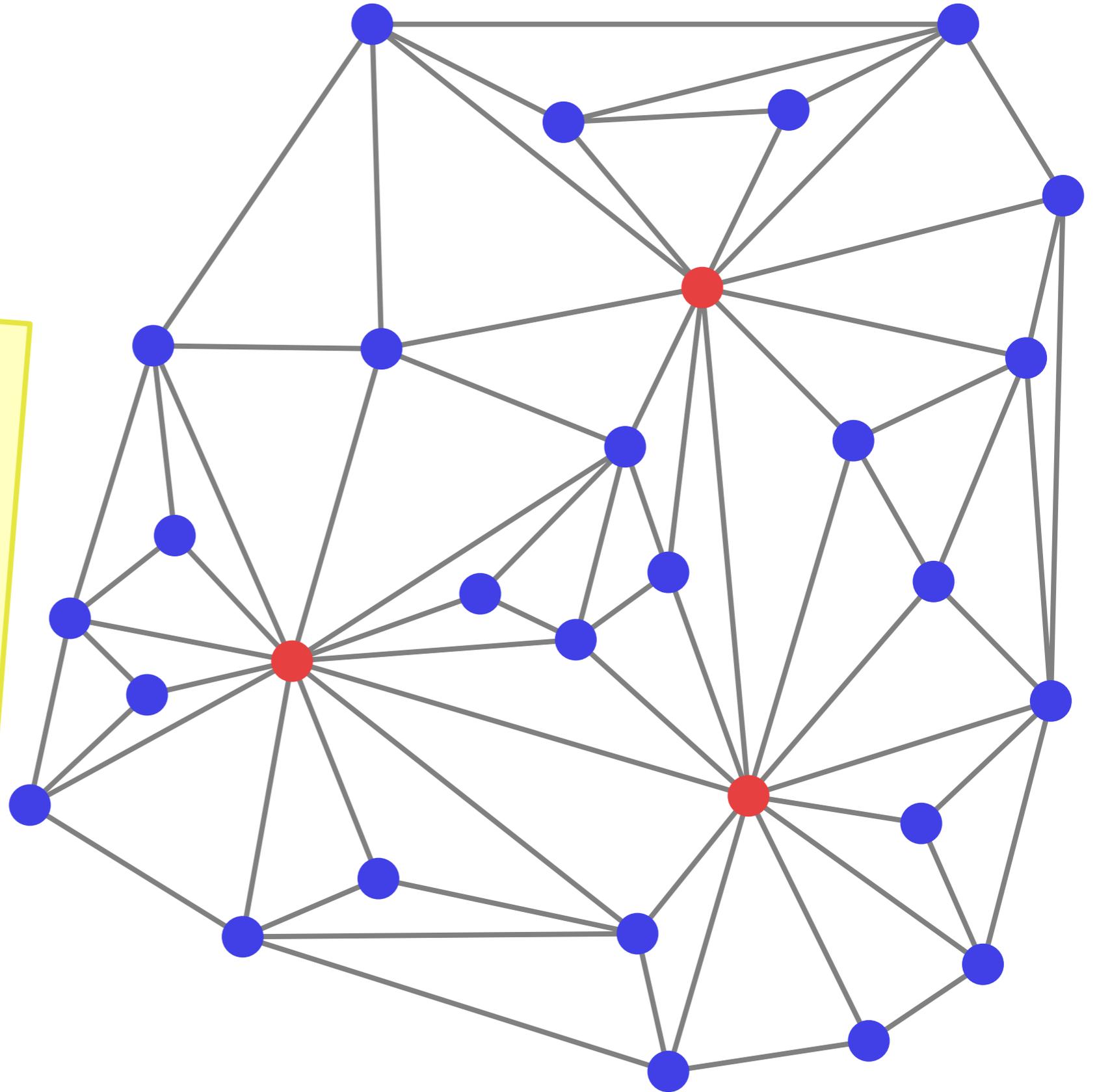
Slight problem:
they might all
be blue.



There is always
a vertex of
constant
degree.

Slight problem:
they might all
be blue.

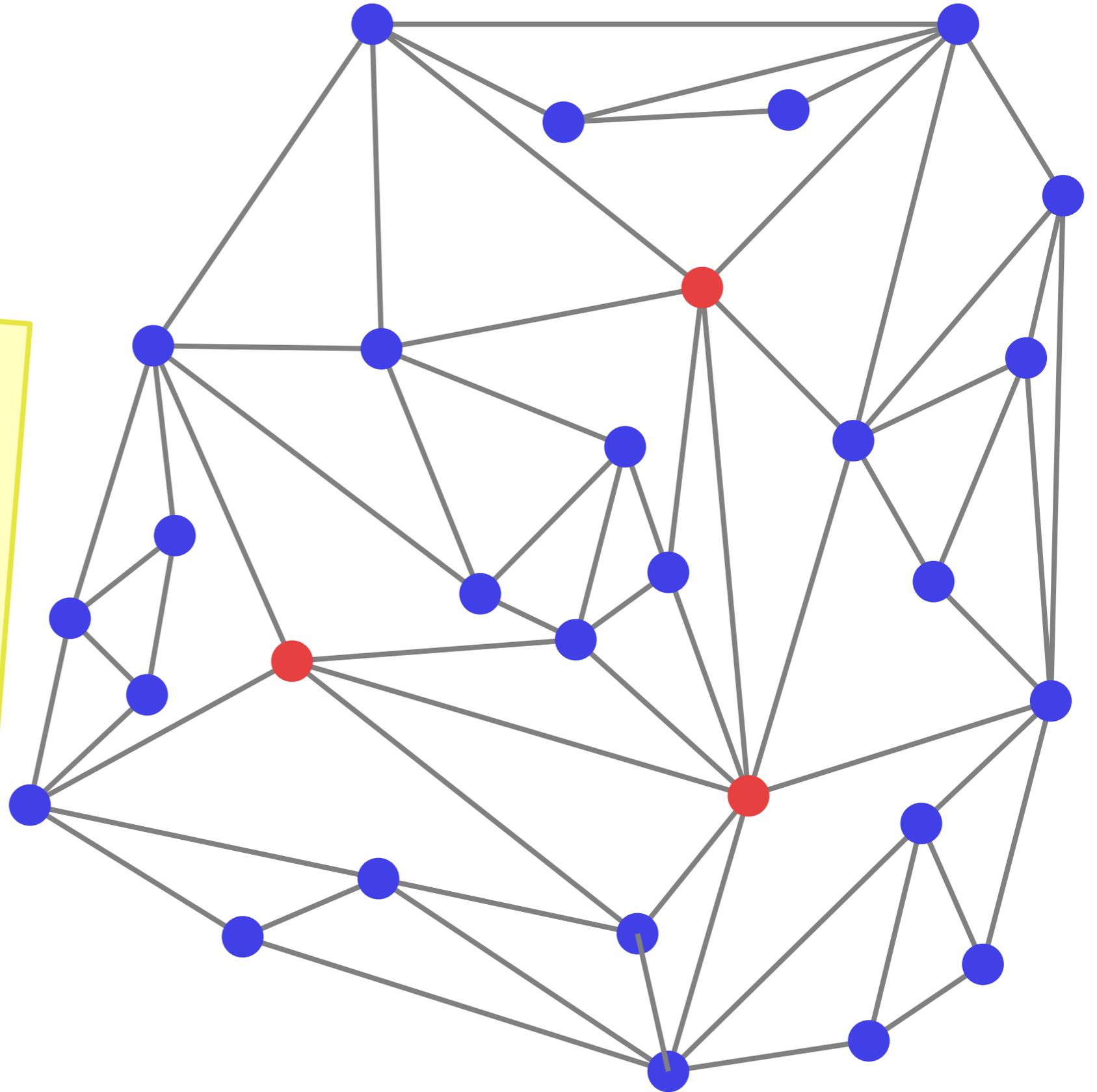
Instead, we
maintain a
pseudo-
triangulation.



There is always
a vertex of
constant
degree.

Slight problem:
they might all
be blue.

Instead, we
maintain a
pseudo-
triangulation.

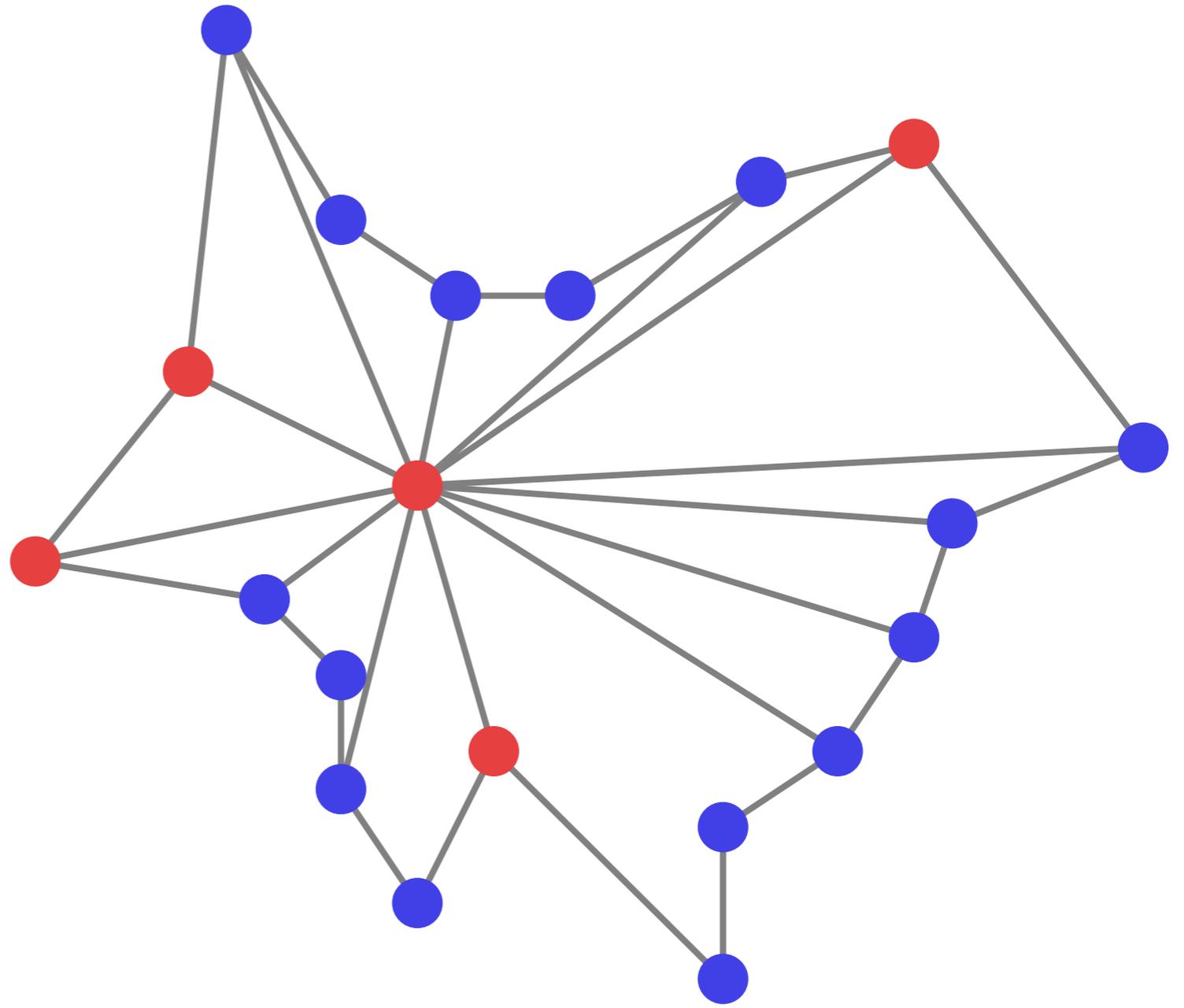


We need a way
to simplify a
red point in
time linear in
its degree.

A red vertex v
should not
have many
more blue than
red neighbours.

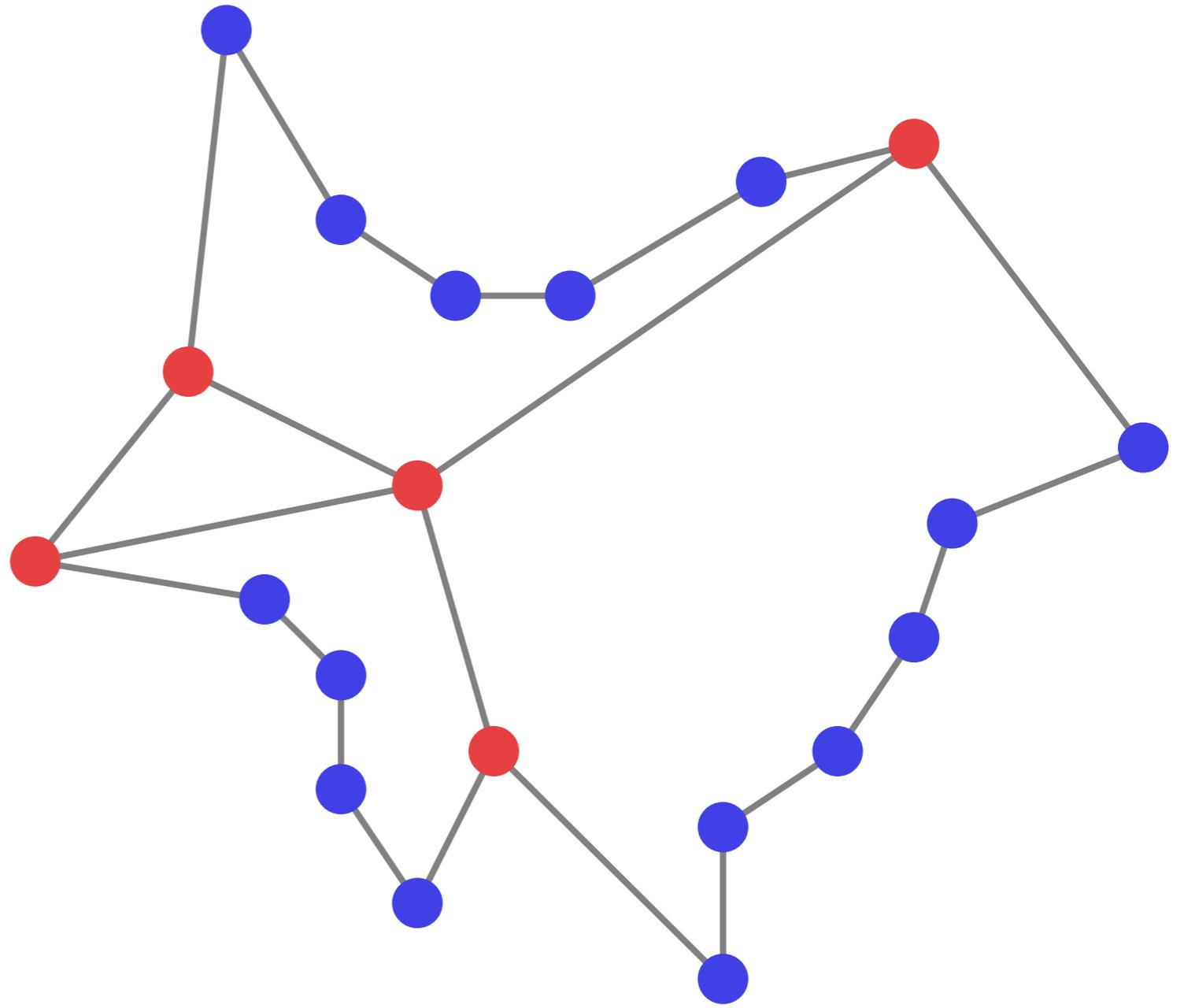
A red vertex v should not have many more blue than red neighbours.

Remove all red-blue edges of v .



A red vertex v should not have many more blue than red neighbours.

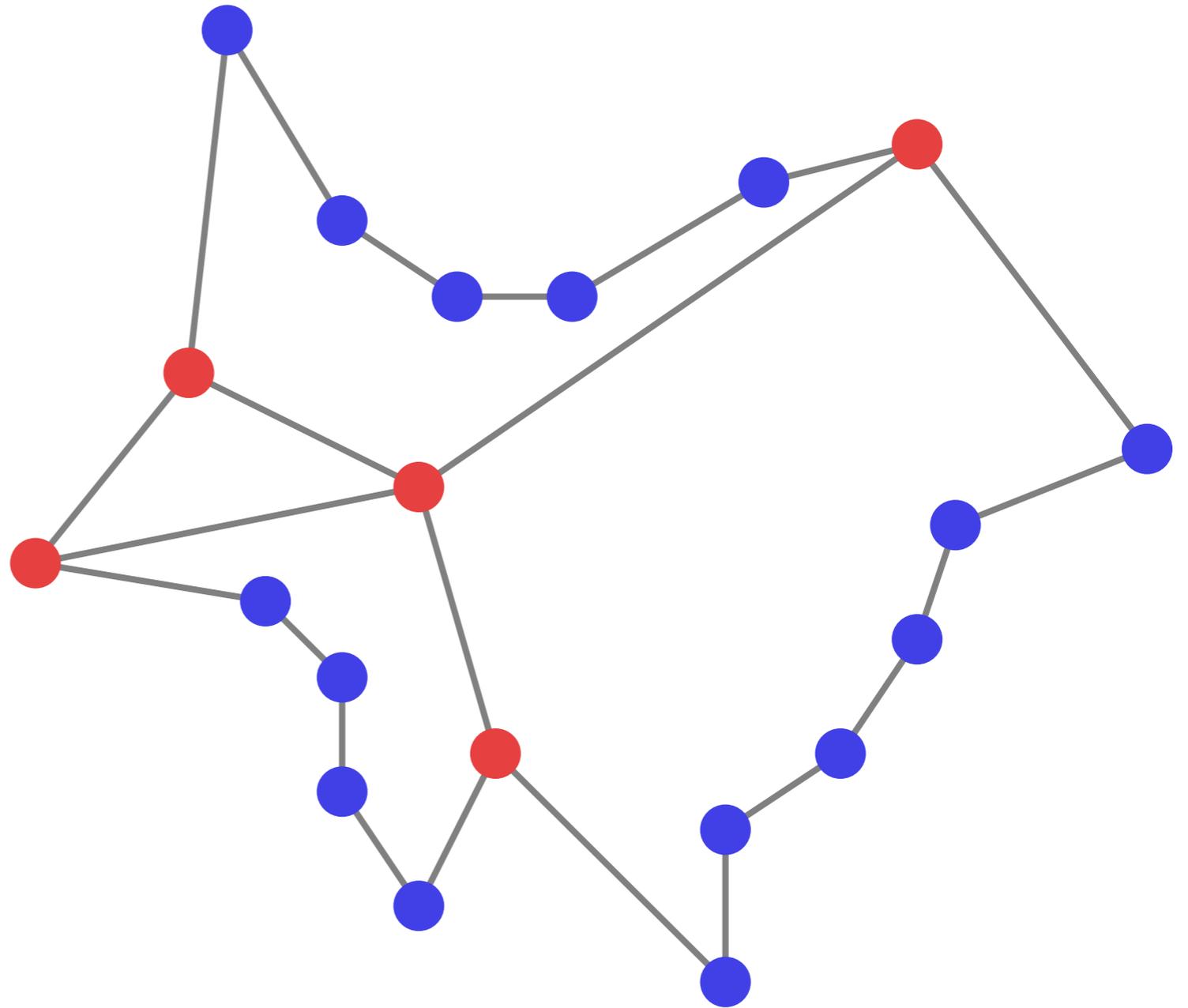
Remove all red-blue edges of v .



A red vertex v should not have many more blue than red neighbours.

Remove all red-blue edges of v .

Add edges to the blue neighbours of red-red edges.

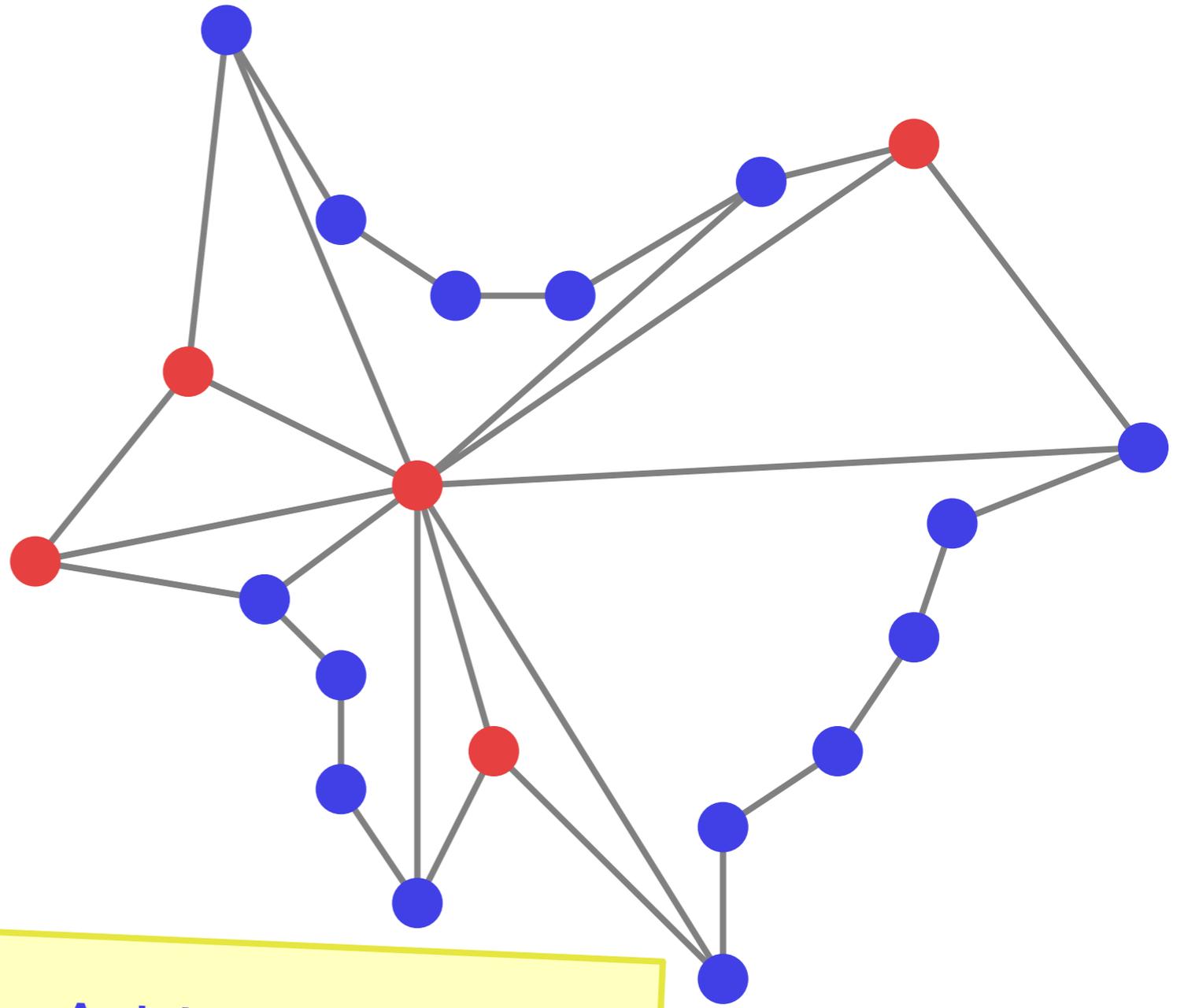


A red vertex v should not have many more blue than red neighbours.

Remove all red-blue edges of v .

Add edges to the blue neighbours of red-red edges.

Add convex paths connecting them.

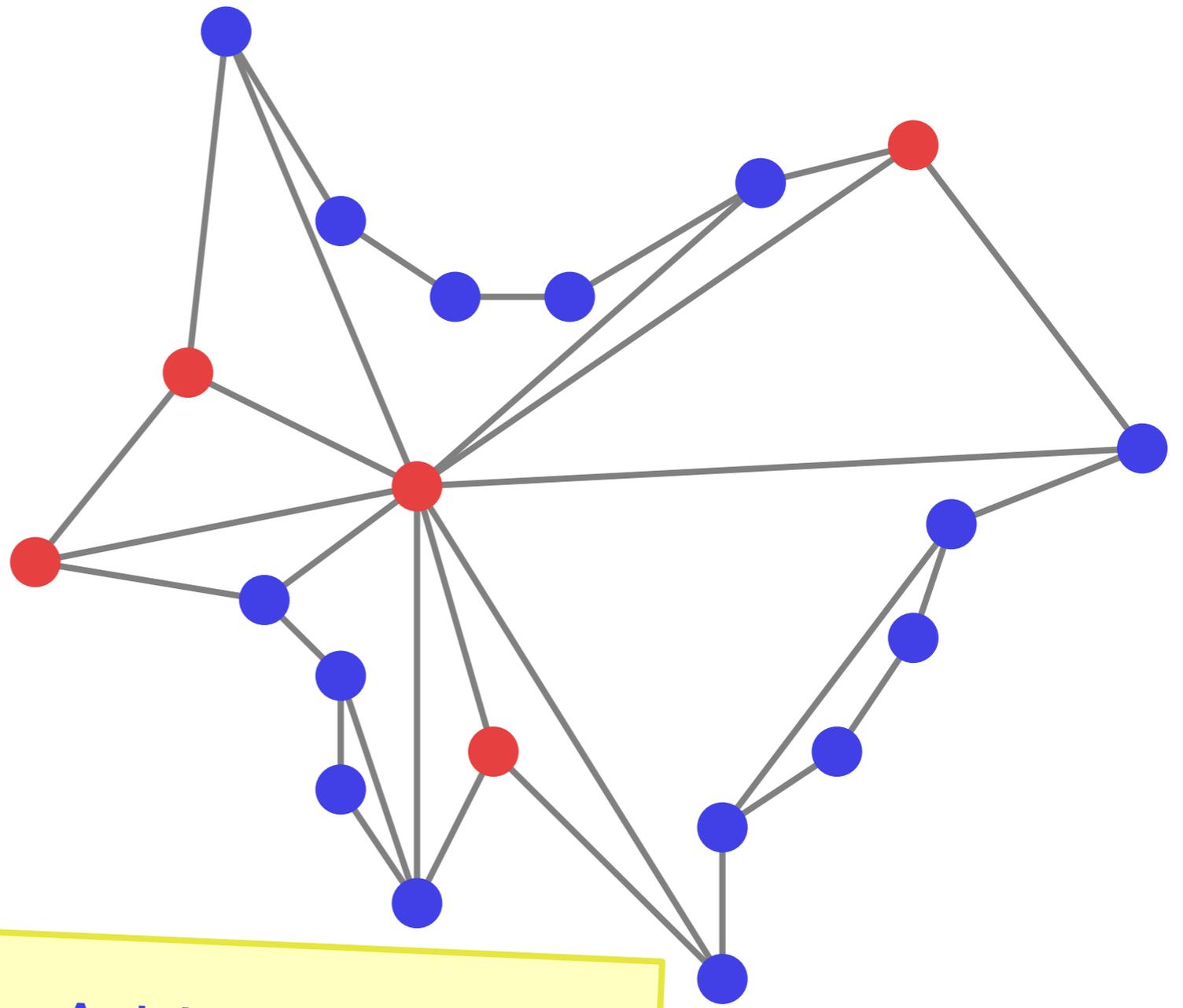


A red vertex v should not have many more blue than red neighbours.

Remove all red-blue edges of v .

Add edges to the blue neighbours of red-red edges.

Add convex paths connecting them.



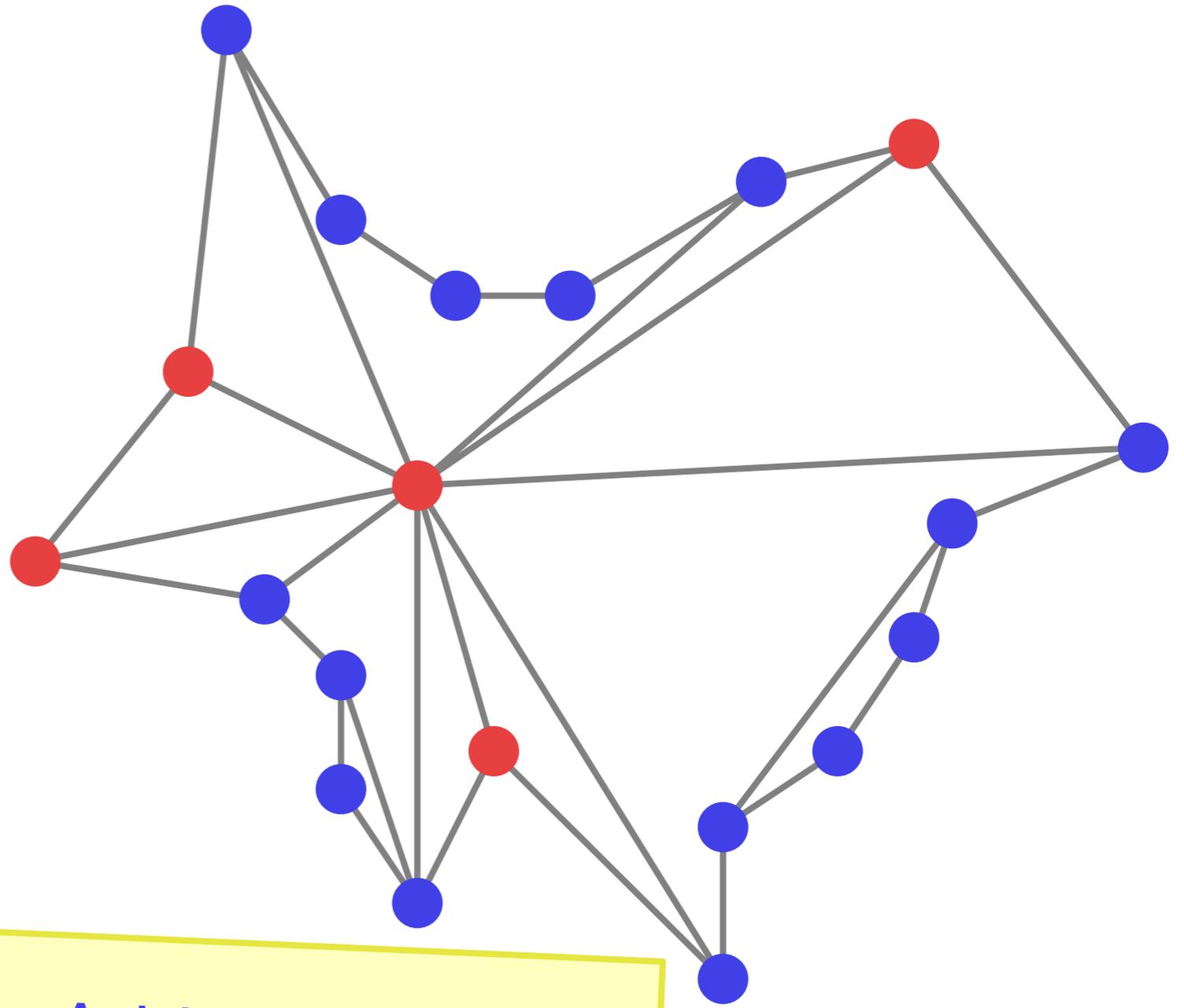
A red vertex v should not have many more blue than red neighbours.

Remove all red-blue edges of v .

Add edges to the blue neighbours of red-red edges.

Add convex paths connecting them.

Finally, triangulate the remaining blue regions.



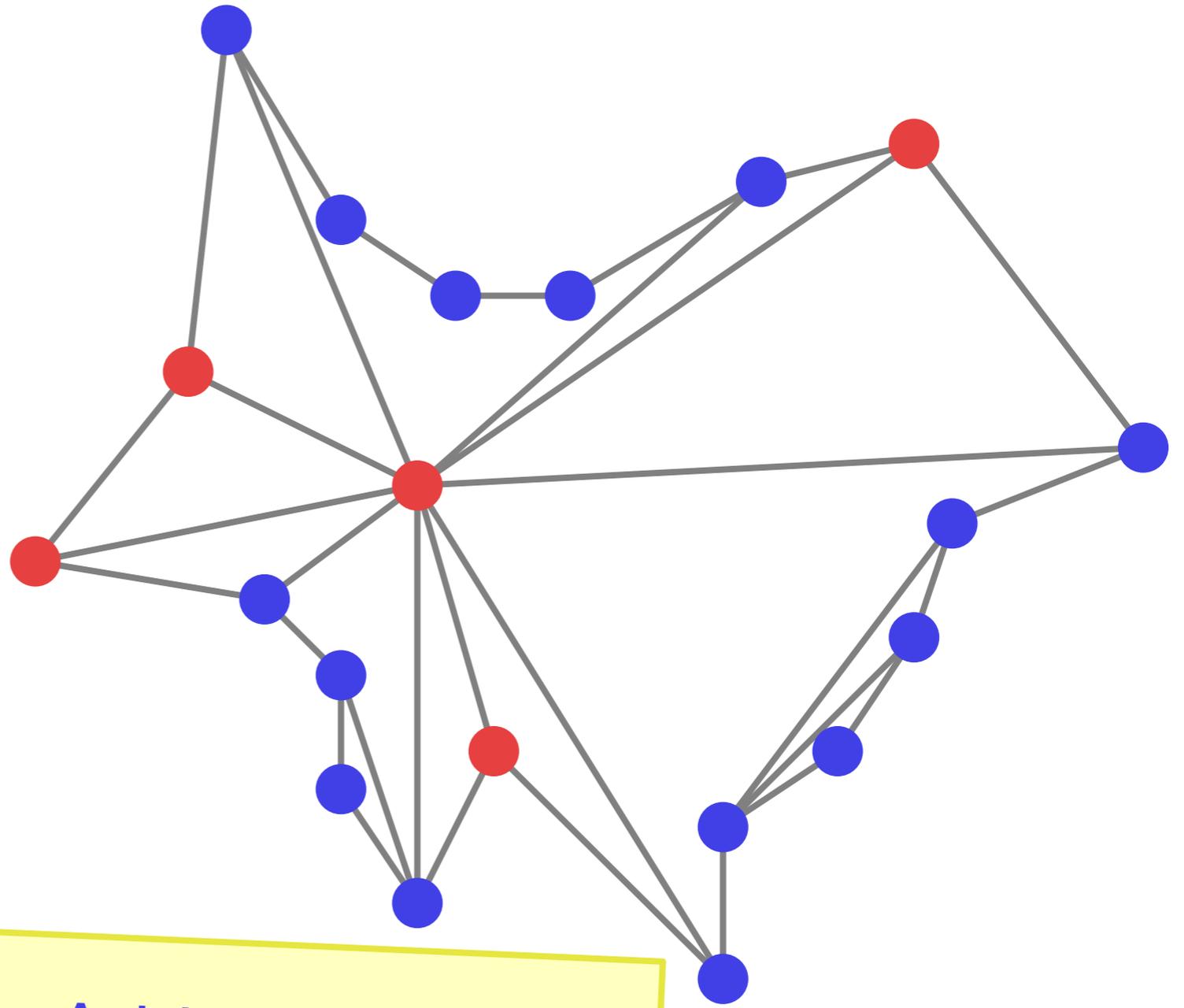
A red vertex v should not have many more blue than red neighbours.

Remove all red-blue edges of v .

Add edges to the blue neighbours of red-red edges.

Add convex paths connecting them.

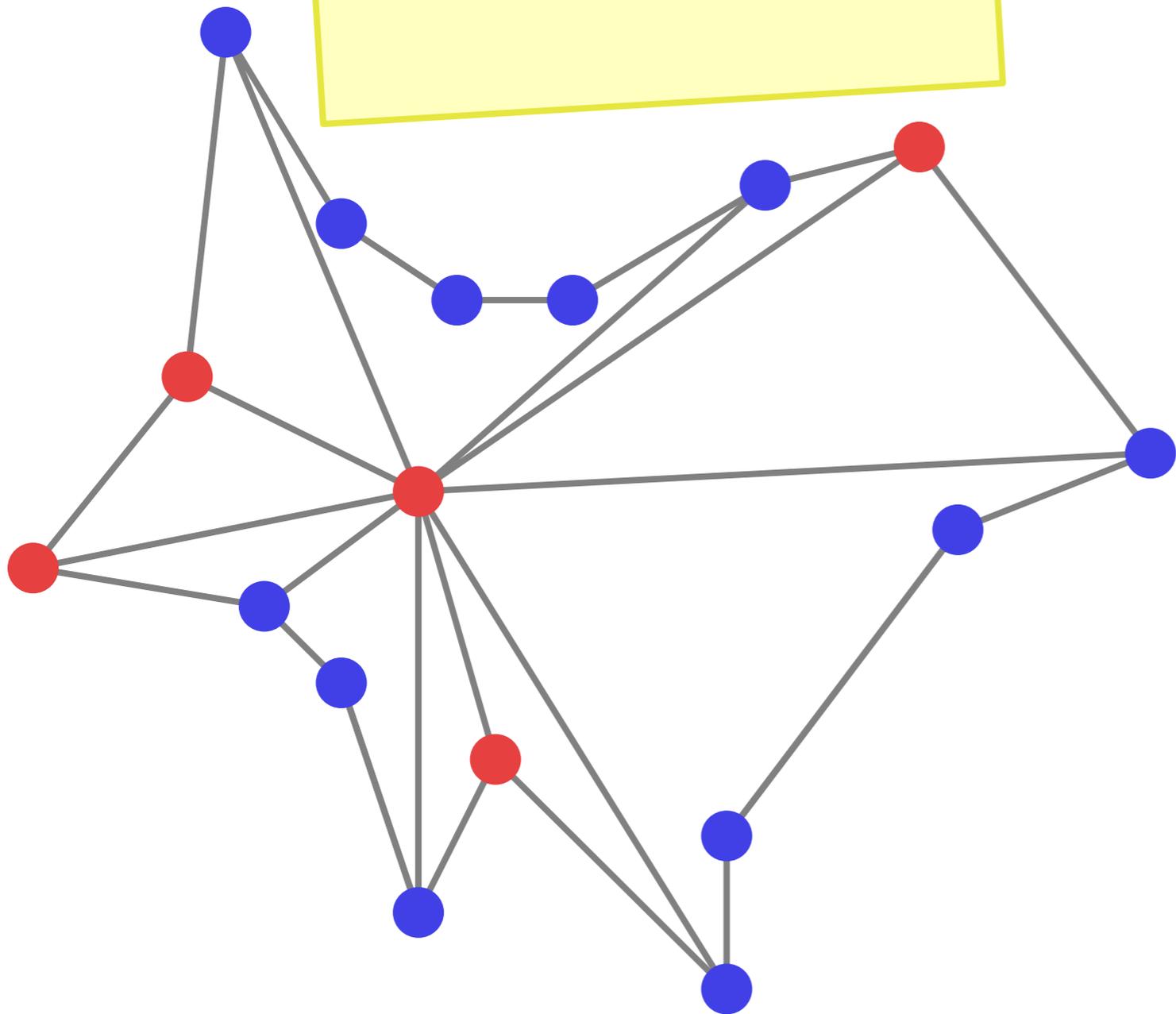
Finally, triangulate the remaining blue regions.



We also need a way to remove a constant degree red point.

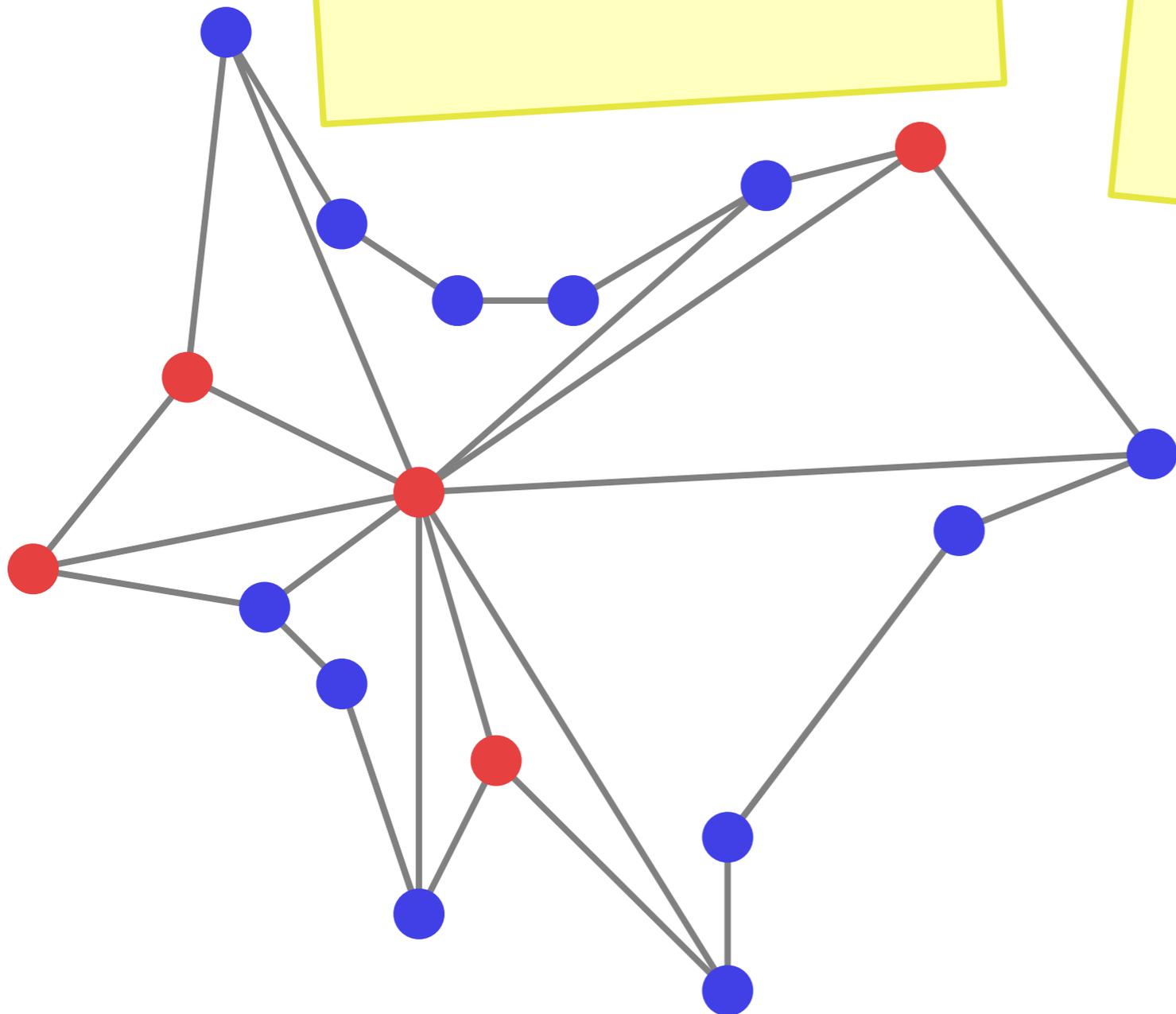
We can remove
a constant
degree red
point in
 $O(\log b)$ time.

We can remove
a constant
degree red
point in
 $O(\log b)$ time.



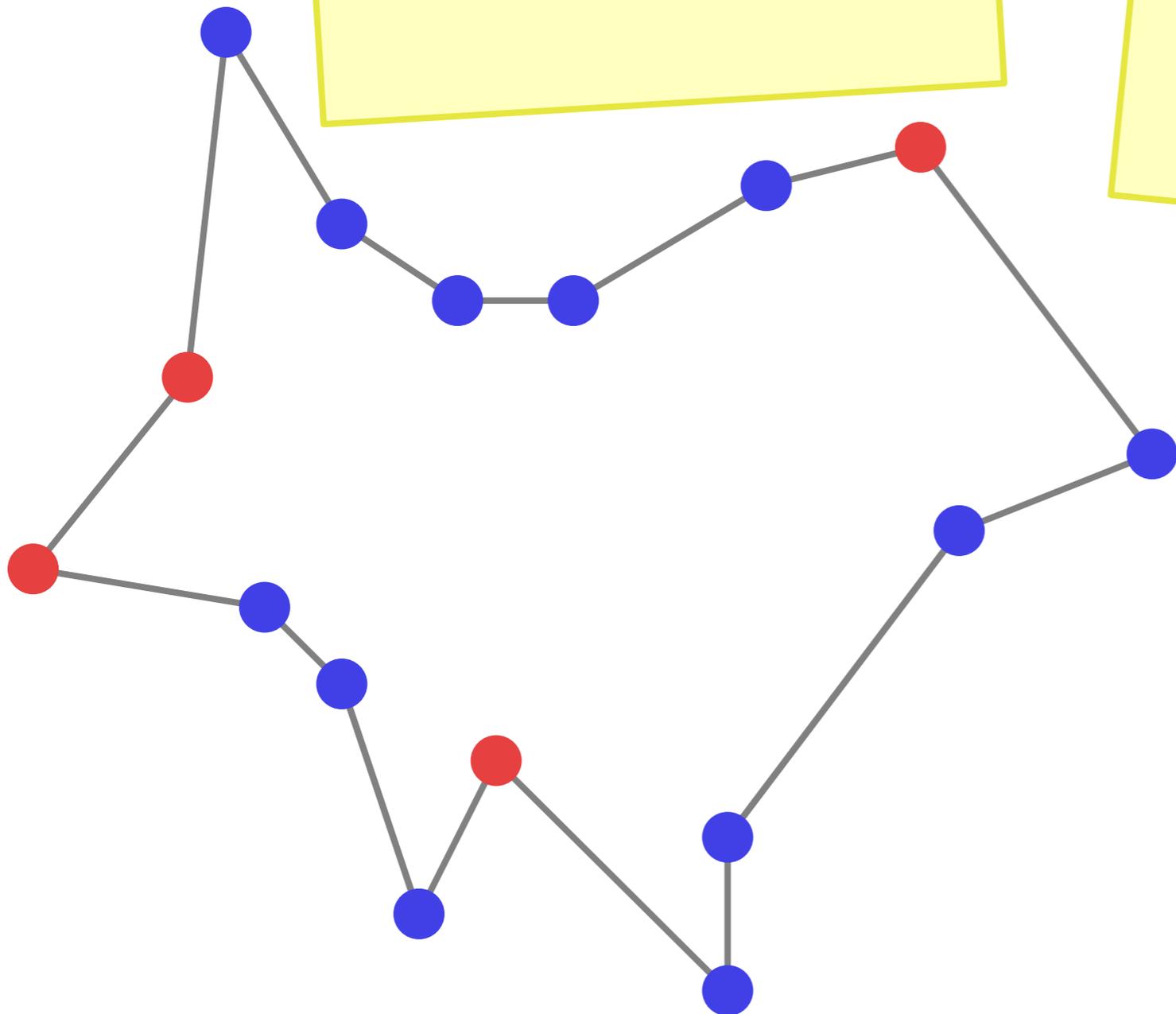
We can remove
a constant
degree red
point in
 $O(\log b)$ time.

Remove the
red point and
its $O(1)$ edges.



We can remove
a constant
degree red
point in
 $O(\log b)$ time.

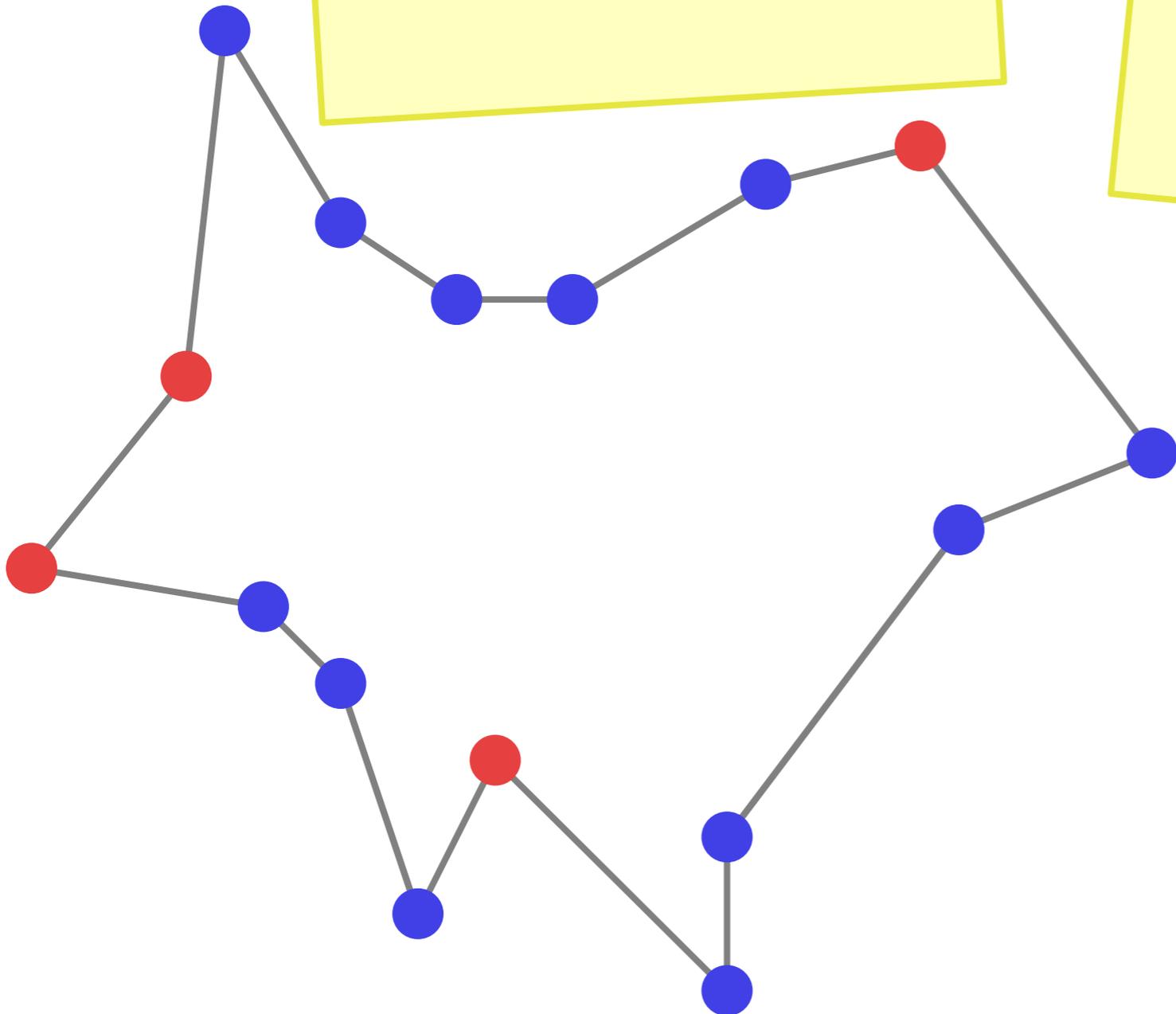
Remove the
red point and
its $O(1)$ edges.



We can remove a constant degree red point in $O(\log b)$ time.

Remove the red point and its $O(1)$ edges.

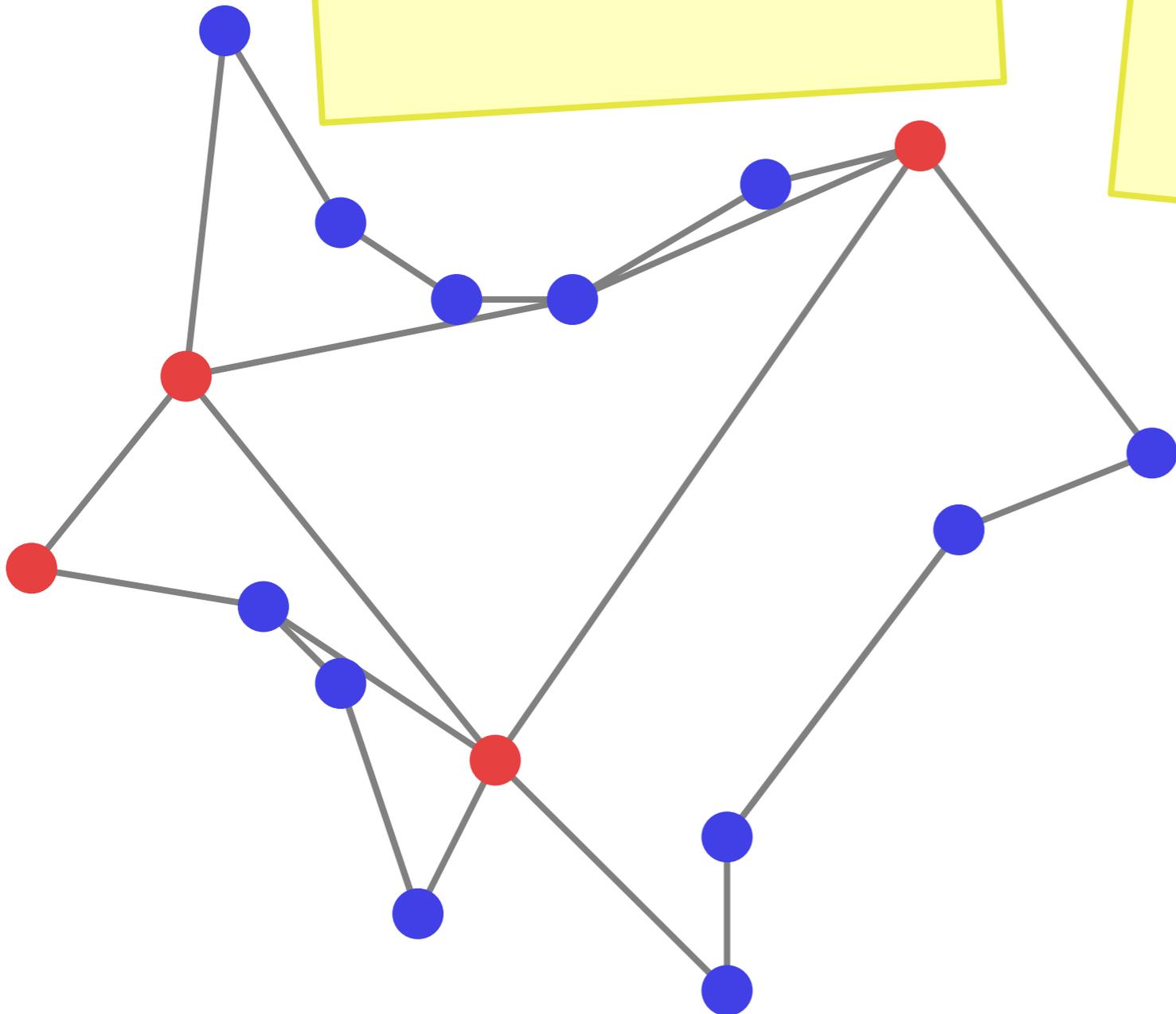
Compute shortest paths between the other red points.



We can remove a constant degree red point in $O(\log b)$ time.

Remove the red point and its $O(1)$ edges.

Compute shortest paths between the other red points.

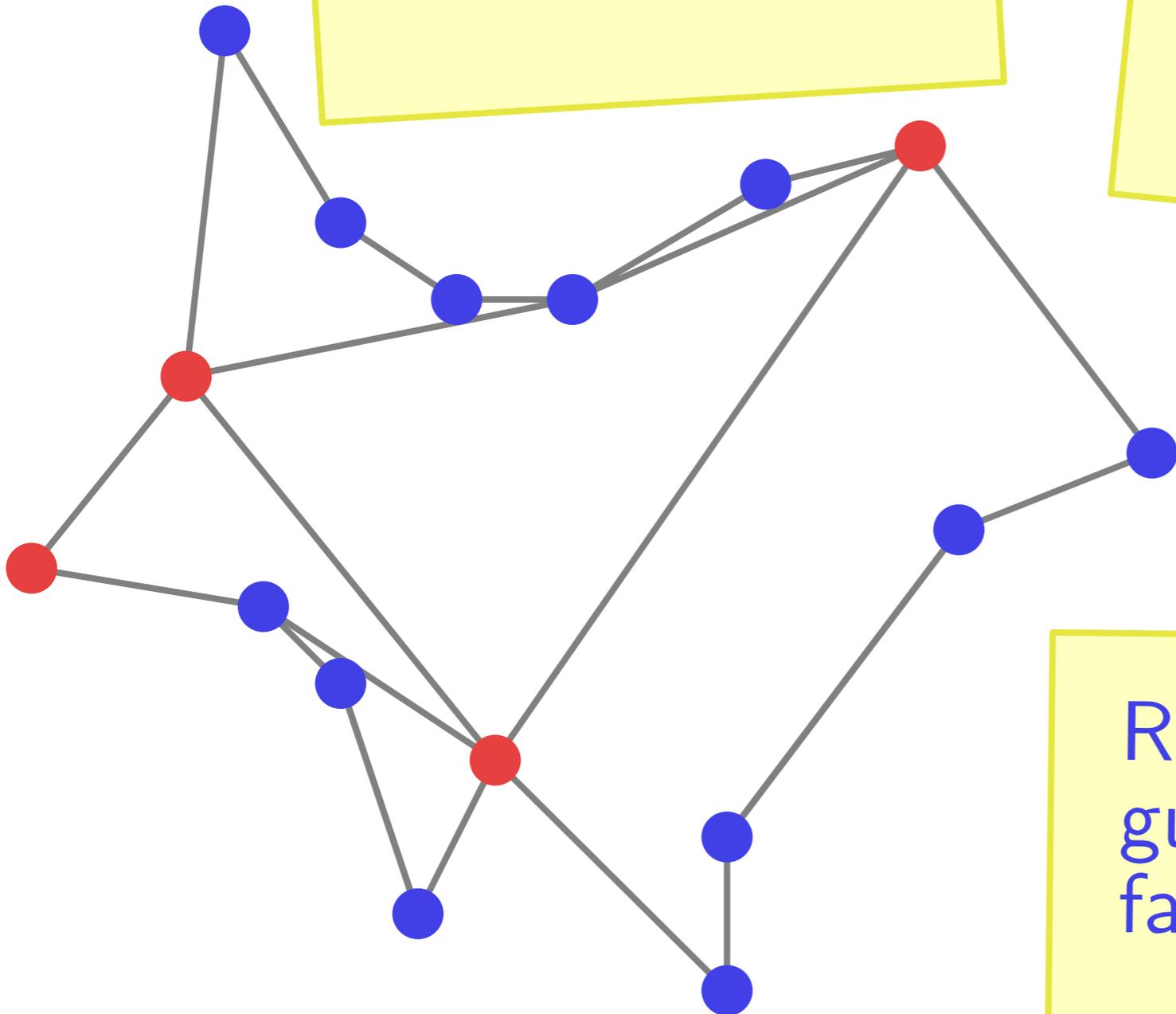


We can remove a constant degree red point in $O(\log b)$ time.

Remove the red point and its $O(1)$ edges.

Compute shortest paths between the other red points.

Repseudotriangulate the new faces.

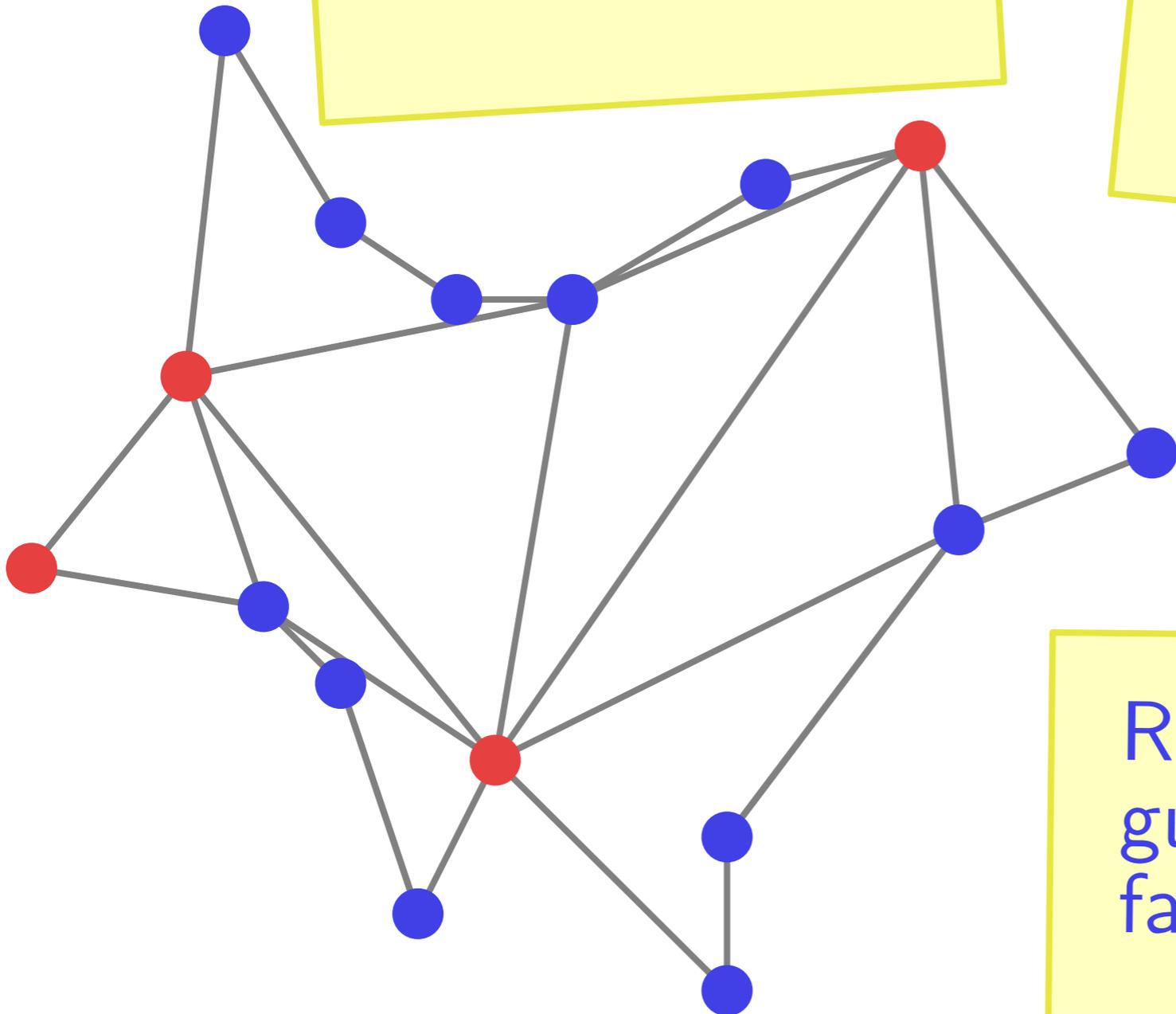


We can remove a constant degree red point in $O(\log b)$ time.

Remove the red point and its $O(1)$ edges.

Compute shortest paths between the other red points.

Repseudotriangulate the new faces.



What is the
total running
time?

We remove a
fraction of
 $1 - \frac{1}{f}$ red
points in each
phase.

We remove a fraction of $1 - \frac{1}{f}$ red points in each phase.

In the worst case, this takes $O(k \cdot \log \frac{n}{k})$ time if there are k red points left.

We remove a fraction of $1 - \frac{1}{f}$ red points in each phase.

In the worst case, this takes $O(k \cdot \log \frac{n}{k})$ time if there are k red points left.

There are $\log_f n$ phases, with $k = n/f^i$ red points remaining after the i th phase.

We remove a fraction of $1 - \frac{1}{f}$ red points in each phase.

In the worst case, this takes $O(k \cdot \log \frac{n}{k})$ time if there are k red points left.

There are $\log_f n$ phases, with $k = n/f^i$ red points remaining after the i th phase.

$$\sum_{i=1}^{\log_f n} O\left(\frac{n}{f^i} \log f^i\right)$$

We remove a fraction of $1 - \frac{1}{f}$ red points in each phase.

In the worst case, this takes $O(k \cdot \log \frac{n}{k})$ time if there are k red points left.

There are $\log_f n$ phases, with $k = n/f^i$ red points remaining after the i th phase.

$$\sum_{i=1}^{\log_f n} O\left(\frac{n}{f^i} \log f^i\right) = \dots$$

We remove a fraction of $1 - \frac{1}{f}$ red points in each phase.

In the worst case, this takes $O(k \cdot \log \frac{n}{k})$ time if there are k red points left.

There are $\log_f n$ phases, with $k = n/f^i$ red points remaining after the i th phase.

$$\sum_{i=1}^{\log_f n} O\left(\frac{n}{f^i} \log f^i\right) = O(n)$$

We remove a fraction of $1 - \frac{1}{f}$ red points in each phase.

In the worst case, this takes $O(k \cdot \log \frac{n}{k})$ time if there are k red points left.

There are $\log_f n$ phases, with $k = n/f^i$ red points remaining after the i th phase.

$$\sum_{i=1}^{\log_f n} O\left(\frac{n}{f^i} \log f^i\right) = O(n)$$

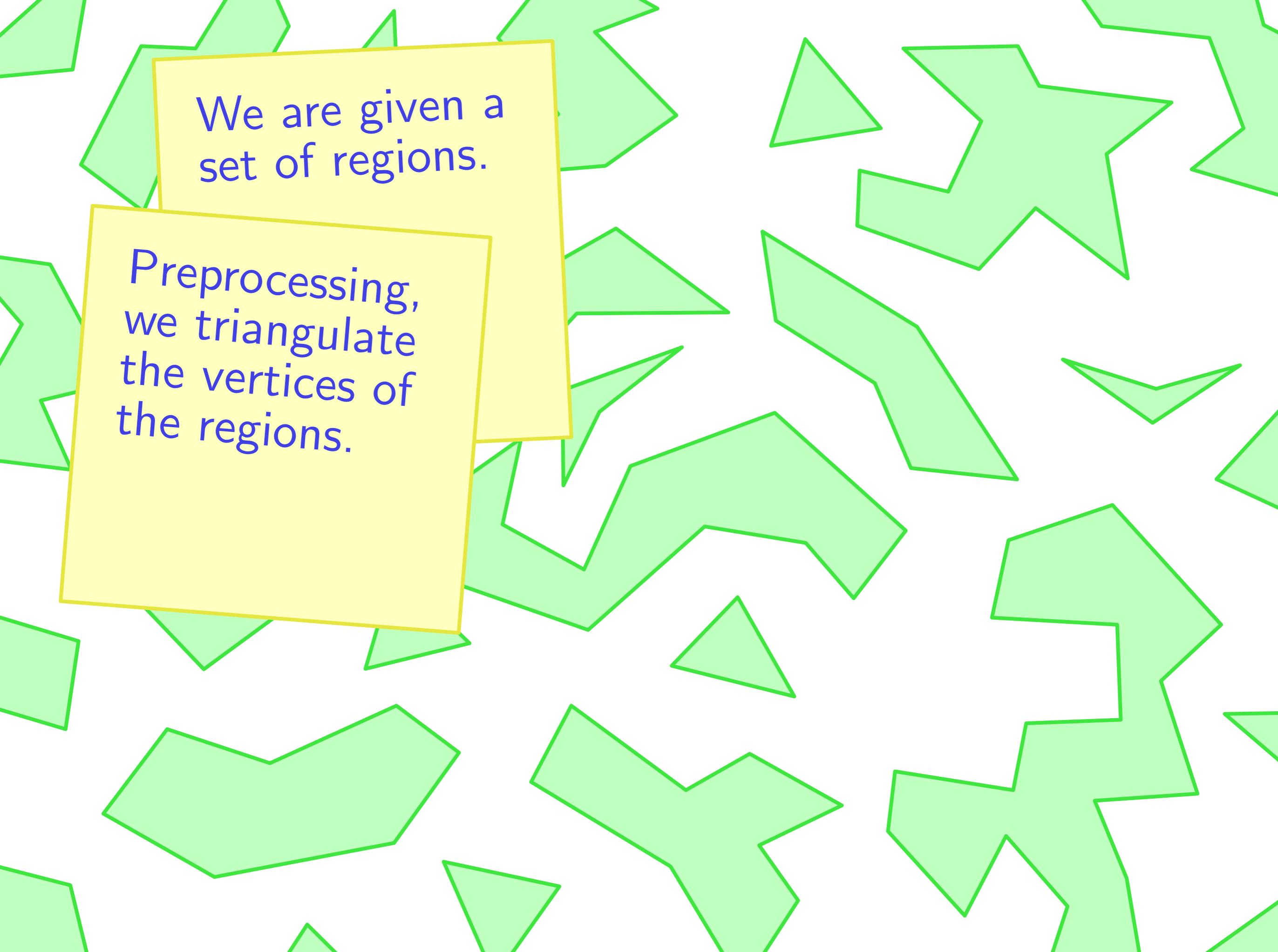
Problem solved!

Now, we are
ready to return
to the original
problem.

We are given a set of regions.

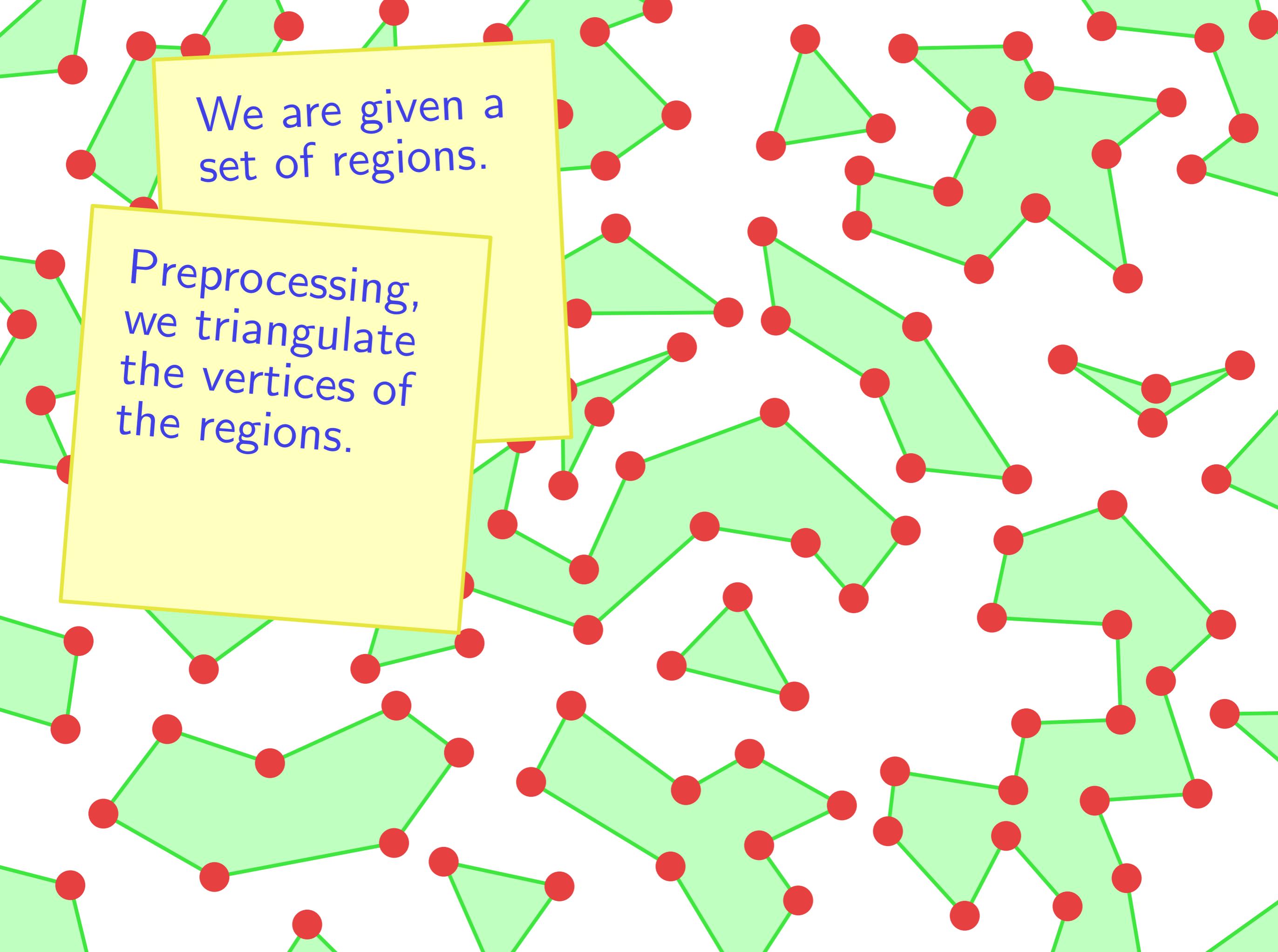


We are given a set of regions.

The background of the slide is filled with various light green polygons of different shapes and sizes, including triangles, quadrilaterals, and more complex irregular shapes. These polygons are scattered across the white background.

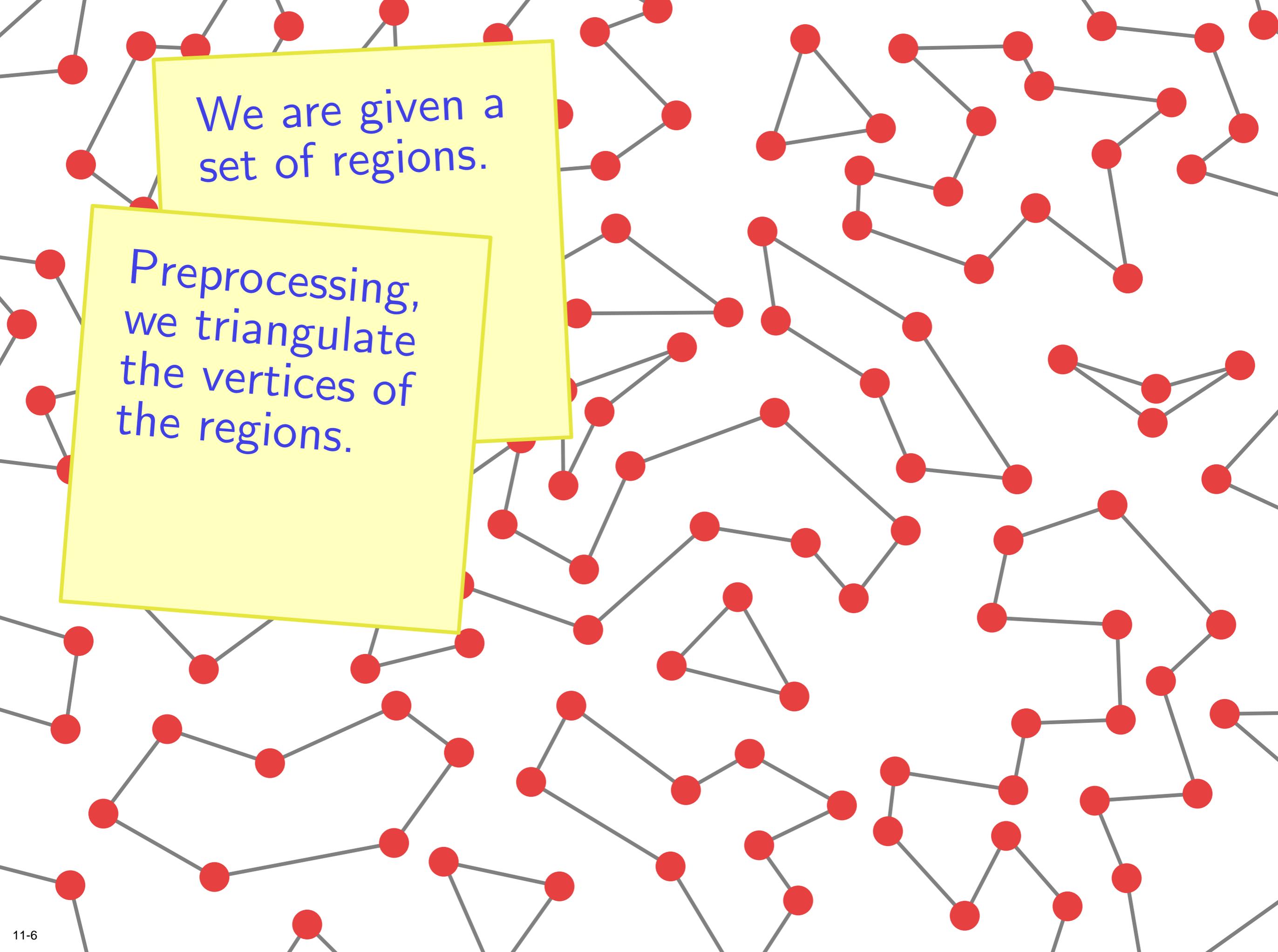
We are given a set of regions.

Preprocessing, we triangulate the vertices of the regions.



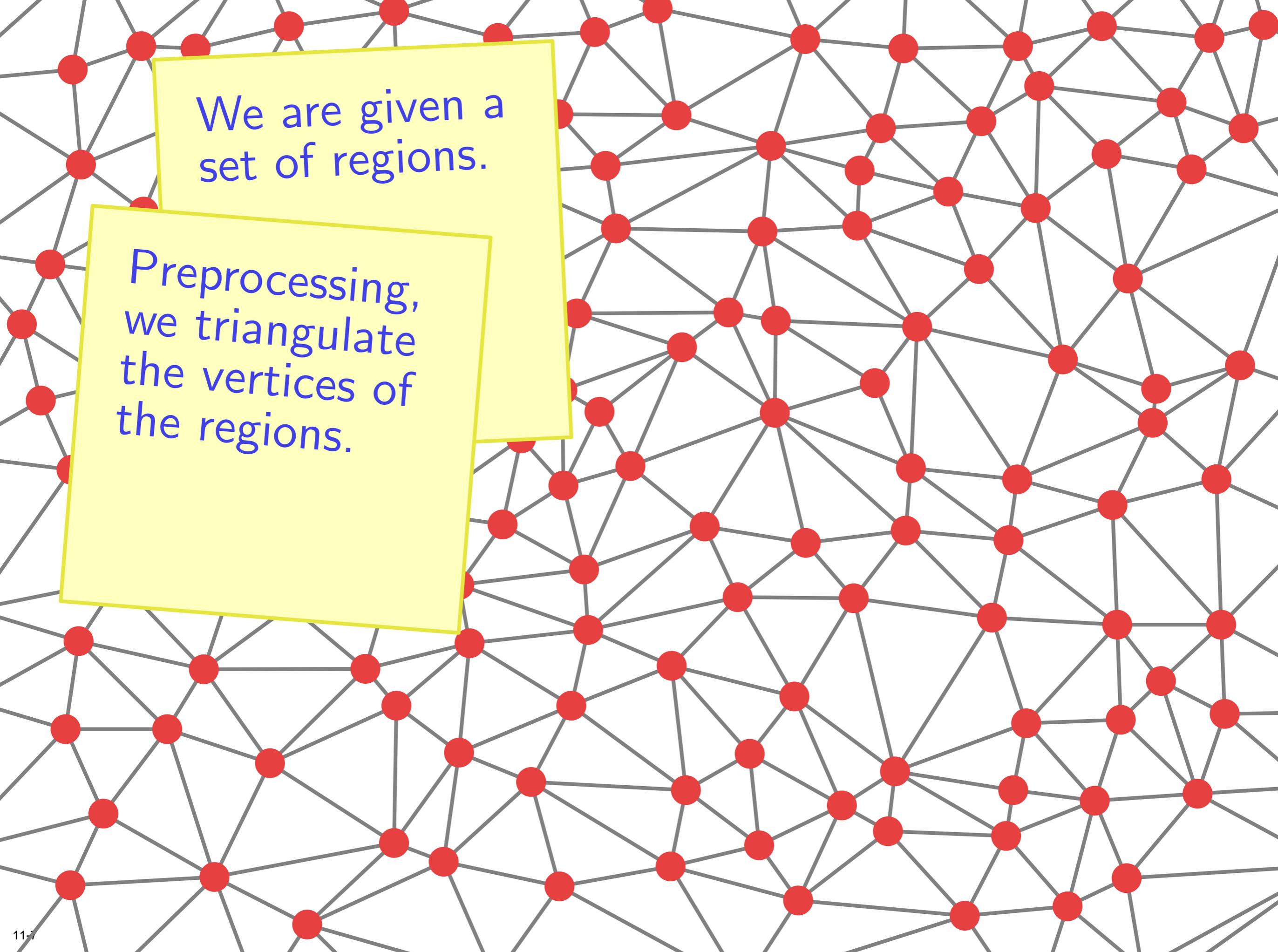
We are given a set of regions.

Preprocessing, we triangulate the vertices of the regions.



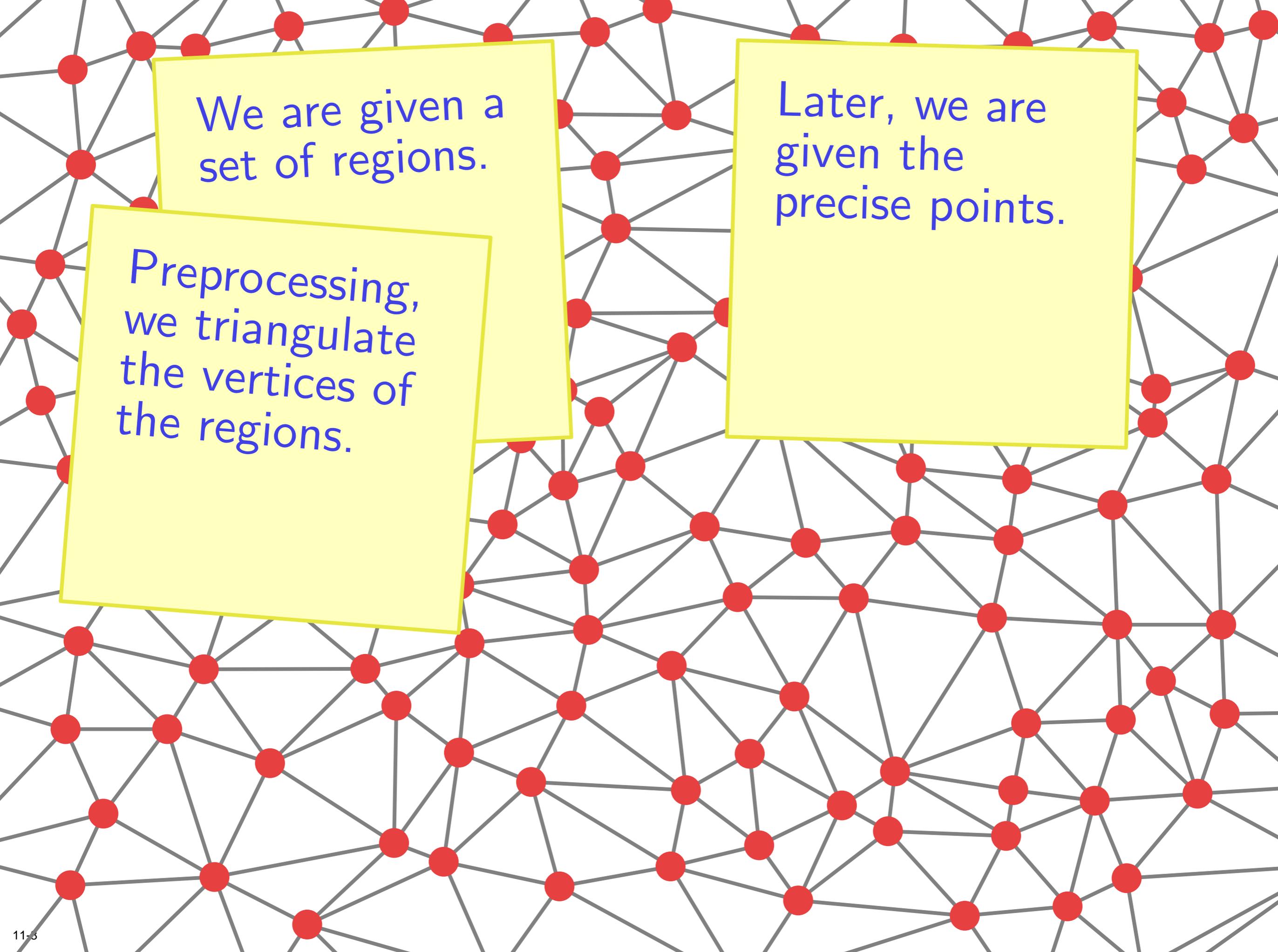
We are given a set of regions.

Preprocessing, we triangulate the vertices of the regions.



We are given a set of regions.

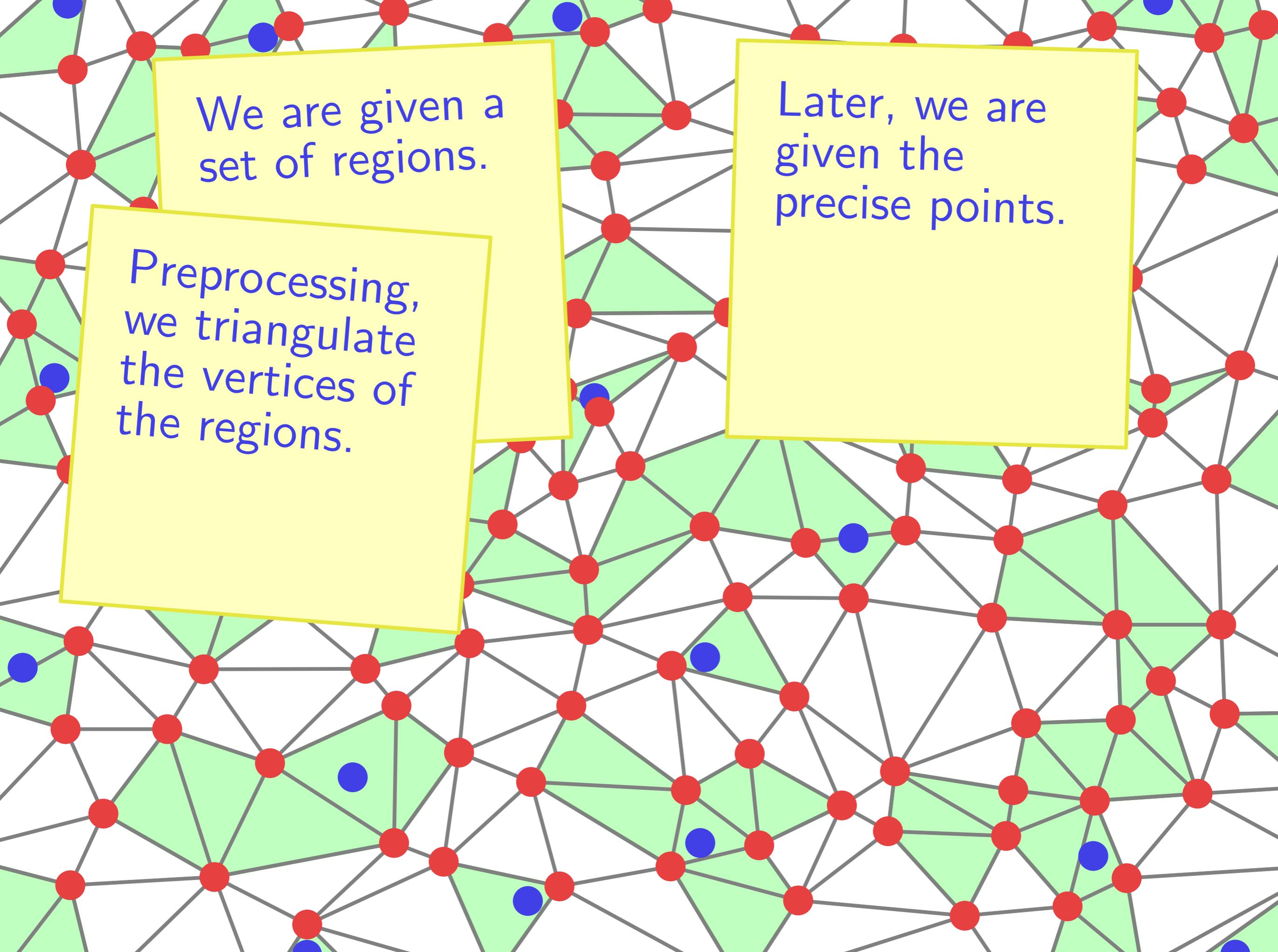
Preprocessing, we triangulate the vertices of the regions.



We are given a set of regions.

Preprocessing, we triangulate the vertices of the regions.

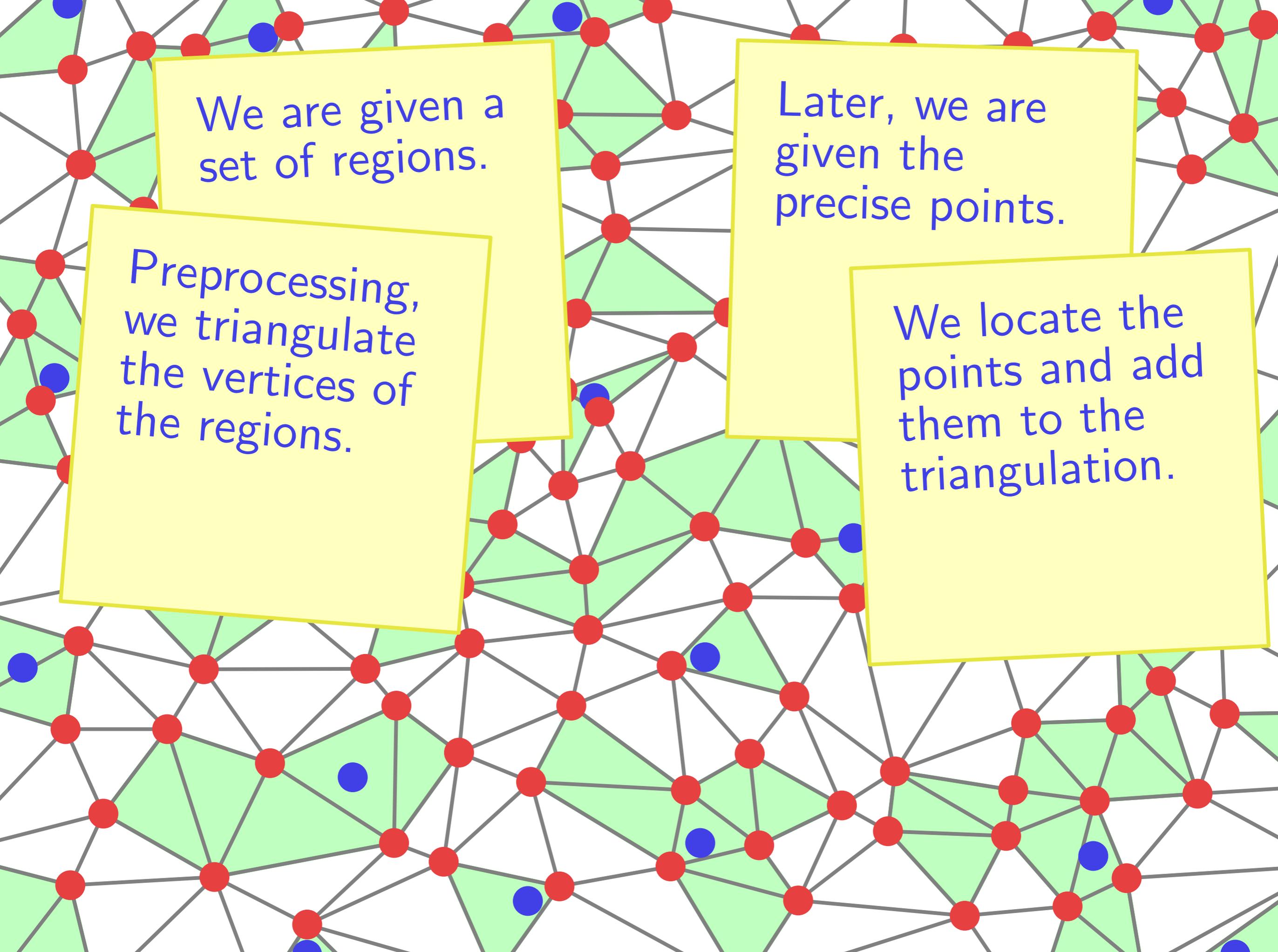
Later, we are given the precise points.



We are given a set of regions.

Preprocessing, we triangulate the vertices of the regions.

Later, we are given the precise points.

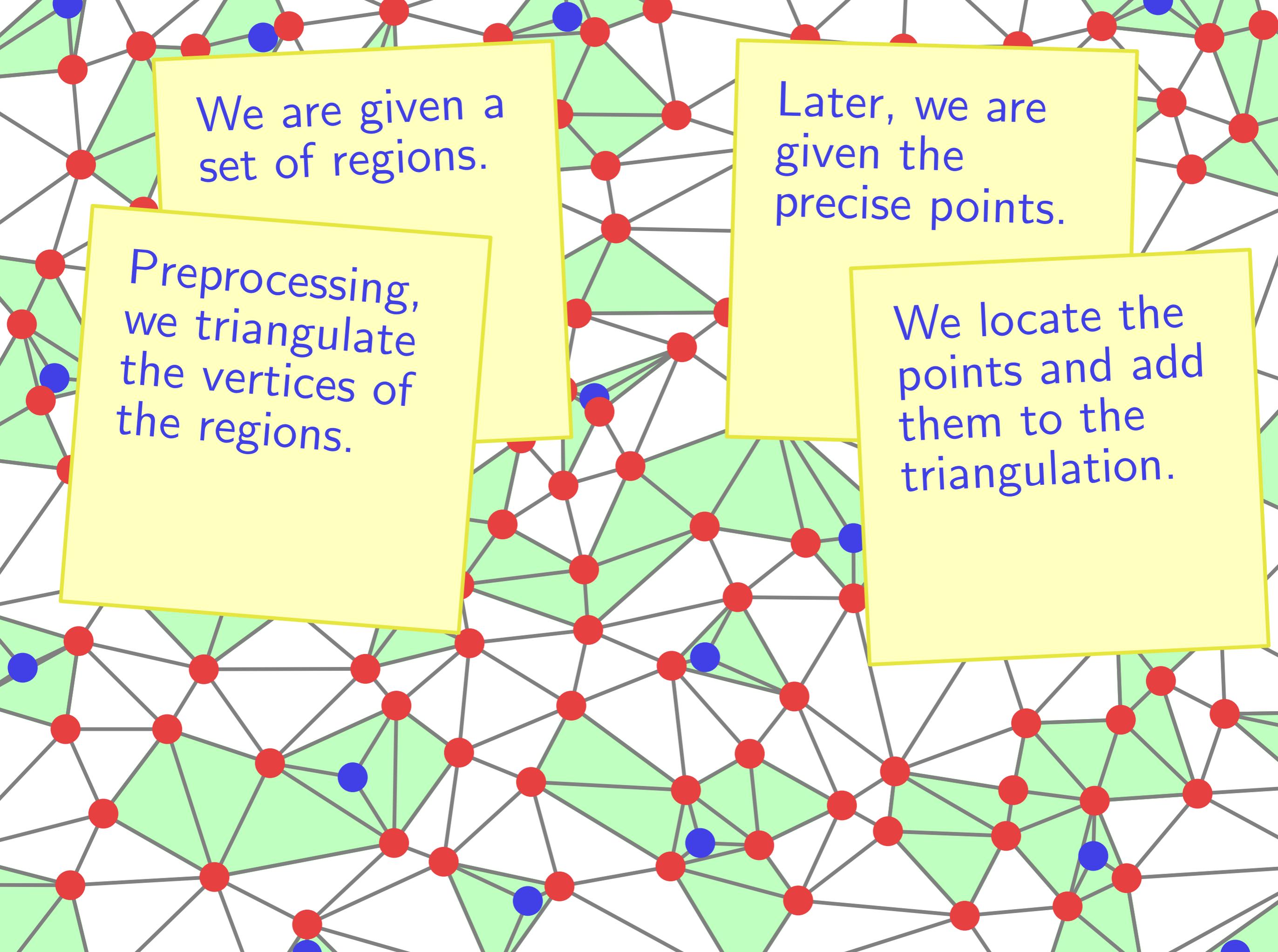


We are given a set of regions.

Preprocessing, we triangulate the vertices of the regions.

Later, we are given the precise points.

We locate the points and add them to the triangulation.

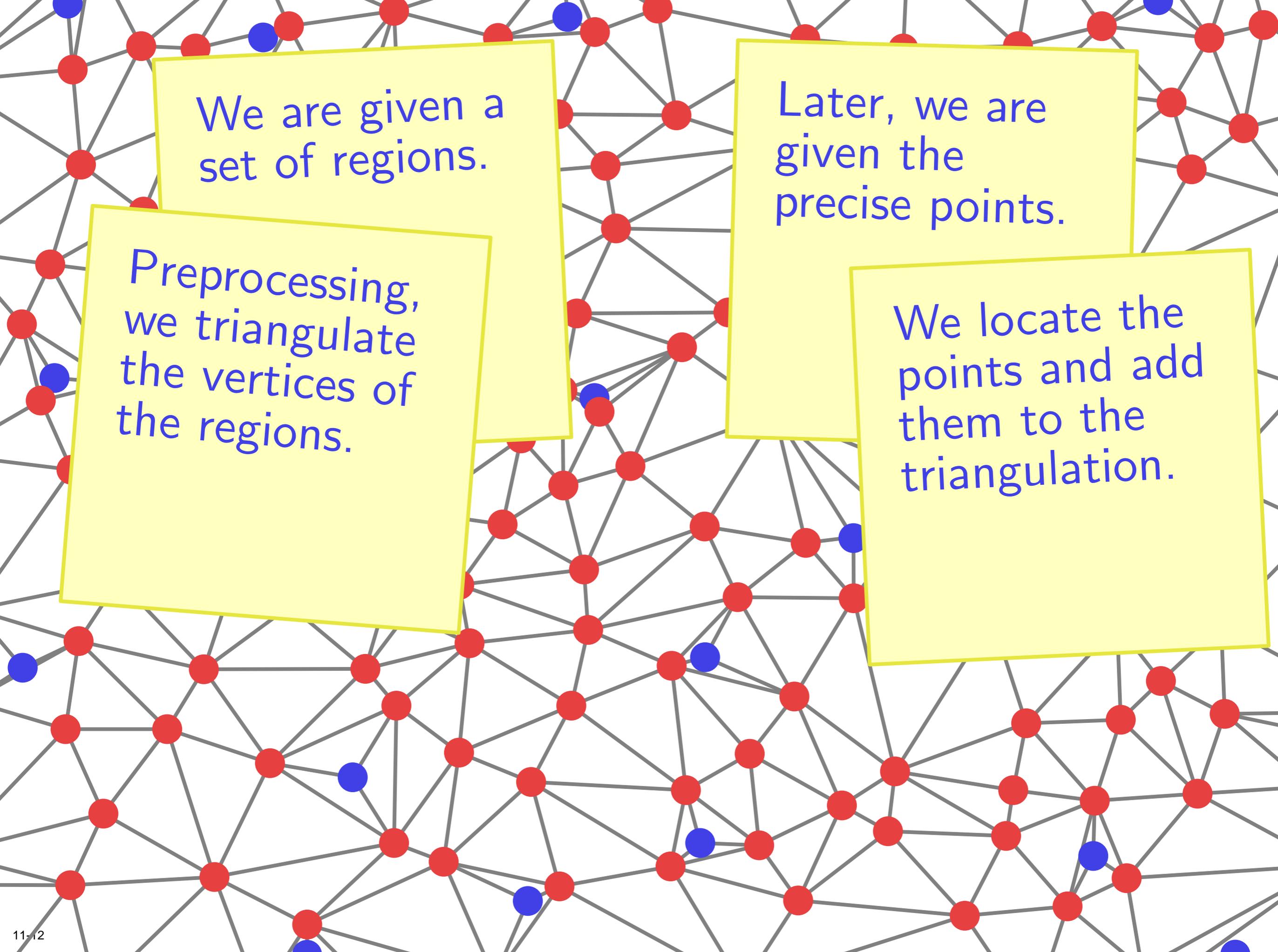


We are given a set of regions.

Preprocessing, we triangulate the vertices of the regions.

Later, we are given the precise points.

We locate the points and add them to the triangulation.

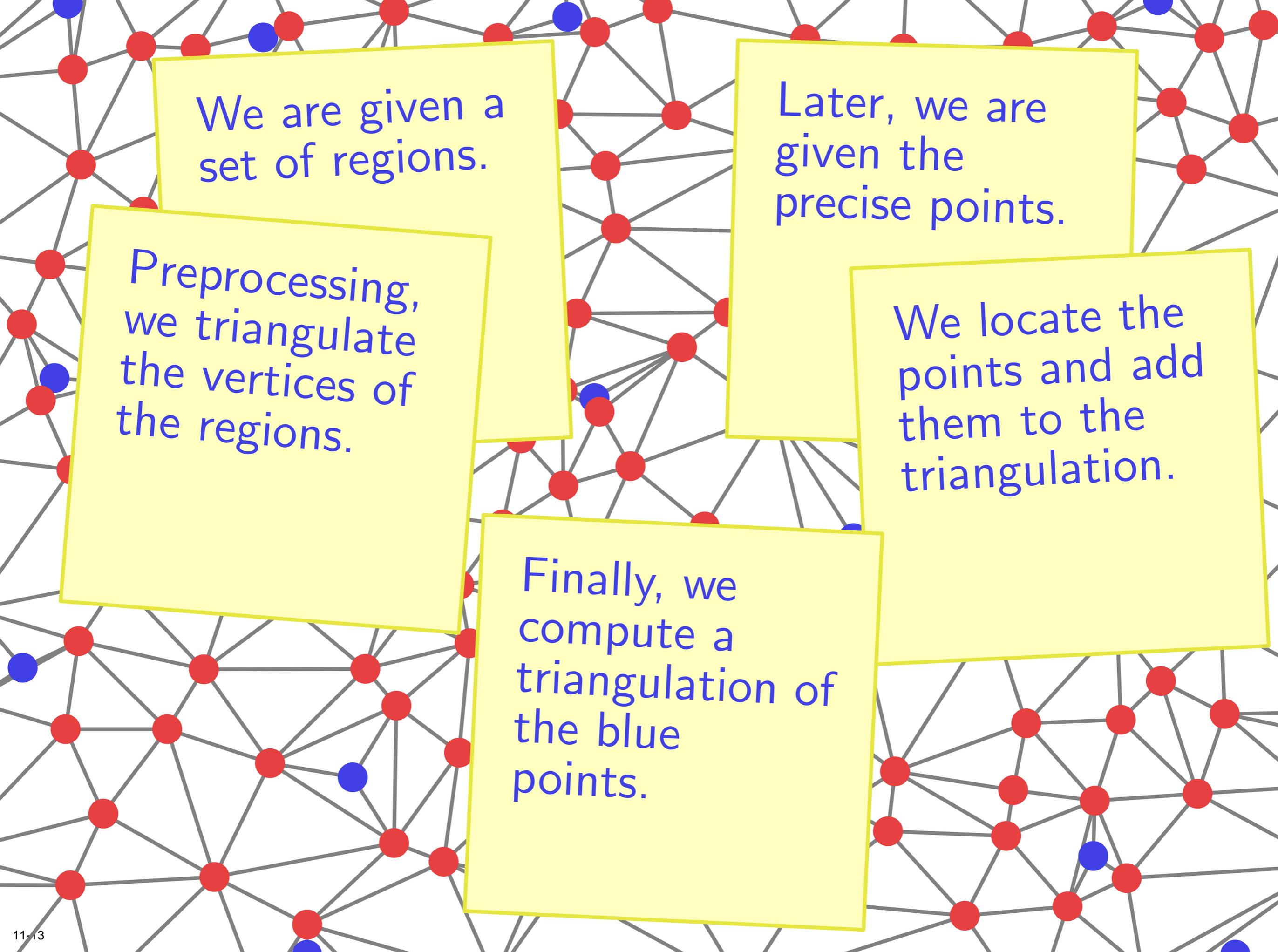


We are given a set of regions.

Preprocessing, we triangulate the vertices of the regions.

Later, we are given the precise points.

We locate the points and add them to the triangulation.

The background of the slide is a complex network of gray lines connecting red and blue circular points. The points are scattered across the frame, with a higher density of red points and a few blue points interspersed. The lines form a dense web of triangles of various sizes and orientations.

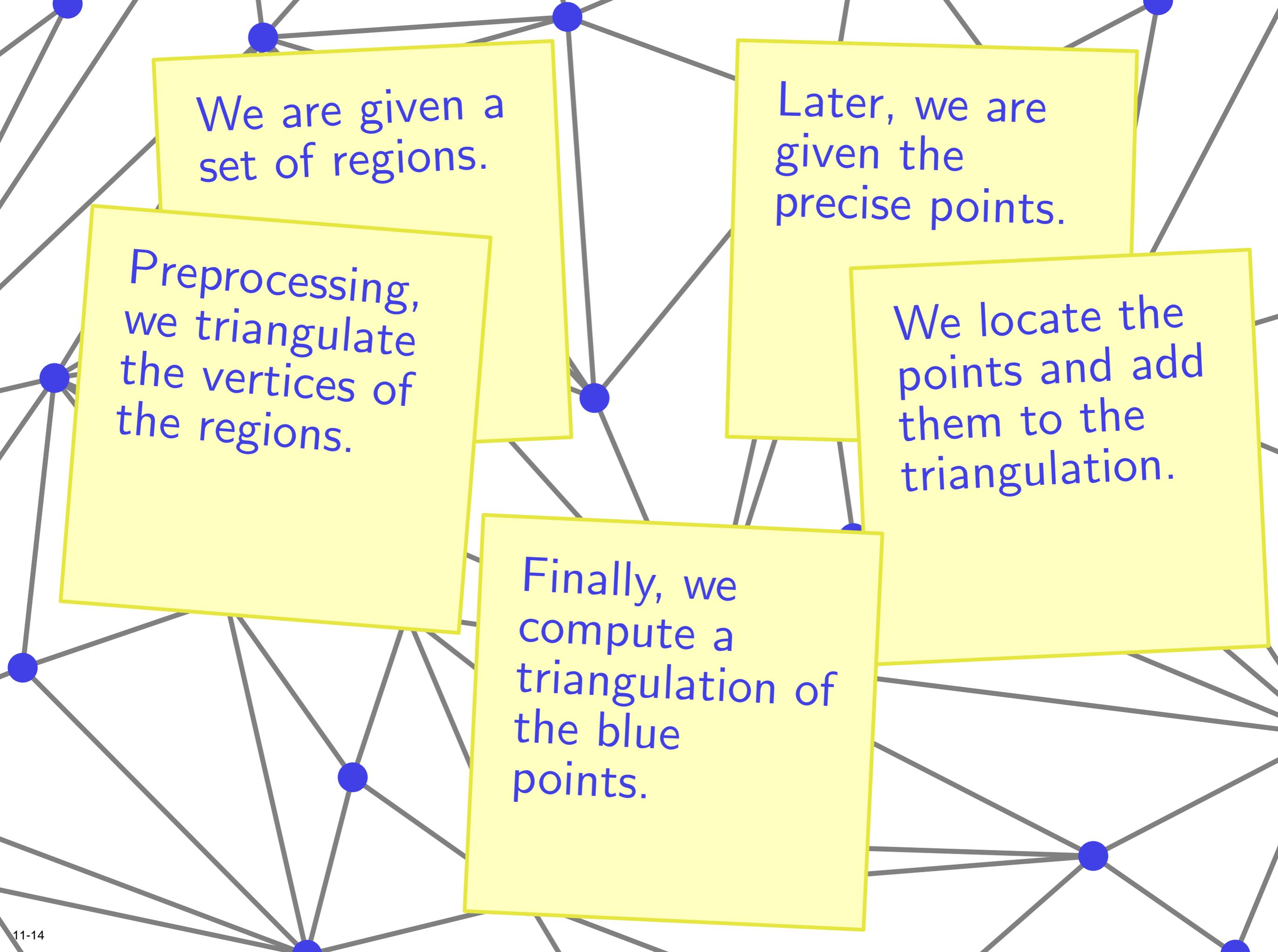
We are given a set of regions.

Preprocessing, we triangulate the vertices of the regions.

Later, we are given the precise points.

We locate the points and add them to the triangulation.

Finally, we compute a triangulation of the blue points.



We are given a set of regions.

Preprocessing, we triangulate the vertices of the regions.

Later, we are given the precise points.

We locate the points and add them to the triangulation.

Finally, we compute a triangulation of the blue points.

Time to
conclude.

We preprocess
a set of regions
in the plane in
 $O(n \log n)$
time.

We preprocess a set of regions in the plane in $O(n \log n)$ time.

Given a point inside each region, we compute a triangulation in $O(n)$ time.

We preprocess a set of regions in the plane in $O(n \log n)$ time.

Given a point inside each region, we compute a triangulation in $O(n)$ time.

Can be extended to partially overlapping regions.

We preprocess a set of regions in the plane in $O(n \log n)$ time.

Given a point inside each region, we compute a triangulation in $O(n)$ time.

Can be extended to partially overlapping regions.

OPEN PROBLEM
Preprocess a set of lines in the plane for linear time triangulation.

We preprocess a set of regions in the plane in $O(n \log n)$ time.

Given a point inside each region, we compute a triangulation in $O(n)$ time.

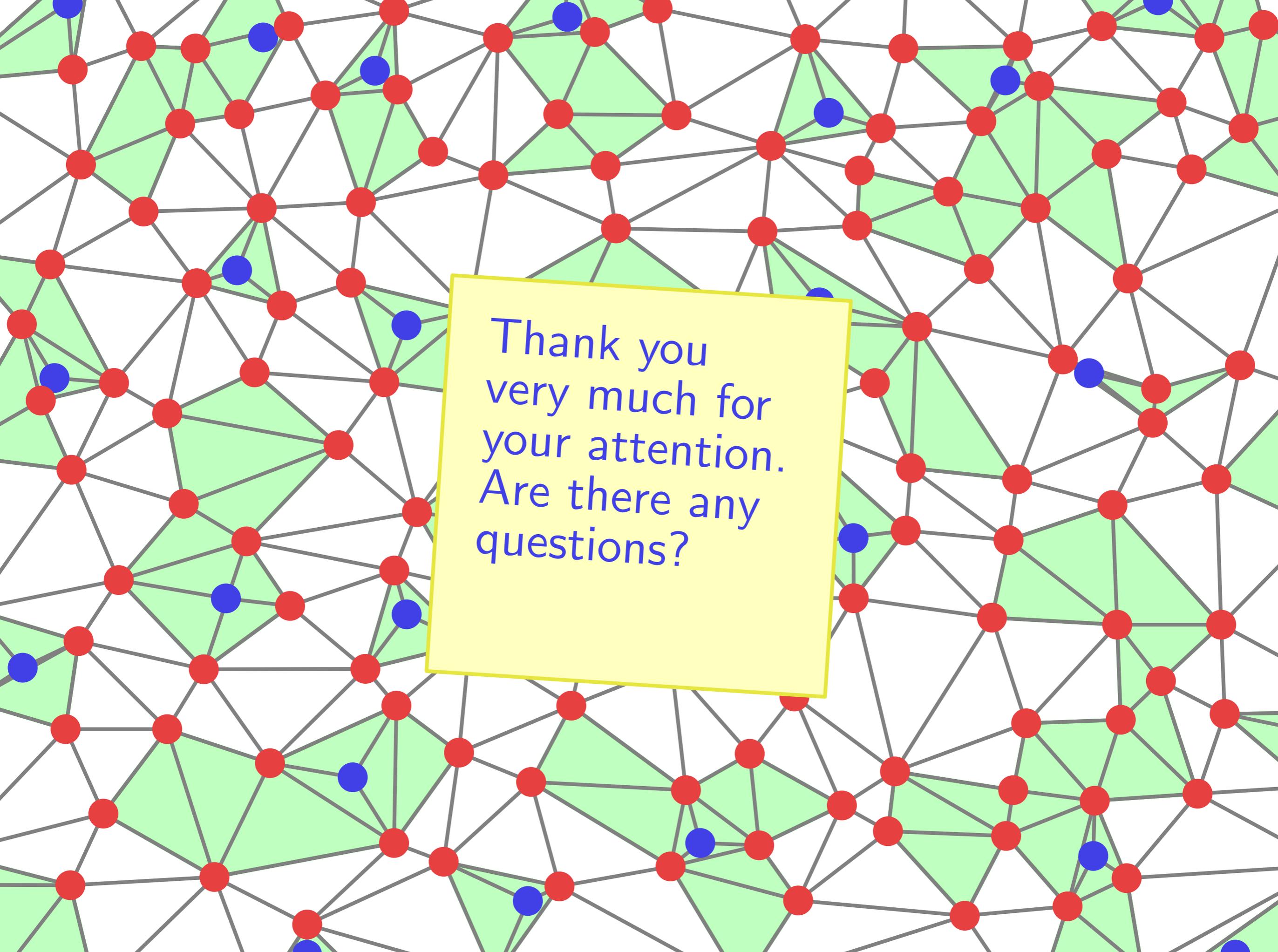
Can be extended to partially overlapping regions.

OPEN PROBLEM

Preprocess a set of lines in the plane for linear time triangulation.

OPEN PROBLEM

Can similar results be obtained in higher dimensions?

A network diagram consisting of a dense web of gray lines connecting nodes. Most nodes are red, but several are blue. The background is filled with light green shaded regions of various shapes, creating a complex, interconnected pattern.

Thank you
very much for
your attention.
Are there any
questions?