UNIONS OF ONIONS

Maarten Löffler
Universiteit Utrecht

Wolfgang Mulzer
Freie Universität Berlin

CHAPTER I

THE PROBLEM

PREPROCESSING PARADIGM

PREPROCESSING PARADIGM

Once upon a time, we were given $\mathcal{R}$.

PREPROCESSING PARADIGM

Once upon a time, we were given $\mathcal{R}$.

PREPROCESSING PARADIGM

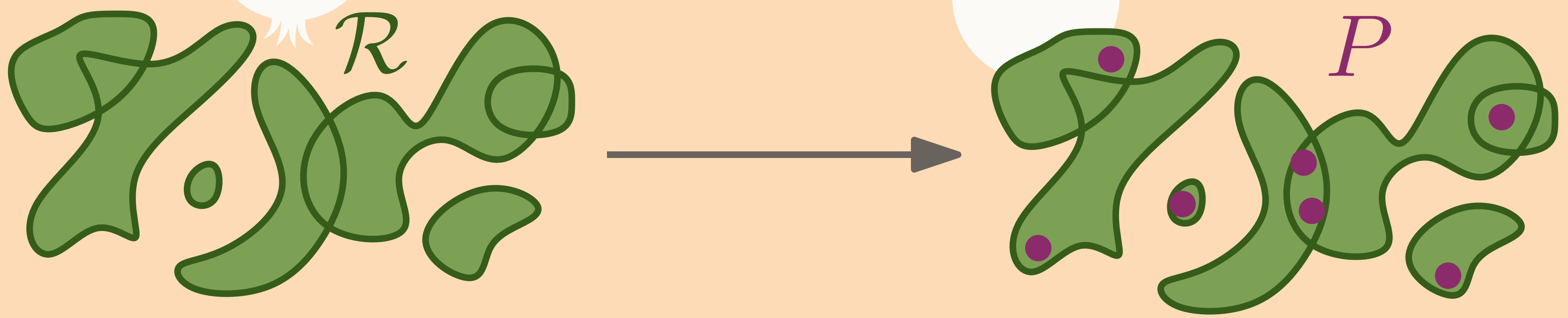
Once upon a time, we were given $\mathcal{R}$.

Soon, we get P: one point per region of $\mathcal{R}$.

PREPROCESSING PARADIGM

Once upon a time, we were given $\mathcal{R}$.

Soon, we get P: one point per region of $\mathcal{R}$.

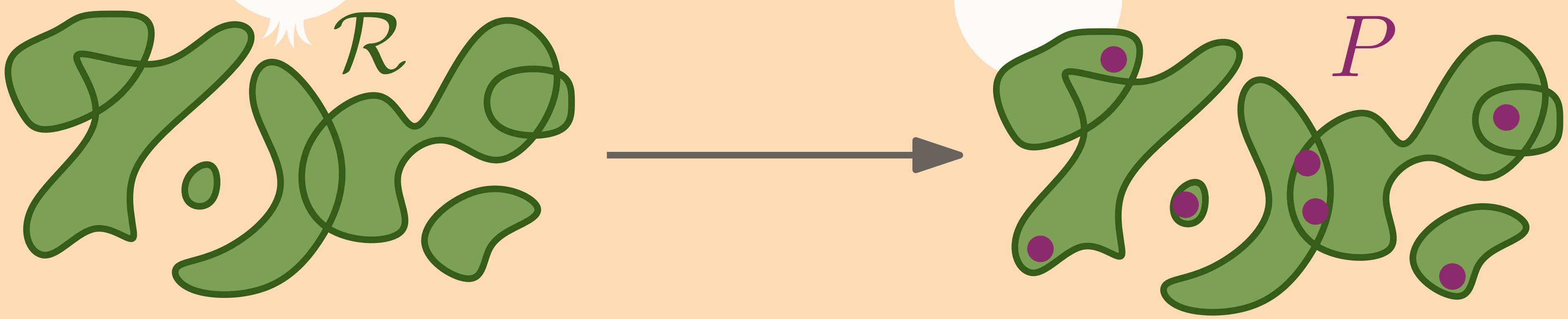


PREPROCESSING PARADIGM

Once upon a time, we were given $\mathcal{R}$.

Soon, we get P: one point per region of $\mathcal{R}$.

What we really want is some structure $S(P)$.

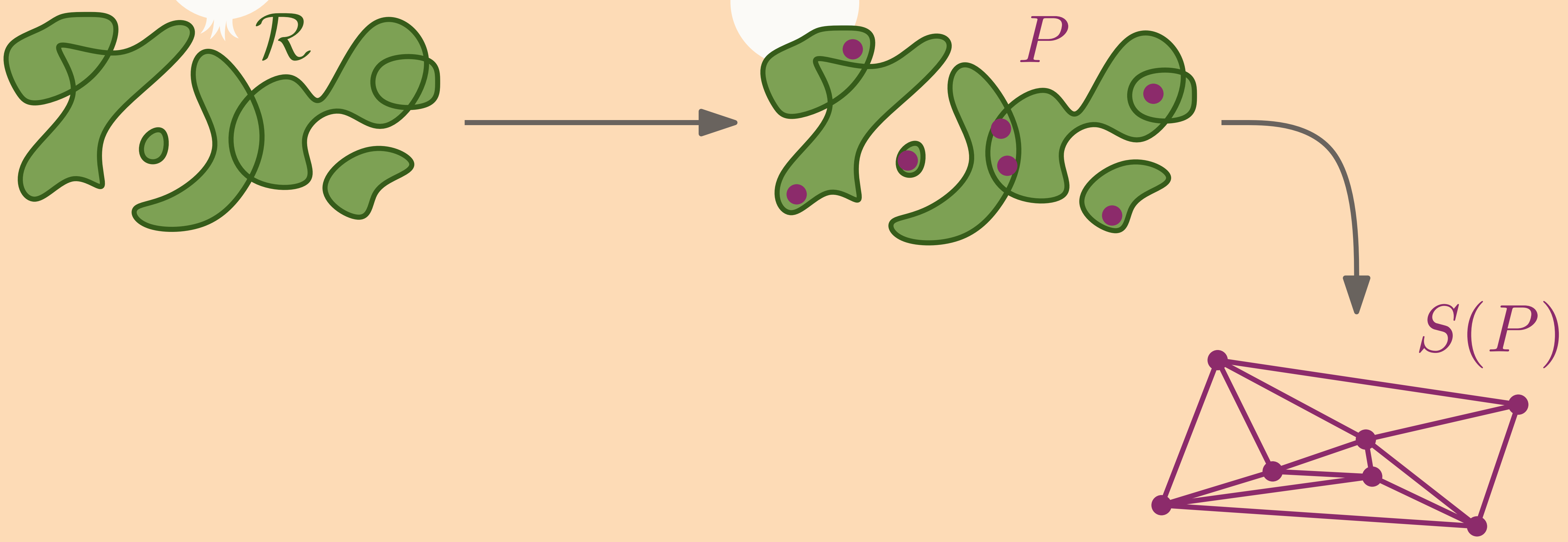


PREPROCESSING PARADIGM

Once upon a time, we were given $\mathcal{R}$.

Soon, we get P: one point per region of $\mathcal{R}$.

What we really want is some structure $S(P)$.



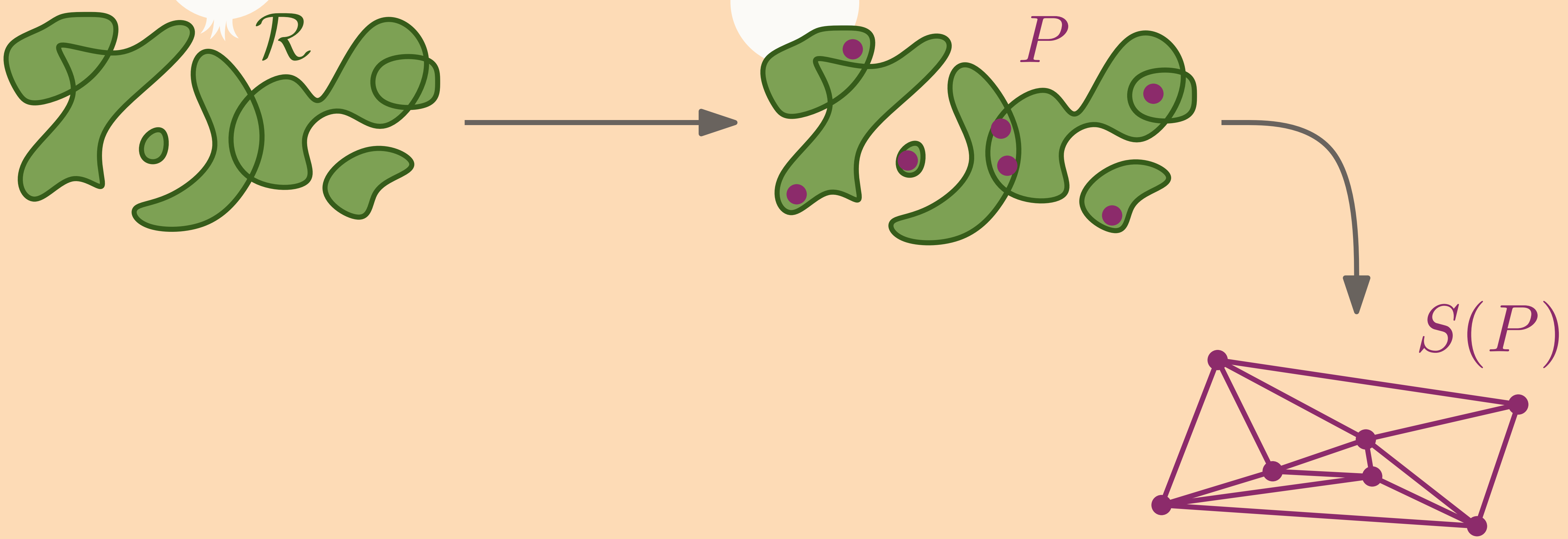
PREPROCESSING PARADIGM

Once upon a time, we were given $\mathcal{R}$.

Soon, we get P: one point per region of $\mathcal{R}$.

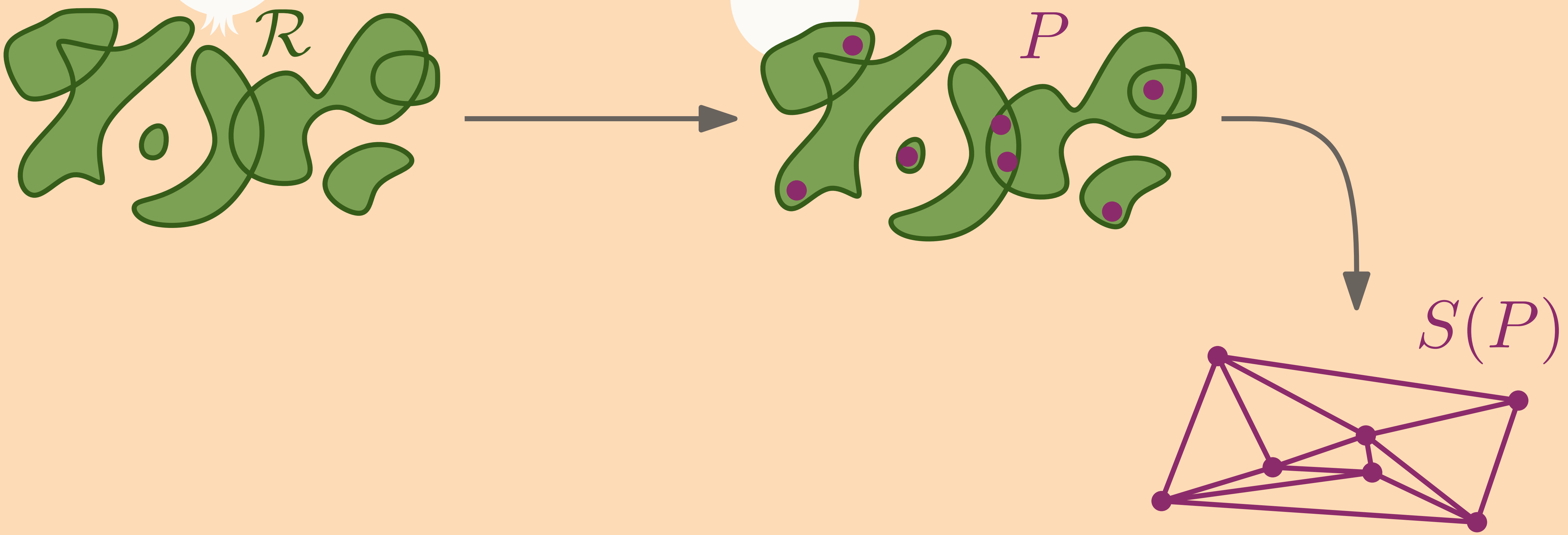
What we really want is some structure $S(P)$.

But time is precious!



PREPROCESSING PARADIGM

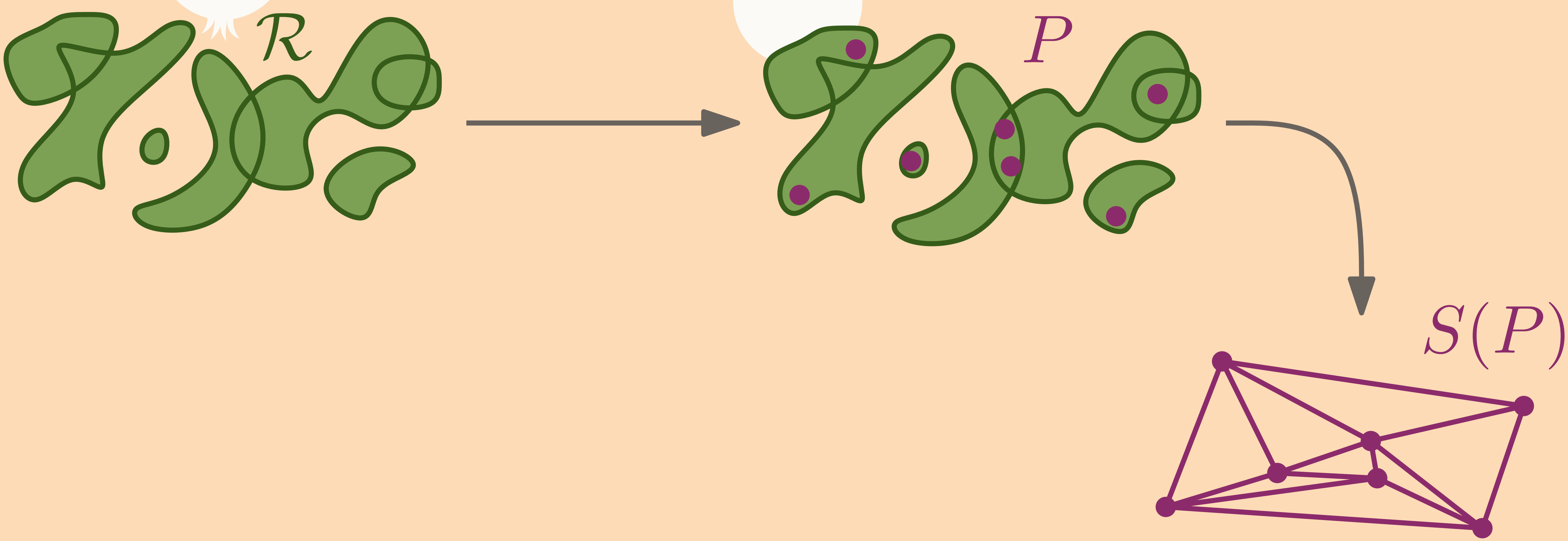
QUESTION: *Can we do some of the computation of $S(P)$ before we know P?*



PREPROCESSING PARADIGM

QUESTION: *Can we do some of the computation of $S(P)$ before we know P?*

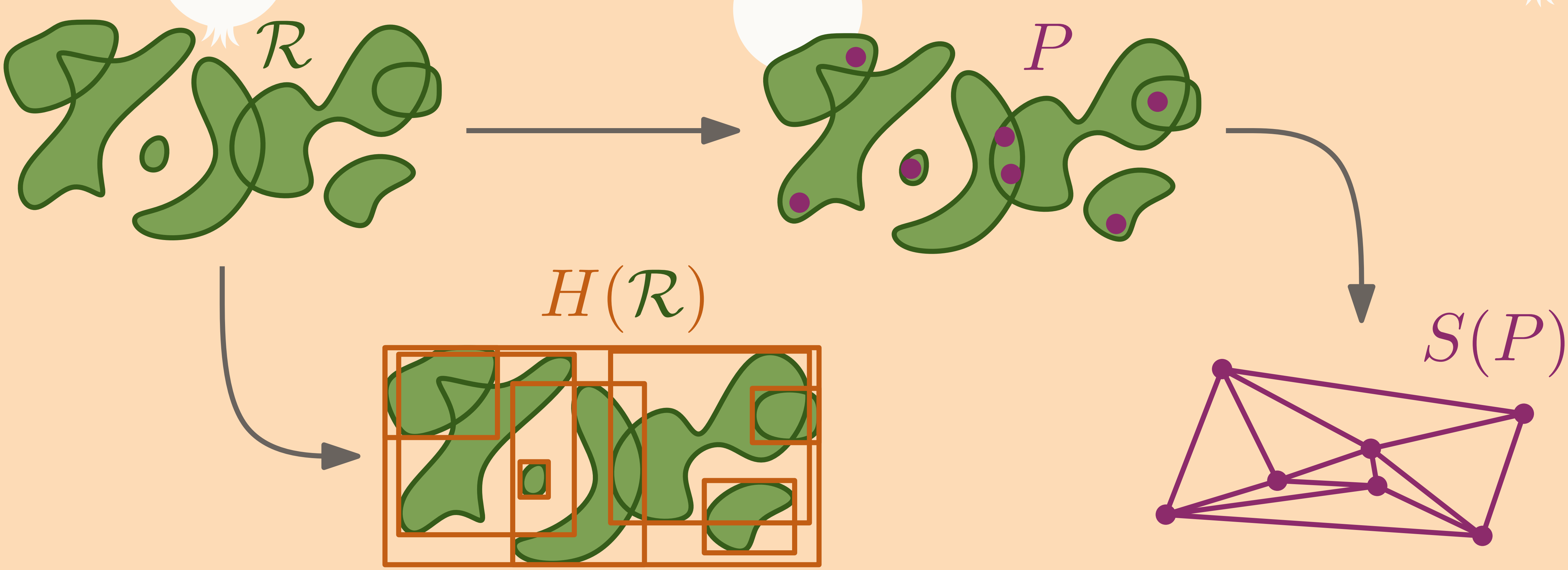
First, compute an intermediary structure $H(\mathcal{R})$.



PREPROCESSING PARADIGM

QUESTION: *Can we do some of the computation of $S(P)$ before we know P?*

First, compute an intermediary structure $H(\mathcal{R})$.

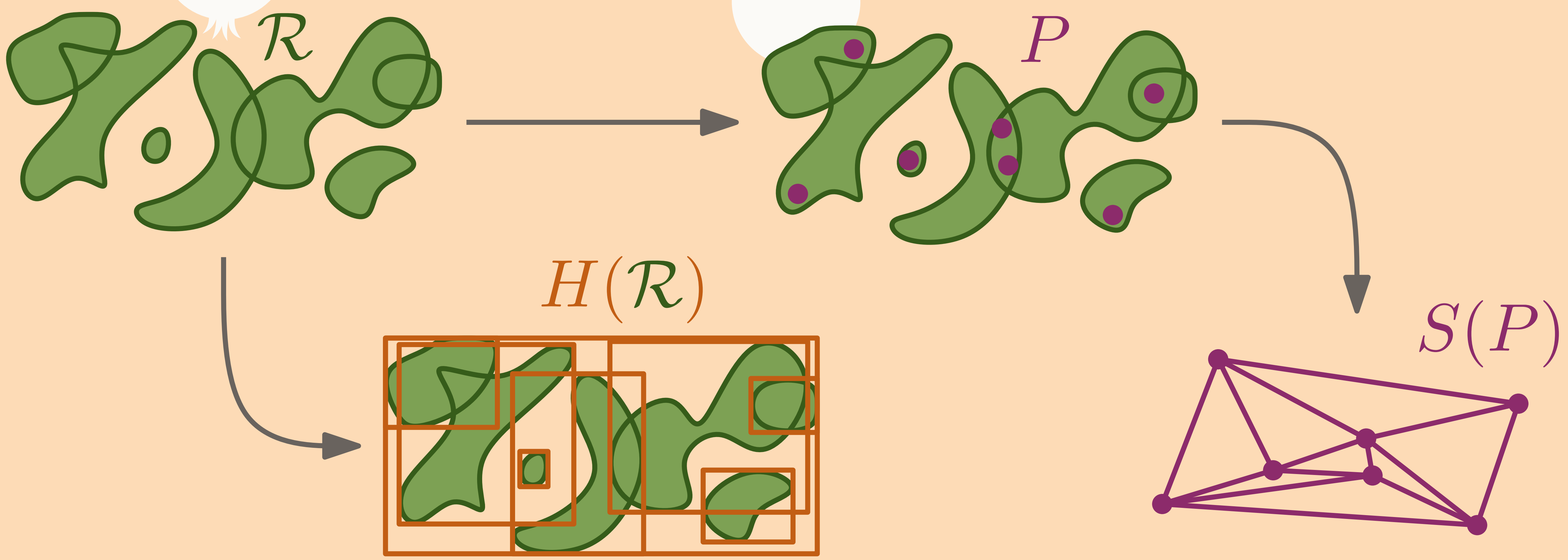


PREPROCESSING PARADIGM

QUESTION: *Can we do some of the computation of $S(P)$ before we know P?*

First, compute an intermediary structure $H(\mathcal{R})$.

Then, once we get P, compute $S(P)$ using $H(\mathcal{R})$.

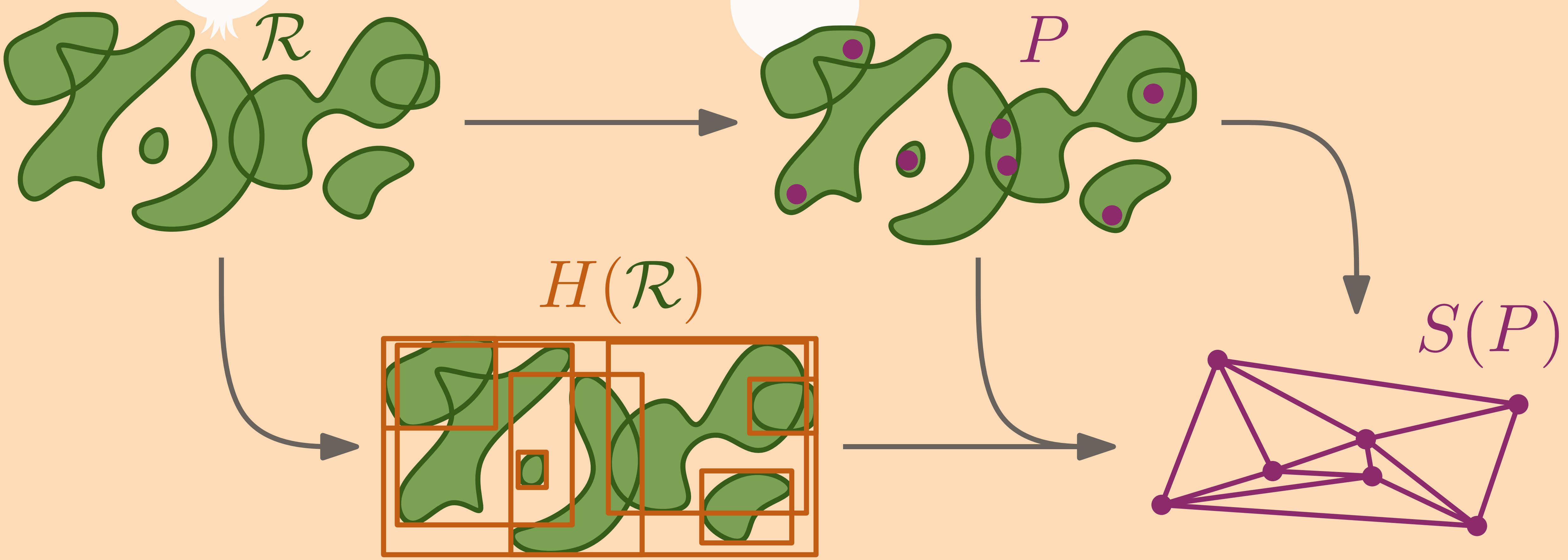


PREPROCESSING PARADIGM

QUESTION: *Can we do some of the computation of $S(P)$ before we know P?*

First, compute an intermediary structure $H(\mathcal{R})$.

Then, once we get P, compute $S(P)$ using $H(\mathcal{R})$.



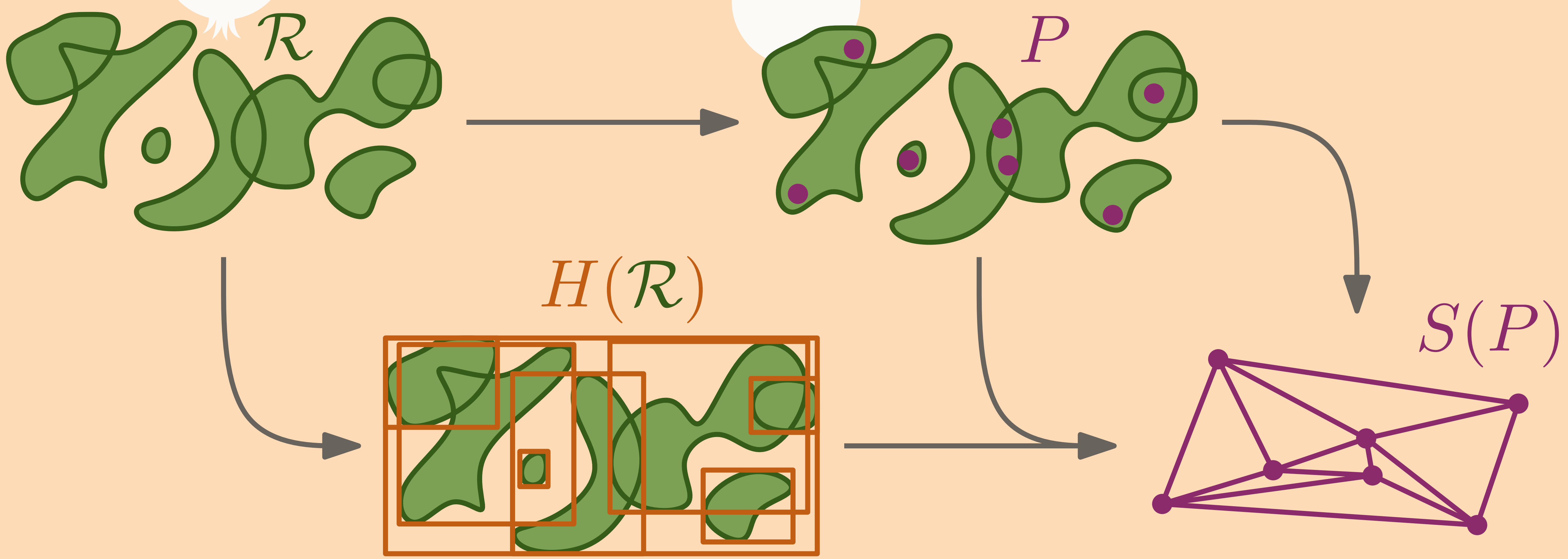
PREPROCESSING PARADIGM

Linear time algorithms are known for:

- Triangulations [Held & Mitchell]
- Delaunay triangulations [L & Snoeyink]

Superlinear, but $o(n \log n)$:

- Convex hull (lines) [Ezra & Mulzer]



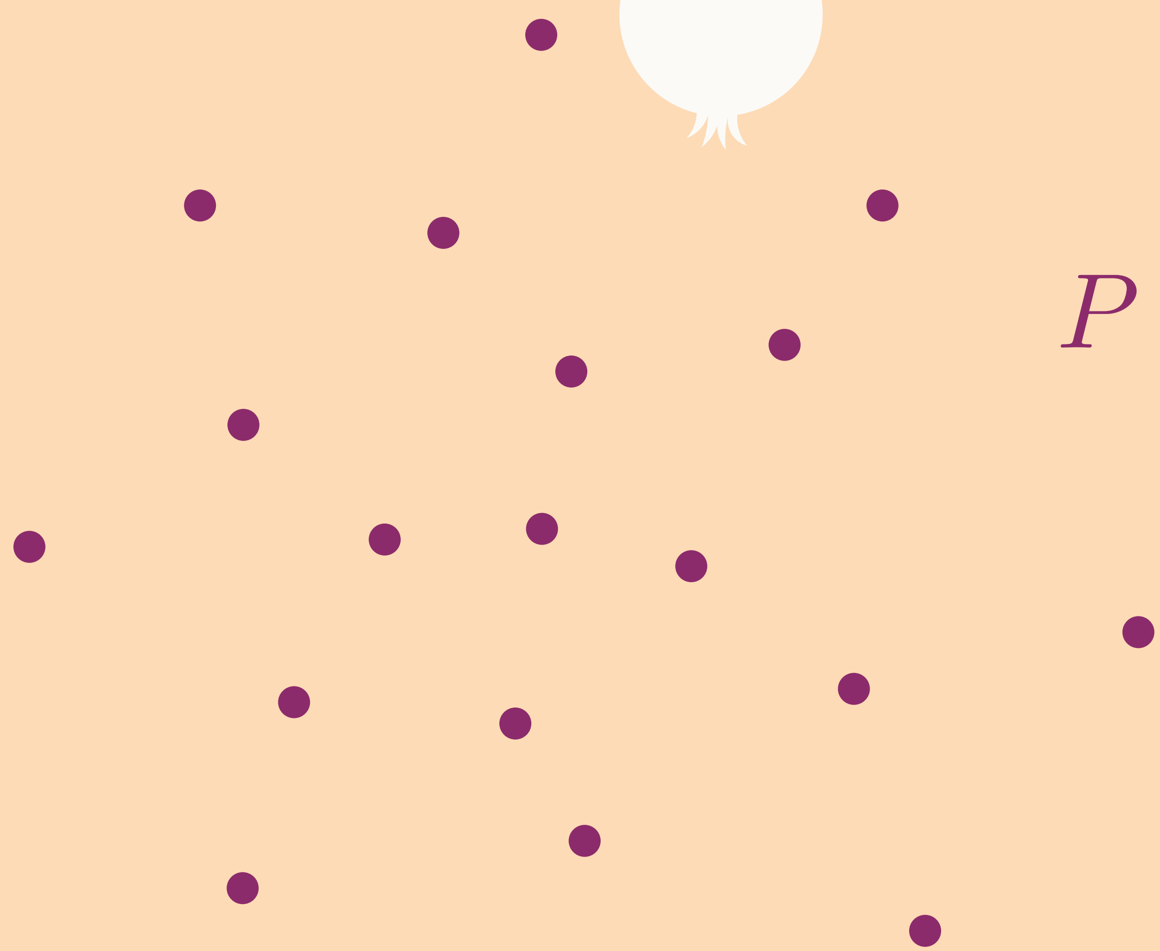
ONIONS

ONIONS

Let P be a set of n points in the plane.

ONIONS

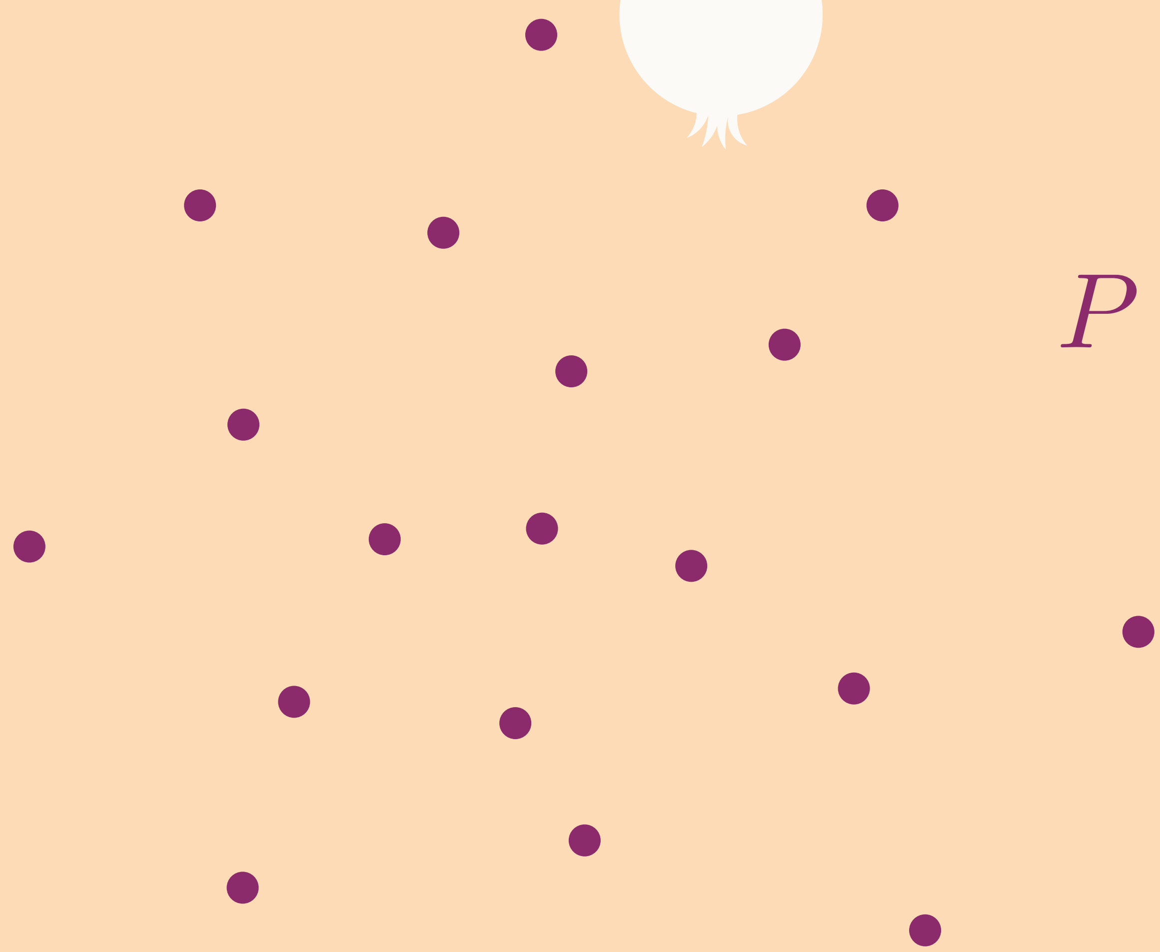
Let P be a set of n points in the plane.



ONIONS

Let P be a set of n points in the plane.

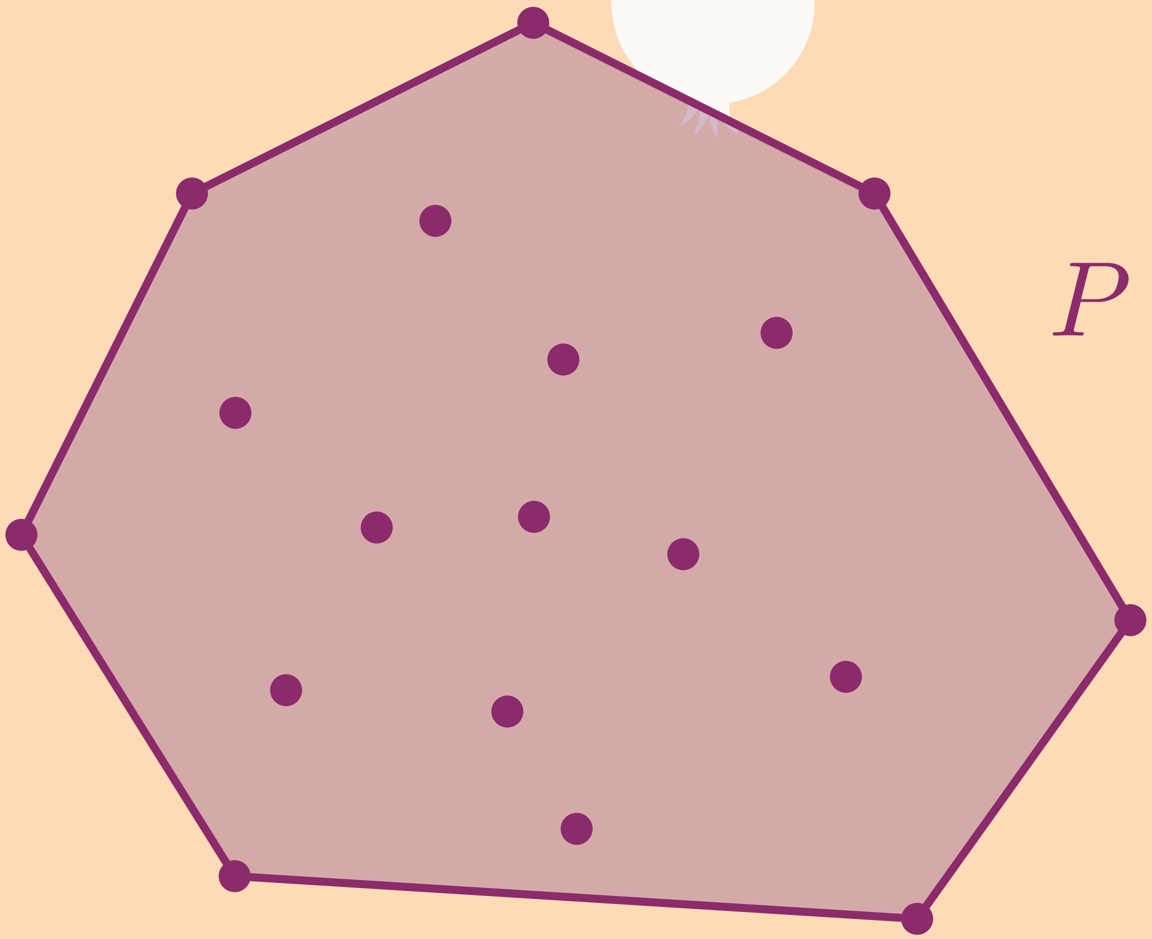
DEFINITION: The *onion* of P is the convex hull of P, plus the onion of the rest of the points.



ONIONS

Let P be a set of n points in the plane.

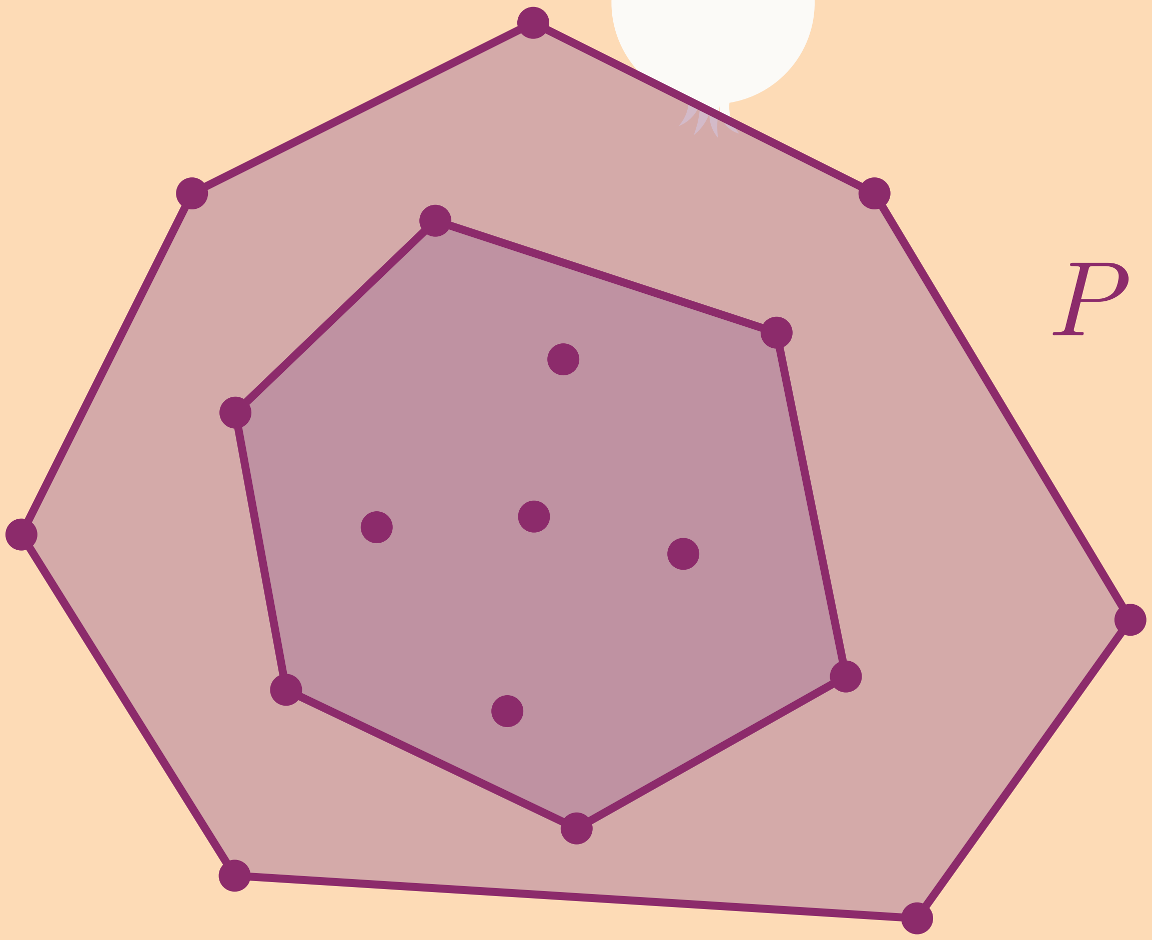
DEFINITION: The *onion* of P is the convex hull of P, plus the onion of the rest of the points.



ONIONS

Let P be a set of n points in the plane.

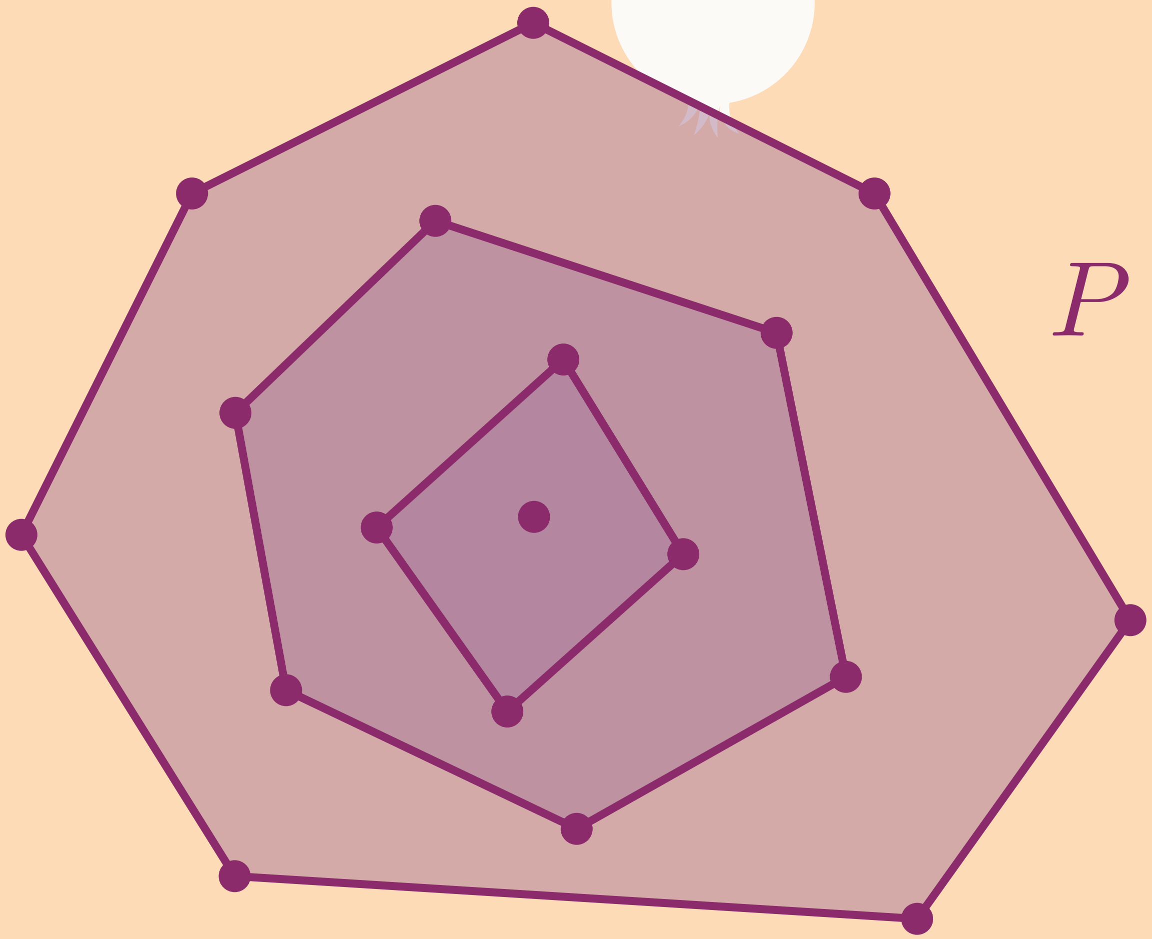
DEFINITION: The *onion* of P is the convex hull of P, plus the onion of the rest of the points.



ONIONS

Let P be a set of n points in the plane.

DEFINITION: The *onion* of P is the convex hull of P, plus the onion of the rest of the points.

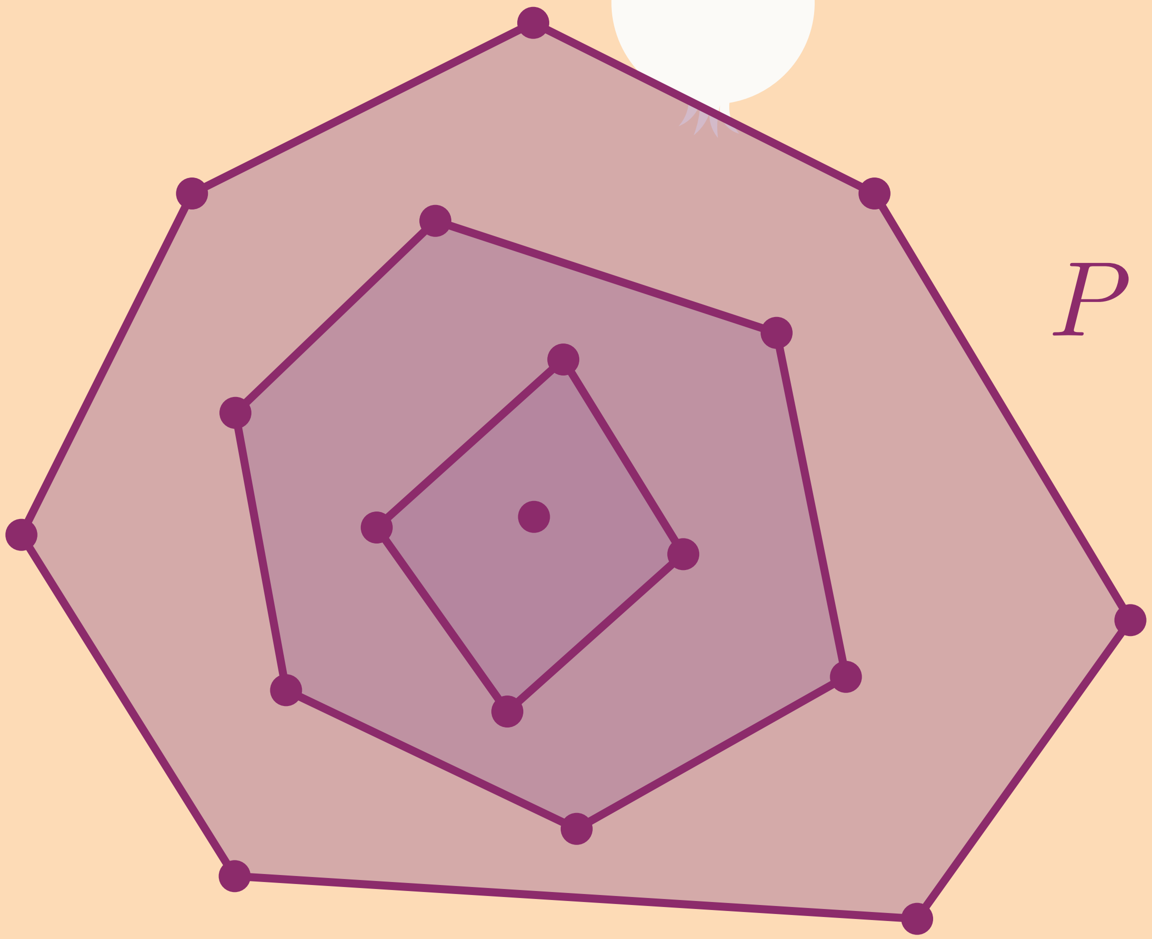


ONIONS

Let P be a set of n points in the plane.

DEFINITION: The *onion* of P is the convex hull of P, plus the onion of the rest of the points.

Onions have many useful and tasty applications.



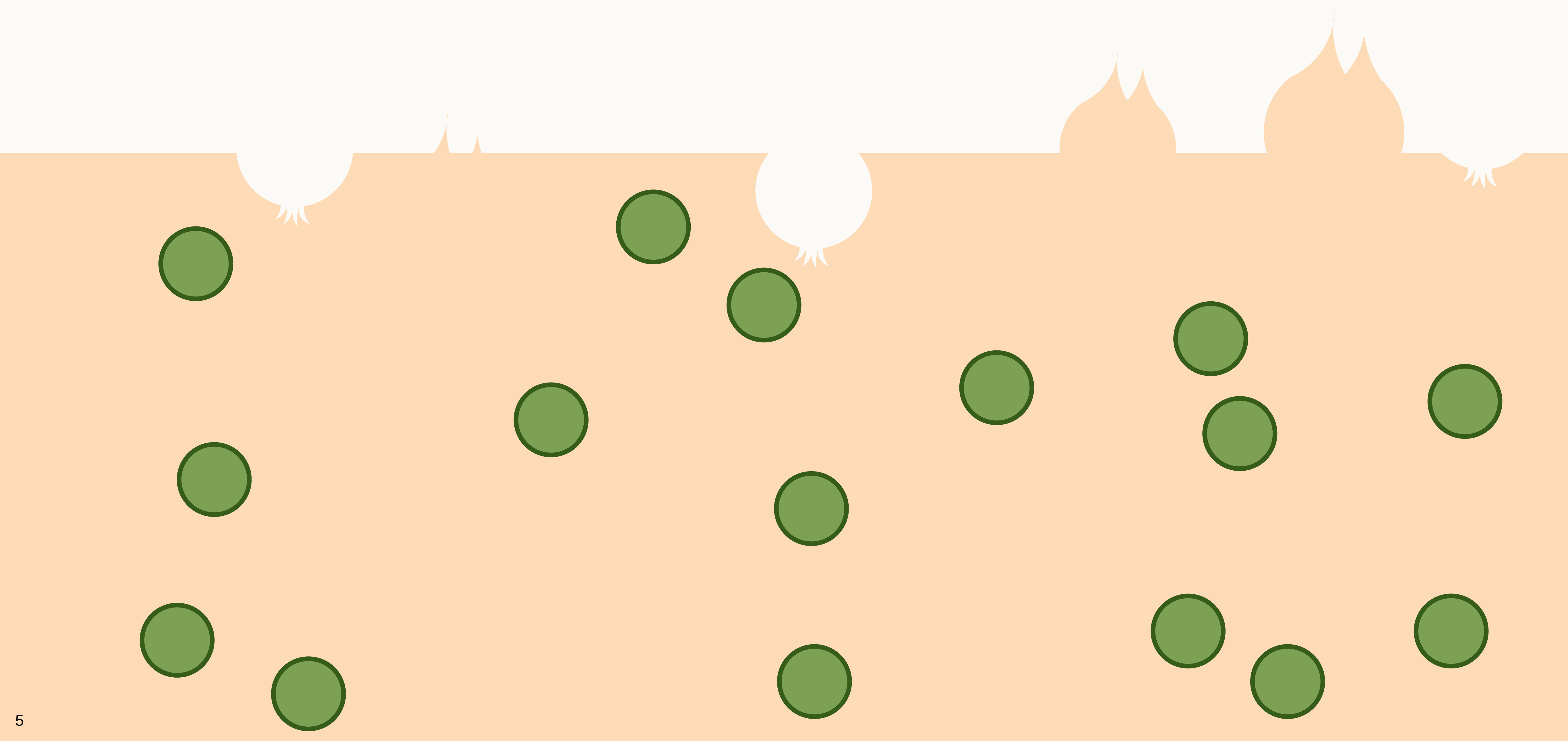
THE ONION PREPROCESSING PROBLEM

THE ONION PREPROCESSING PROBLEM

Let $\mathcal{R}$ be a set of n disjoint unit disks.

THE ONION PREPROCESSING PROBLEM

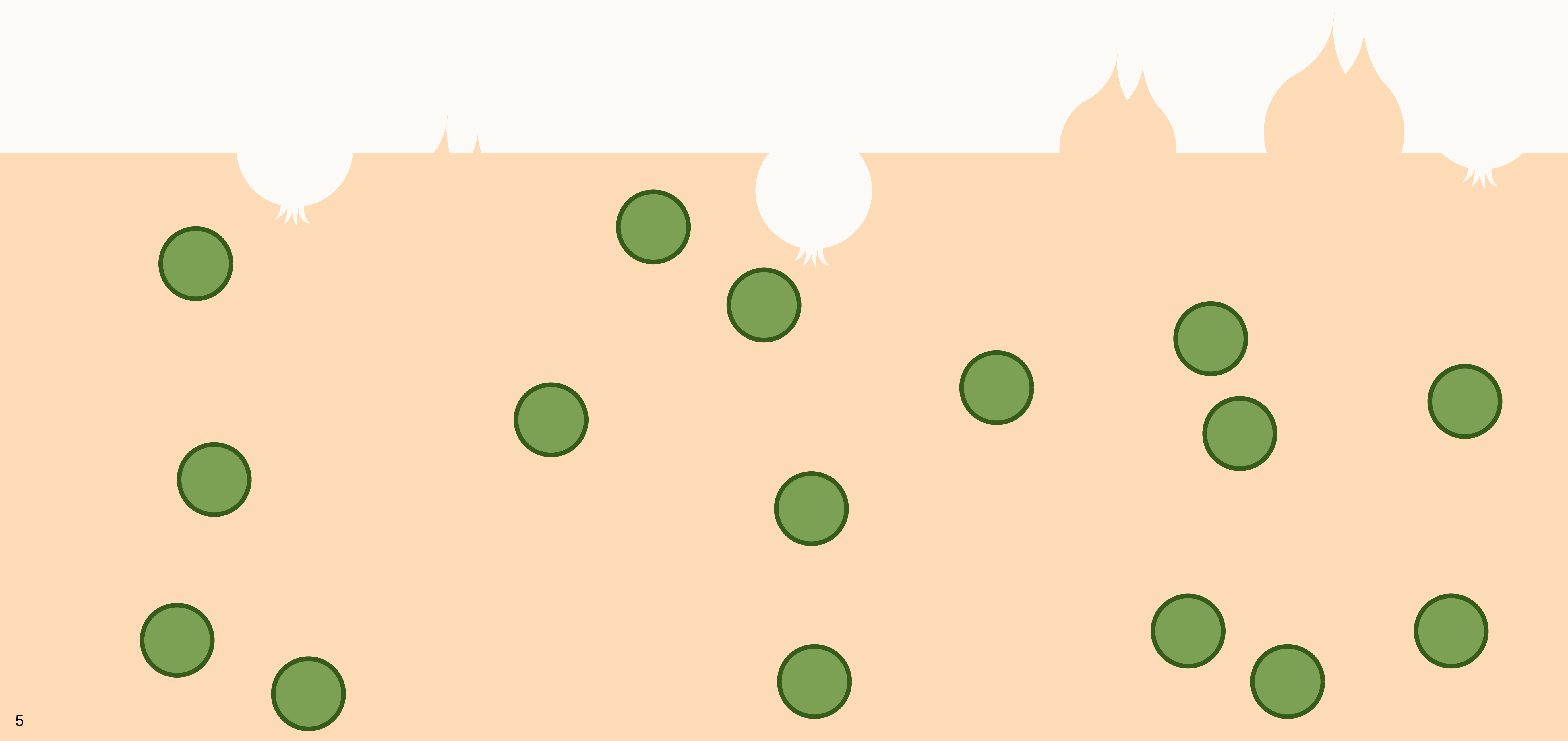
Let $\mathcal{R}$ be a set of n disjoint unit disks.



THE ONION PREPROCESSING PROBLEM

Let $\mathcal{R}$ be a set of n disjoint unit disks.

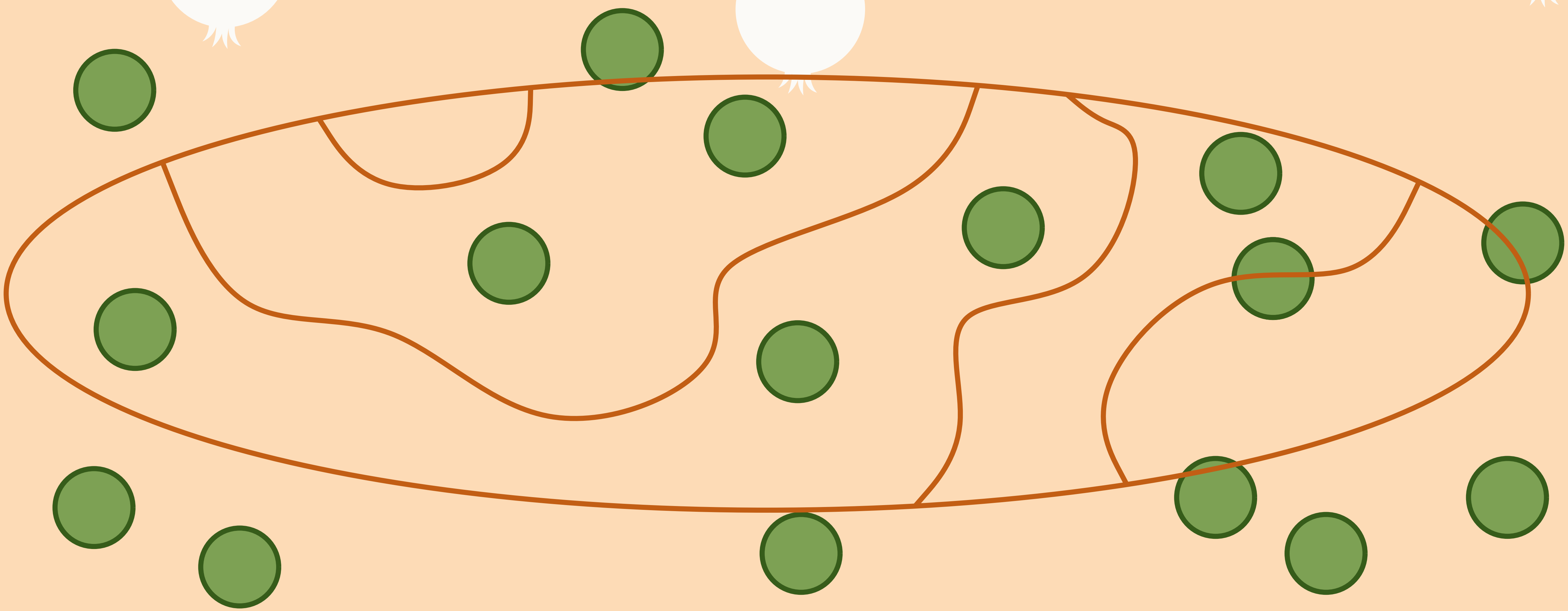
Preprocessing algorithm: compute some magical structure $H(\mathcal{R})$.



THE ONION PREPROCESSING PROBLEM

Let $\mathcal{R}$ be a set of n disjoint unit disks.

Preprocessing algorithm: compute some magical structure $H(\mathcal{R})$.

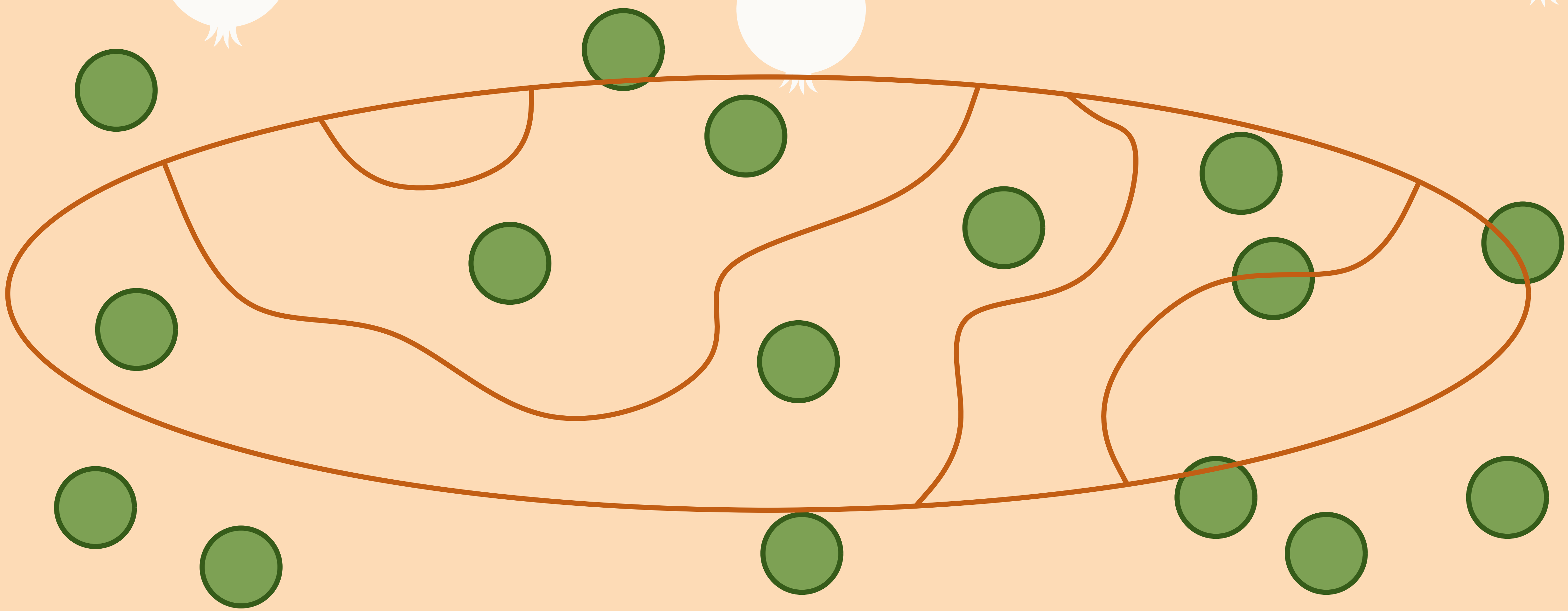


THE ONION PREPROCESSING PROBLEM

Let $\mathcal{R}$ be a set of n disjoint unit disks.

Preprocessing algorithm: compute some magical structure $H(\mathcal{R})$.

Reconstruction algorithm: given P and $H(\mathcal{R})$, compute the onion of P.

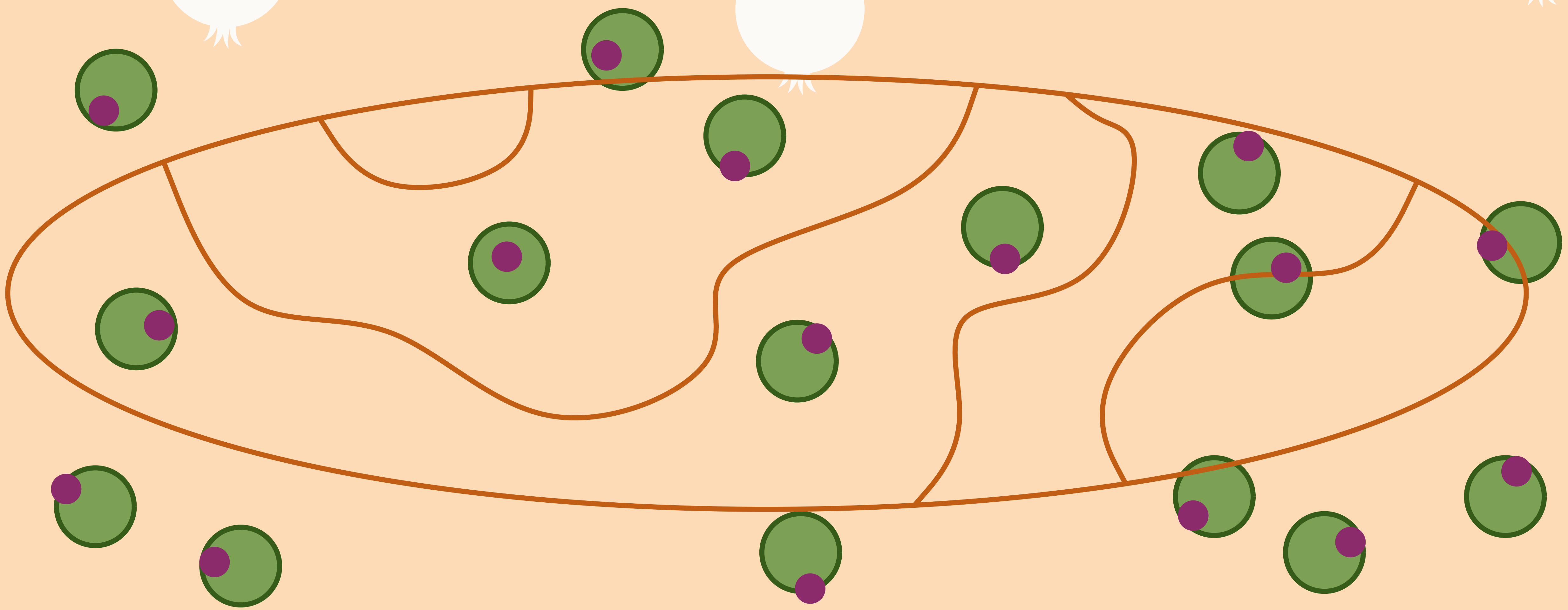


THE ONION PREPROCESSING PROBLEM

Let $\mathcal{R}$ be a set of n disjoint unit disks.

Preprocessing algorithm: compute some magical structure $H(\mathcal{R})$.

Reconstruction algorithm: given P and $H(\mathcal{R})$, compute the onion of P.

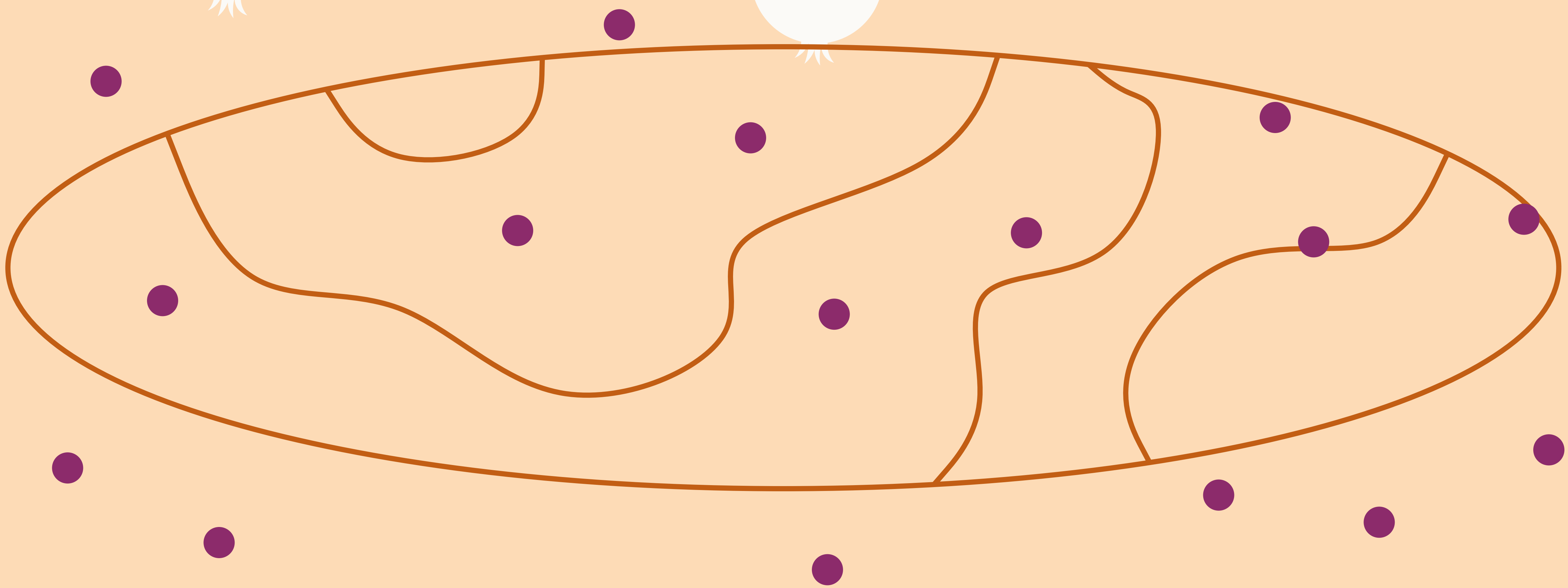


THE ONION PREPROCESSING PROBLEM

Let $\mathcal{R}$ be a set of n disjoint unit disks.

Preprocessing algorithm: compute some magical structure $H(\mathcal{R})$.

Reconstruction algorithm: given P and $H(\mathcal{R})$, compute the onion of P.

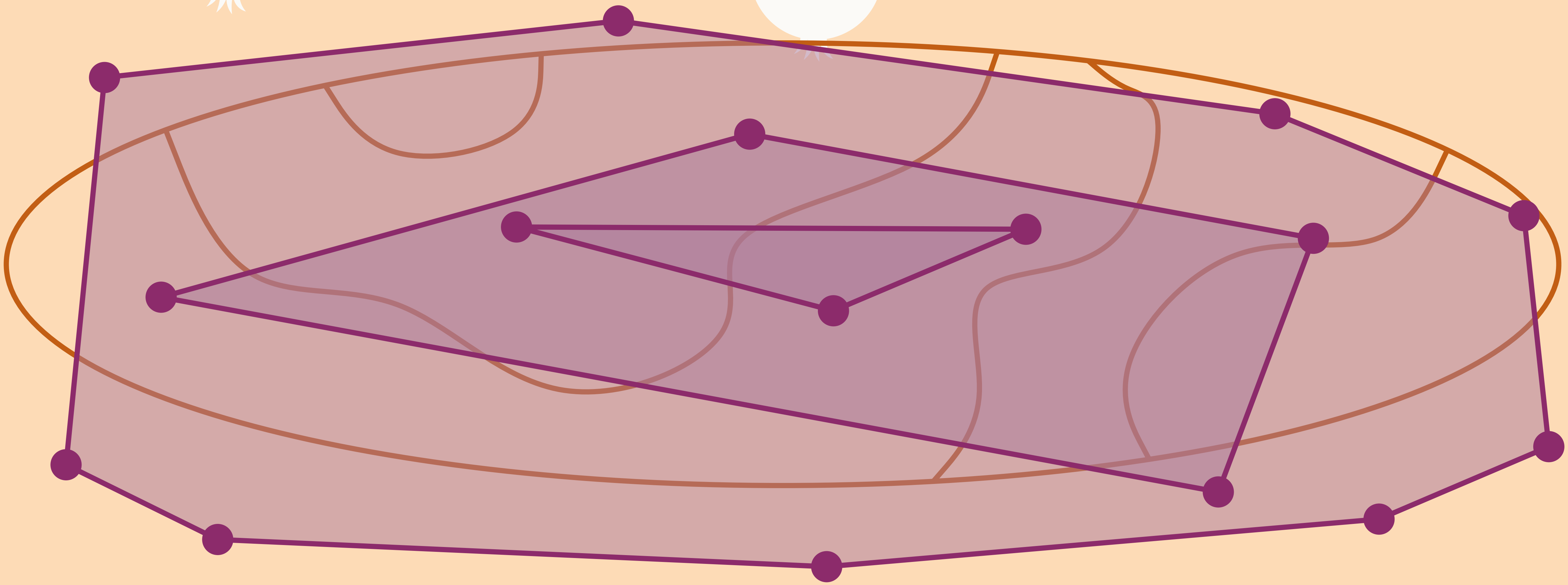


THE ONION PREPROCESSING PROBLEM

Let $\mathcal{R}$ be a set of n disjoint unit disks.

Preprocessing algorithm: compute some magical structure $H(\mathcal{R})$.

Reconstruction algorithm: given P and $H(\mathcal{R})$, compute the onion of P.



CHAPTER II

THE SOLUTION

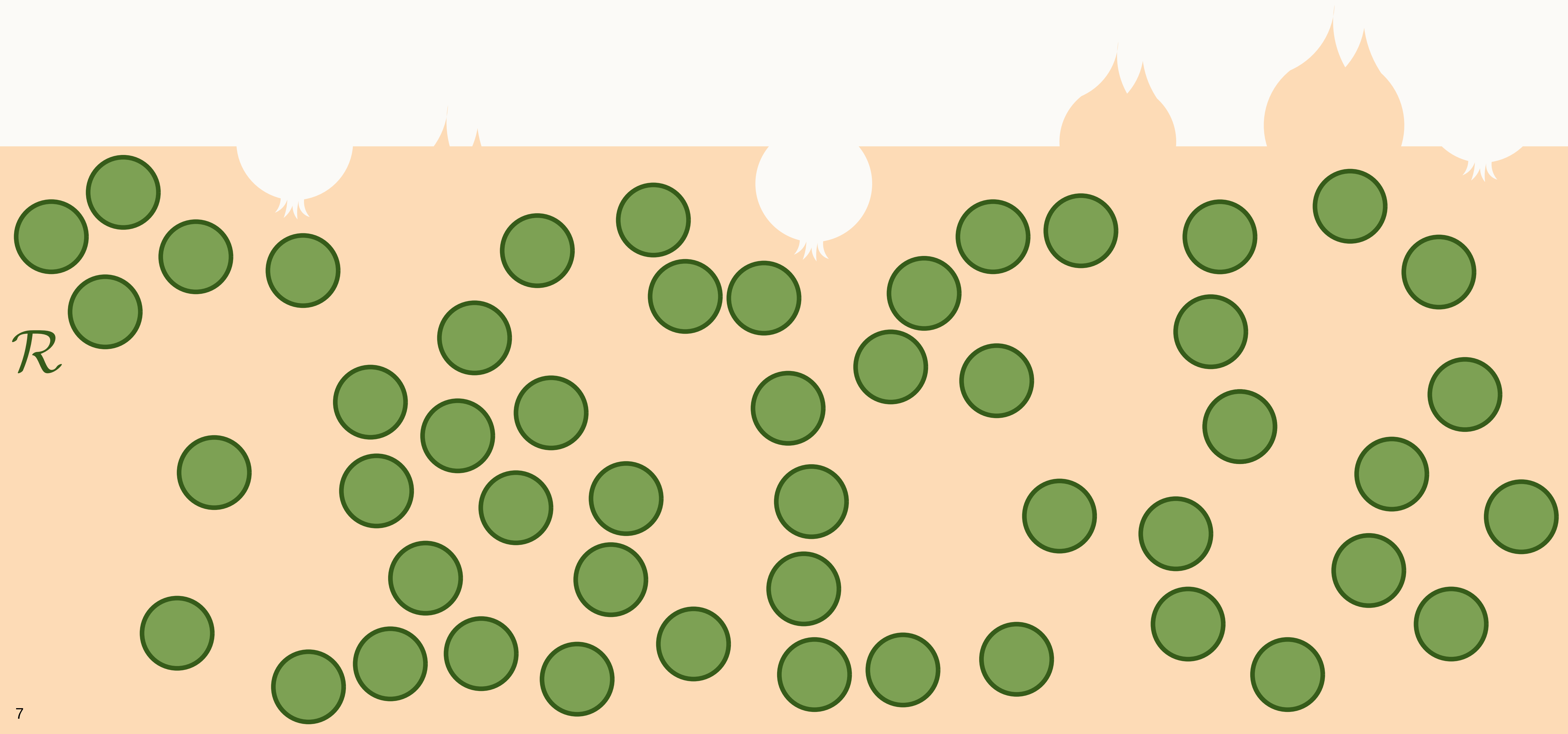
PREPROCESSING ALGORITHM

PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

PREPROCESSING ALGORITHM

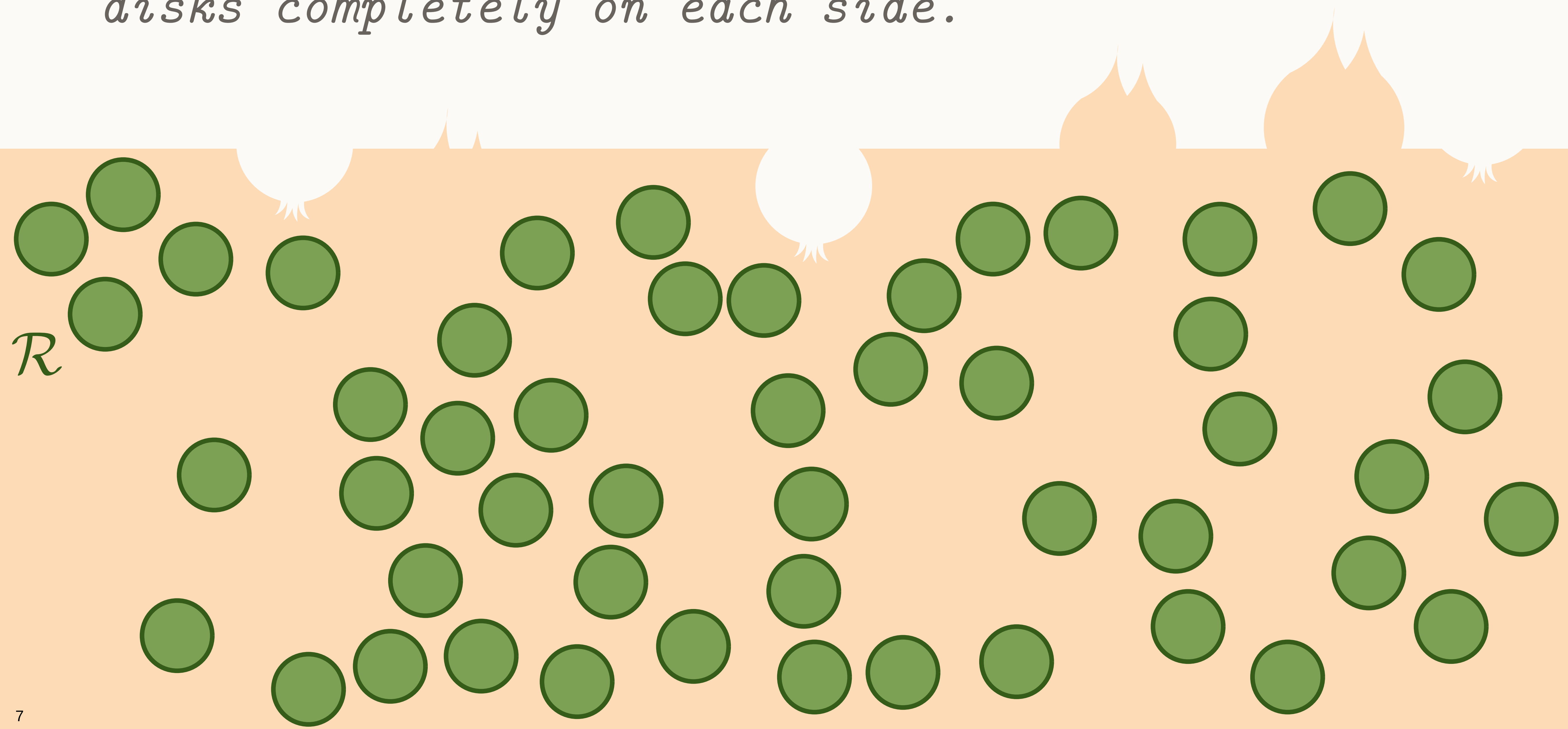
Consider a set $\mathcal{R}$ of n disjoint unit disks.



PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

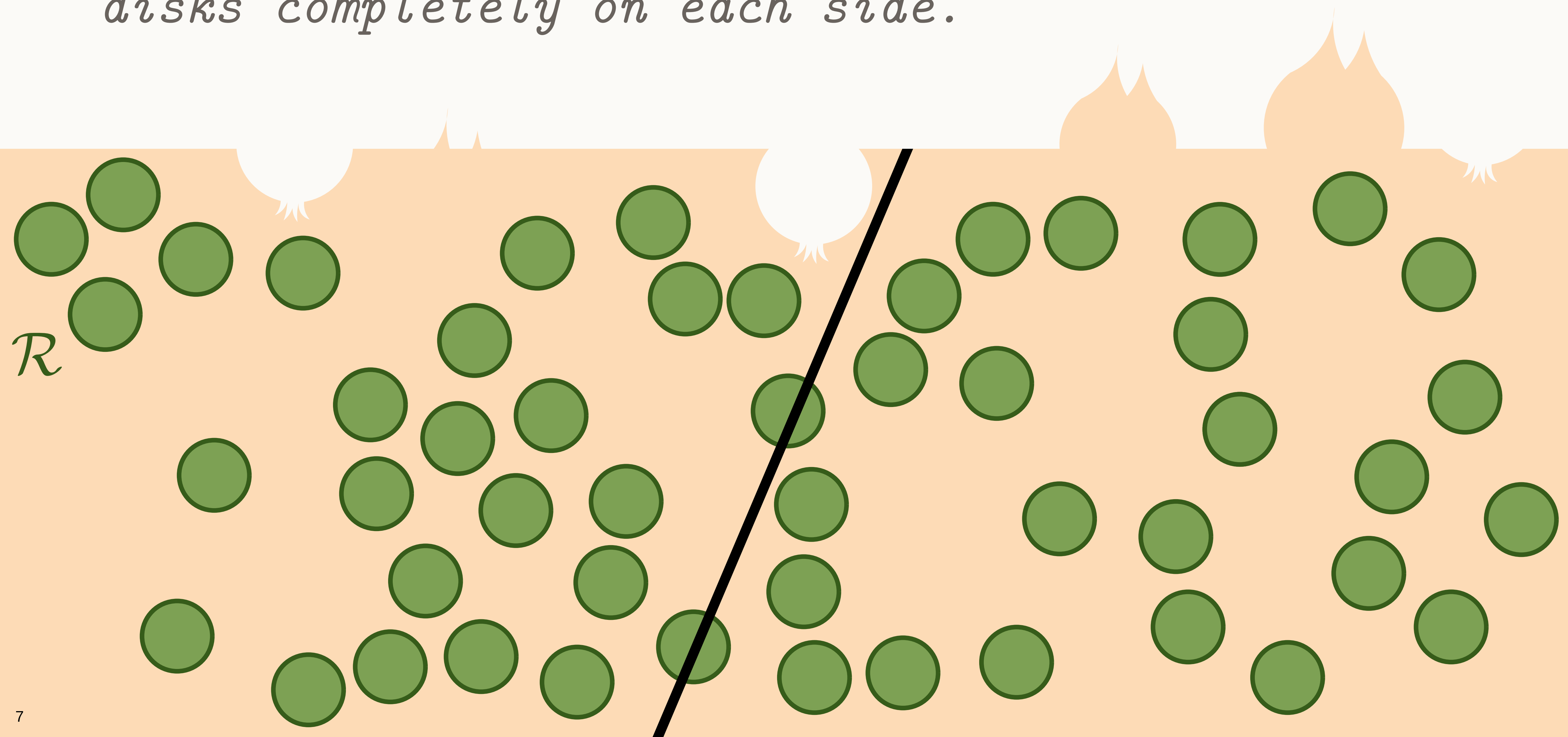
LEMMA: *There exists a line that intersects at most $\sqrt{n \log n}$ disks and has at most half of the disks completely on each side.*



PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

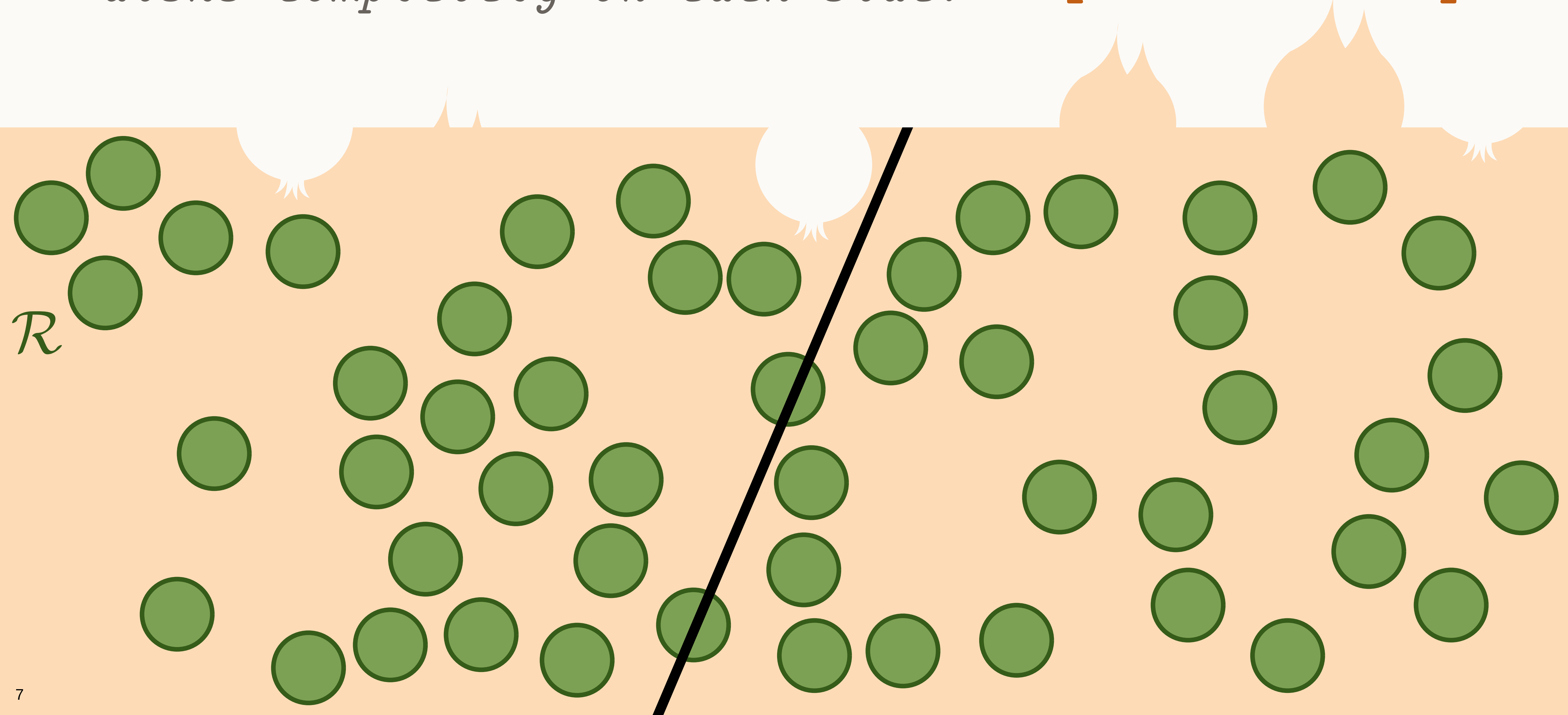
LEMMA: *There exists a line that intersects at most $\sqrt{n \log n}$ disks and has at most half of the disks completely on each side.*



PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

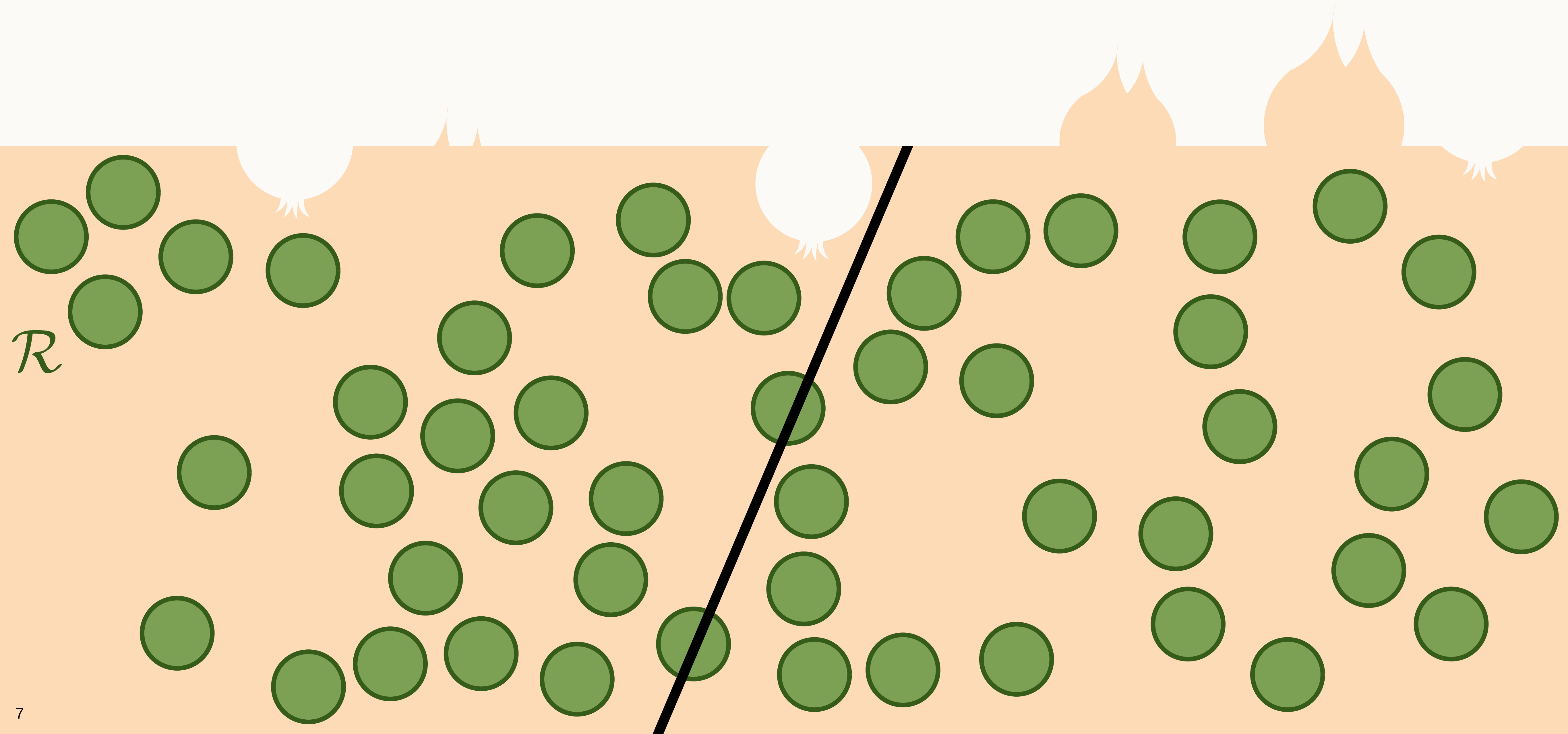
LEMMA: *There exists a line that intersects at most $\sqrt{n \log n}$ disks and has at most half of the disks completely on each side.* [Alon *et al.*]



PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

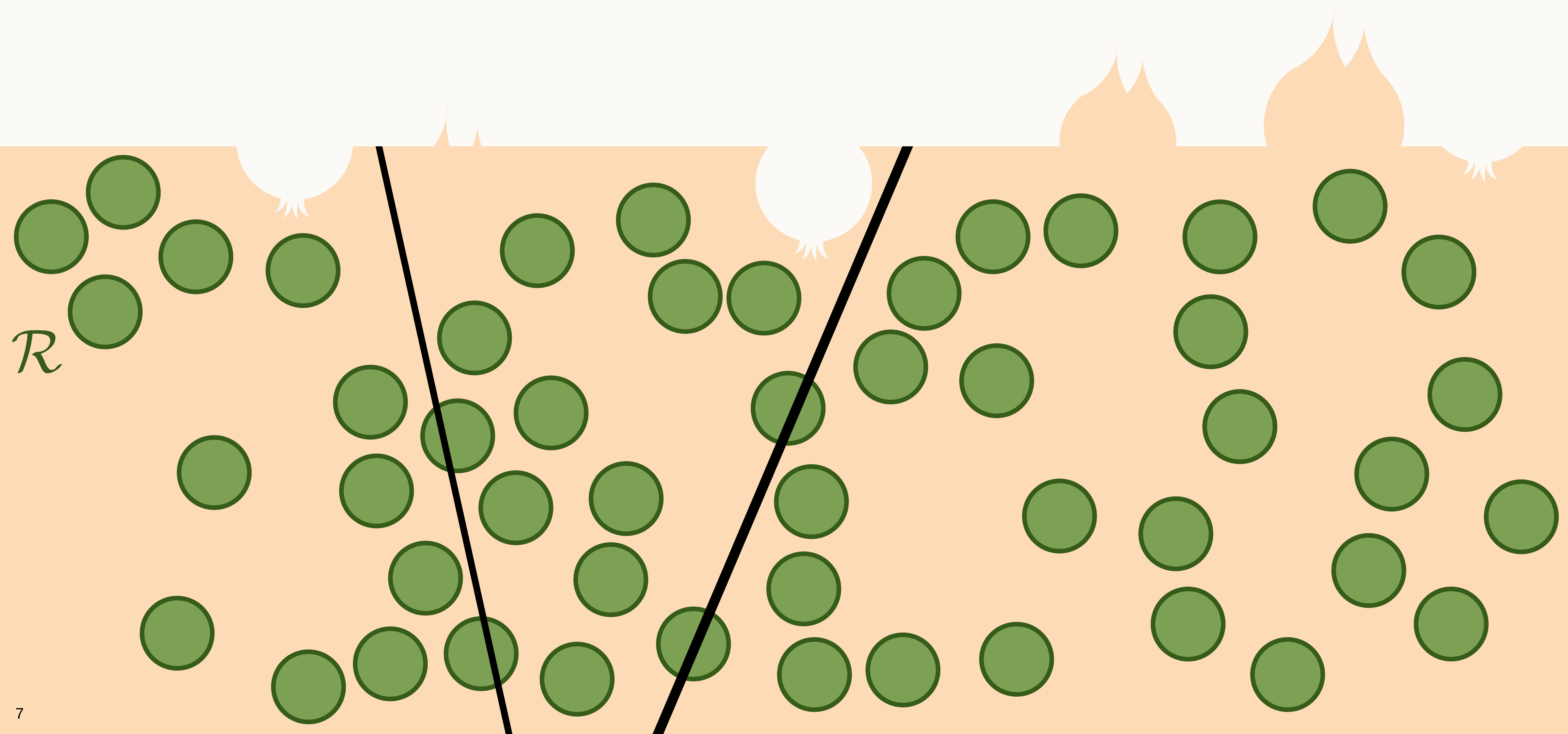
We apply the lemma recursively, and get a binary space partition tree T.



PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

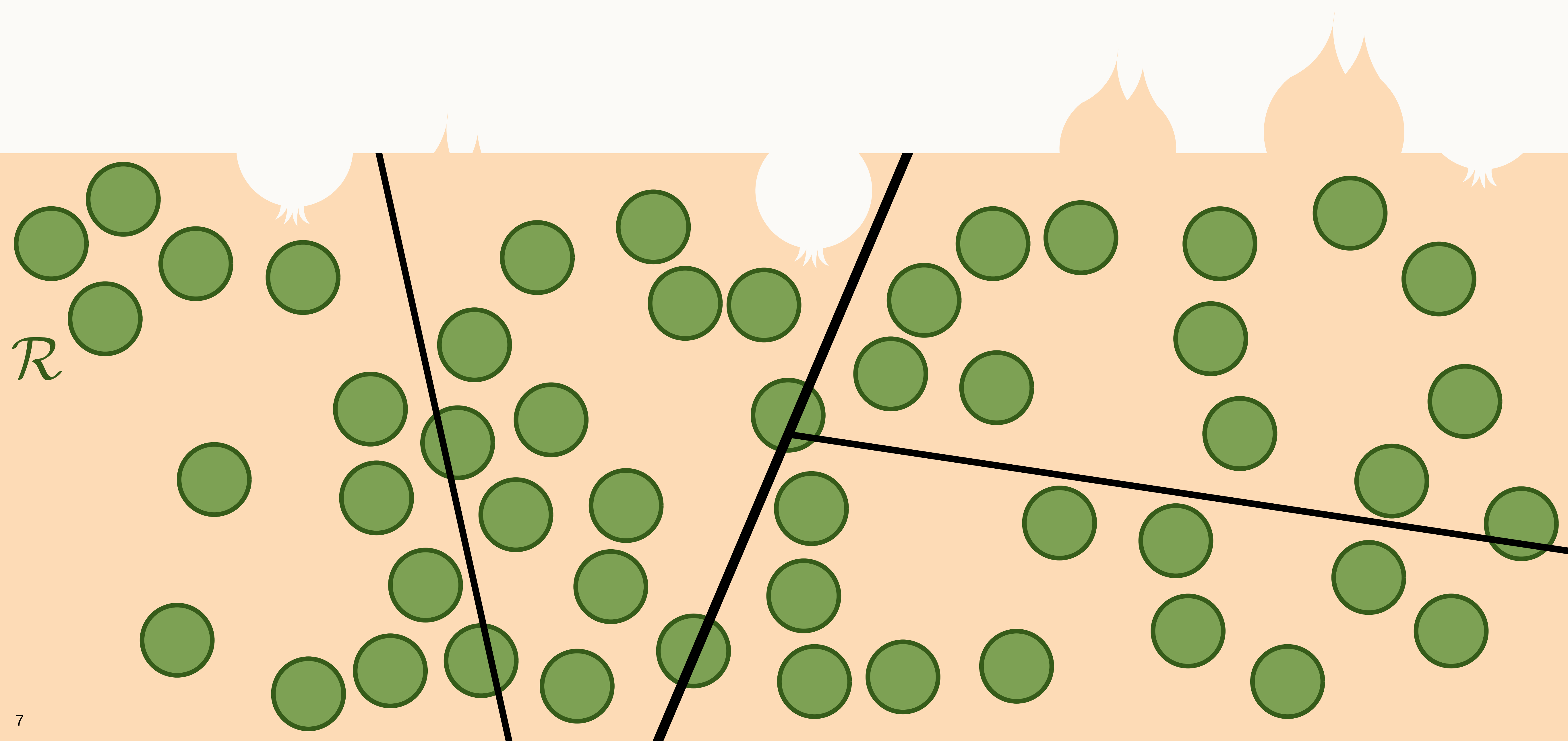
We apply the lemma recursively, and get a binary space partition tree T.



PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

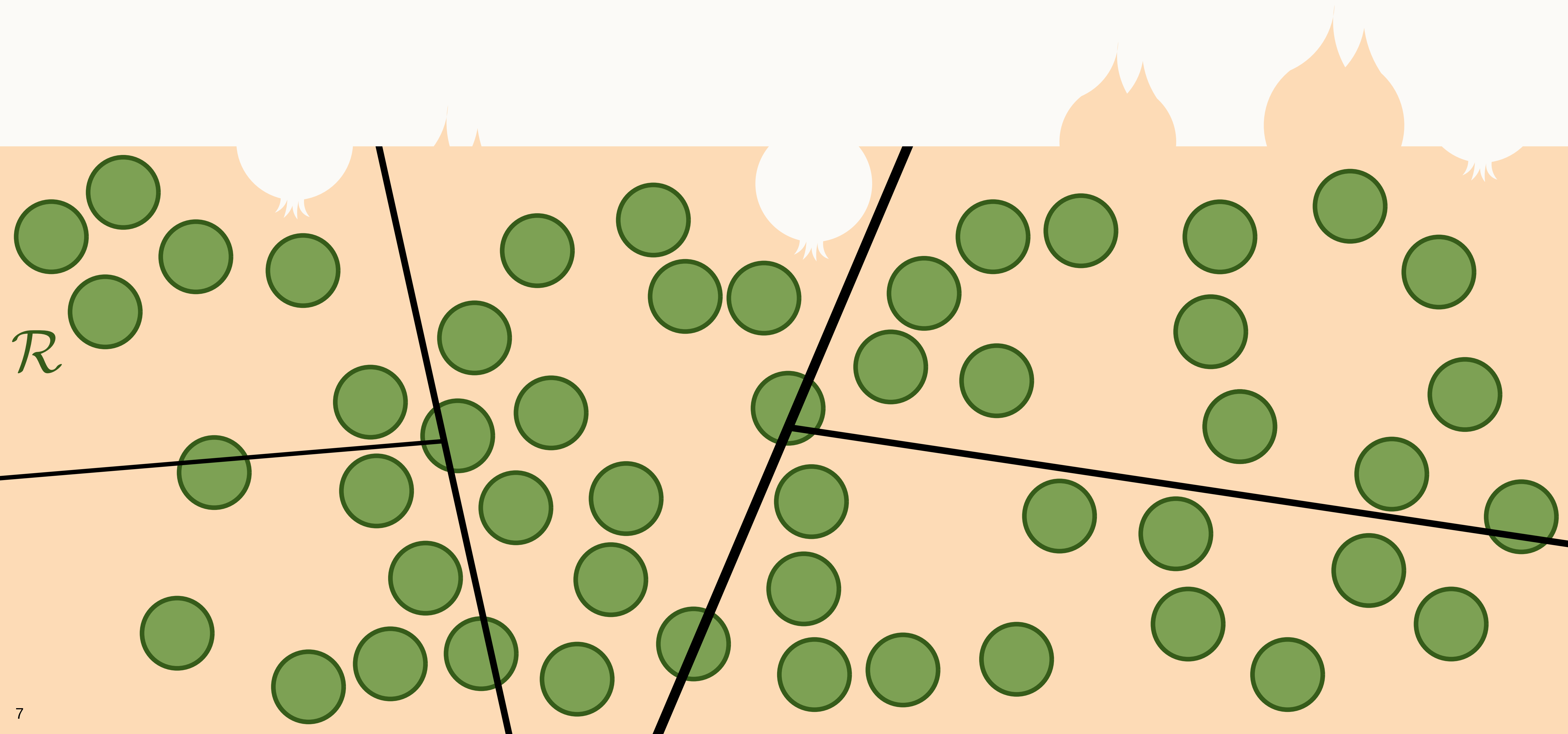
We apply the lemma recursively, and get a binary space partition tree T.



PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

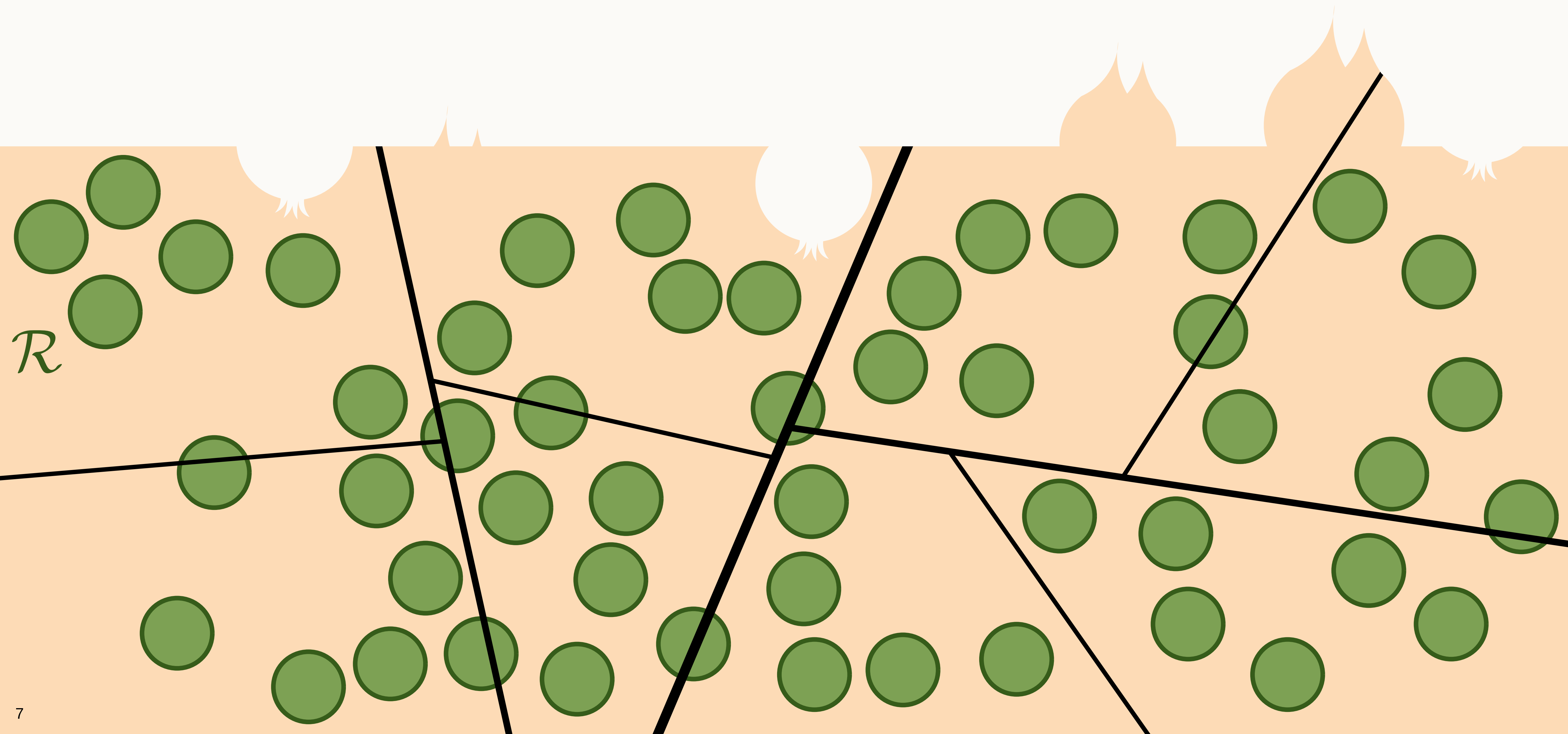
We apply the lemma recursively, and get a binary space partition tree T.



PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

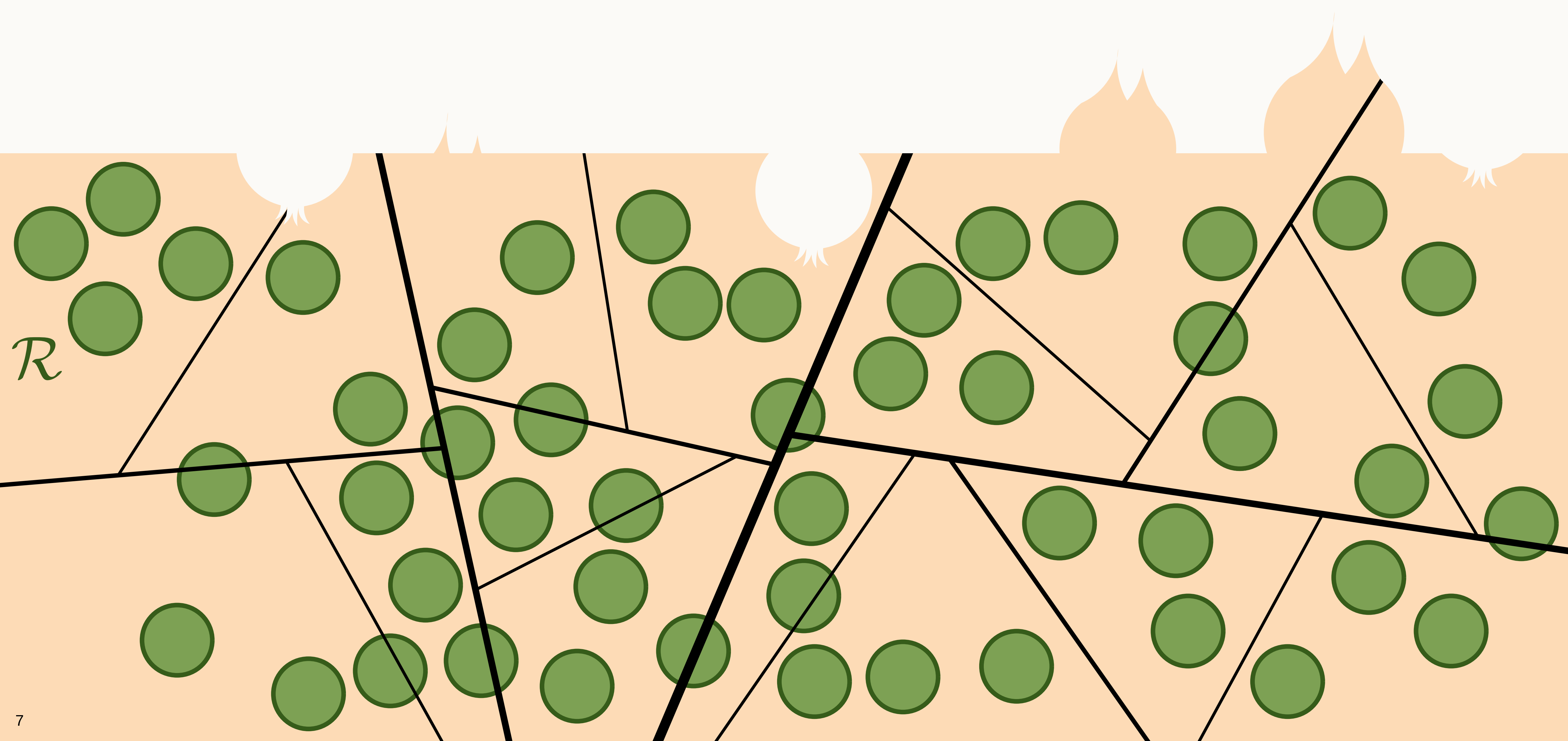
We apply the lemma recursively, and get a binary space partition tree T.



PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

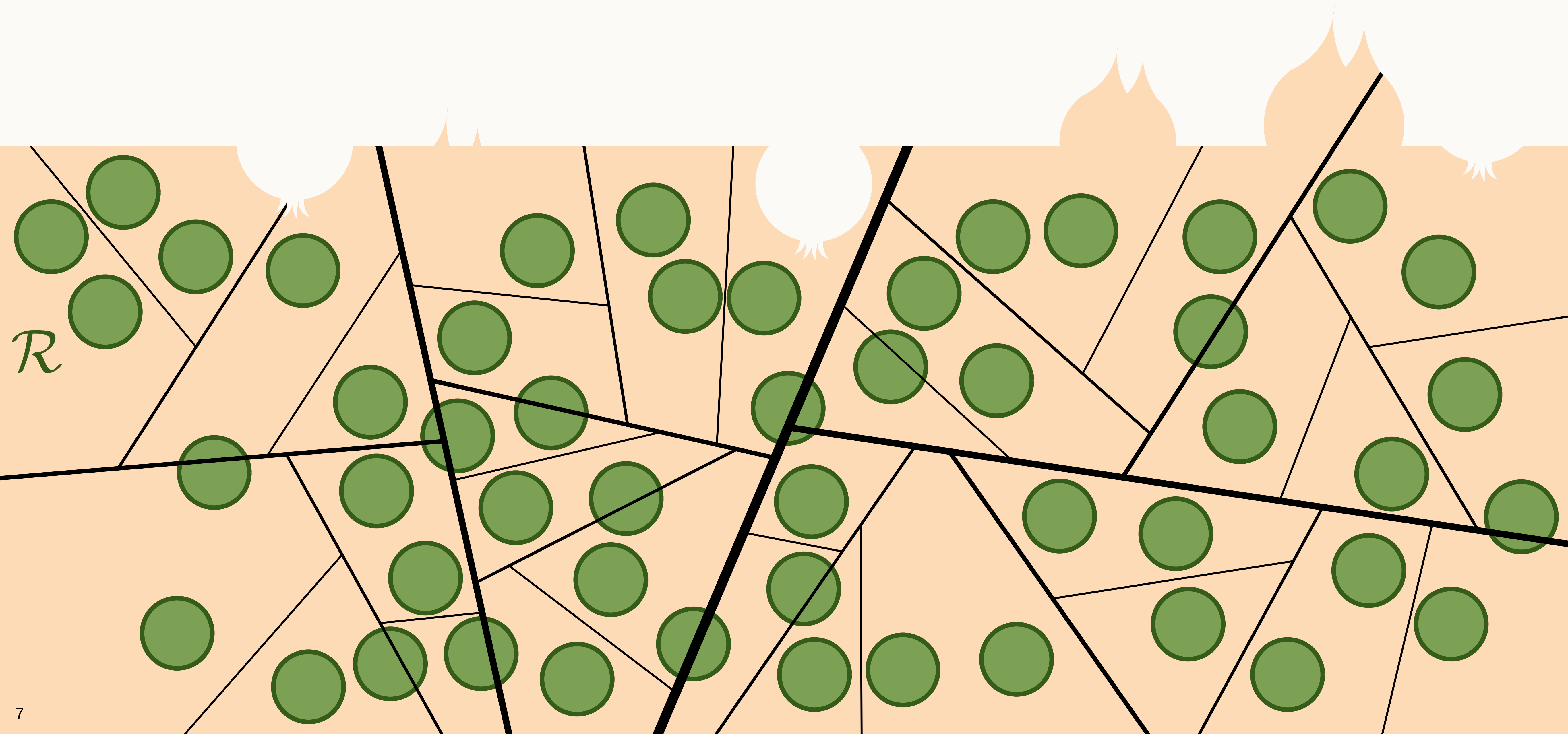
We apply the lemma recursively, and get a binary space partition tree T.



PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

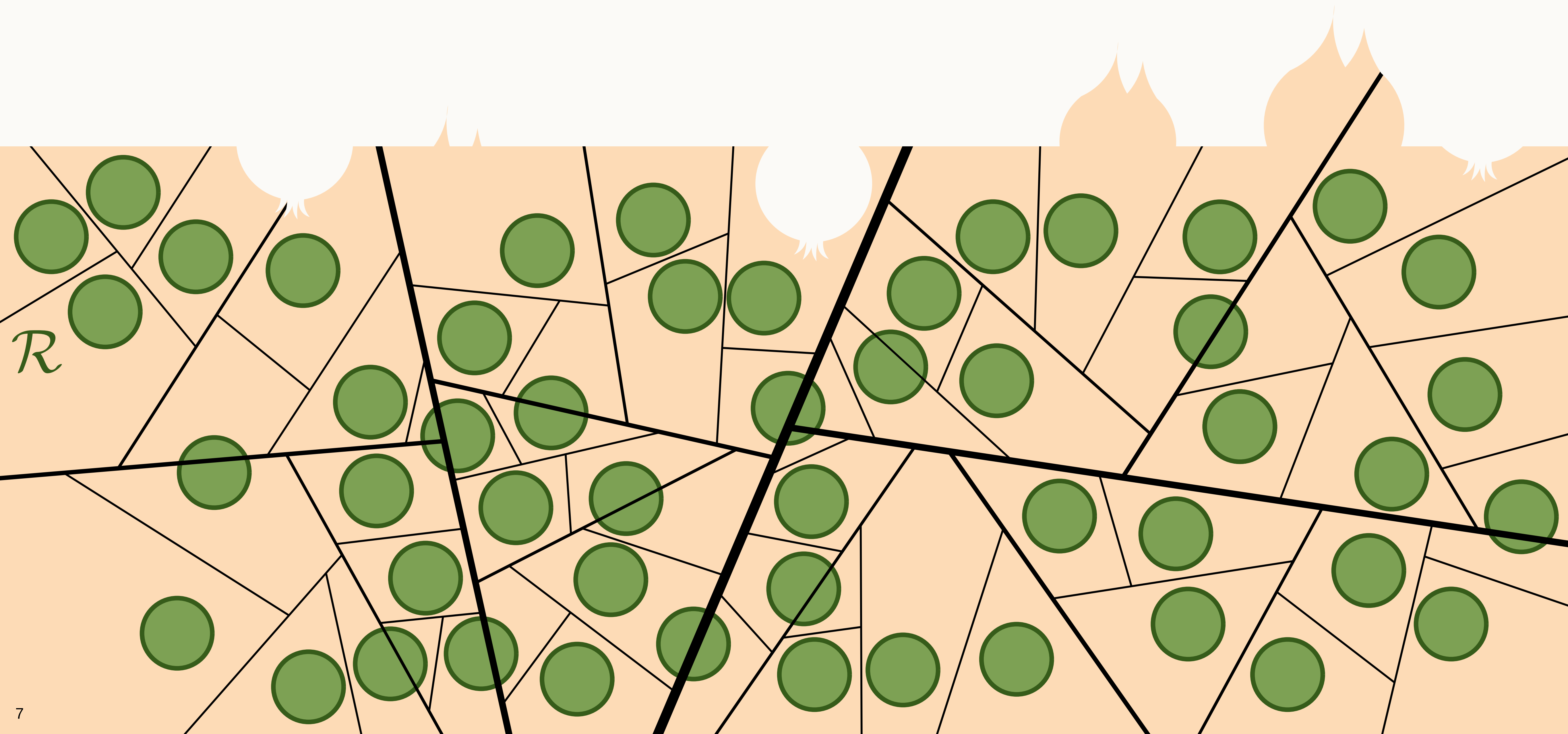
We apply the lemma recursively, and get a binary space partition tree T.



PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

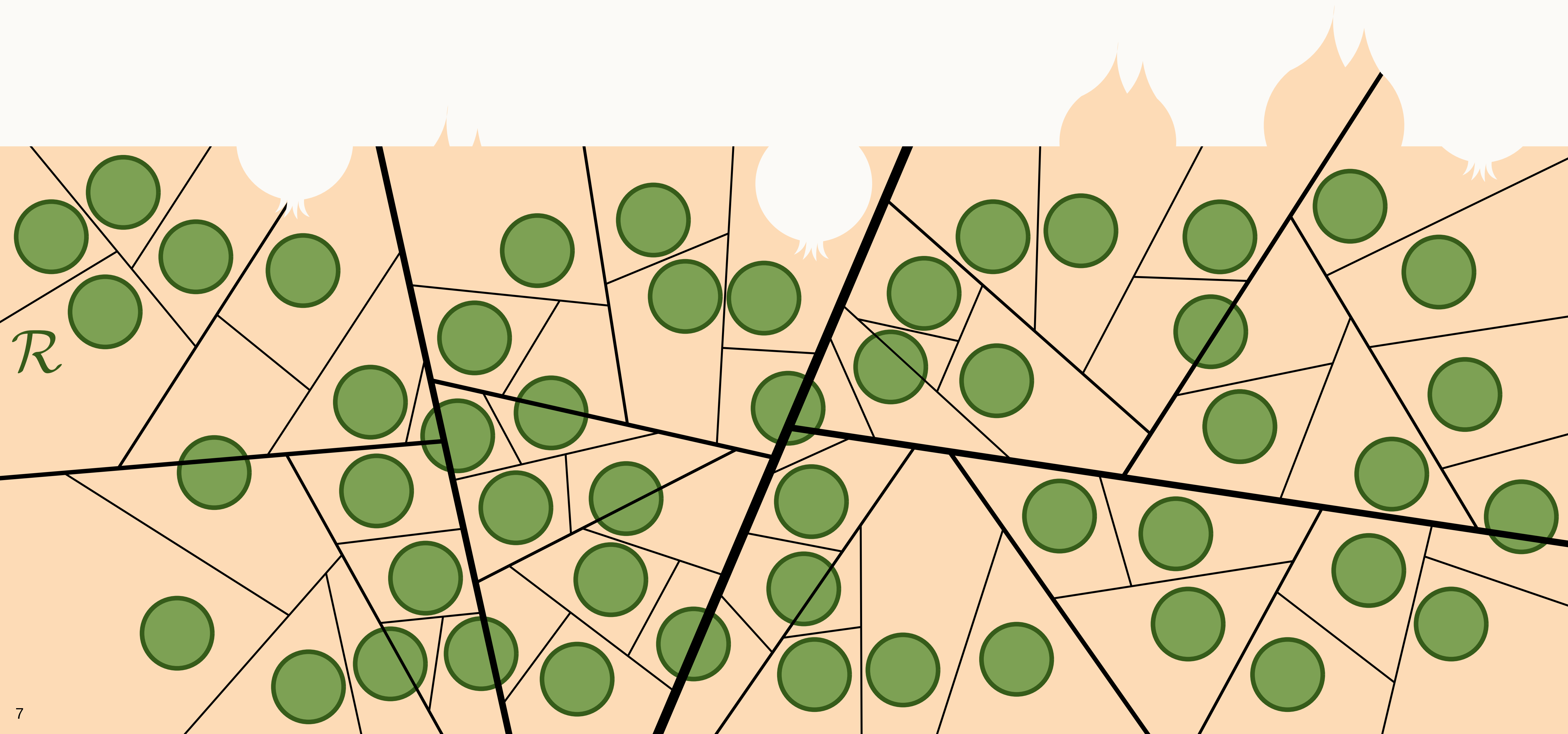
We apply the lemma recursively, and get a binary space partition tree T.



PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

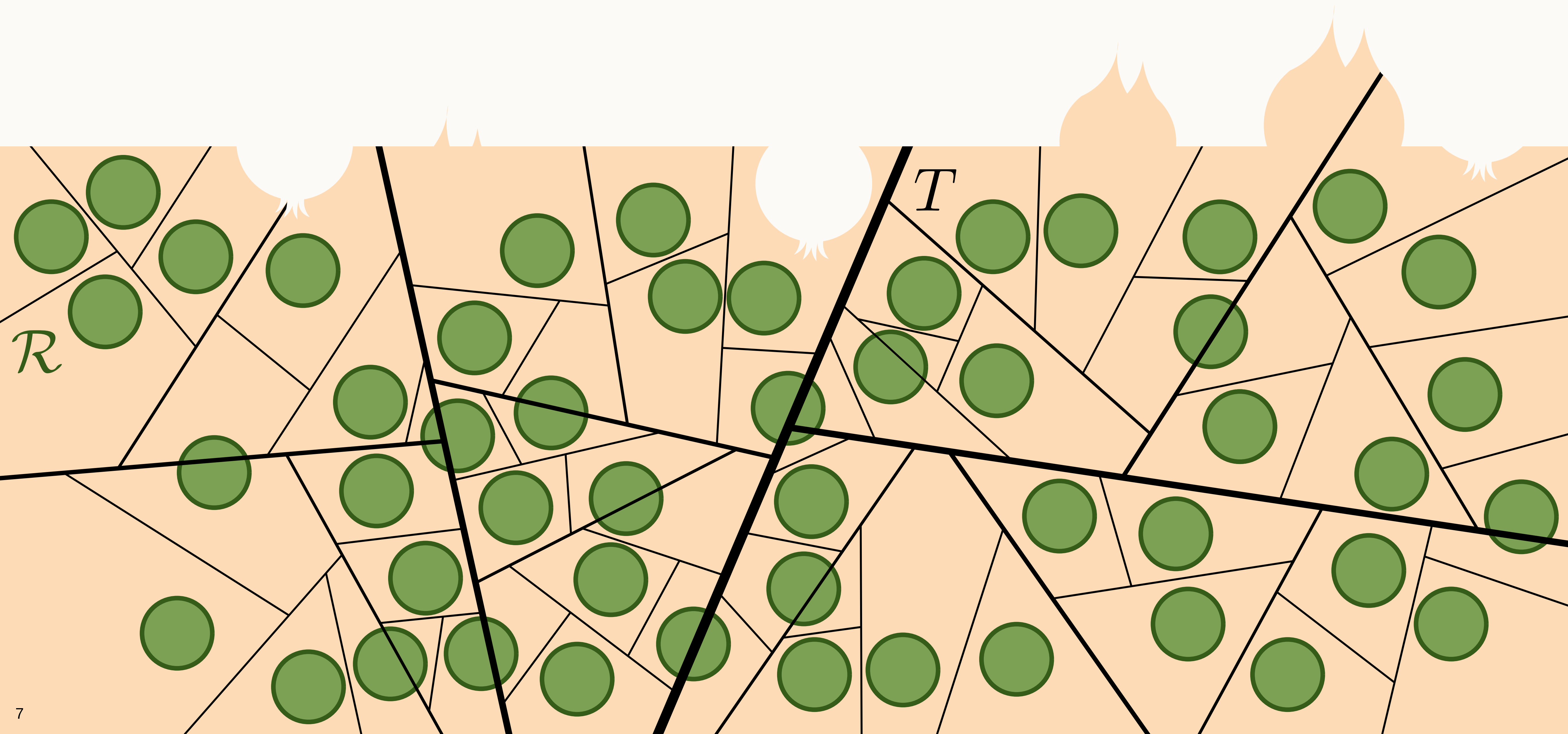
We apply the lemma recursively, and get a binary space partition tree T.



PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

We apply the lemma recursively, and get a binary space partition tree T.

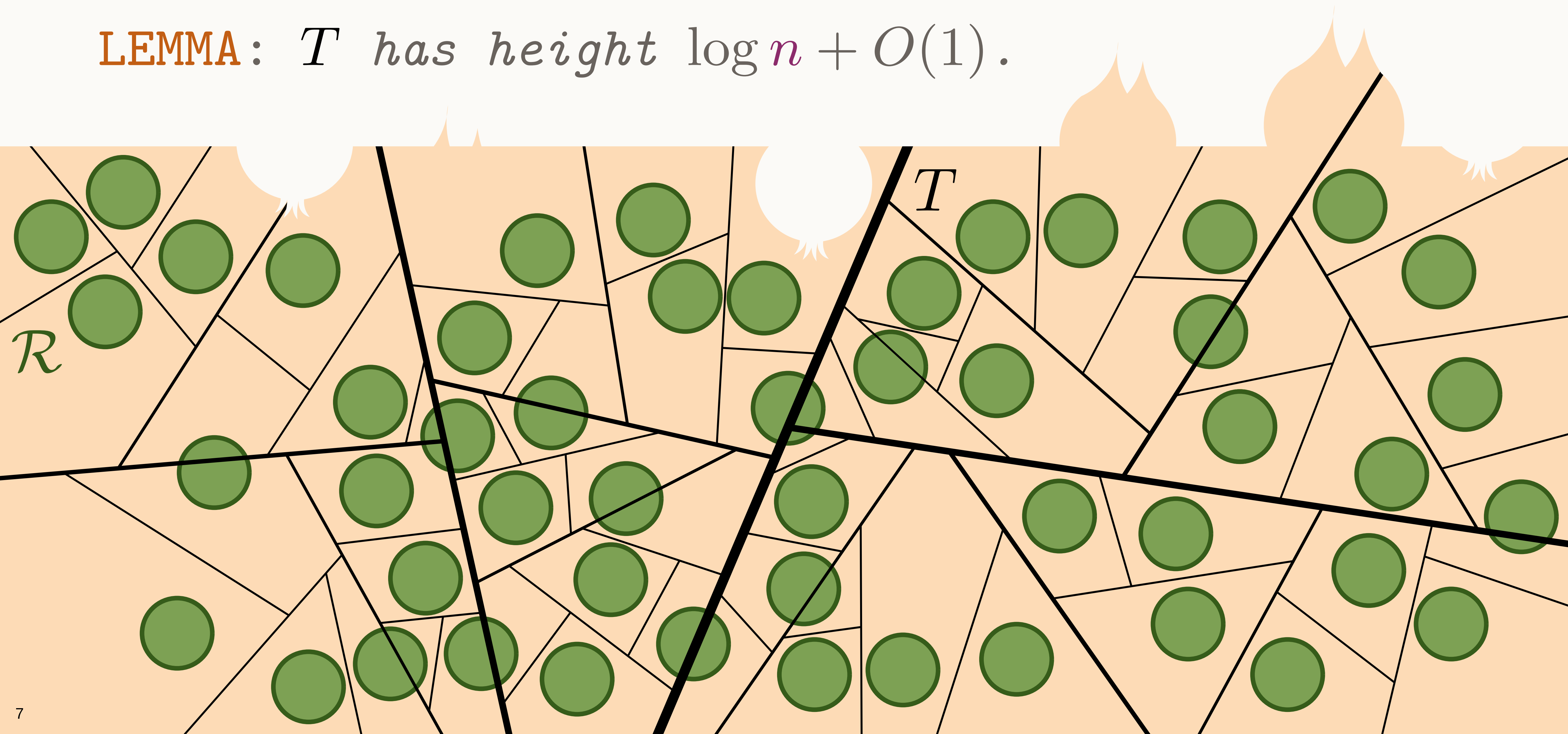


PREPROCESSING ALGORITHM

Consider a set $\mathcal{R}$ of n disjoint unit disks.

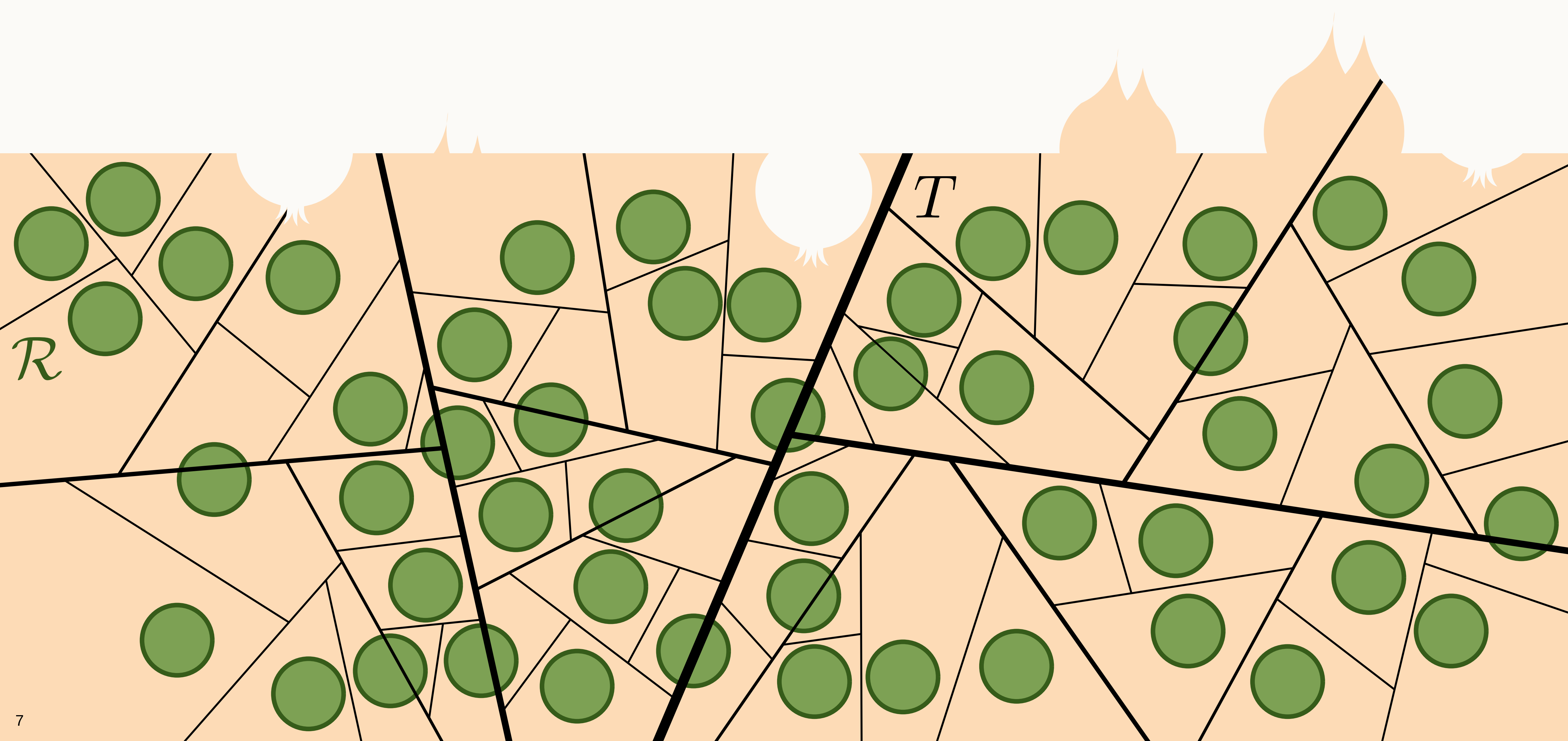
We apply the lemma recursively, and get a binary space partition tree T.

LEMMA: T *has height* $\log n + O(1)$.

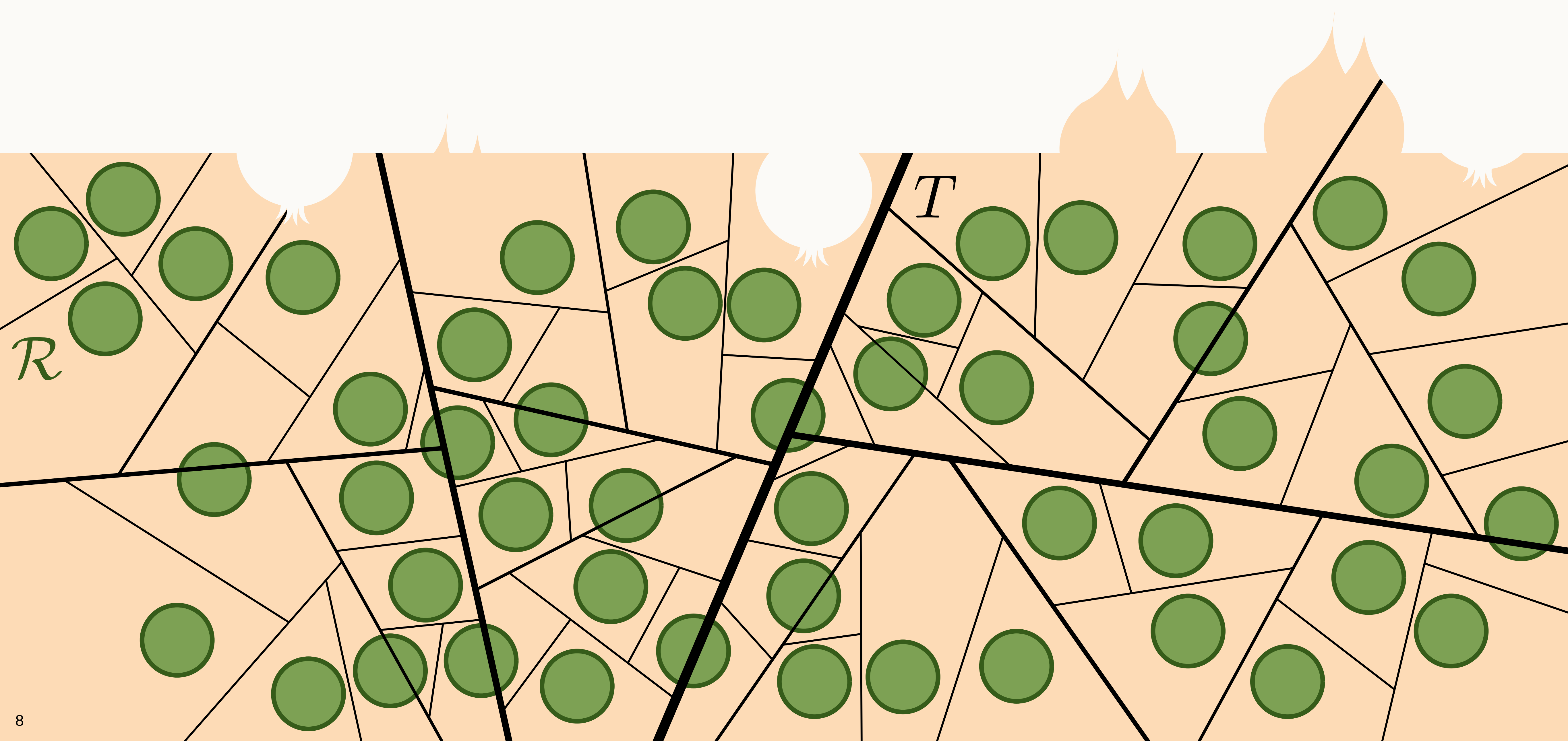


PREPROCESSING ALGORITHM

HINT: $T = H(\mathcal{R})$ *is our magical structure!*

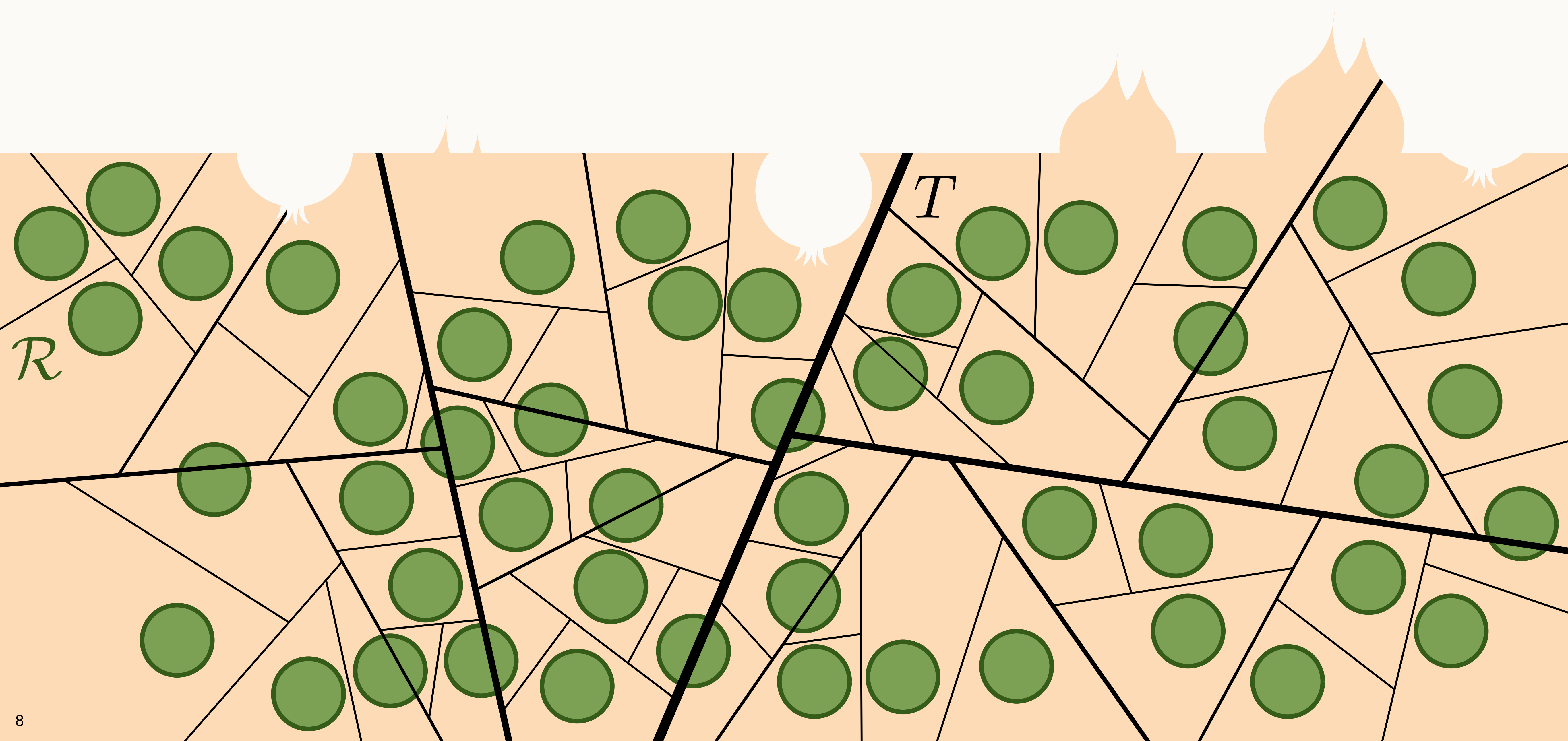


RECONSTRUCTION ALGORITHM



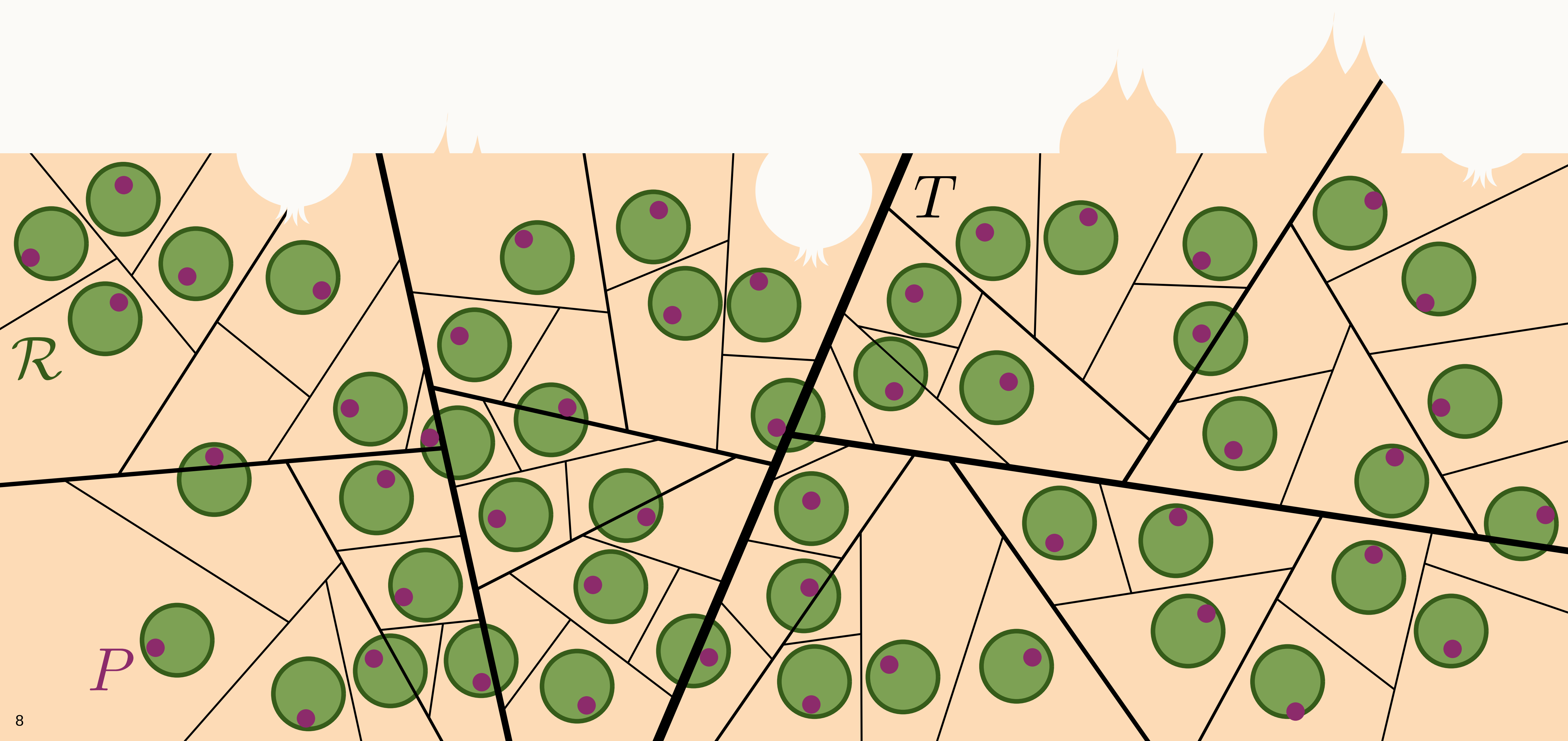
RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.



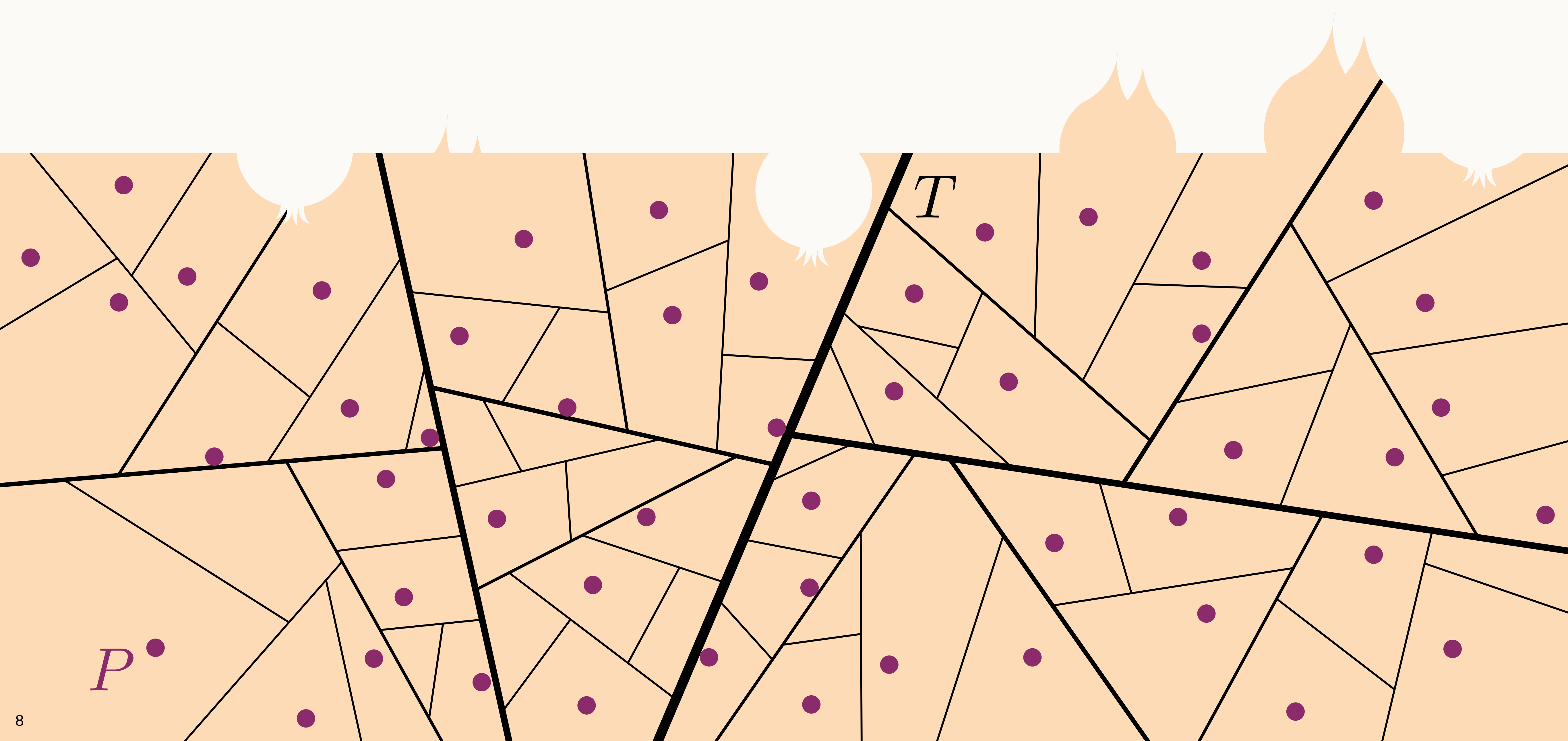
RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.



RECONSTRUCTION ALGORITHM

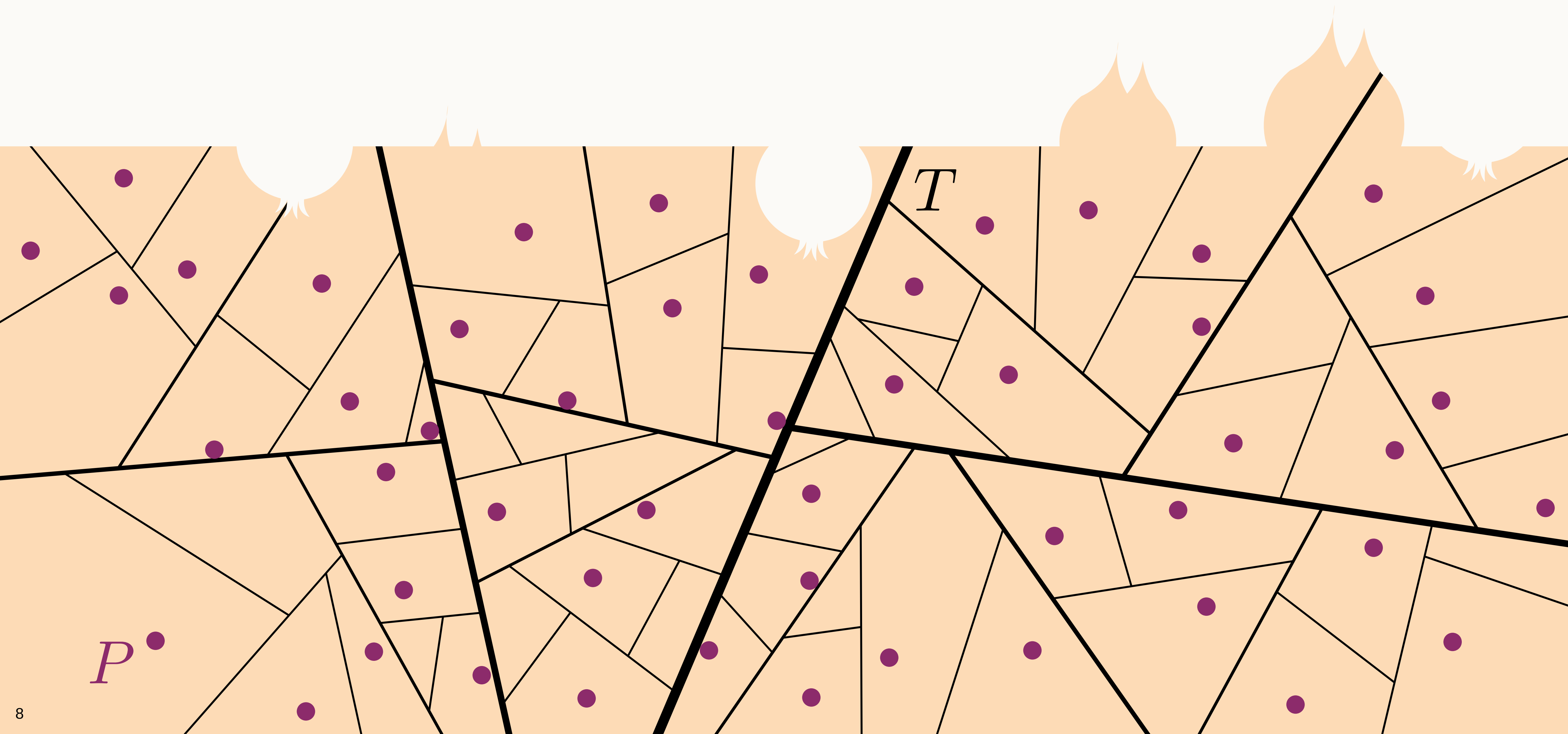
First, locate P in $\mathcal{R}$ and T.



RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

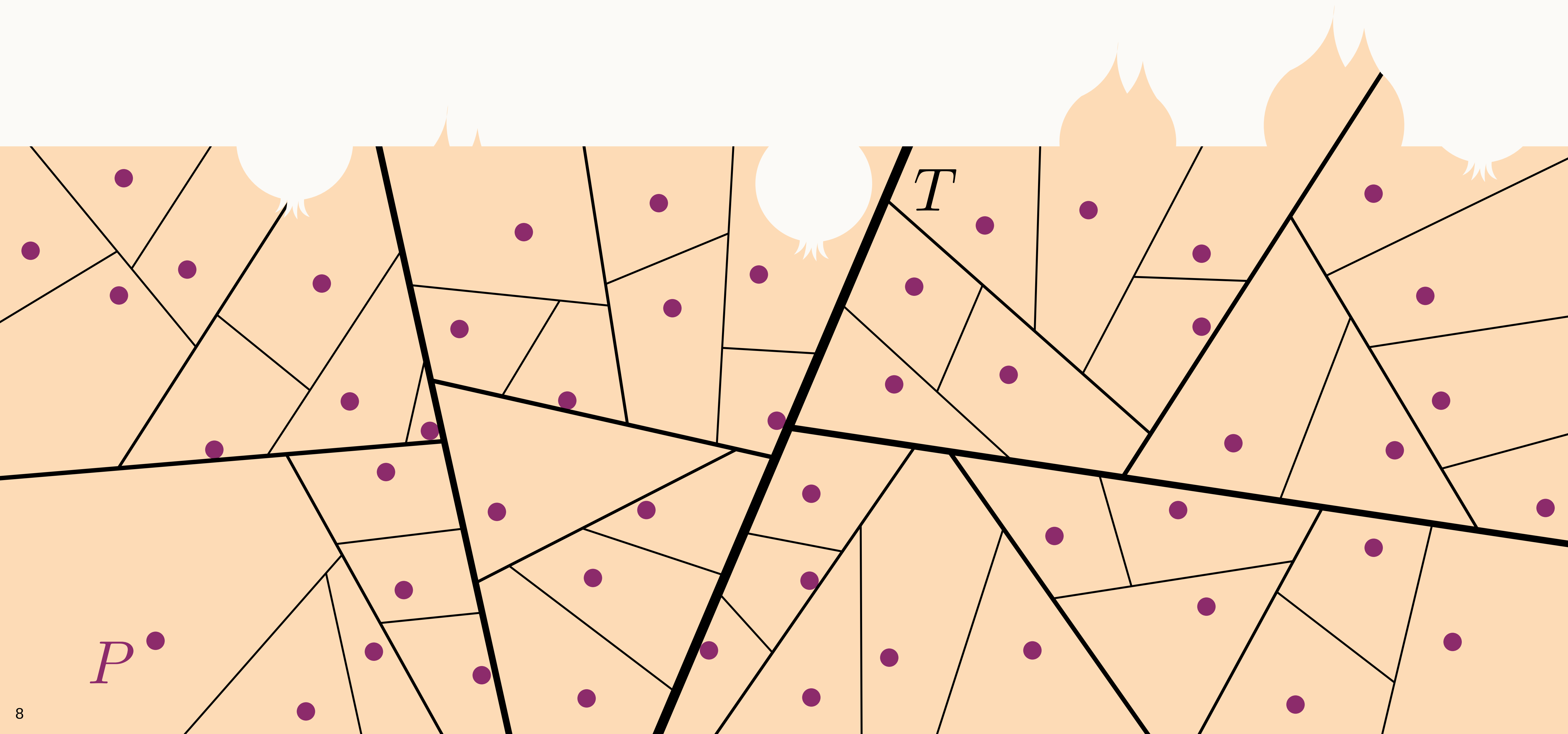
Remove any empty leaves of T.



RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

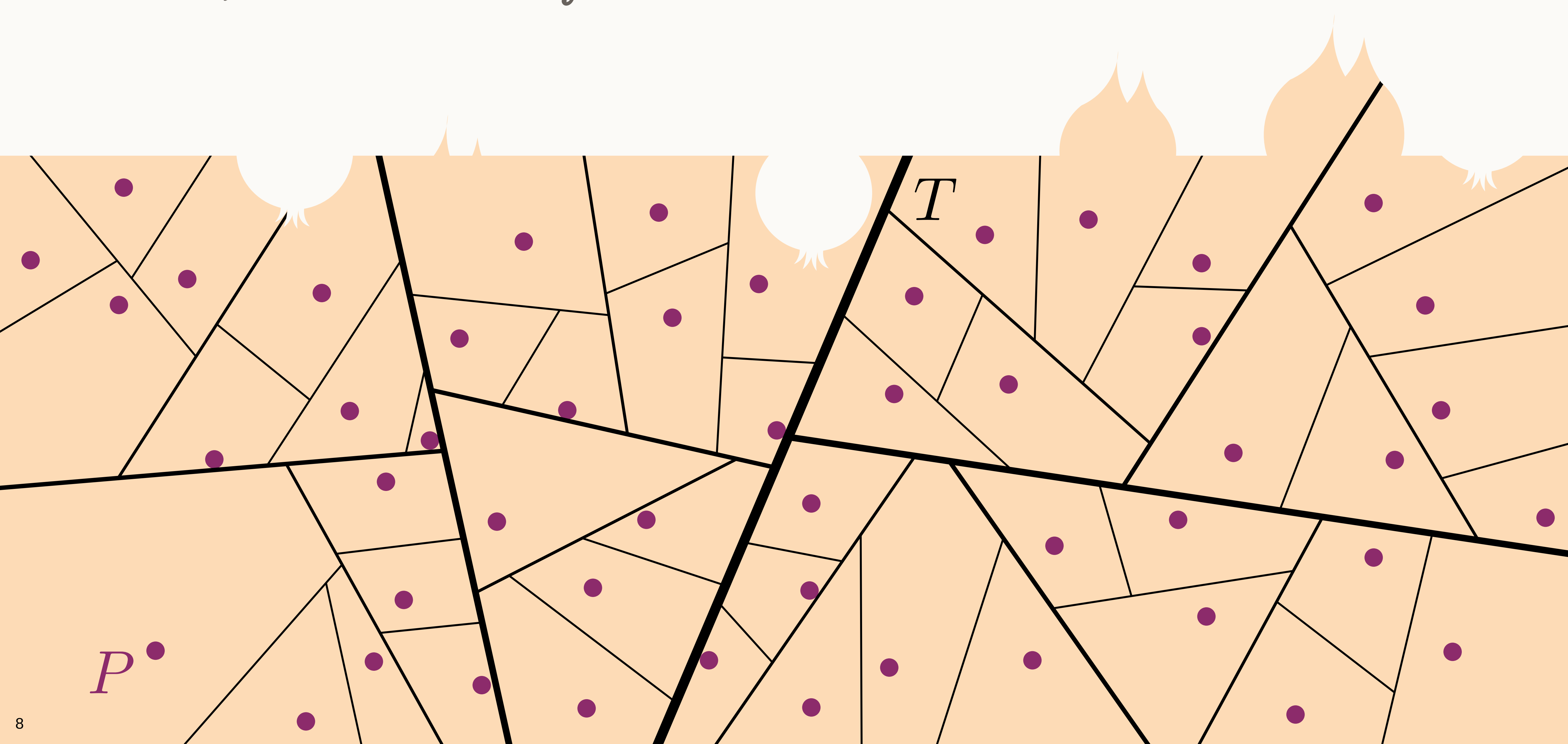


RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

Then, recursively unite the onions.

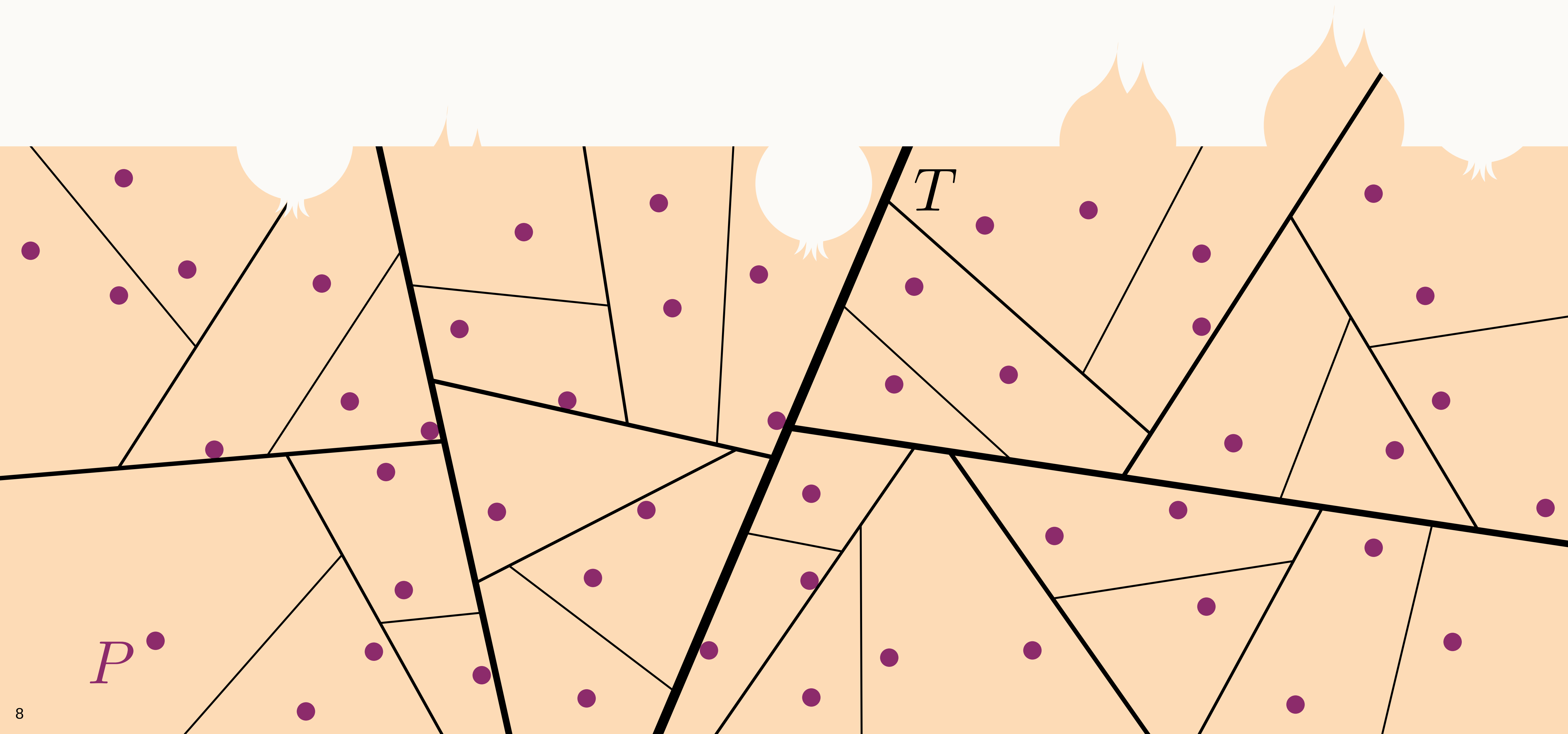


RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

Then, recursively unite the onions.

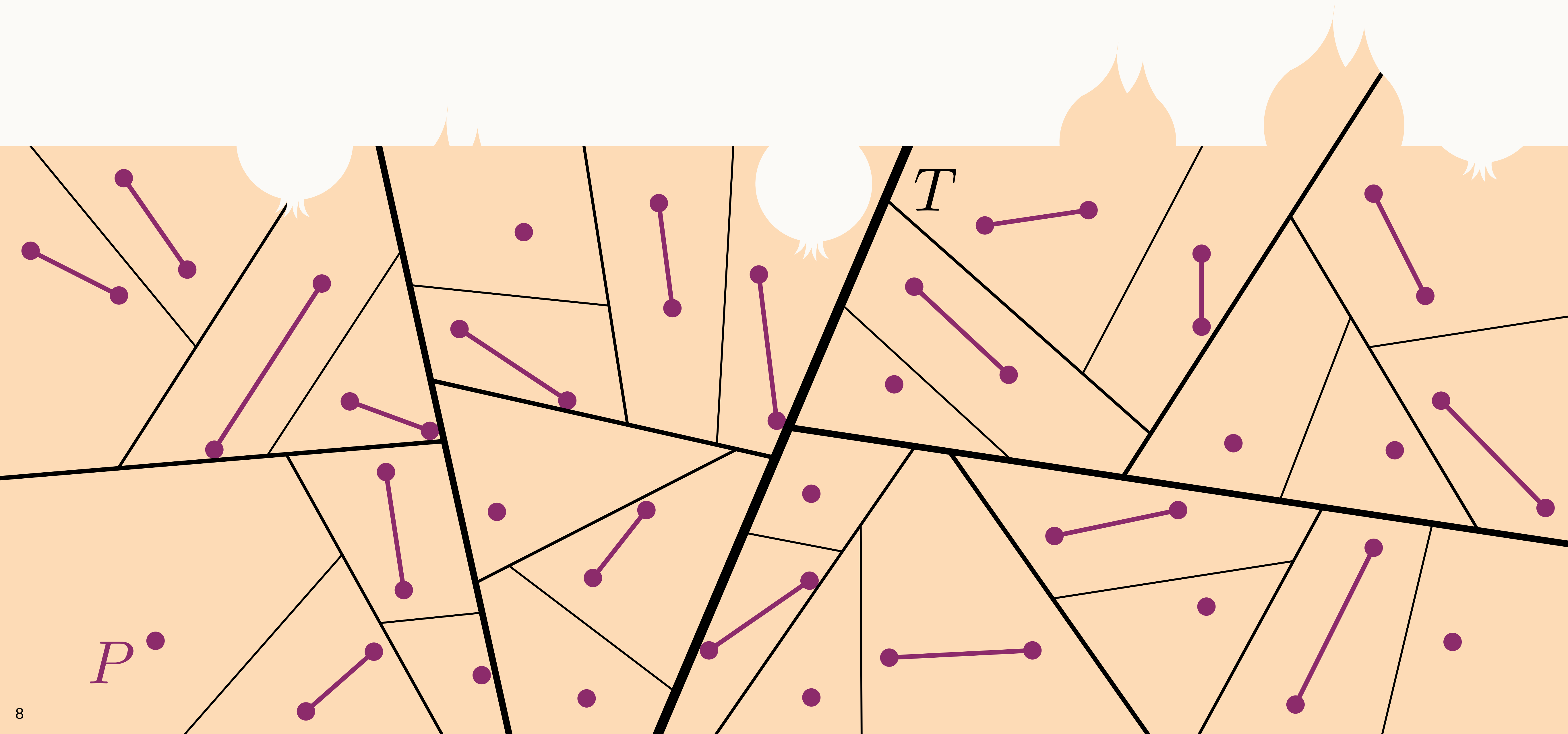


RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

Then, recursively unite the onions.

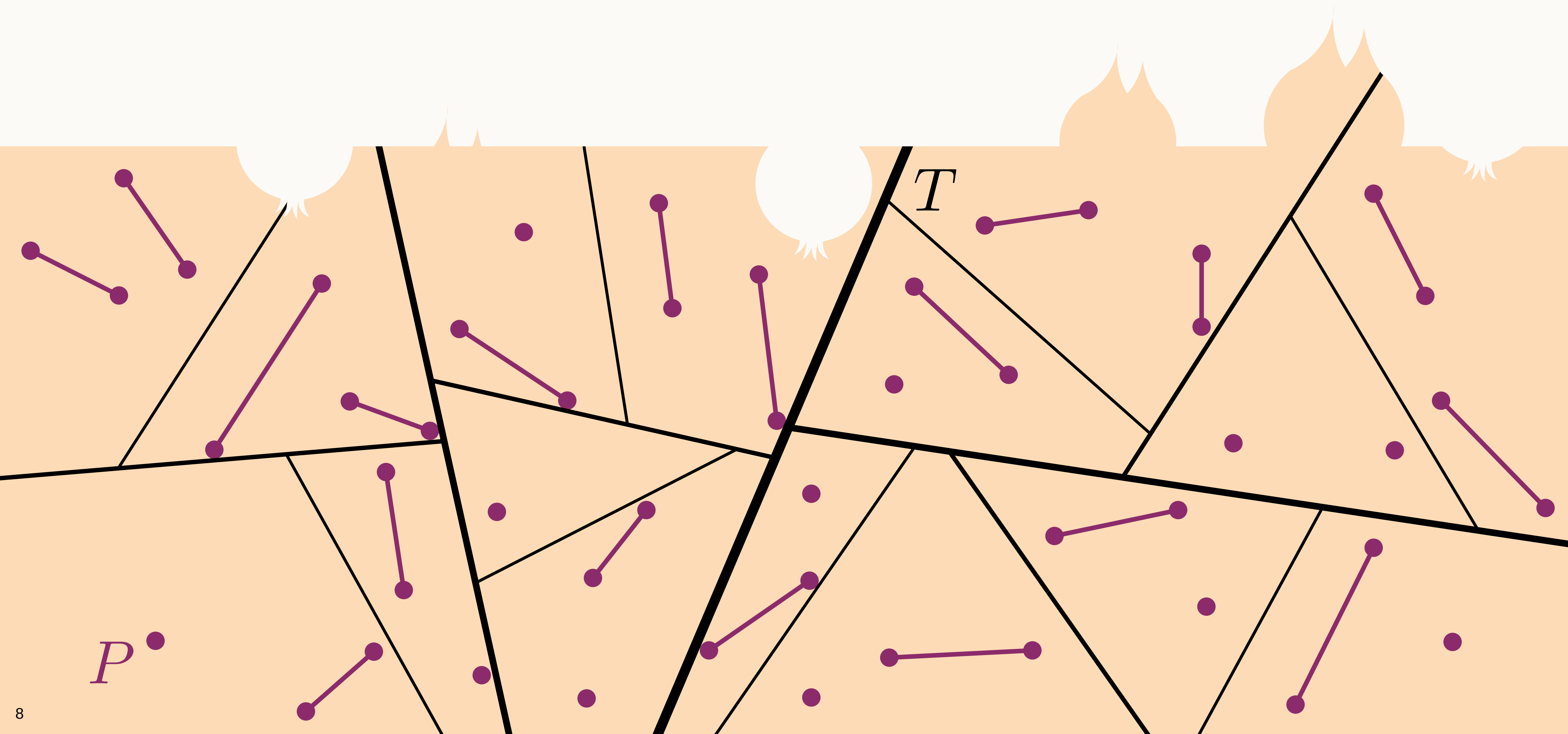


RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

Then, recursively unite the onions.

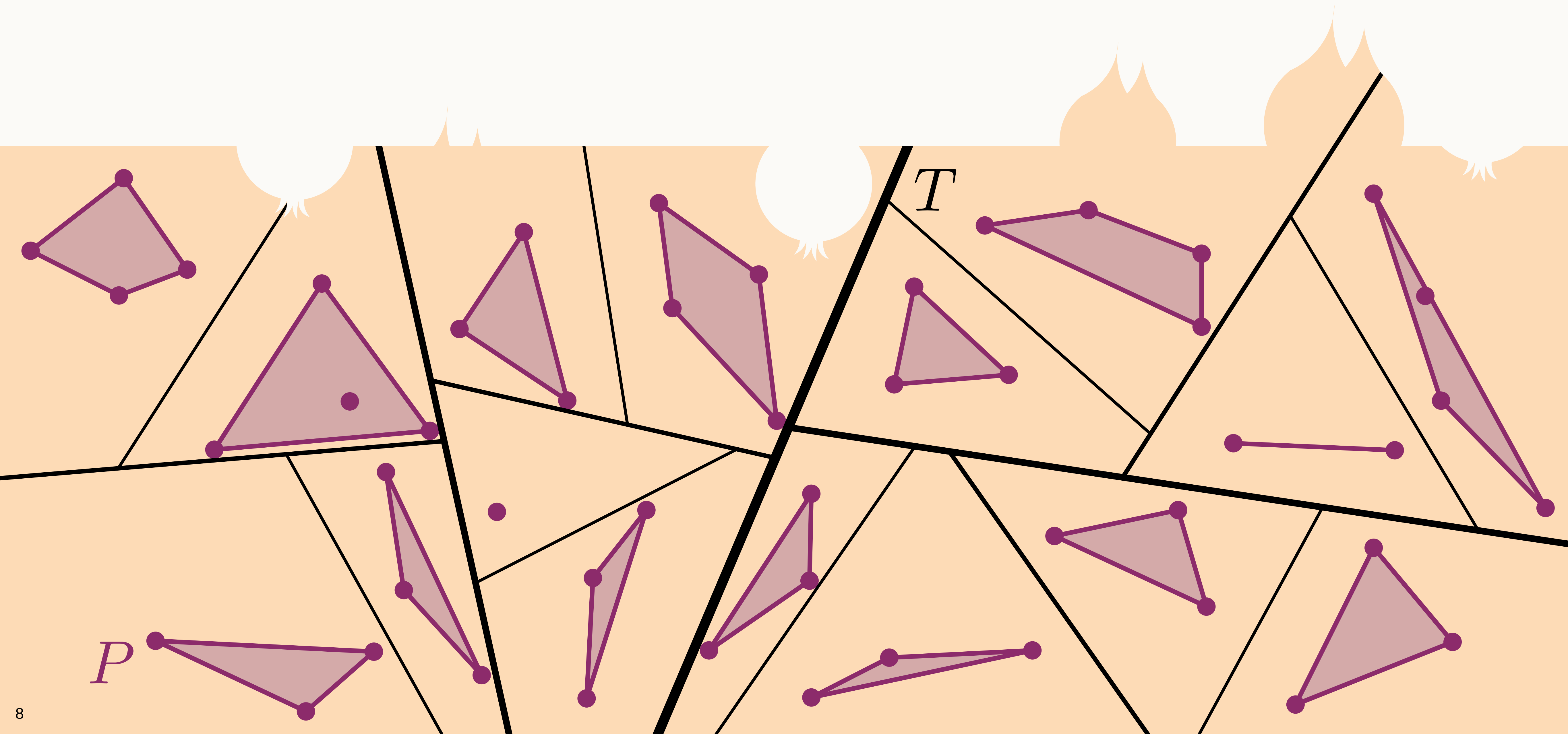


RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

Then, recursively unite the onions.

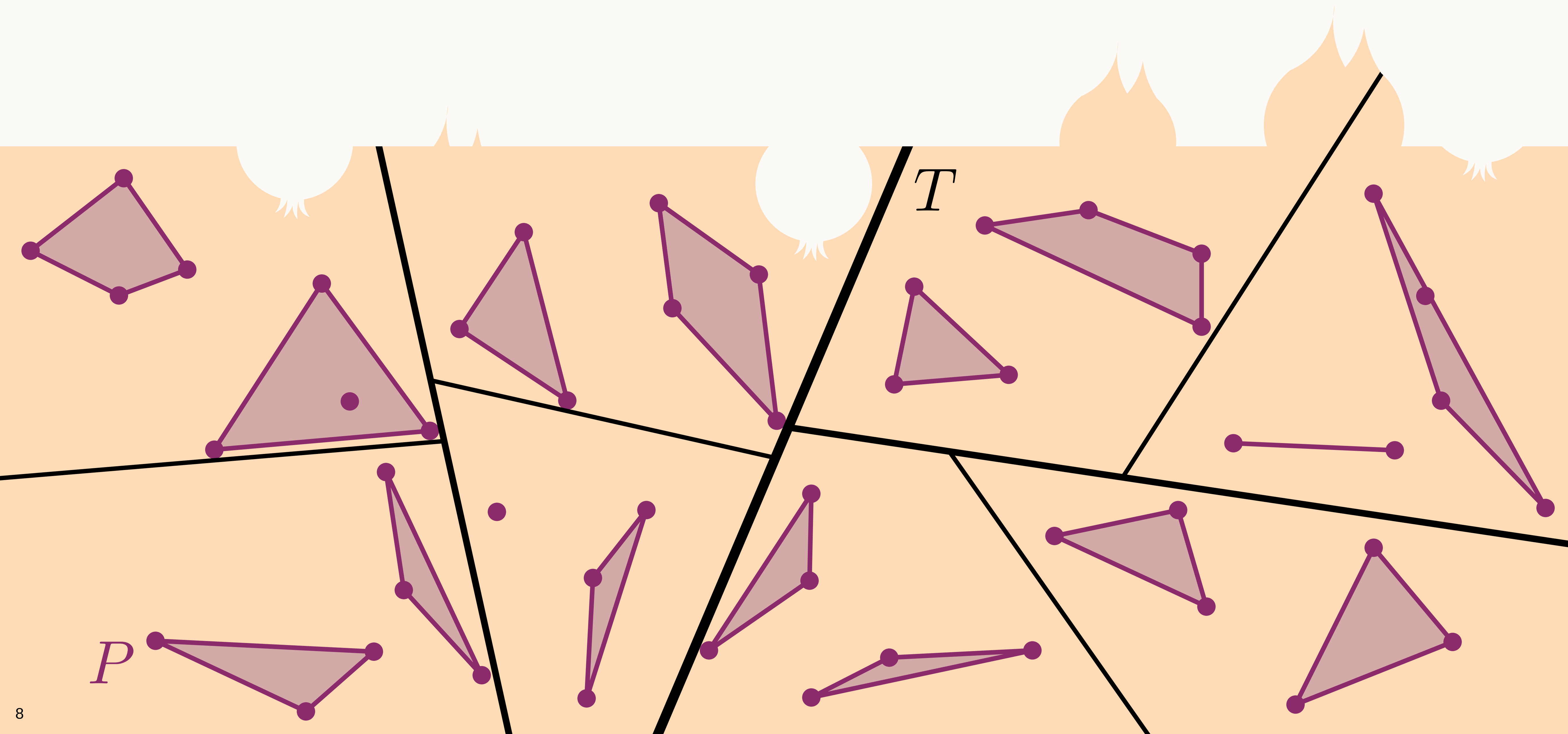


RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

Then, recursively unite the onions.

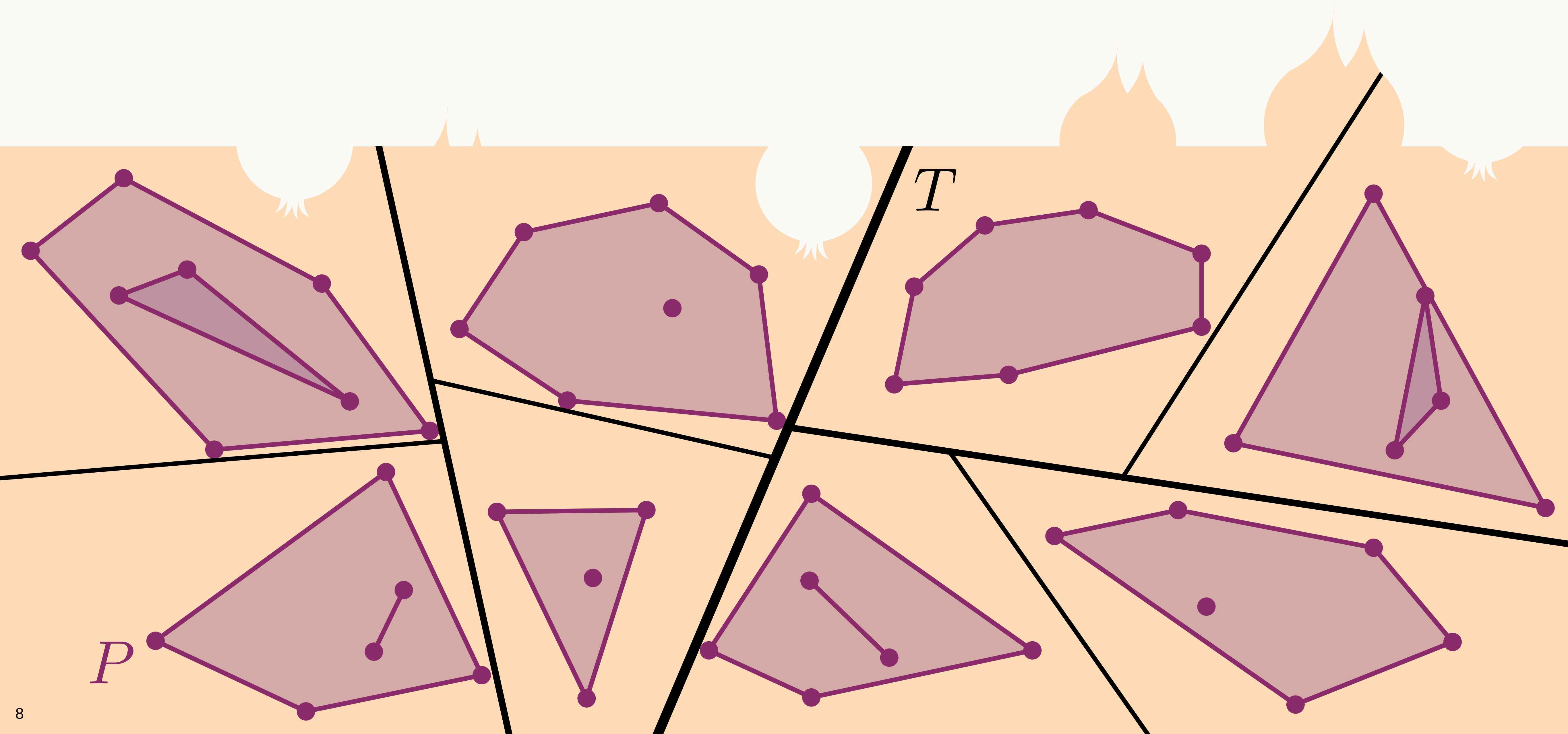


RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

Then, recursively unite the onions.

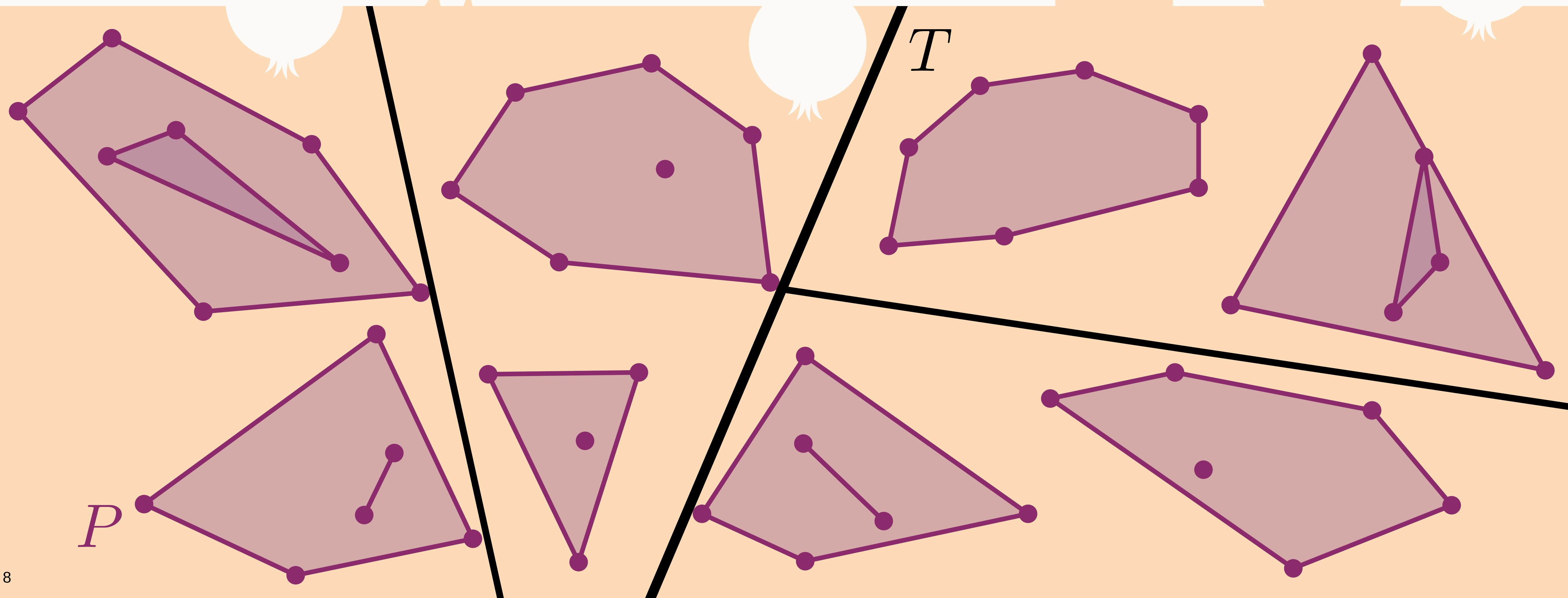


RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

Then, recursively unite the onions.

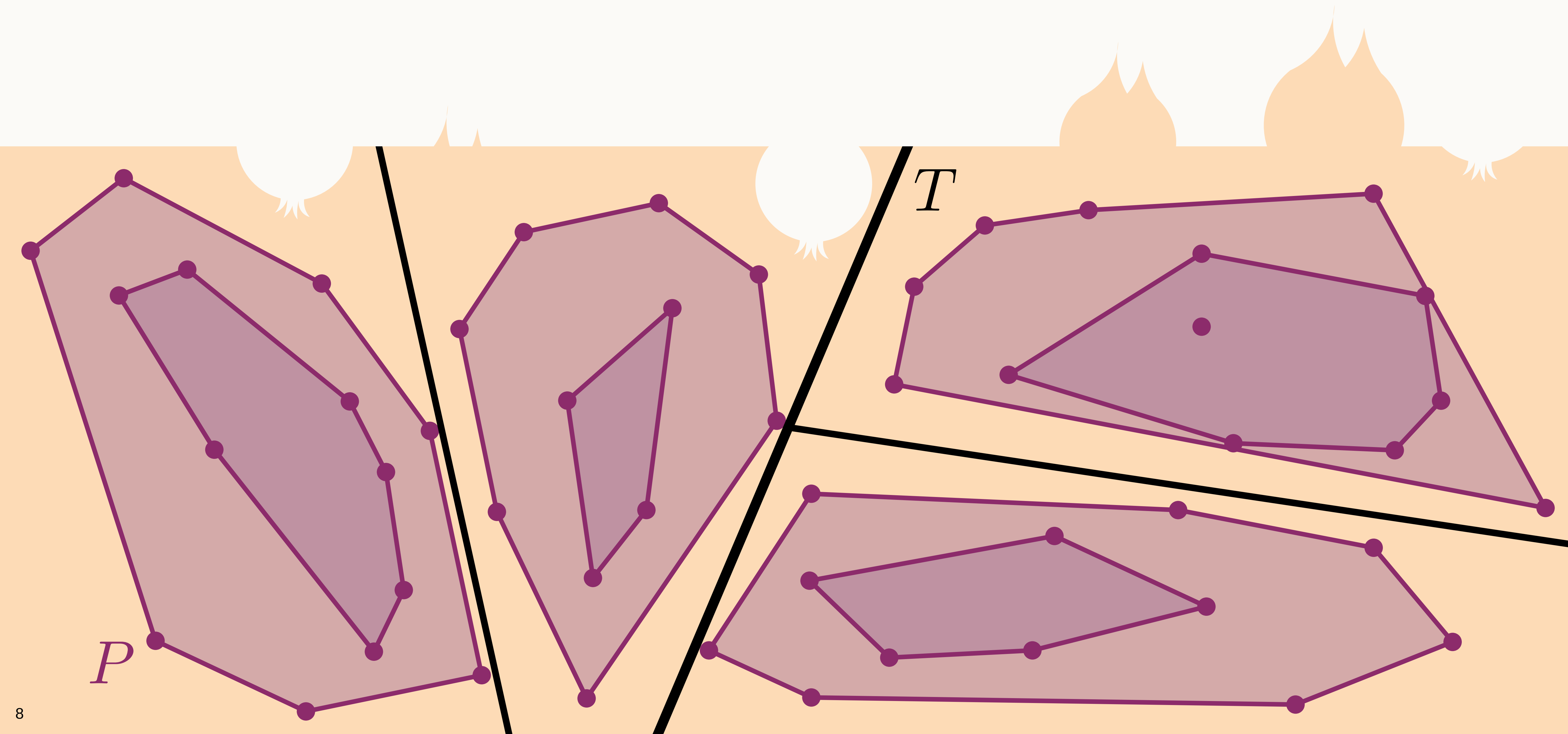


RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

Then, recursively unite the onions.

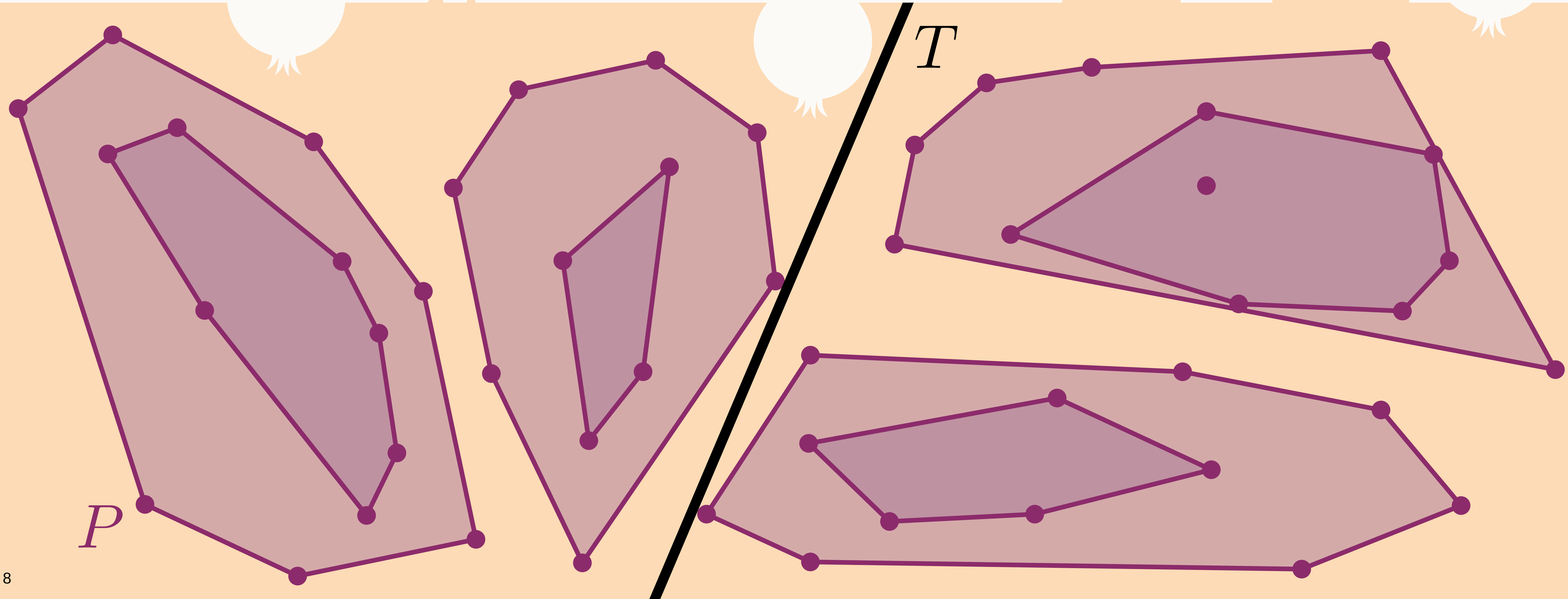


RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

Then, recursively unite the onions.

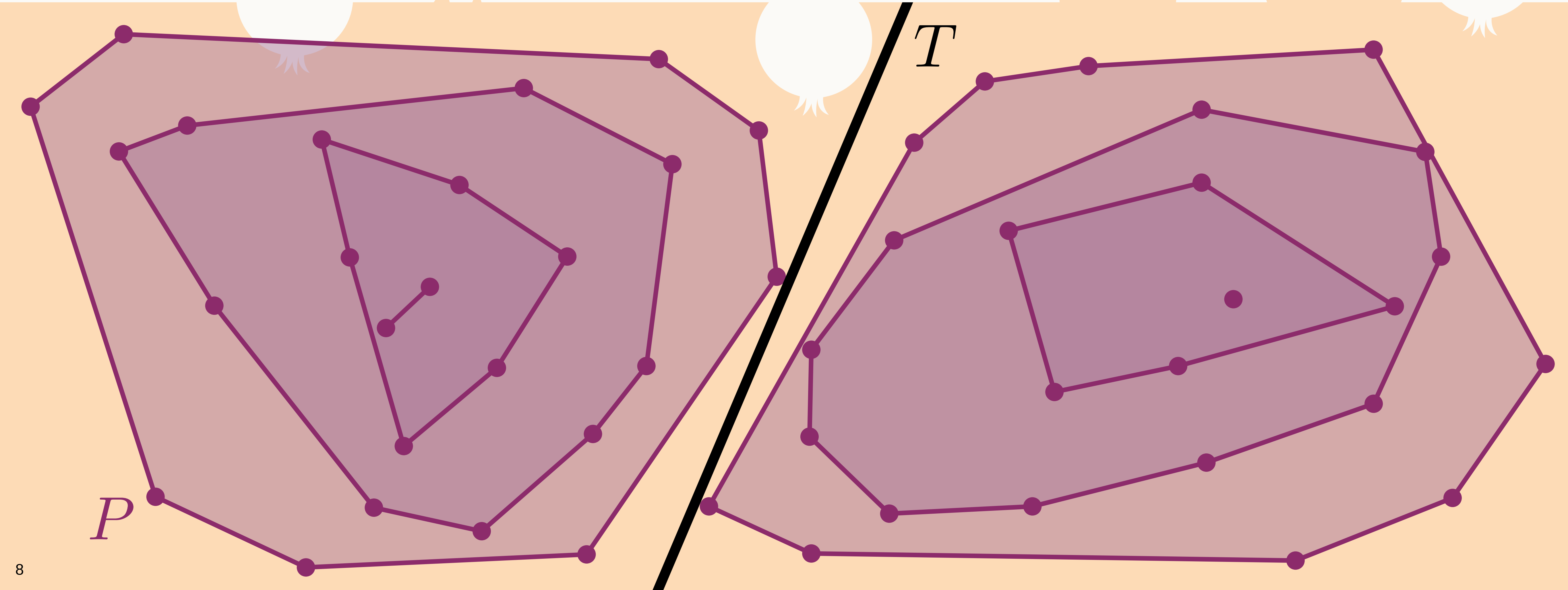


RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

Then, recursively unite the onions.

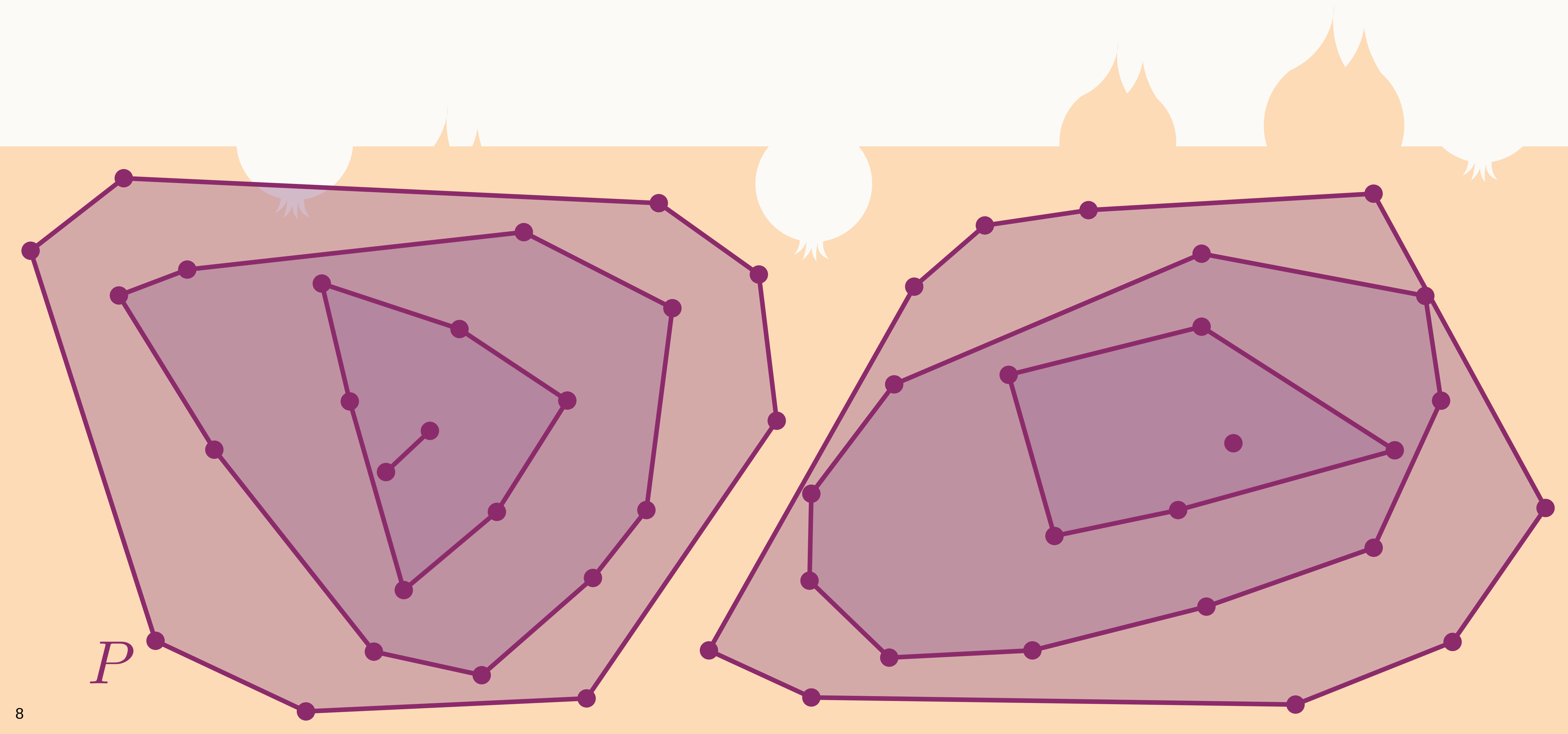


RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

Then, recursively unite the onions.

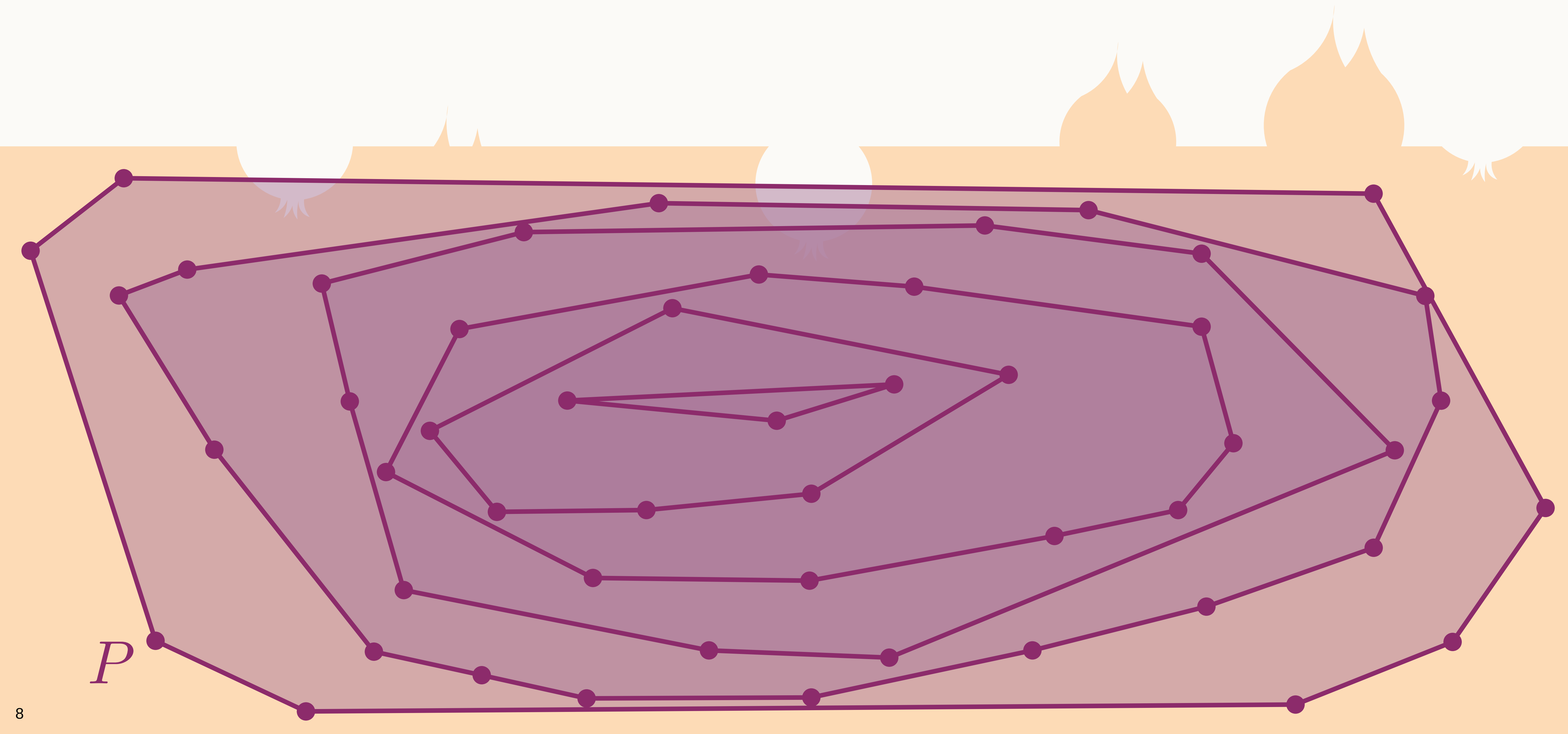


RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

Then, recursively unite the onions.



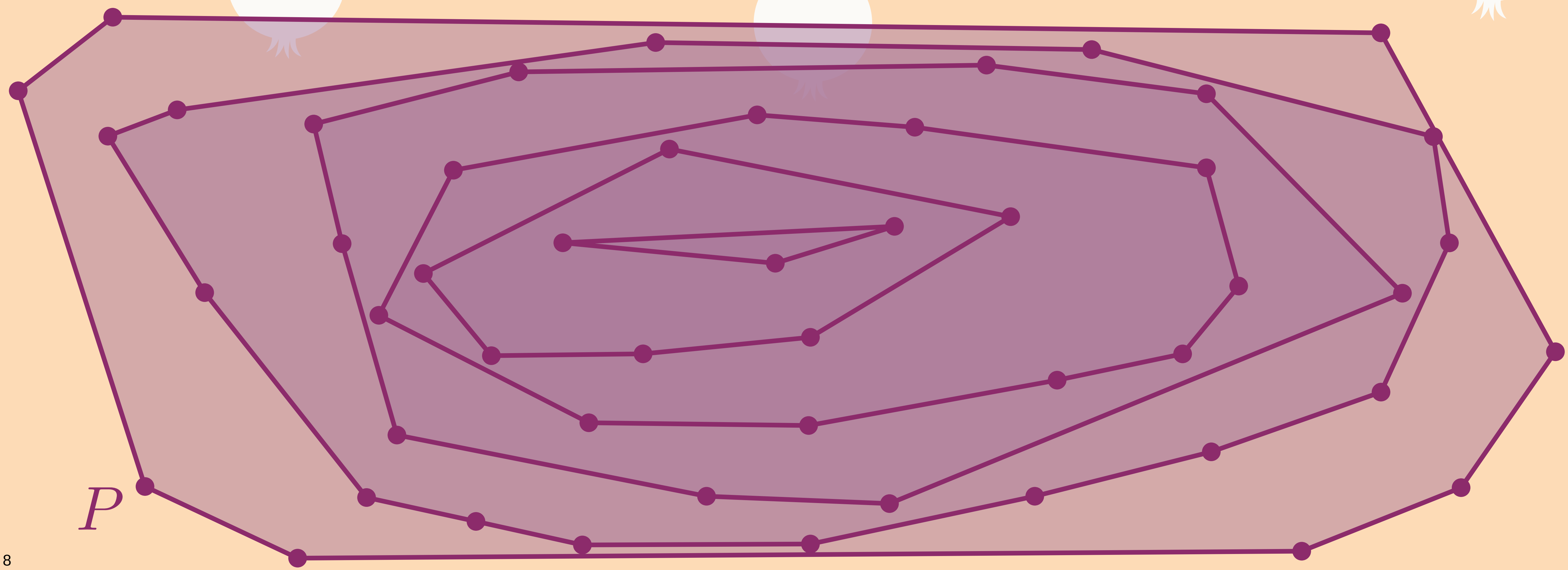
RECONSTRUCTION ALGORITHM

First, locate P in $\mathcal{R}$ and T.

Remove any empty leaves of T.

Then, recursively unite the onions.

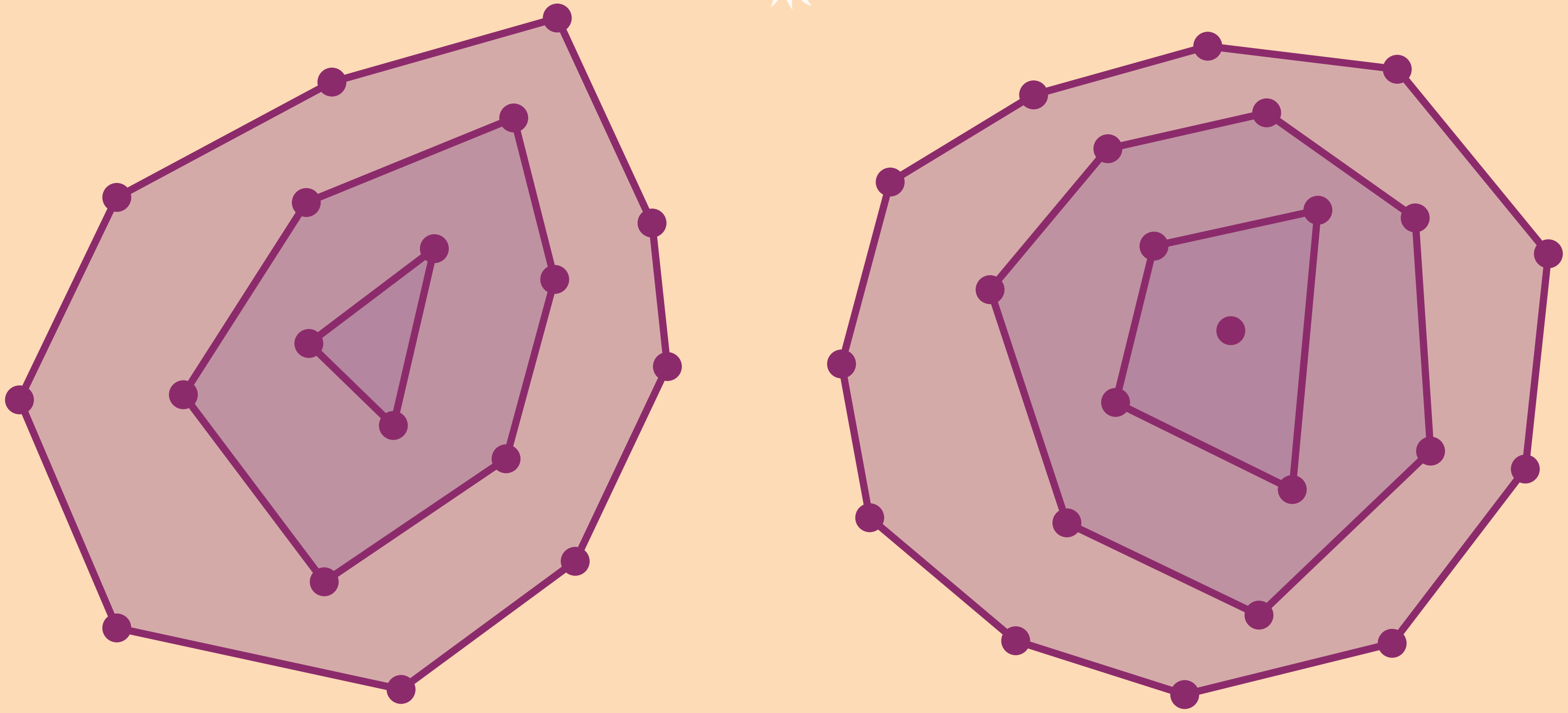
Done!



BUT!

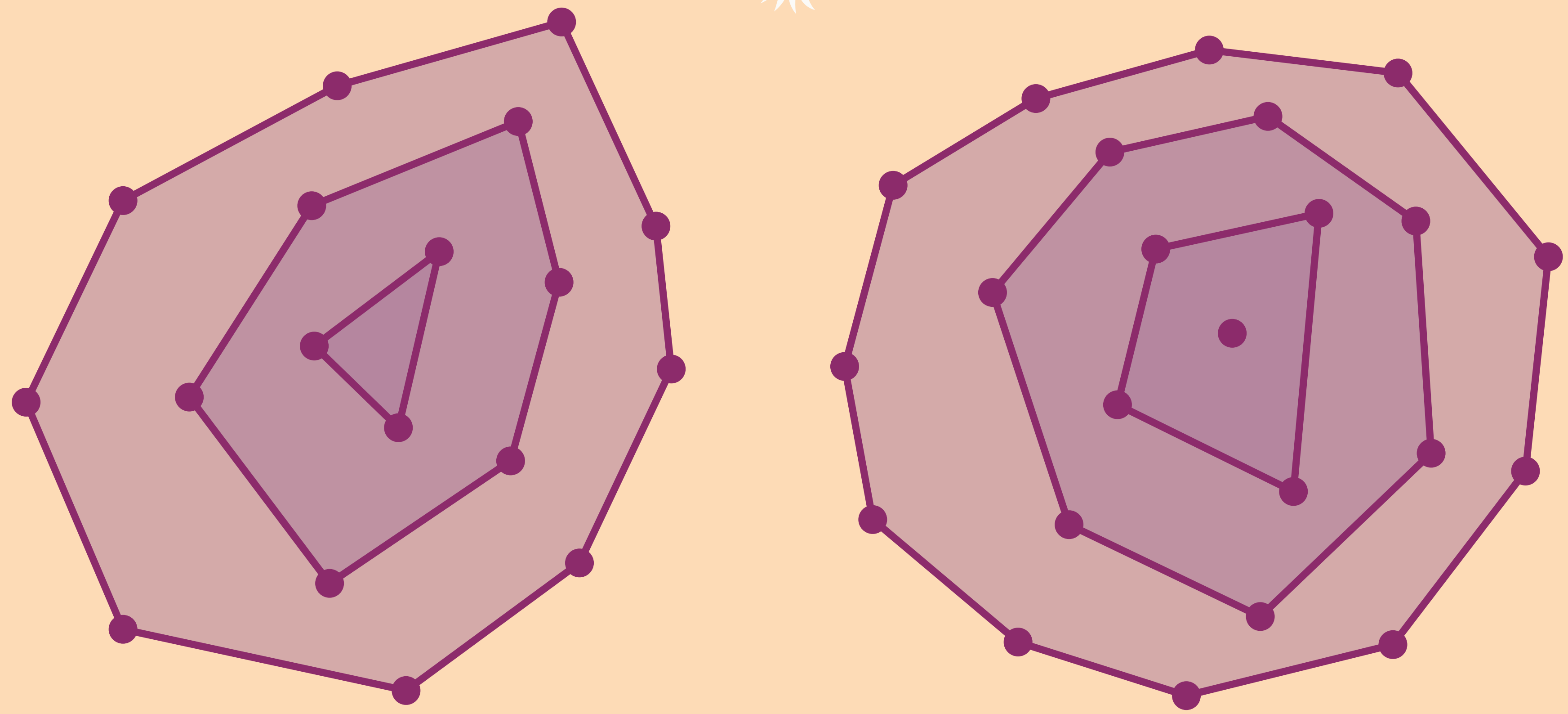
BUT! ...how do you unite two onions?

BUT! ...how do you unite two onions?



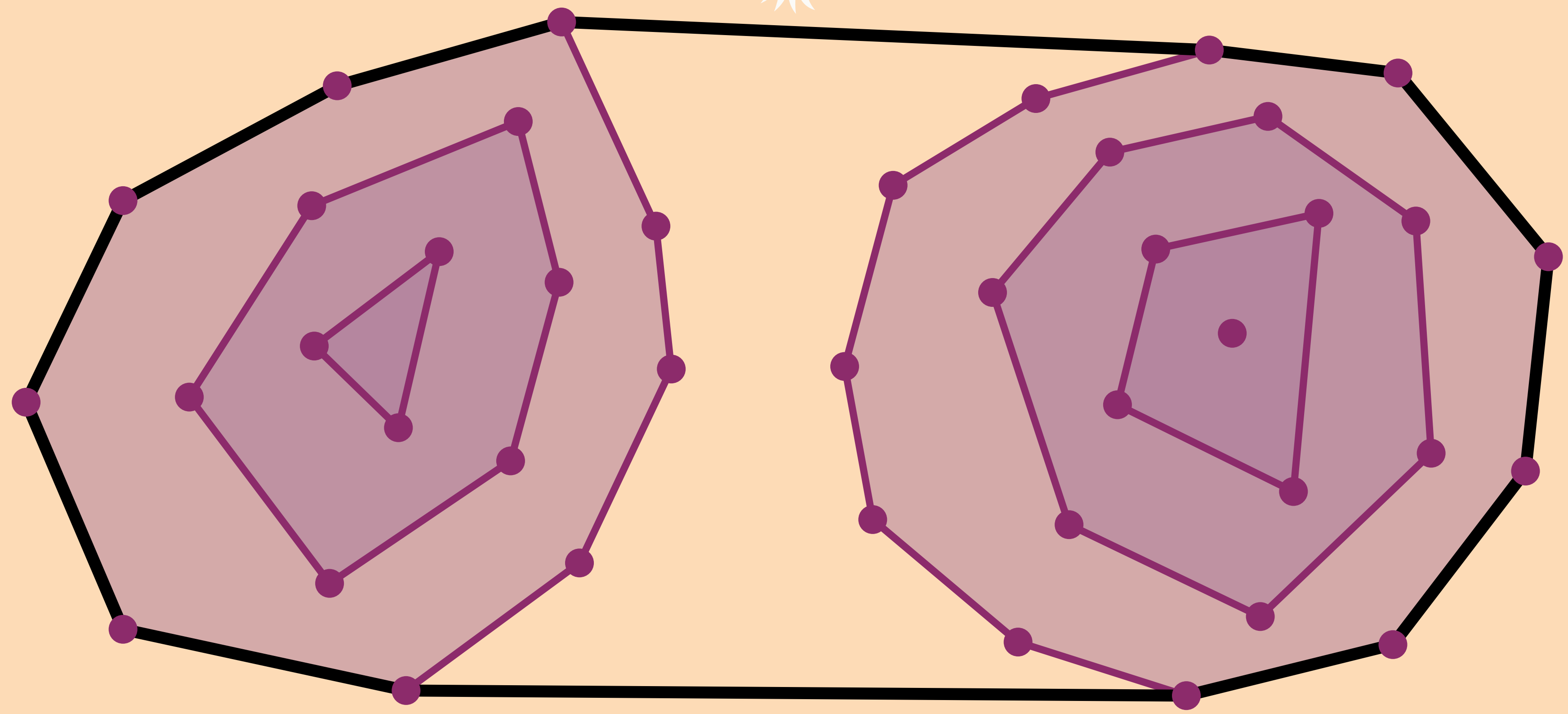
BUT! ...how do you unite two onions?

OBSERVATION: *We can find the convex hull of two onions in $O(\log n)$ time.*



BUT! ...how do you unite two onions?

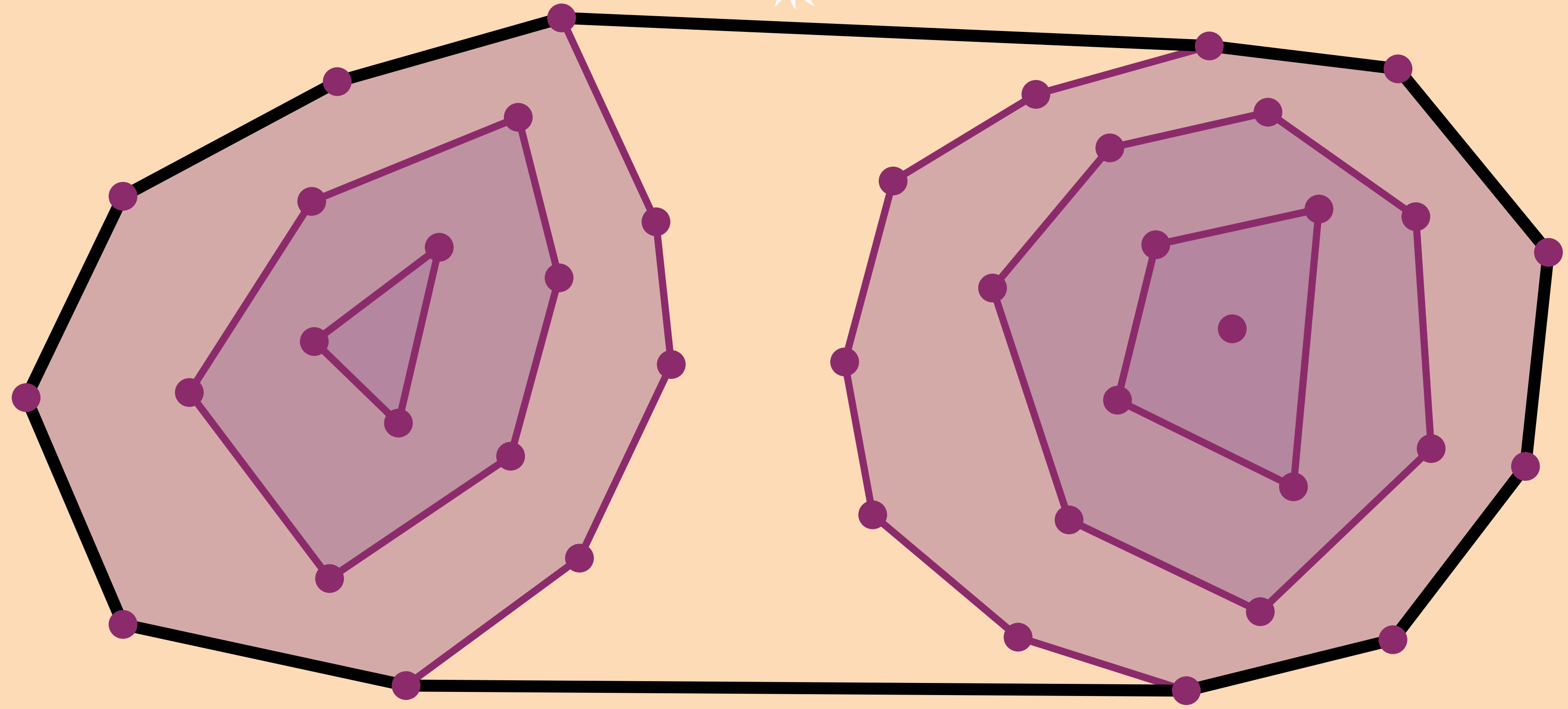
OBSERVATION: *We can find the convex hull of two onions in $O(\log n)$ time.*



BUT! ...how do you unite two onions?

OBSERVATION: *We can find the convex hull of two onions in $O(\log n)$ time.*

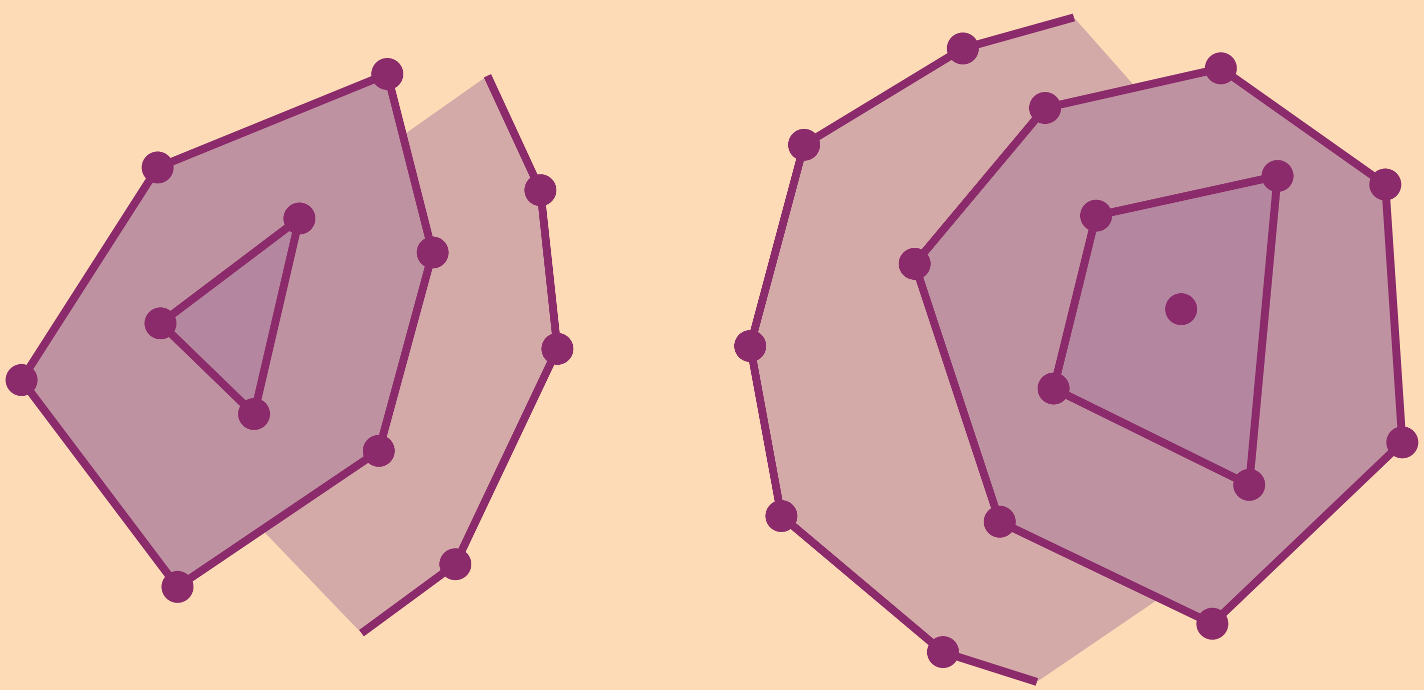
Let's remove it and recurse.



BUT! ...how do you unite two onions?

OBSERVATION: *We can find the convex hull of two onions in $O(\log n)$ time.*

Let's remove it and recurse.

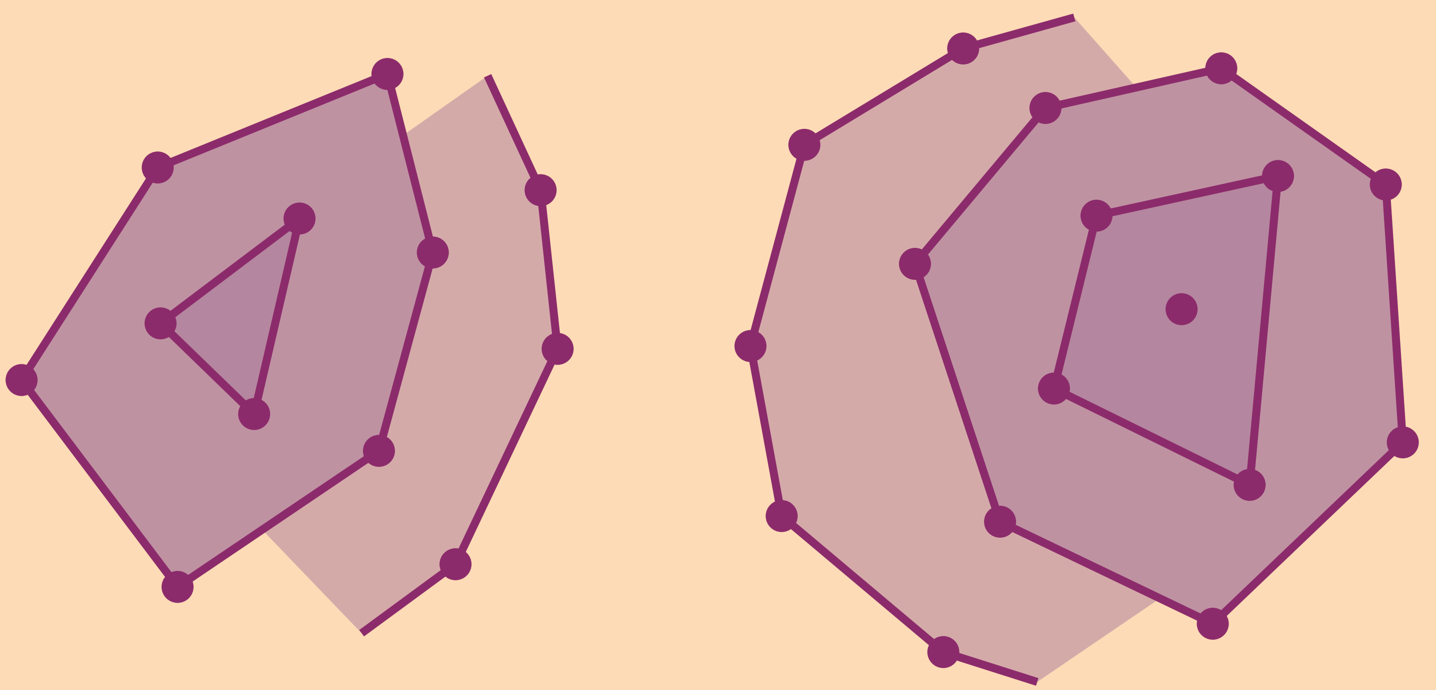


BUT! ...how do you unite two onions?

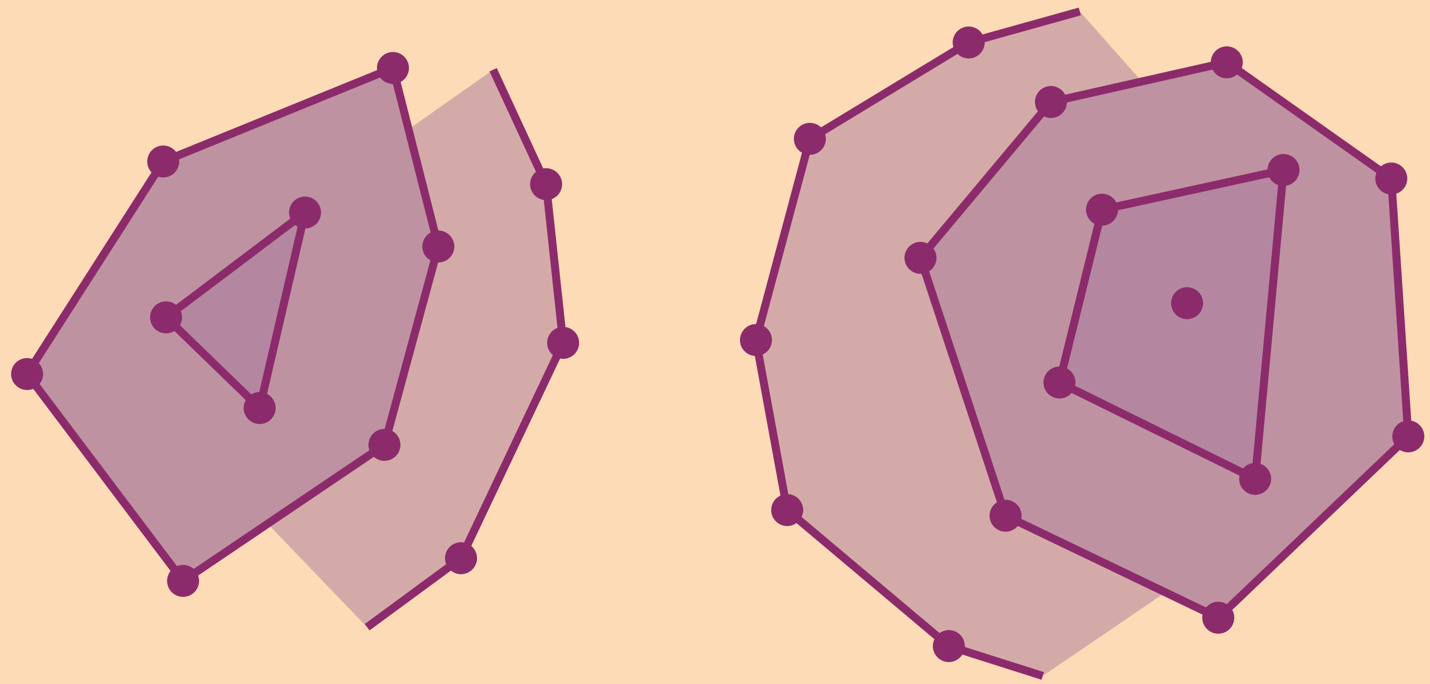
OBSERVATION: *We can find the convex hull of two onions in $O(\log n)$ time.*

Let's remove it and recurse.

Unfortunately, what is left are no longer proper onions...

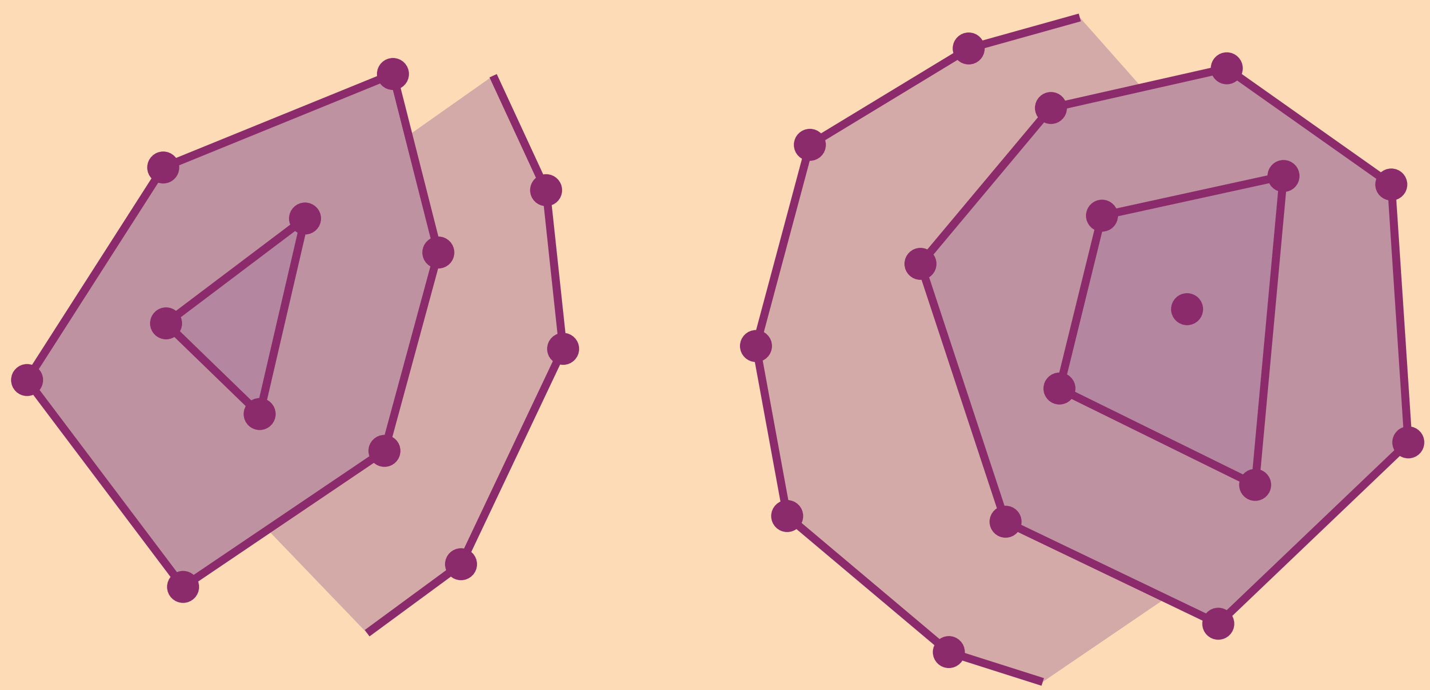


ONION RESTAURATION & UNIFICATION



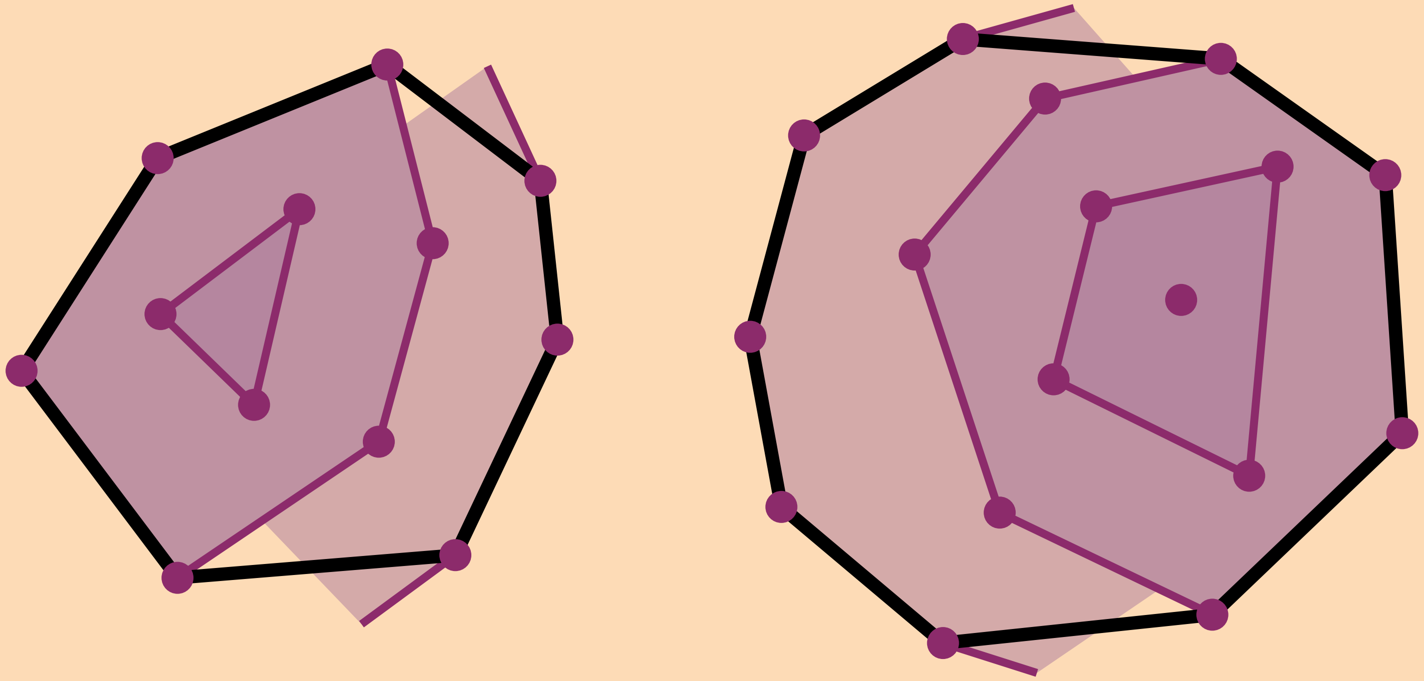
ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.



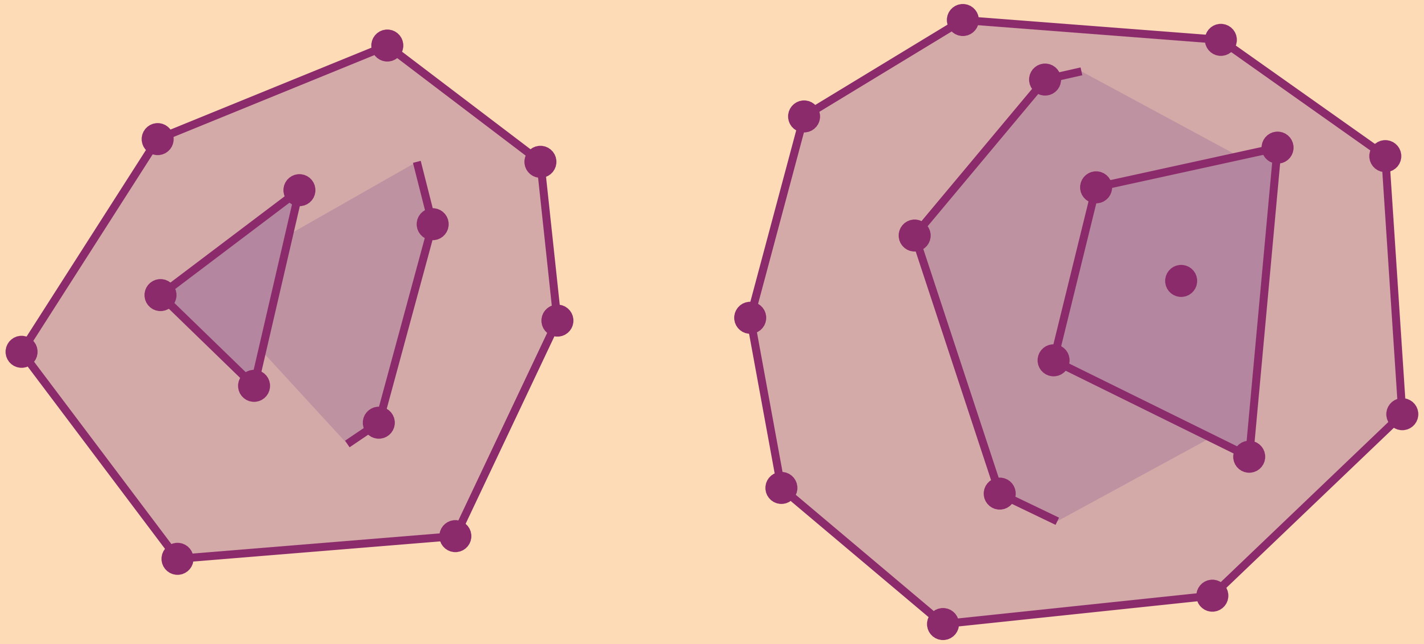
ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.



ONION RESTAURATION & UNIFICATION

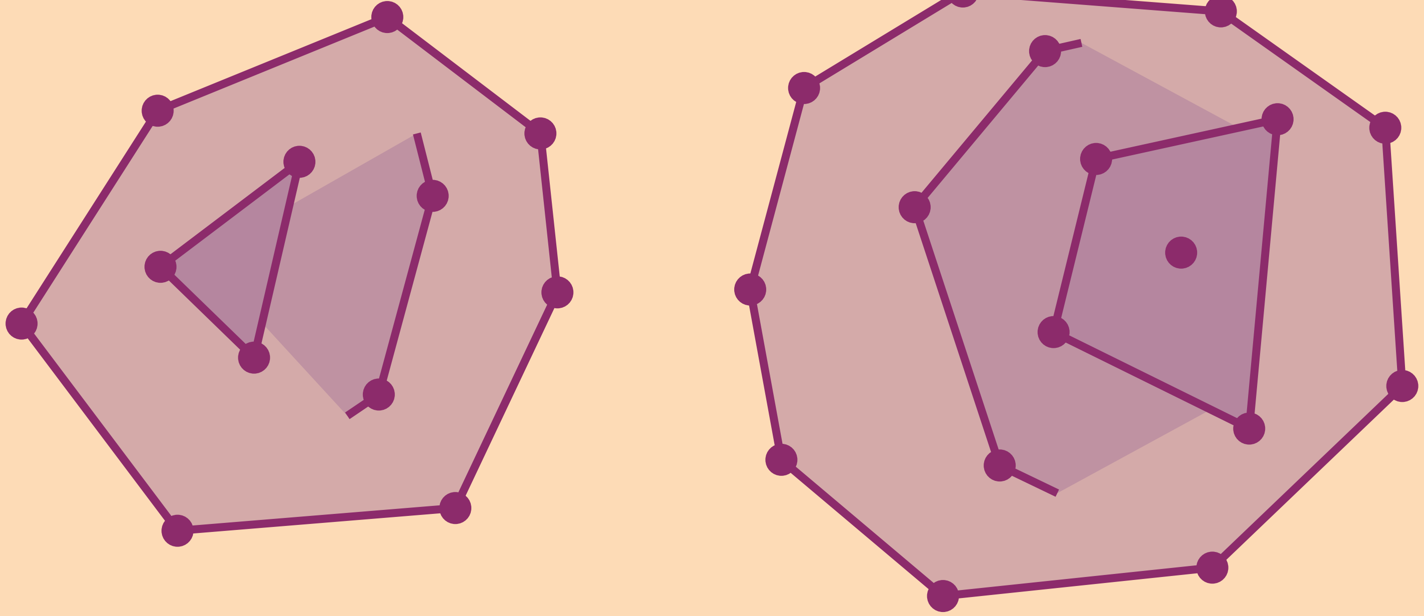
Compute convex hulls of the individual onions.



ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

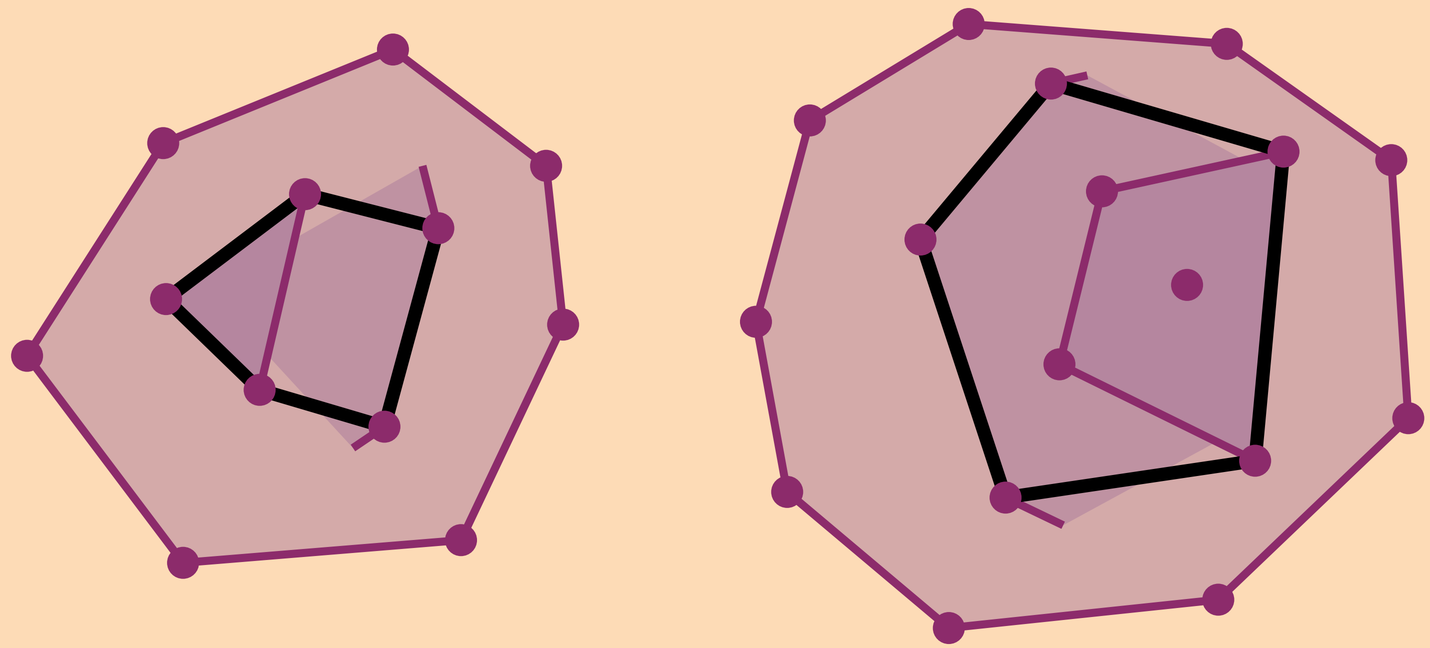
Recurse once again.



ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

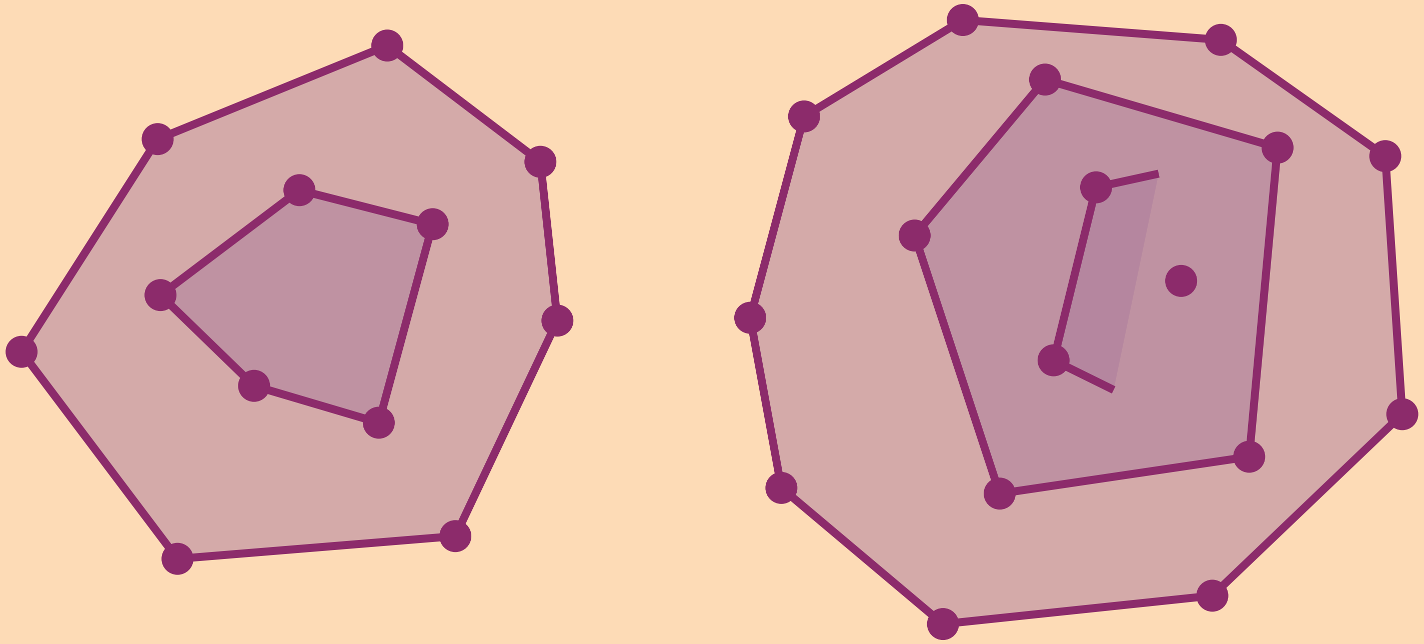
Recurse once again.



ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

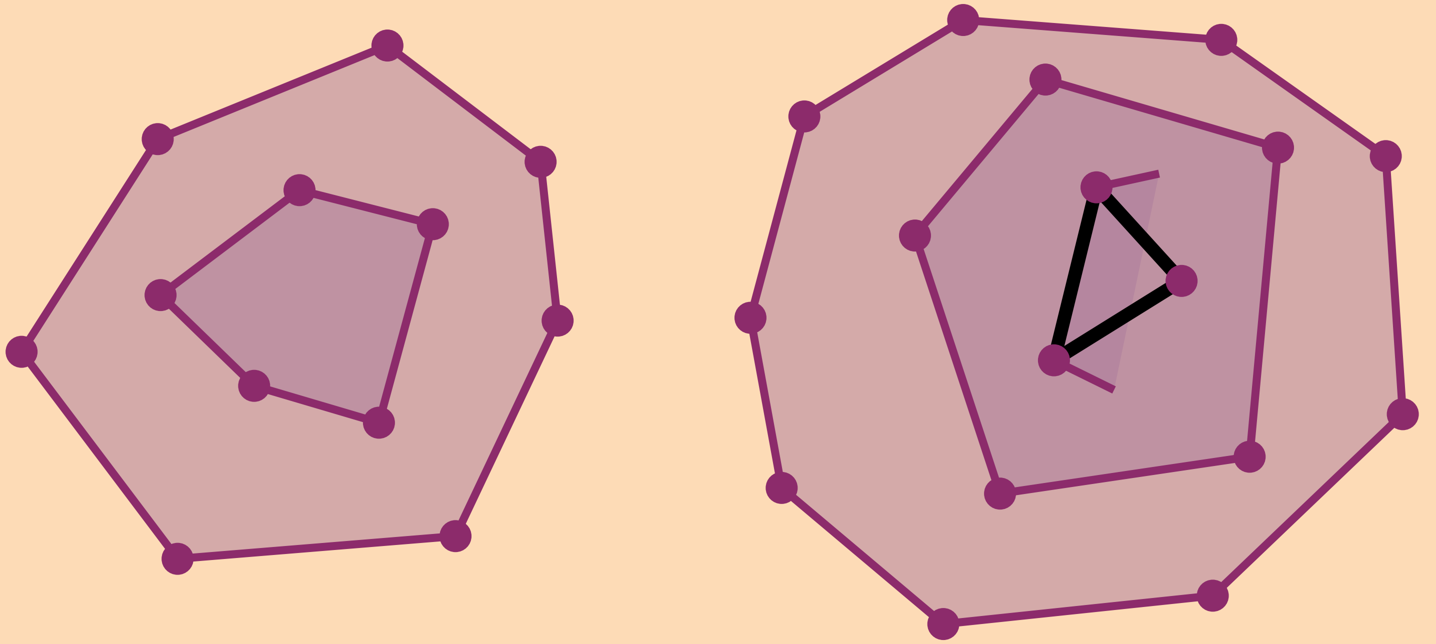
Recurse once again.



ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

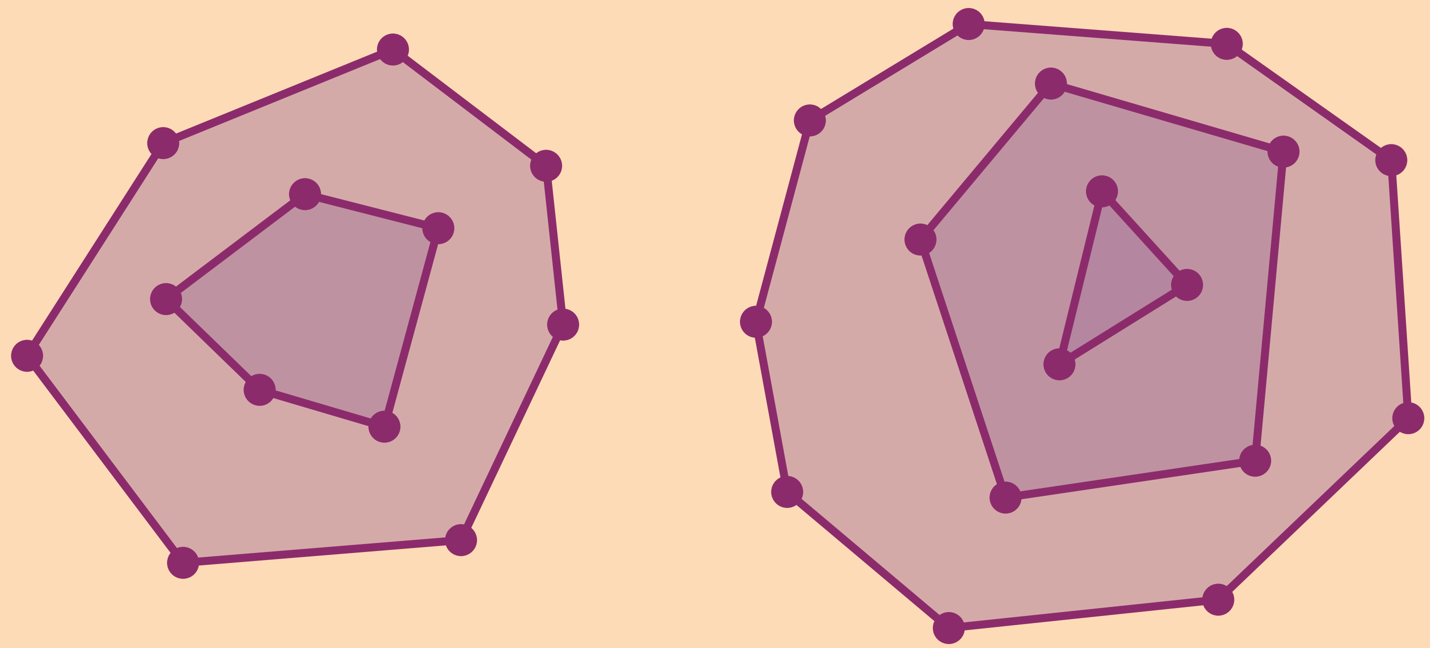
Recurse once again.



ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

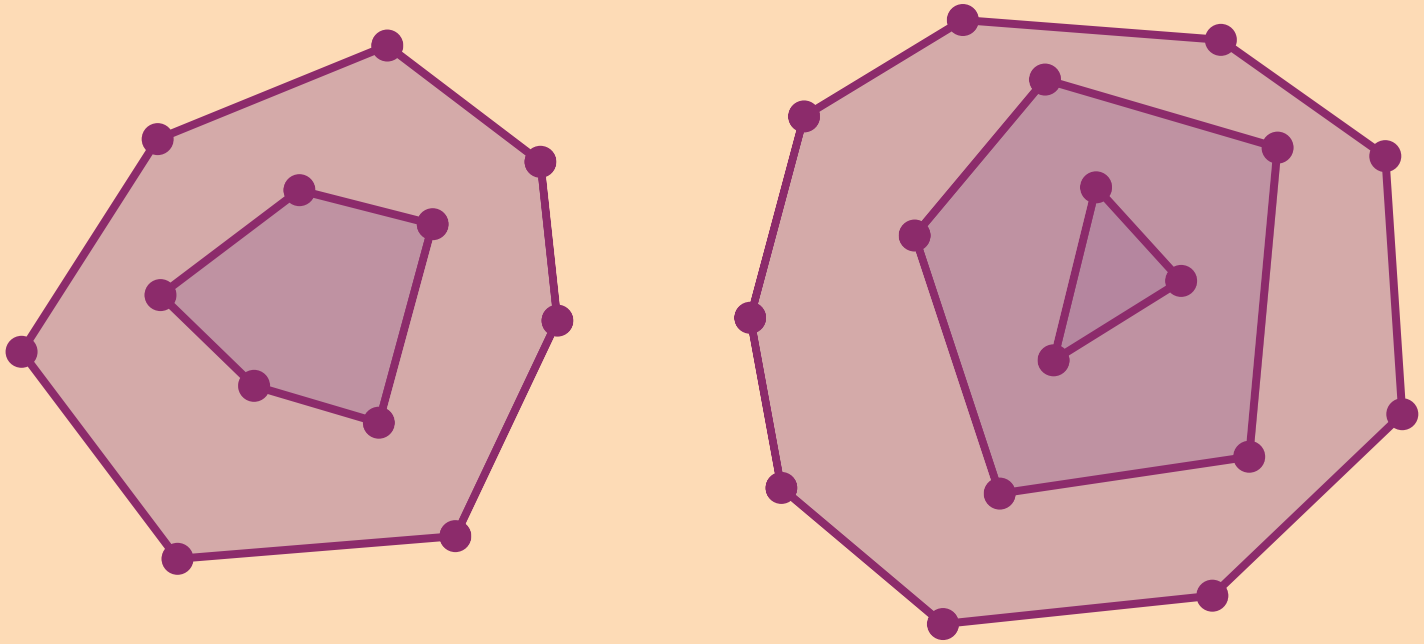


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

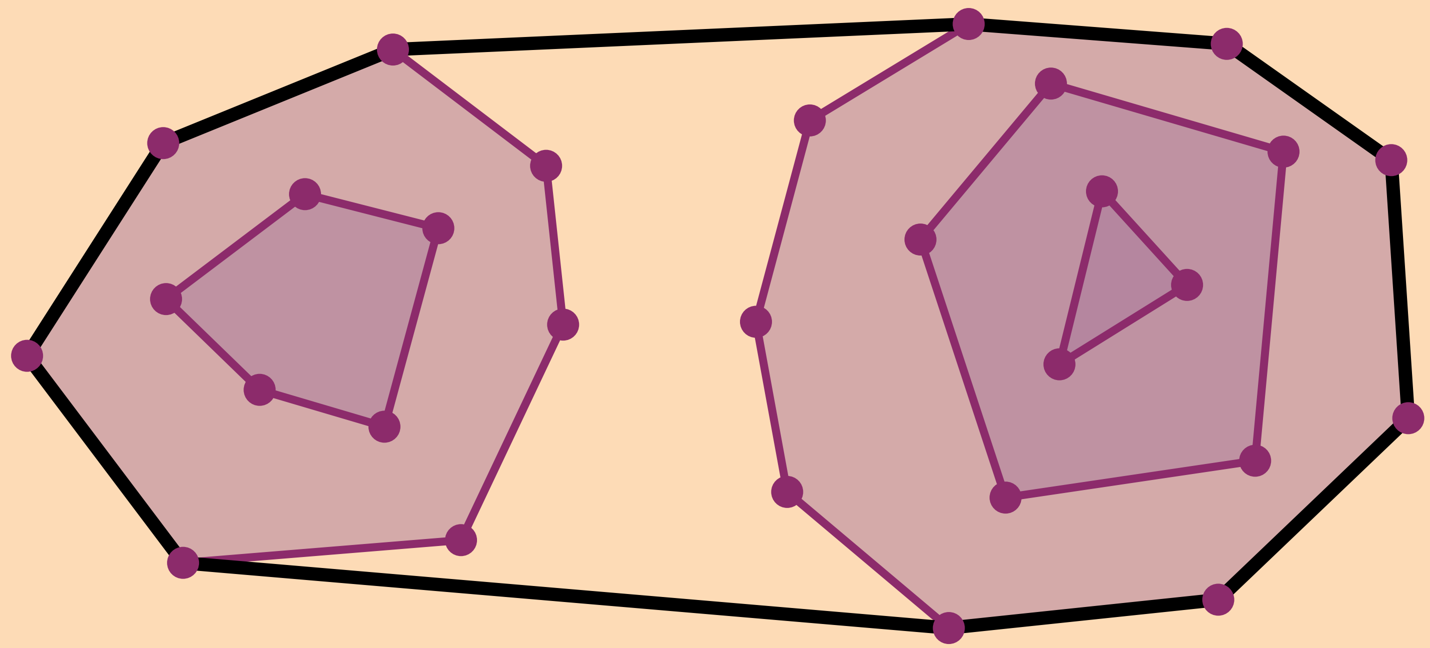


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

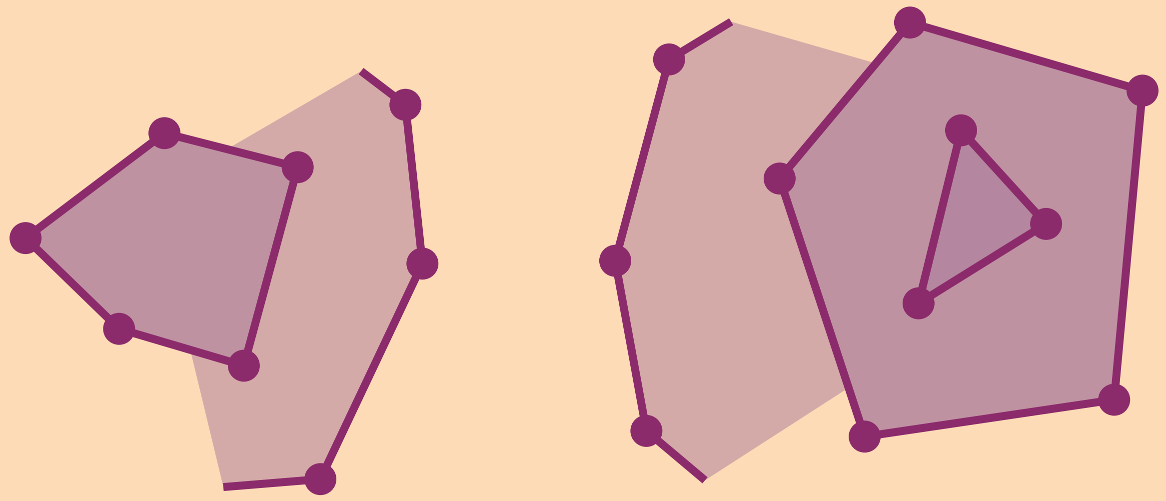


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

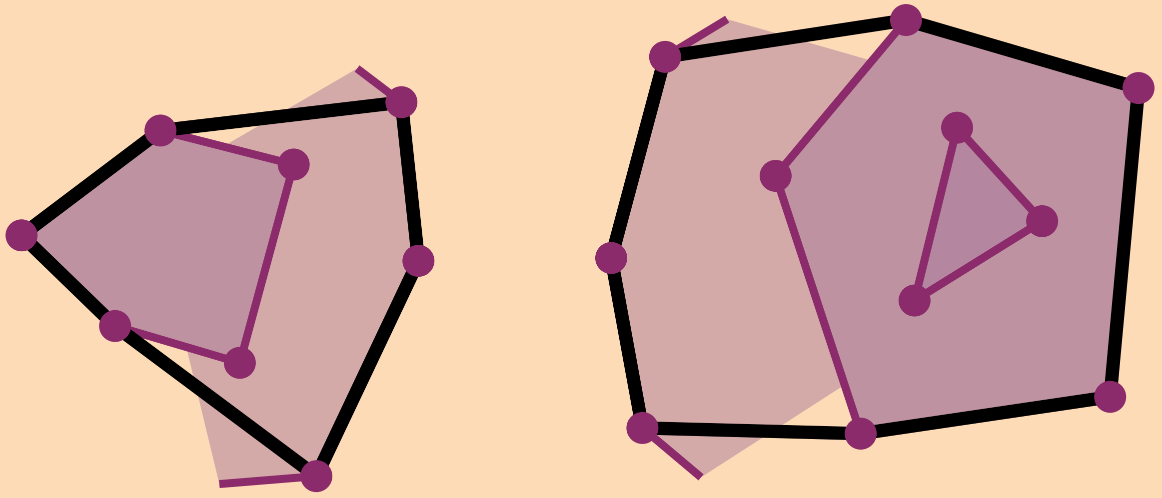


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

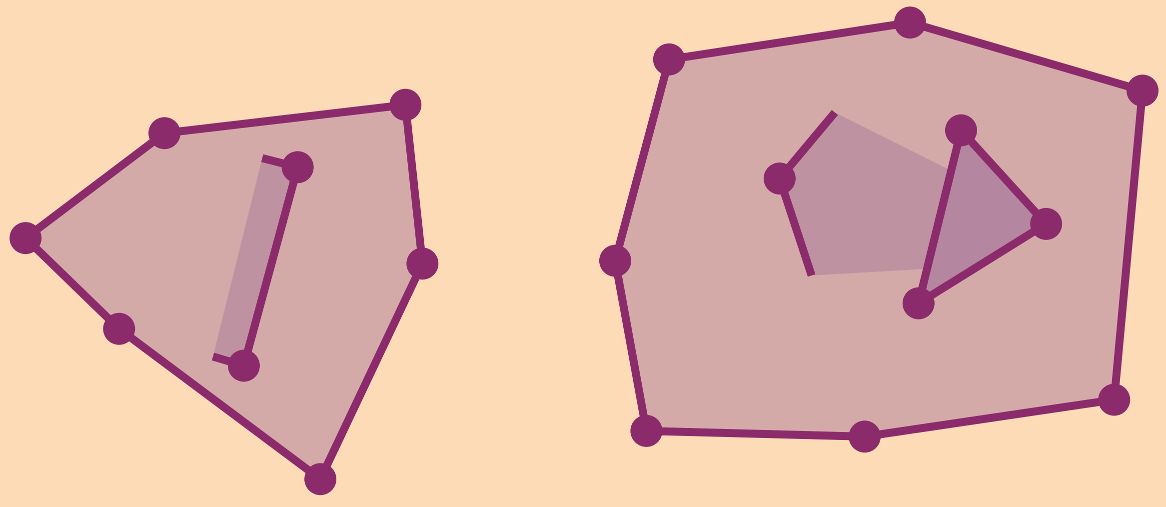


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

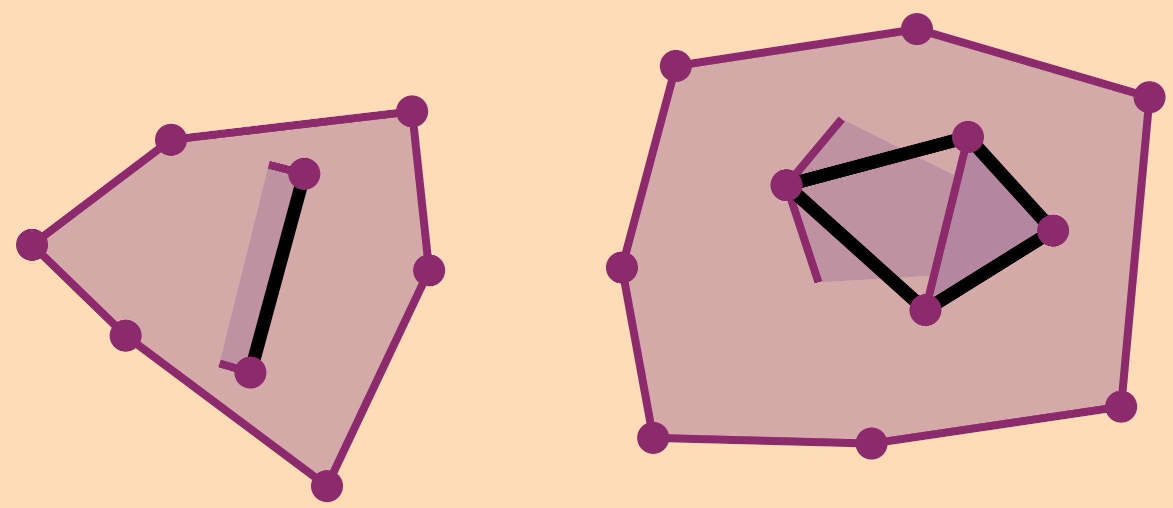


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

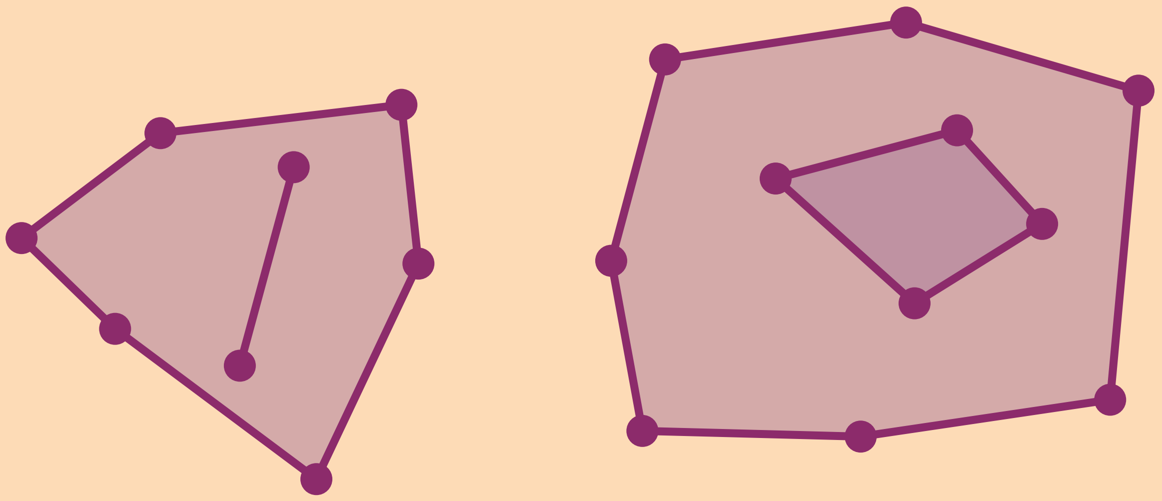


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

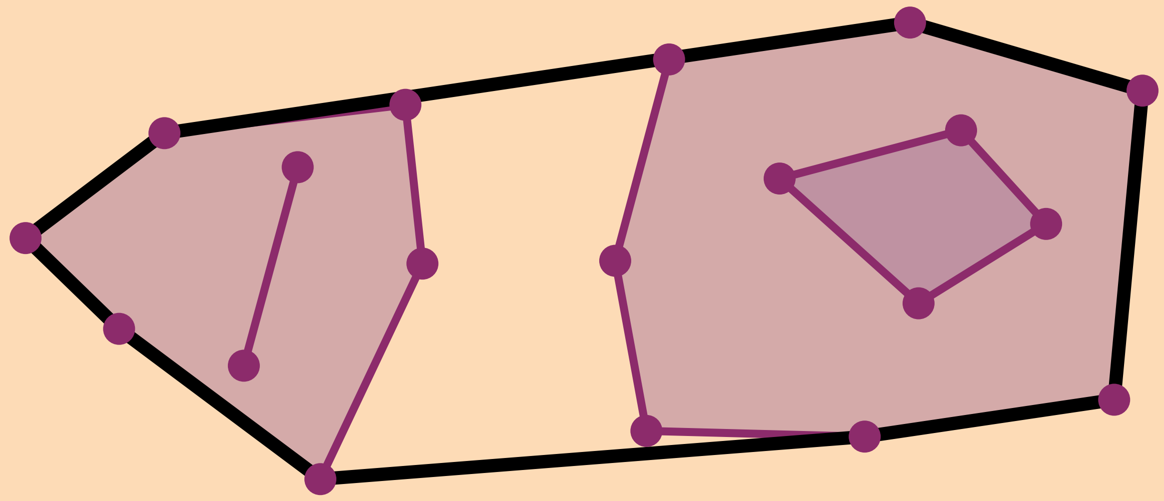


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

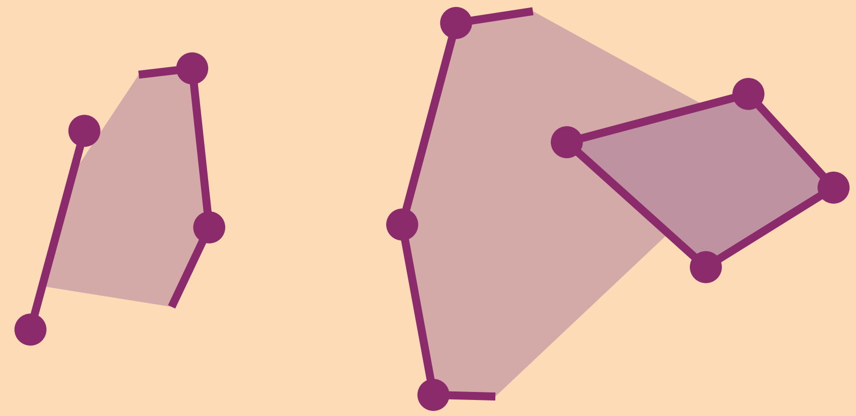


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

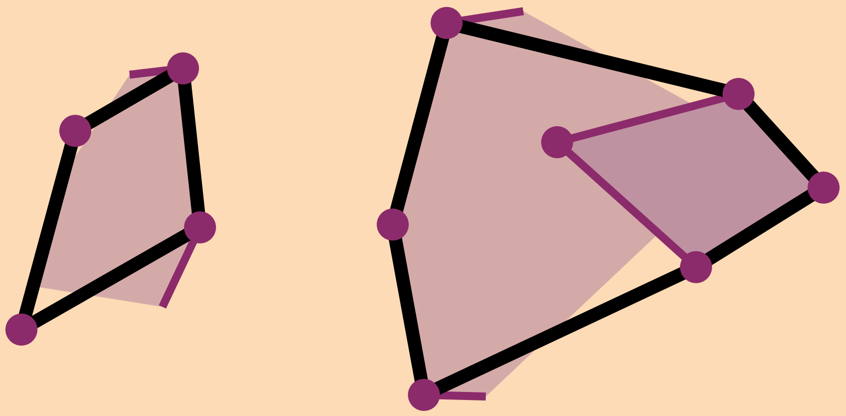


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

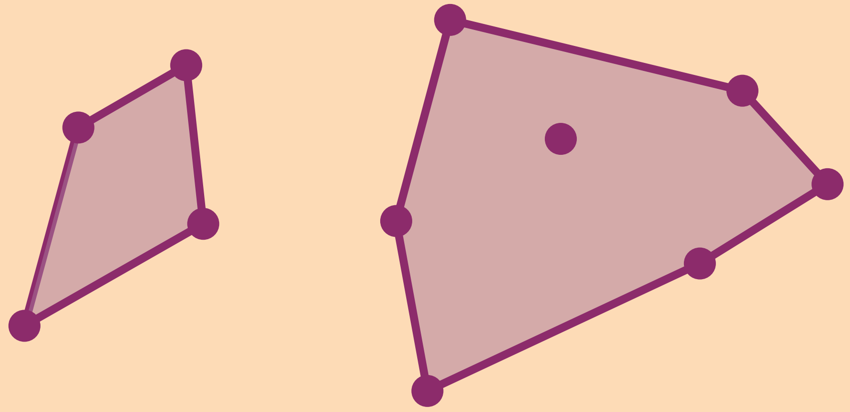


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

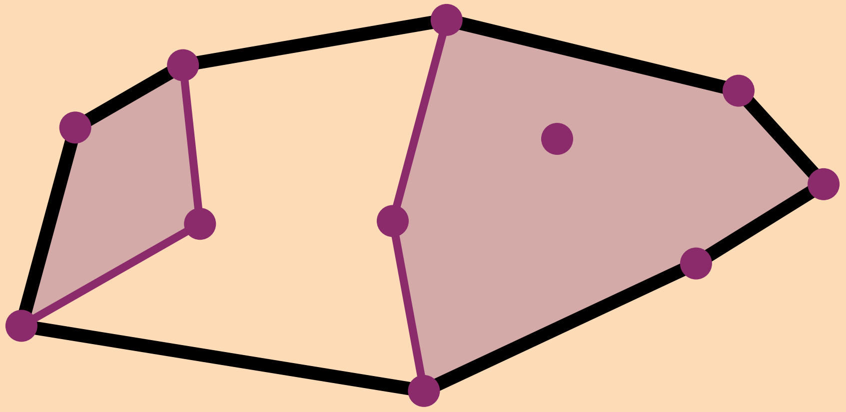


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

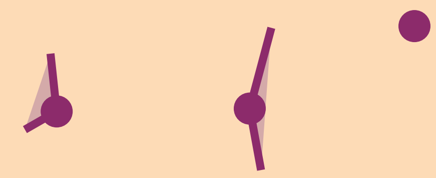


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

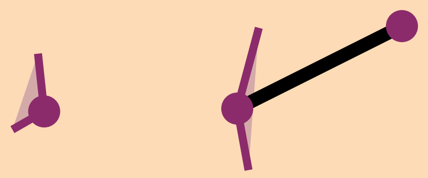


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

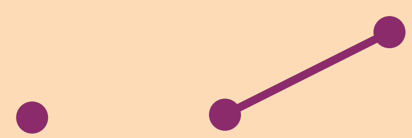


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

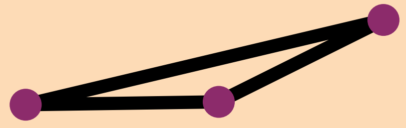


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...



ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

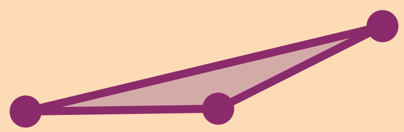
Etc...

ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

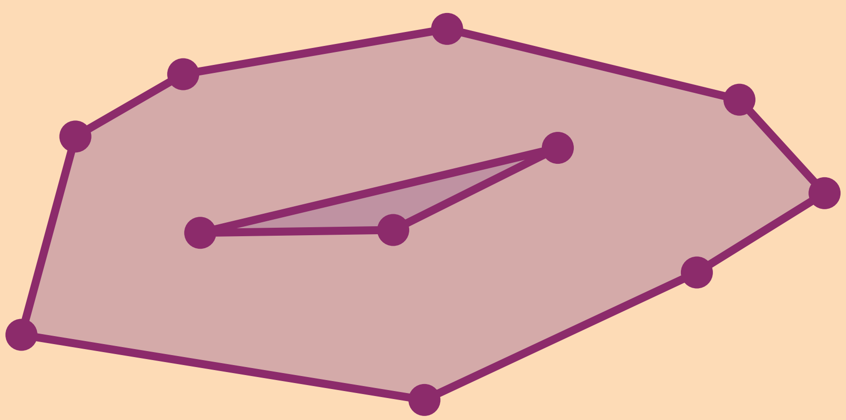


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

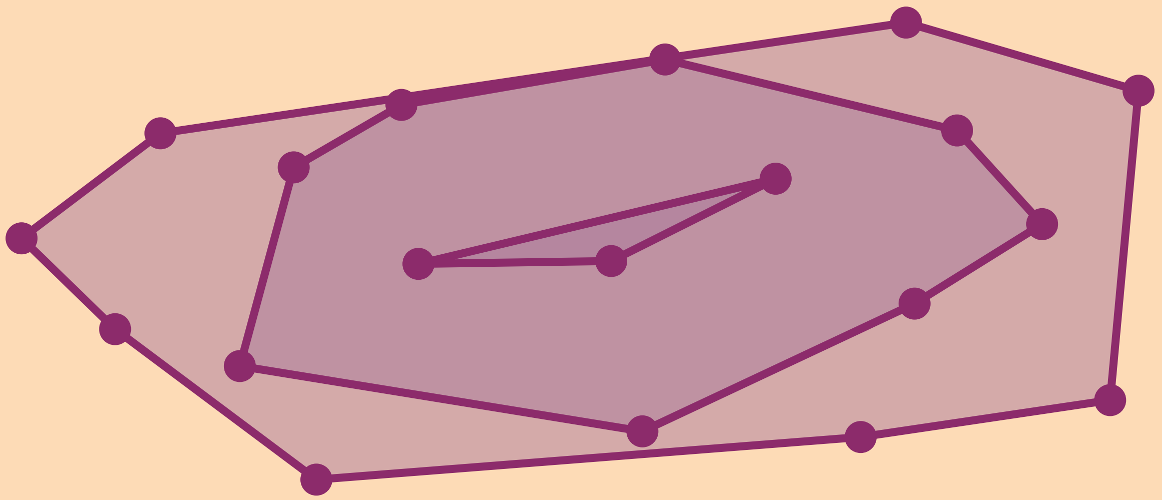


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

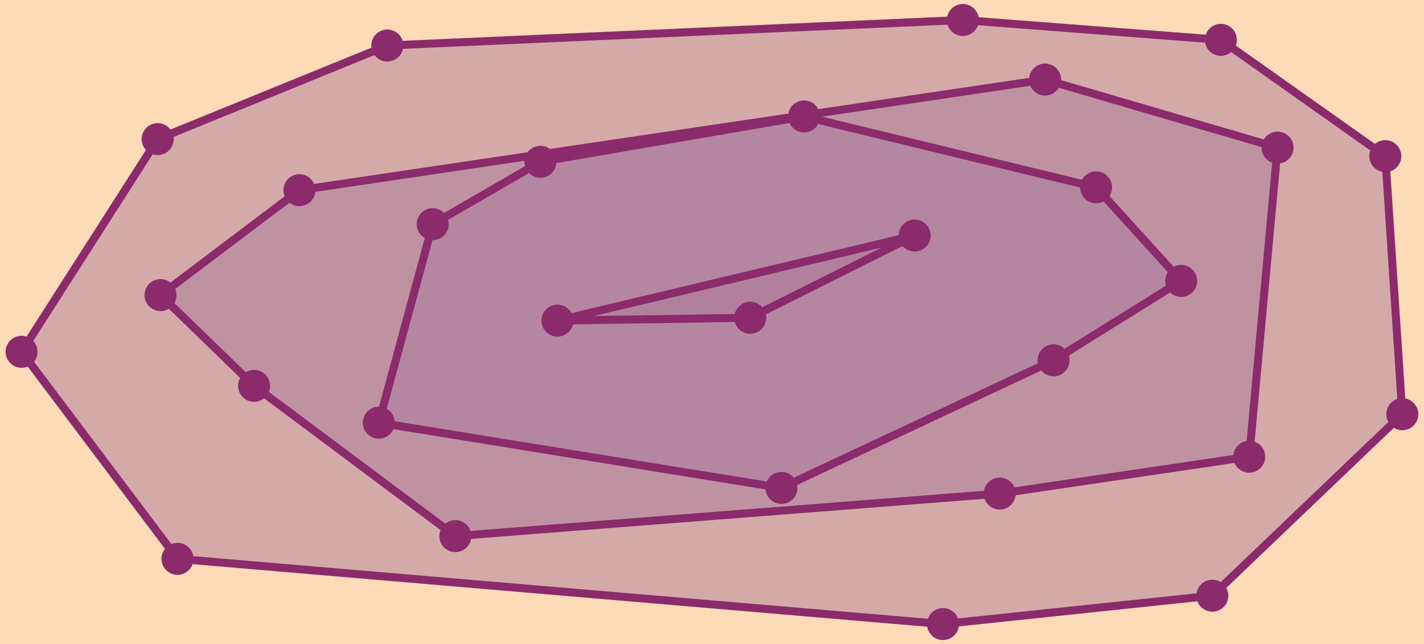


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

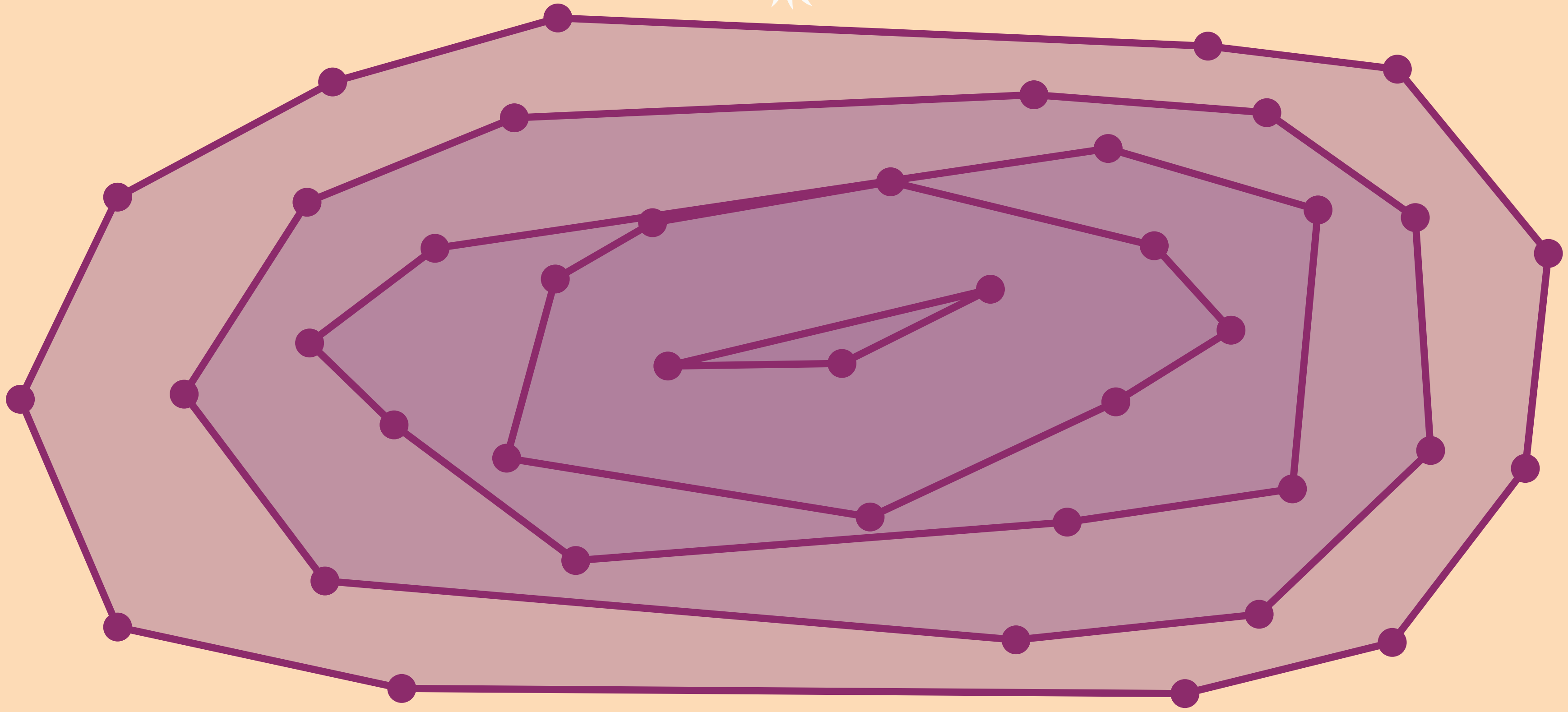


ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...



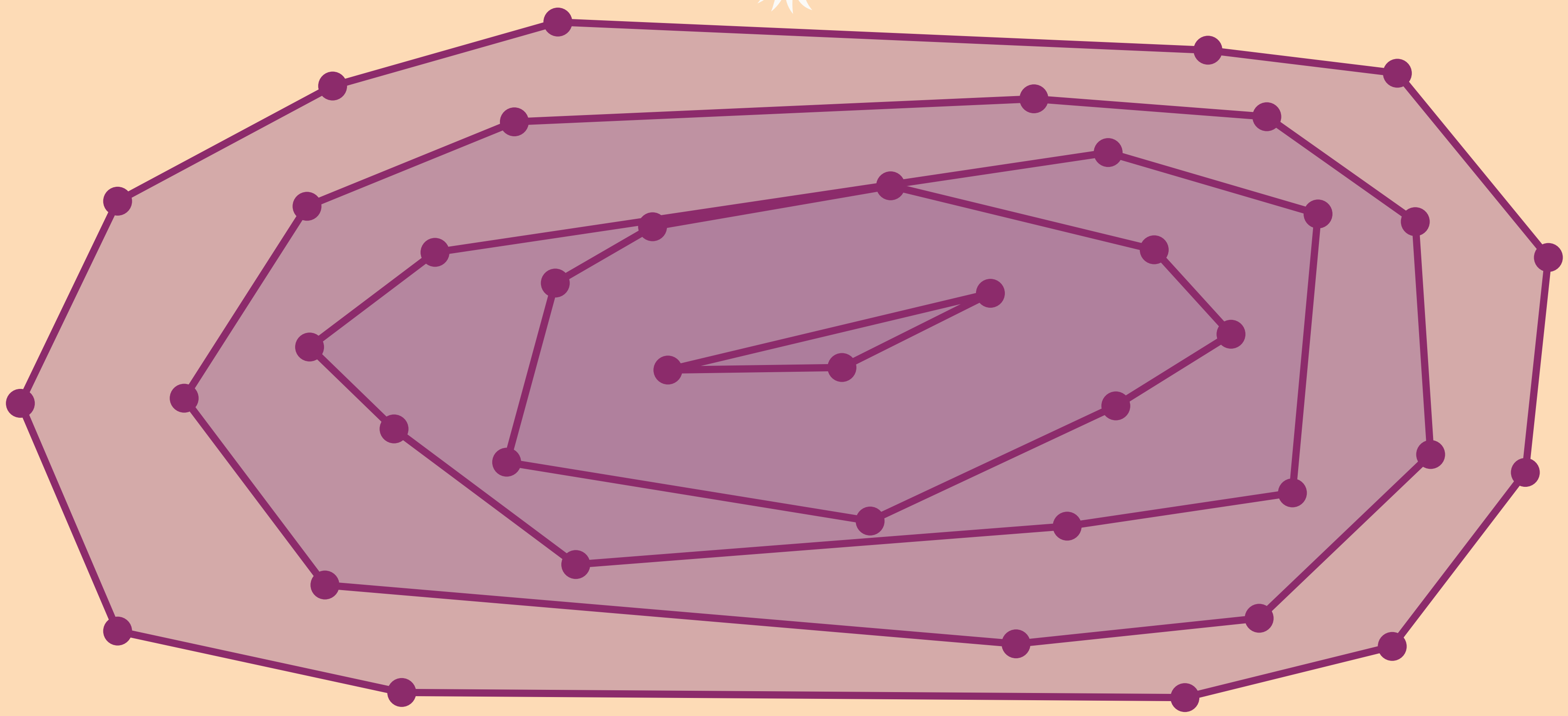
ONION RESTAURATION & UNIFICATION

Compute convex hulls of the individual onions.

Recurse once again.

Etc...

That took $O(k^2 \log n)$ time to compute k layers.



THE FINAL RESULT

THE FINAL RESULT

We can split a set of n unit disks in $O(n)$ time.

THE FINAL RESULT

We can split a set of n unit disks in $O(n)$ time.

So, we can compute $H(\mathcal{R})$ in $(n \log n)$ time.

THE FINAL RESULT

We can split a set of n unit disks in $O(n)$ time.

So, we can compute $H(\mathcal{R})$ in $(n \log n)$ time.

We can unite a pair of onions in $O(k^2 \log n)$ time.

THE FINAL RESULT

We can split a set of n unit disks in $O(n)$ time.

So, we can compute $H(\mathcal{R})$ in $(n \log n)$ time.

We can unite a pair of onions in $O(k^2 \log n)$ time.

So, we can reconstruct an onion in $O(n \log k)$ time.

THANK YOU!

any questions?