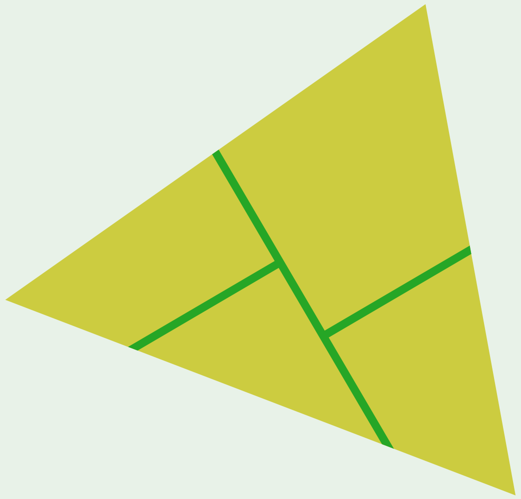
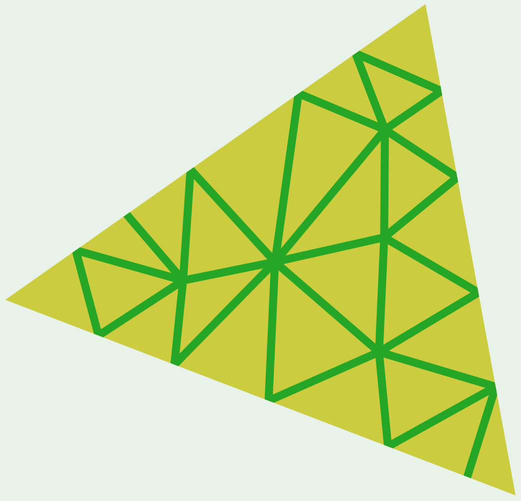
TRIANGULATING THE SQUARE & SQUARING THE TRIANGLE



Maarten Löffler — University of California, Irvine

Wolfgang Mulzer — Freie Universität Berlin

OR, QUADTREES *AND DELAUNAY TRIANGULATIONS* ARE EQUIV*ALENT*



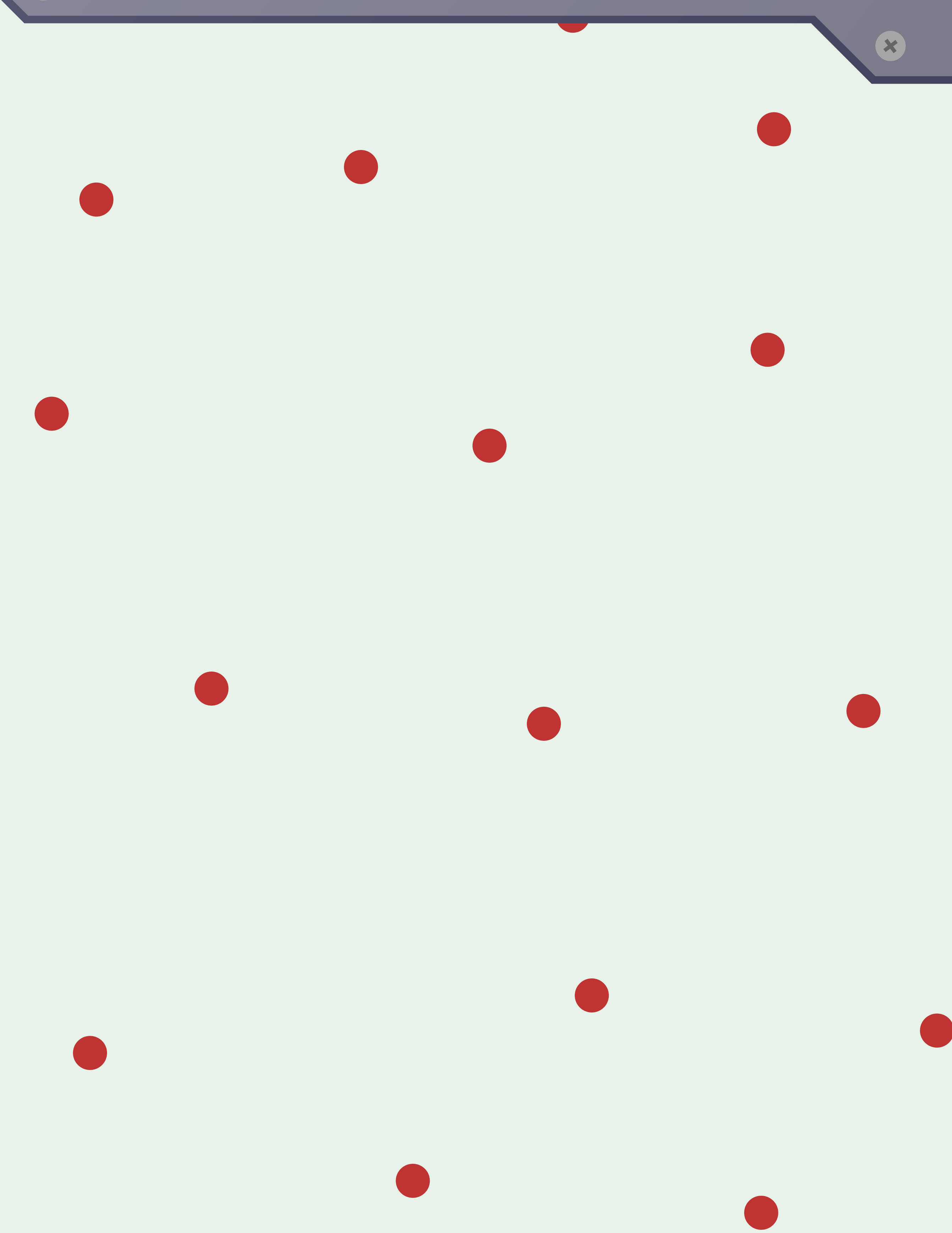
Maarten Löffler — *University of California, Irvine*

Wolfgang Mulzer — *Freie Universität Berlin*

PART I *INTRODUCTION*

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

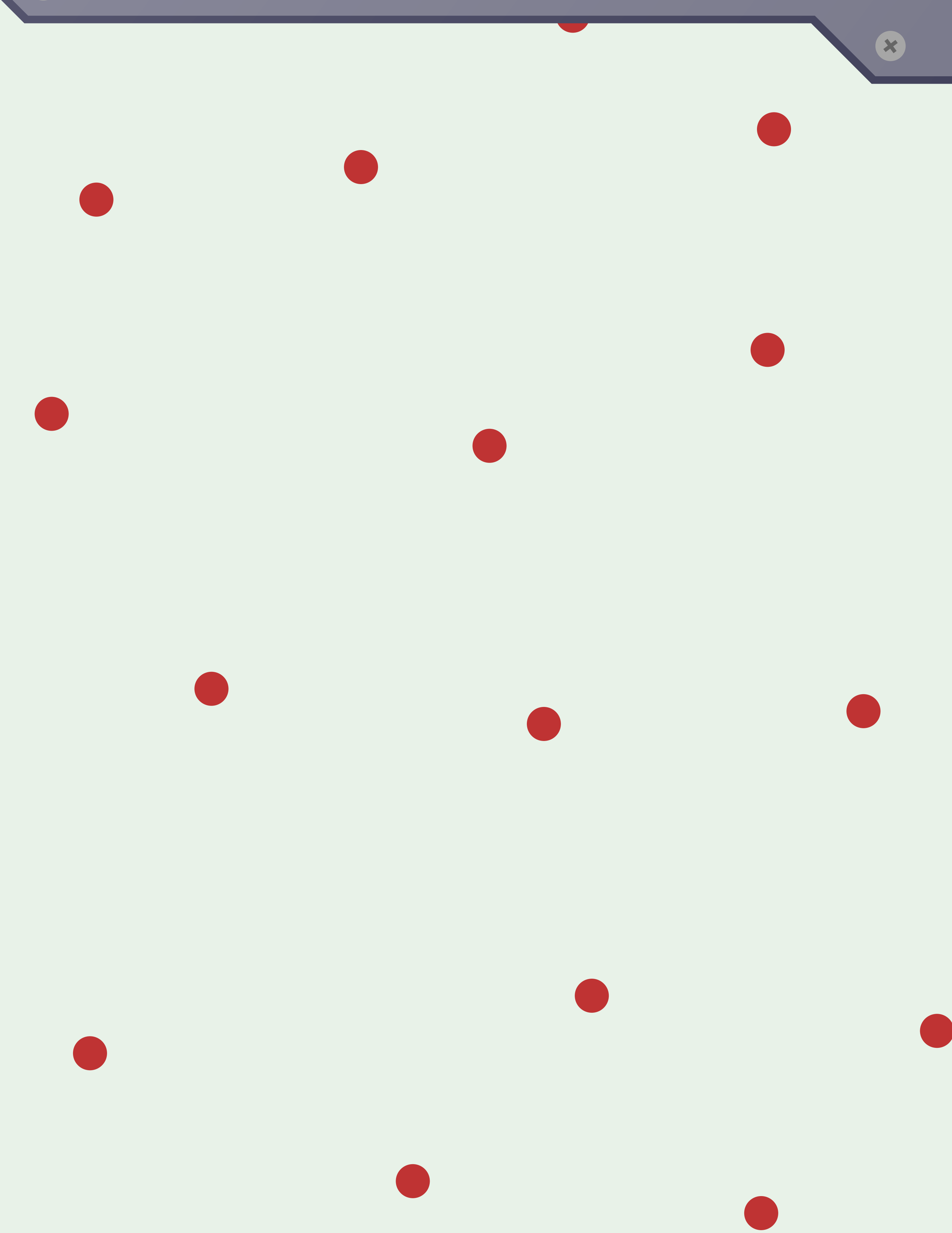
Let P be a set of n points in the plane.



PROXIMITY STRUCTURES

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

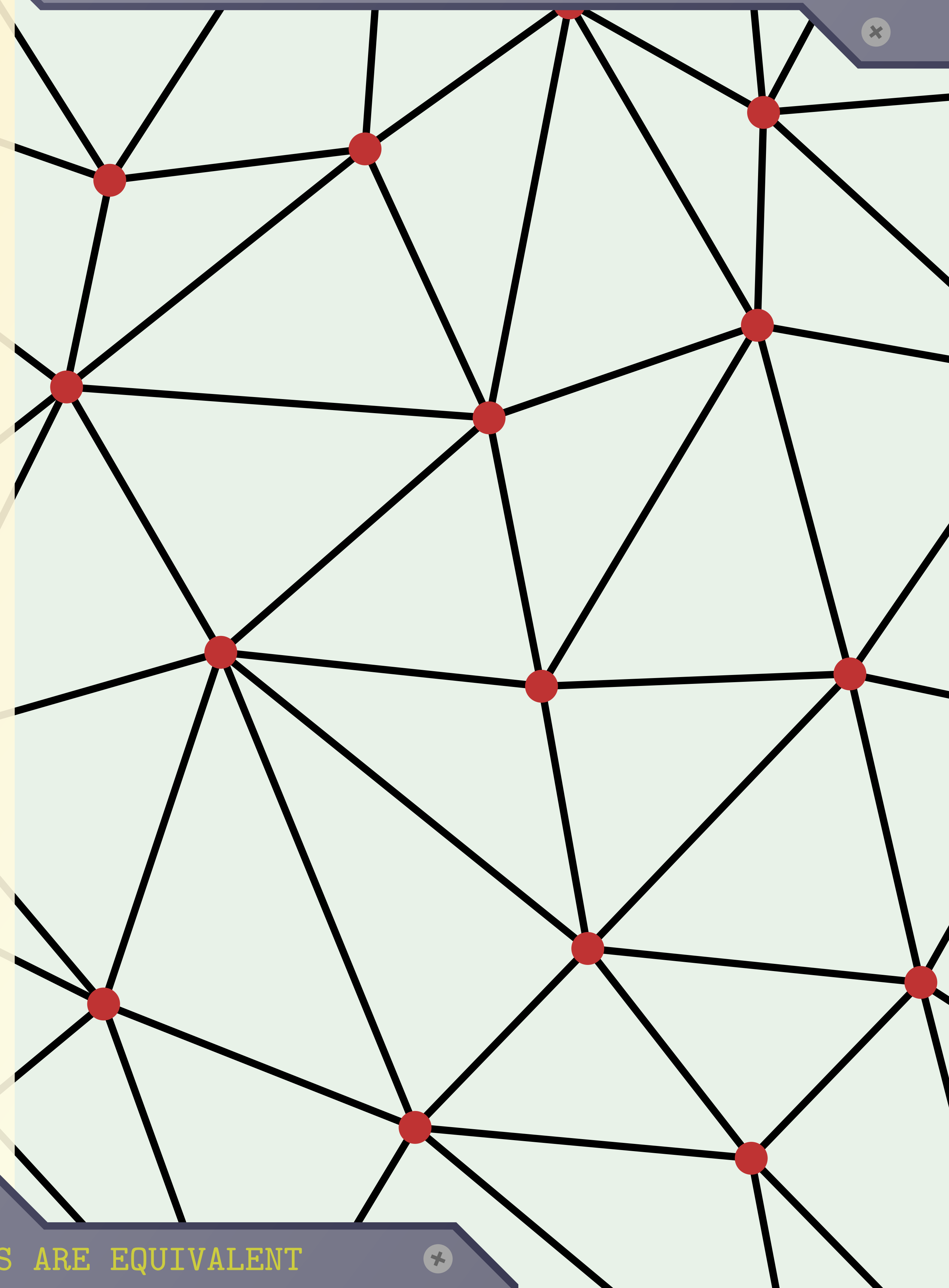
A *proximity structure* on P is ‘a structure that stores some kind of useful local information’.



PROXIMITY STRUCTURES

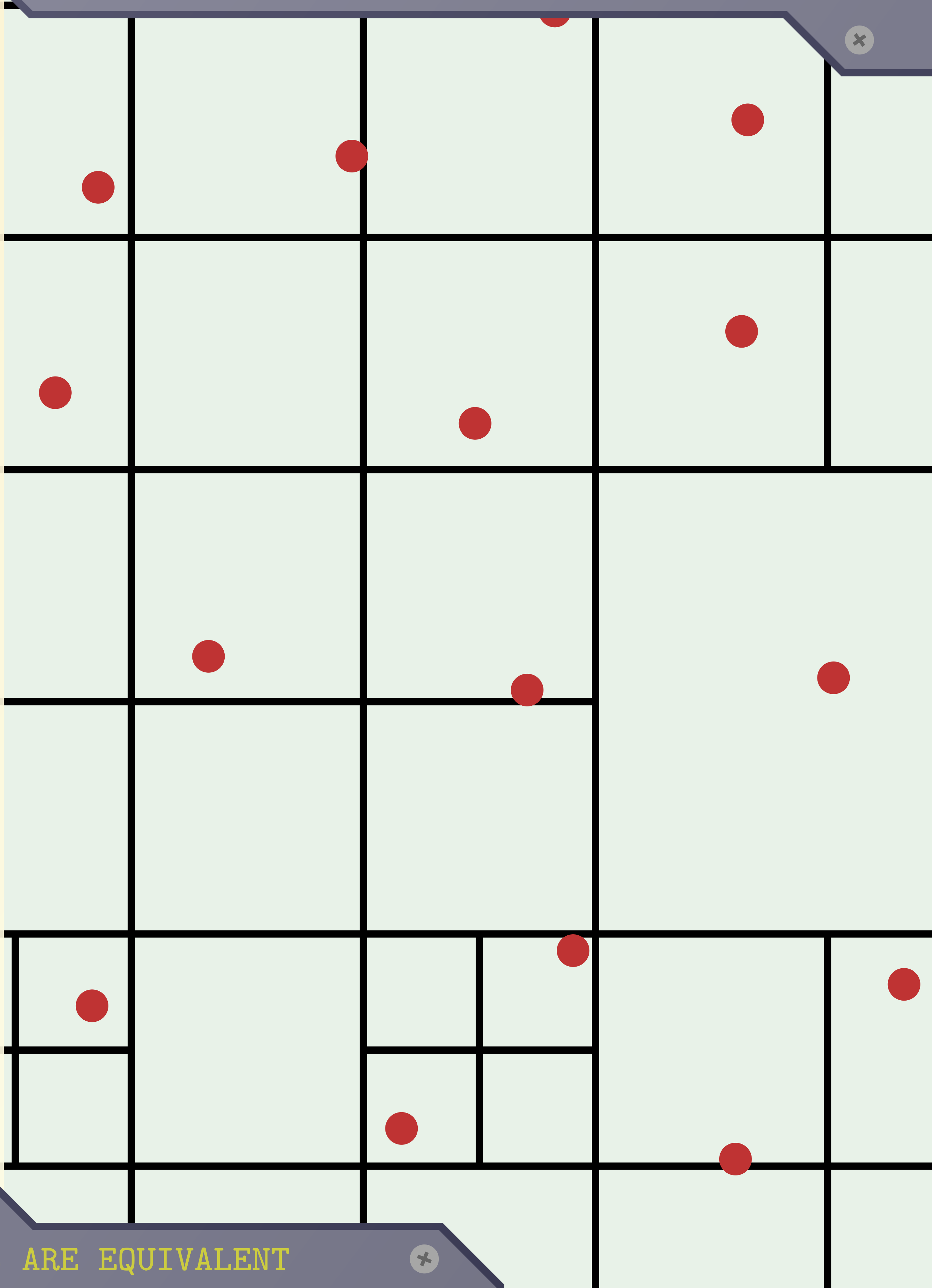
Let P be a set of n points in the plane.

A *proximity structure* on P is ‘a structure that stores some kind of useful local information’.



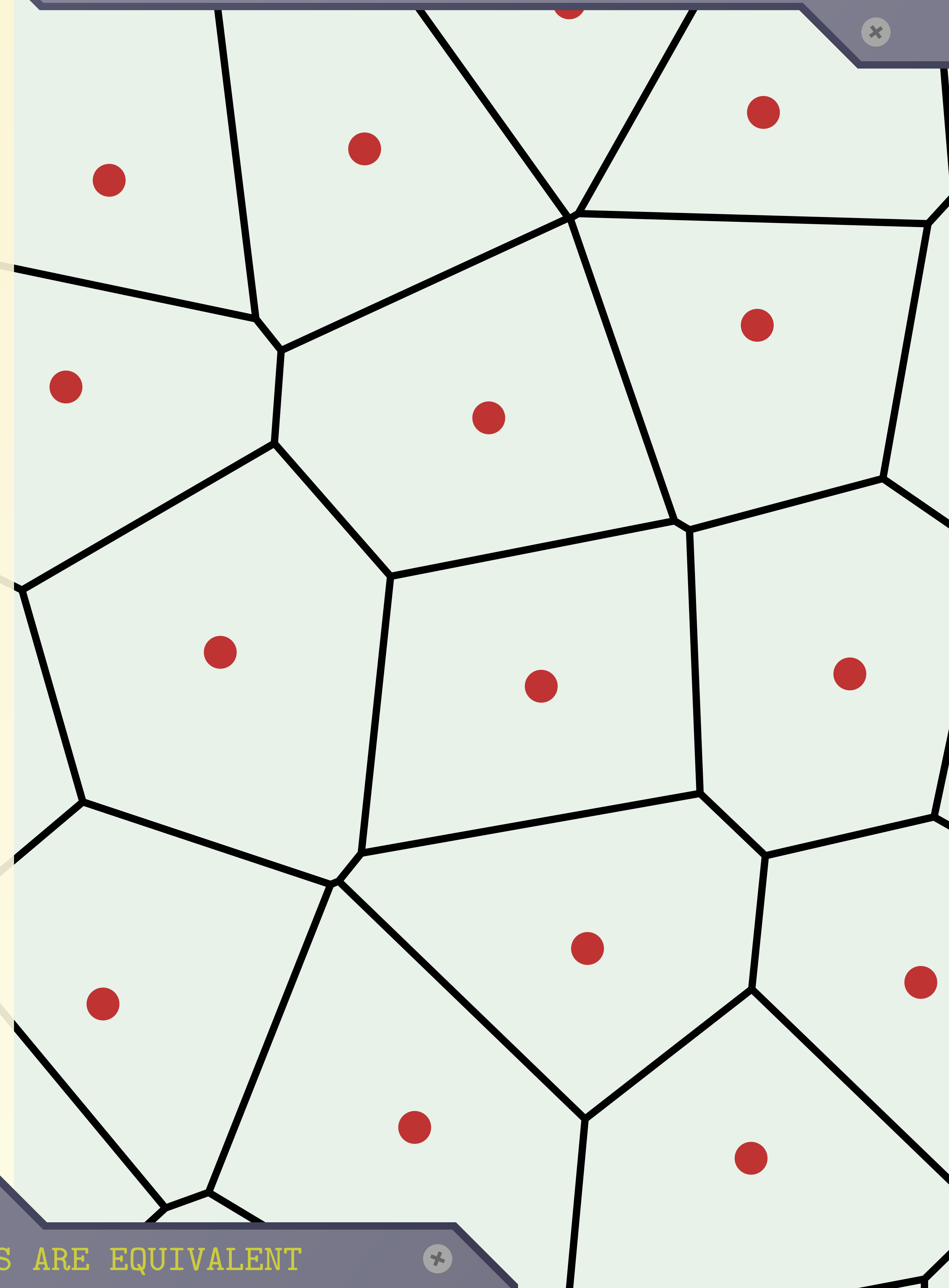
Let P be a set of n points in the plane.

A *proximity structure* on P is 'a structure that stores some kind of useful local information'.



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

A *proximity structure* on P is ‘a structure that stores some kind of useful local information’.



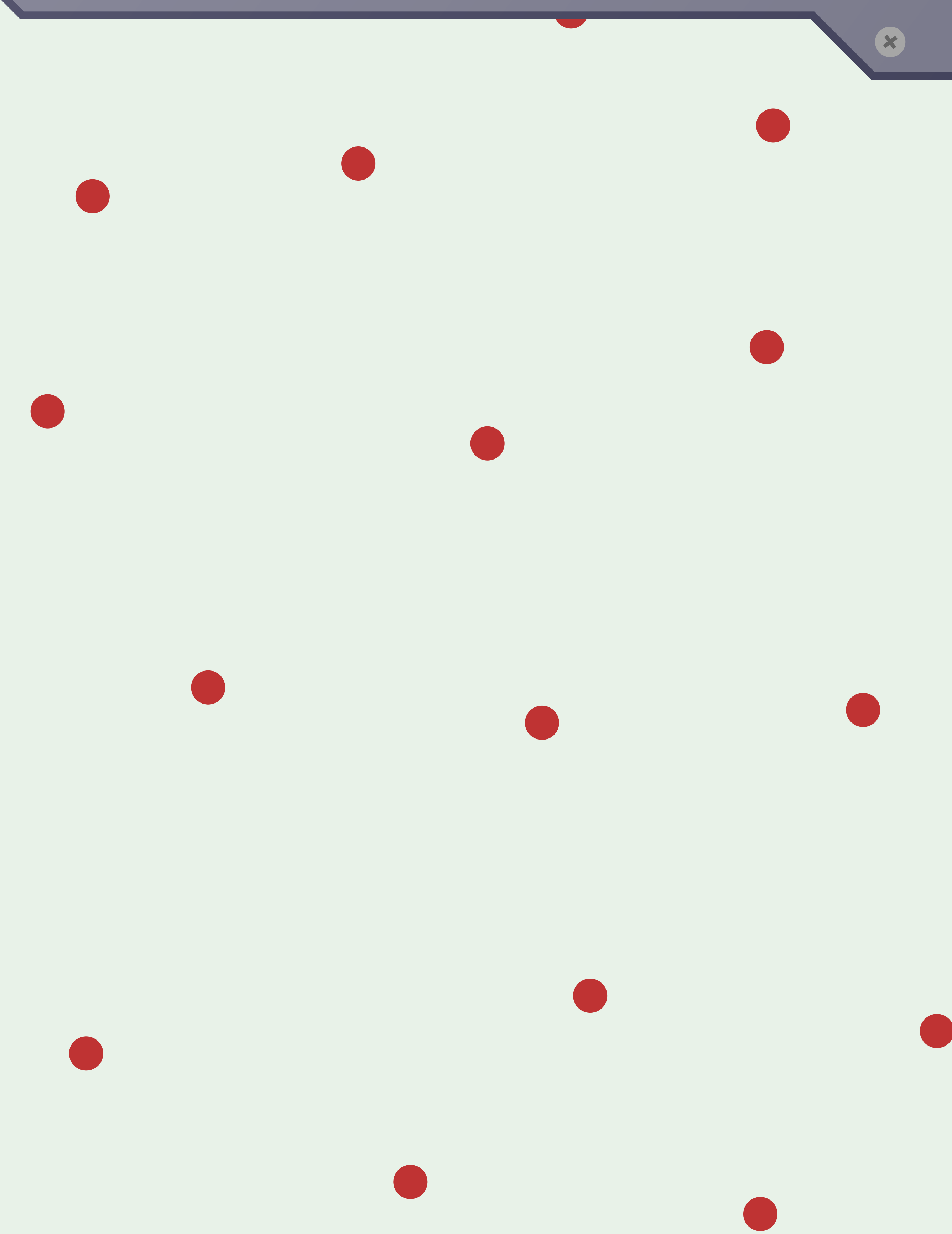
All these structures take $\Omega(n \log n)$ time to build.

All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.

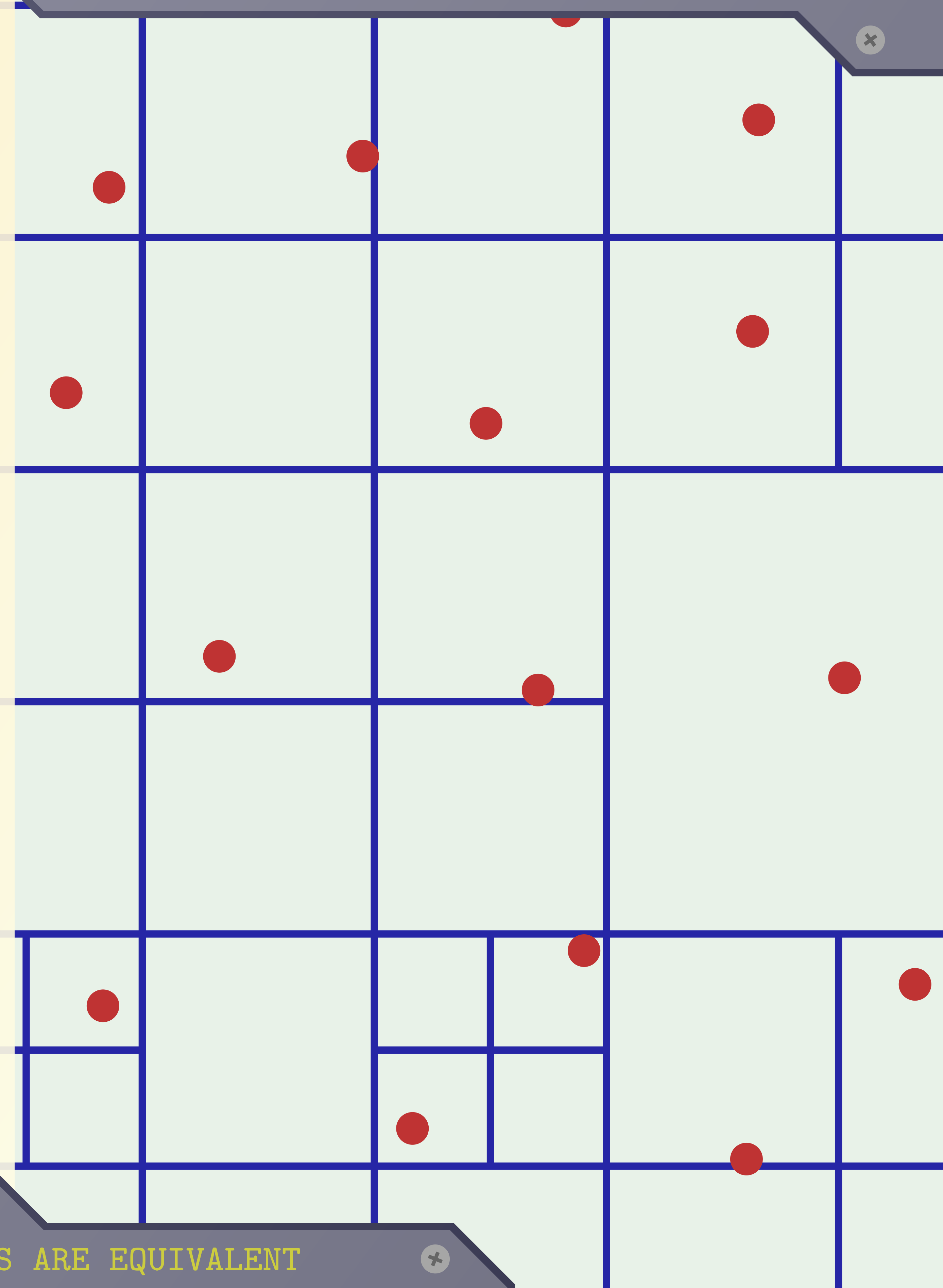
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



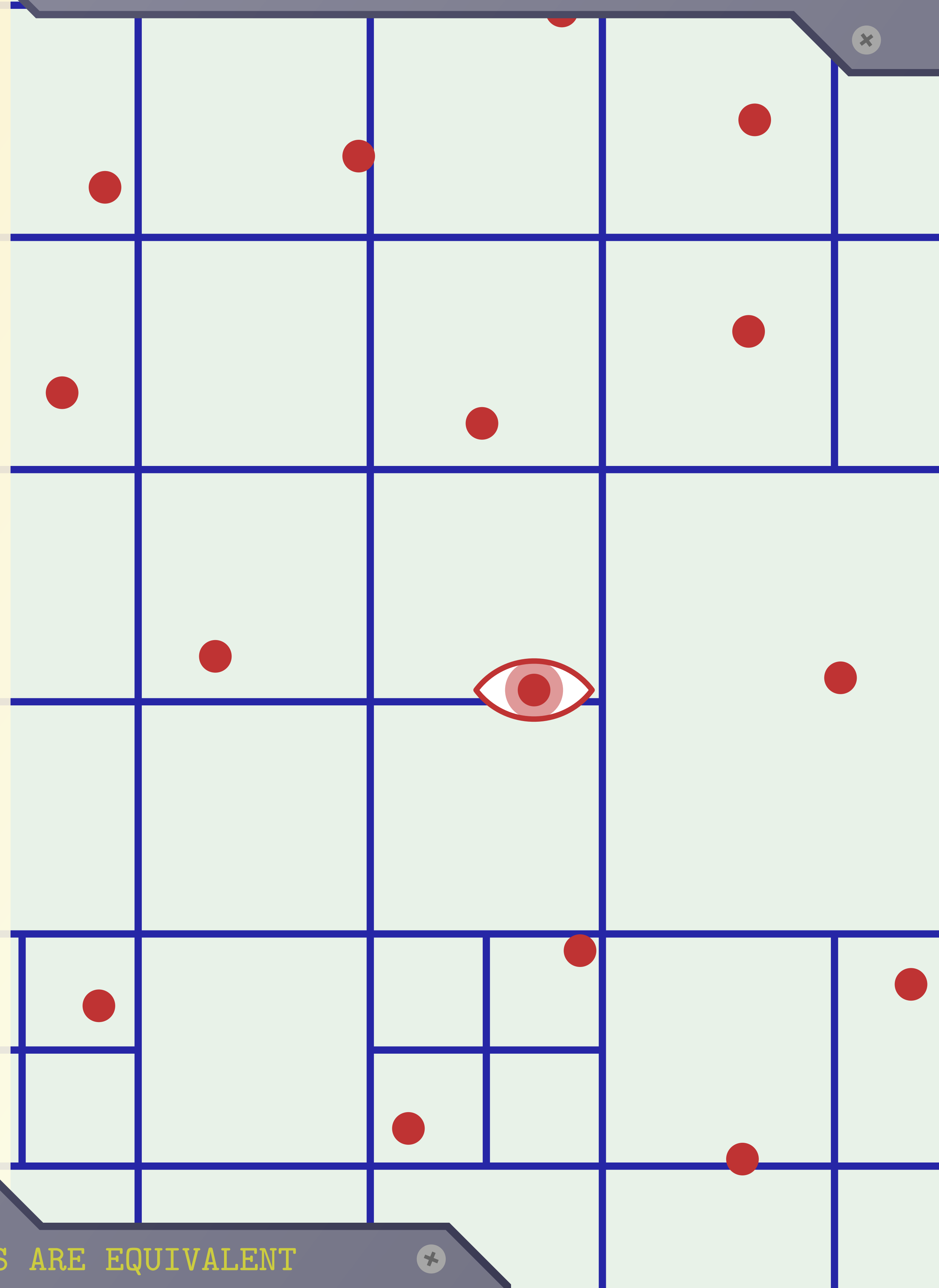
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



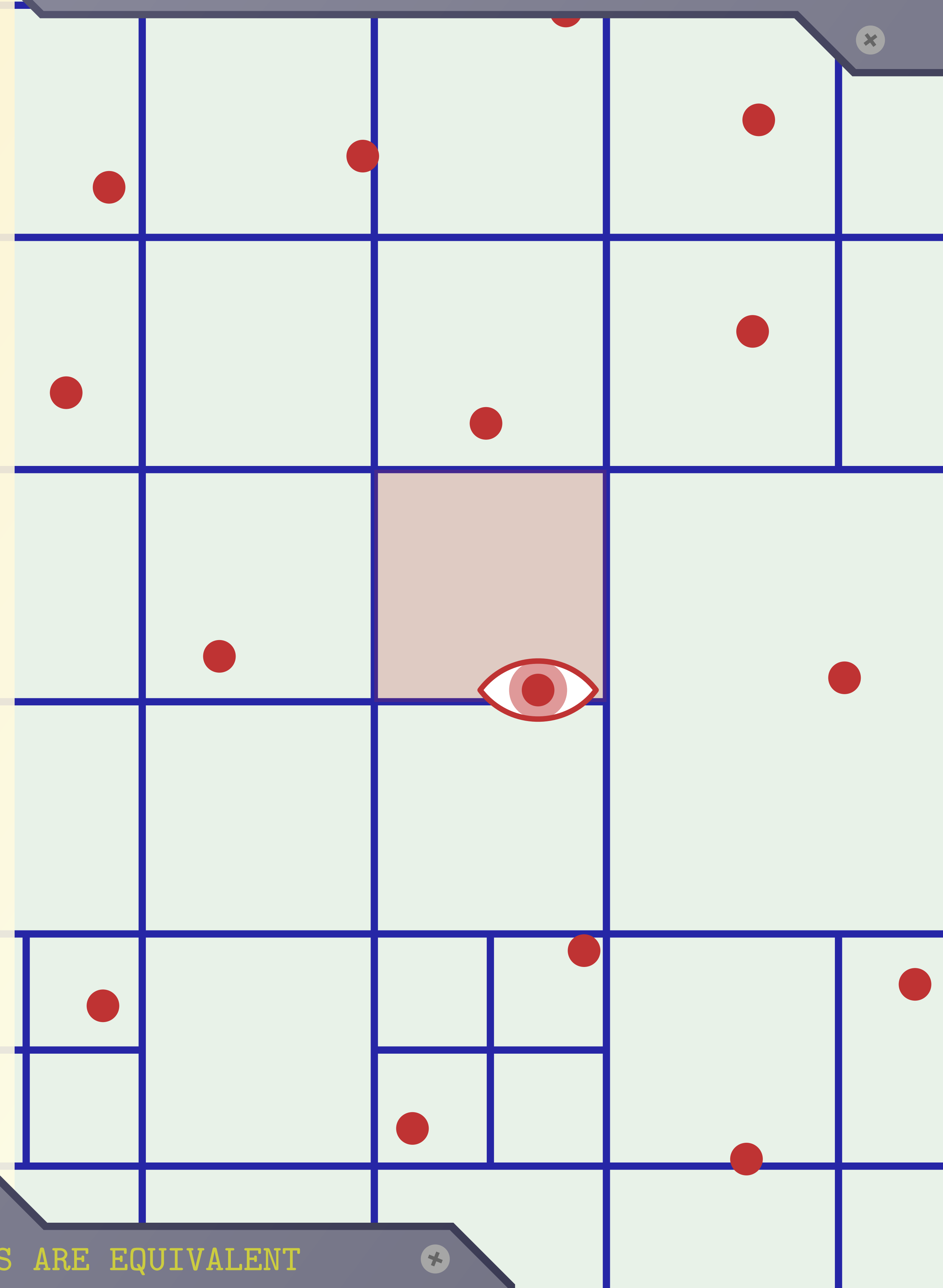
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



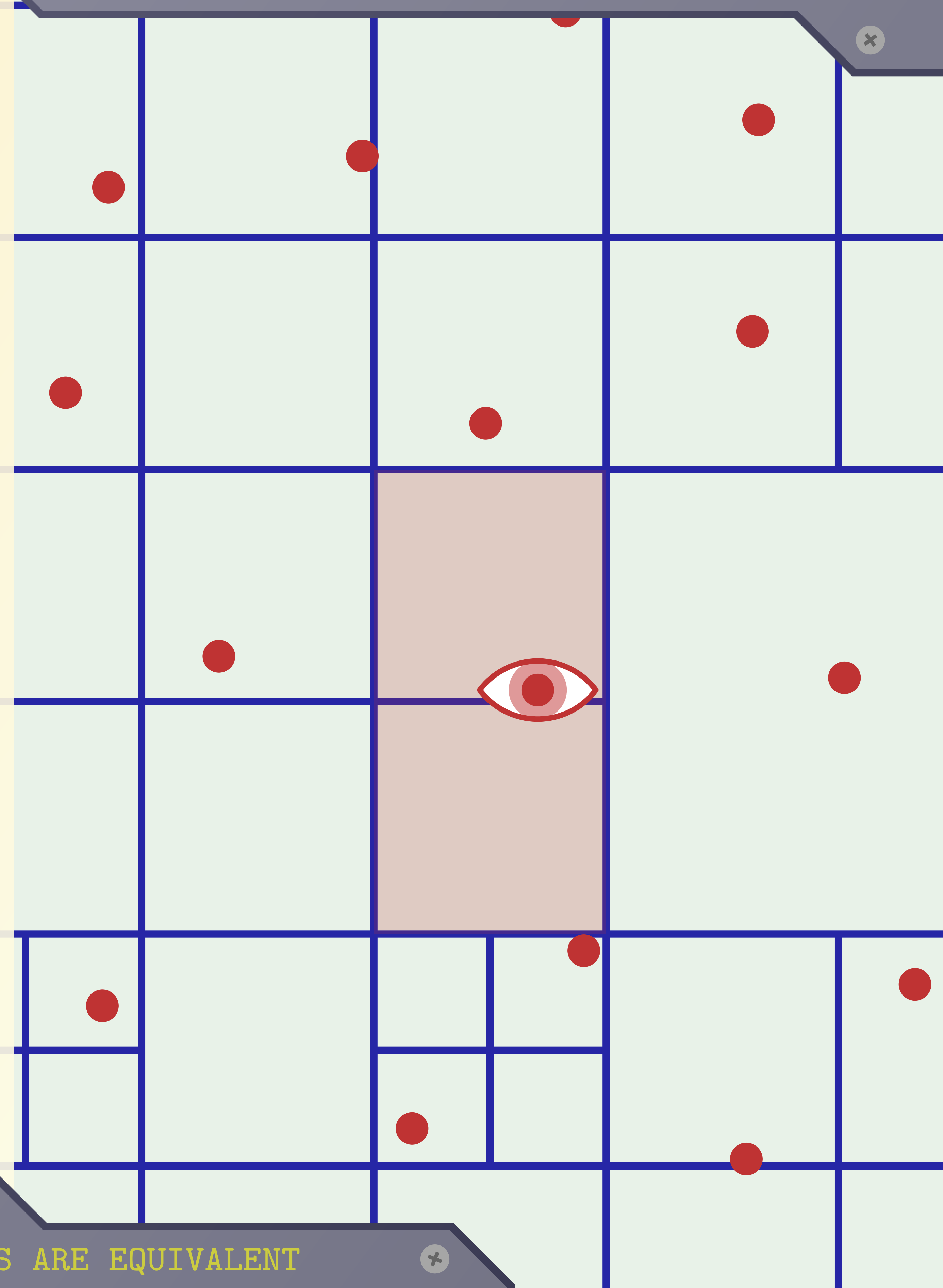
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



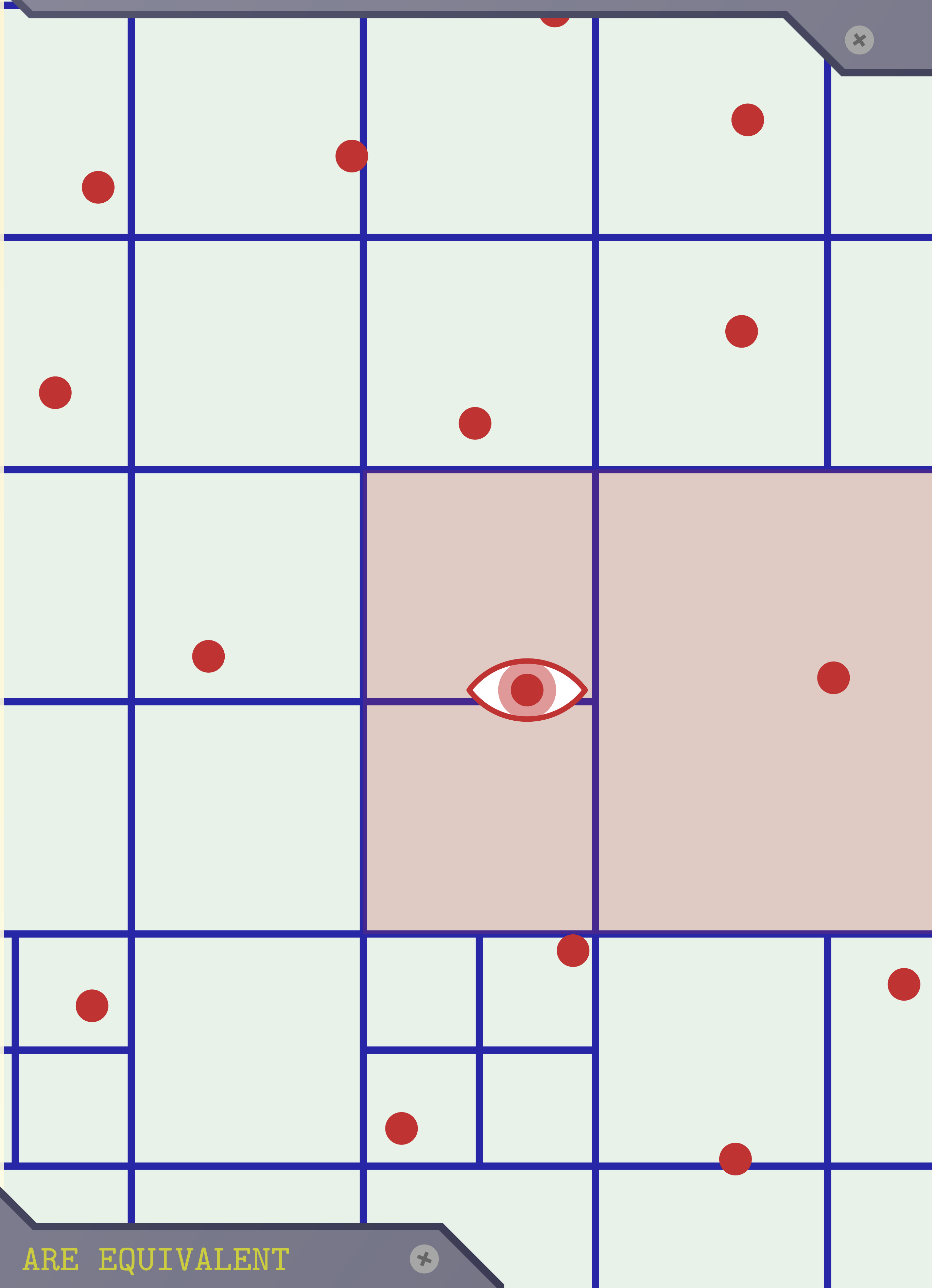
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



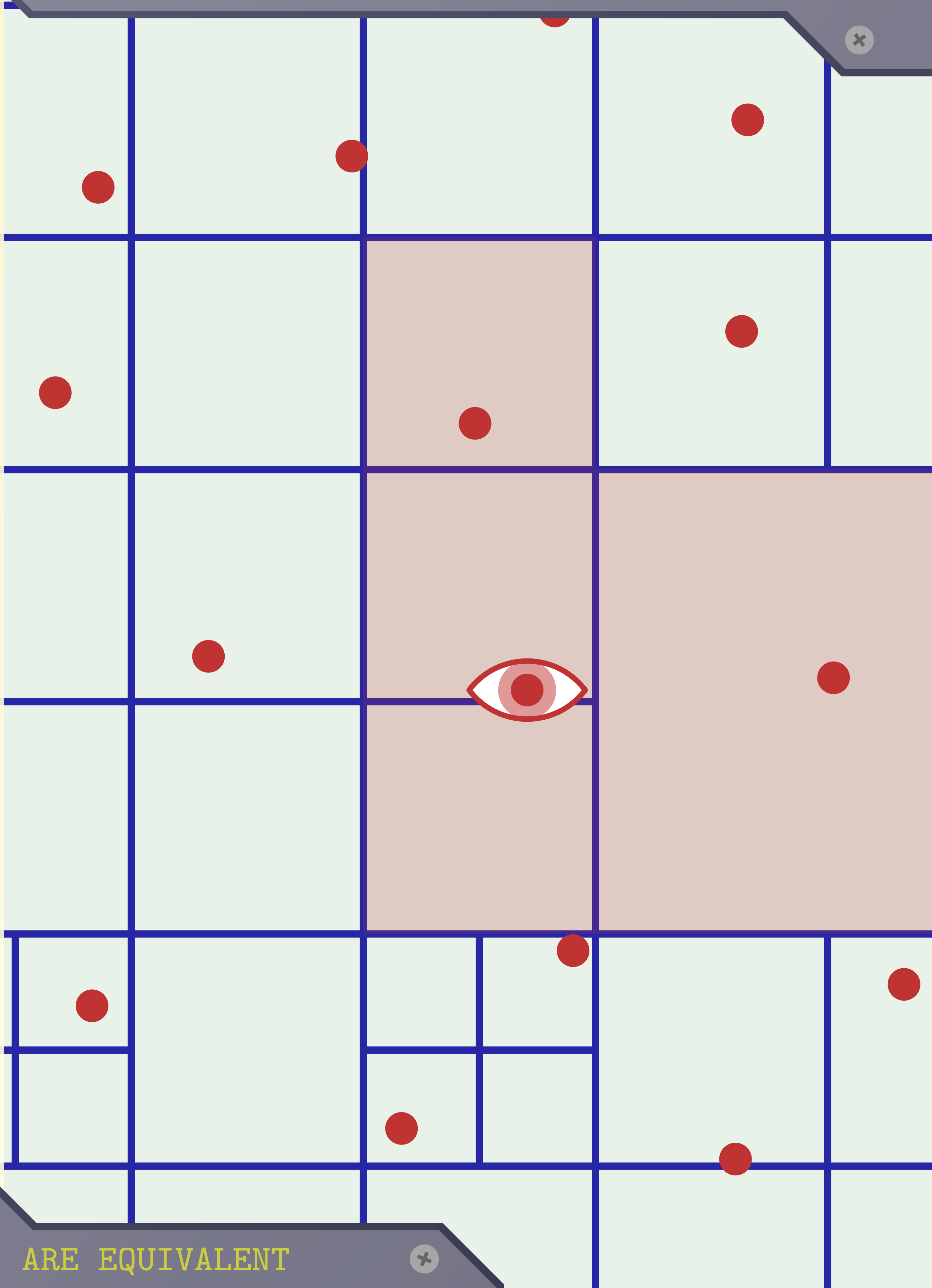
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



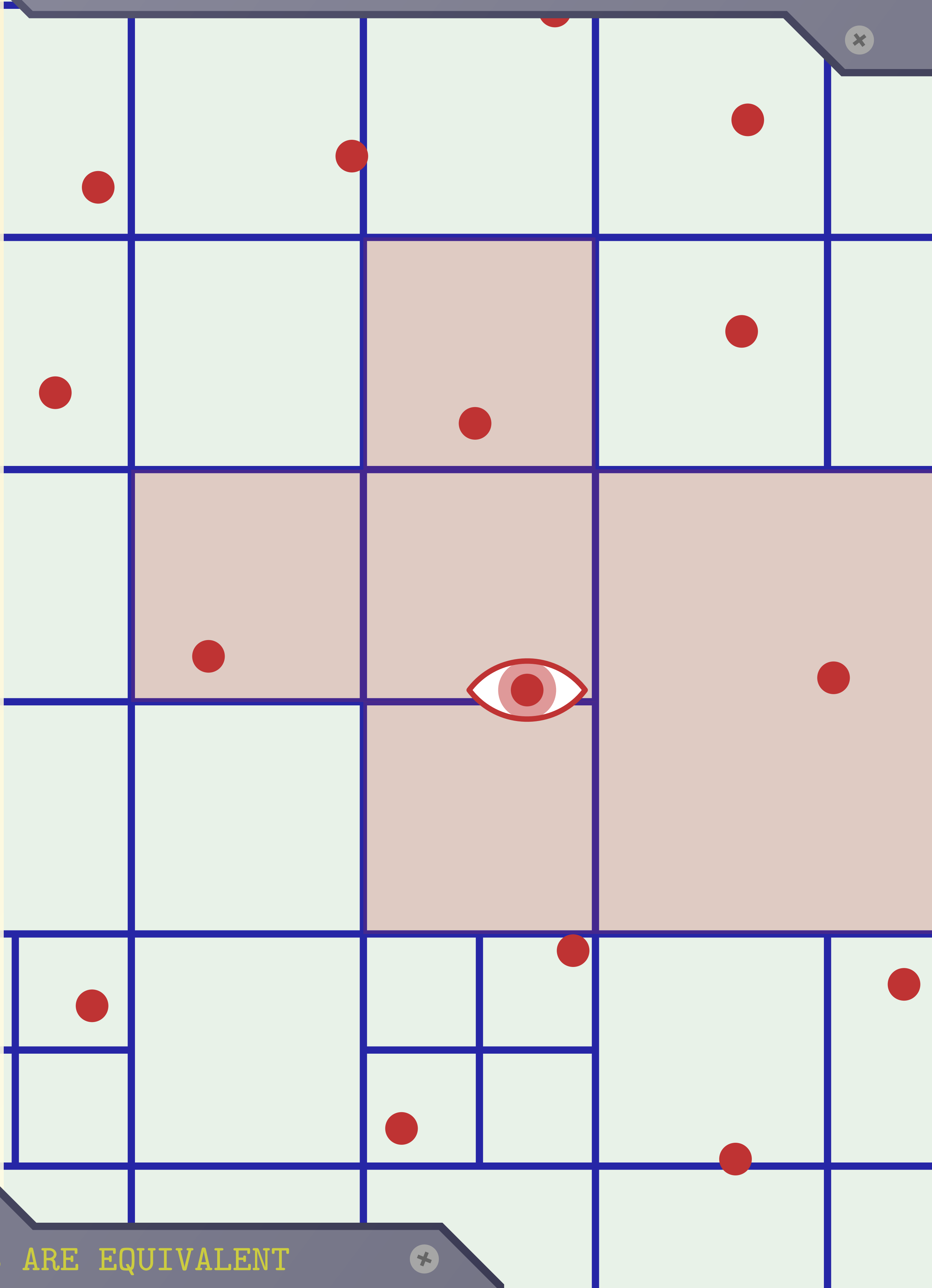
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



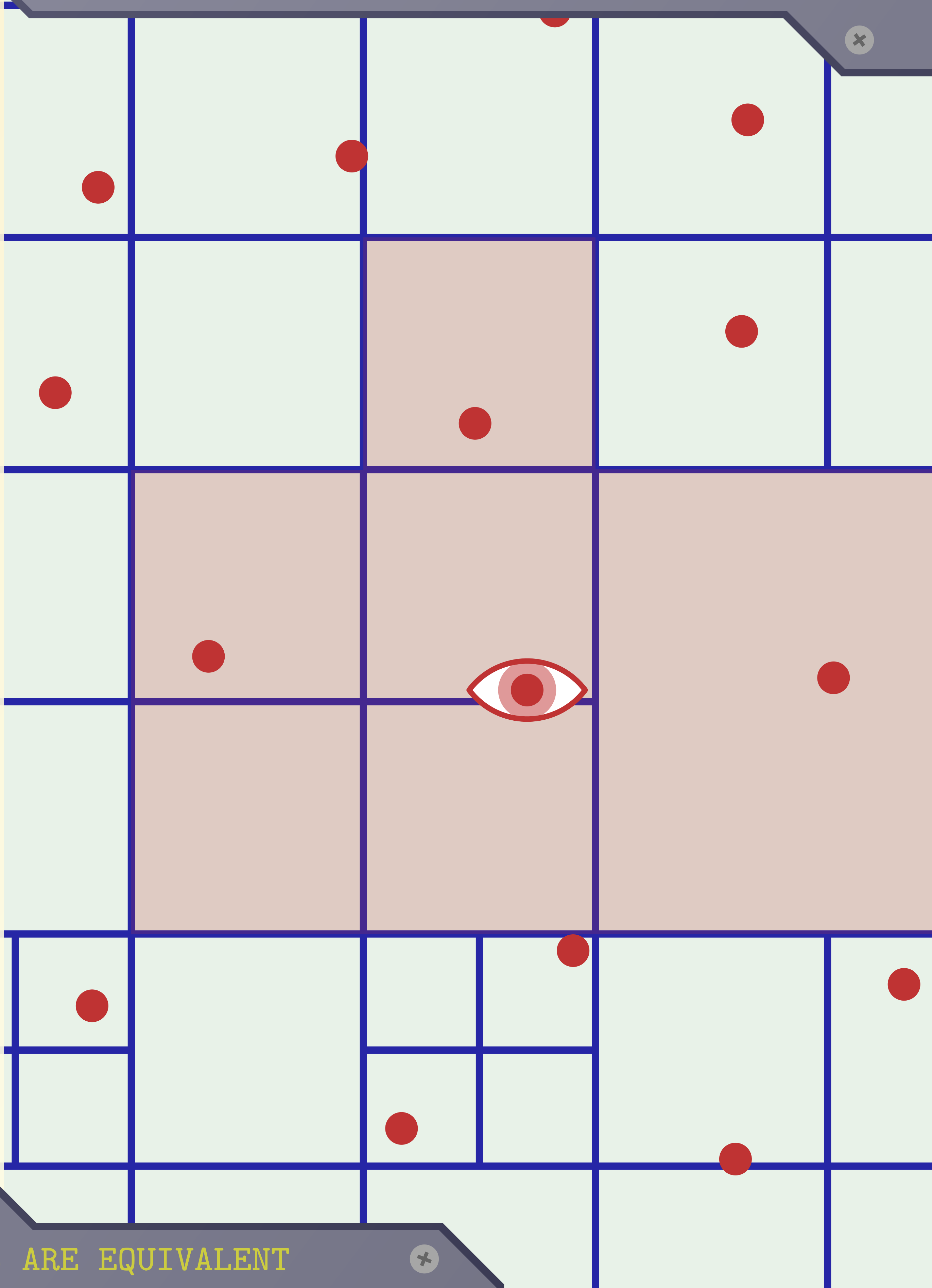
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



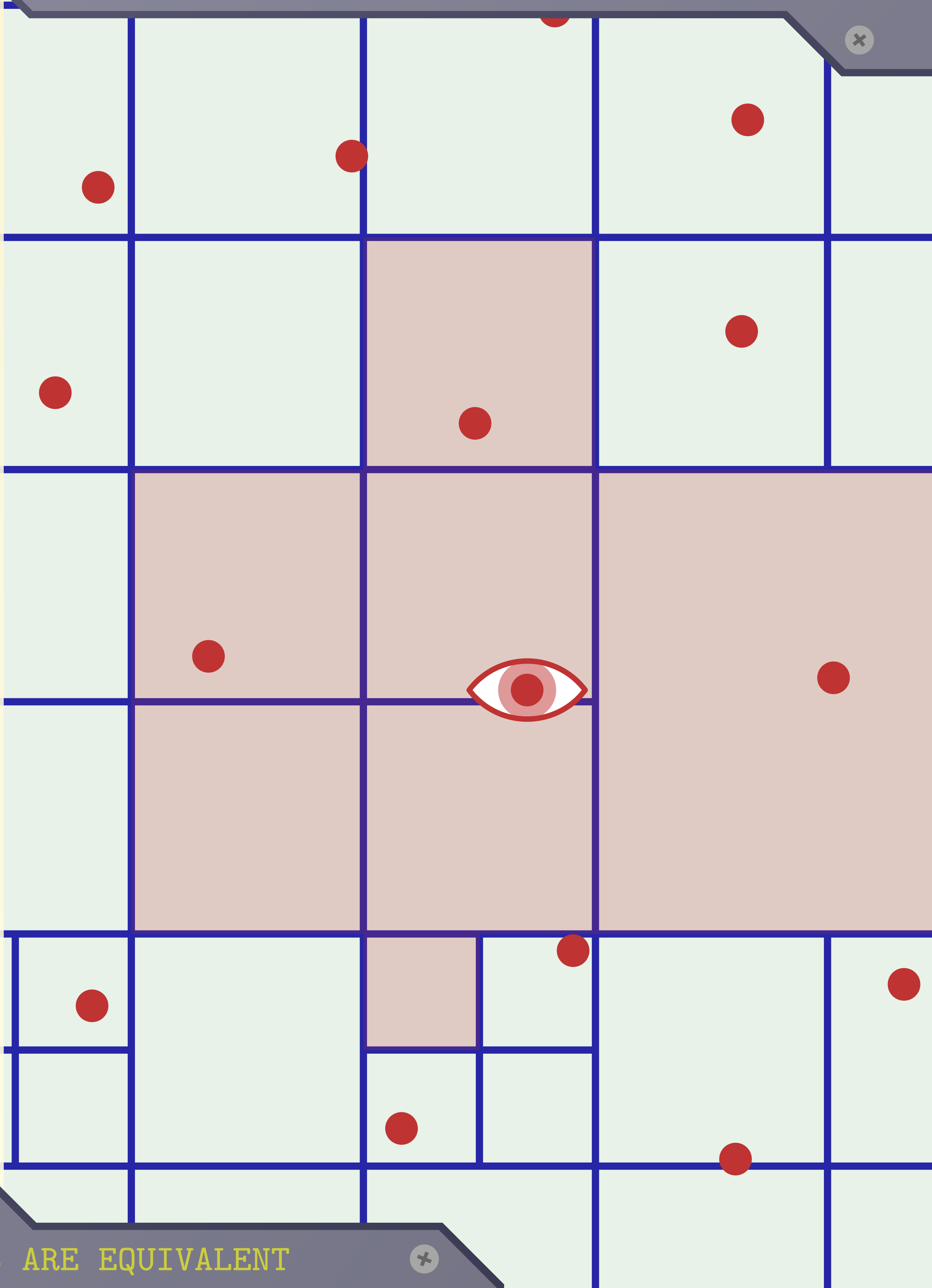
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



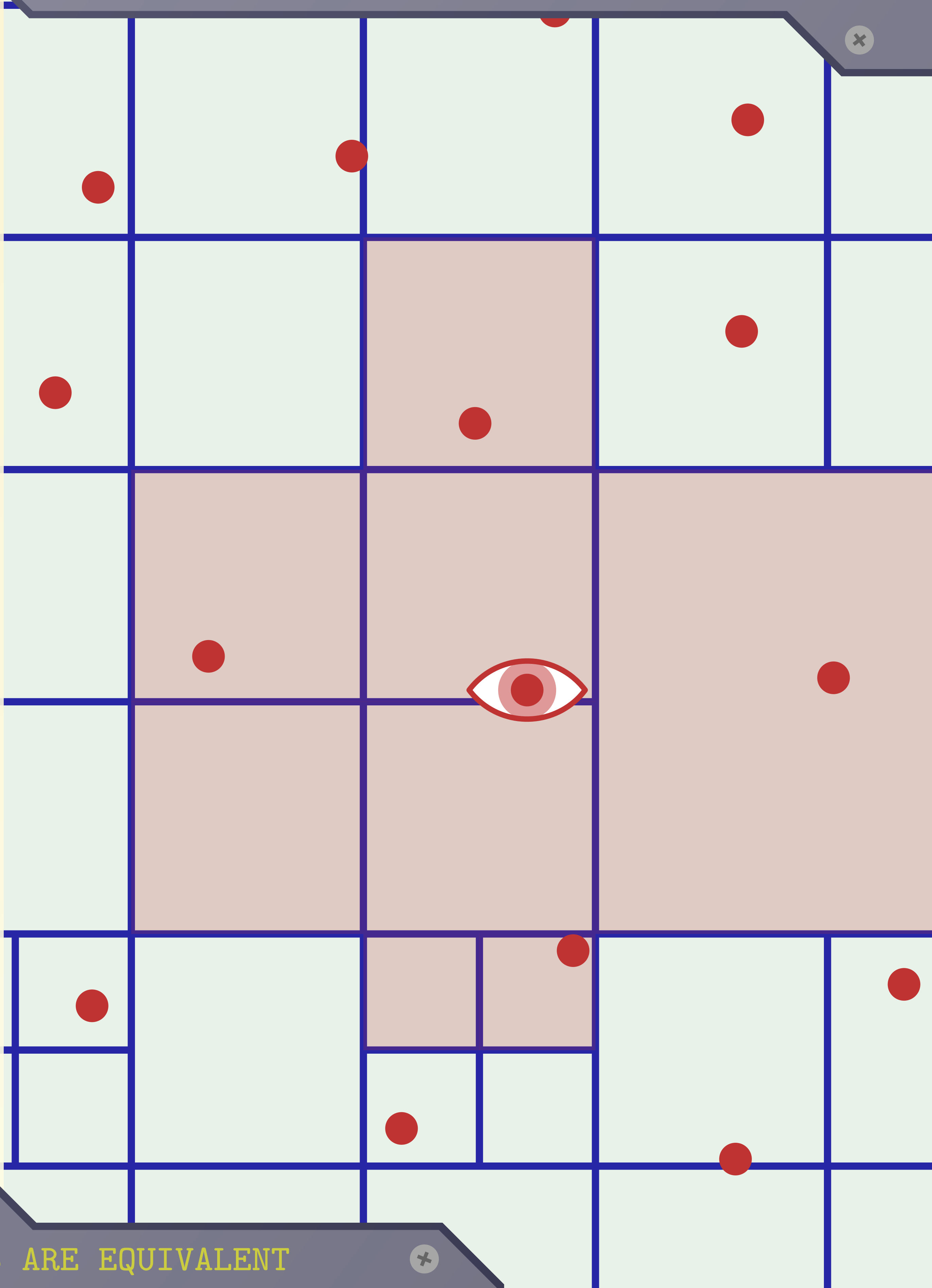
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



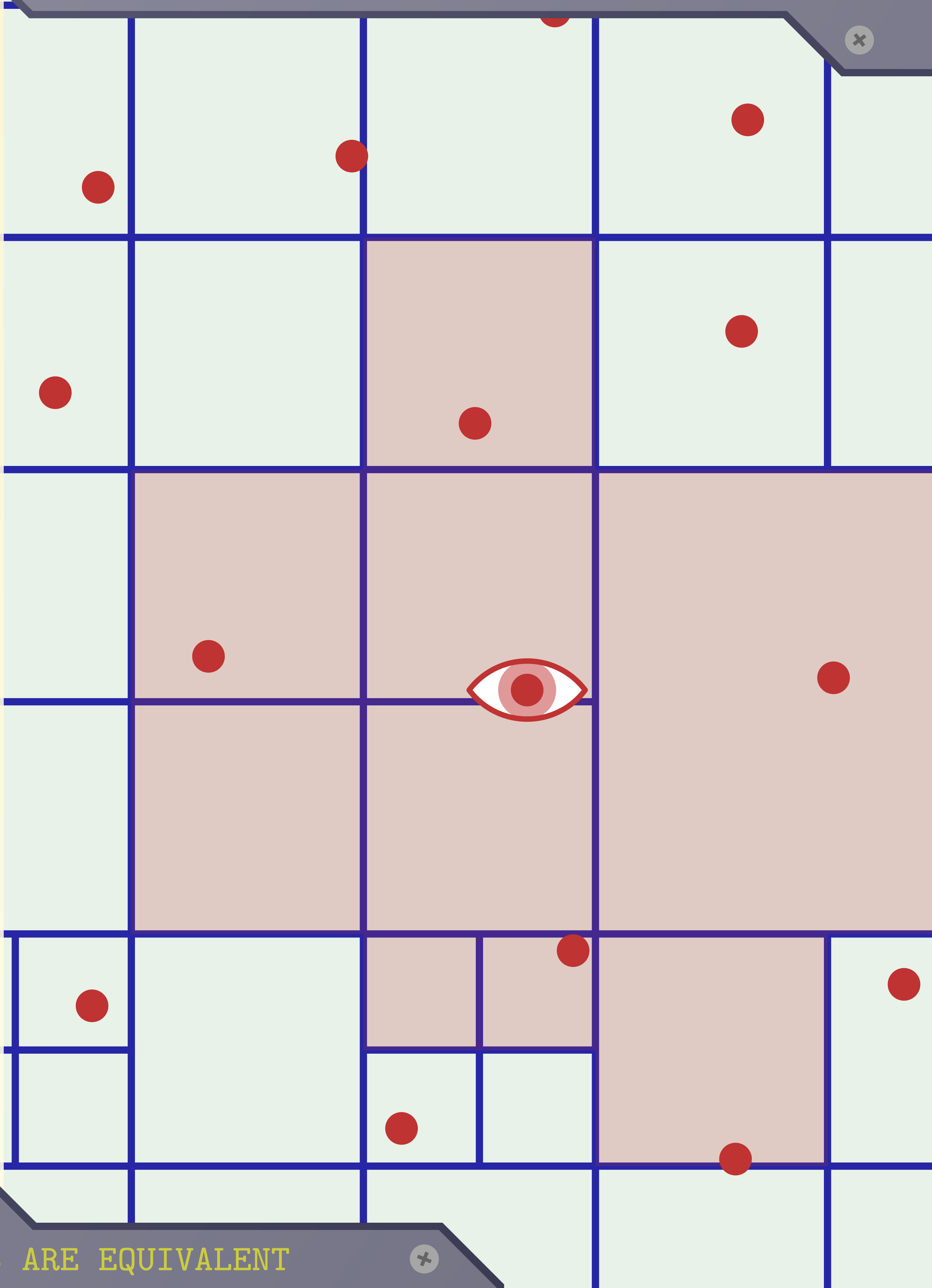
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



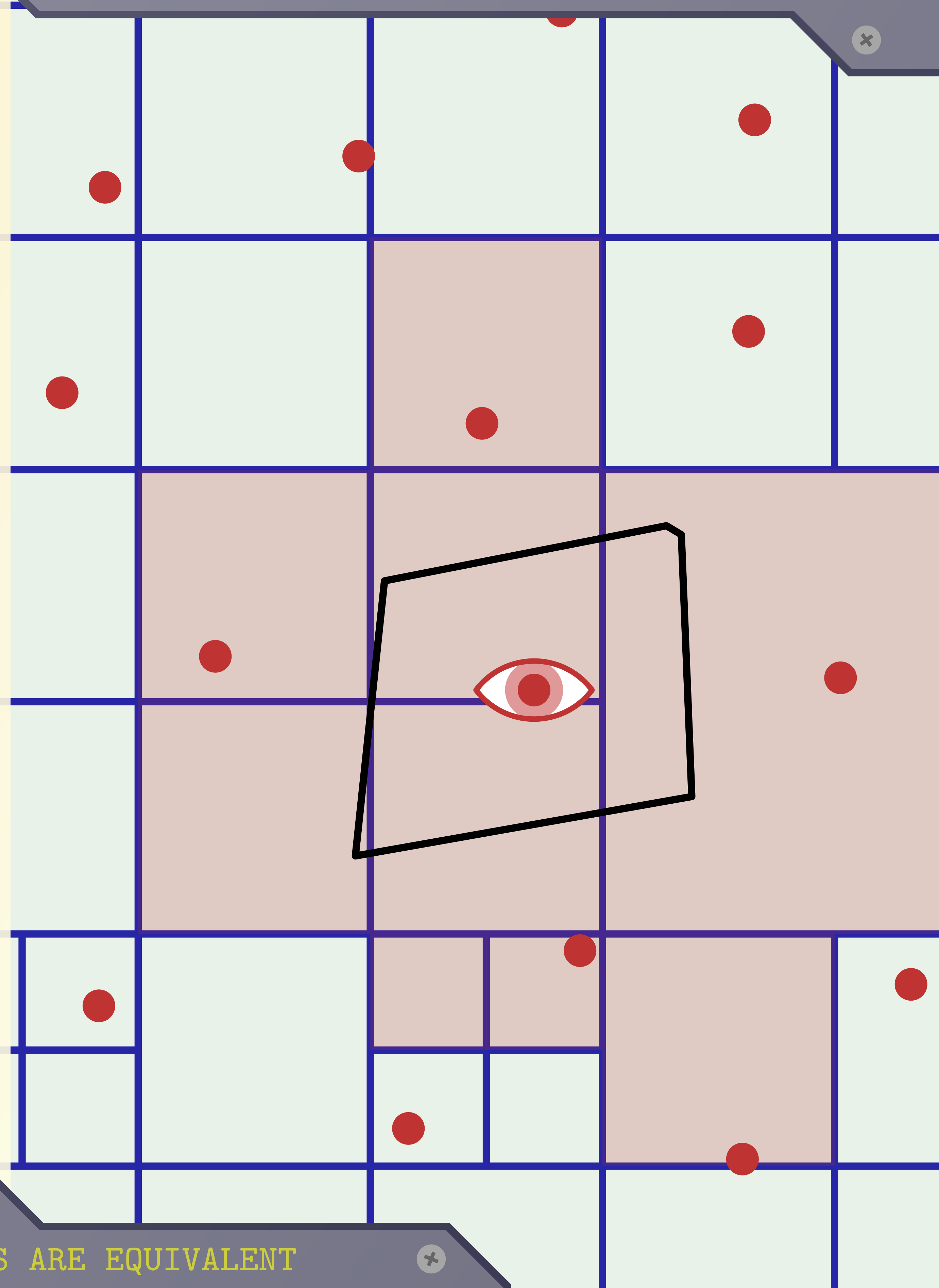
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



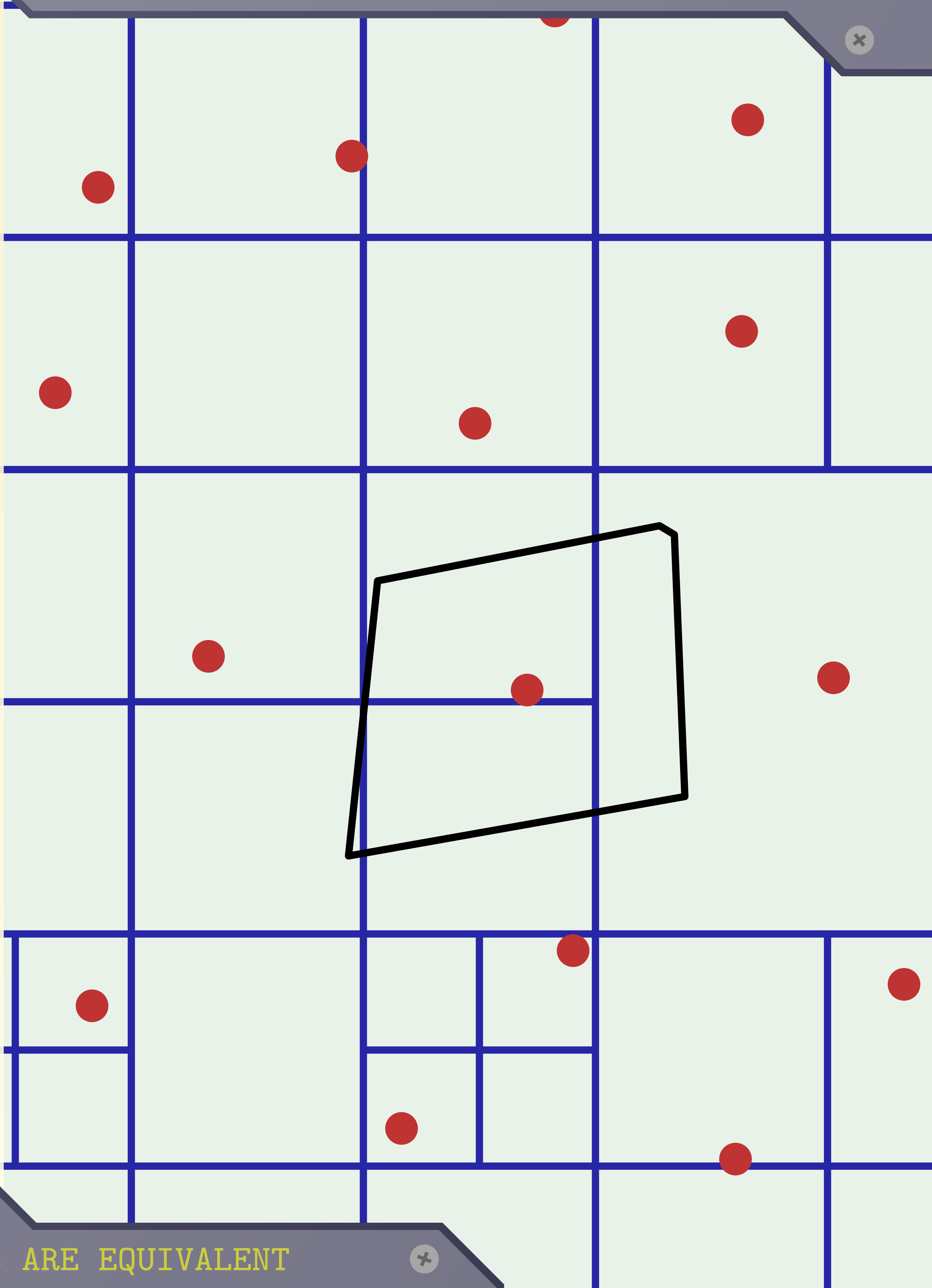
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



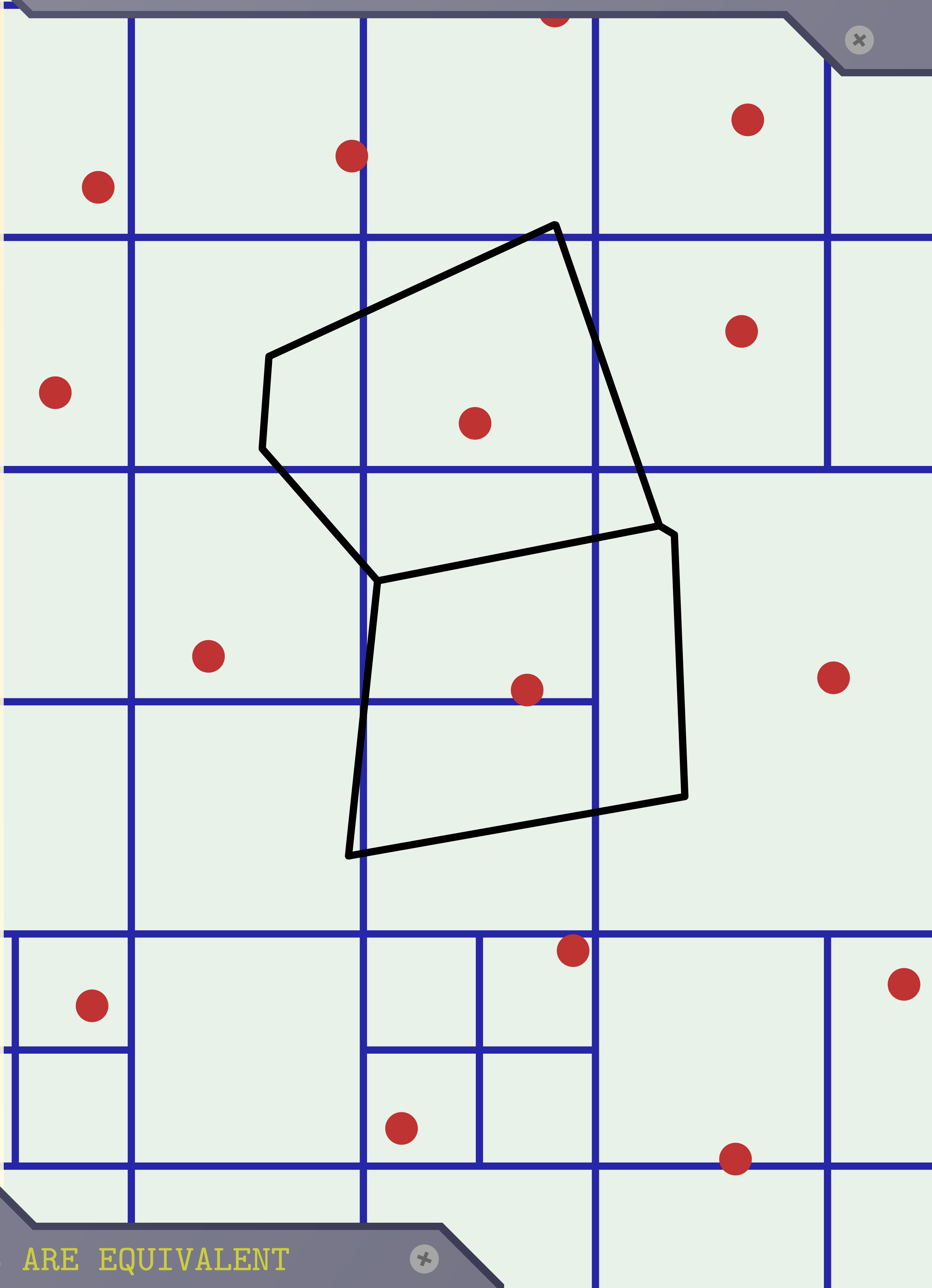
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



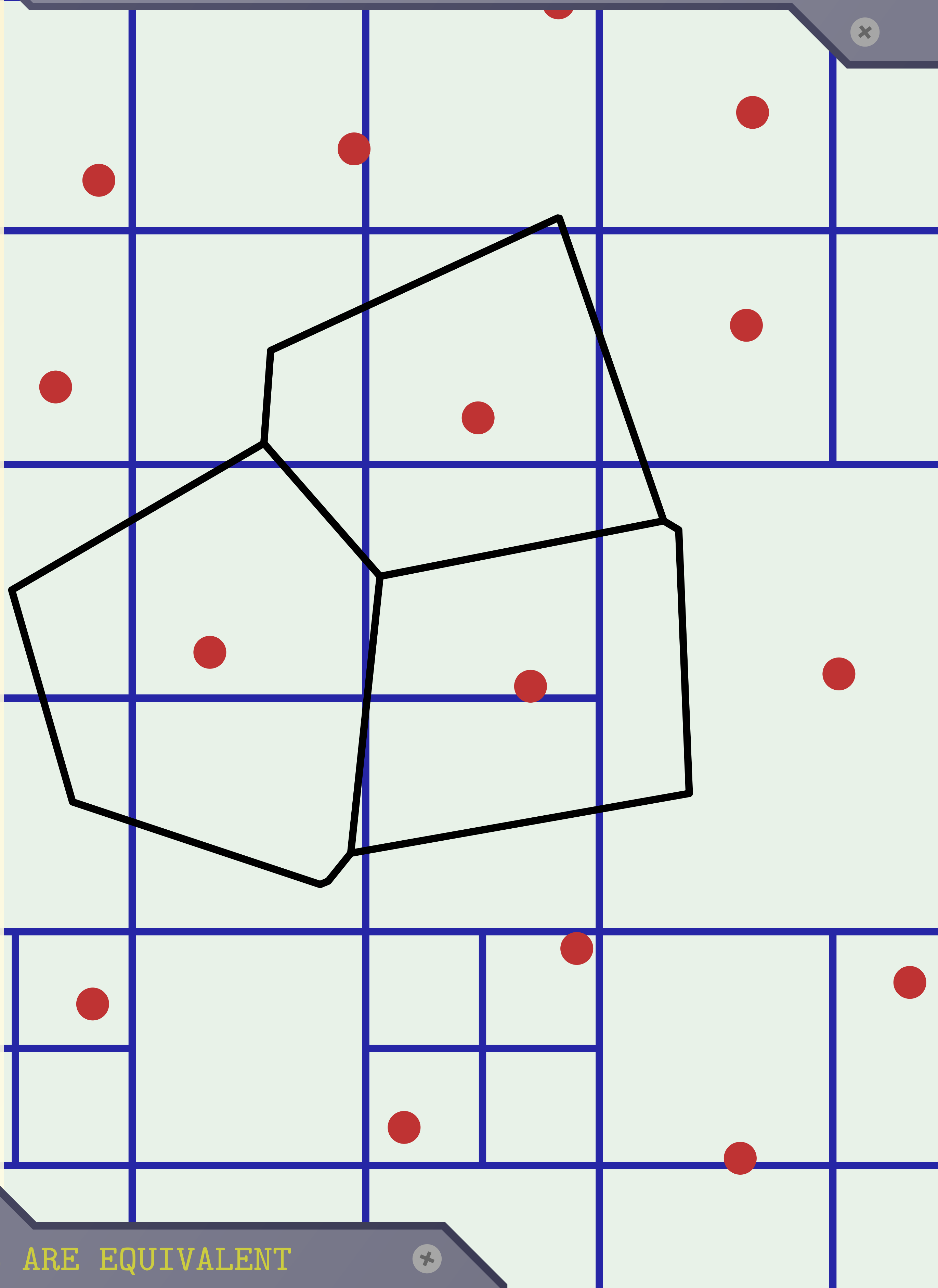
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



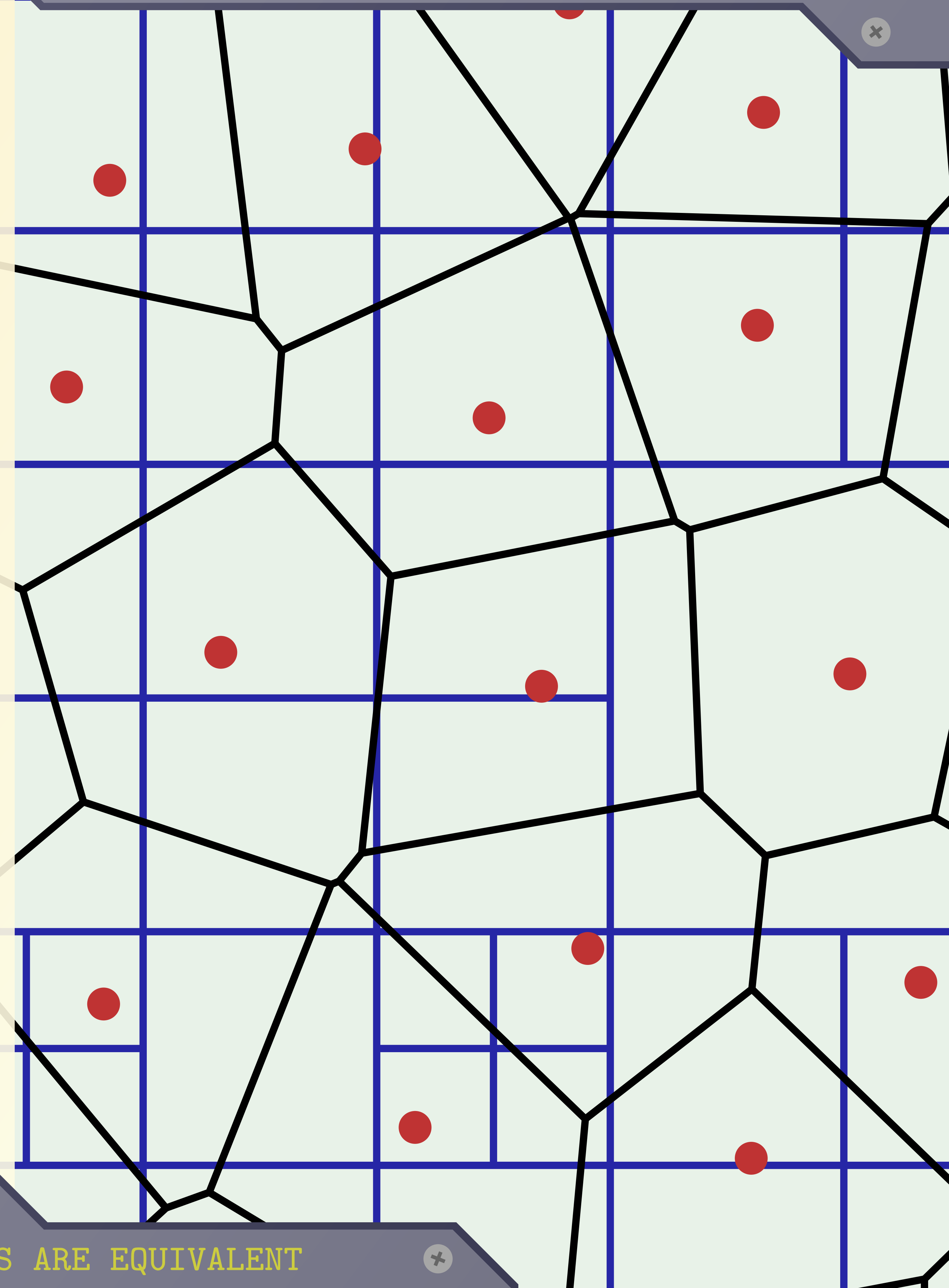
All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.



All these structures take $\Omega(n \log n)$ time to build.

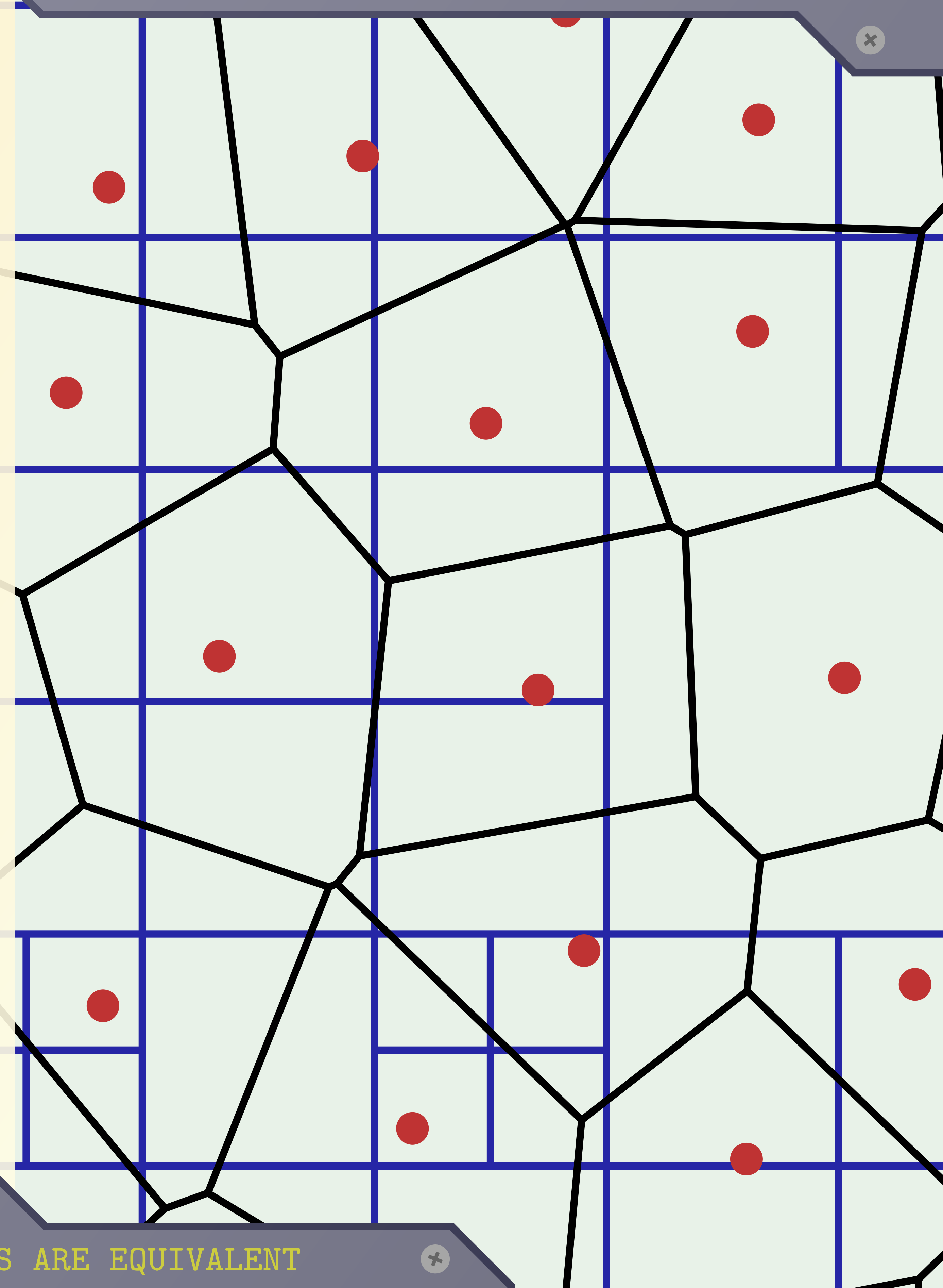
Intuitively, though, once you built one, you should be able to derive the others.



All these structures take $\Omega(n \log n)$ time to build.

Intuitively, though, once you built one, you should be able to derive the others.

Since information is *local*, such a conversion should take $O(n)$ time.



Unfortunately, not all point sets behave themselves.

Unfortunately, not all point sets behave themselves.

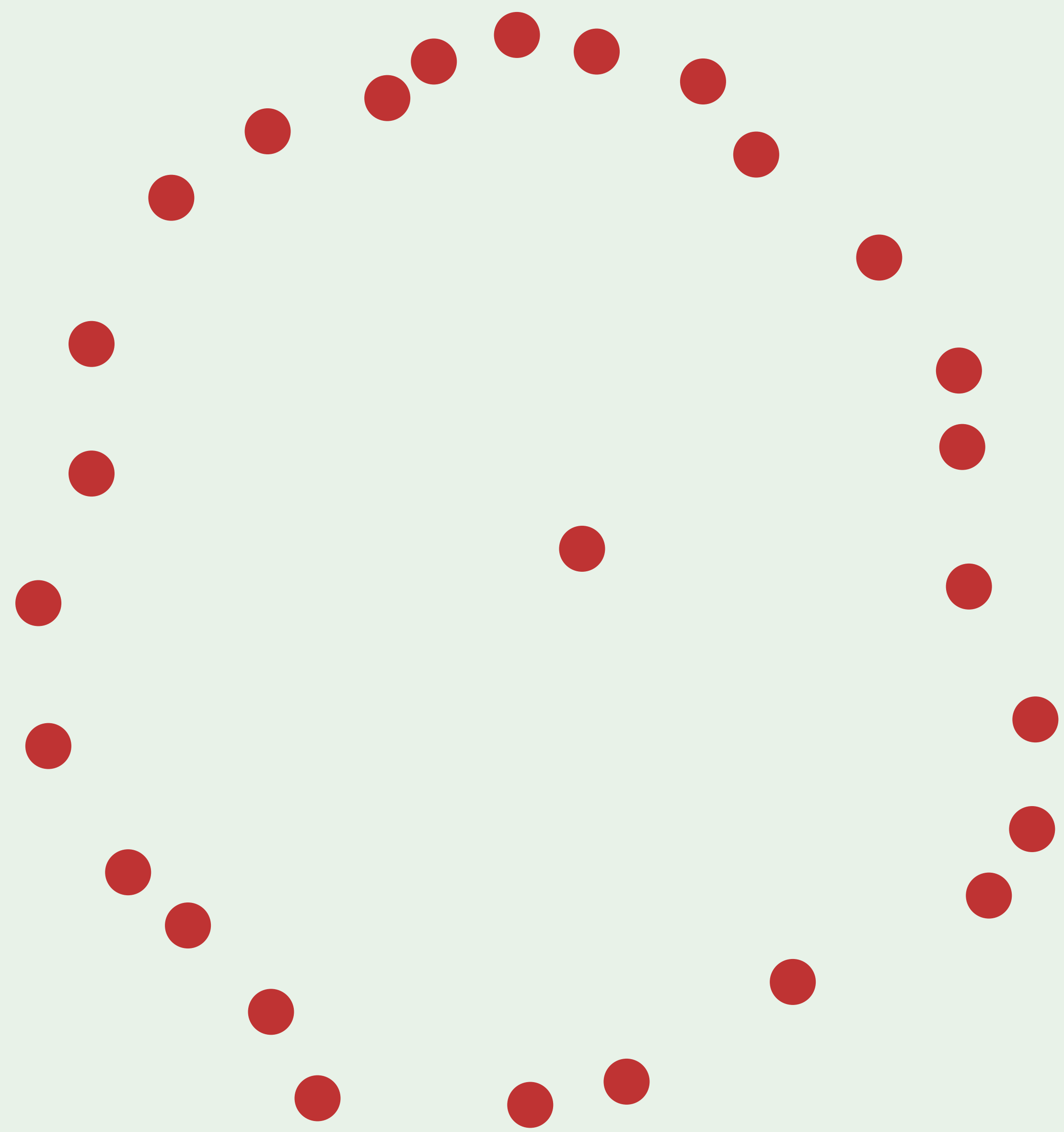
A single point may have more than $\Omega(1)$ neighbours.

Unfortunately, not all point sets behave themselves.

A single point may have more than $\Omega(1)$ neighbours.

Unfortunately, not all point sets behave themselves.

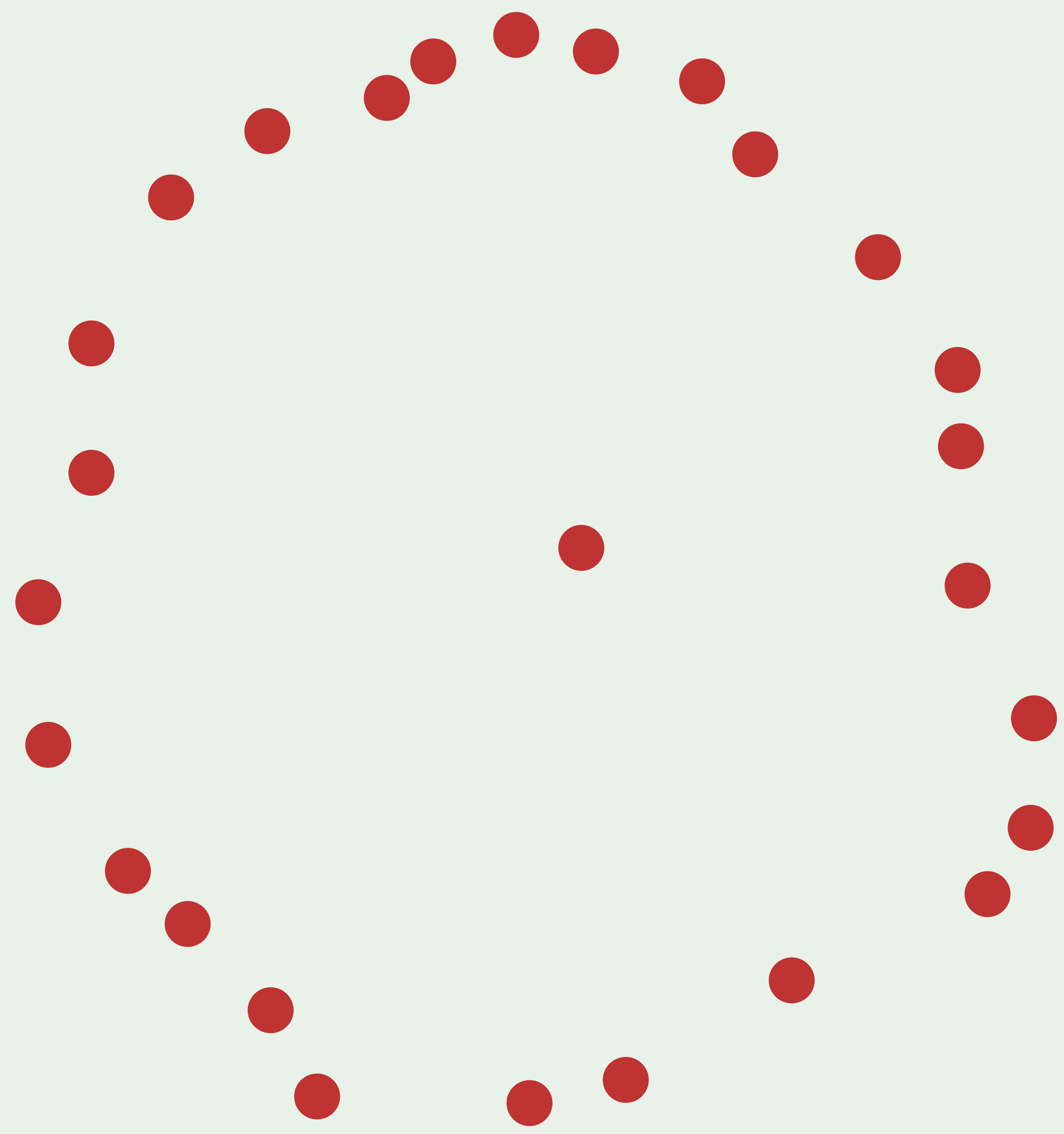
A single point may have more than $\Omega(1)$ neighbours.



Unfortunately, not all point sets behave themselves.

A single point may have more than $\Omega(1)$ neighbours.

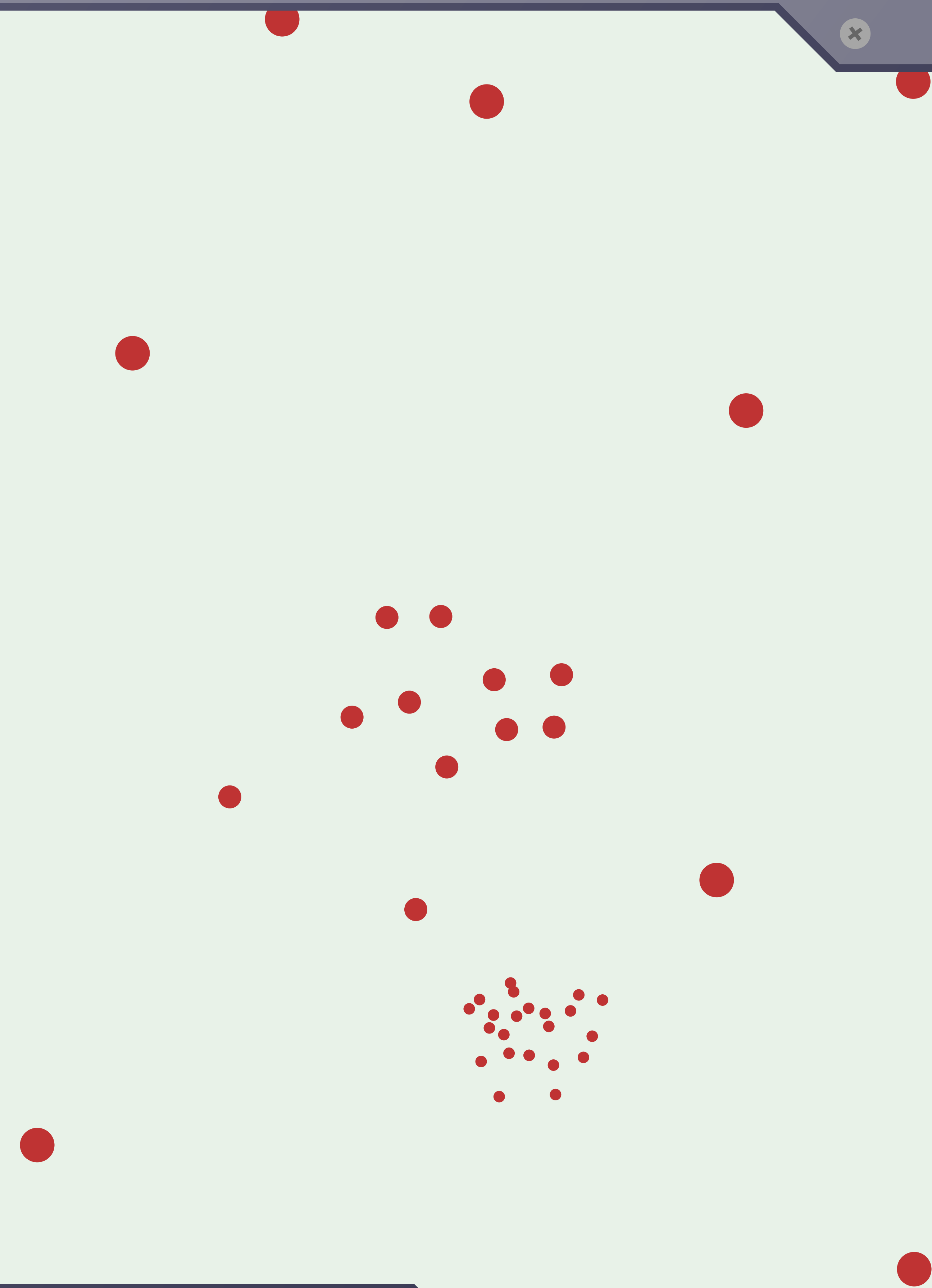
The scale of a point set need not be uniform.



Unfortunately, not all point sets behave themselves.

A single point may have more than $\Omega(1)$ neighbours.

The scale of a point set need not be uniform.

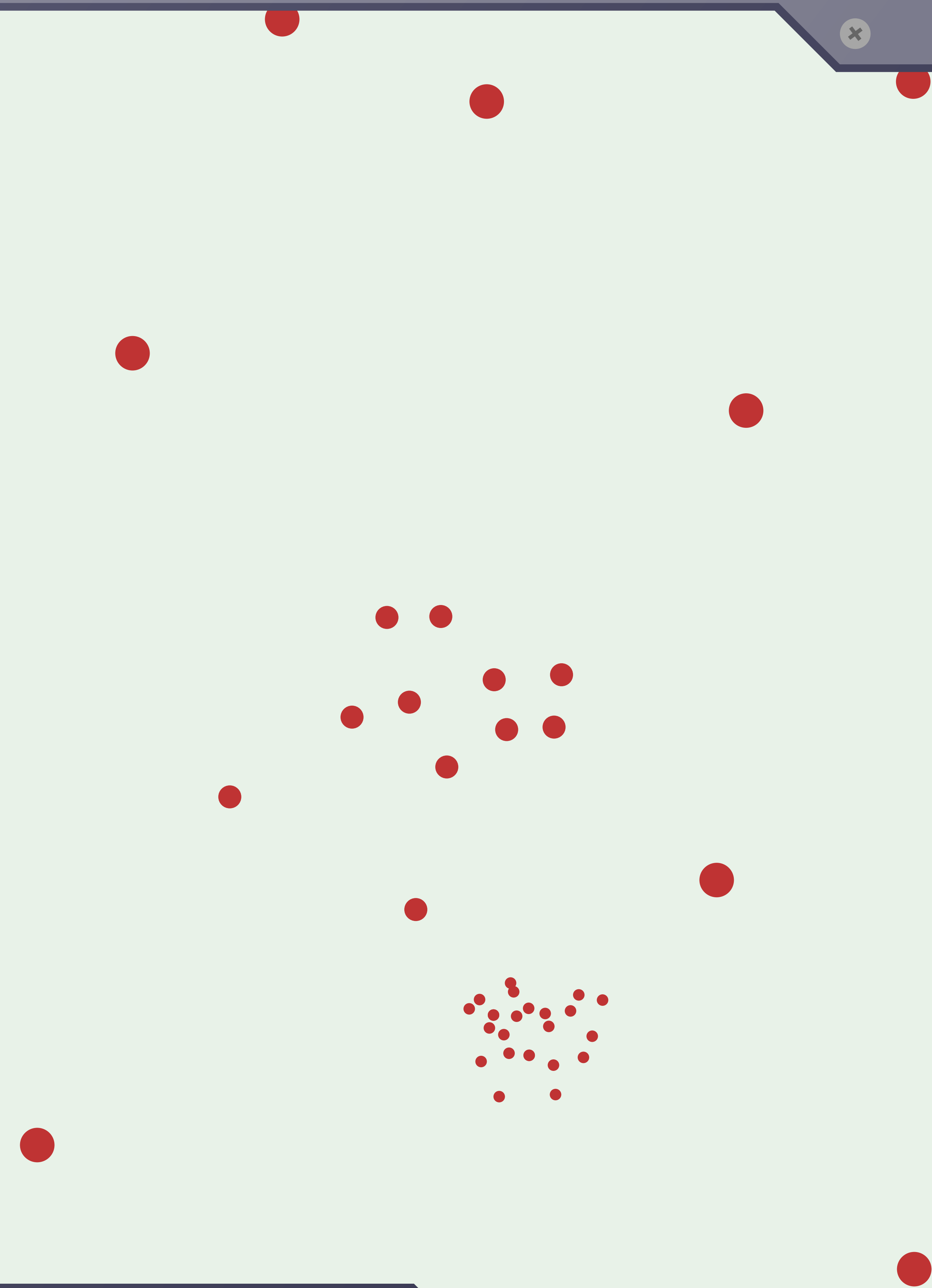


Unfortunately, not all point sets behave themselves.

A single point may have more than $\Omega(1)$ neighbours.

The scale of a point set need not be uniform.

What can we do?



Do we...

Do we...

... assume that in practice, point sets are well-behaved, and design useable algorithms that only work (or are only efficient) on certain classes of point sets?

Do we...

... assume that in practice, point sets are well-behaved, and design useable algorithms that only work (or are only efficient) on certain classes of point sets?

-OR-

Do we...

... assume that in practice, point sets are well-behaved, and design useable algorithms that only work (or are only efficient) on certain classes of point sets?

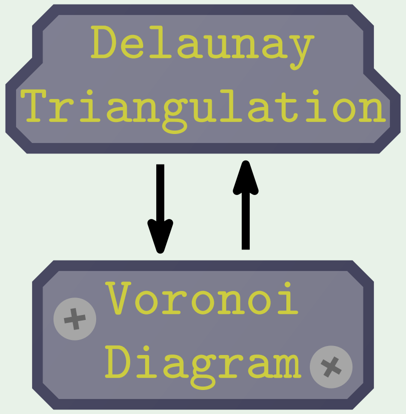
-OR-

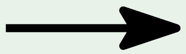
... insist on being general, and design devilishly complicated algorithms that nobody will ever use, but that are able to handle arbitrary point sets?

[Dirichlet, 1850]

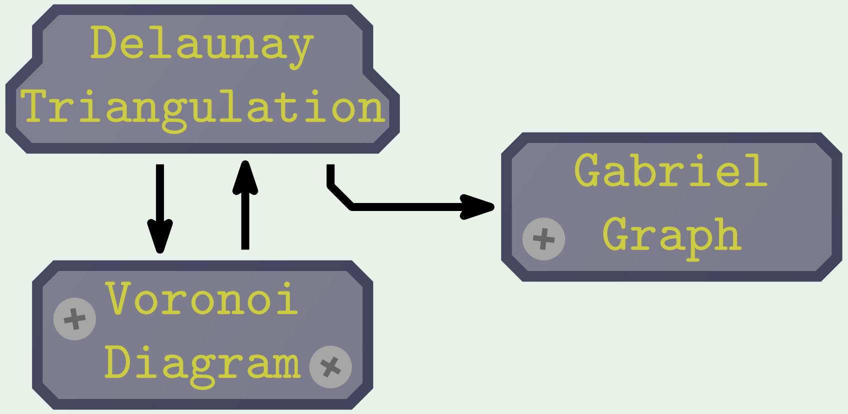
Voronoi Diagram

[Dirichlet, 1850]
[Delaunay, 1934]





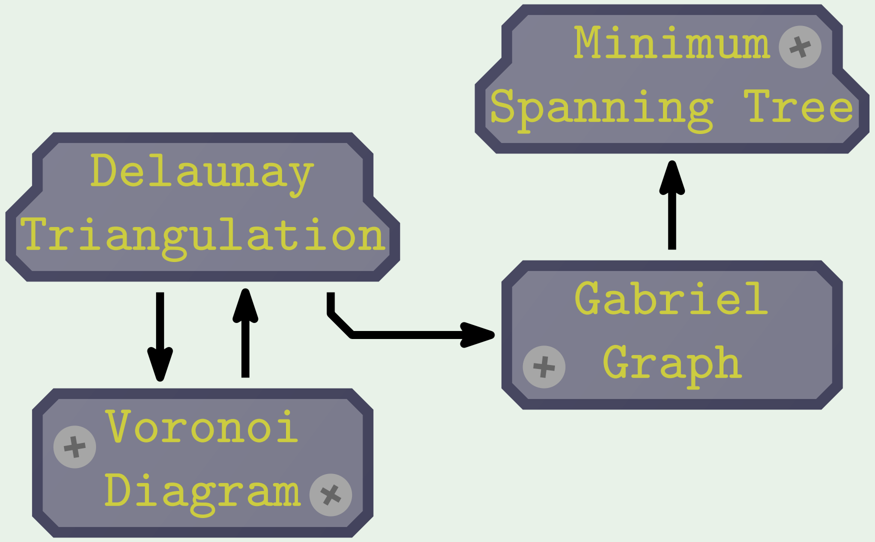
[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]



Deterministic linear time

[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]

[Matsui, 1995]

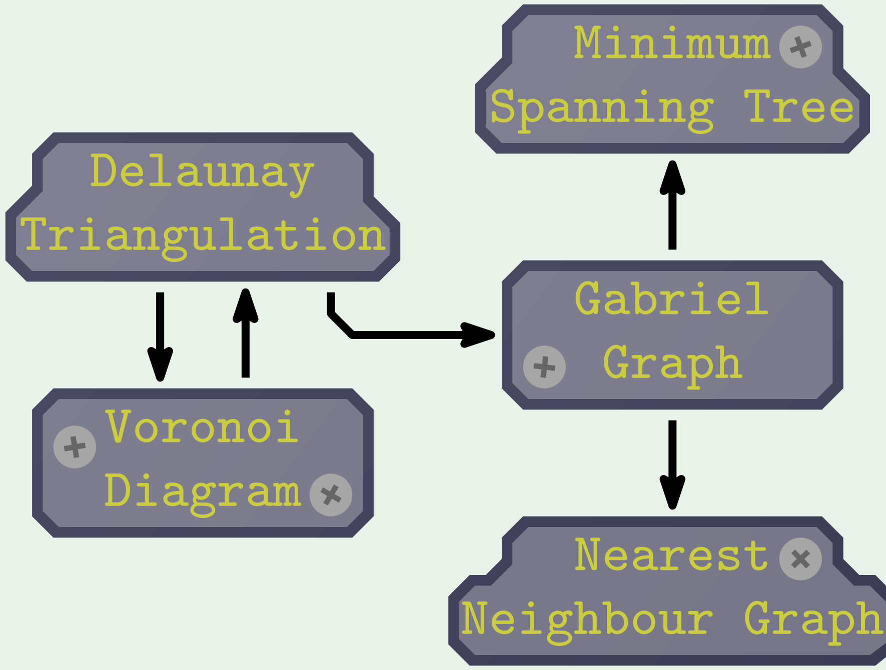


→ Deterministic linear time

[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]

[Preparata & Shamos, 1985]

[Matsui, 1995]

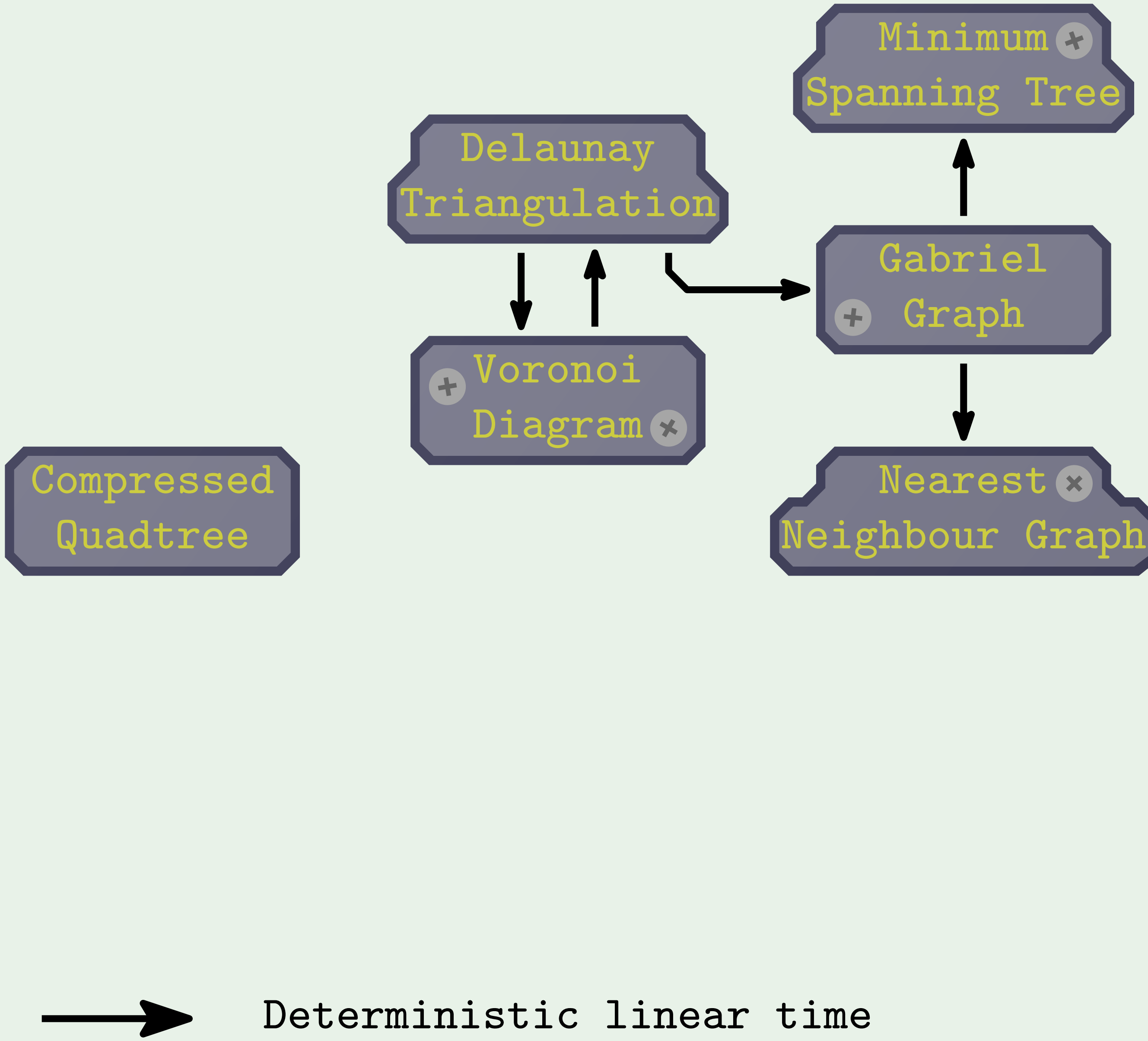



Deterministic linear time

A BRIEF HISTORY

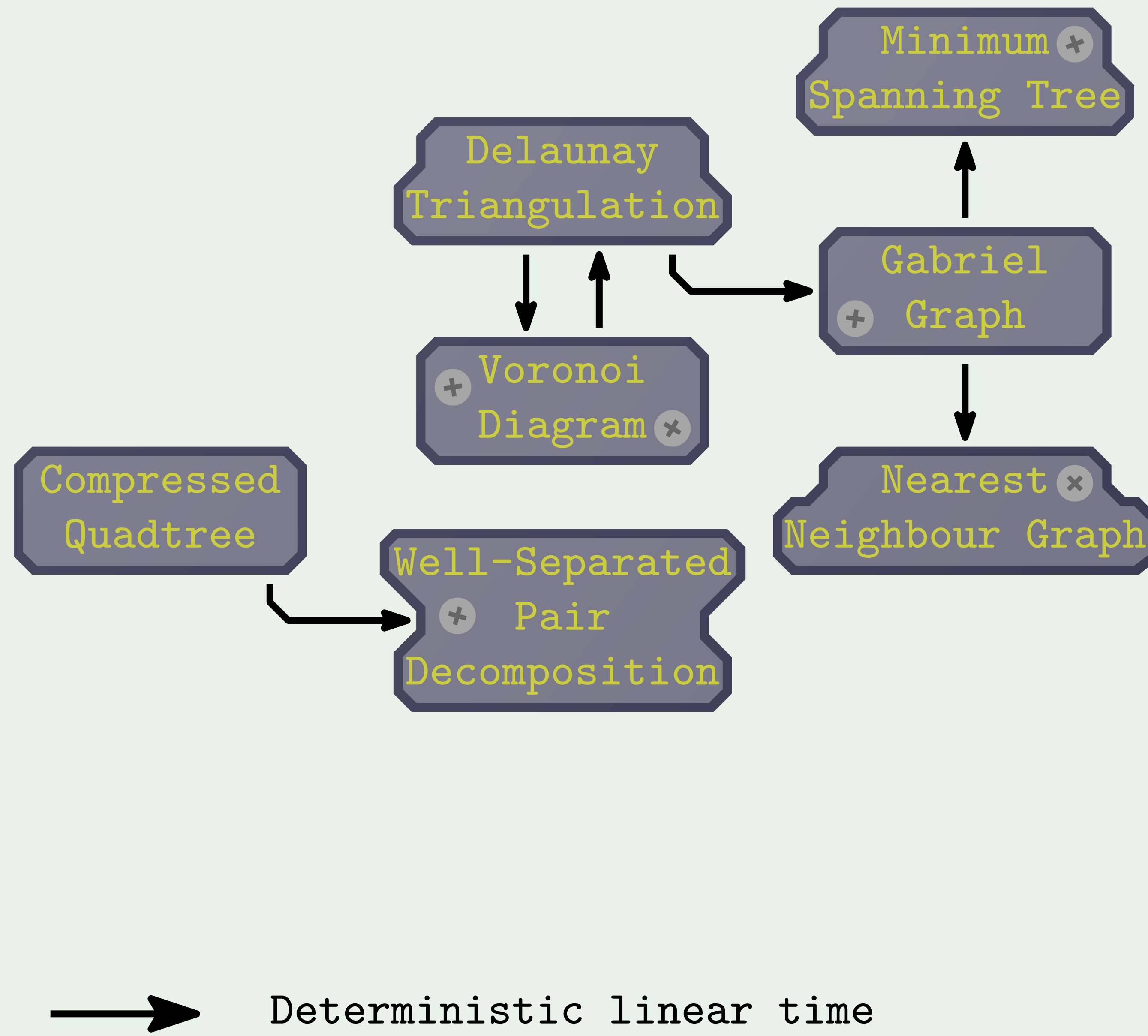
[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]
[Clarkson, 1983]
[Preparata & Shamos, 1985]

[Matsui, 1995]



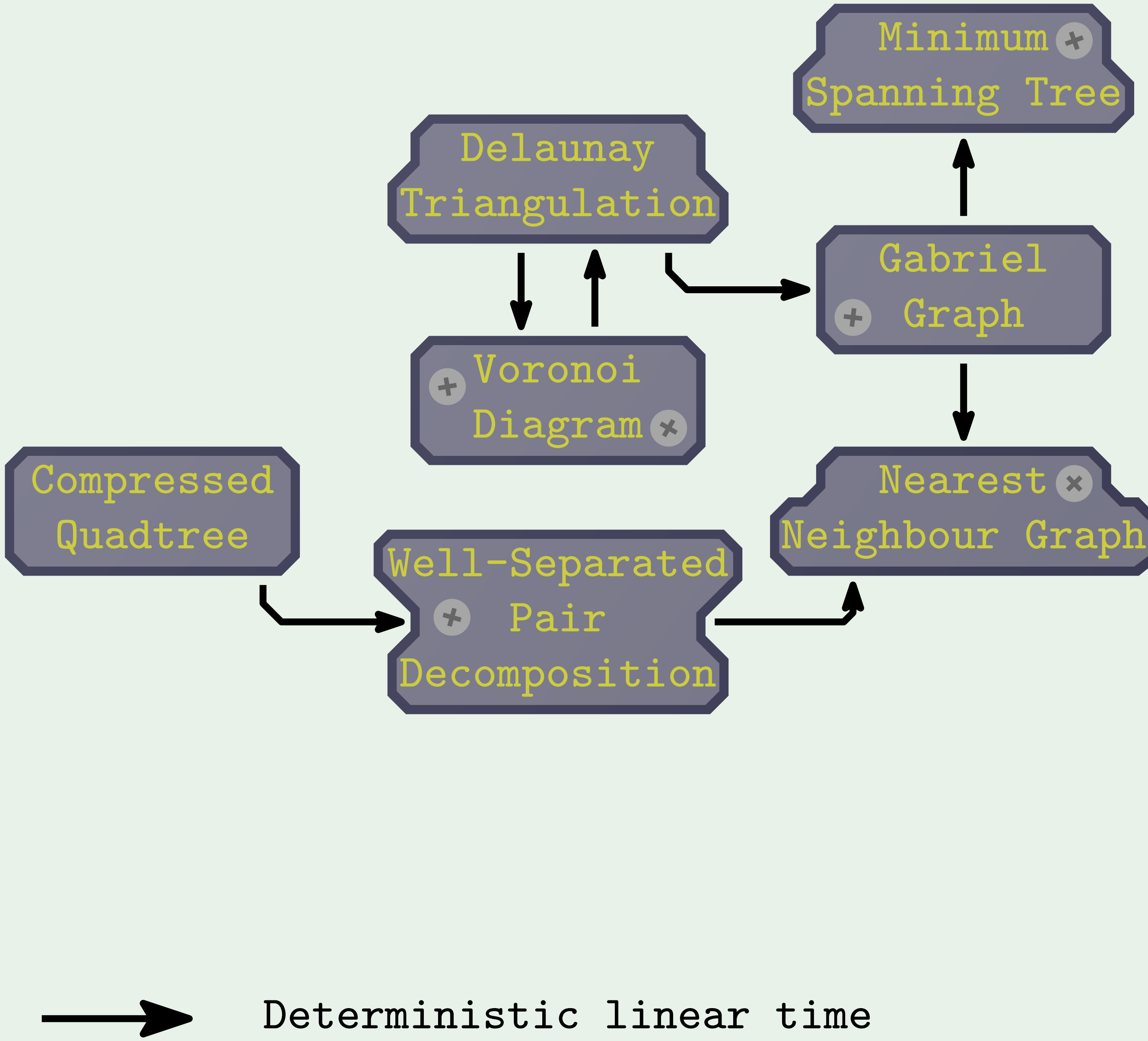
[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]
[Clarkson, 1983]
[Preparata & Shamos, 1985]

[Matsui, 1995]
[Callahan & Kosaraju, 1995]



[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]
[Clarkson, 1983]
[Preparata & Shamos, 1985]

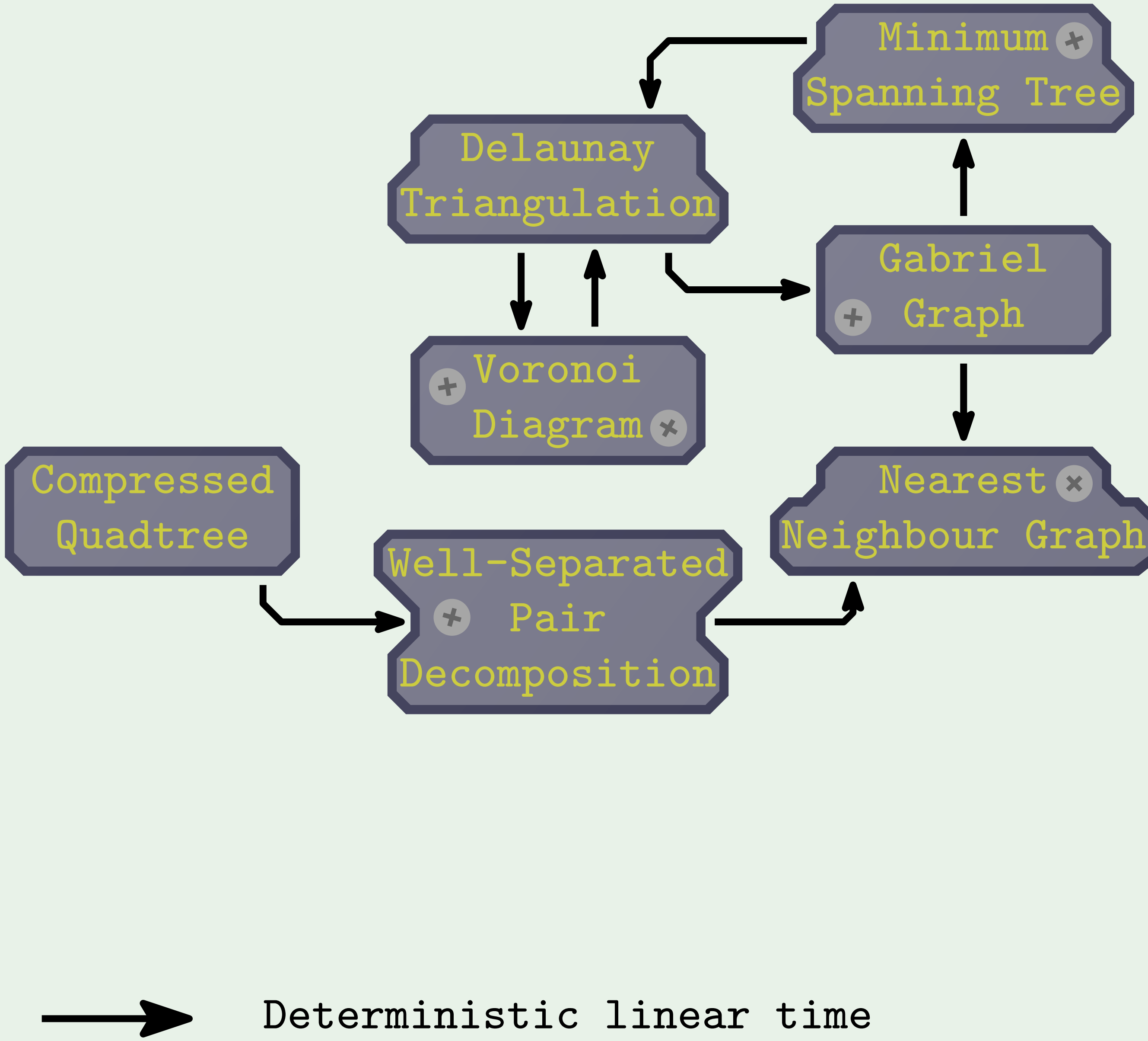
[Matsui, 1995]
[Callahan & Kosaraju, 1995]



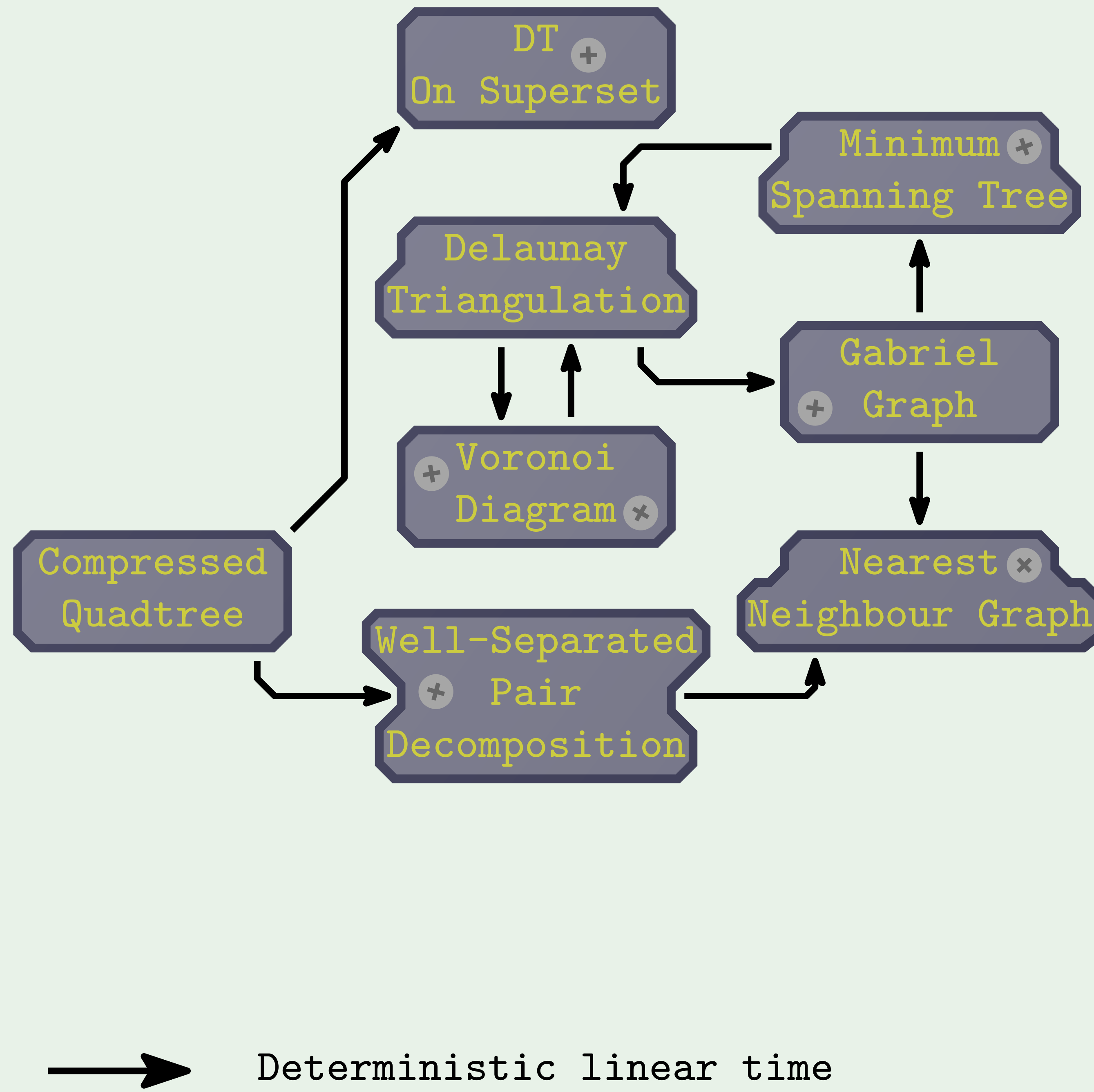
A BRIEF HISTORY

[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]
[Clarkson, 1983]
[Preparata & Shamos, 1985]

[Chazelle, 1991]
[Matsui, 1995]
[Callahan & Kosaraju, 1995]
[Chin & Wang, 1998]

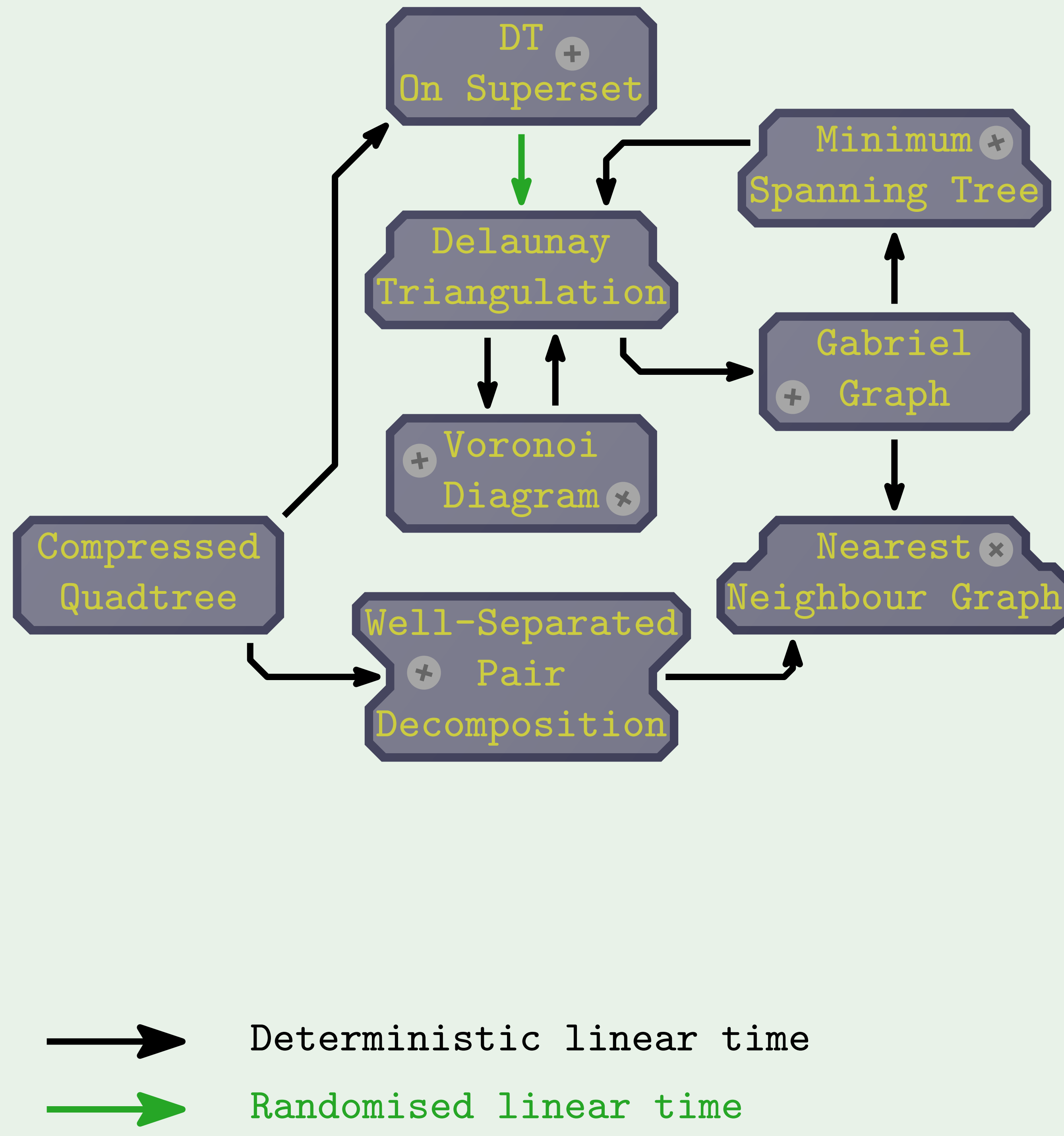


[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]
[Clarkson, 1983]
[Preparata & Shamos, 1985]
[Bern, Eppstein & Gilbert, 1990]
[Chazelle, 1991]
[Matsui, 1995]
[Callahan & Kosaraju, 1995]
[Chin & Wang, 1998]



[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]
[Clarkson, 1983]
[Preparata & Shamos, 1985]
[Bern, Eppstein & Gilbert, 1990]
[Chazelle, 1991]
[Matsui, 1995]
[Callahan & Kosaraju, 1995]
[Chin & Wang, 1998]

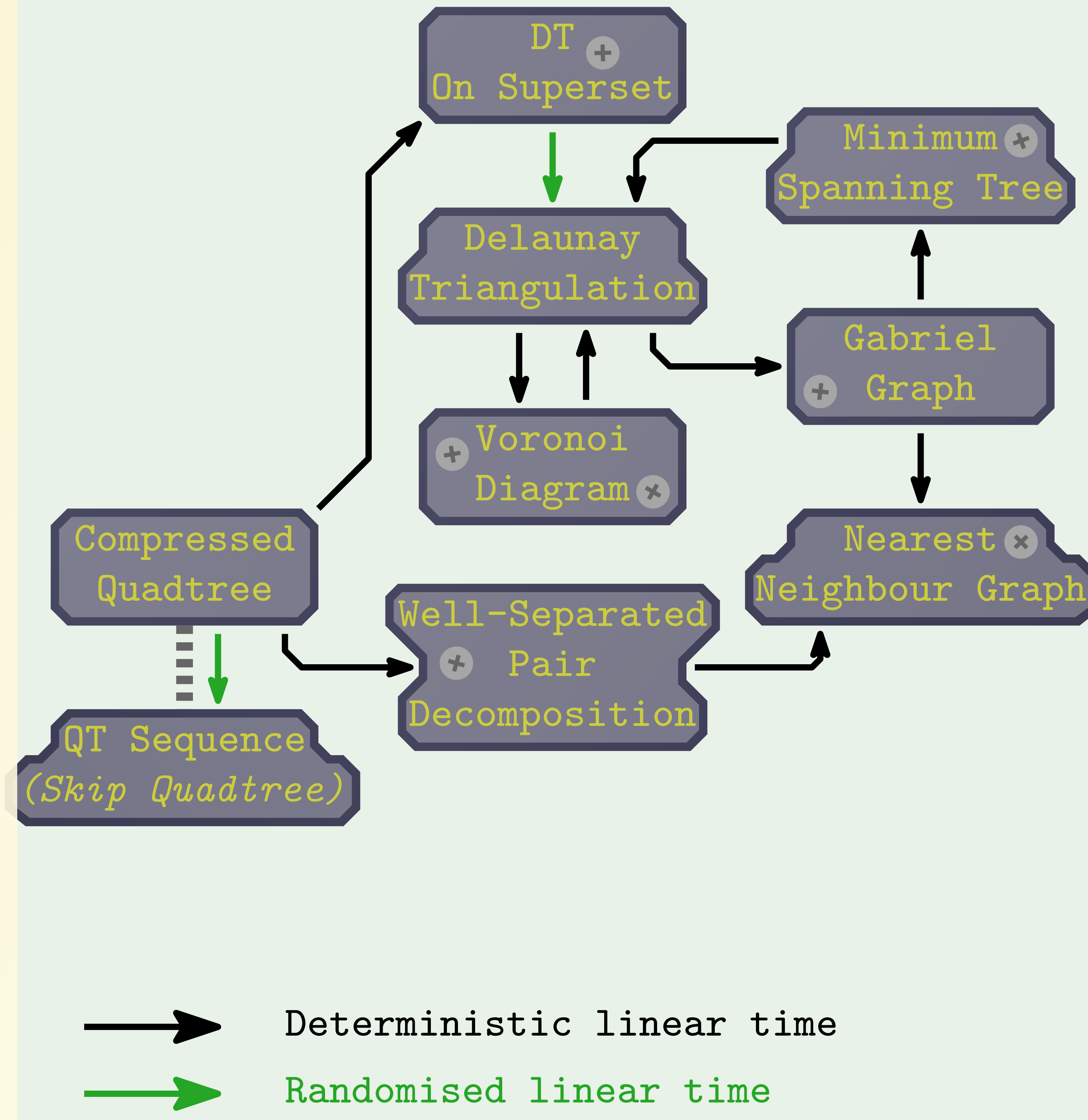
[Chazelle, Devillers, Hurtado, Mora, Sacristán & Teillaud, 2001]



A BRIEF HISTORY

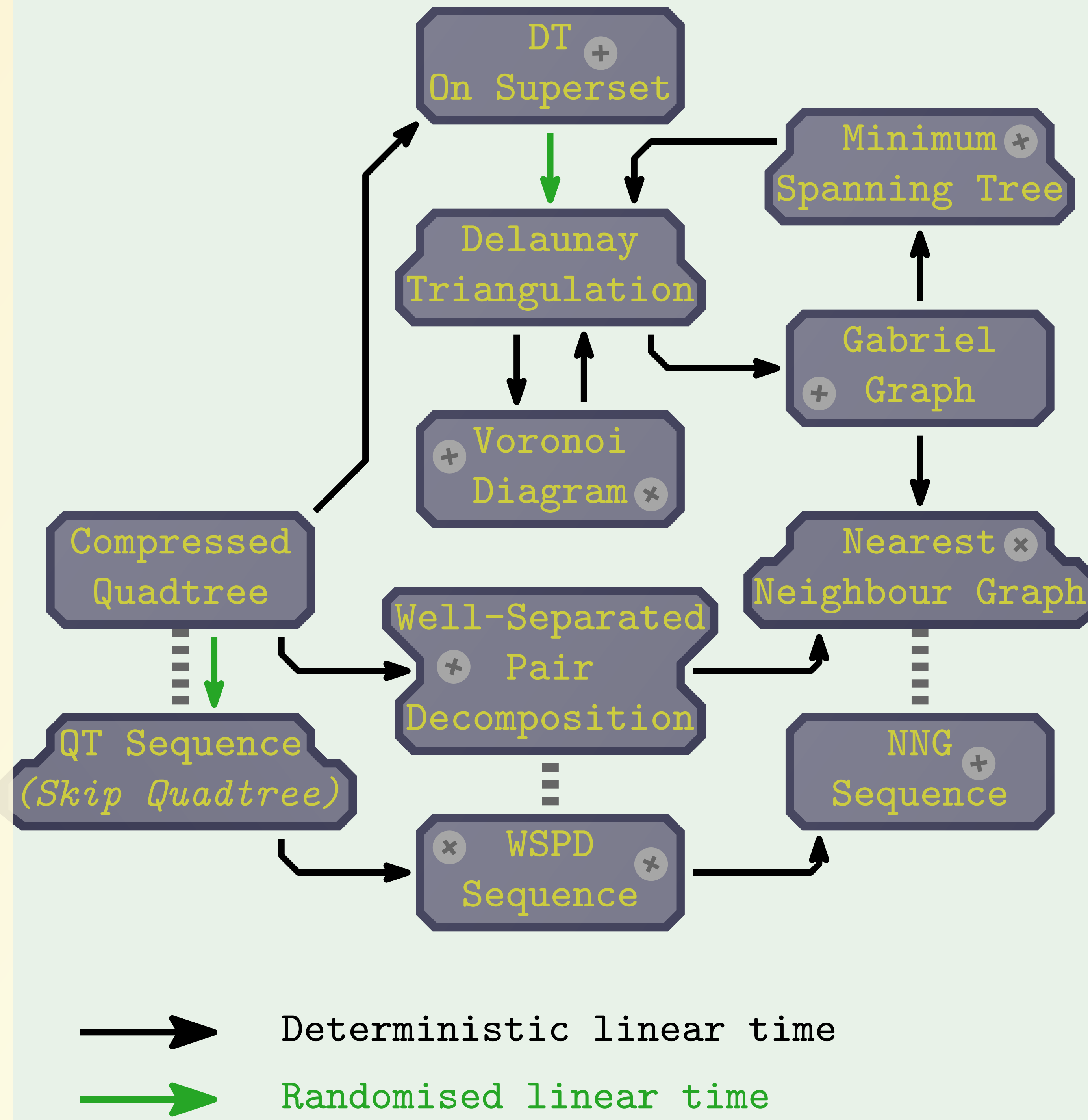
[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]
[Clarkson, 1983]
[Preparata & Shamos, 1985]
[Bern, Eppstein & Gilbert, 1990]
[Chazelle, 1991]
[Matsui, 1995]
[Callahan & Kosaraju, 1995]
[Chin & Wang, 1998]

[Chazelle, Devillers, Hurtado, Mora, Sacristán & Teillaud, 2001]
[Eppstein, Goodrich & Sun, 2005]



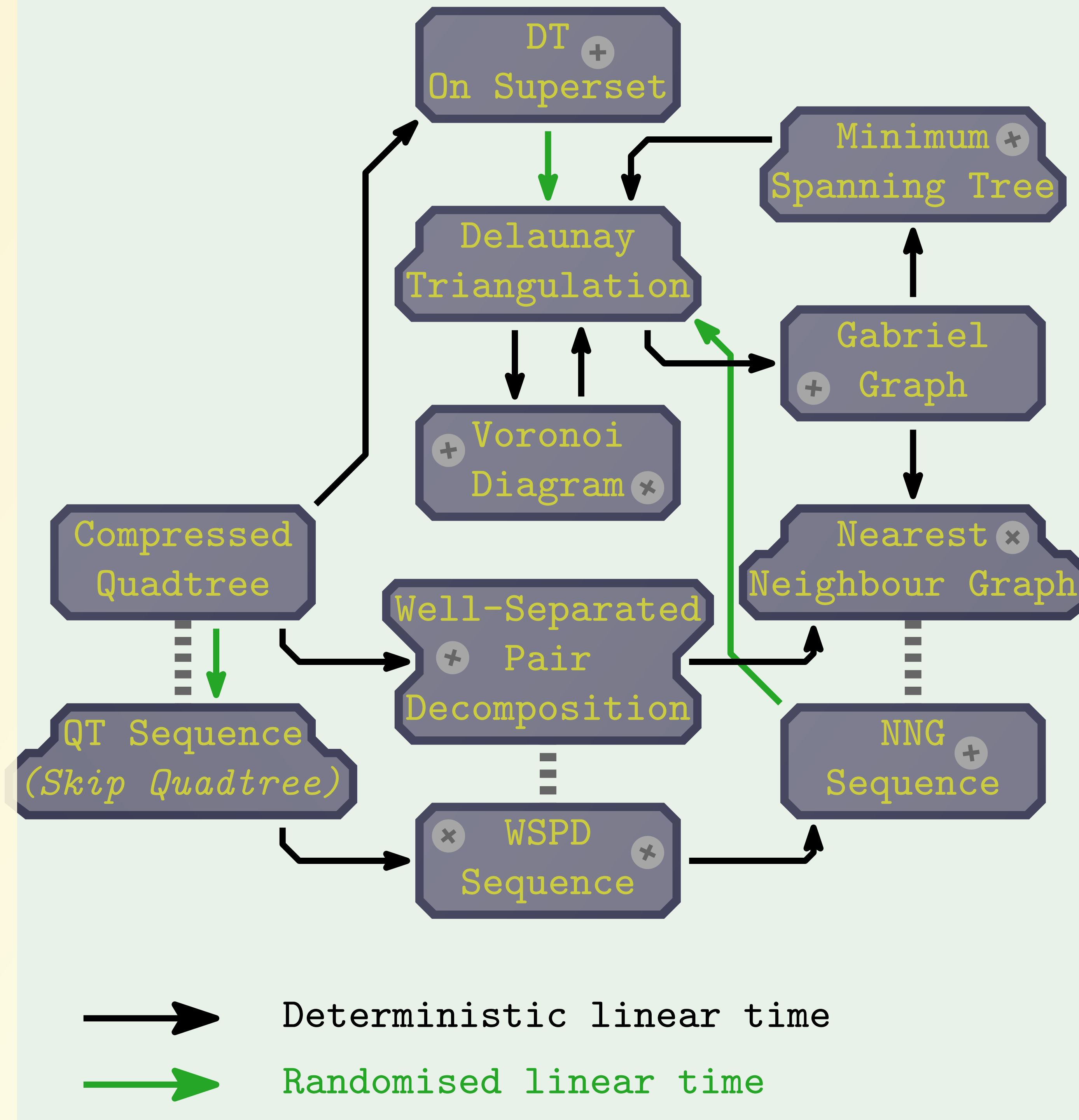
[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]
[Clarkson, 1983]
[Preparata & Shamos, 1985]
[Bern, Eppstein & Gilbert, 1990]
[Chazelle, 1991]
[Matsui, 1995]
[Callahan & Kosaraju, 1995]
[Chin & Wang, 1998]

[Chazelle, Devillers, Hurtado, Mora, Sacristán & Teillaud, 2001]
[Eppstein, Goodrich & Sun, 2005]



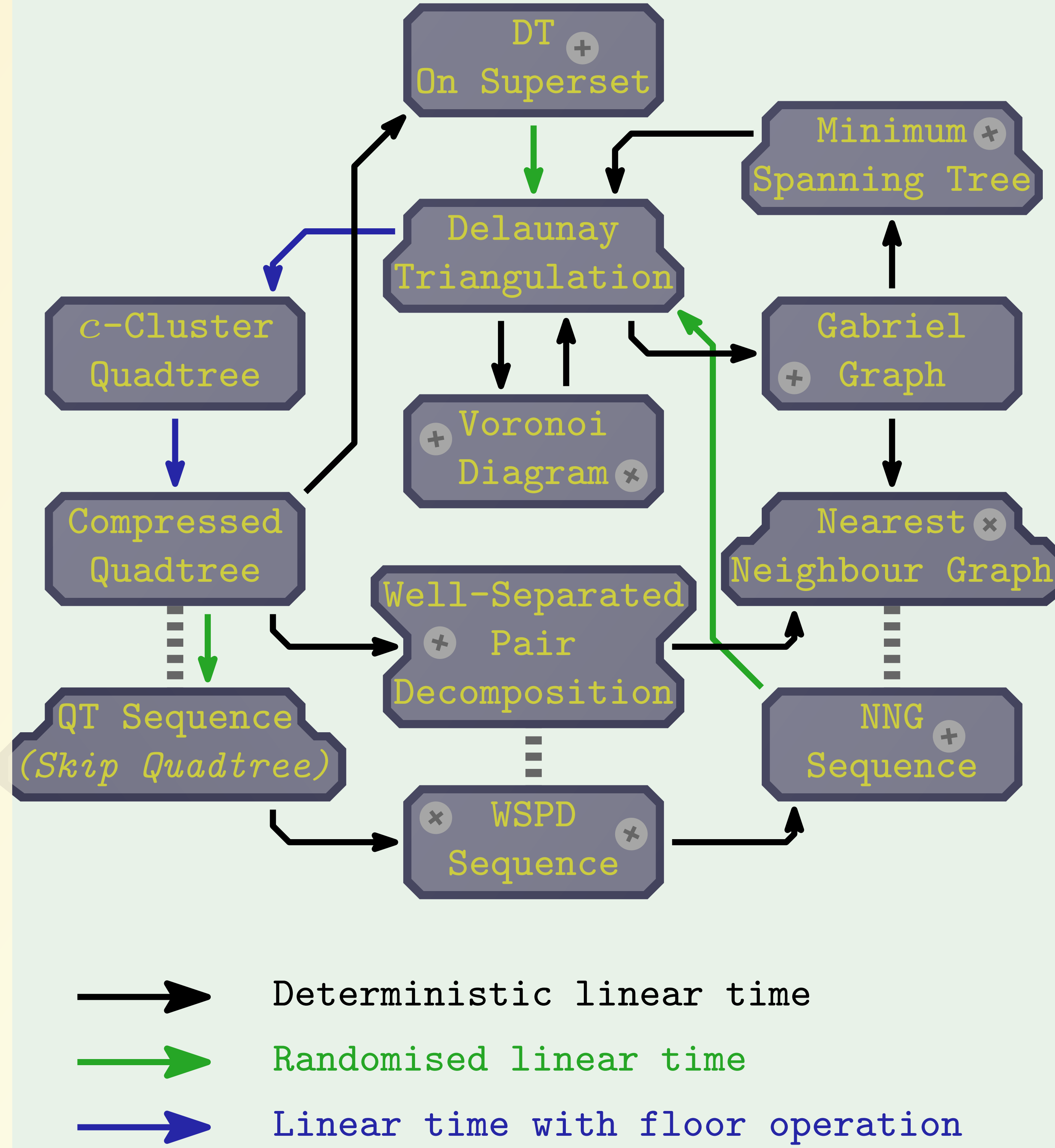
[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]
[Clarkson, 1983]
[Preparata & Shamos, 1985]
[Bern, Eppstein & Gilbert, 1990]
[Chazelle, 1991]
[Matsui, 1995]
[Callahan & Kosaraju, 1995]
[Chin & Wang, 1998]

[Chazelle, Devillers, Hurtado, Mora, Sacristán & Teillaud, 2001]
[Eppstein, Goodrich & Sun, 2005]
[Buchin & Mulzer, 2009]



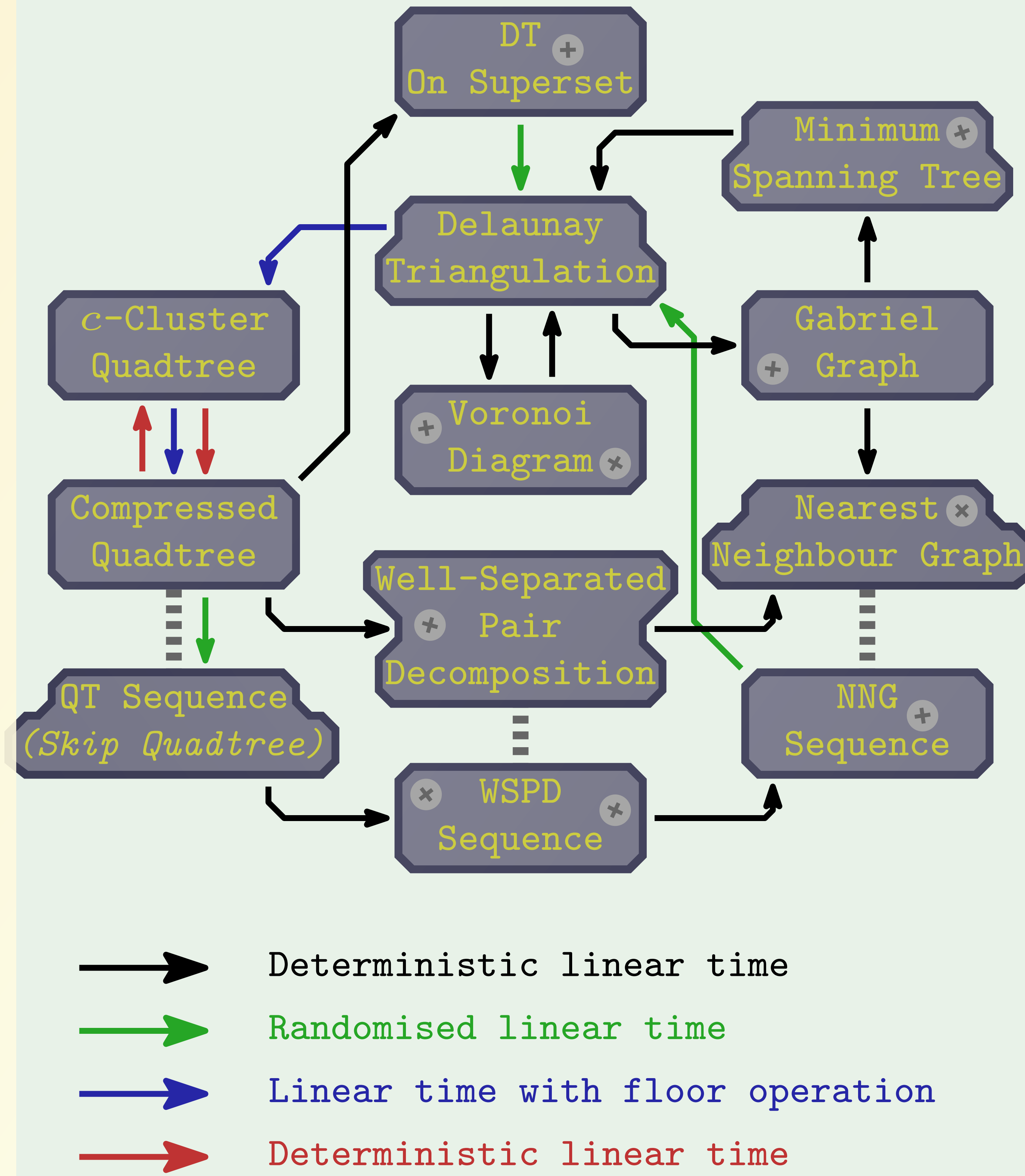
A BRIEF HISTORY

[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]
[Clarkson, 1983]
[Preparata & Shamos, 1985]
[Bern, Eppstein & Gilbert, 1990]
[Chazelle, 1991]
[Matsui, 1995]
[Callahan & Kosaraju, 1995]
[Chin & Wang, 1998]
[Krznaric & Levcopoulos, 1998]
[Chazelle, Devillers, Hurtado, Mora, Sacristán & Teillaud, 2001]
[Eppstein, Goodrich & Sun, 2005]
[Buchin & Mulzer, 2009]



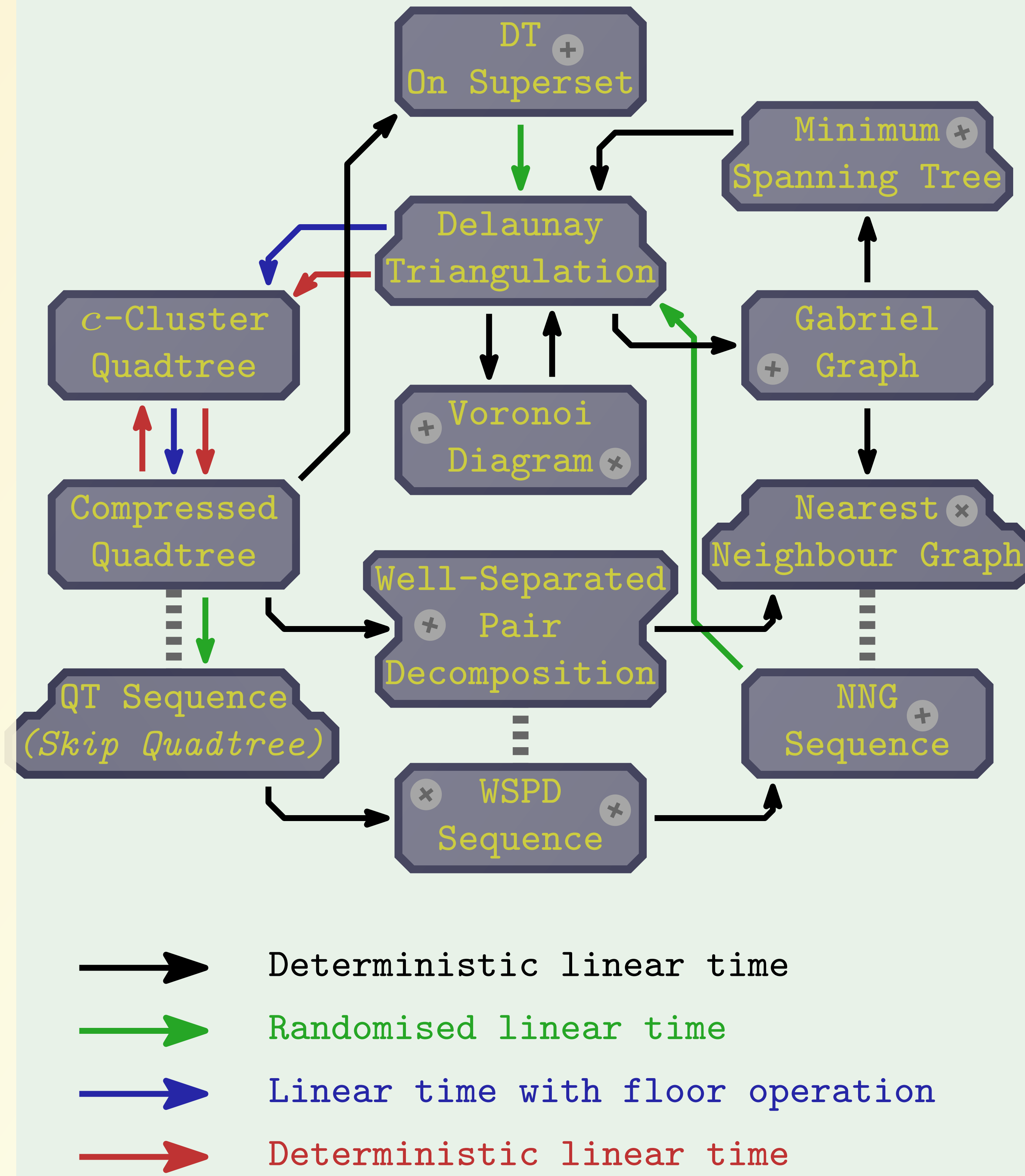
A BRIEF HISTORY

[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]
[Clarkson, 1983]
[Preparata & Shamos, 1985]
[Bern, Eppstein & Gilbert, 1990]
[Chazelle, 1991]
[Matsui, 1995]
[Callahan & Kosaraju, 1995]
[Chin & Wang, 1998]
[Krznaric & Levcopoulos, 1998]
[Chazelle, Devillers, Hurtado, Mora, Sacristán & Teillaud, 2001]
[Eppstein, Goodrich & Sun, 2005]
[Buchin & Mulzer, 2009]
[Us, 2011]

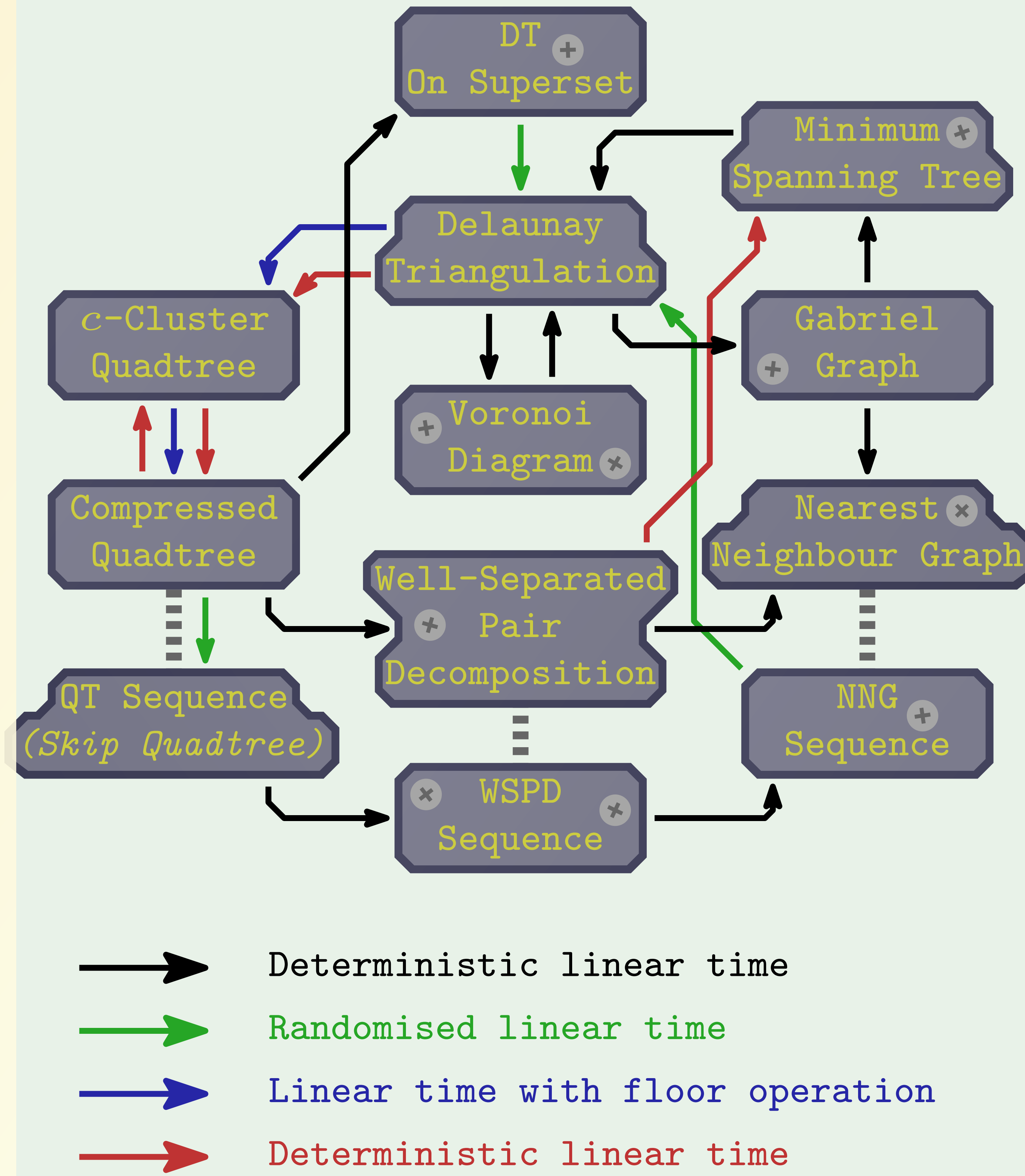


A BRIEF HISTORY

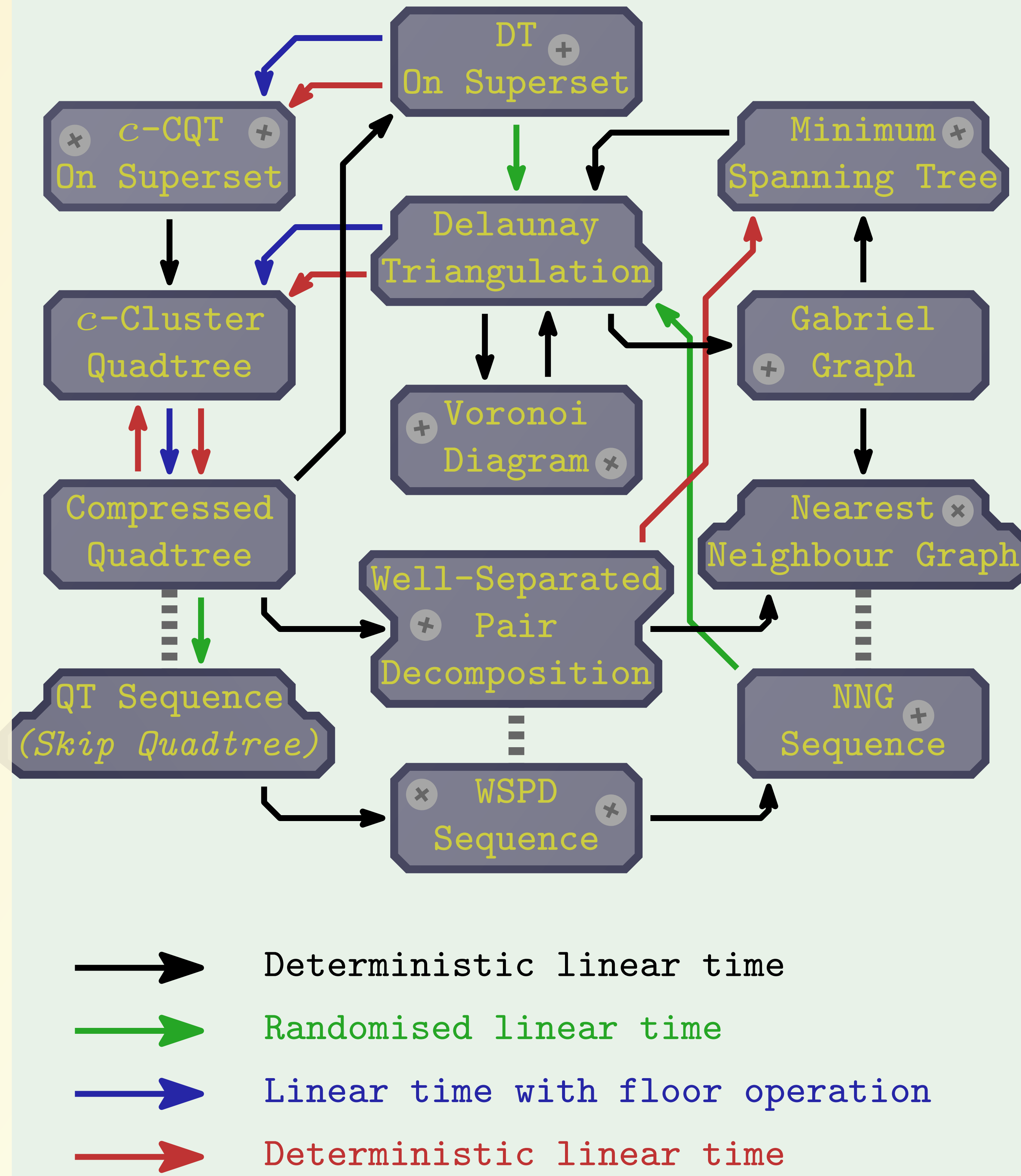
[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]
[Clarkson, 1983]
[Preparata & Shamos, 1985]
[Bern, Eppstein & Gilbert, 1990]
[Chazelle, 1991]
[Matsui, 1995]
[Callahan & Kosaraju, 1995]
[Chin & Wang, 1998]
[Krznaric & Levcopoulos, 1998]
[Chazelle, Devillers, Hurtado, Mora, Sacristán & Teillaud, 2001]
[Eppstein, Goodrich & Sun, 2005]
[Buchin & Mulzer, 2009]
[Us, 2011]



[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]
[Clarkson, 1983]
[Preparata & Shamos, 1985]
[Bern, Eppstein & Gilbert, 1990]
[Chazelle, 1991]
[Matsui, 1995]
[Callahan & Kosaraju, 1995]
[Chin & Wang, 1998]
[Krznaric & Levcopoulos, 1998]
[Chazelle, Devillers, Hurtado, Mora, Sacristán & Teillaud, 2001]
[Eppstein, Goodrich & Sun, 2005]
[Buchin & Mulzer, 2009]
[Us, 2011]



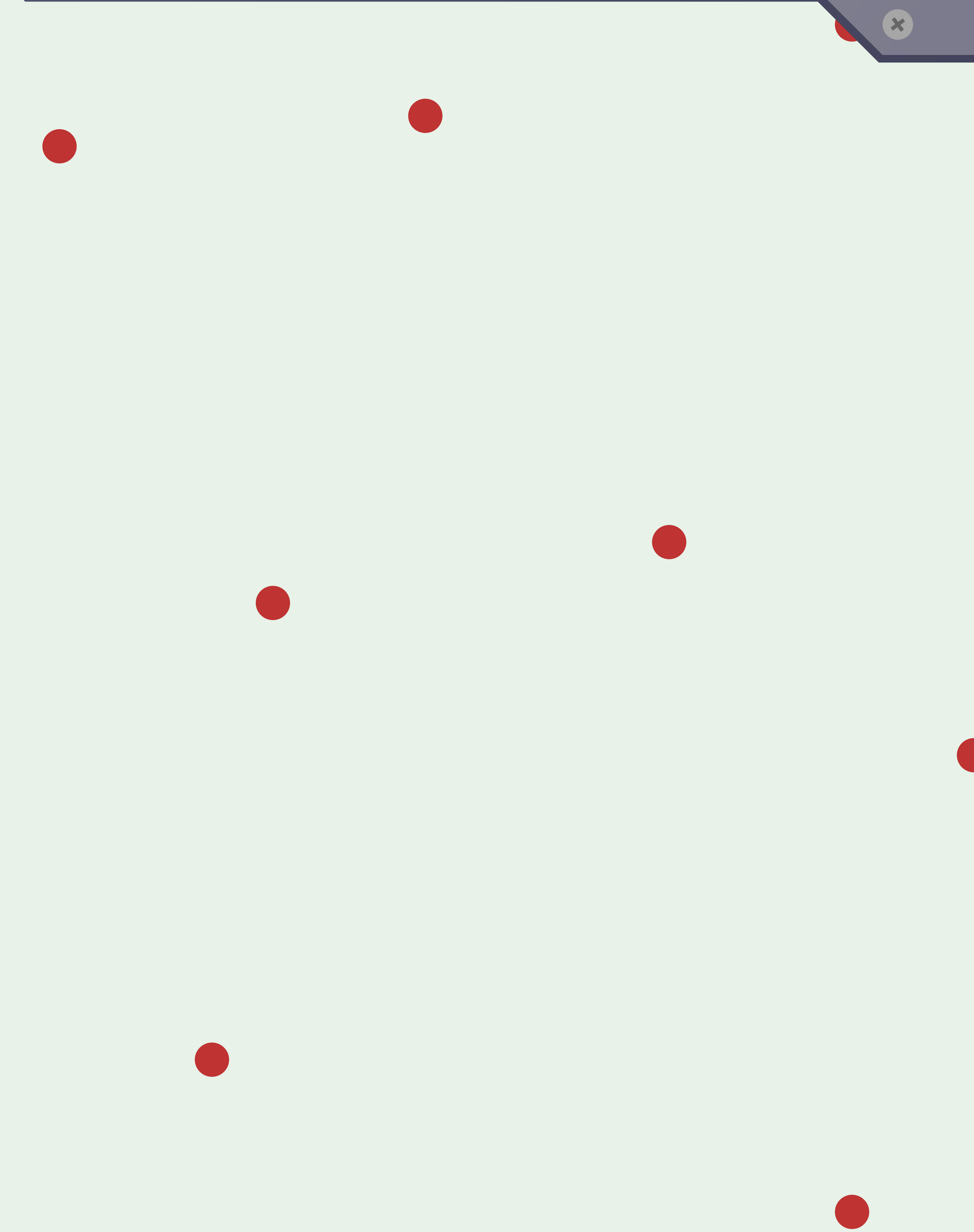
[Dirichlet, 1850]
[Delaunay, 1934]
[Gabriel, 1969]
[Clarkson, 1983]
[Preparata & Shamos, 1985]
[Bern, Eppstein & Gilbert, 1990]
[Chazelle, 1991]
[Matsui, 1995]
[Callahan & Kosaraju, 1995]
[Chin & Wang, 1998]
[Krznaric & Levcopoulos, 1998]
[Chazelle, Devillers, Hurtado, Mora, Sacristán & Teillaud, 2001]
[Eppstein, Goodrich & Sun, 2005]
[Buchin & Mulzer, 2009]
[Us, 2011]



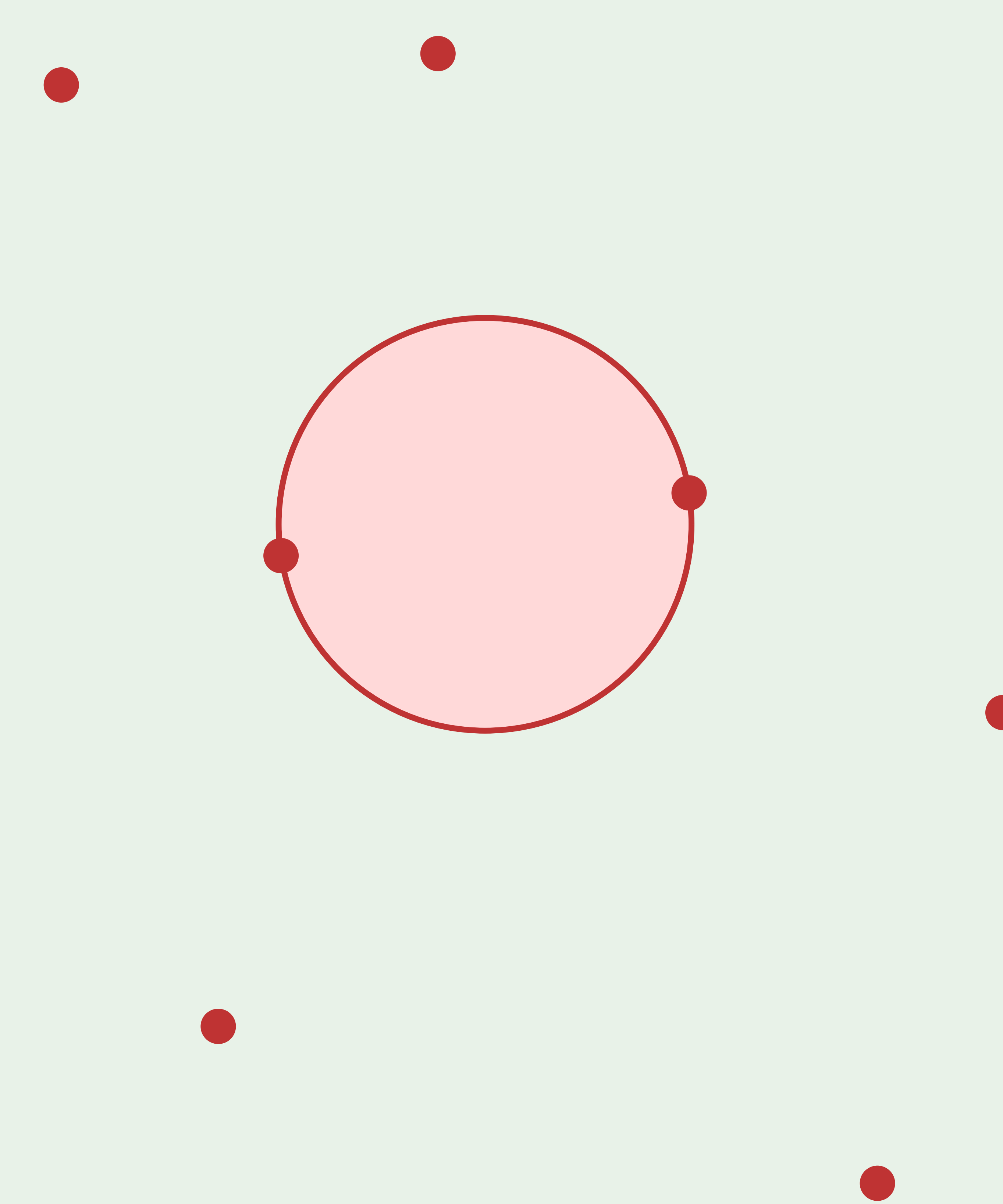
PART II *PRELIMINARIES*

The *Delaunay triangulation (DT)* is a triangulation D of a point set P that has an edge between two points if there is an empty circle through these points.

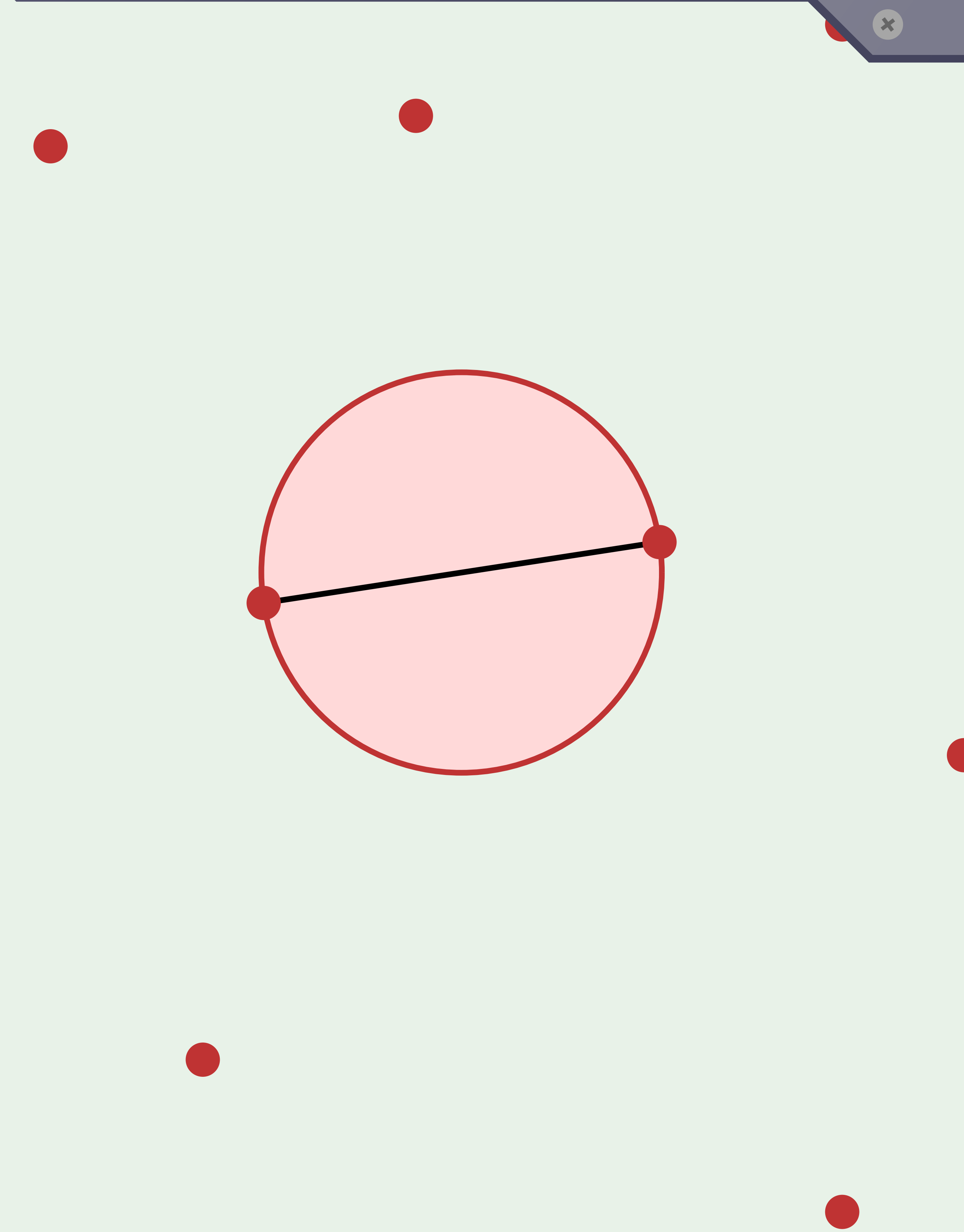
The *Delaunay triangulation (DT)* is a triangulation D of a point set P that has an edge between two points if there is an empty circle through these points.



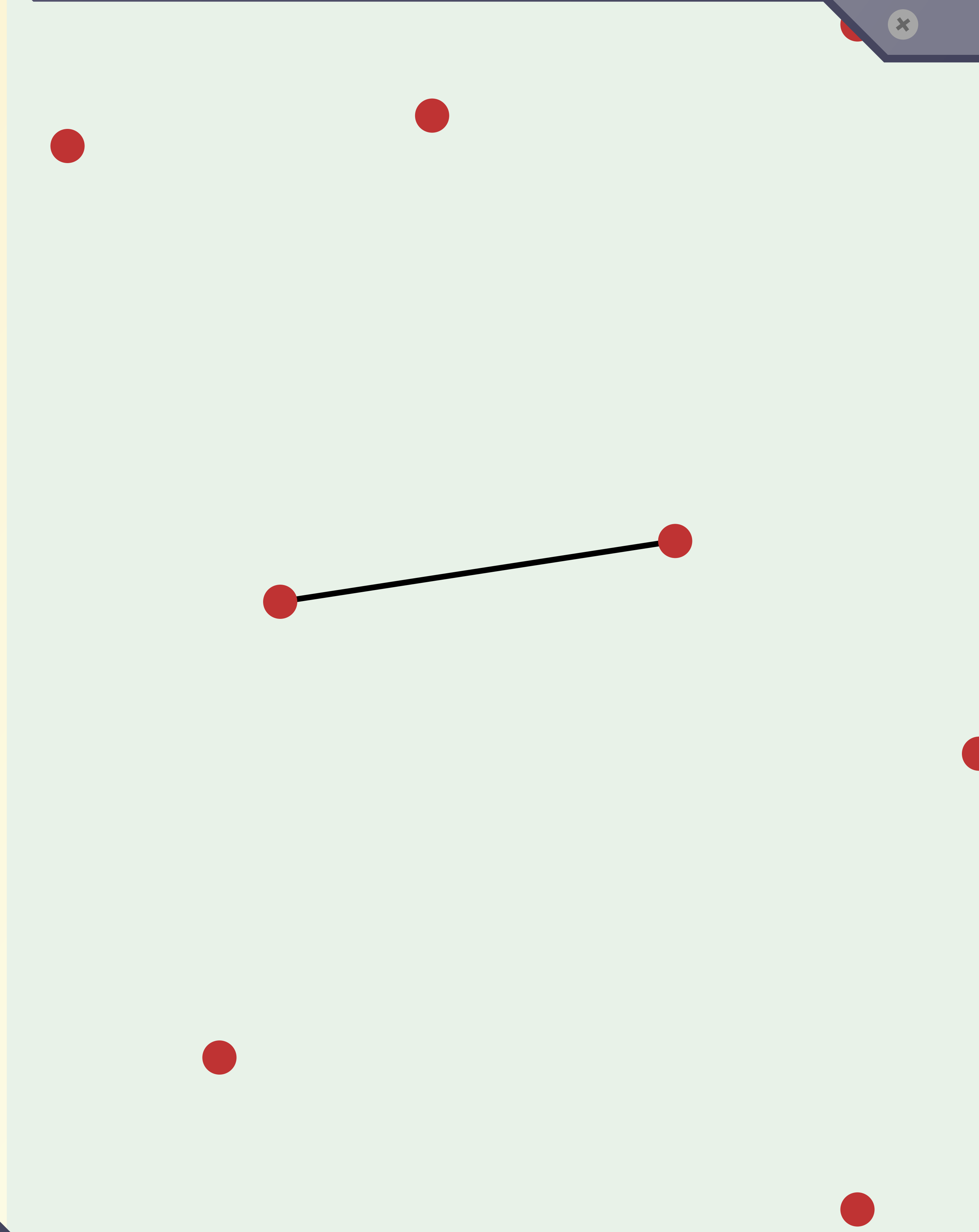
The *Delaunay triangulation (DT)* is a triangulation D of a point set P that has an edge between two points if there is an empty circle through these points.



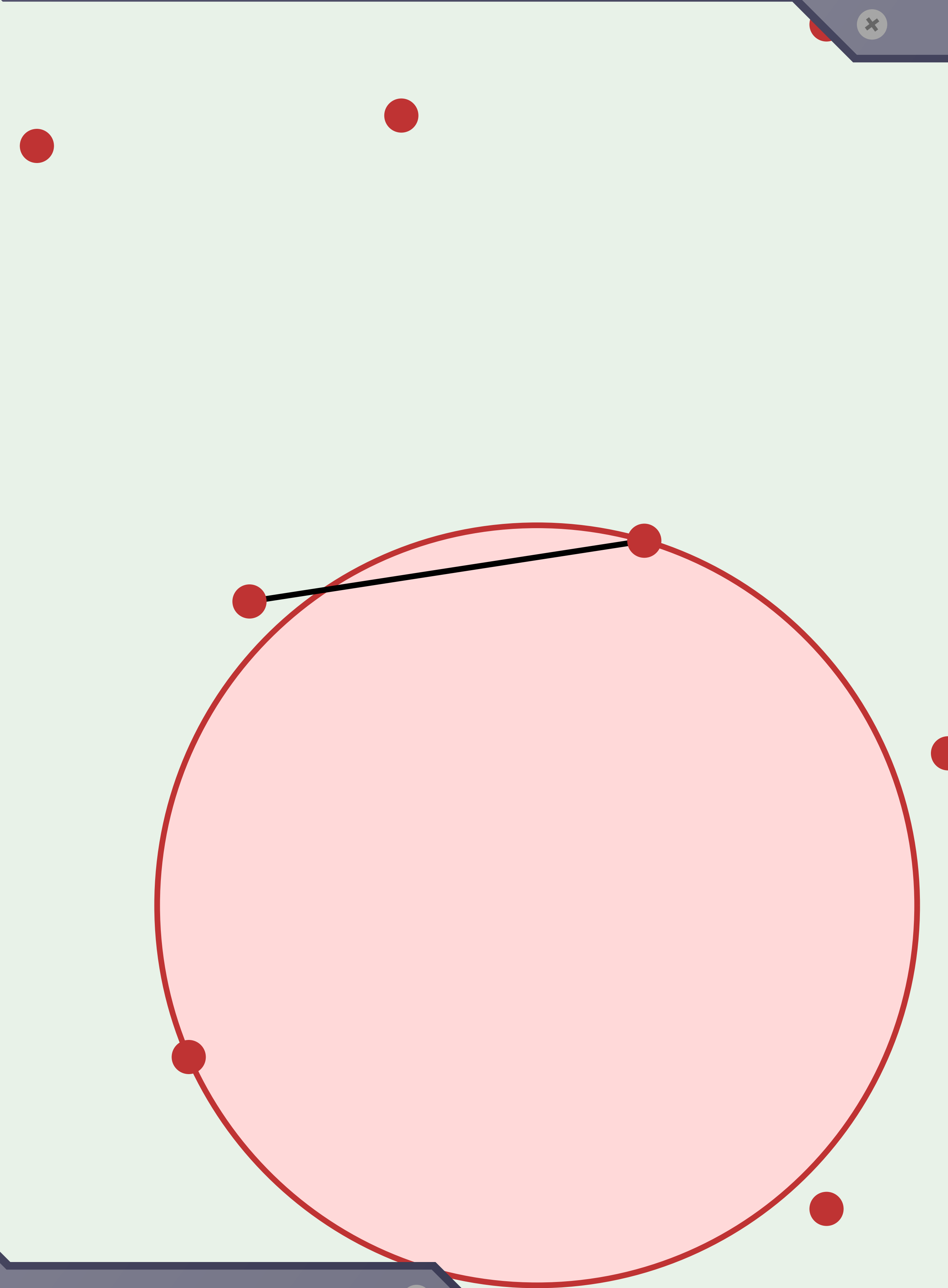
The *Delaunay triangulation (DT)* is a triangulation D of a point set P that has an edge between two points if there is an empty circle through these points.



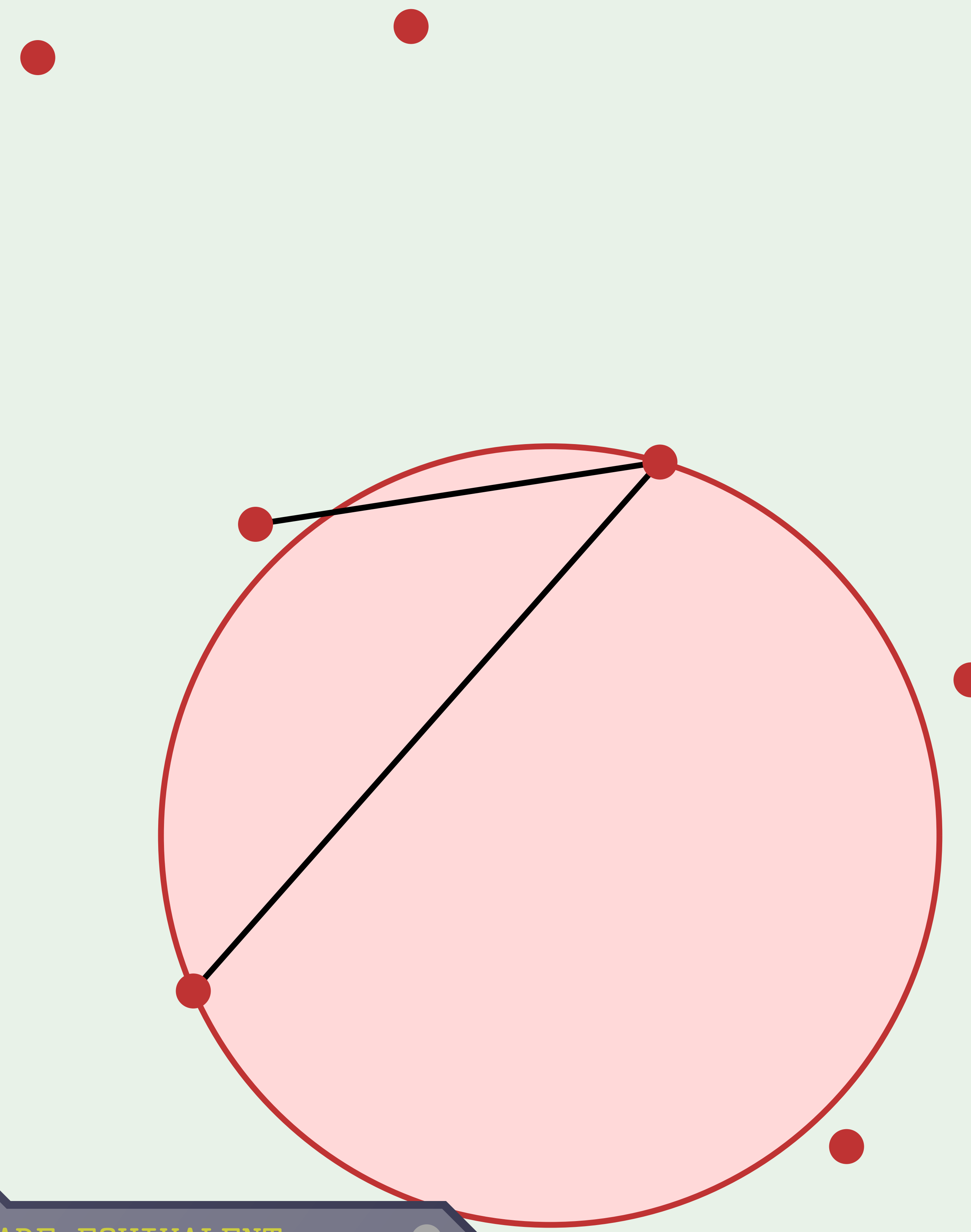
The *Delaunay triangulation (DT)* is a triangulation D of a point set P that has an edge between two points if there is an empty circle through these points.



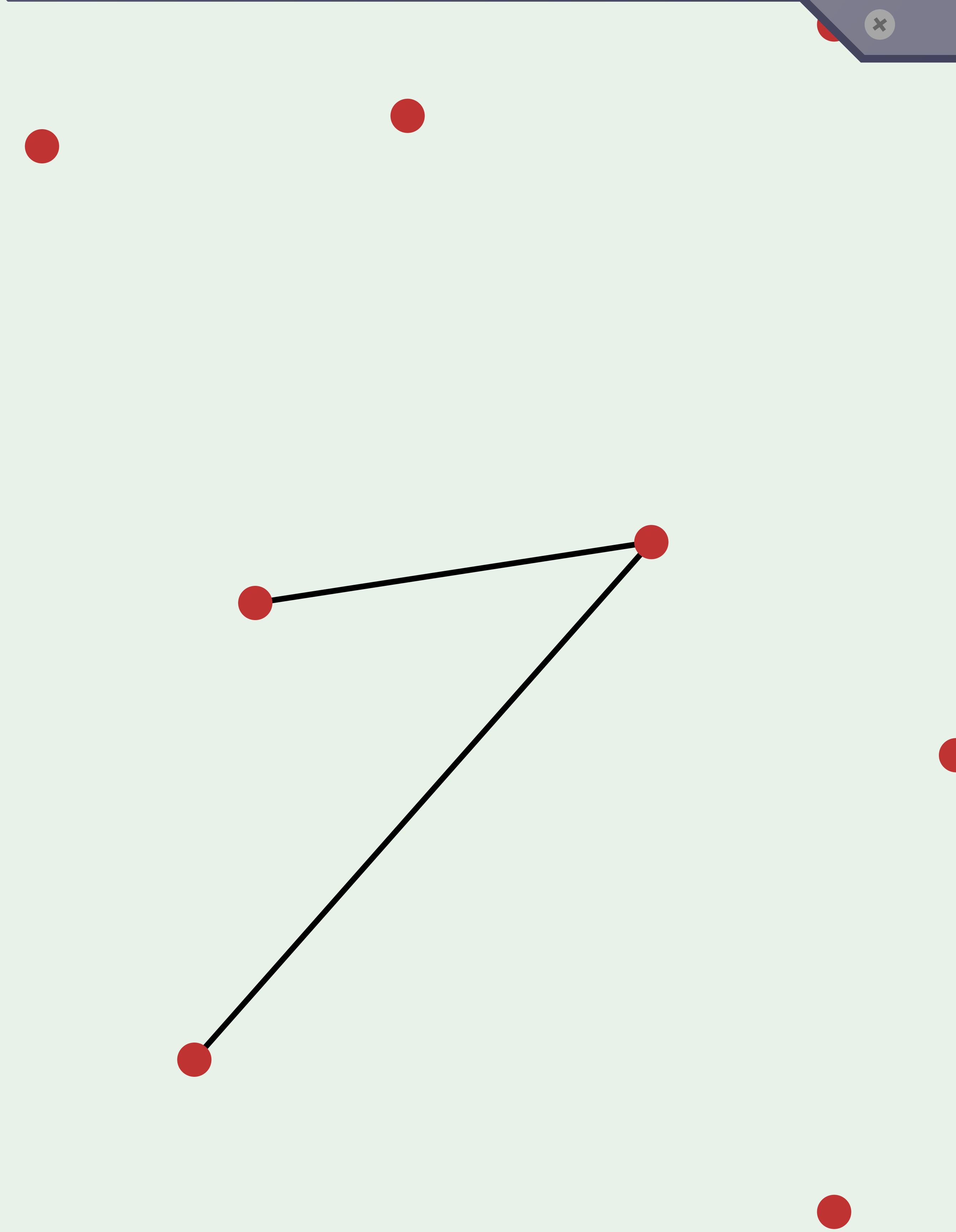
The *Delaunay triangulation (DT)* is a triangulation D of a point set P that has an edge between two points if there is an empty circle through these points.



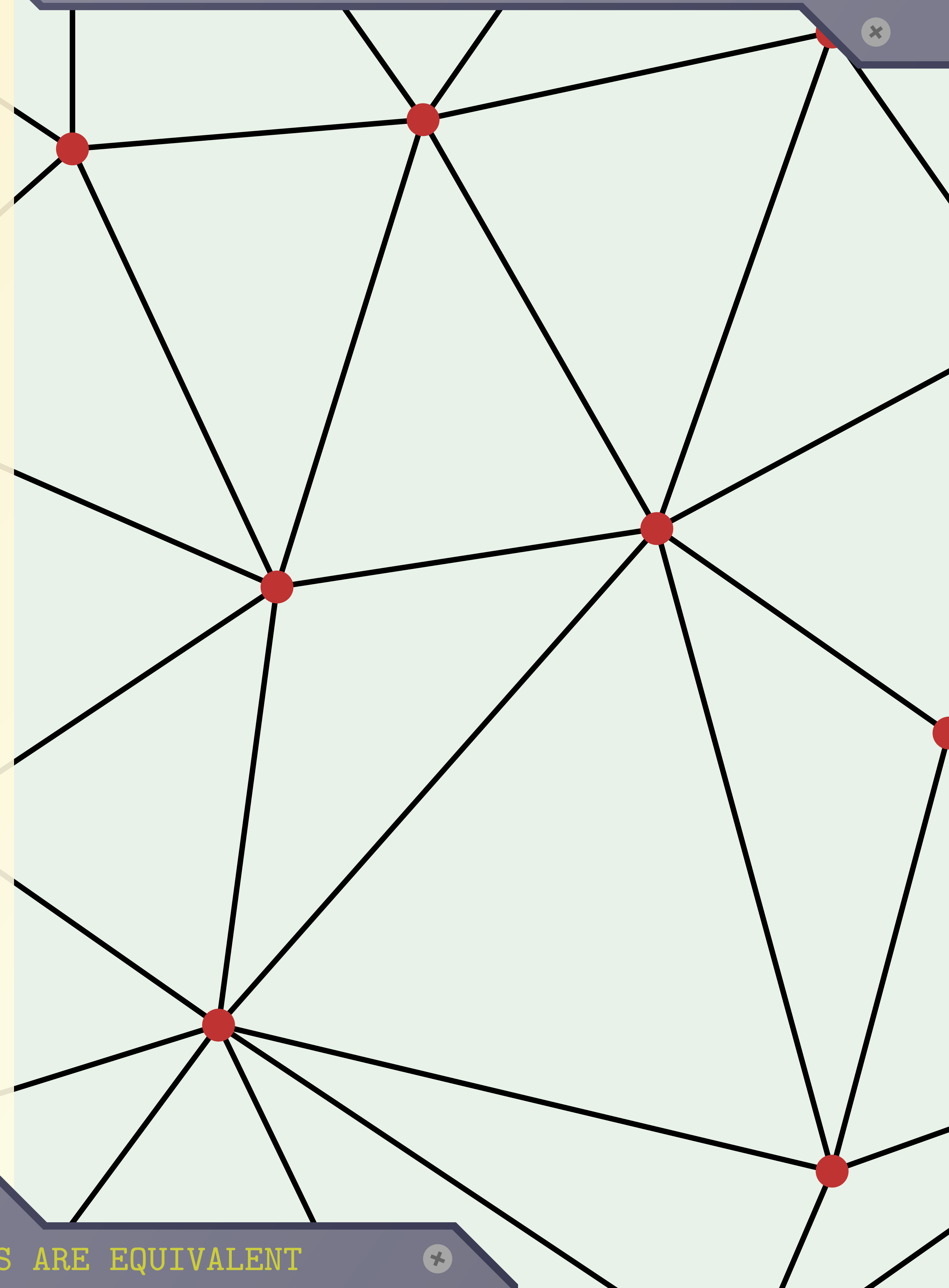
The *Delaunay triangulation (DT)* is a triangulation D of a point set P that has an edge between two points if there is an empty circle through these points.



The *Delaunay triangulation (DT)* is a triangulation D of a point set P that has an edge between two points if there is an empty circle through these points.



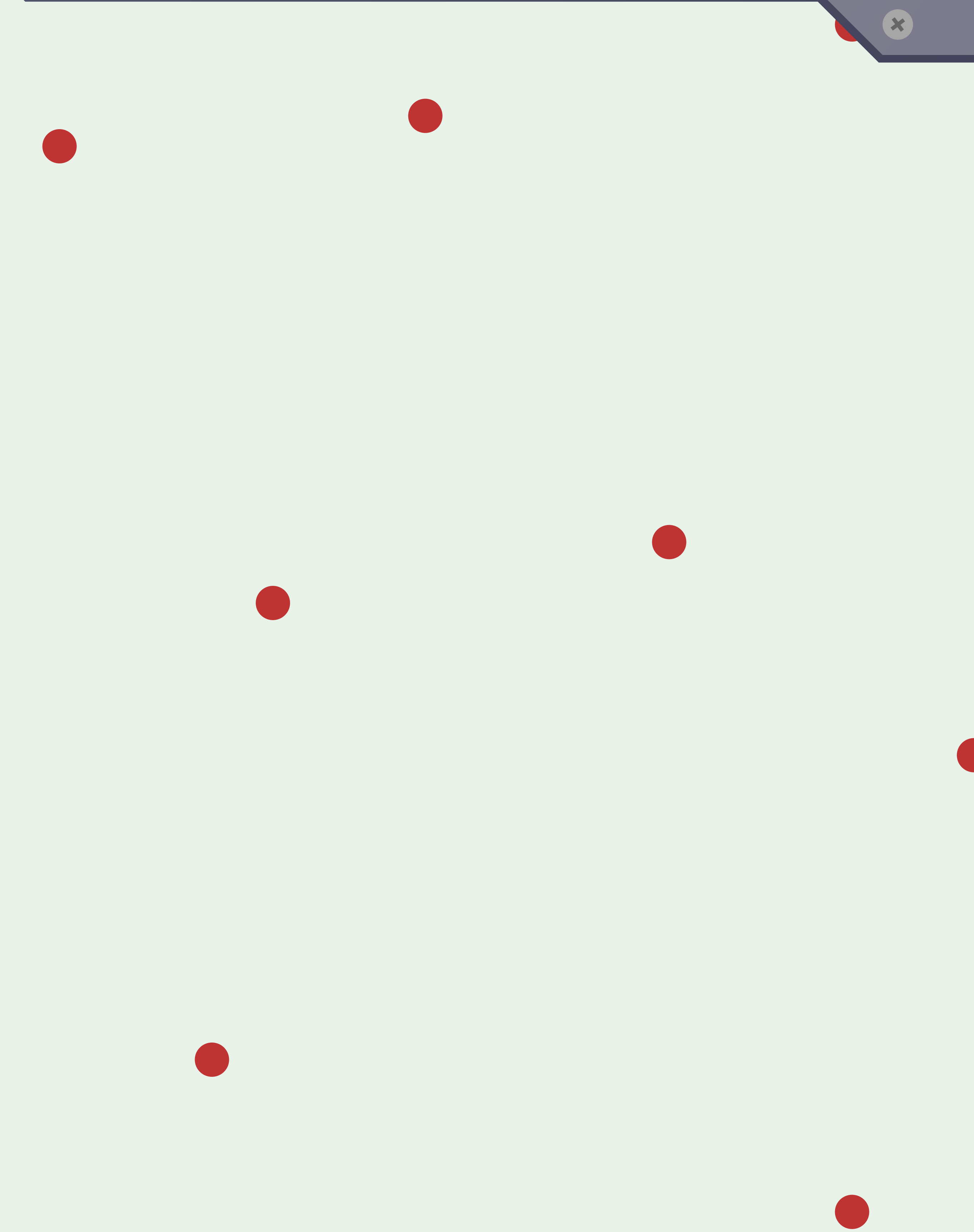
The *Delaunay triangulation (DT)* is a triangulation D of a point set P that has an edge between two points if there is an empty circle through these points.



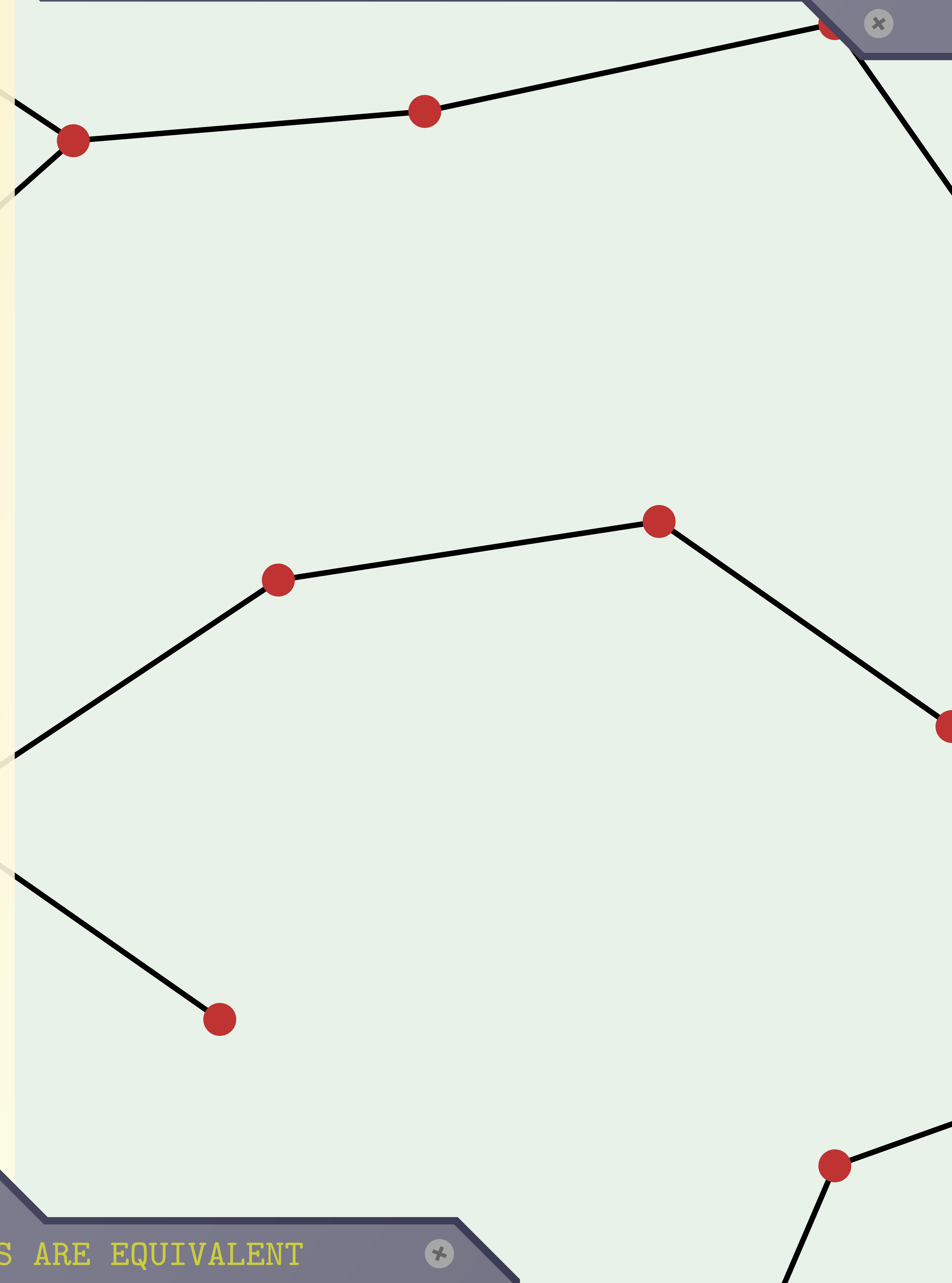
MINIMUM SPANNING TREE

The *Euclidean minimum spanning tree (EMST)* is the minimum spanning tree of the complete graph on a point set P.

The *Euclidean minimum spanning tree (EMST)* is the minimum spanning tree of the complete graph on a point set P.

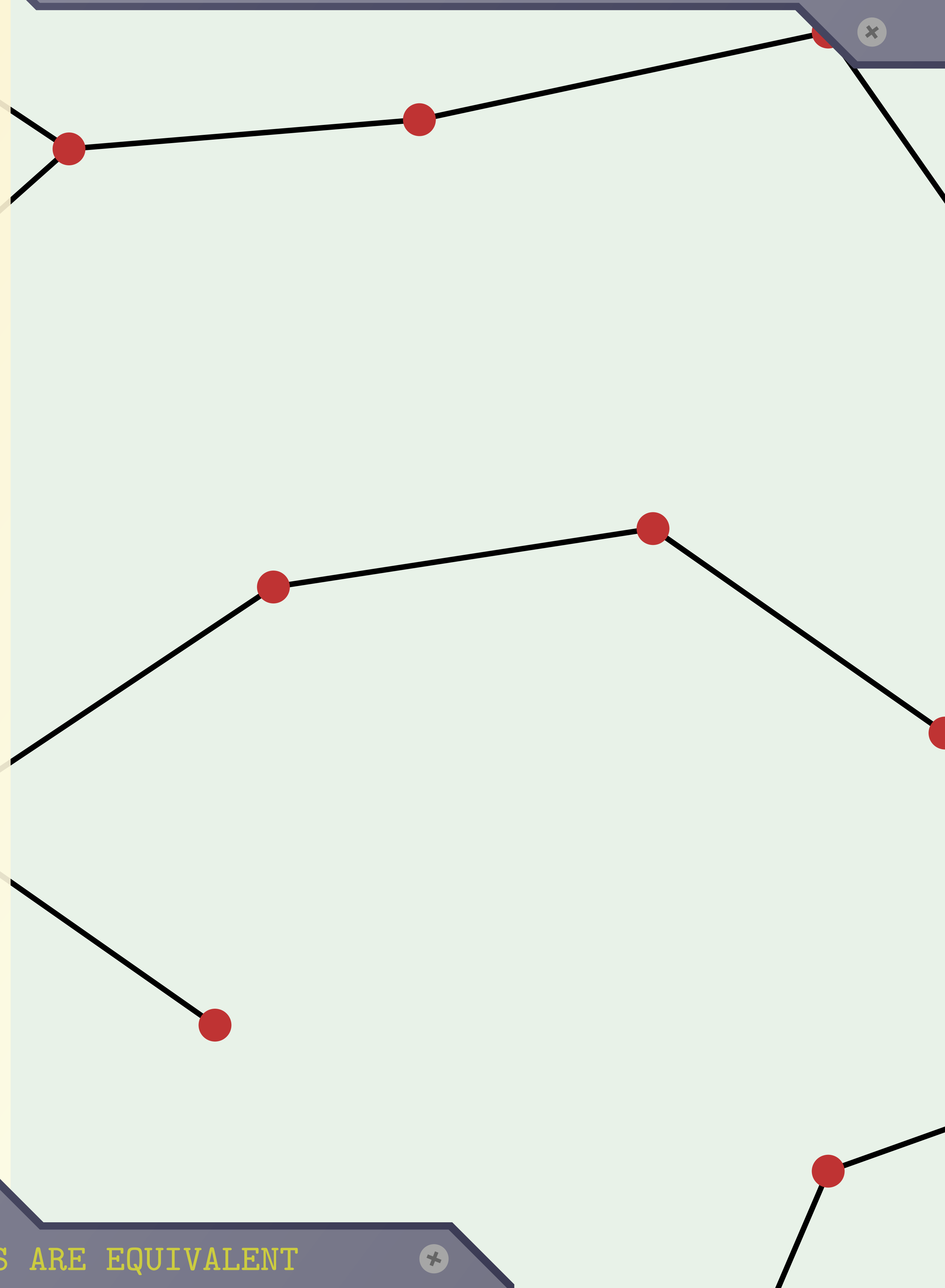


The *Euclidean minimum spanning tree (EMST)* is the minimum spanning tree of the complete graph on a point set P.



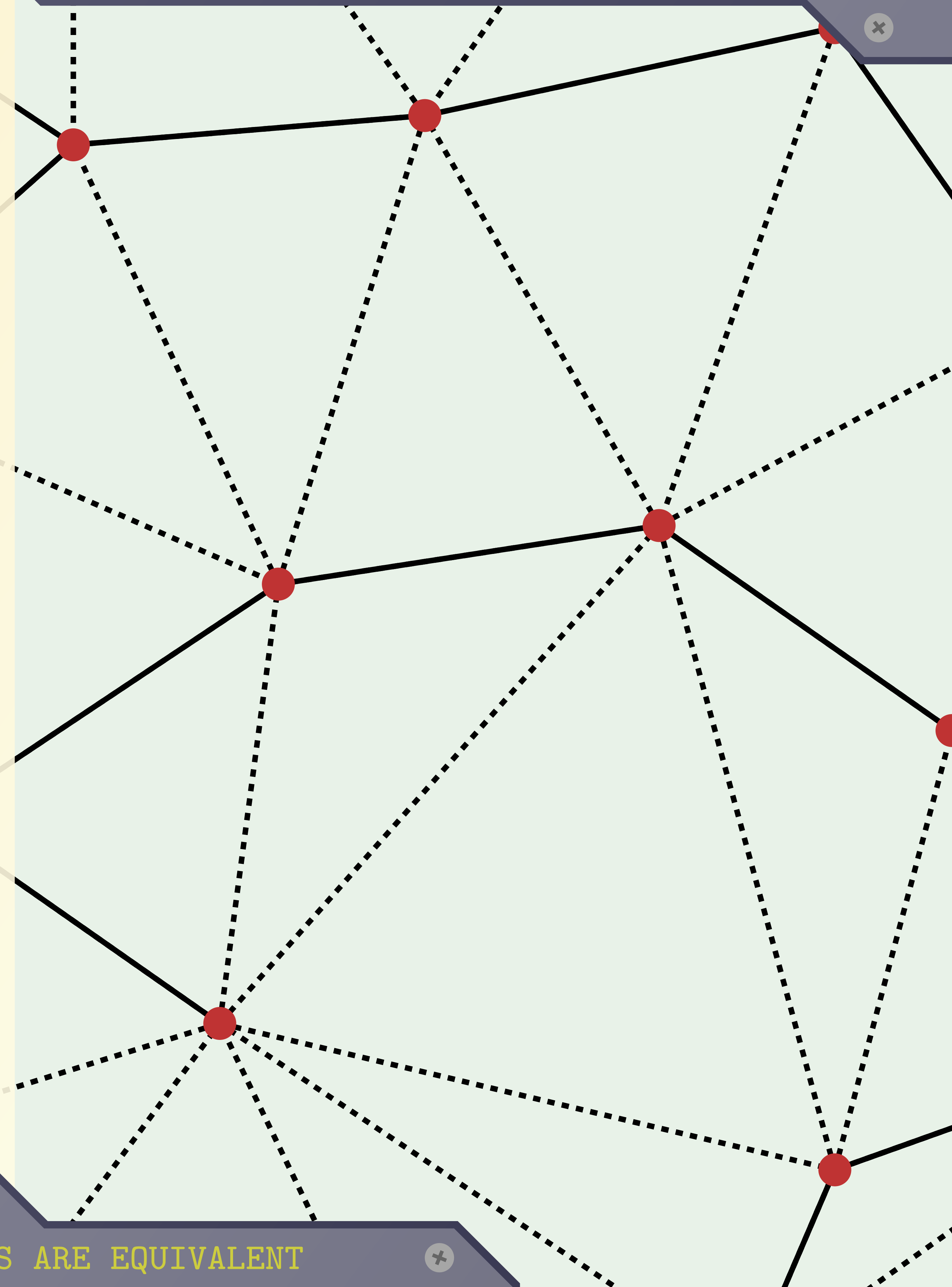
The *Euclidean minimum spanning tree (EMST)* is the minimum spanning tree of the complete graph on a point set P.

The EMST is a subgraph of the DT.



The *Euclidean minimum spanning tree (EMST)* is the minimum spanning tree of the complete graph on a point set P.

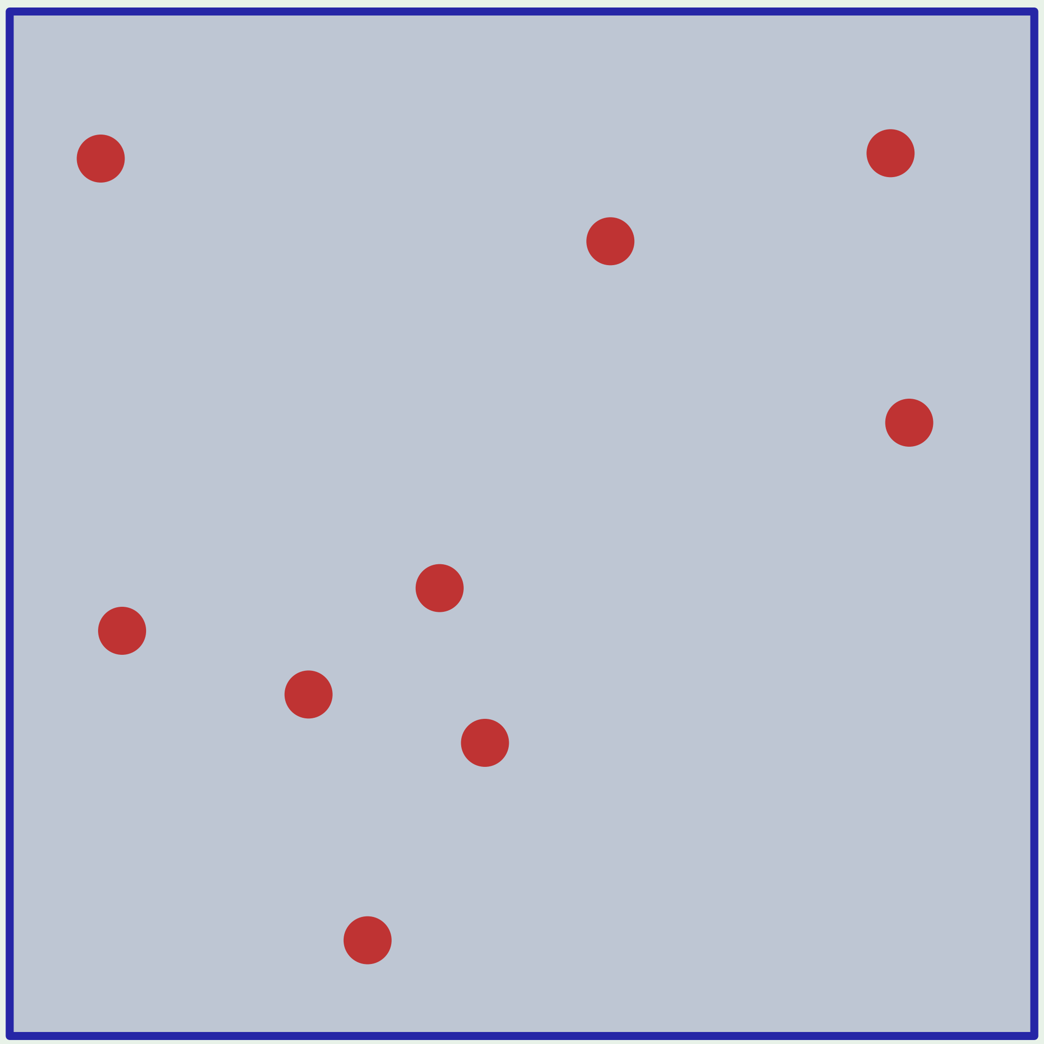
The EMST is a subgraph of the DT.



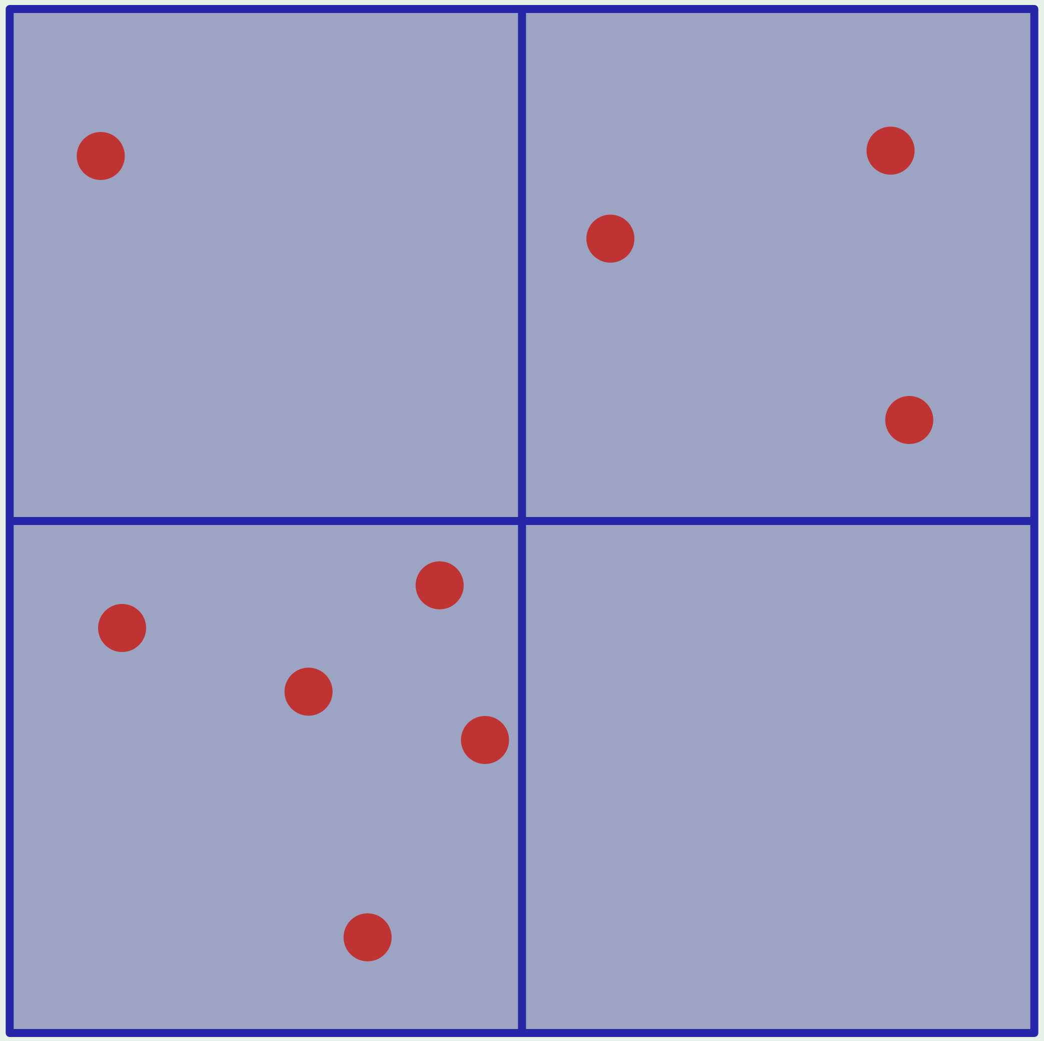
COMPRESSED QUADTREE

A *quadtree* T for a set of points P is a hierarchical subdivision of a square into smaller squares that separates P.

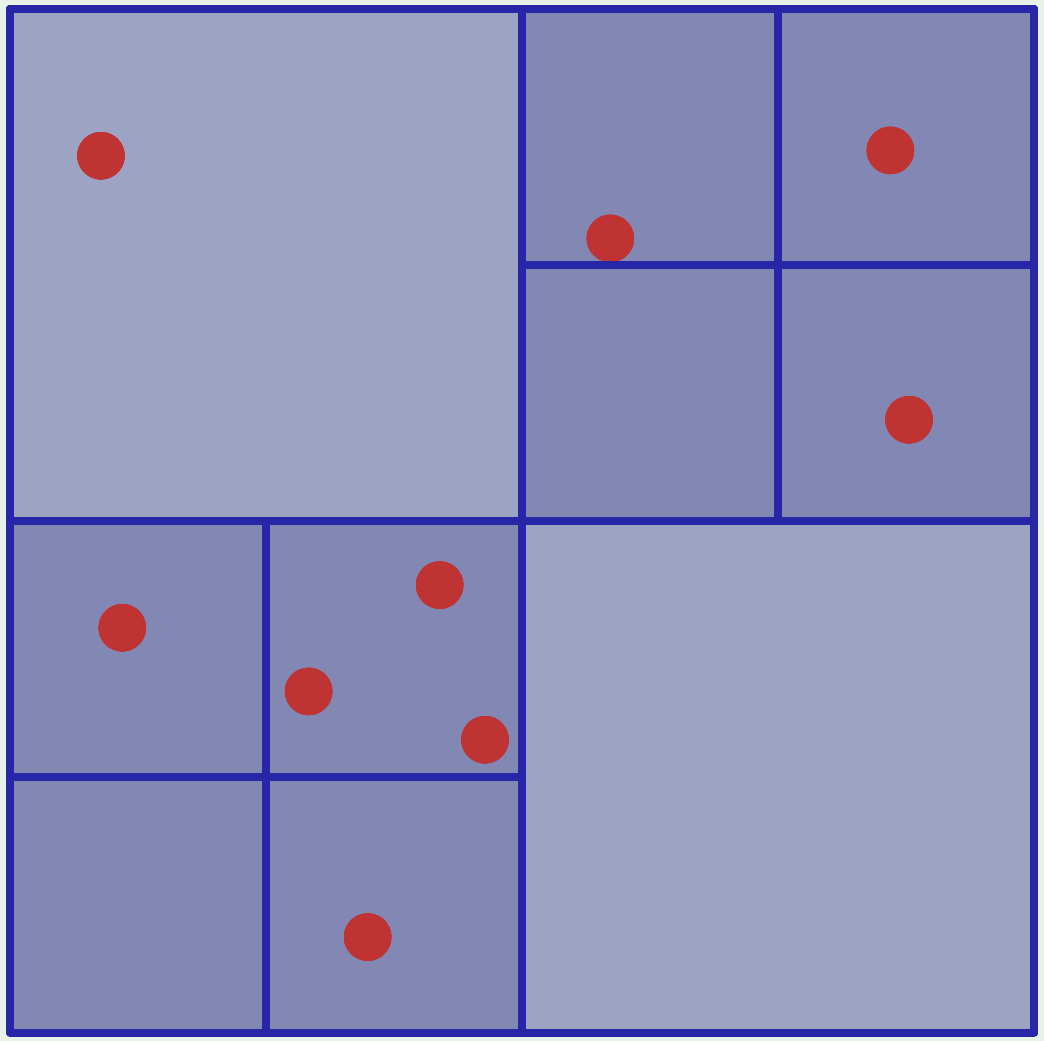
A *quadtree* T for a set of points P is a hierarchical subdivision of a square into smaller squares that separates P.



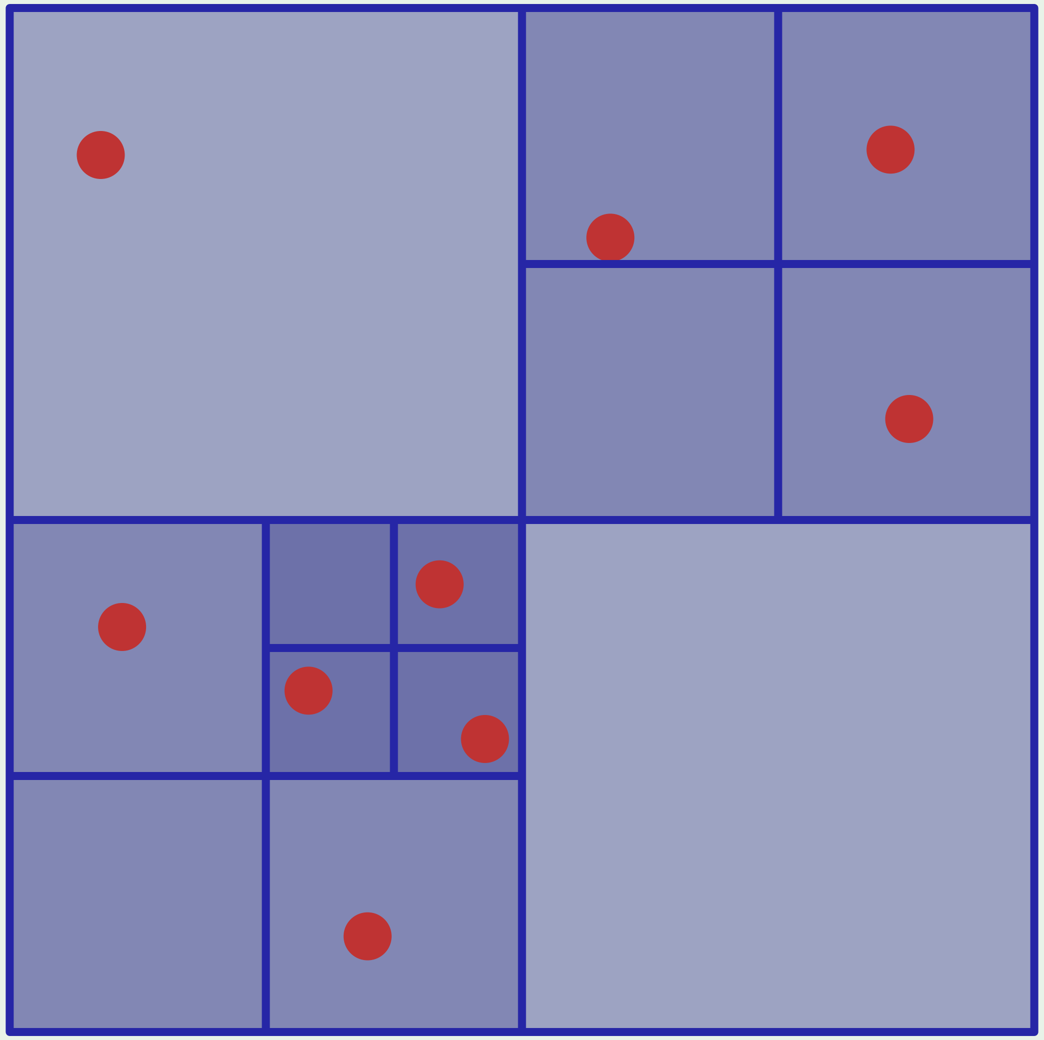
A *quadtree* T for a set of points P is a hierarchical subdivision of a square into smaller squares that separates P.



A *quadtree* T for a set of points P is a hierarchical subdivision of a square into smaller squares that separates P.

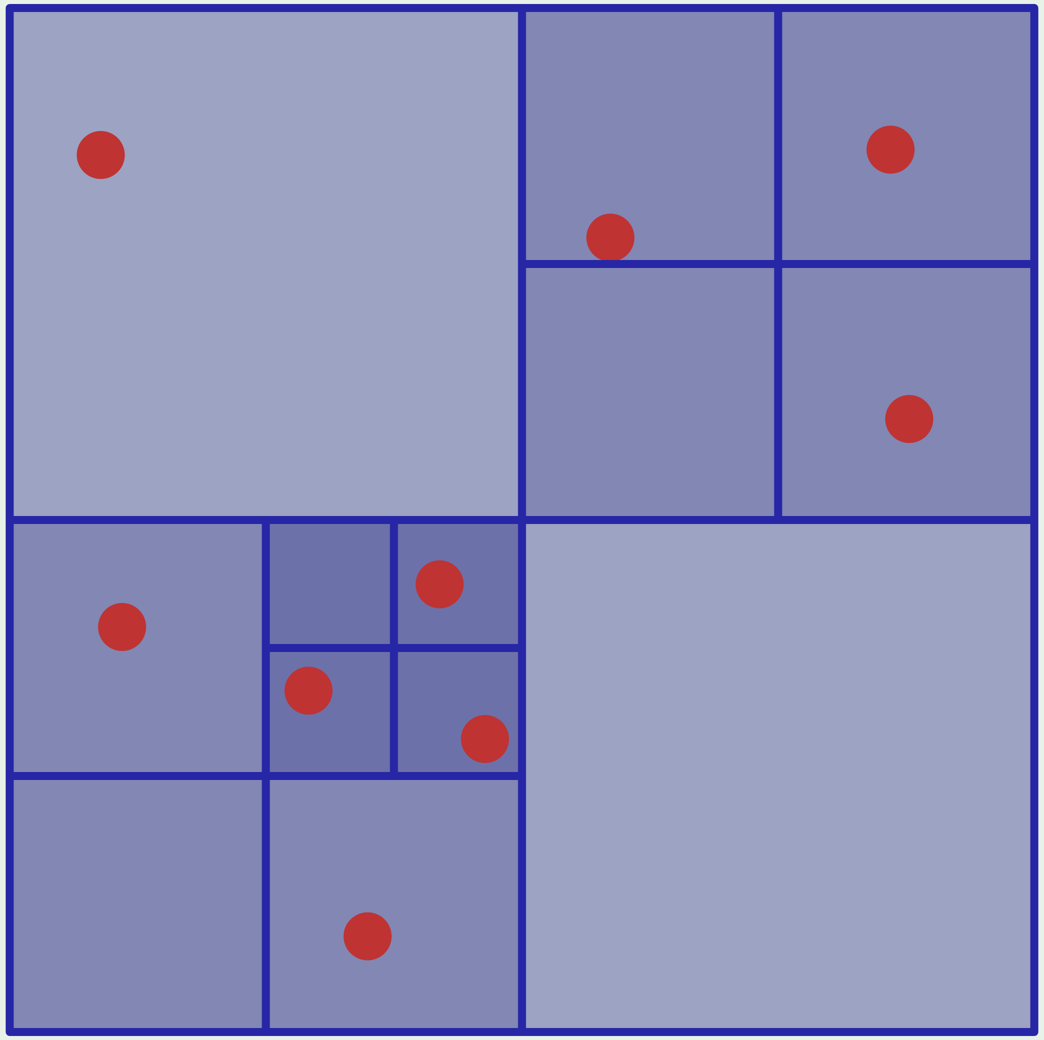


A *quadtree* T for a set of points P is a hierarchical subdivision of a square into smaller squares that separates P.



A *quadtree* T for a set of points P is a hierarchical subdivision of a square into smaller squares that separates P.

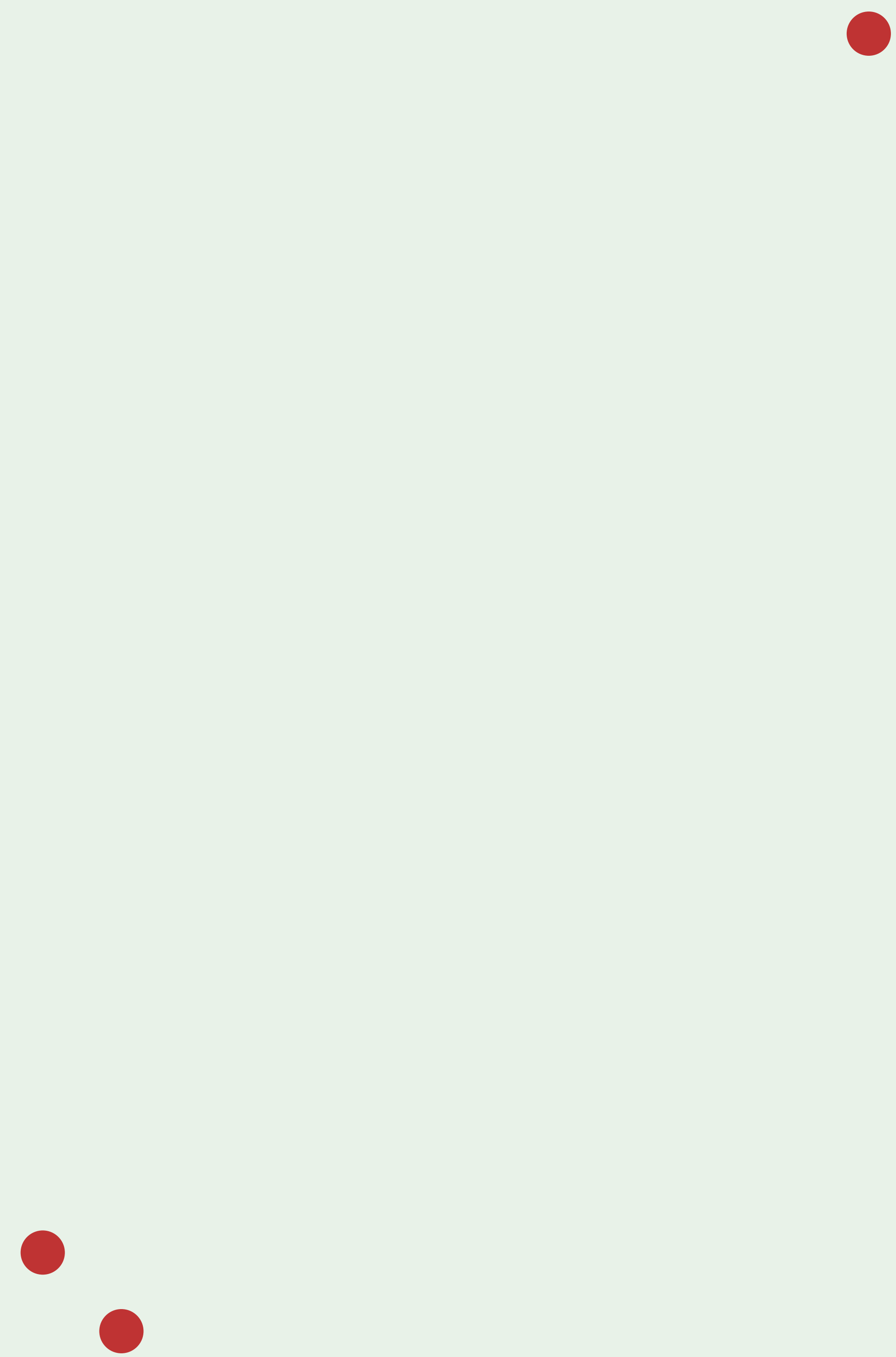
For point sets with large spread, a quadtree can be *compressed*.



COMPRESSED QUADTREE

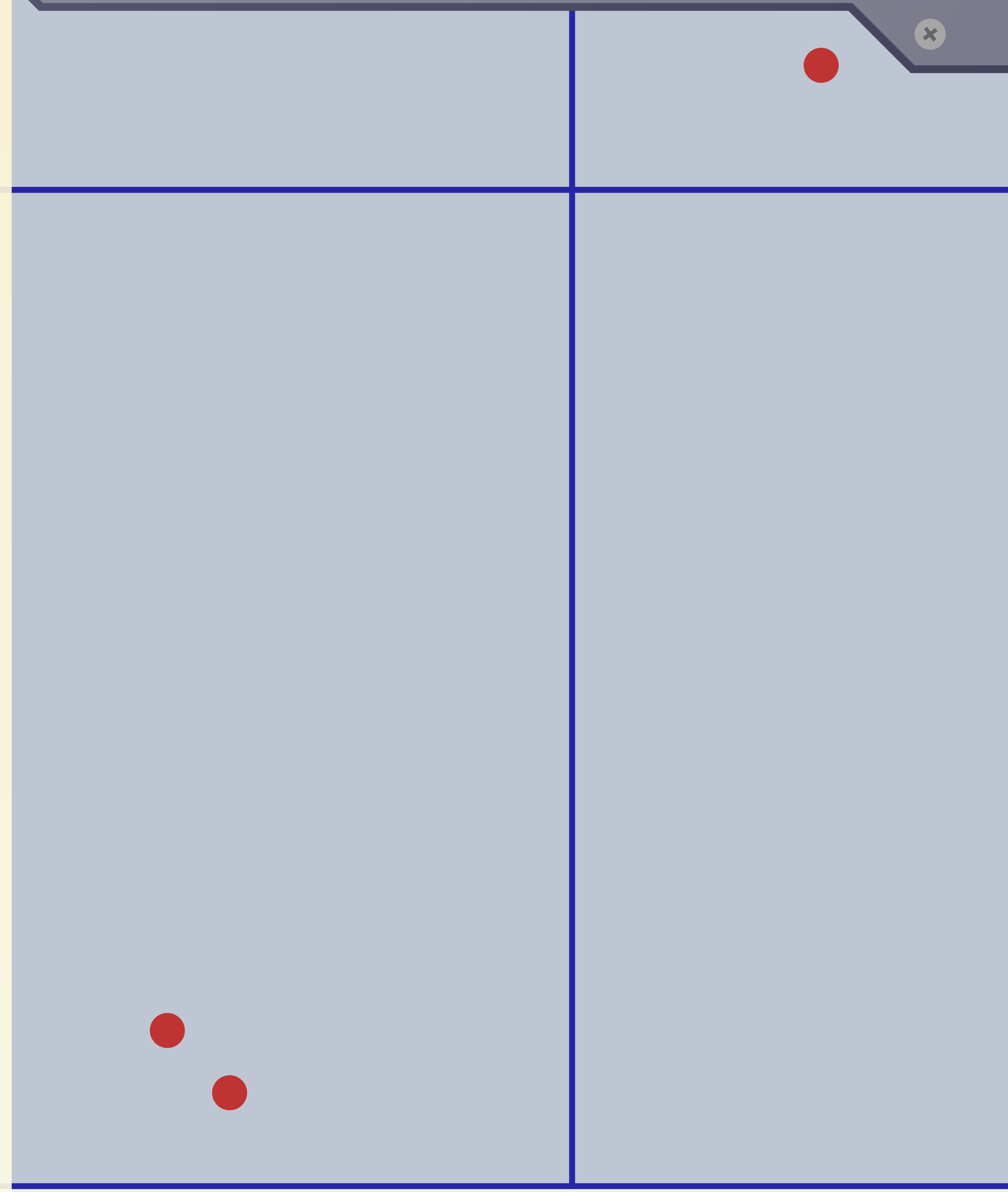
A *quadtree* T for a set of points P is a hierarchical subdivision of a square into smaller squares that separates P.

For point sets with large spread, a quadtree can be *compressed*.



A *quadtree* T for a set of points P is a hierarchical subdivision of a square into smaller squares that separates P.

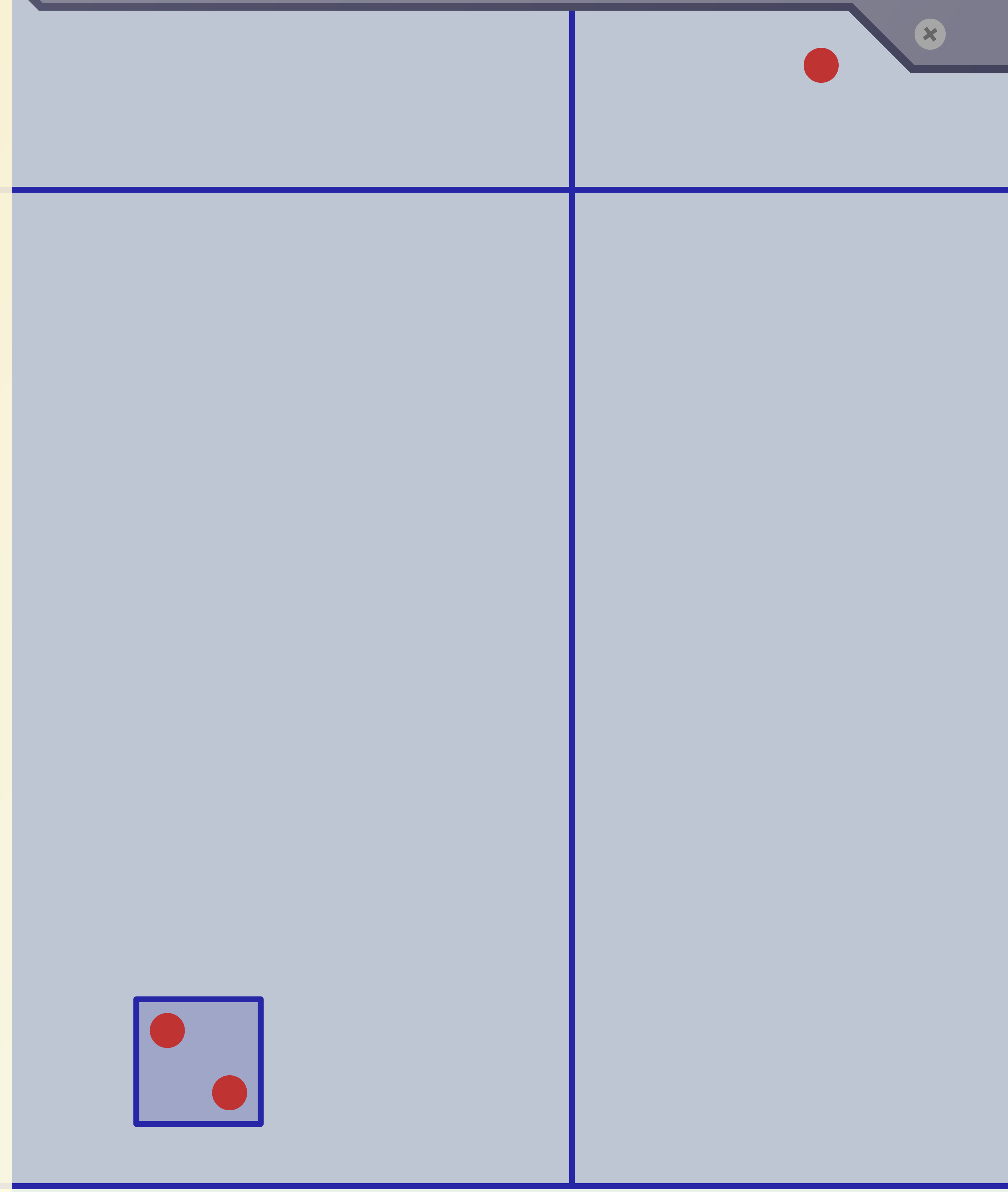
For point sets with large spread, a quadtree can be *compressed*.



COMPRESSED QUADTREE

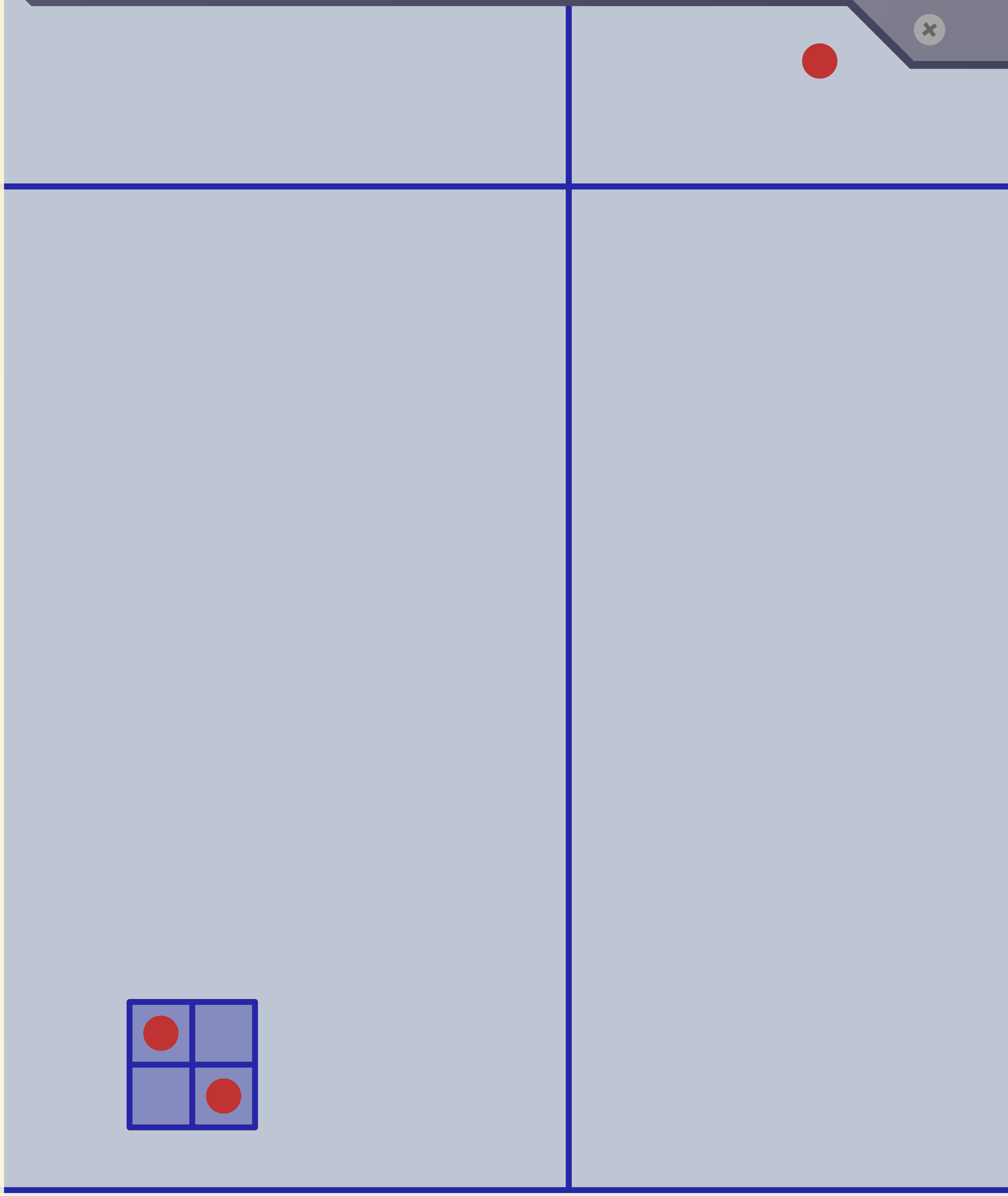
A *quadtree* T for a set of points P is a hierarchical subdivision of a square into smaller squares that separates P.

For point sets with large spread, a quadtree can be *compressed*.



A *quadtree* T for a set of points P is a hierarchical subdivision of a square into smaller squares that separates P.

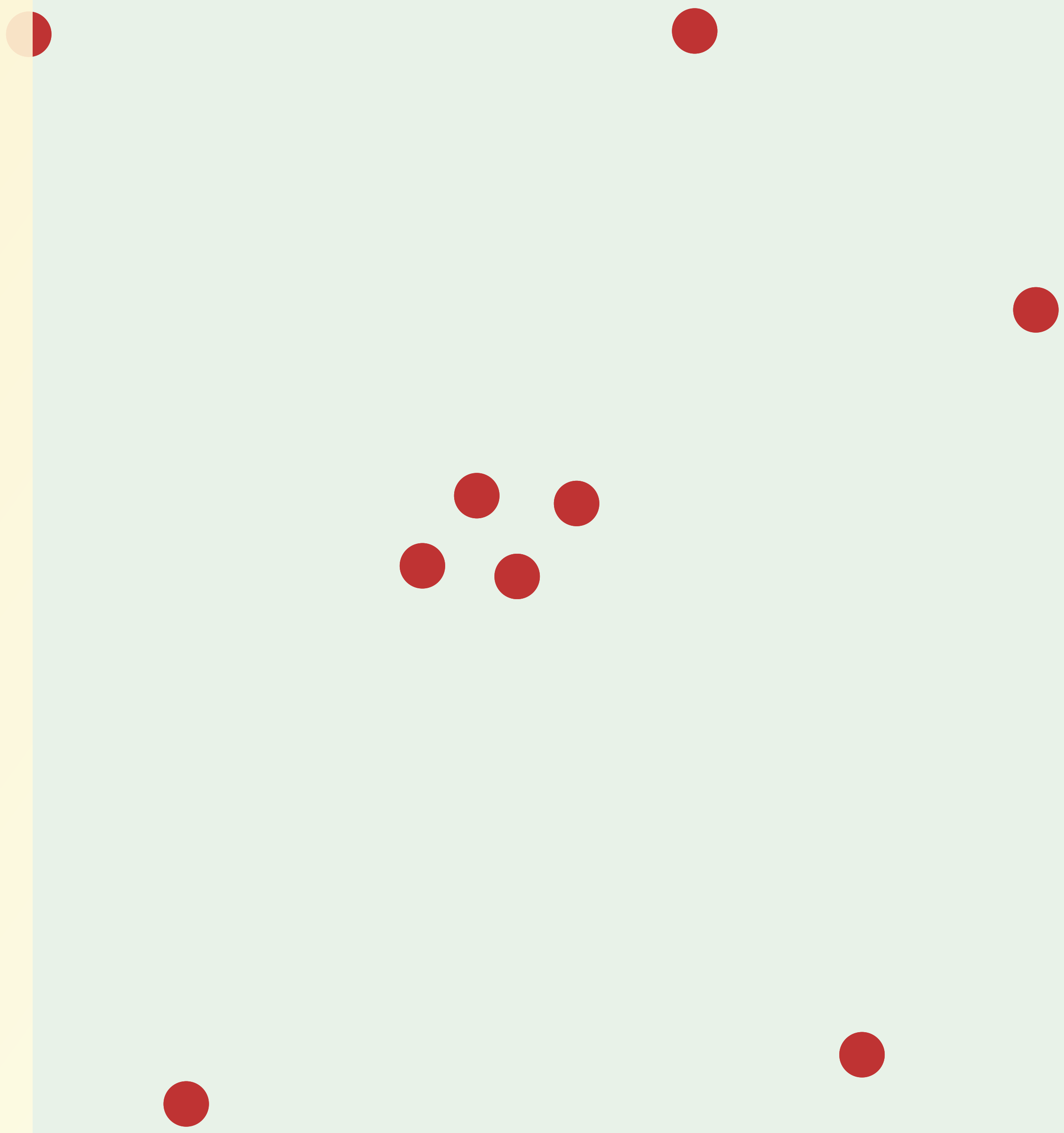
For point sets with large spread, a quadtree can be *compressed*.



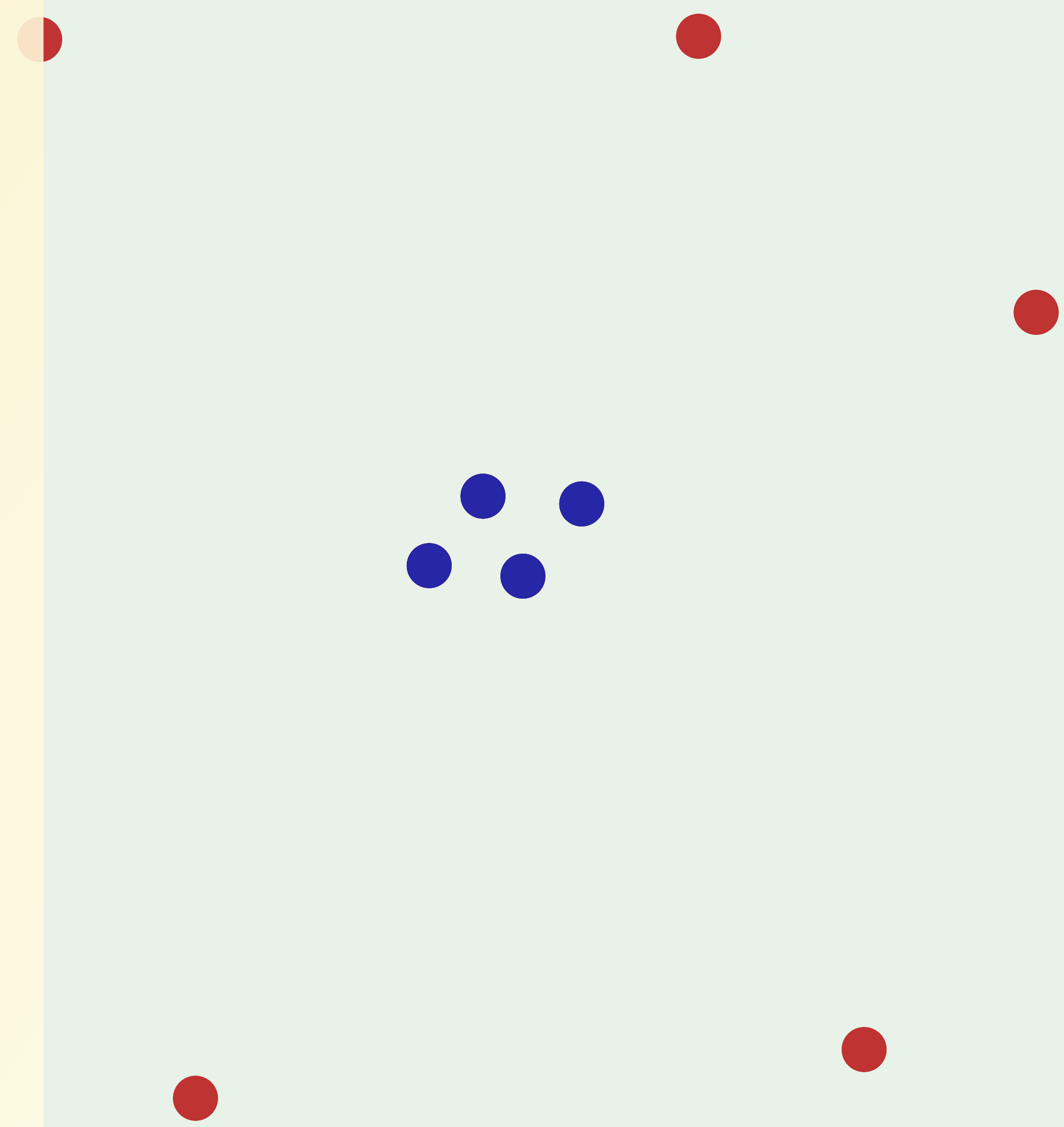
c-CLUSTER QUADTREE

A *c-cluster* in a point set P is a subset $C \subset P$ whose distance to the rest of P is at least c its diameter.

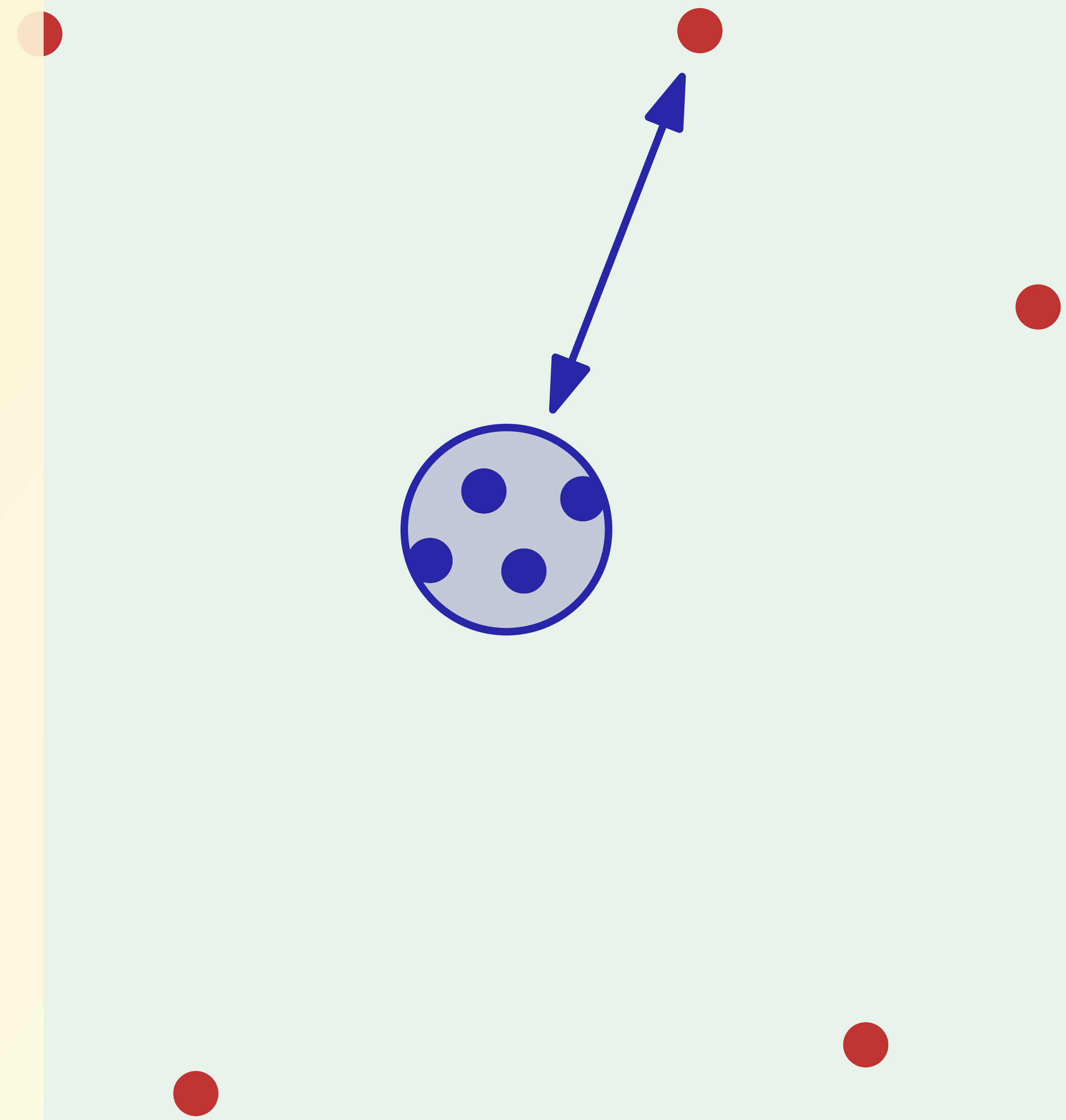
A *c-cluster* in a point set P is a subset $C \subset P$ whose distance to the rest of P is at least c its diameter.



A *c-cluster* in a point set P is a subset $C \subset P$ whose distance to the rest of P is at least c its diameter.

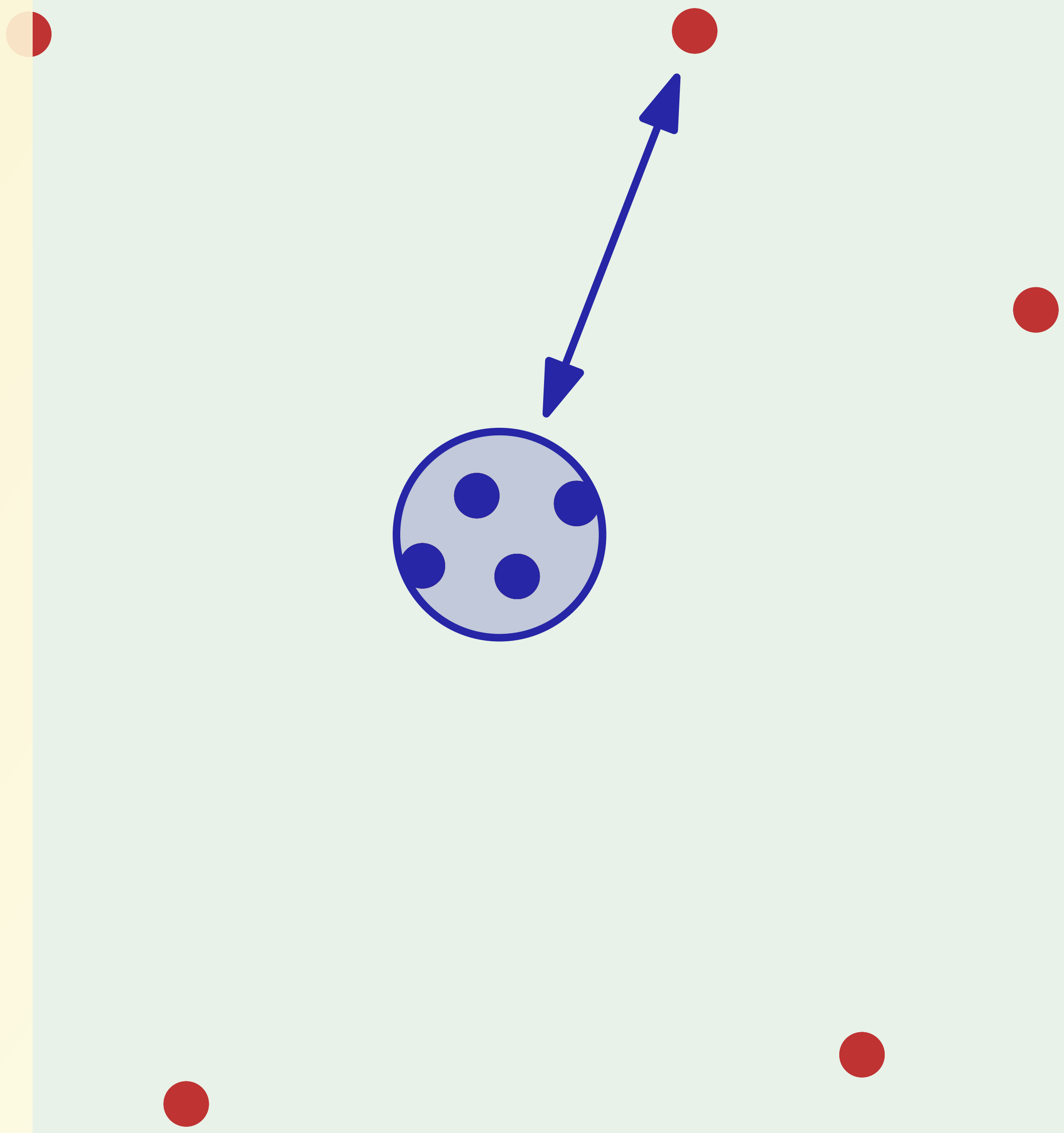


A *c-cluster* in a point set P is a subset $C \subset P$ whose distance to the rest of P is at least c its diameter.



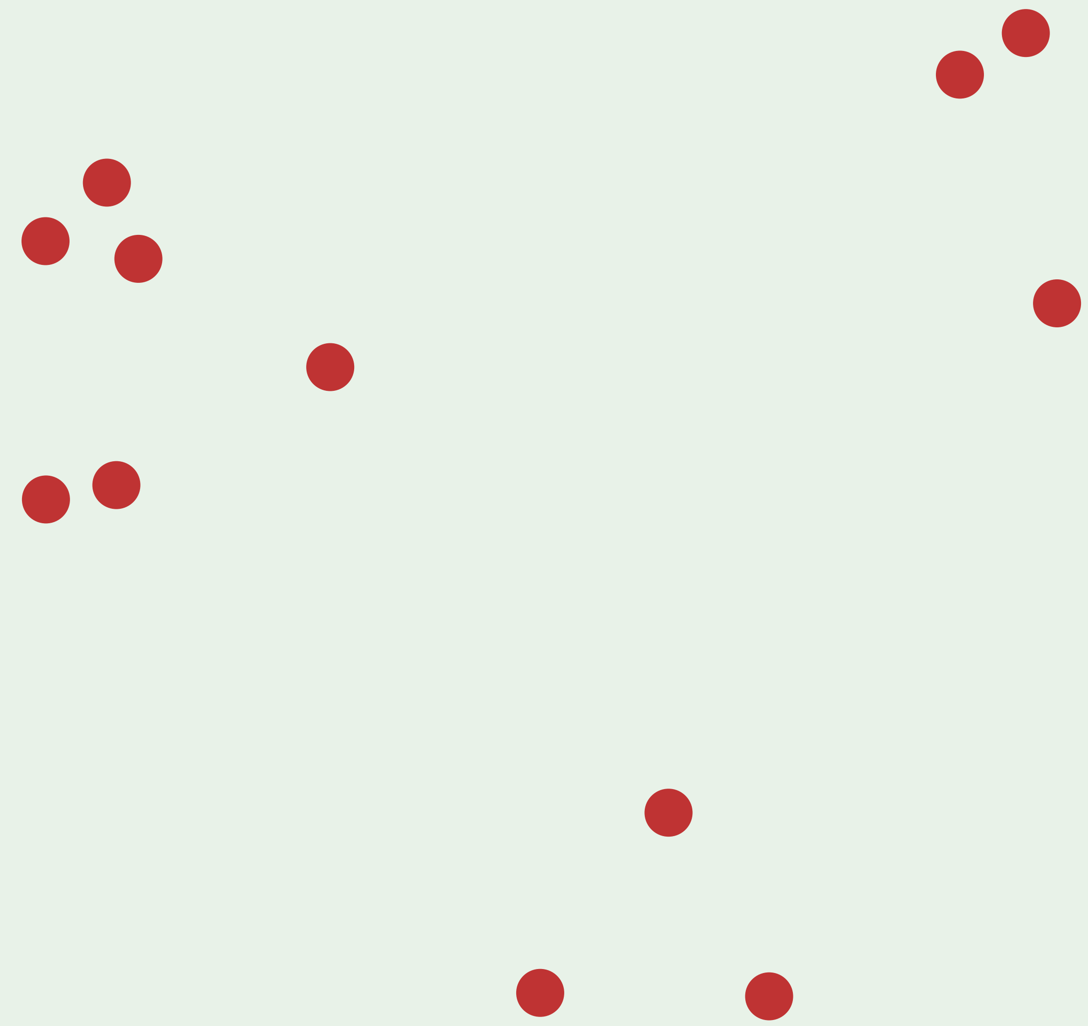
A *c-cluster* in a point set P is a subset $C \subset P$ whose distance to the rest of P is at least c its diameter.

The c-clusters on P form a hierarchy.



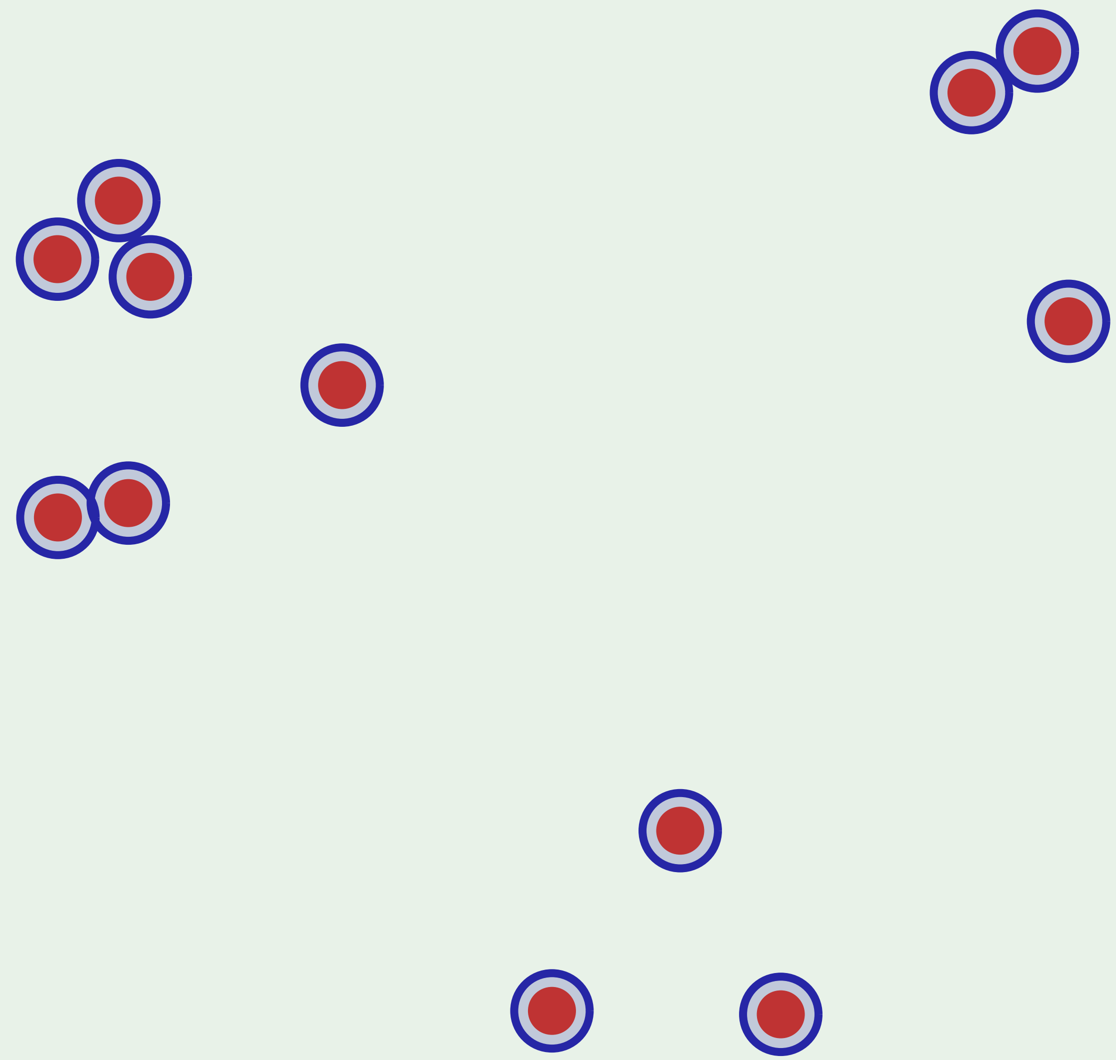
A *c-cluster* in a point set P is a subset $C \subset P$ whose distance to the rest of P is at least c its diameter.

The c-clusters on P form a hierarchy.



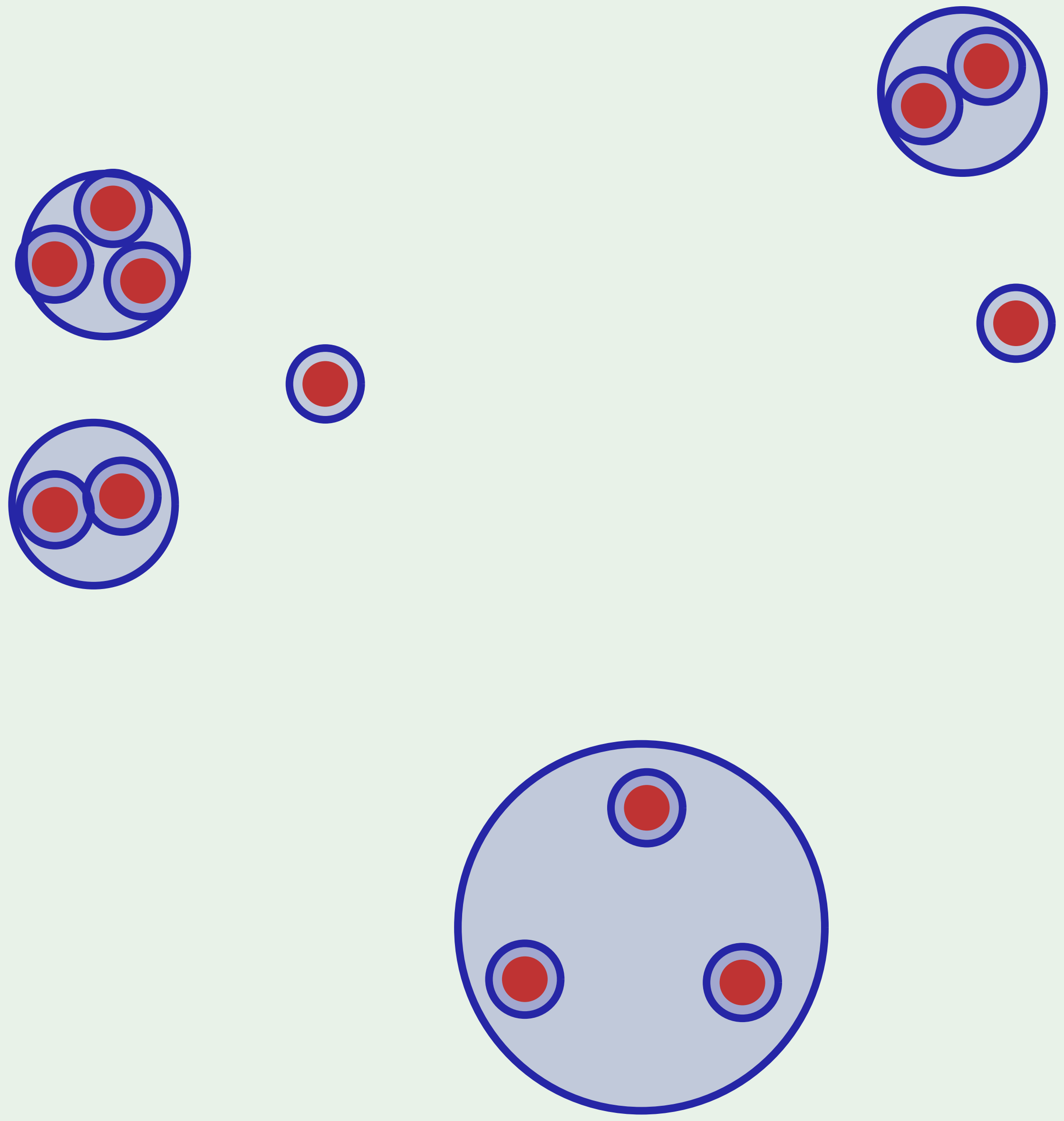
A *c-cluster* in a point set P is a subset $C \subset P$ whose distance to the rest of P is at least c its diameter.

The c-clusters on P form a hierarchy.



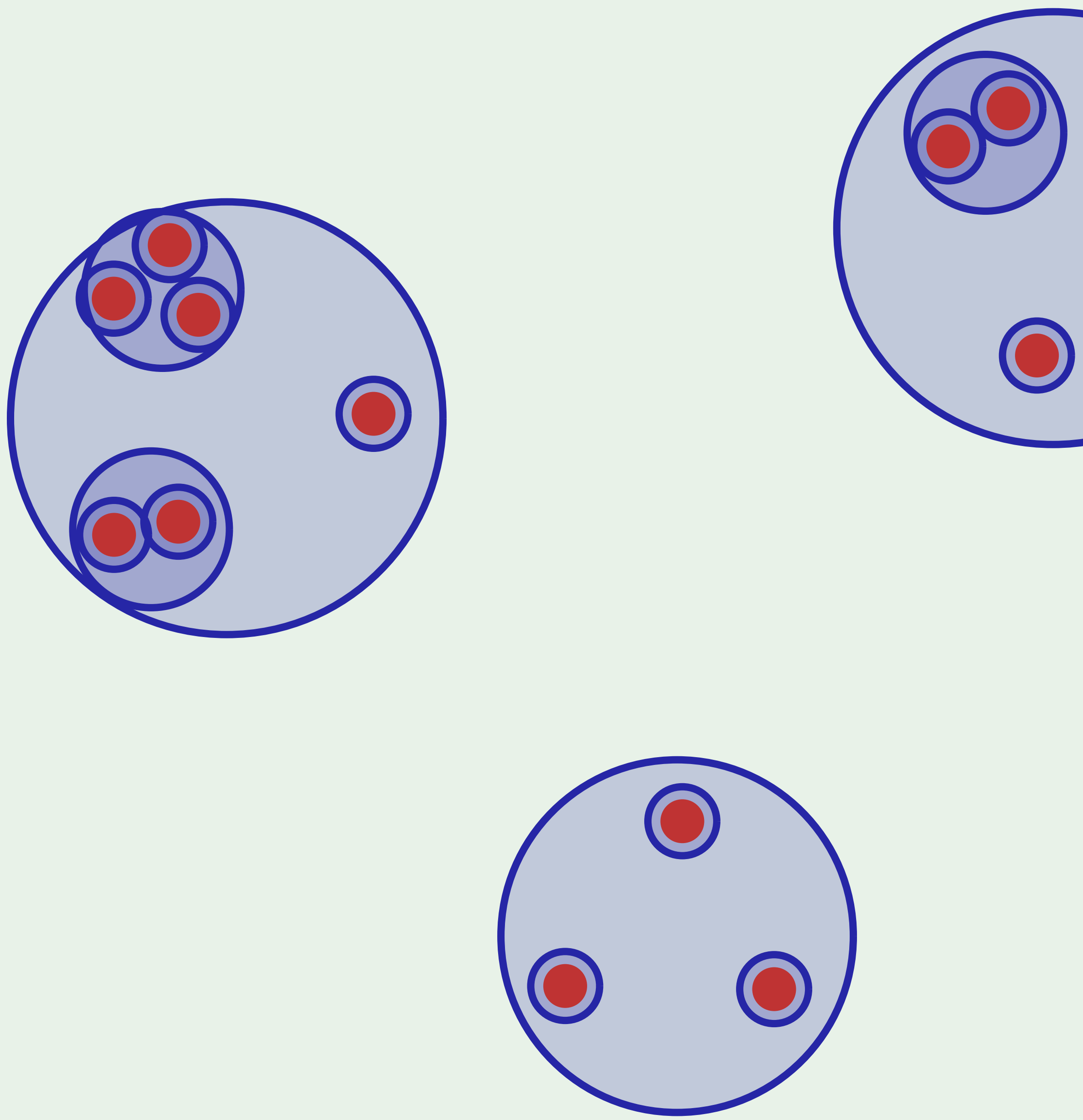
A *c-cluster* in a point set P is a subset $C \subset P$ whose distance to the rest of P is at least c its diameter.

The c-clusters on P form a hierarchy.



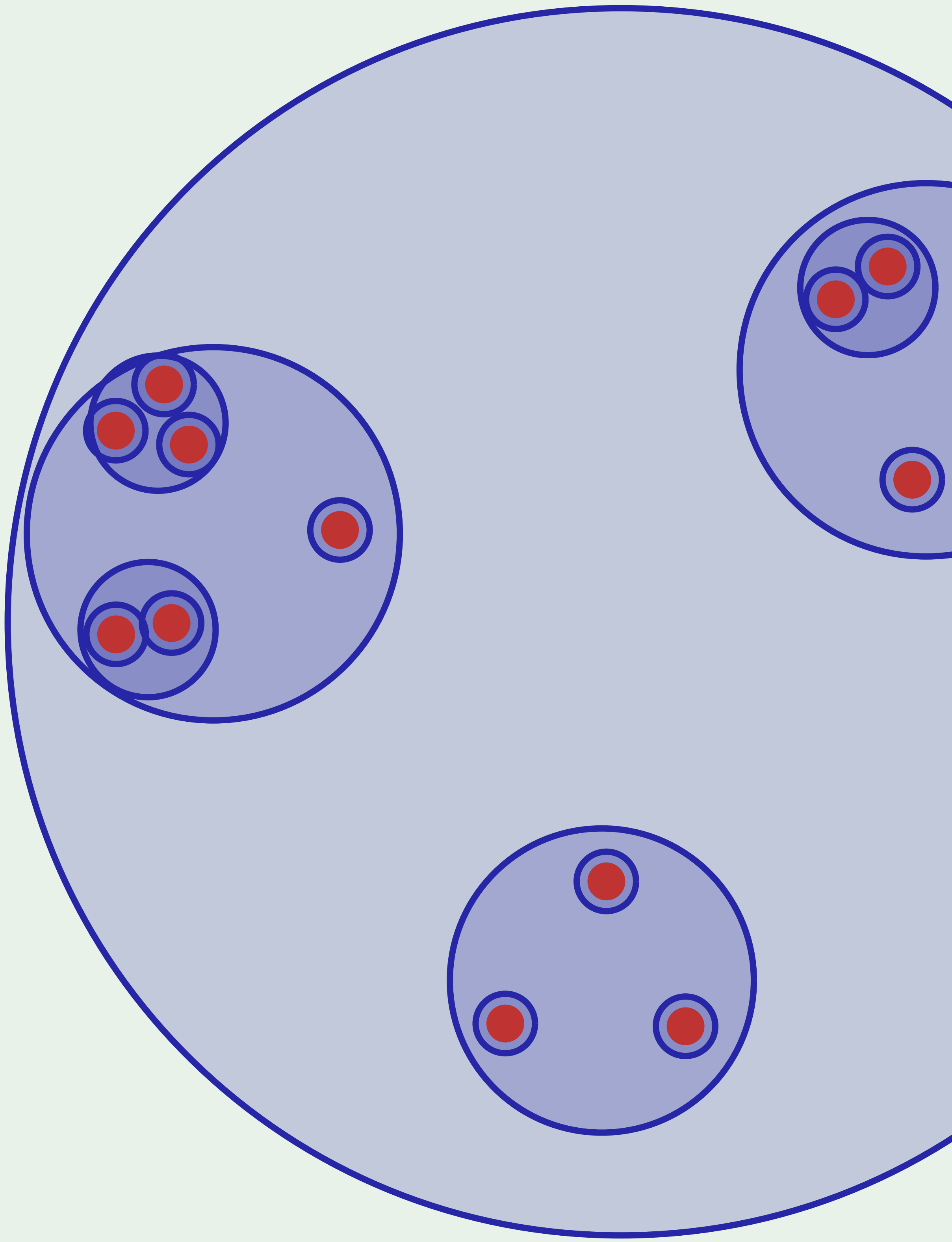
A *c-cluster* in a point set P is a subset $C \subset P$ whose distance to the rest of P is at least c its diameter.

The c-clusters on P form a hierarchy.



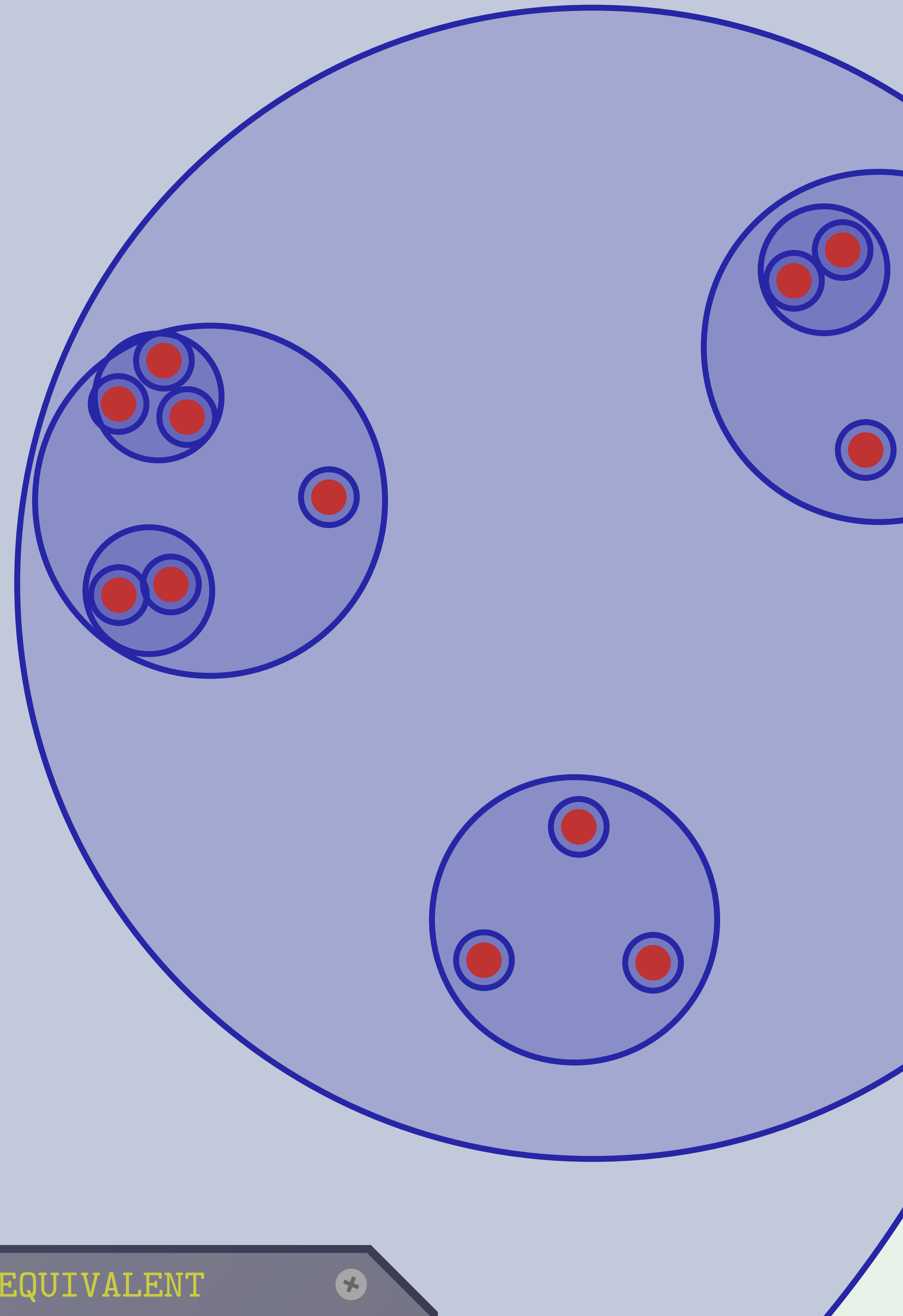
A *c-cluster* in a point set P is a subset $C \subset P$ whose distance to the rest of P is at least c its diameter.

The c-clusters on P form a hierarchy.



A *c-cluster* in a point set P is a subset $C \subset P$ whose distance to the rest of P is at least c its diameter.

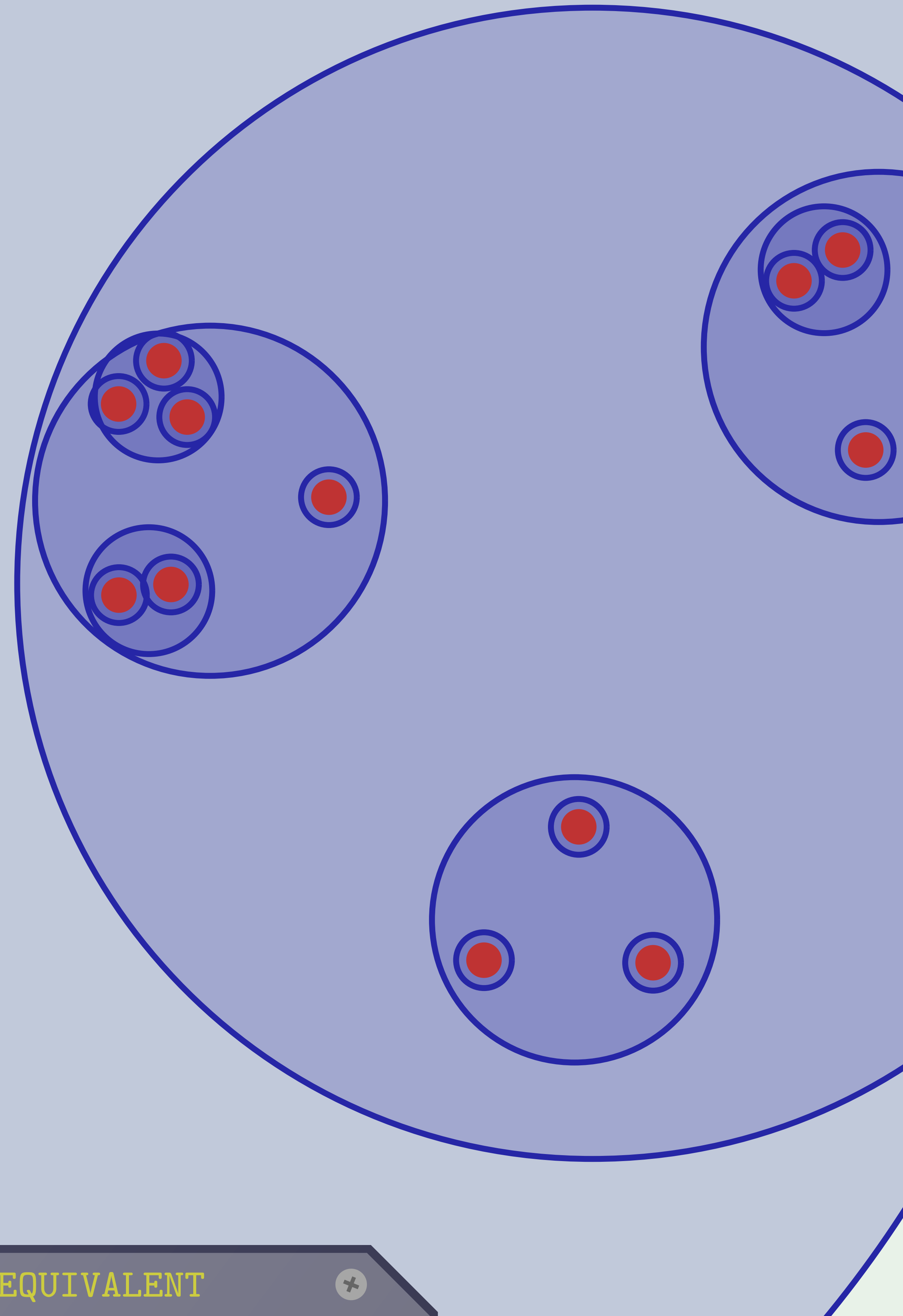
The c-clusters on P form a hierarchy.



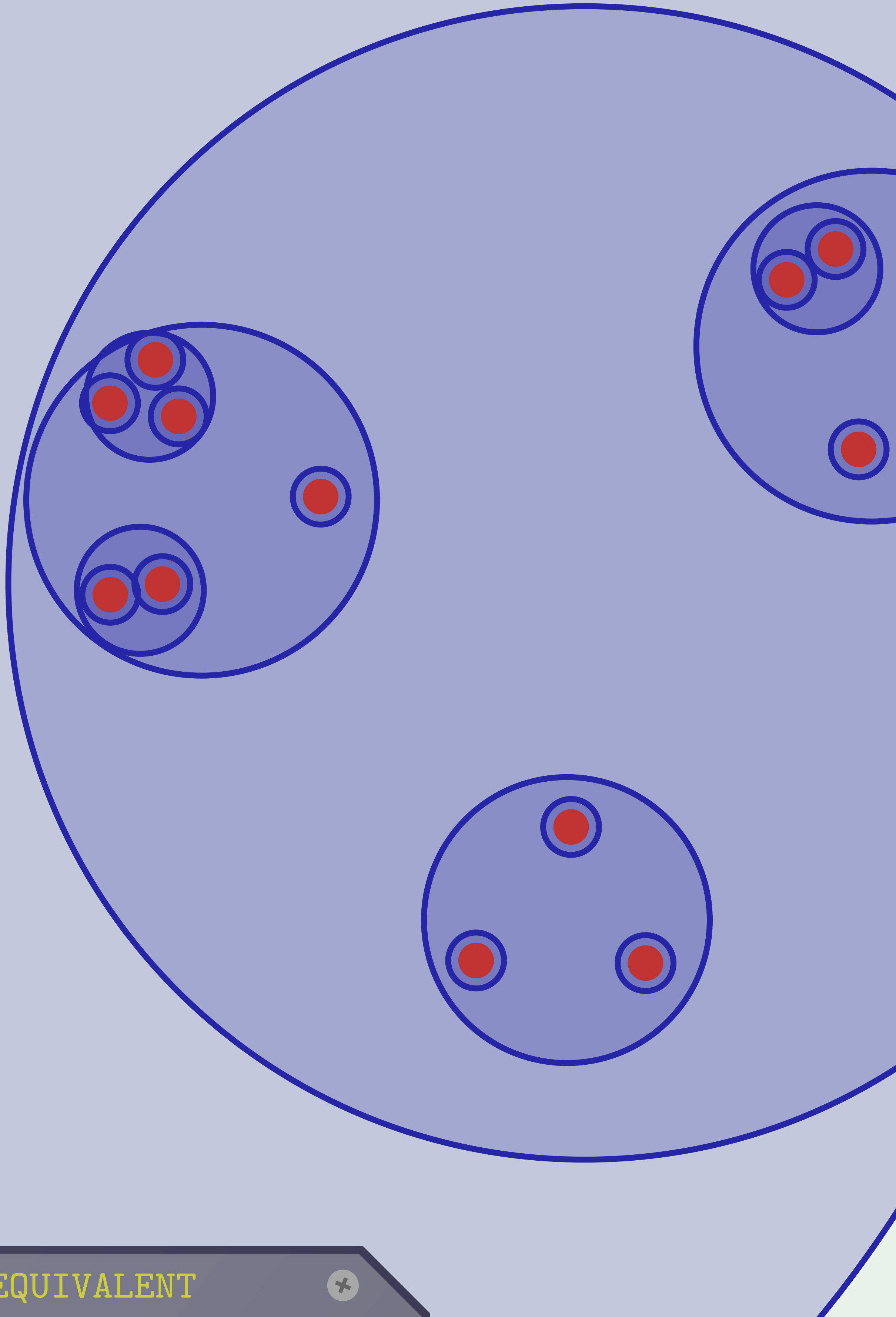
A *c-cluster* in a point set P is a subset $C \subset P$ whose distance to the rest of P is at least c its diameter.

The c-clusters on P form a hierarchy.

This c-cluster tree can have linear degree.

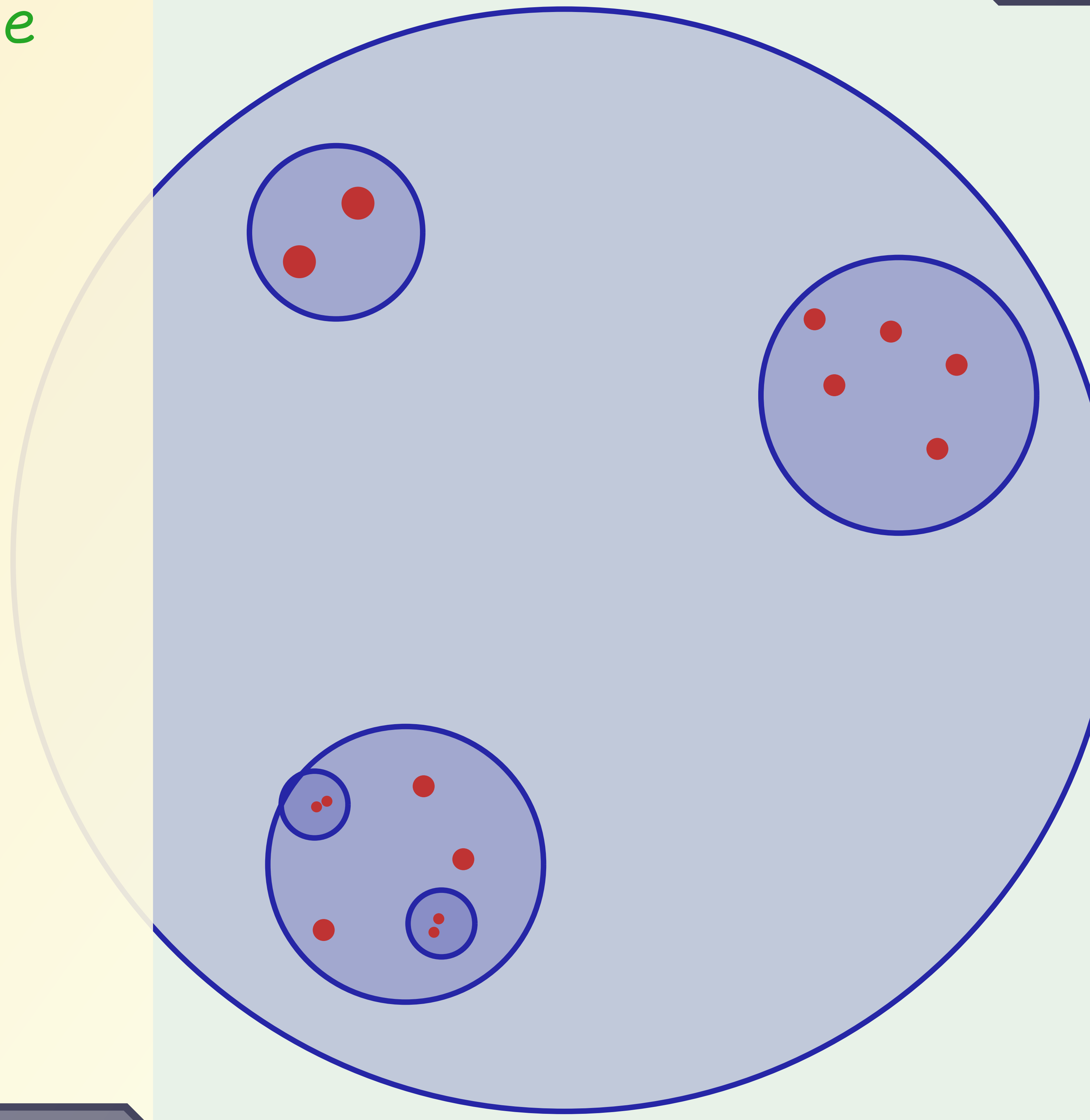


A *c-cluster quadtree* is a c-cluster tree augmented with a quadtree on its high-degree nodes.

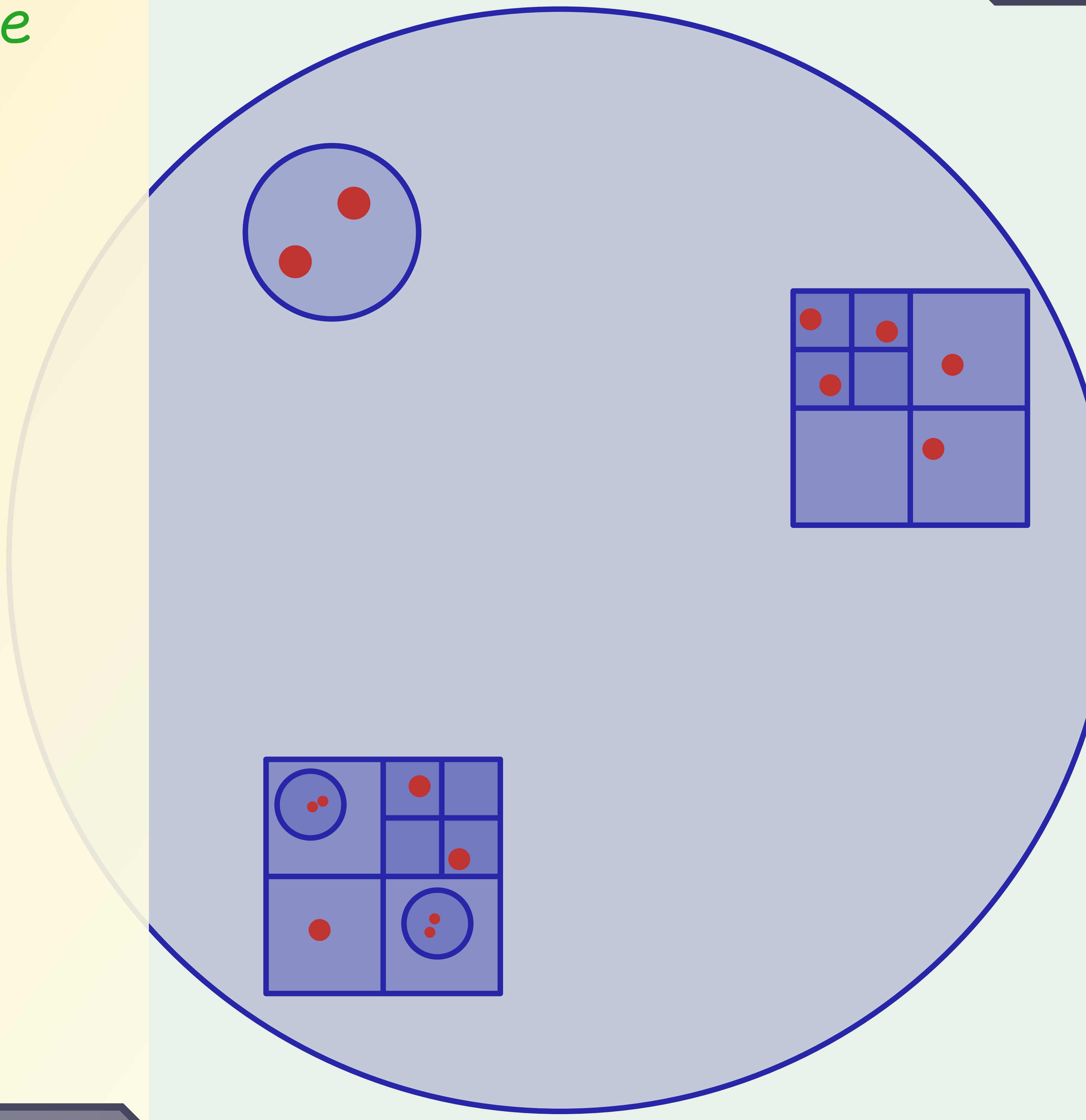


A *c-cluster quadtree* is a c-cluster tree augmented with a quadtree on its high-degree nodes.

A *c-cluster quadtree* is a c-cluster tree augmented with a quadtree on its high-degree nodes.

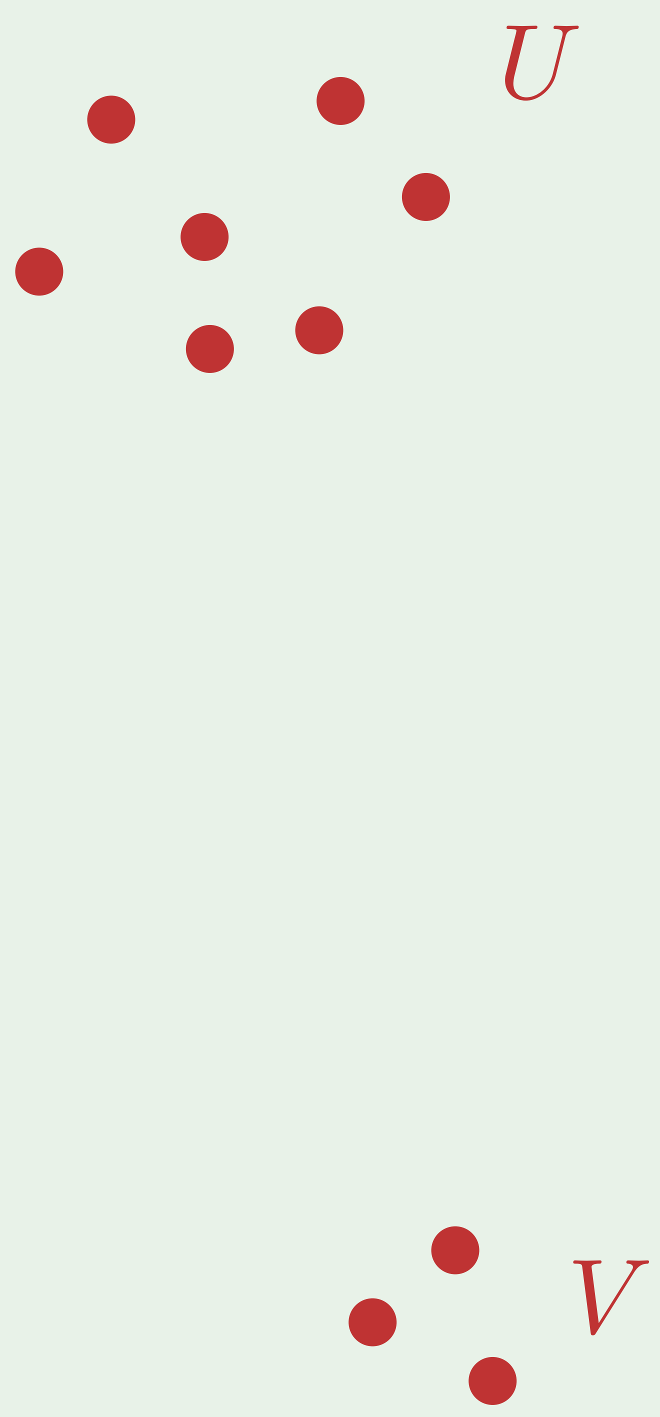


A *c-cluster quadtree* is a c-cluster tree augmented with a quadtree on its high-degree nodes.

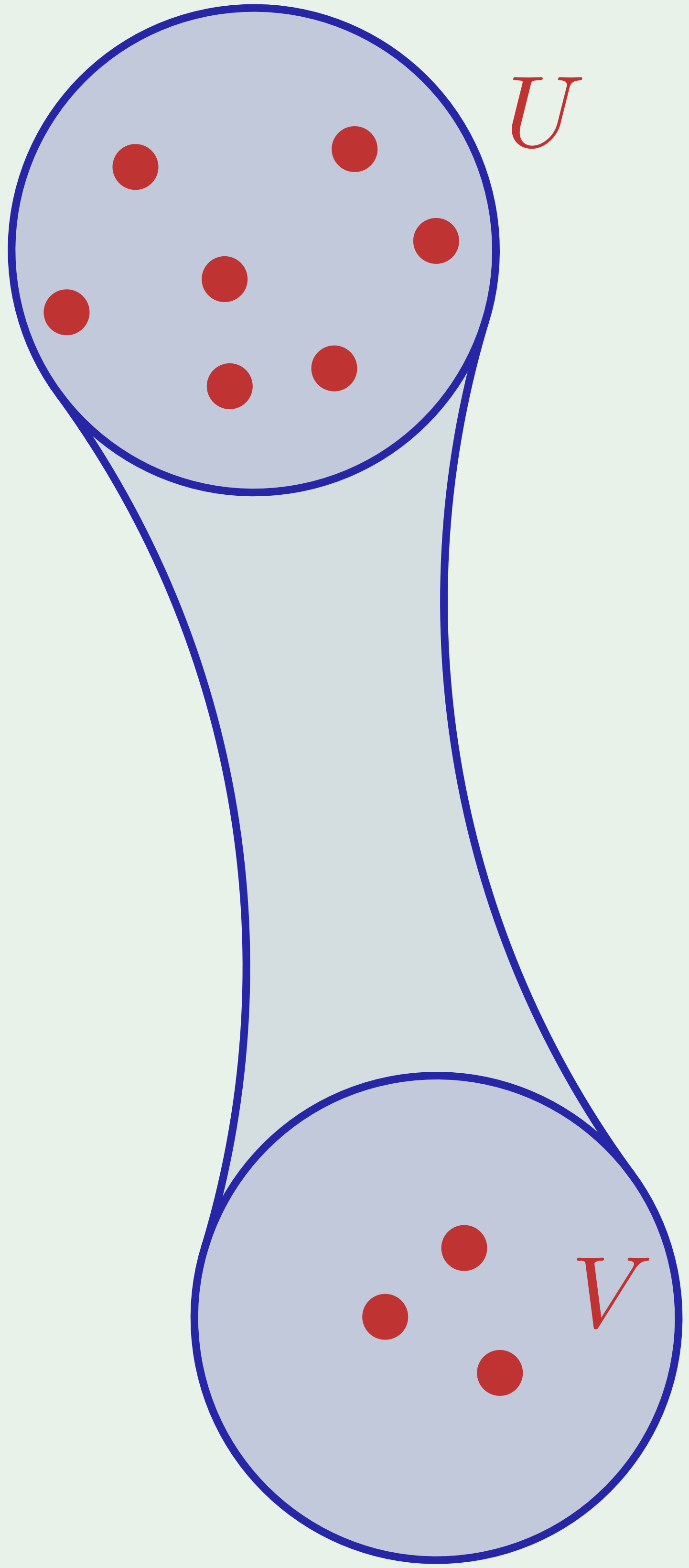


A *c-well-separated pair* is a pair of point sets U, V whose distance to each other is at least c times their diameters.

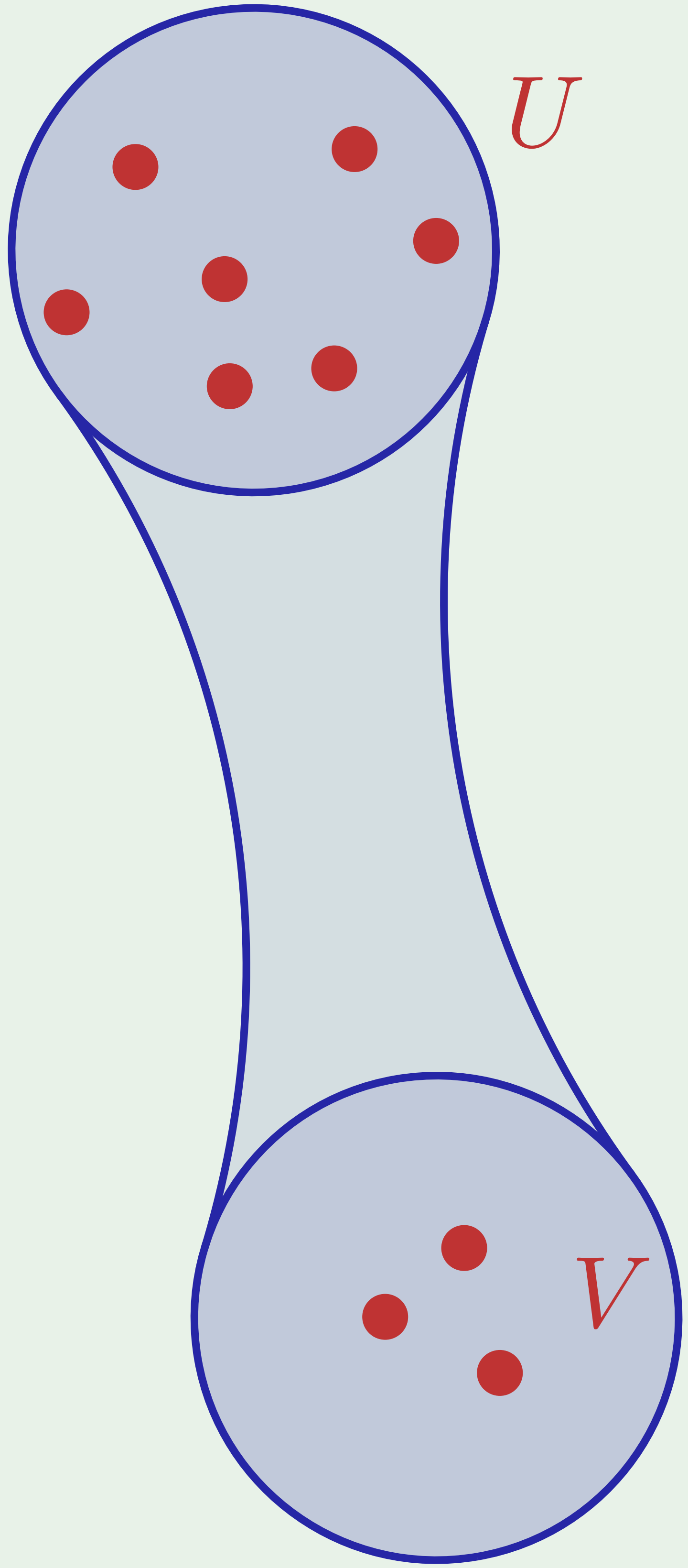
A *c-well-separated pair* is a pair of point sets U, V whose distance to each other is at least c times their diameters.



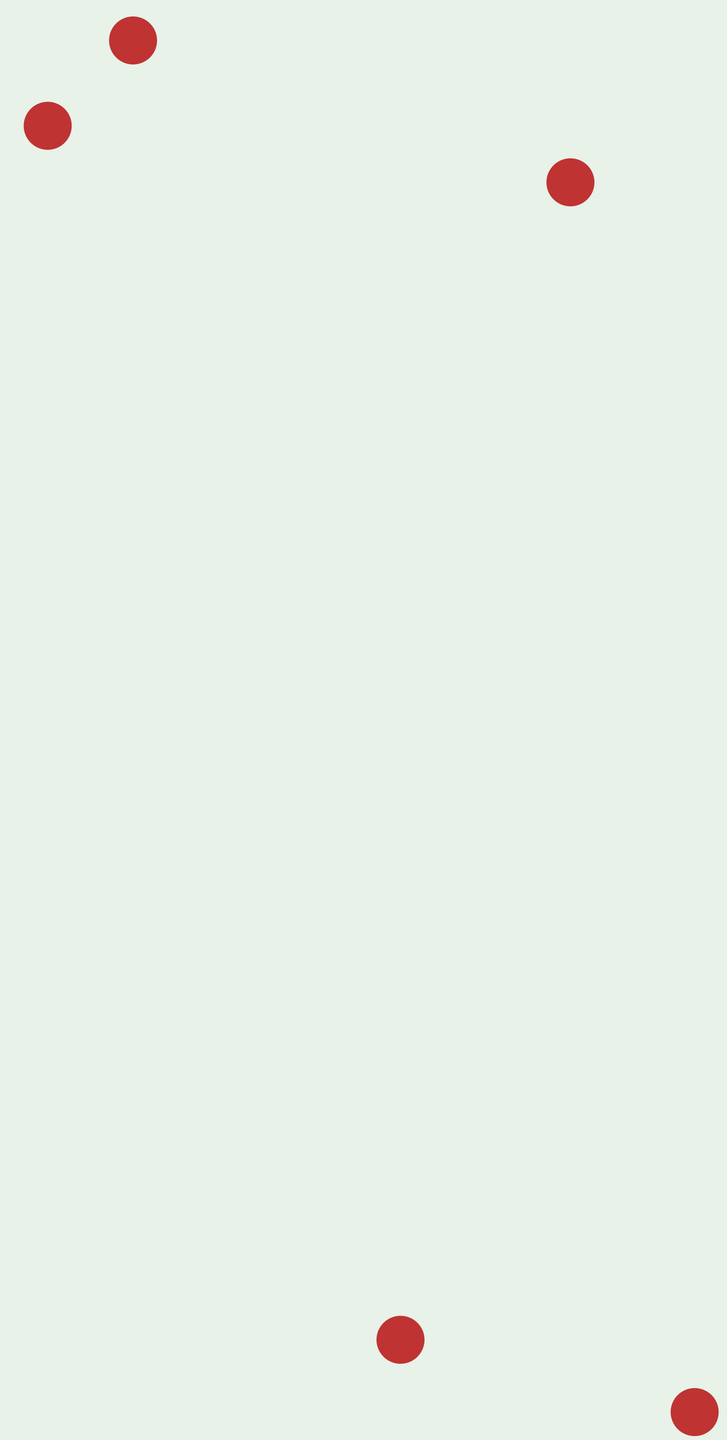
A *c-well-separated pair* is a pair of point sets U, V whose distance to each other is at least c times their diameters.



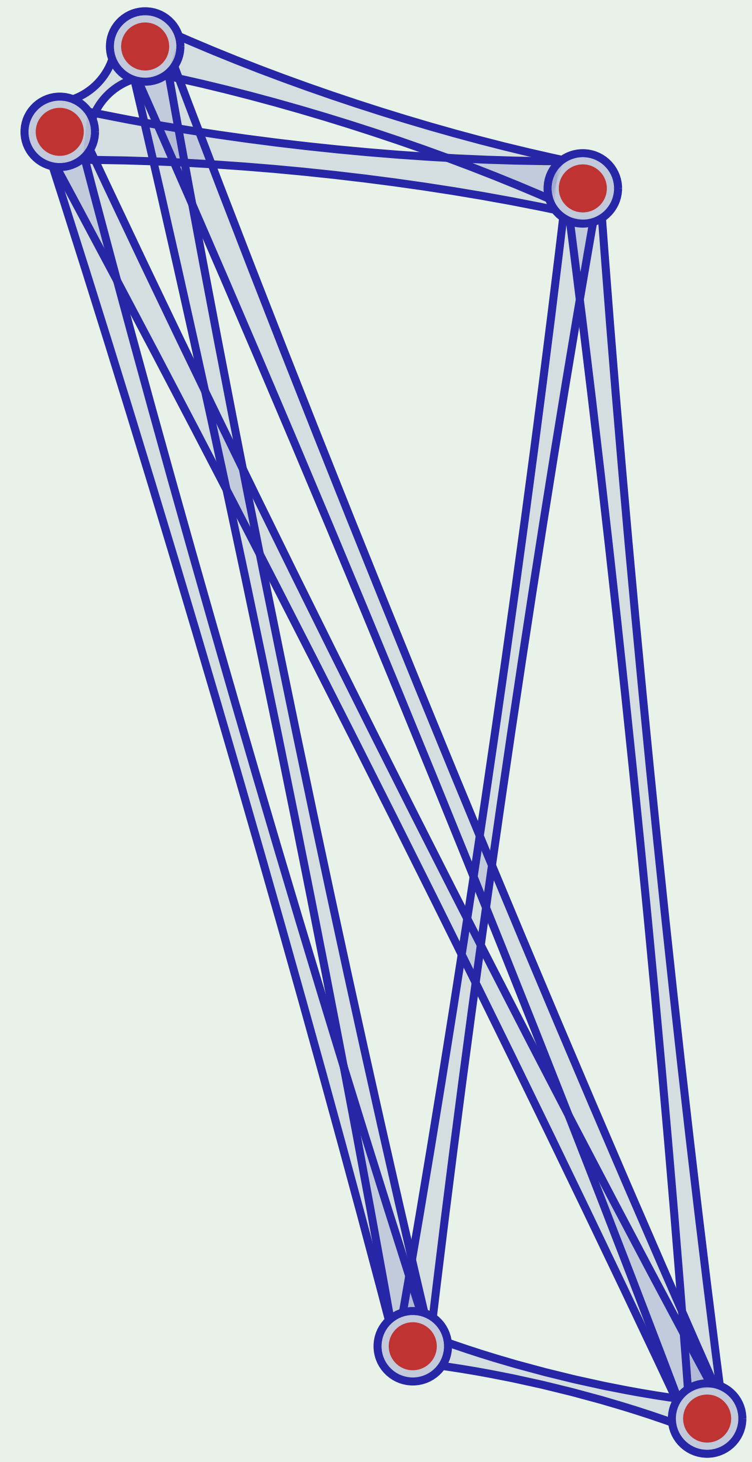
A *c-well-separated pair decomposition (WSPD)* of a set of points P is a set of well-separated pairs of subsets of P such that every pair of points $p, q \in P$ appear in exactly one pair of subsets.



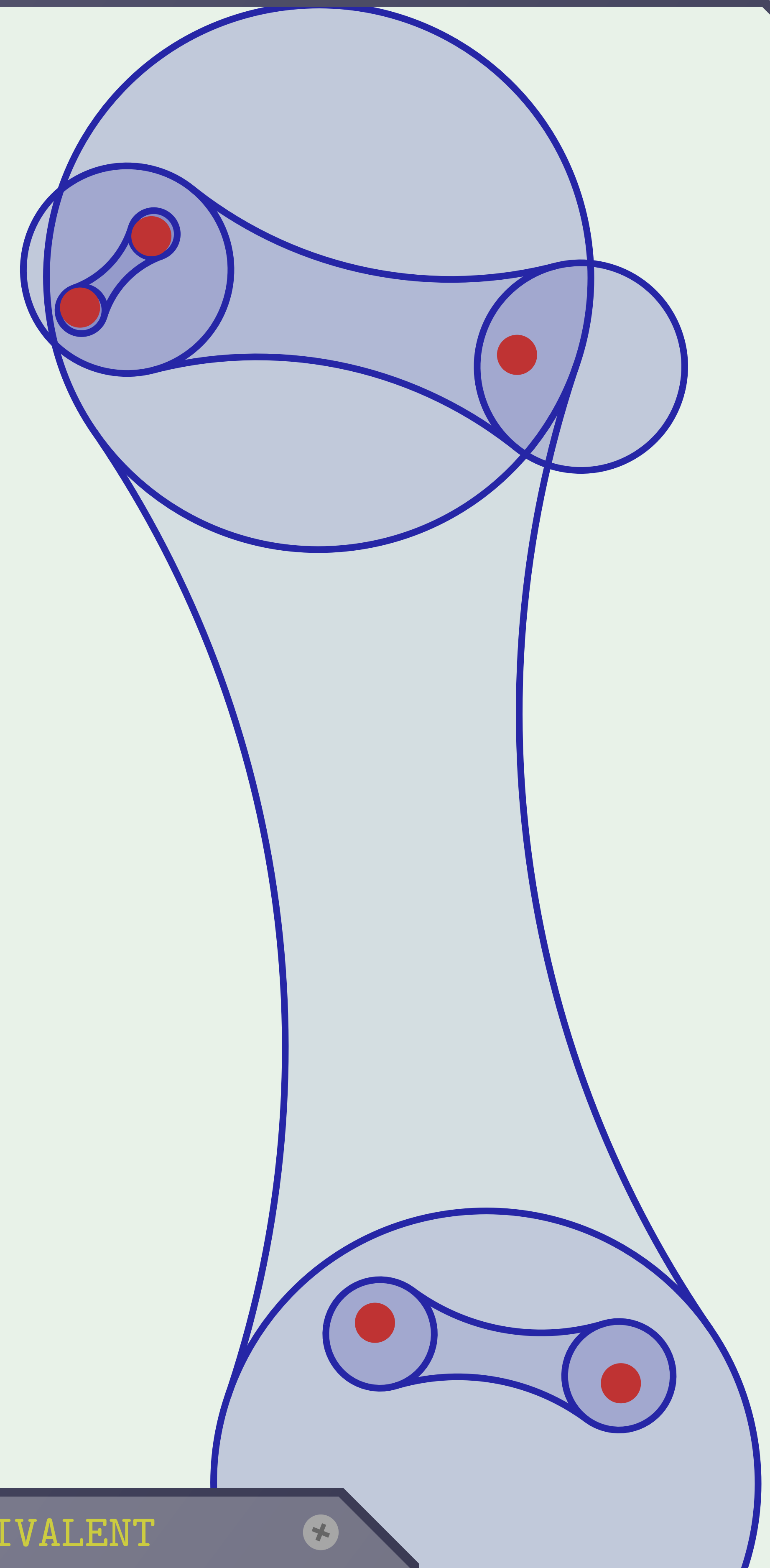
A *c-well-separated pair decomposition (WSPD)* of a set of points P is a set of well-separated pairs of subsets of P such that every pair of points $p, q \in P$ appear in exactly one pair of subsets.



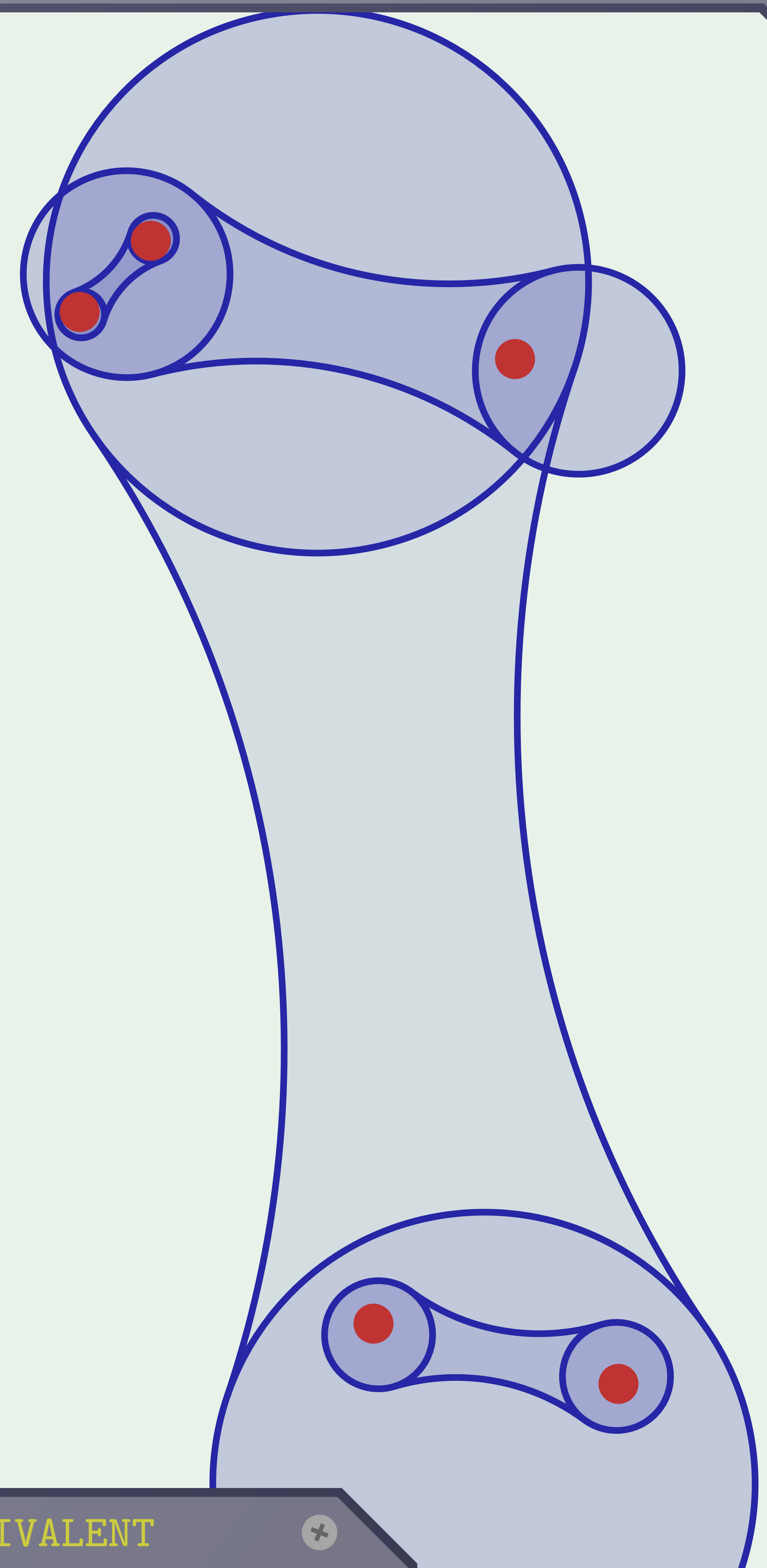
A *c-well-separated pair decomposition (WSPD)* of a set of points P is a set of well-separated pairs of subsets of P such that every pair of points $p, q \in P$ appear in exactly one pair of subsets.



A *c-well-separated pair decomposition (WSPD)* of a set of points P is a set of well-separated pairs of subsets of P such that every pair of points $p, q \in P$ appear in exactly one pair of subsets.

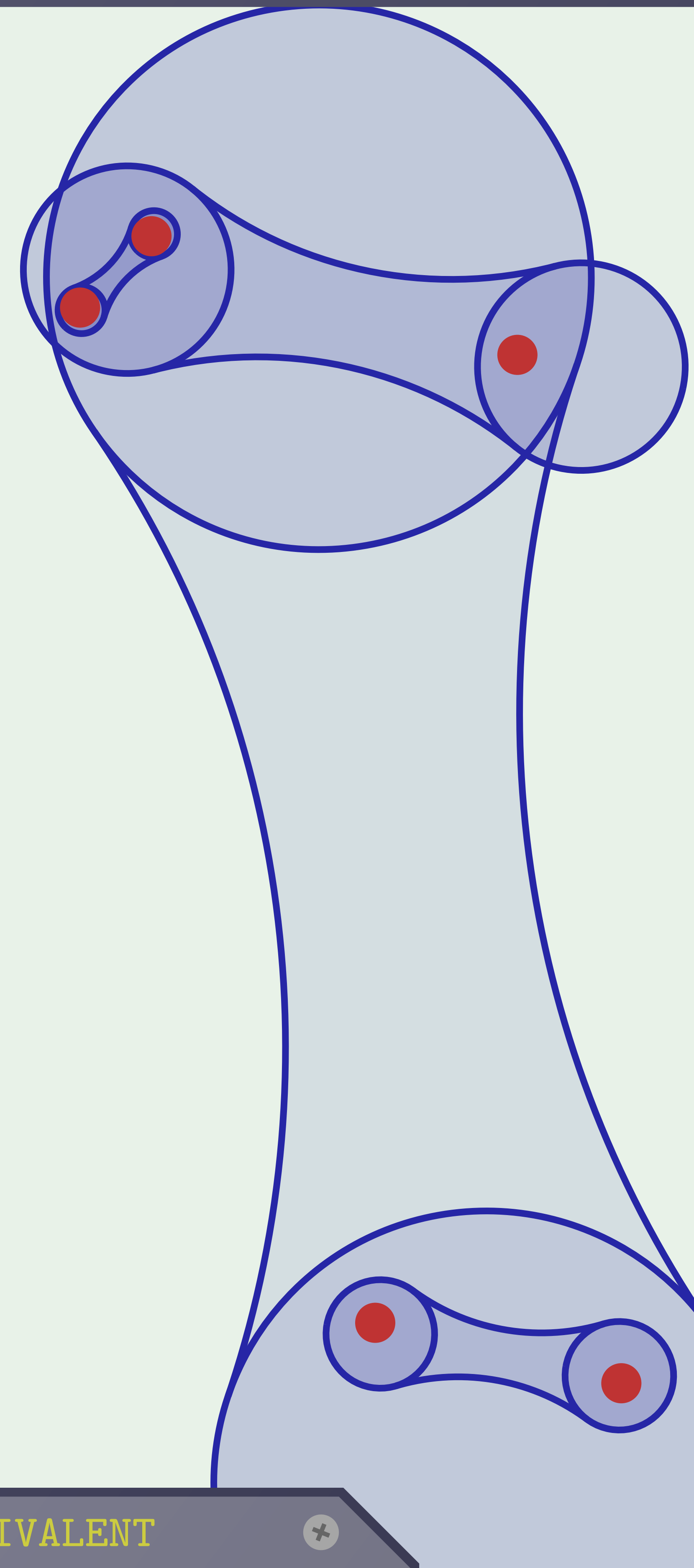


A WSPD with $O(|P|)$ pairs always exists.



A WSPD with $O(|P|)$ pairs always exists.

One way to construct one is from a hierarchical subdivision (such as a c-cluster quadtree).

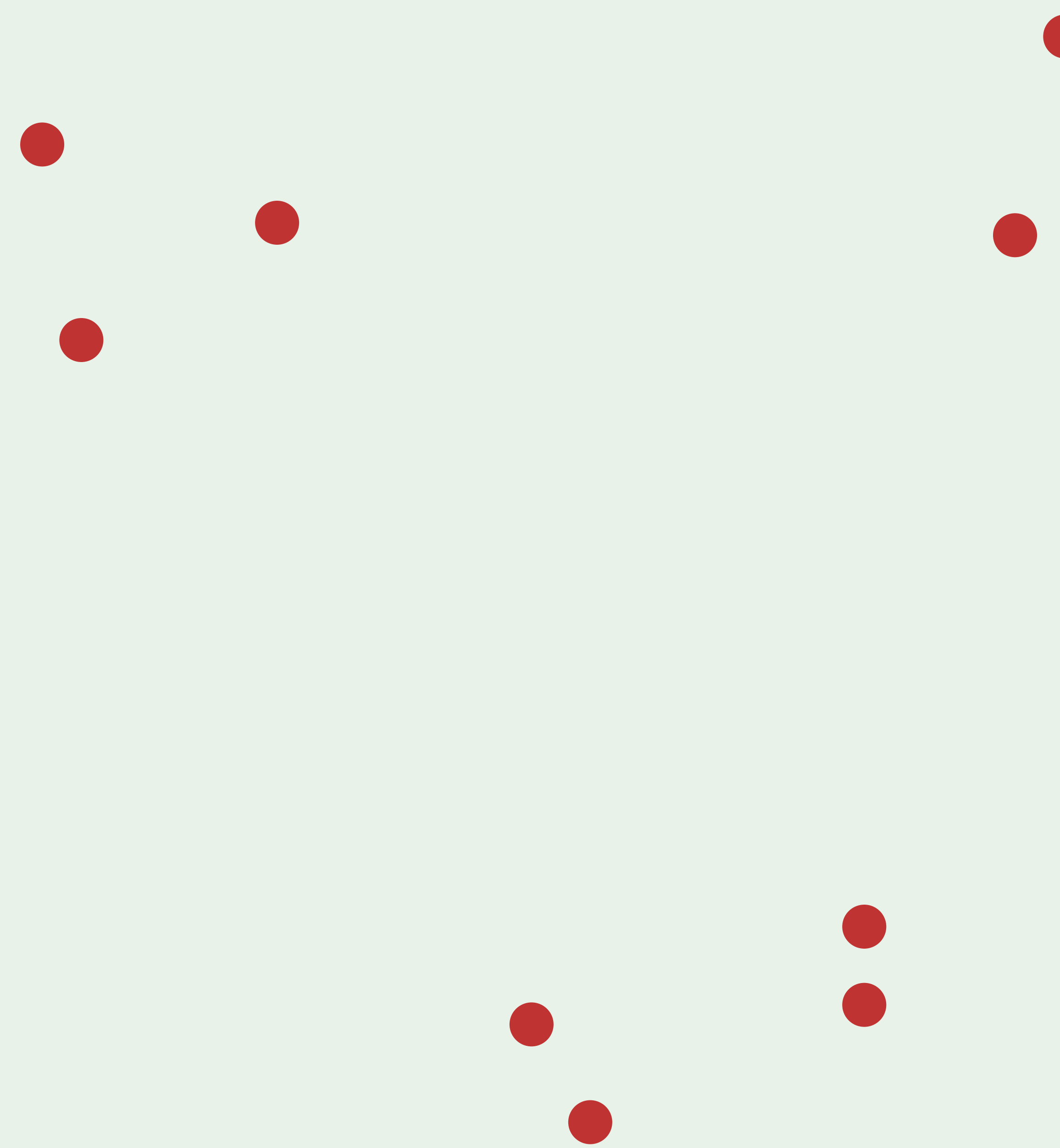


PART III

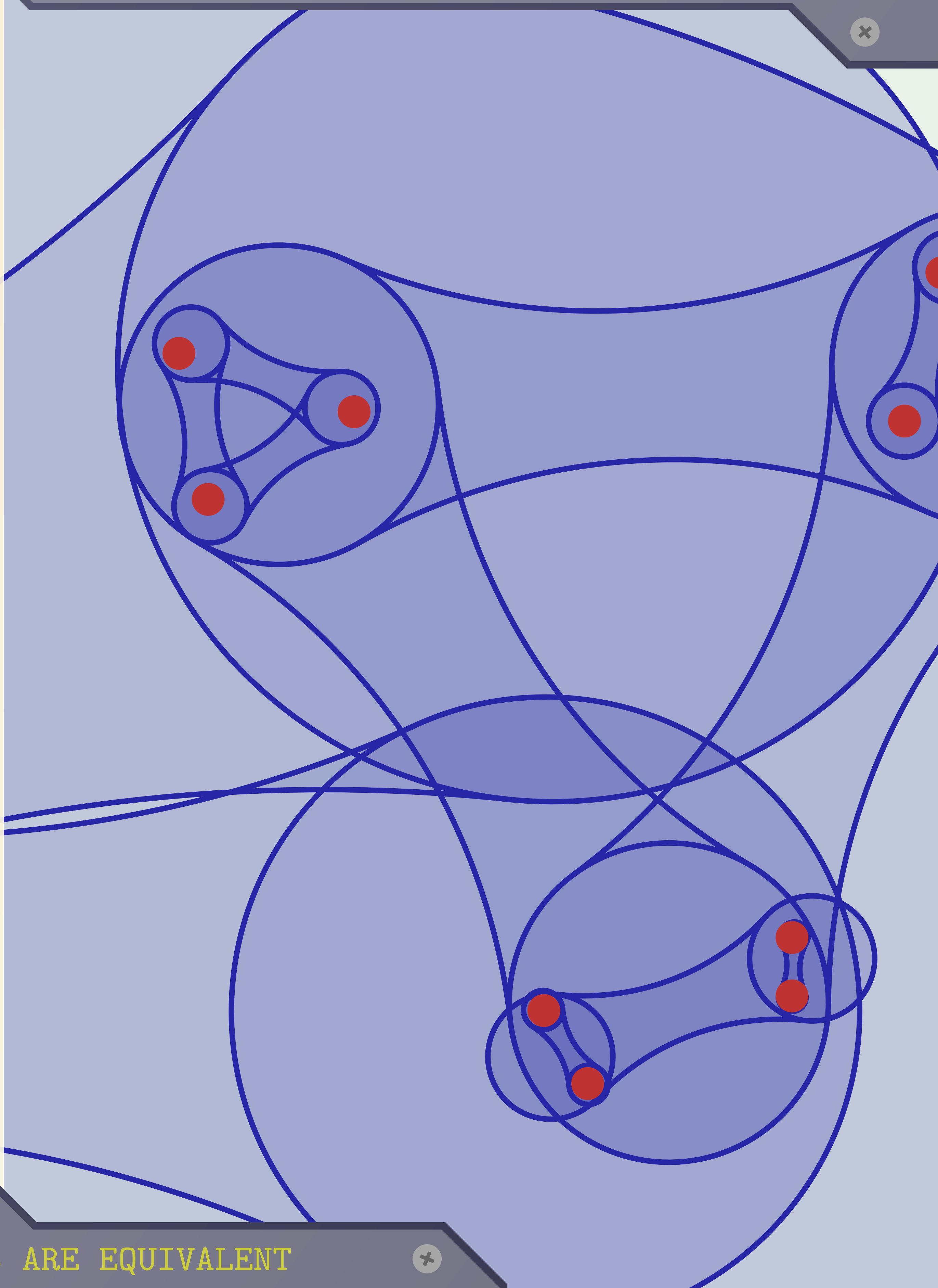
THE PART YOU'VE BEEN WAITING FOR

Suppose we have a WSPD $\mathcal{P}$ on P.

Suppose we have a WSPD $\mathcal{P}$ on P.

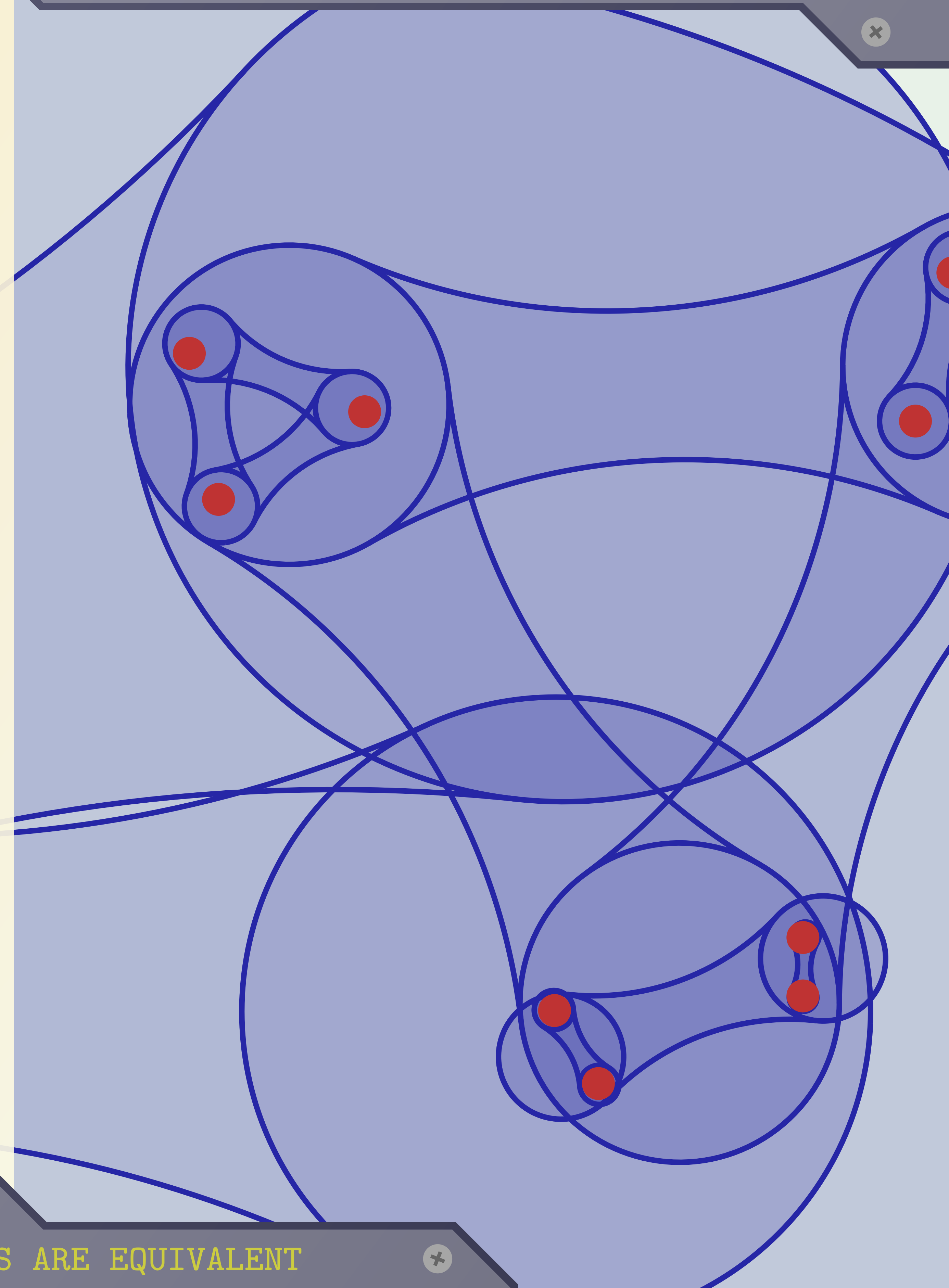


Suppose we have a WSPD $\mathcal{P}$ on P.



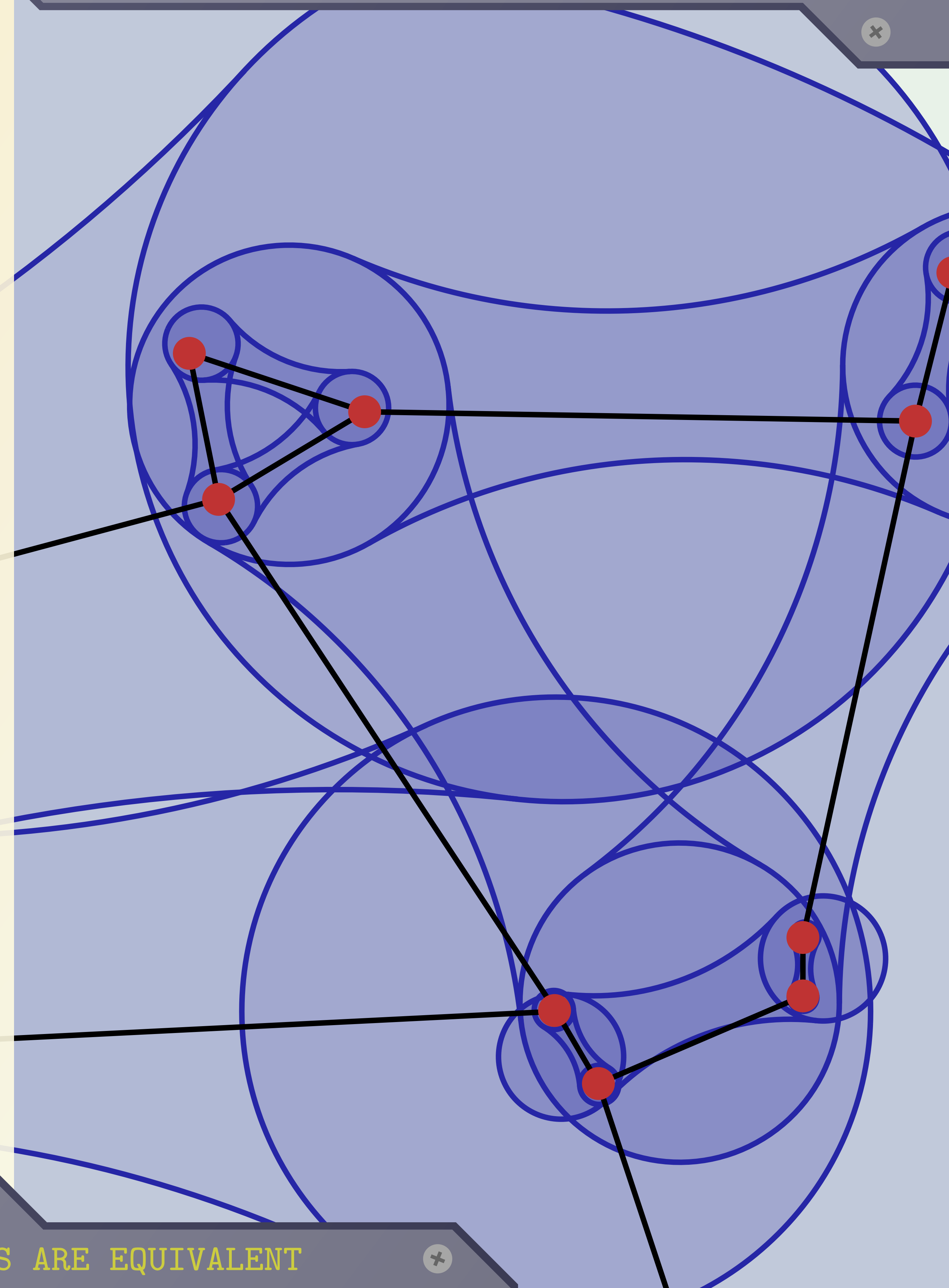
Suppose we have a WSPD $\mathcal{P}$ on P.

Let G be the graph that contains the shortest edge between two points in any pair of $\mathcal{P}$.



Suppose we have a WSPD $\mathcal{P}$ on P.

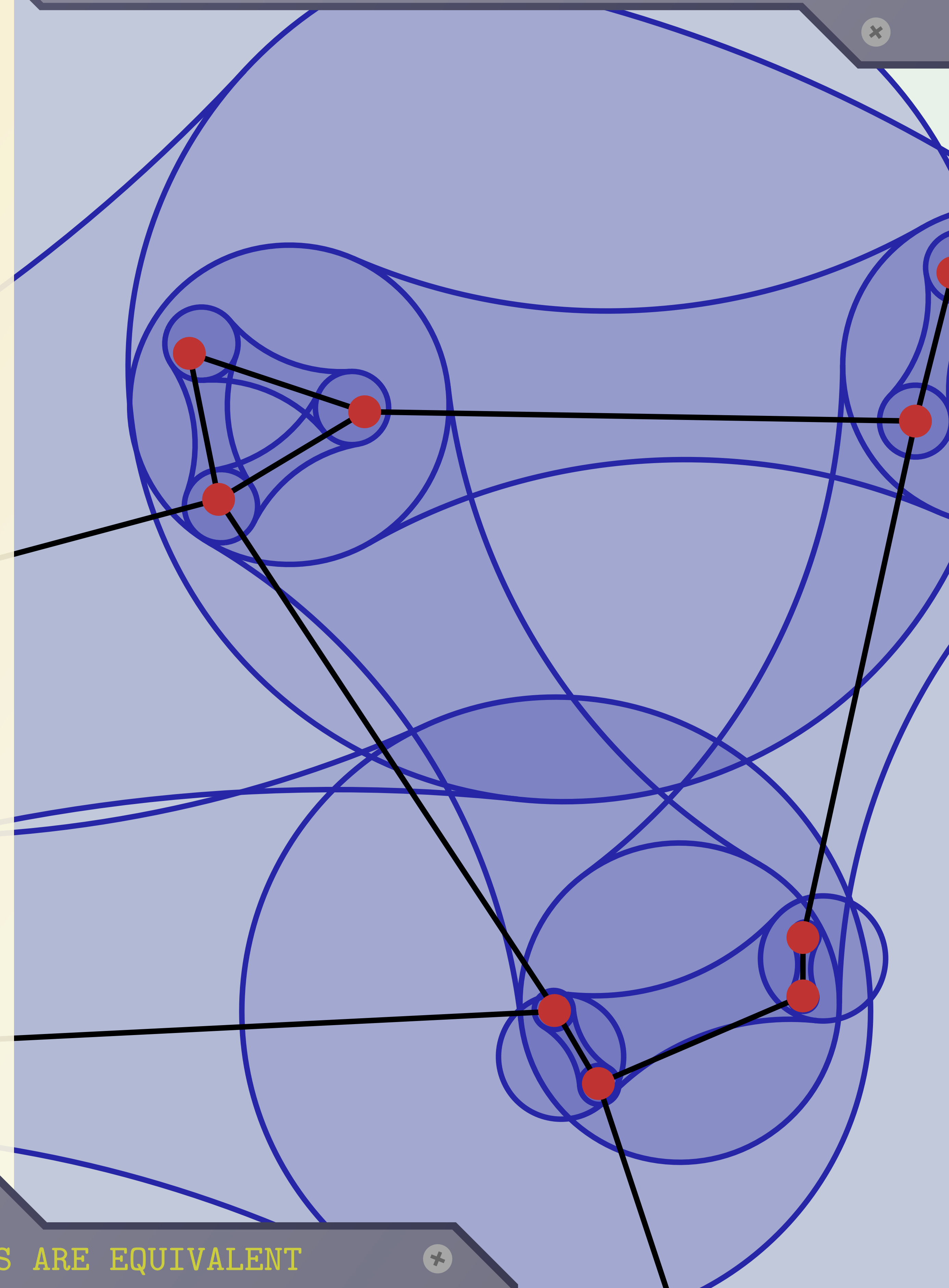
Let G be the graph that contains the shortest edge between two points in any pair of $\mathcal{P}$.



Suppose we have a WSPD $\mathcal{P}$ on P.

Let G be the graph that contains the shortest edge between two points in any pair of $\mathcal{P}$.

Claim: G has linear size and contains the MST of P.



REDUCING THE WEIGHT

Problem: $\mathcal{P}$ has quadratic weight.

Problem: $\mathcal{P}$ has quadratic weight.

Idea: most heavy pairs are far away and don't contribute to the MST anyway.

Problem: $\mathcal{P}$ has quadratic weight.

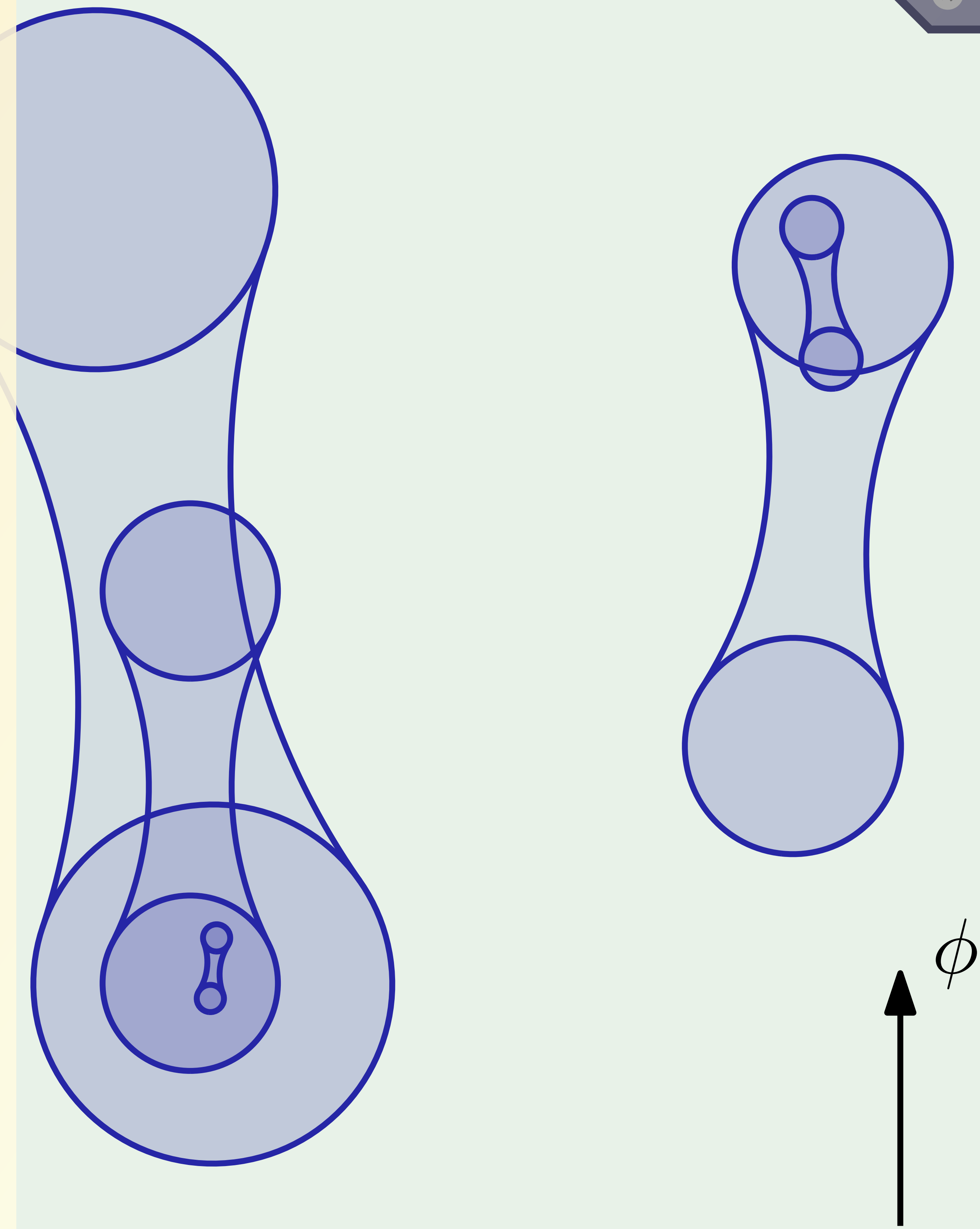
Idea: most heavy pairs are far away and don't contribute to the MST anyway.

Let $\mathcal{P}_\phi$ be the pairs in $\mathcal{P}$ with general direction ϕ.

Problem: $\mathcal{P}$ has quadratic weight.

Idea: most heavy pairs are far away and don't contribute to the MST anyway.

Let $\mathcal{P}_\phi$ be the pairs in $\mathcal{P}$ with general direction ϕ.

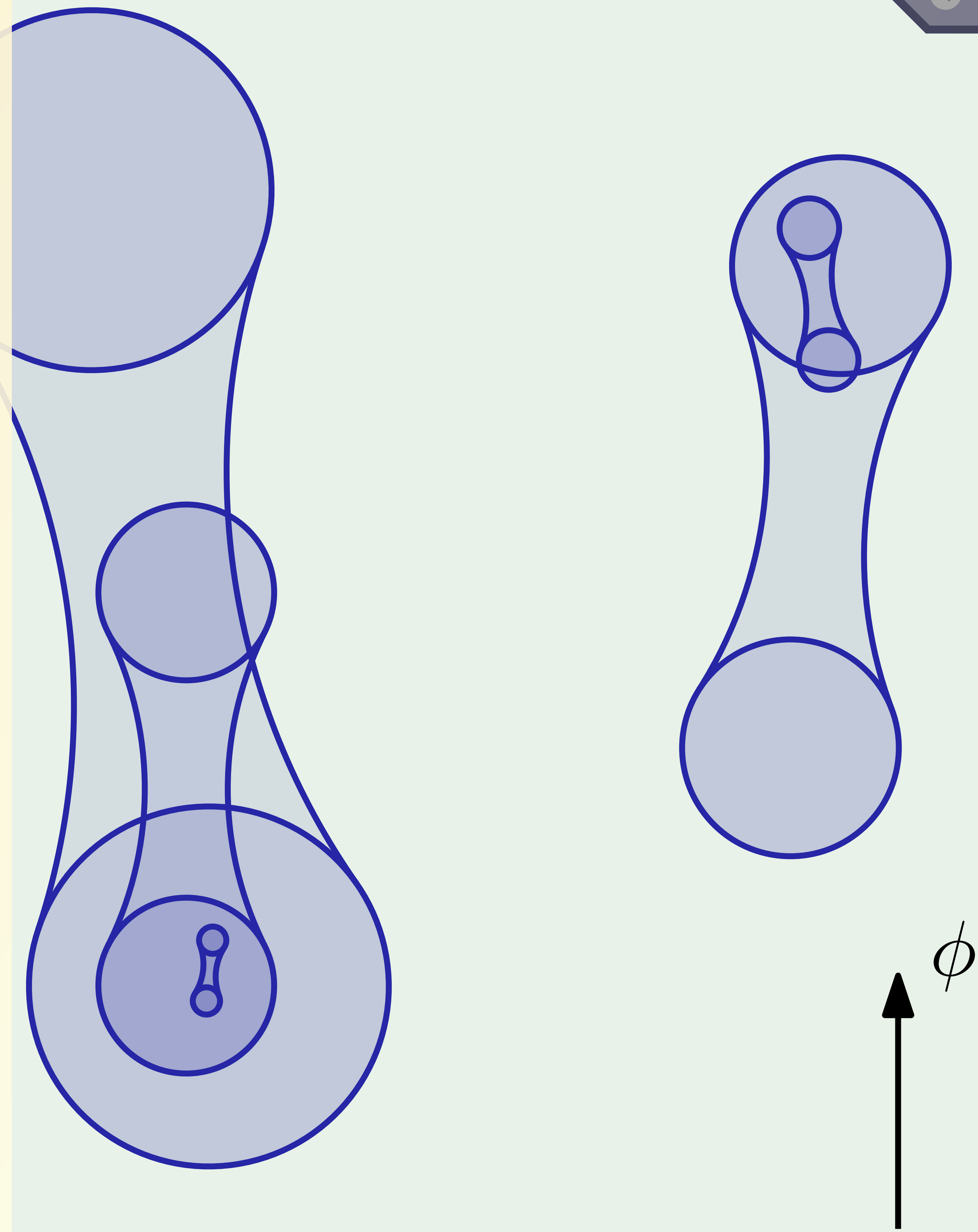


Problem: $\mathcal{P}$ has quadratic weight.

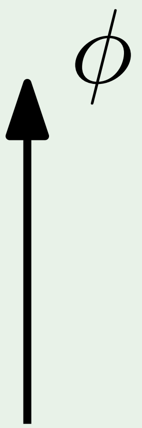
Idea: most heavy pairs are far away and don't contribute to the MST anyway.

Let $\mathcal{P}_\phi$ be the pairs in $\mathcal{P}$ with general direction ϕ.

We construct a G_ϕ separately from $\mathcal{P}_\phi$.

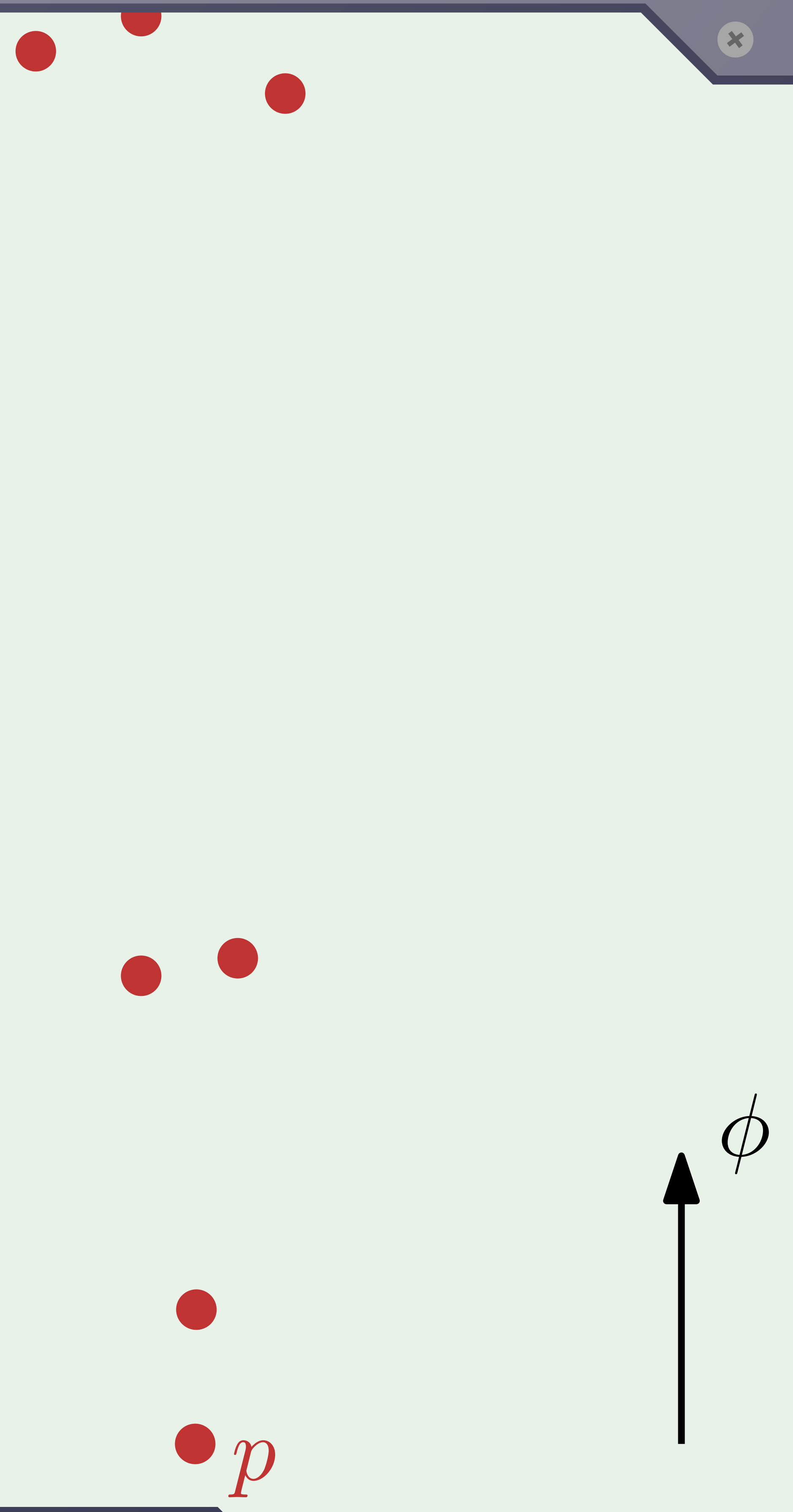


For each $p \in P$, find the k closest pairs in $\mathcal{P}_\phi$, and remove p from all other pairs.

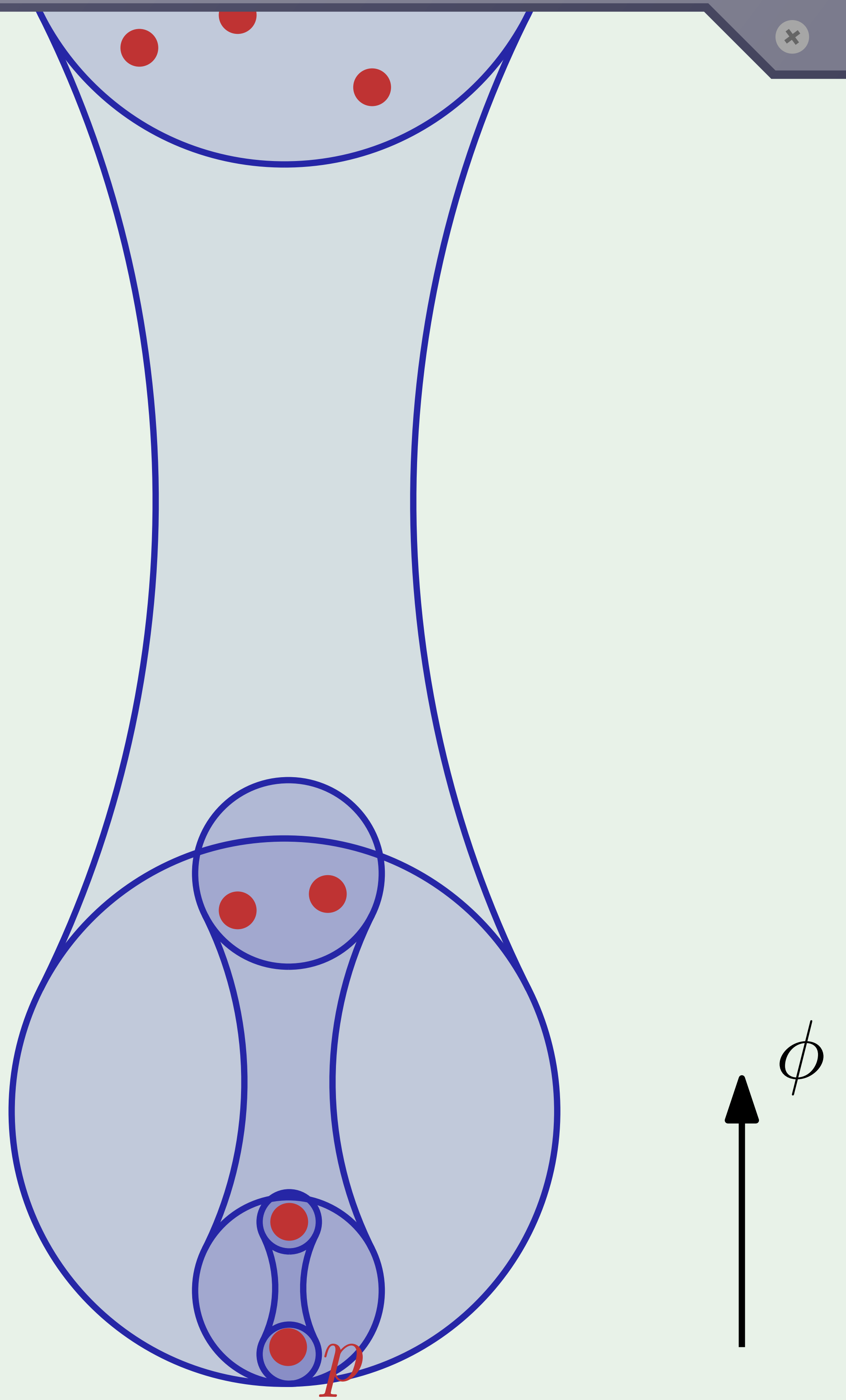


For each $p \in P$, find the k closest pairs in $\mathcal{P}_\phi$, and remove p from all other pairs.

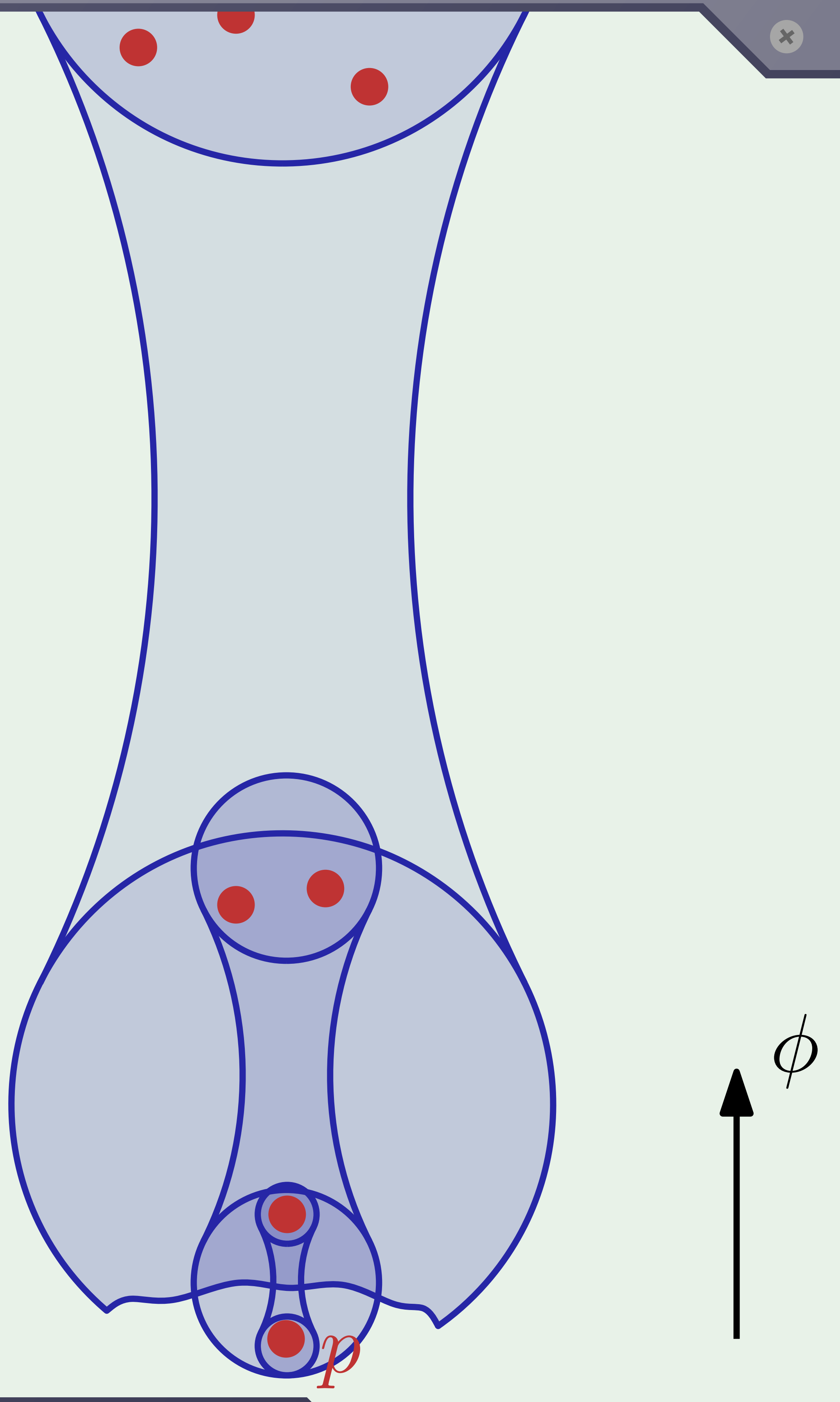
For each $p \in P$, find the k closest pairs in $\mathcal{P}_\phi$, and remove p from all other pairs.



For each $p \in P$, find the k closest pairs in $\mathcal{P}_\phi$, and remove p from all other pairs.

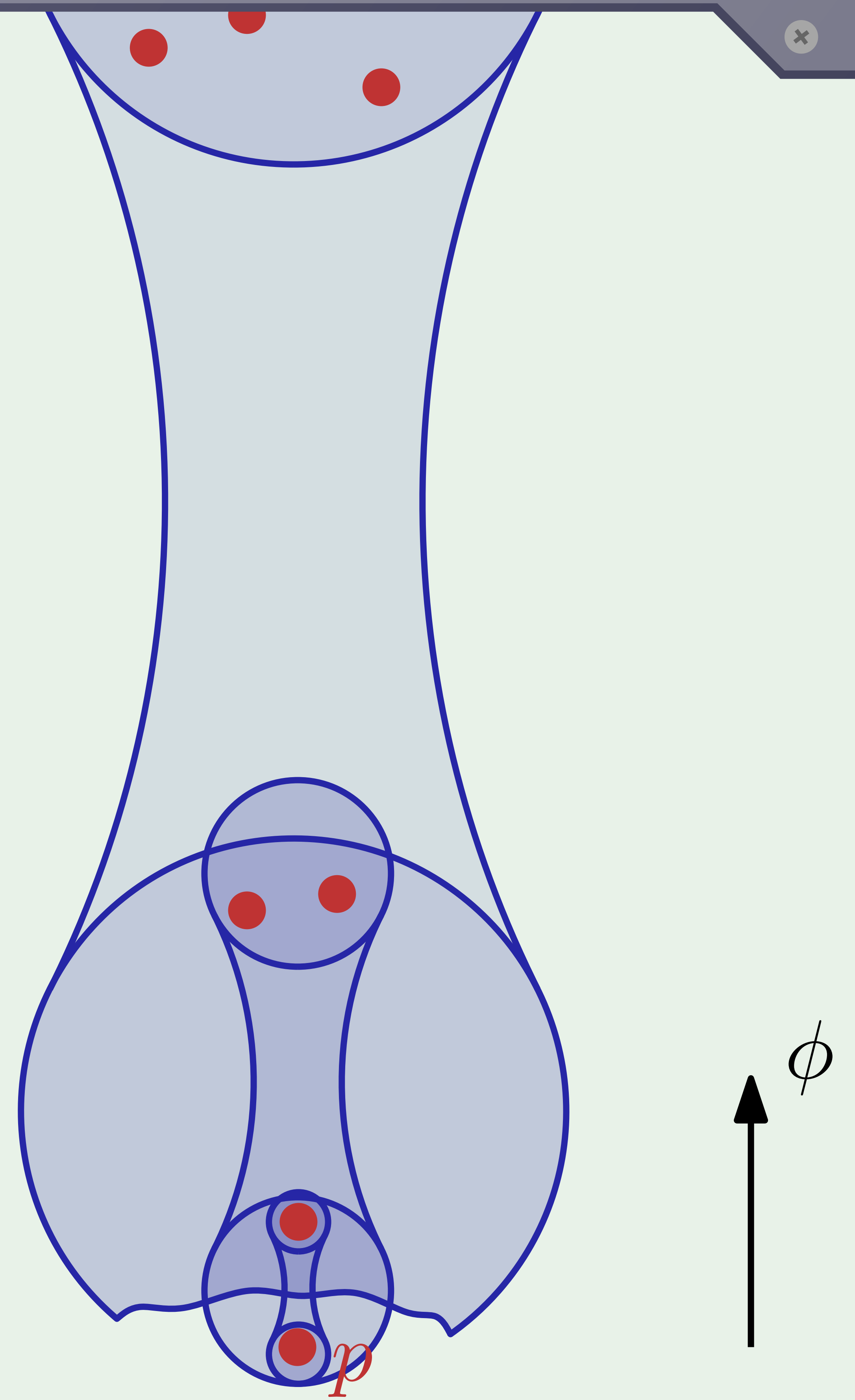


For each $p \in P$, find the k closest pairs in $\mathcal{P}_\phi$, and remove p from all other pairs.



For each $p \in P$, find the k closest pairs in $\mathcal{P}_\phi$, and remove p from all other pairs.

Let $\mathcal{P}'_\phi$ be the resulting pair set.

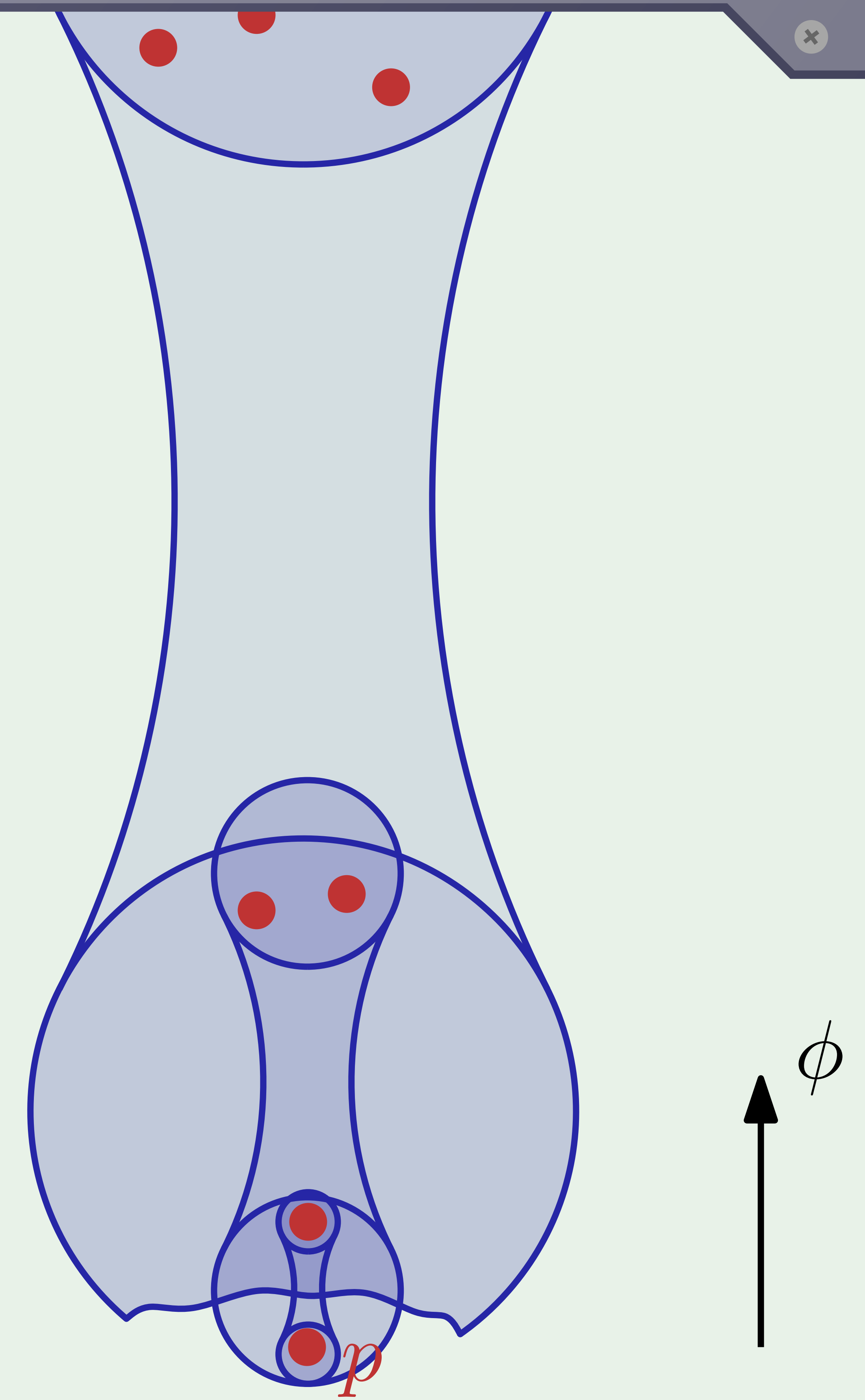


For each $p \in P$, find the k closest pairs in $\mathcal{P}_\phi$, and remove p from all other pairs.

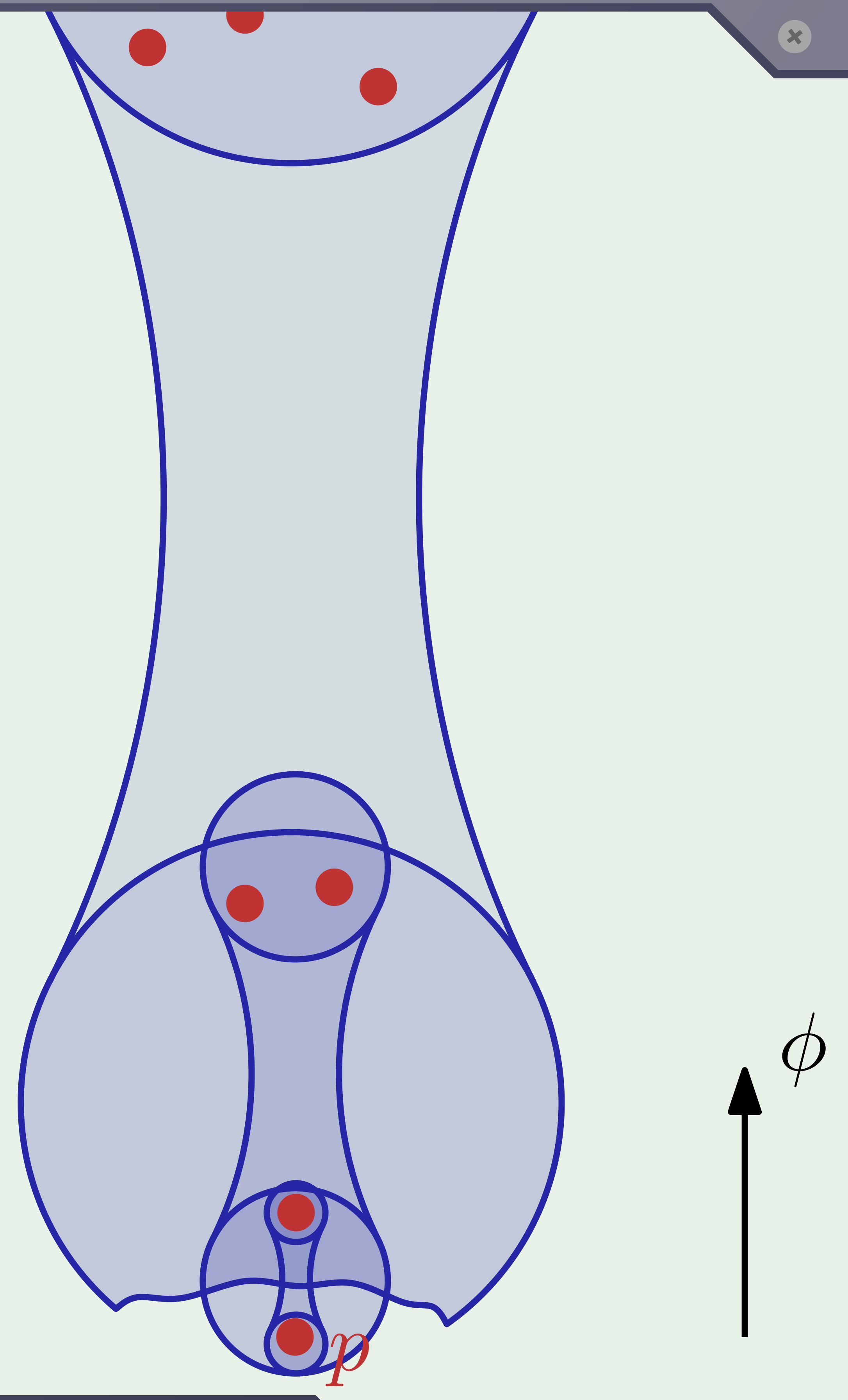
Let $\mathcal{P}'_\phi$ be the resulting pair set.

$\mathcal{P}'$, the union of $\mathcal{P}'_\phi$ over all ϕ, is no longer a pair decomposition of P.

But...

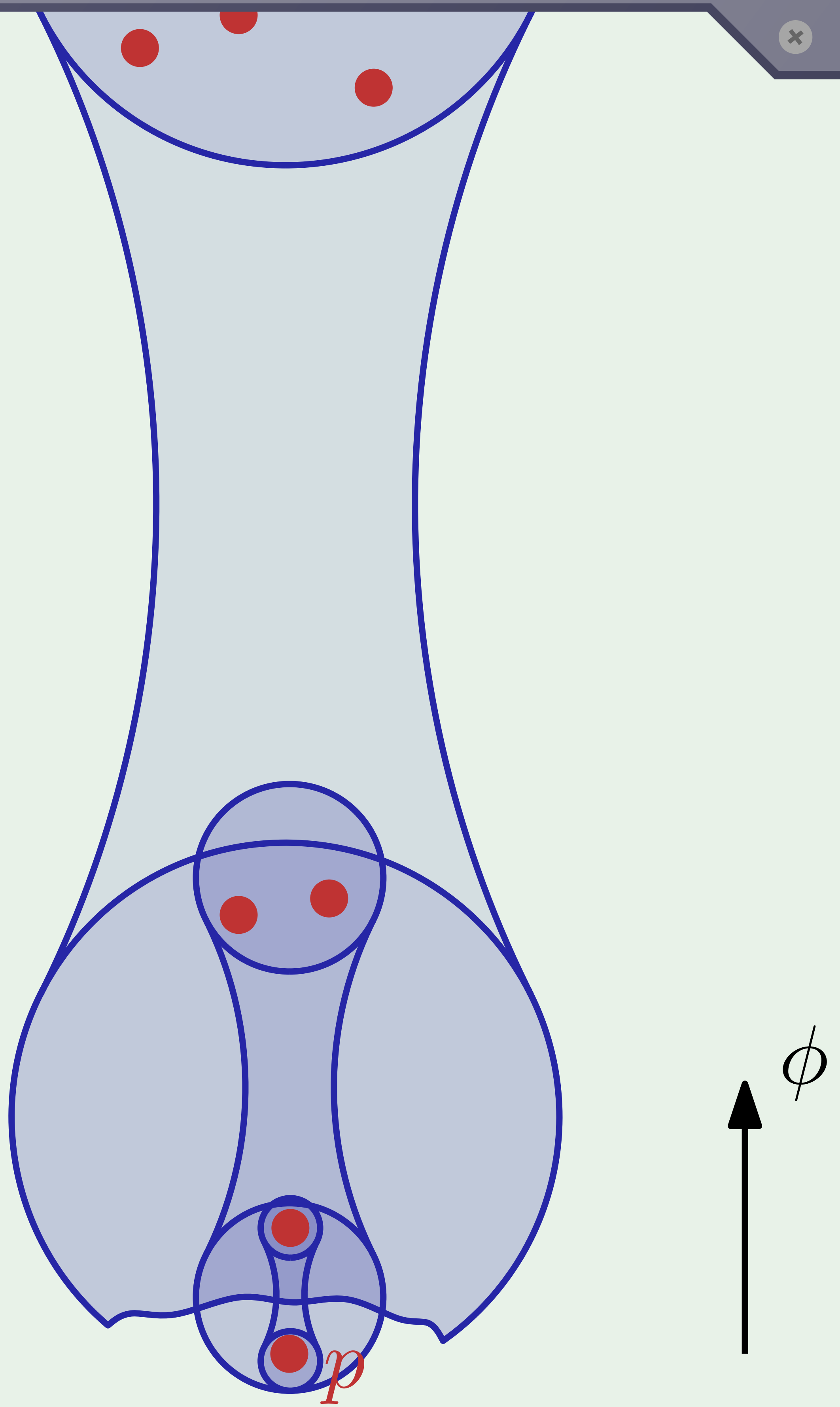


Claim: $\mathcal{P}'$ has linear weight.



Claim: $\mathcal{P}'$ has linear weight.

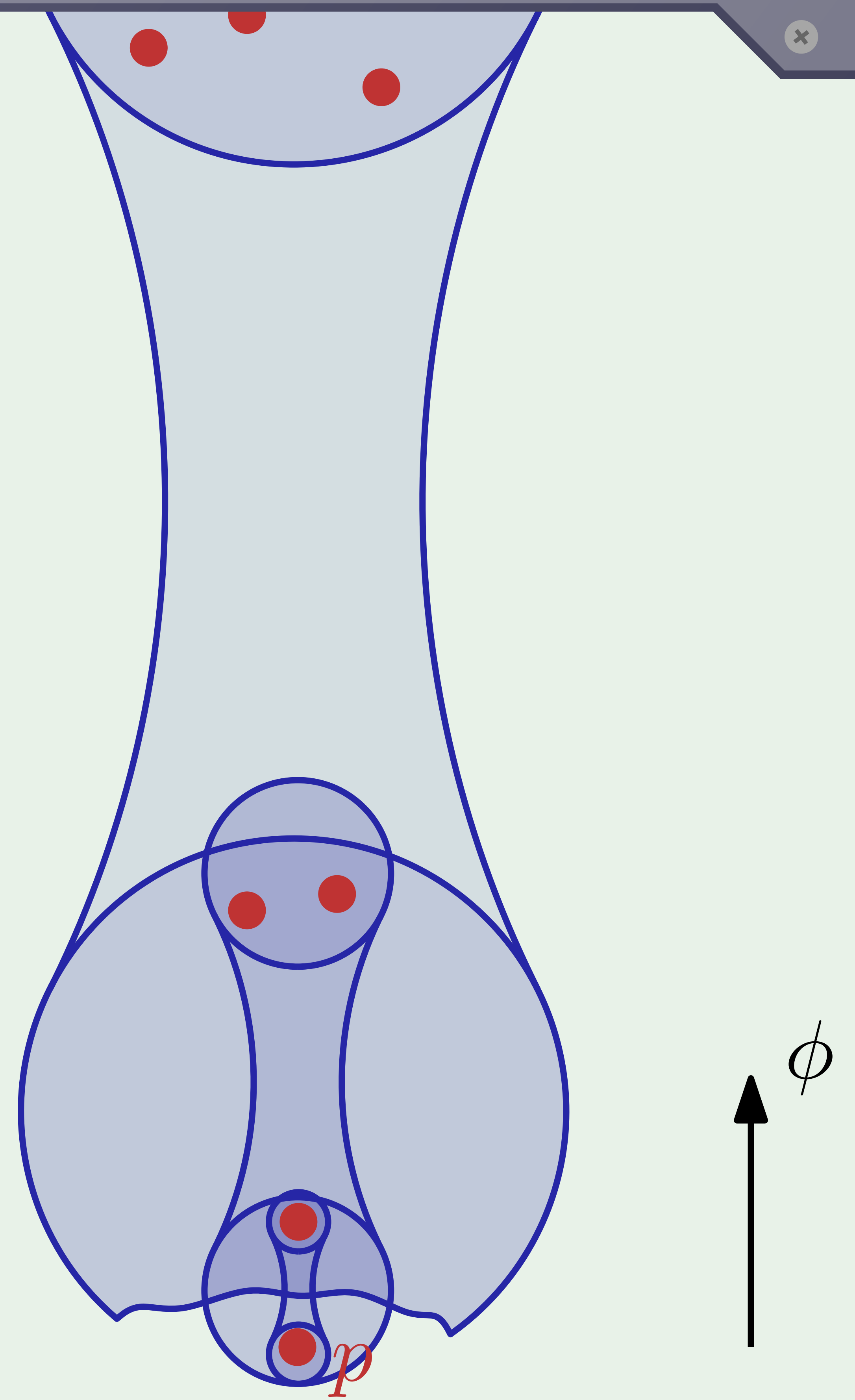
Claim: $\mathcal{P}'$ still contains all edges of the MST of P.

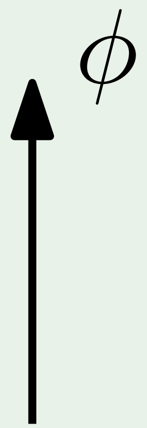


Claim: $\mathcal{P}'$ has linear weight.

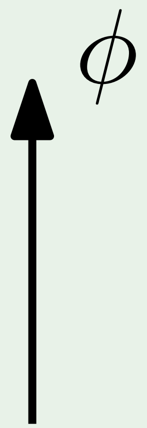
Claim: $\mathcal{P}'$ still contains all edges of the MST of P.

Claim: and as if that wasn't cool enough, $\mathcal{P}'$ can even be computed in linear time.

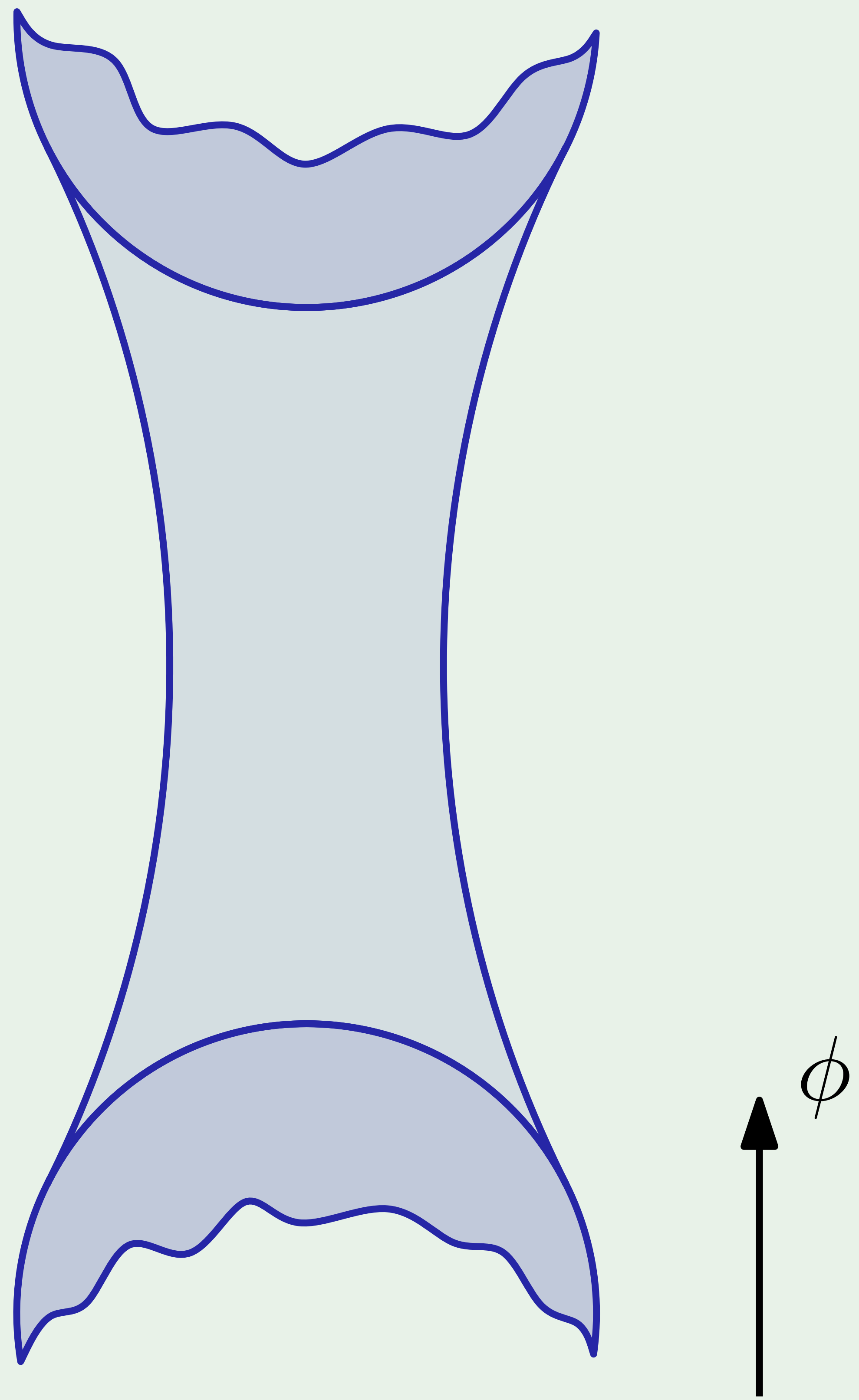



ϕ

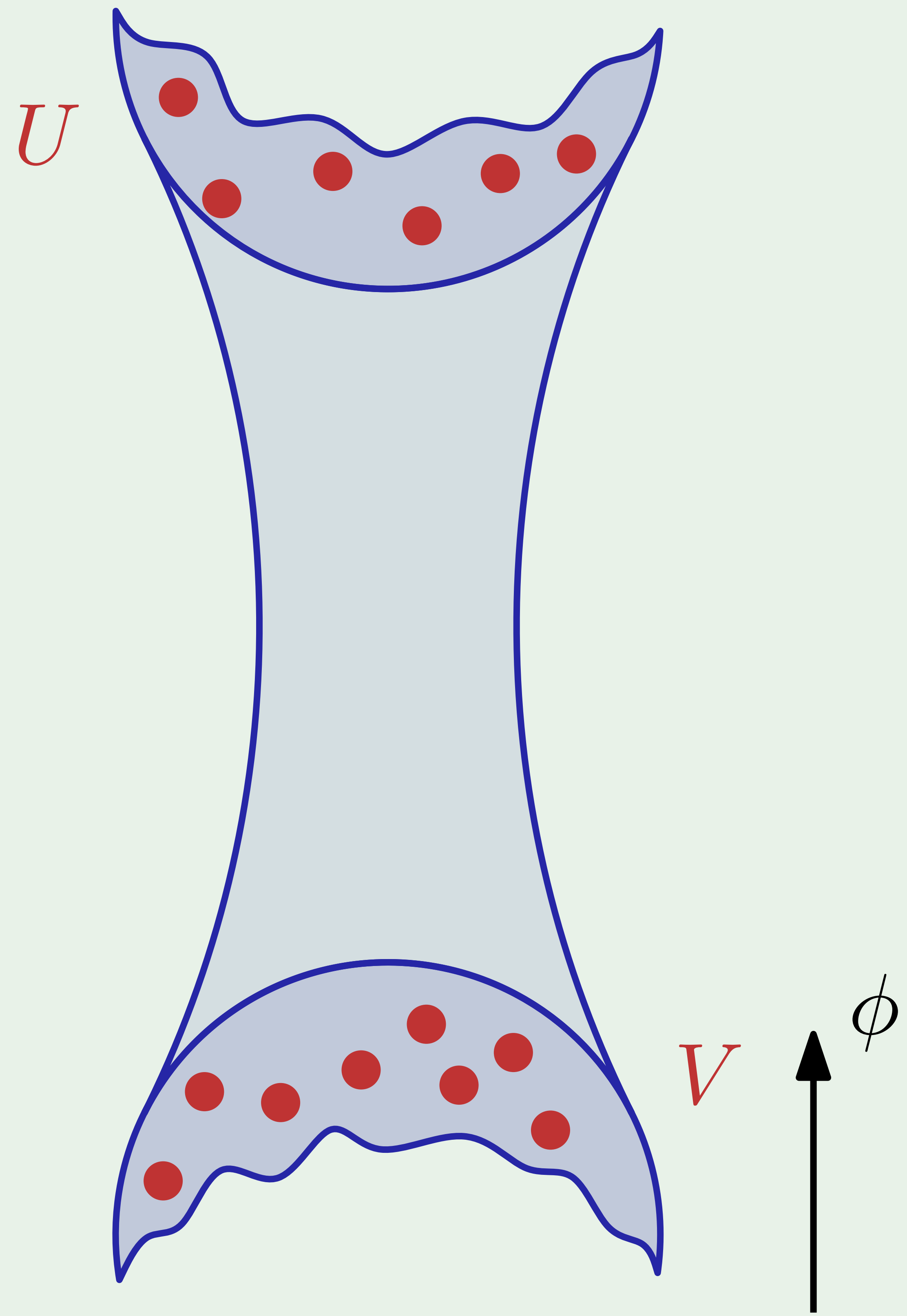
Now, for each pair $(U, V) \in \mathcal{P}'_\phi$, we need to find the closest pair of points.



Now, for each pair $(U, V) \in \mathcal{P}'_\phi$, we need to find the closest pair of points.

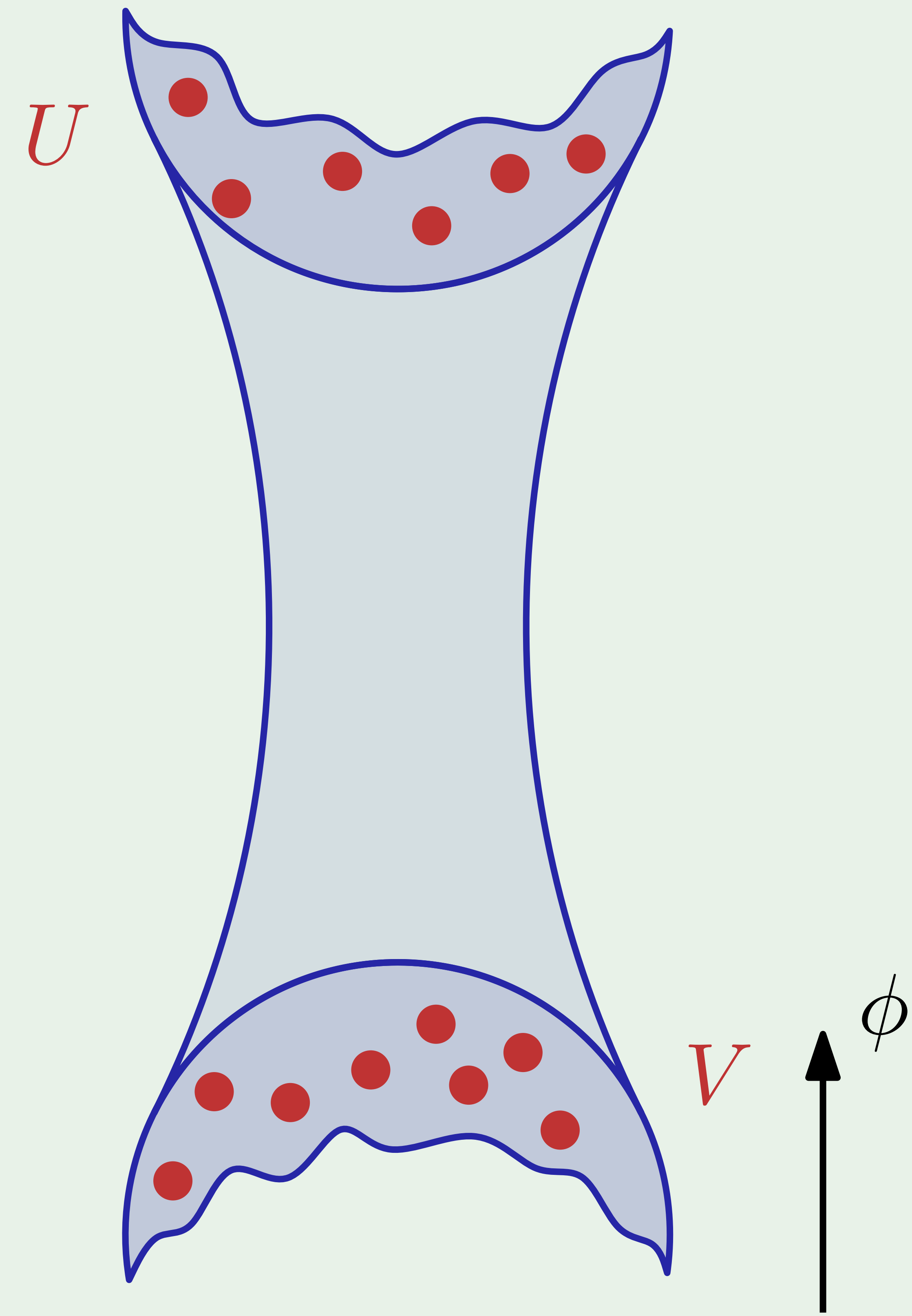


Now, for each pair $(U, V) \in \mathcal{P}'_\phi$, we need to find the closest pair of points.



Now, for each pair $(U, V) \in \mathcal{P}'_\phi$, we need to find the closest pair of points.

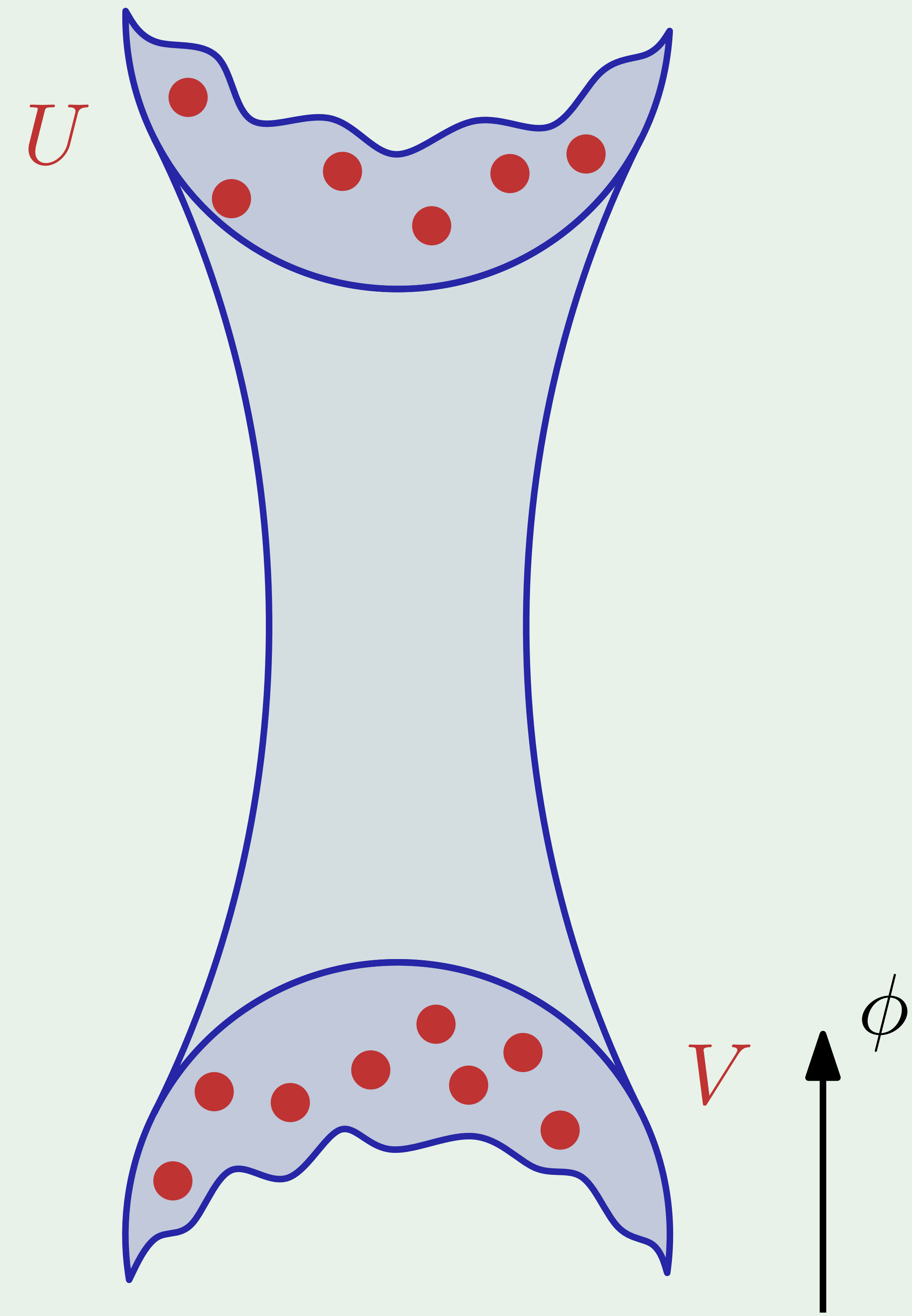
Claim: if the points would be sorted on x-coordinate, we could find it in linear time.



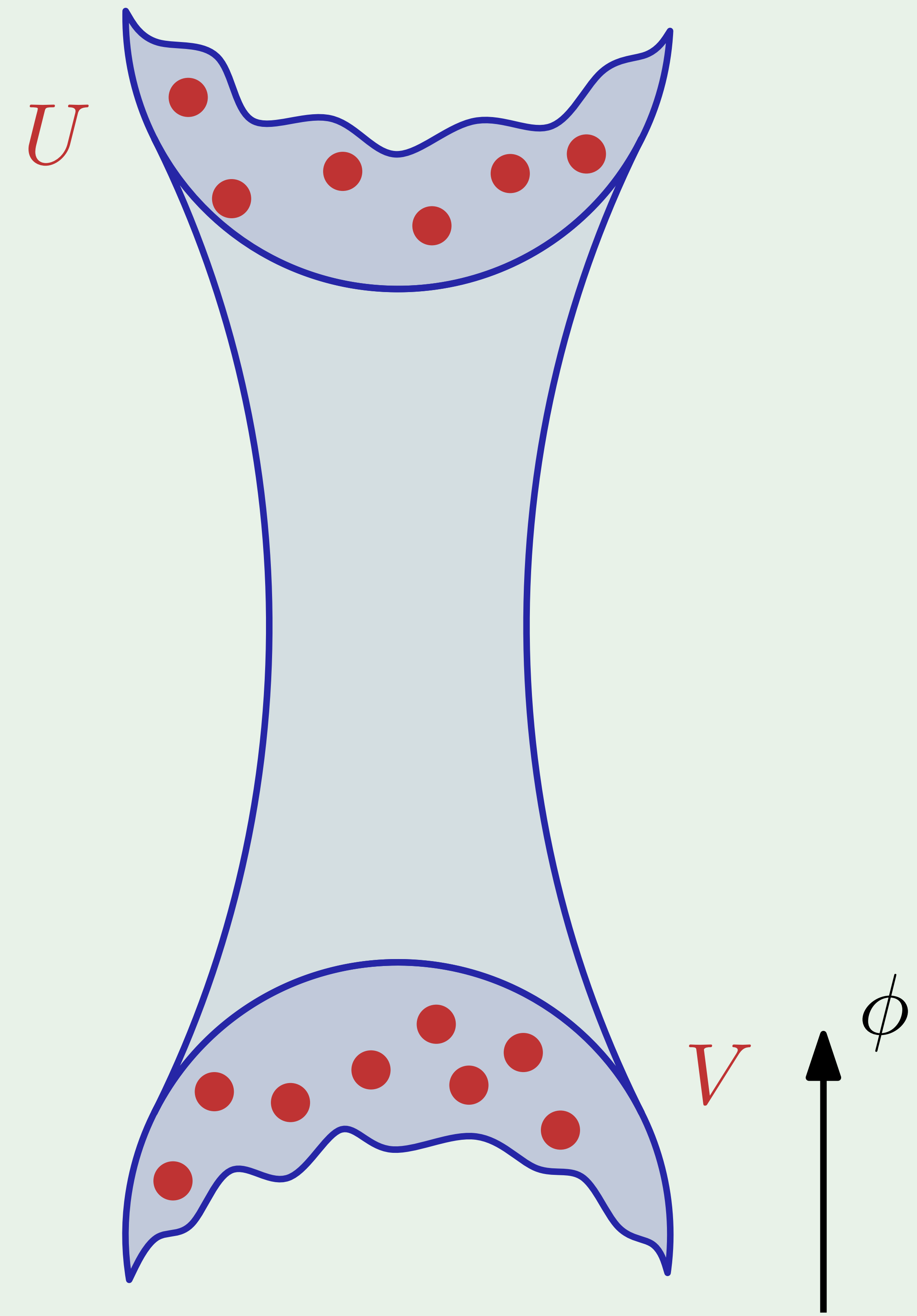
Now, for each pair $(U,V) \in \mathcal{P}'_\phi$, we need to find the closest pair of points.

Claim: if the points would be sorted on x-coordinate, we could find it in linear time.

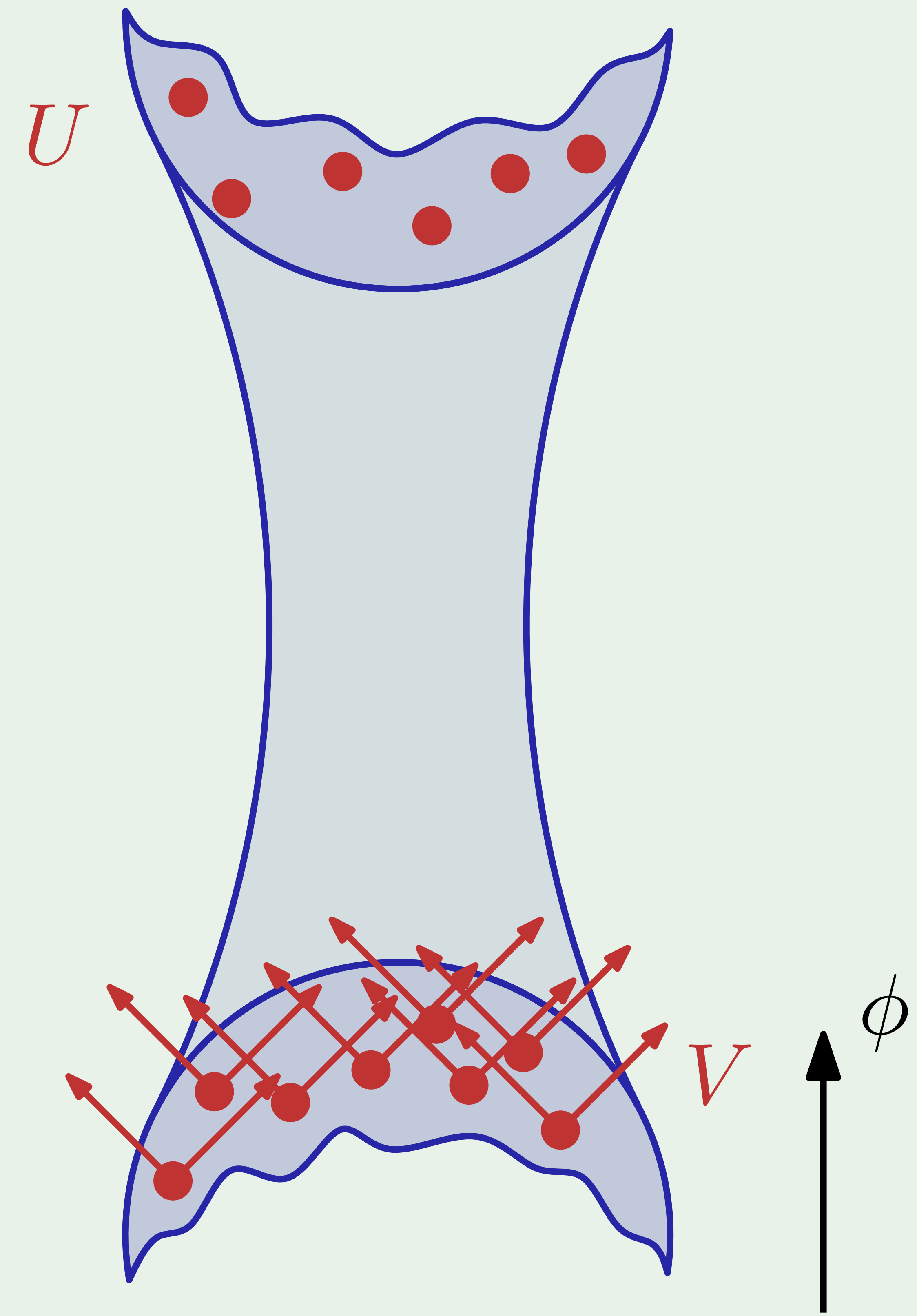
Problem: sorting takes $\Theta(n \log n)$ time.



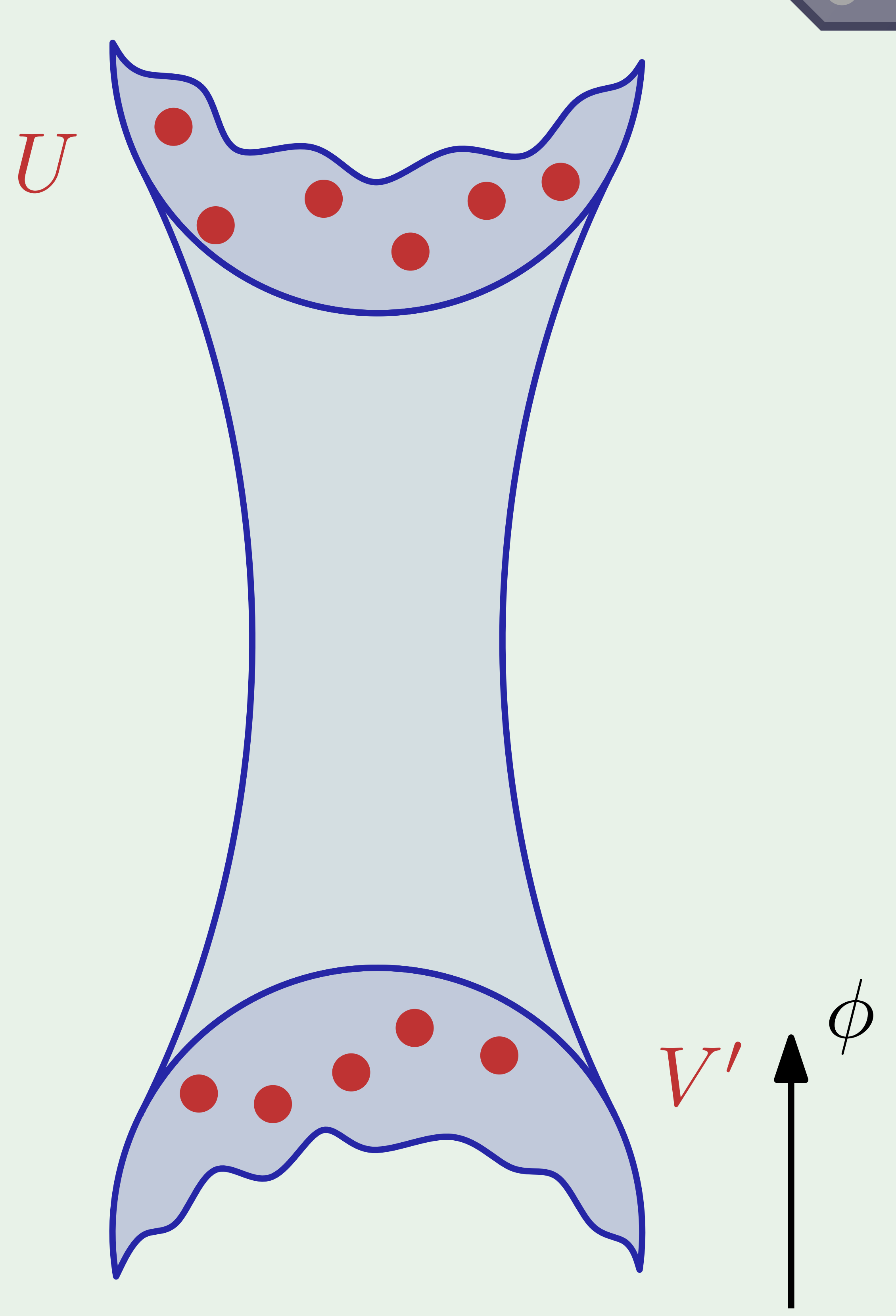
Claim: we only need to sort the points whose 'upward cone' is empty.



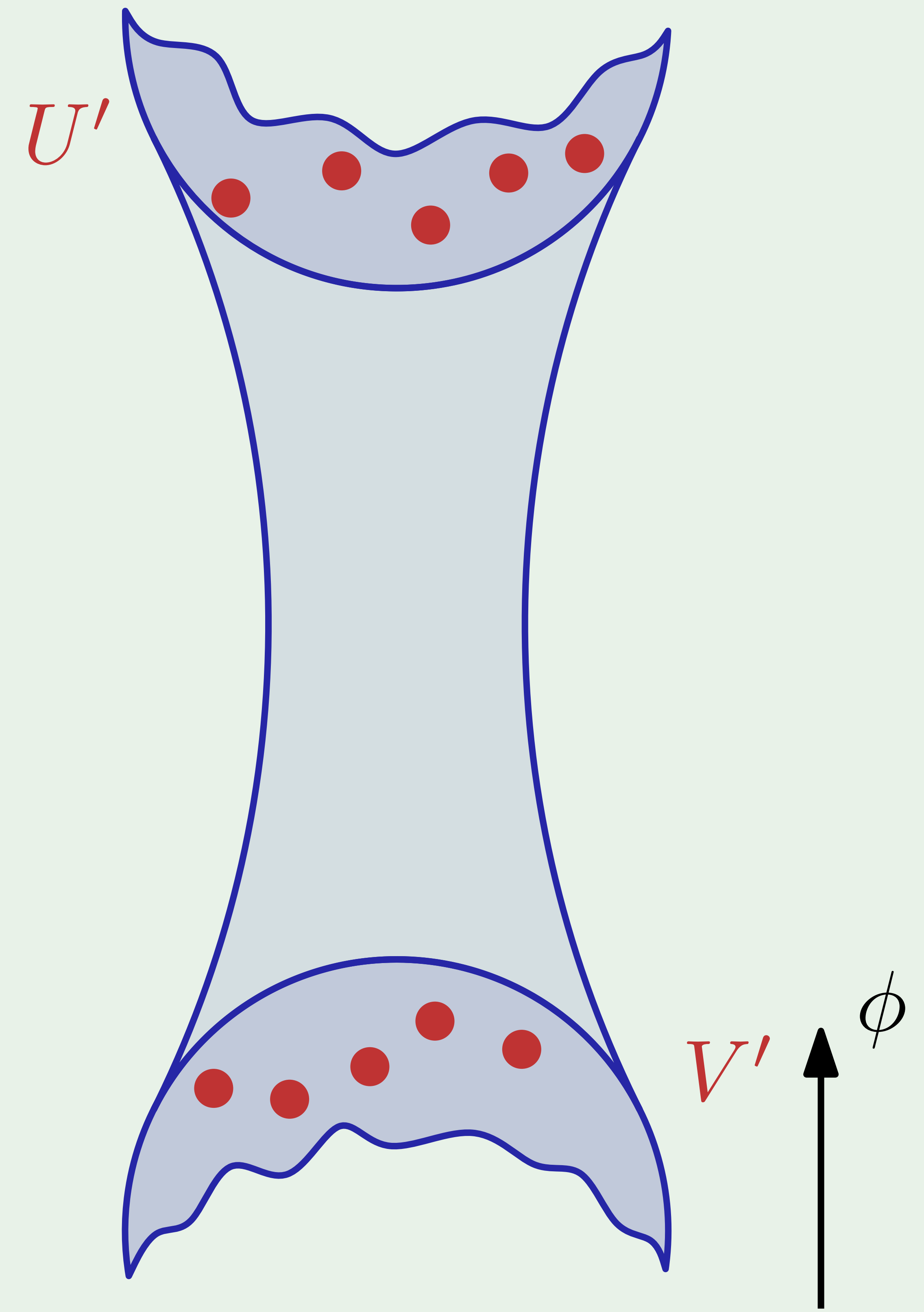
Claim: we only need to sort the points whose 'upward cone' is empty.



Claim: we only need to sort the points whose ‘upward cone’ is empty.

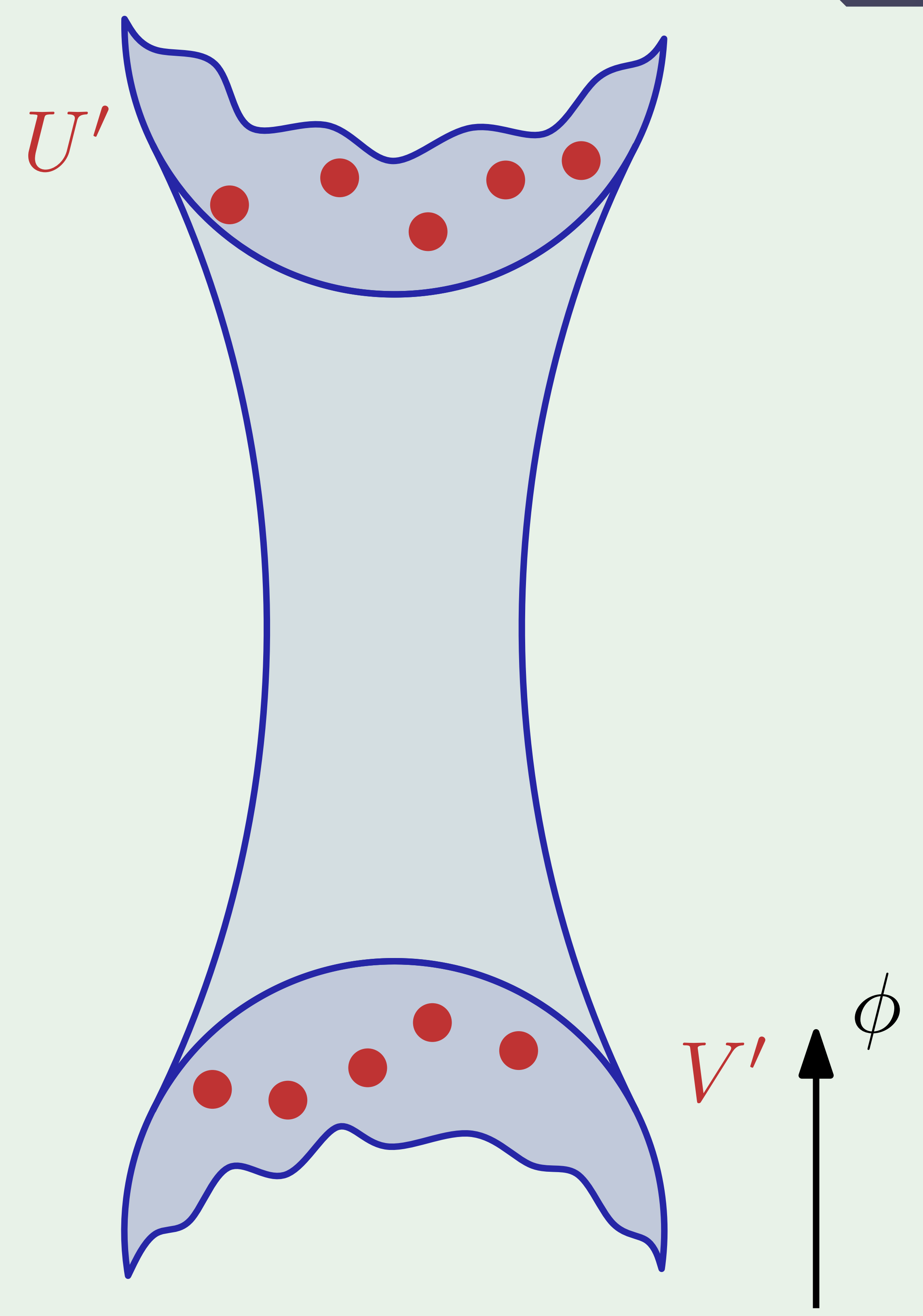


Claim: we only need to sort the points whose ‘upward cone’ is empty.



Claim: we only need to sort the points whose 'upward cone' is empty.

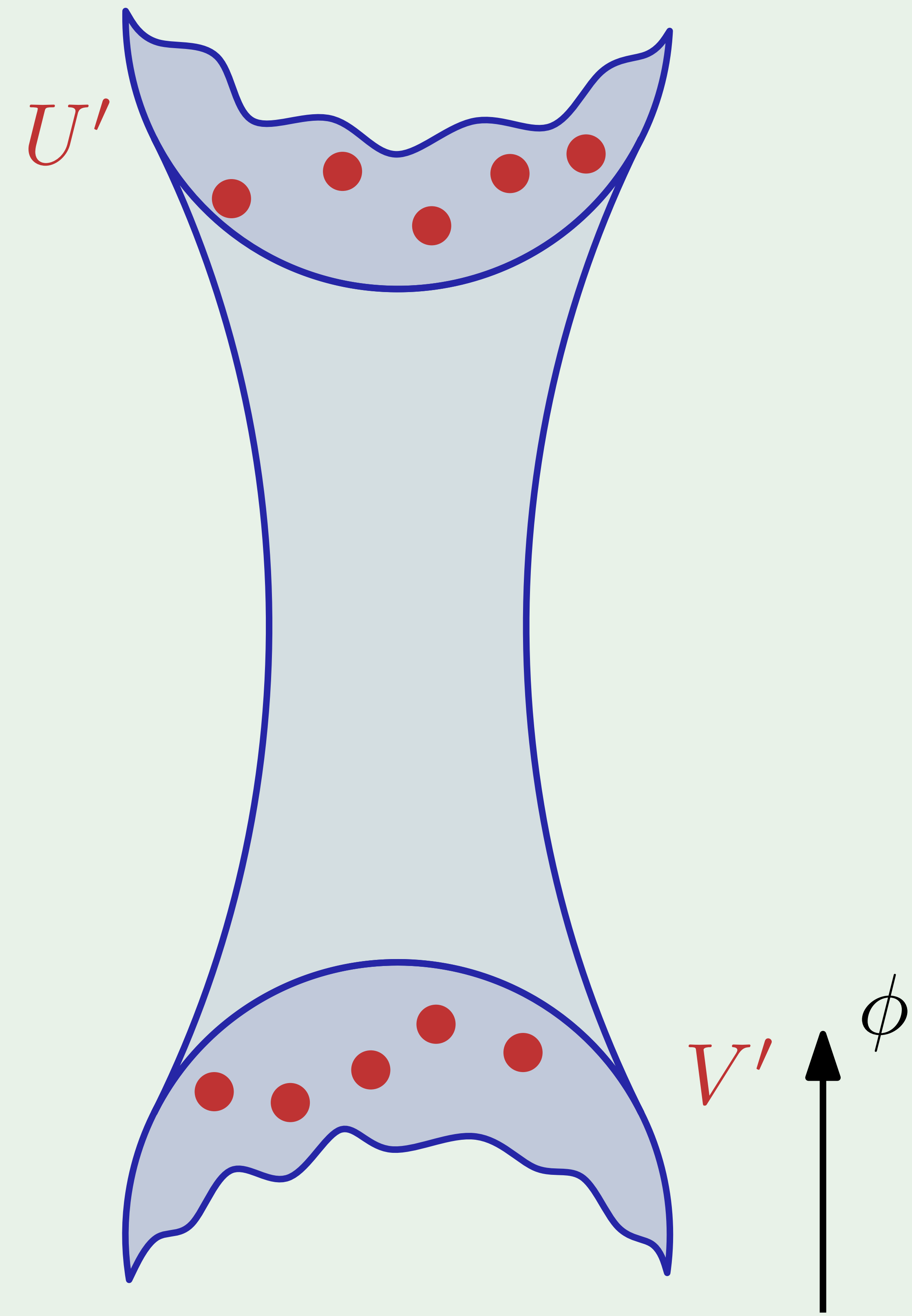
So, we only *locally* need the correct x-order.

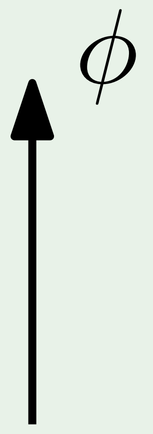


Claim: we only need to sort the points whose 'upward cone' is empty.

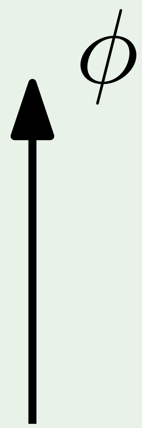
So, we only *locally* need the correct x-order.

But we have that information: we started with a WSPD of P!

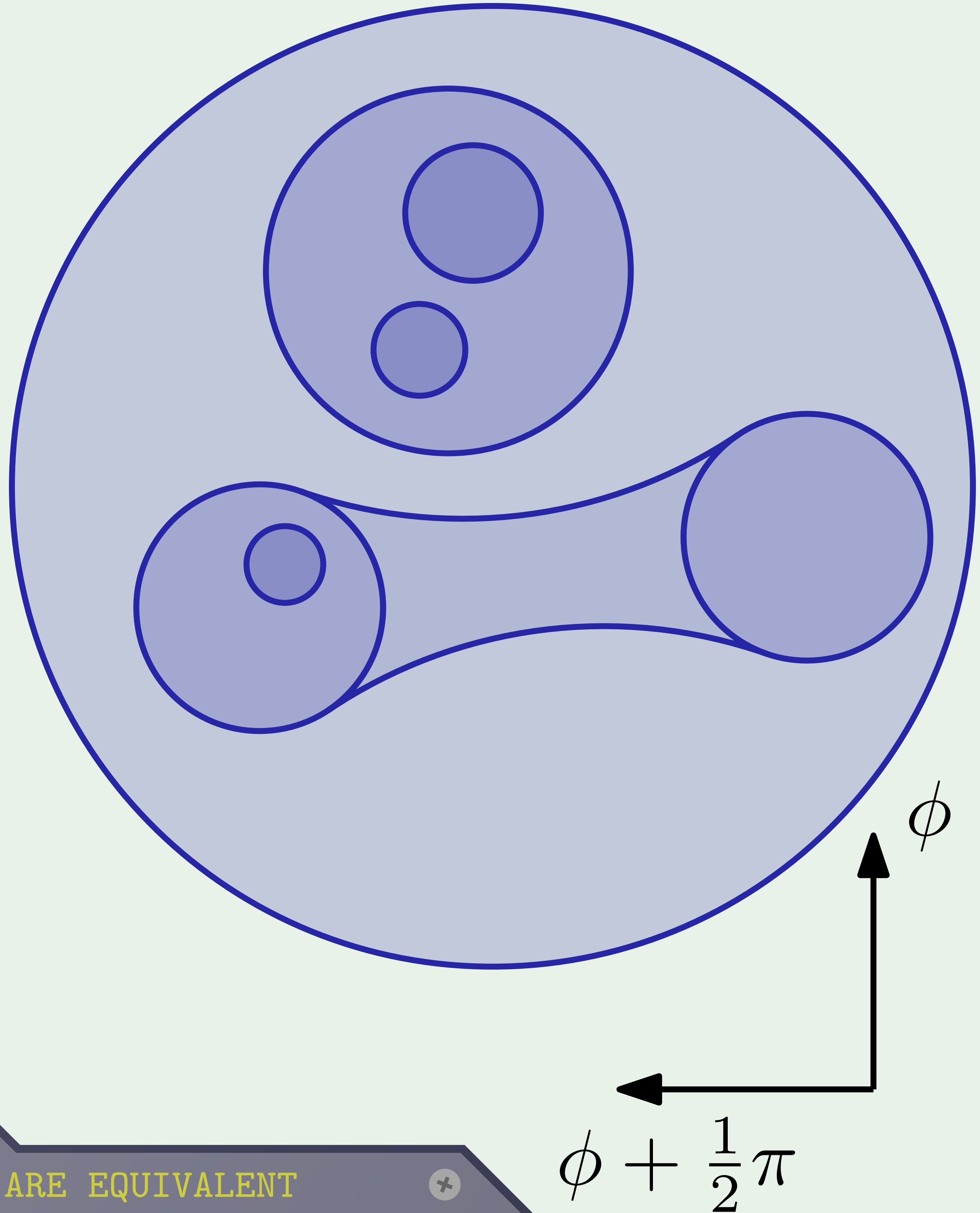



ϕ

Consider $\mathcal{P}_{\phi+\frac{1}{2}\pi}$, the pairs *perpendicular* to those in $\mathcal{P}_\phi$.

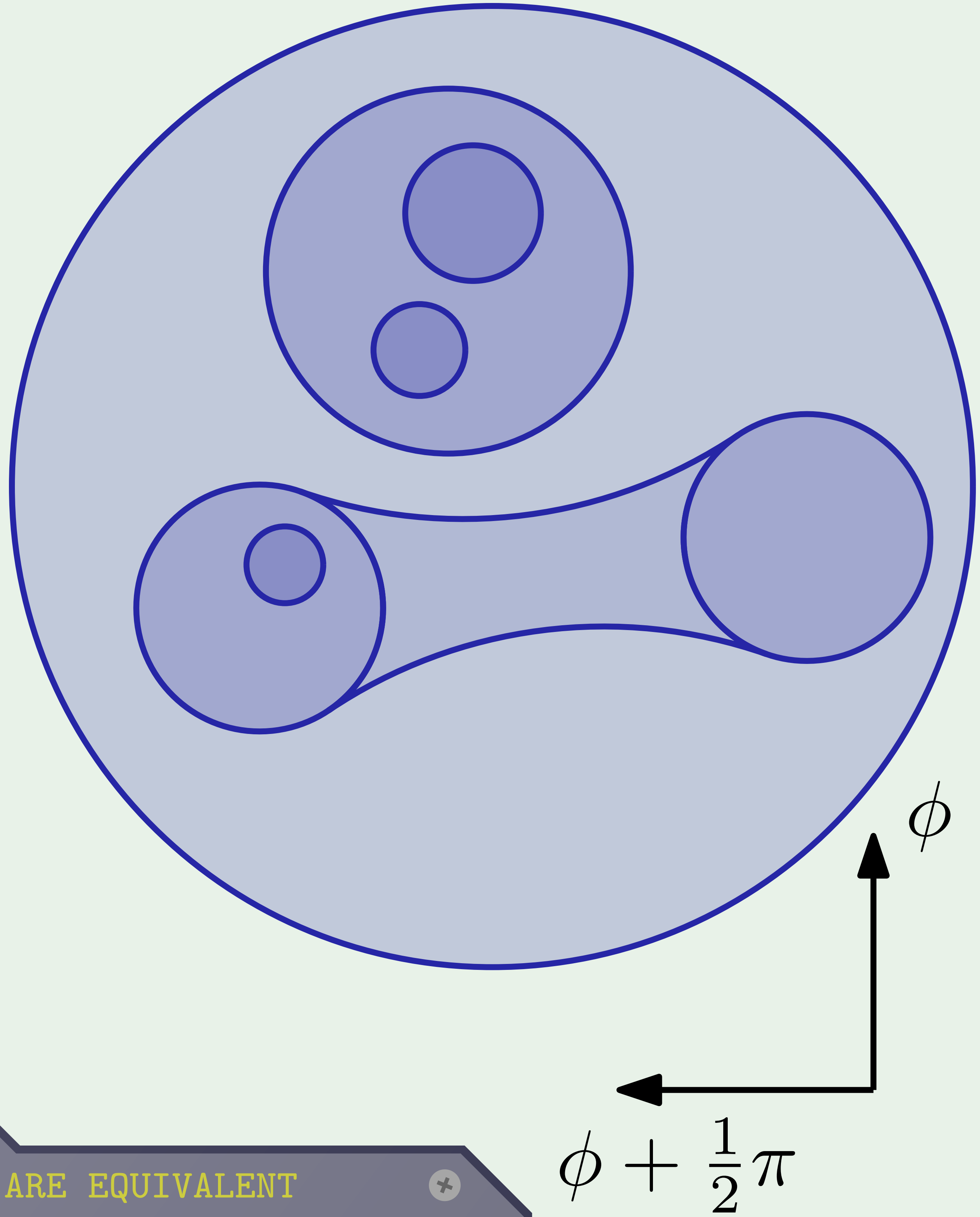


Consider $\mathcal{P}_{\phi+\frac{1}{2}\pi}$, the pairs *perpendicular* to those in $\mathcal{P}_{\phi}$.



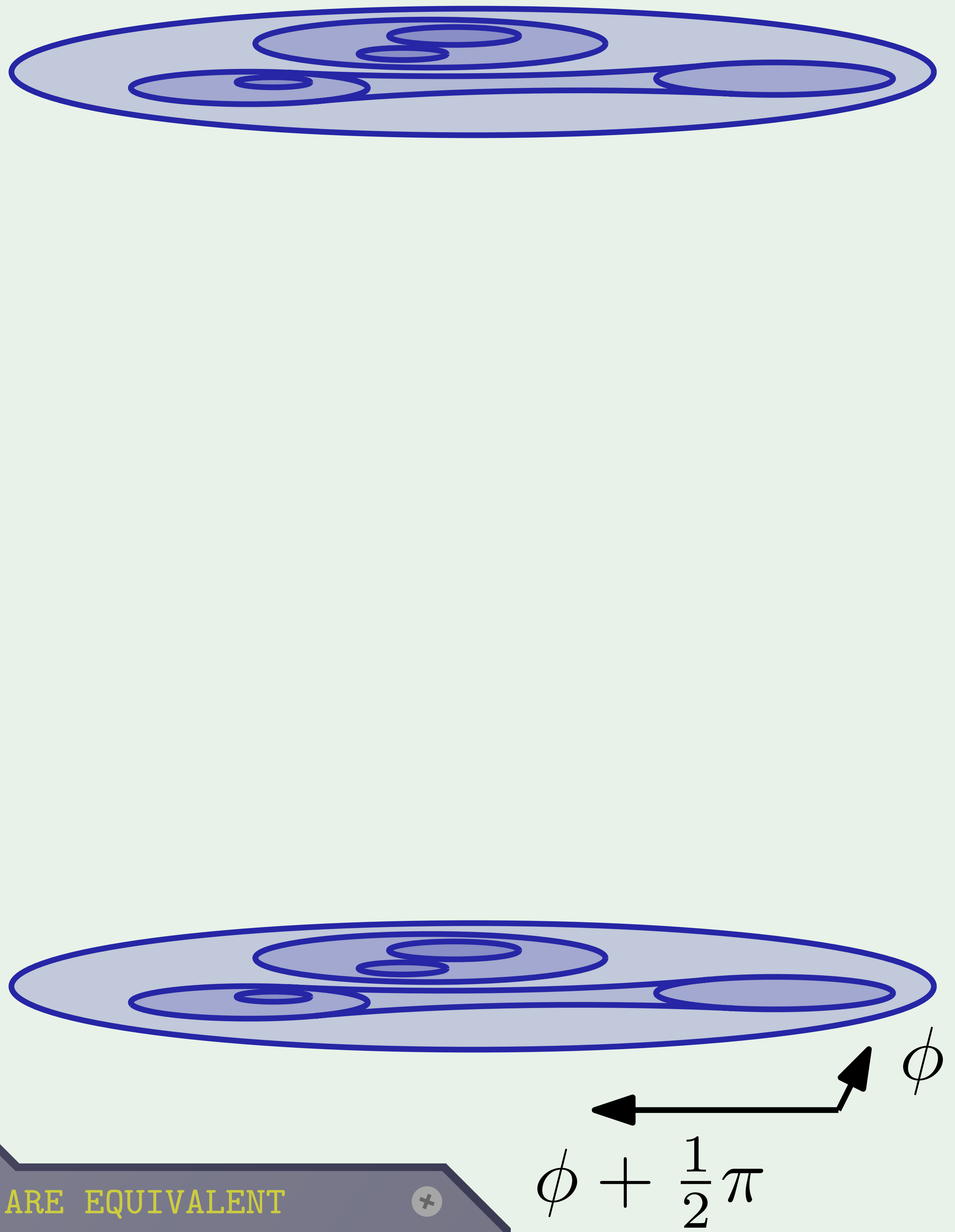
Consider $\mathcal{P}_{\phi+\frac{1}{2}\pi}$, the pairs *perpendicular* to those in $\mathcal{P}_{\phi}$.

Let V be a set with two copies of each set in $\mathcal{P}_{\phi+\frac{1}{2}\pi}$.



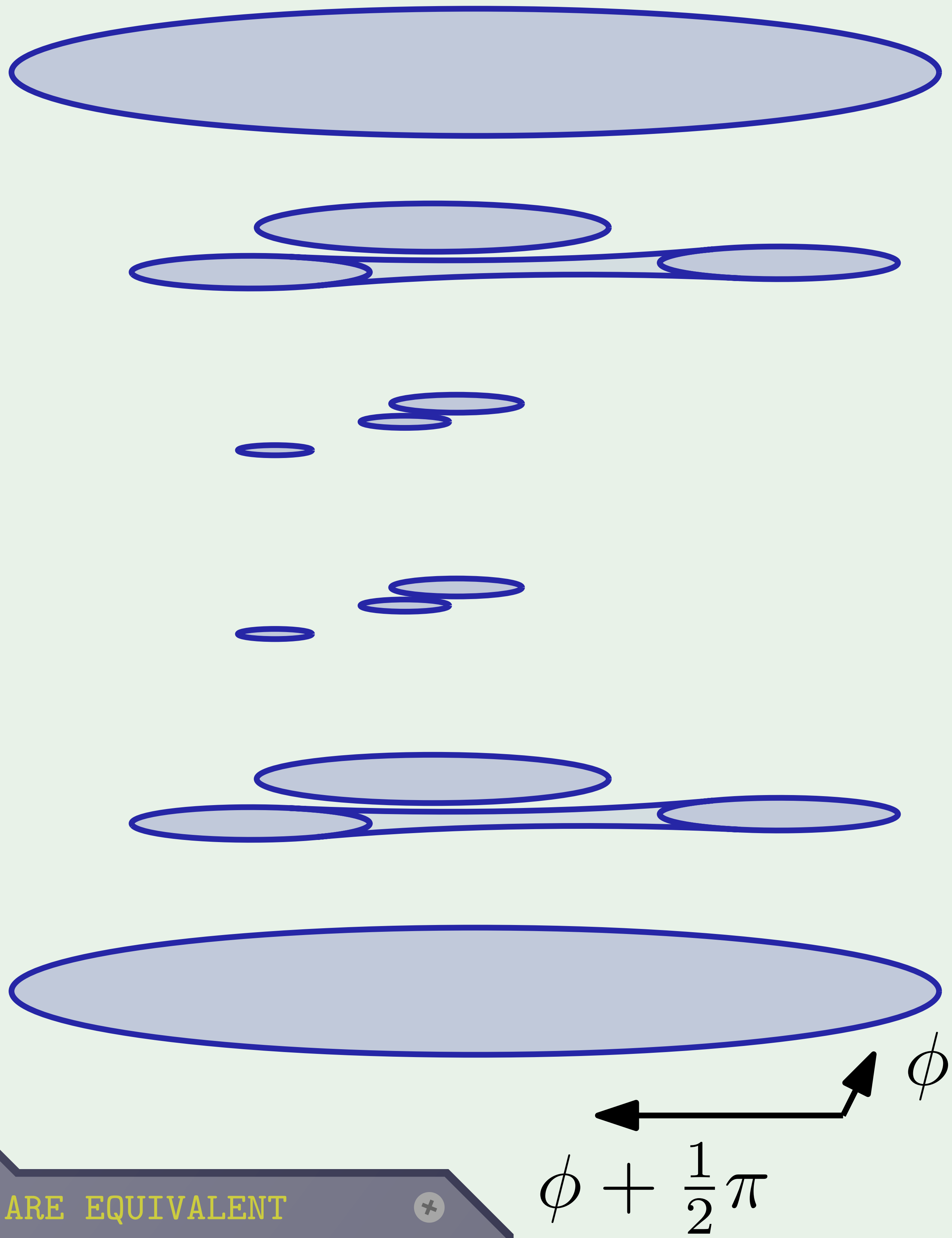
Consider $\mathcal{P}_{\phi+\frac{1}{2}\pi}$, the pairs *perpendicular* to those in $\mathcal{P}_\phi$.

Let V be a set with two copies of each set in $\mathcal{P}_{\phi+\frac{1}{2}\pi}$.



Consider $\mathcal{P}_{\phi+\frac{1}{2}\pi}$, the pairs *perpendicular* to those in $\mathcal{P}_\phi$.

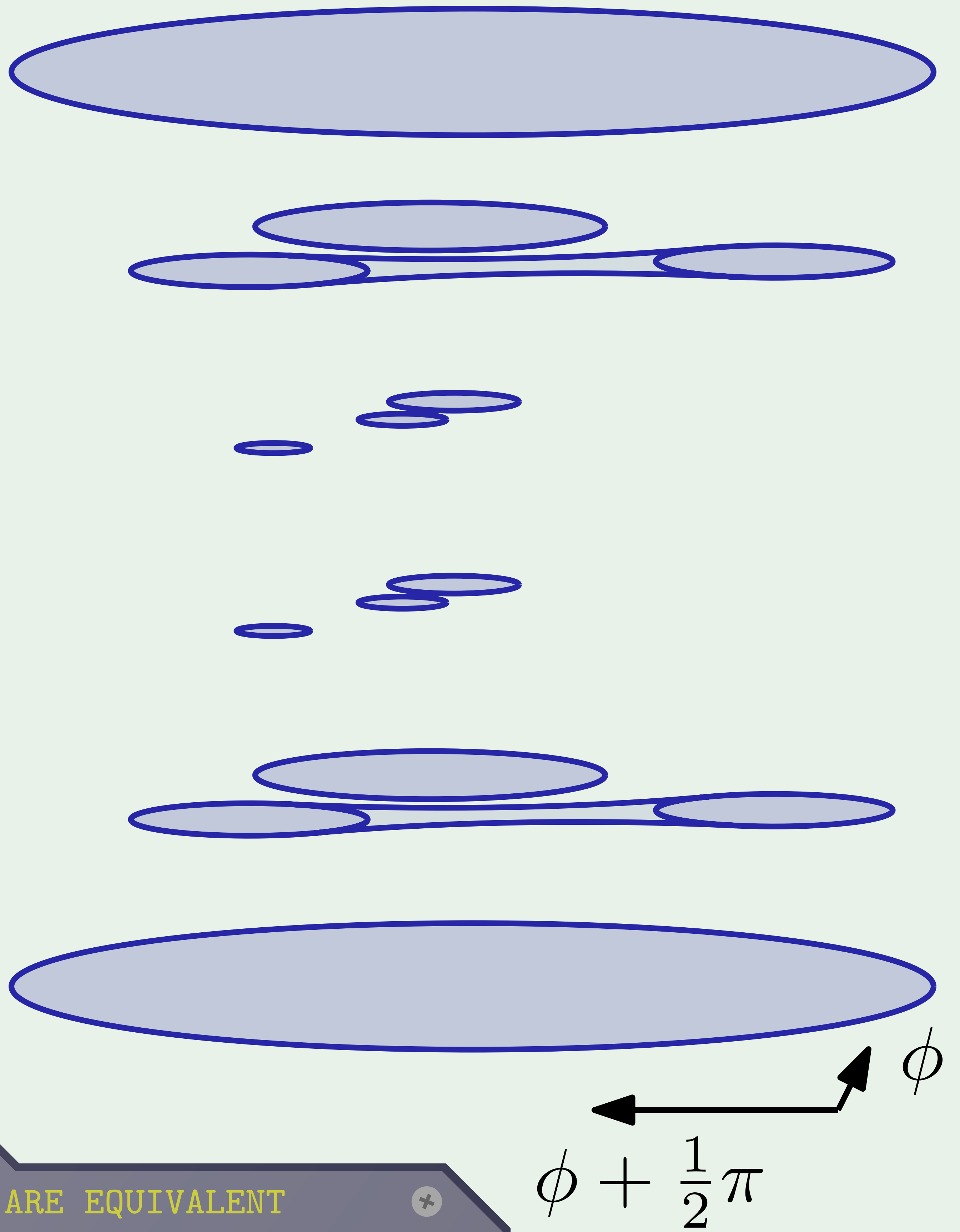
Let V be a set with two copies of each set in $\mathcal{P}_{\phi+\frac{1}{2}\pi}$.



Consider $\mathcal{P}_{\phi+\frac{1}{2}\pi}$, the pairs *perpendicular* to those in $\mathcal{P}_{\phi}$.

Let V be a set with two copies of each set in $\mathcal{P}_{\phi+\frac{1}{2}\pi}$.

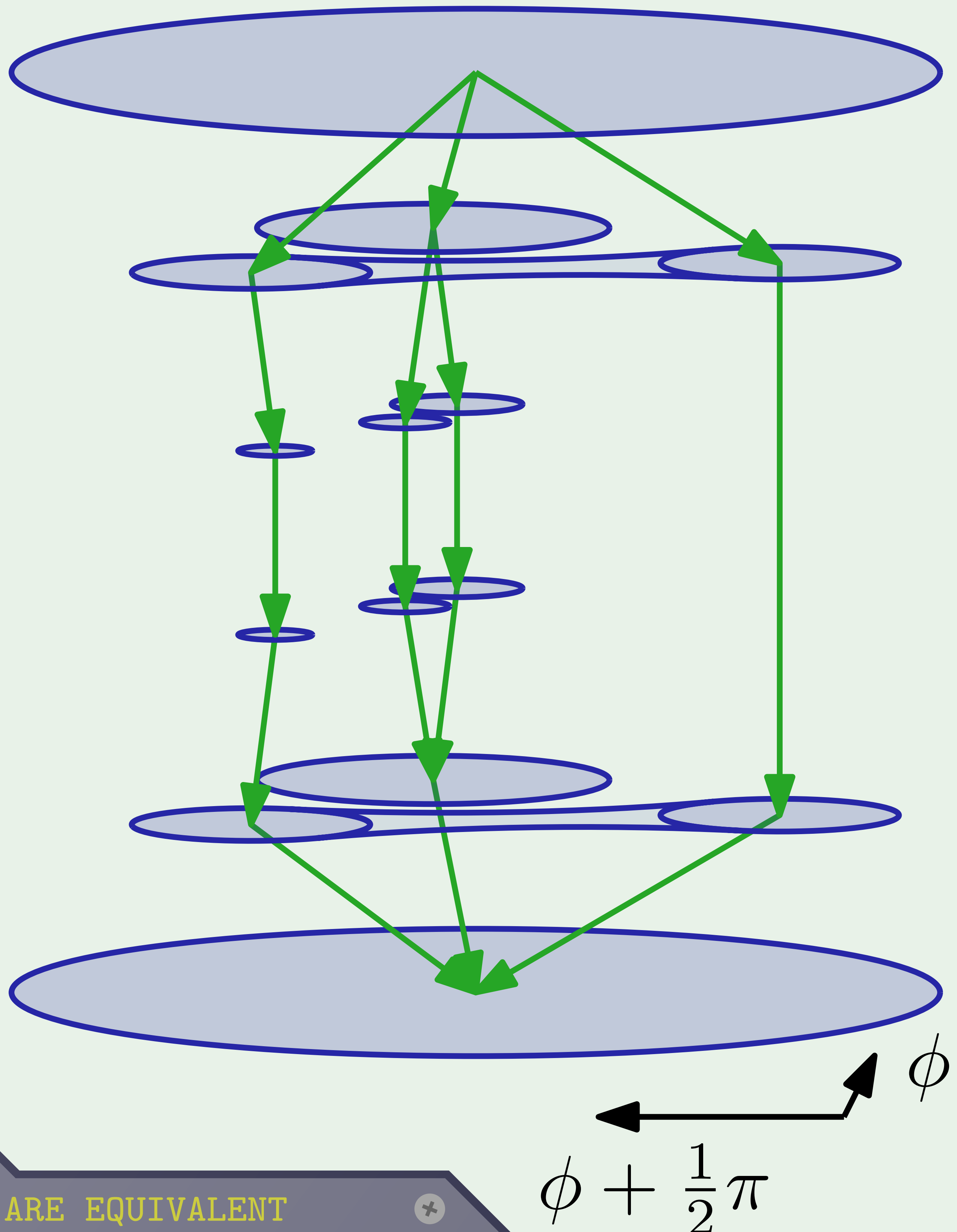
Let Γ be a graph on V that follows the hierarchical structure and WSPD pairs.



Consider $\mathcal{P}_{\phi+\frac{1}{2}\pi}$, the pairs *perpendicular* to those in $\mathcal{P}_\phi$.

Let V be a set with two copies of each set in $\mathcal{P}_{\phi+\frac{1}{2}\pi}$.

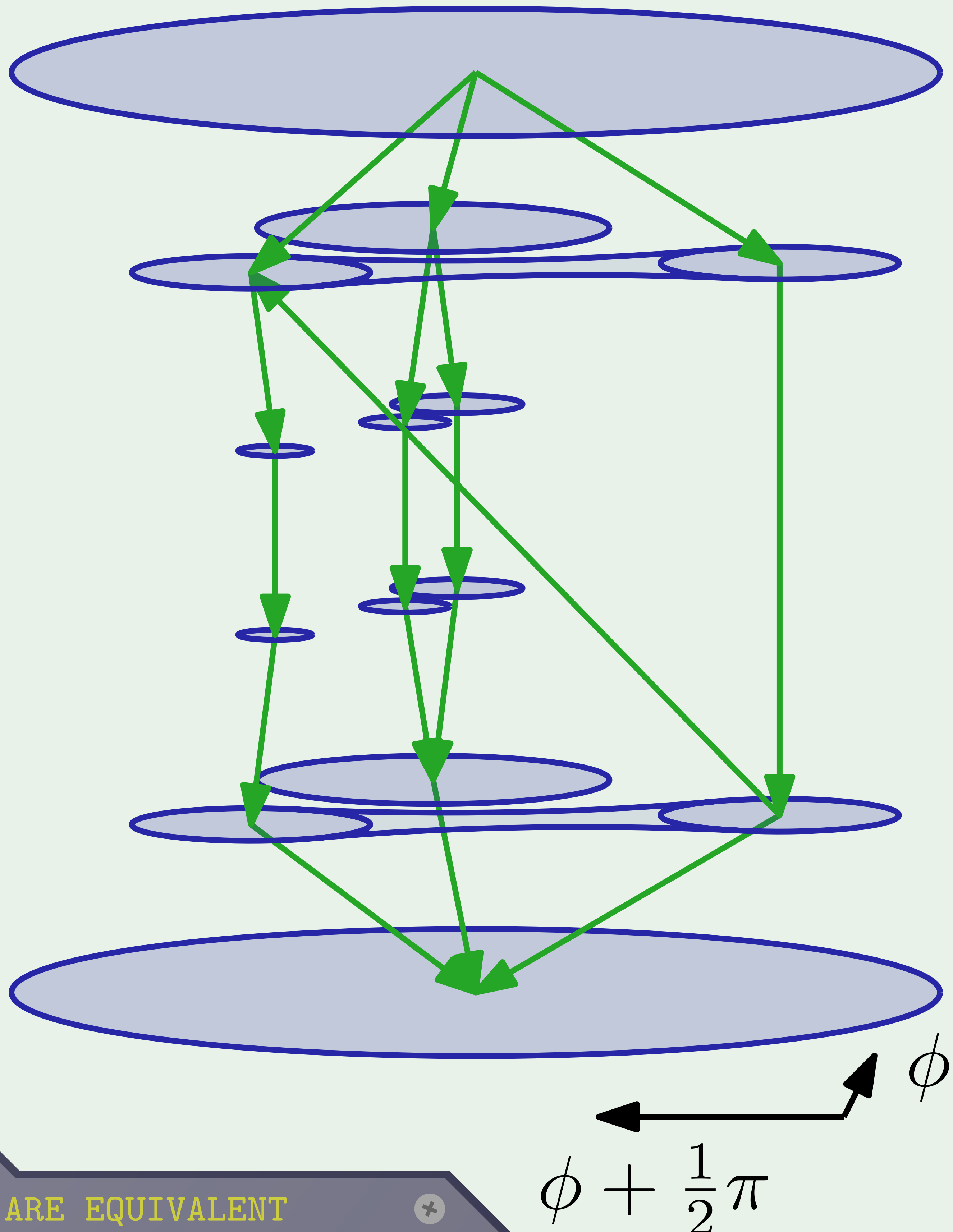
Let Γ be a graph on V that follows the hierarchical structure and WSPD pairs.



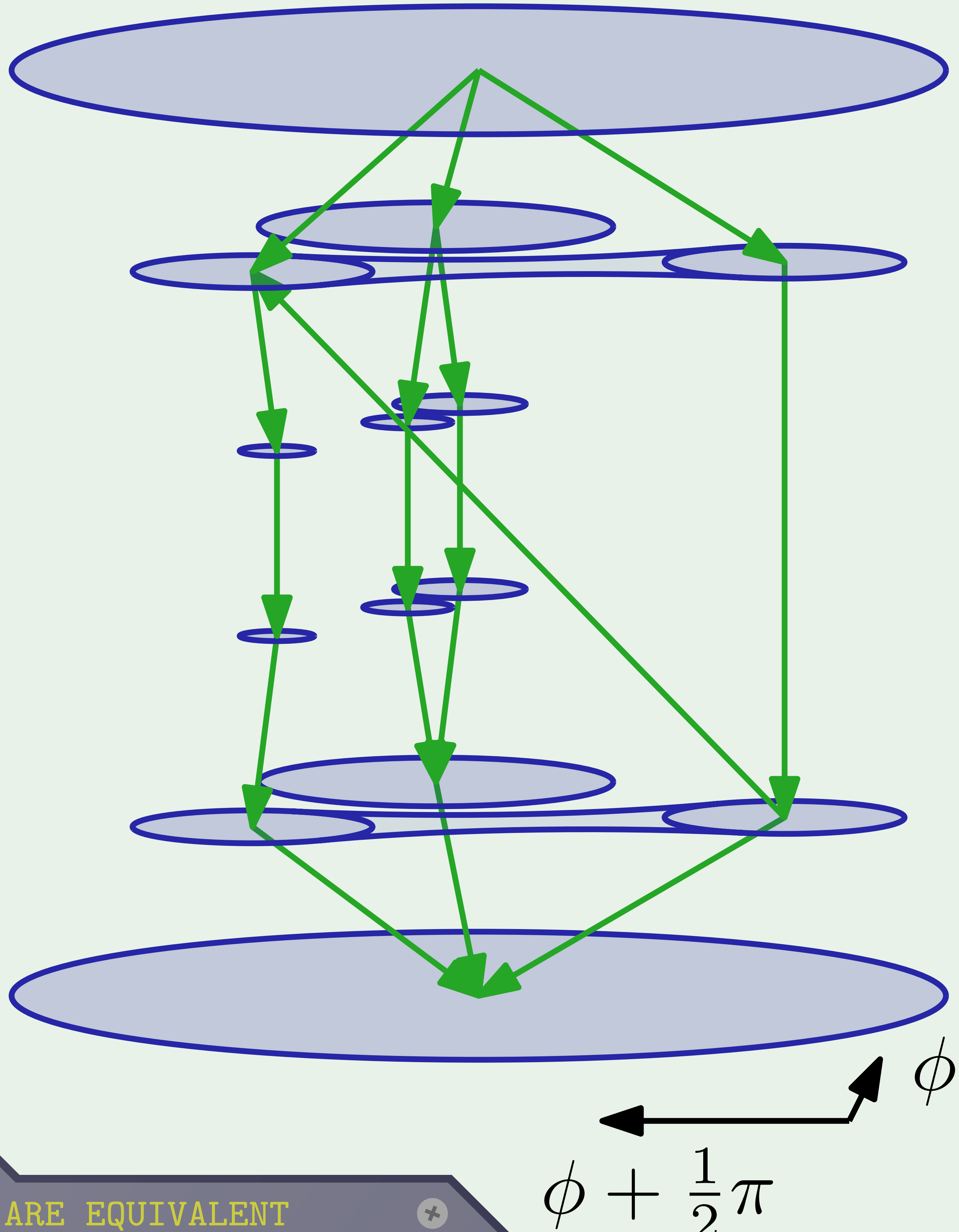
Consider $\mathcal{P}_{\phi+\frac{1}{2}\pi}$, the pairs *perpendicular* to those in $\mathcal{P}_{\phi}$.

Let V be a set with two copies of each set in $\mathcal{P}_{\phi+\frac{1}{2}\pi}$.

Let Γ be a graph on V that follows the hierarchical structure and WSPD pairs.

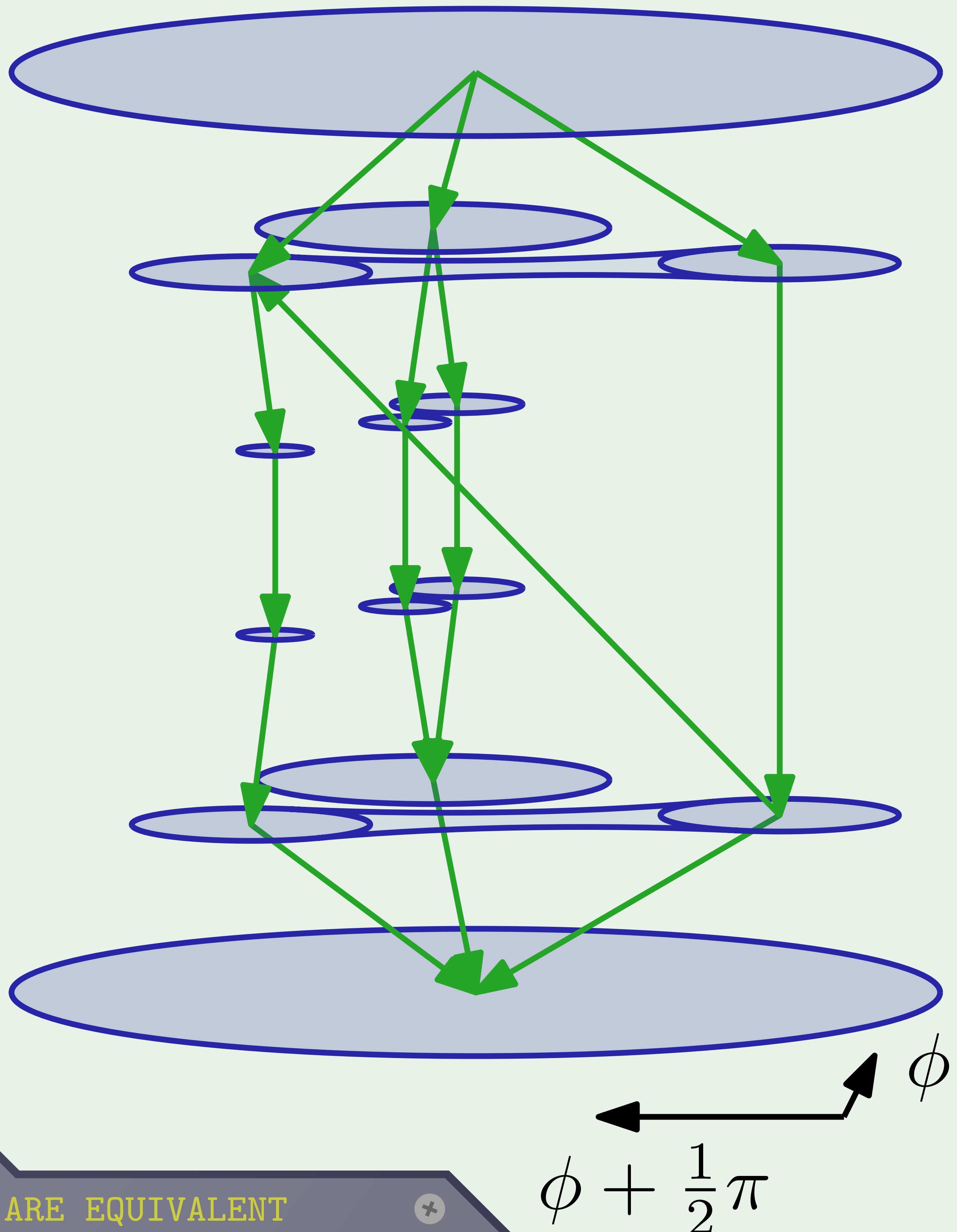


Claim: Γ is acyclic.



Claim: Γ is acyclic.

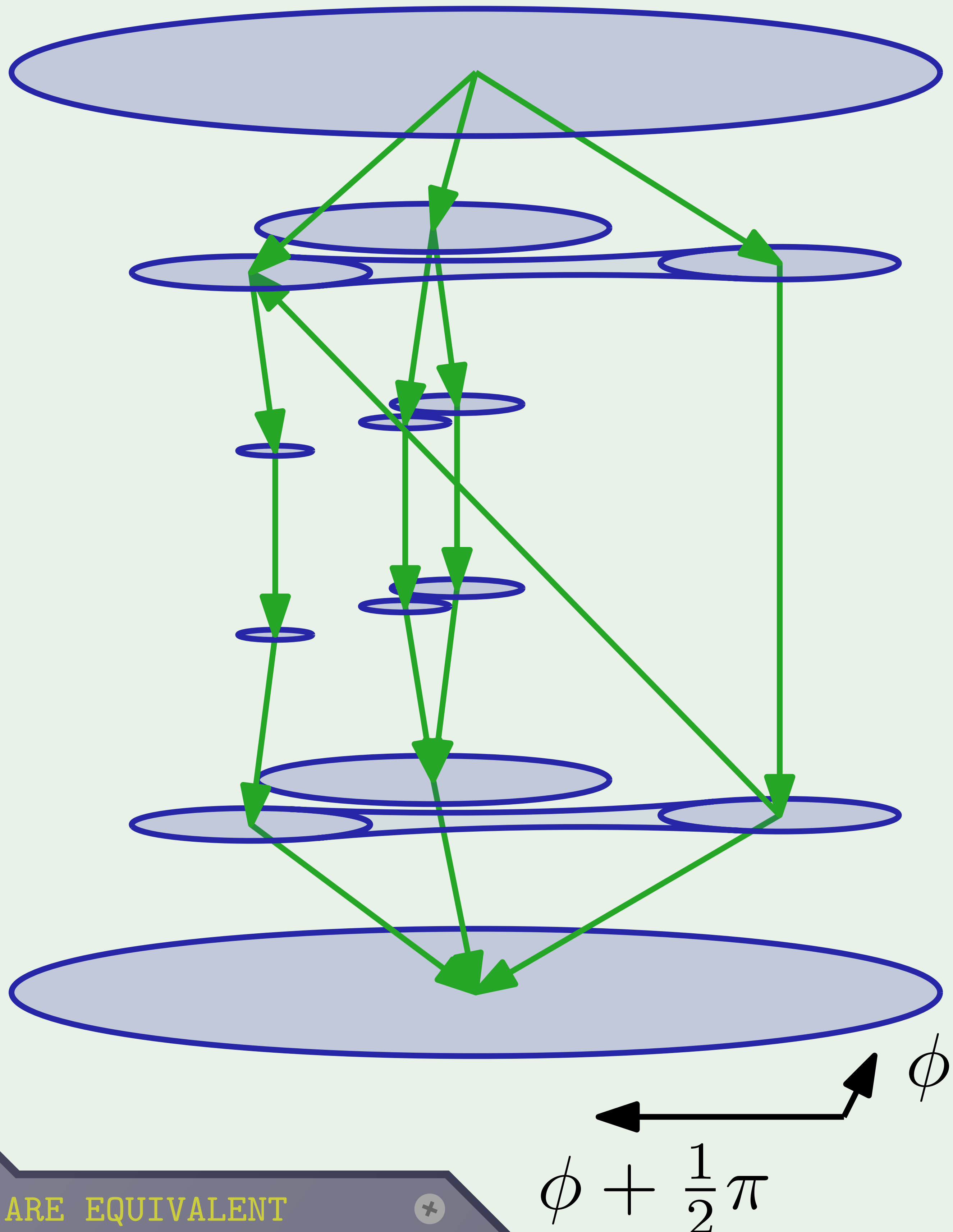
Claim: If a point p is roughly in direction $\phi + \frac{1}{2}\pi$ as seen from a point q, then q comes before p in Γ.



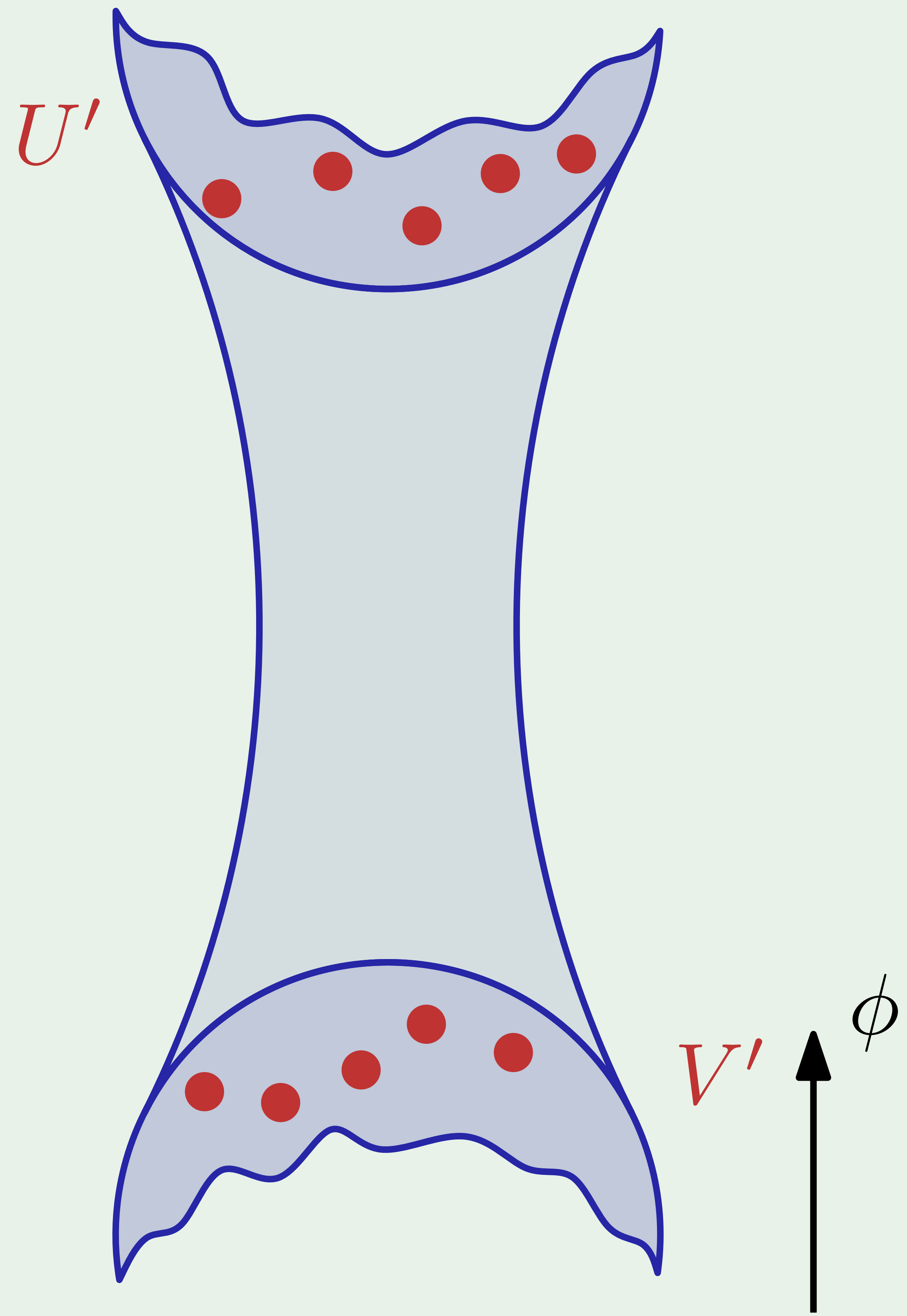
Claim: Γ is acyclic.

Claim: If a point p is roughly in direction $\phi + \frac{1}{2}\pi$ as seen from a point q, then q comes before p in Γ.

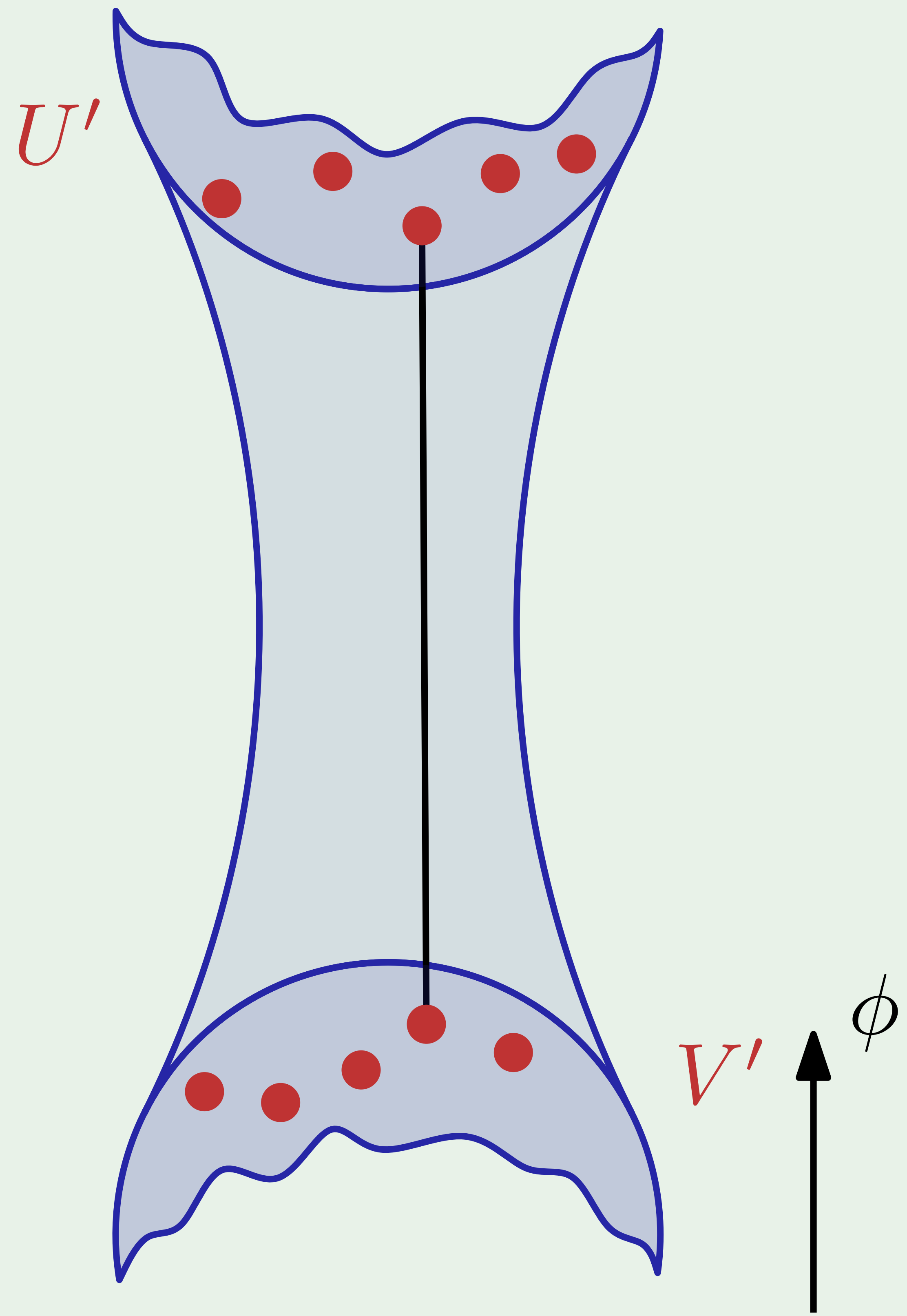
Claim: We can sort P conforming Γ in linear time.



Now, we can find the closest edge e between U' and V' in $O(|U| + |V|)$ time.

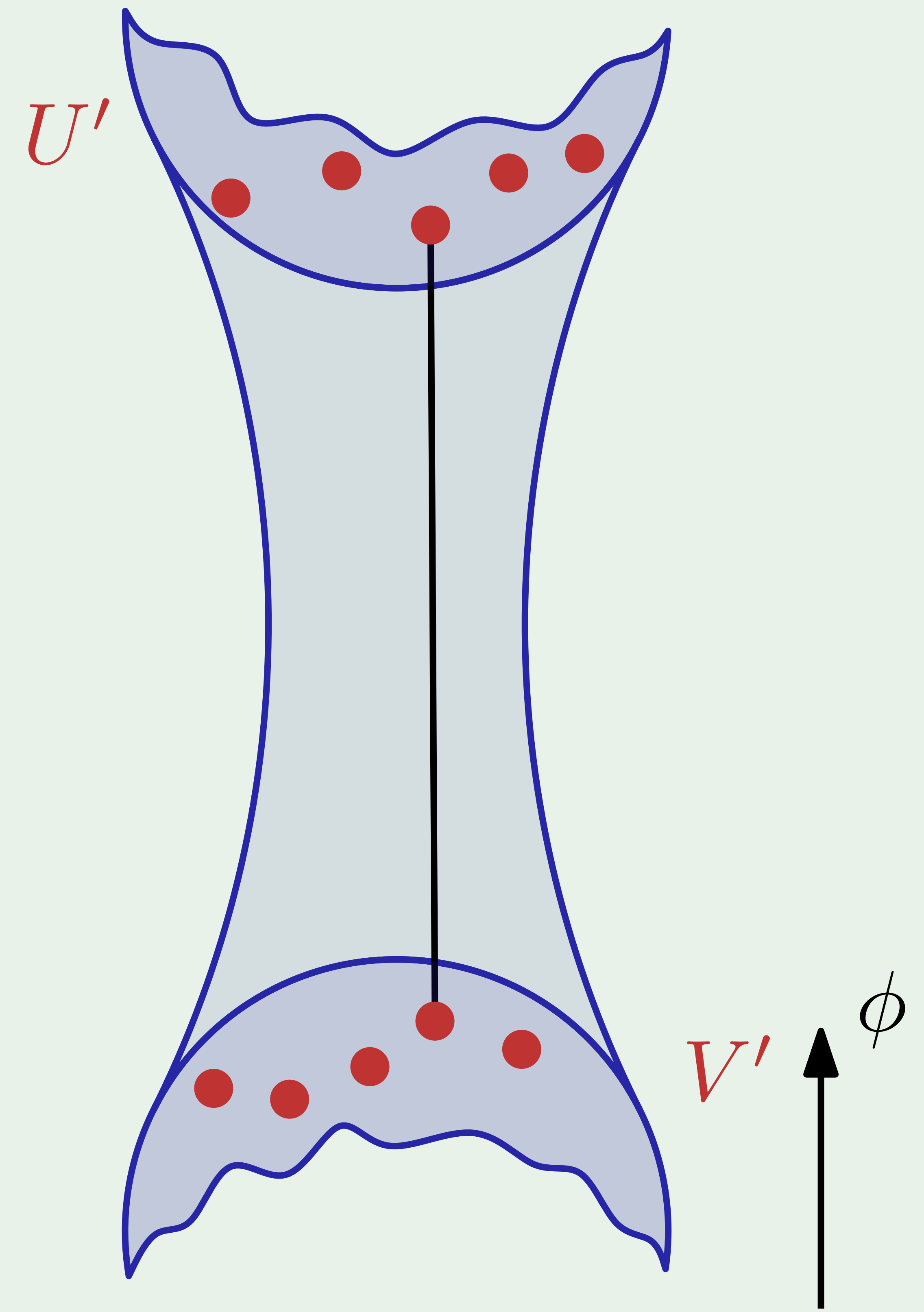


Now, we can find the closest edge e between U' and V' in $O(|U| + |V|)$ time.



Now, we can find the closest edge e between U' and V' in $O(|U| + |V|)$ time.

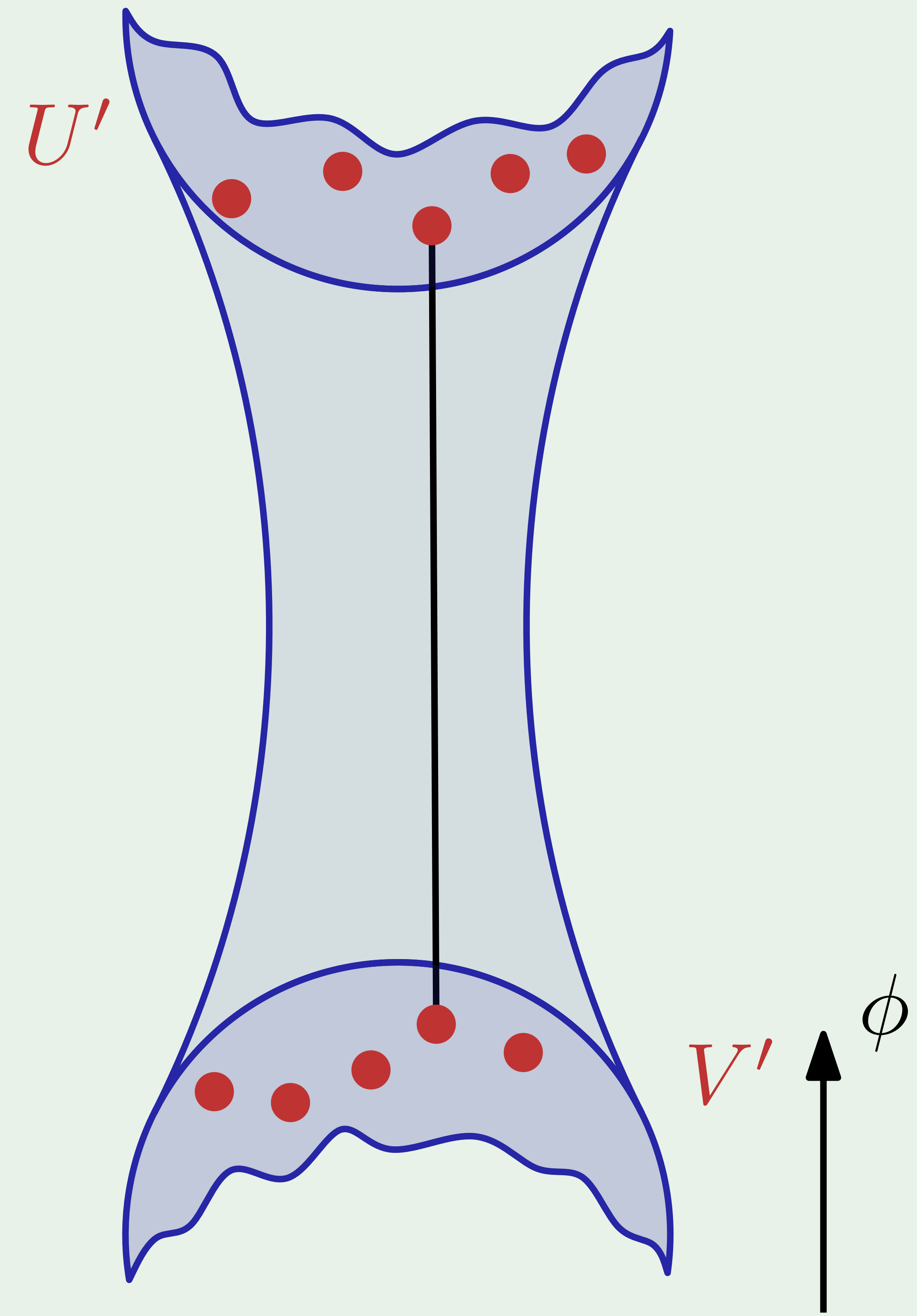
So, we can find G_ϕ in $O(n)$ time.



Now, we can find the closest edge e between U' and V' in $O(|U| + |V|)$ time.

So, we can find G_ϕ in $O(n)$ time.

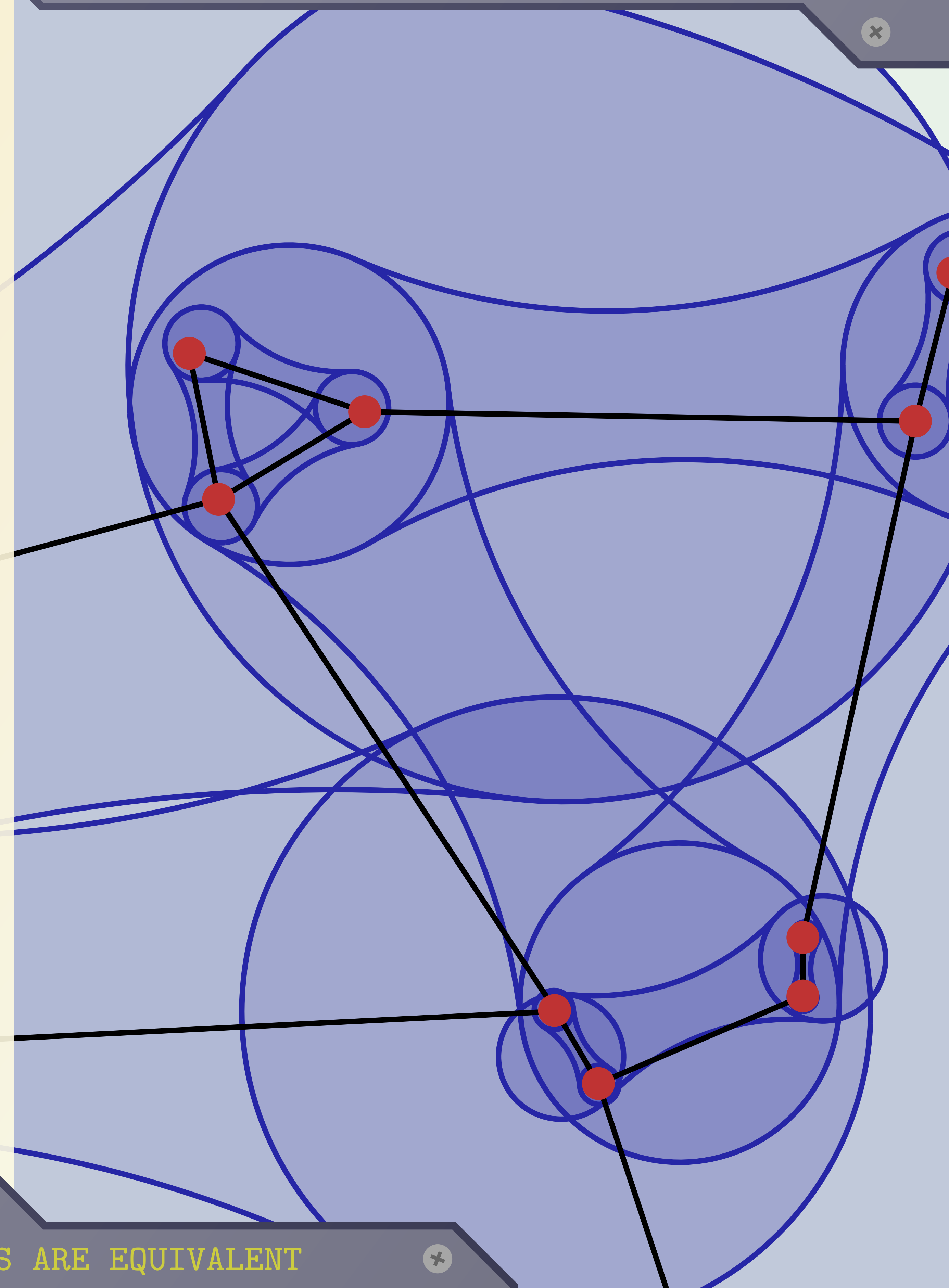
Yay! We found G, a linear size supergraph of the MST of P, in linear time!



Now, we can find the closest edge e between U' and V' in $O(|U| + |V|)$ time.

So, we can find G_ϕ in $O(n)$ time.

Yay! We found G, a linear size supergraph of the MST of P, in linear time!



Problem: computing a MST may take up to $O(n\alpha(n))$ time.

Problem: computing a MST may take up to $O(n\alpha(n))$ time.

For *planar* graphs, this is only $O(n)$.

Problem: computing a MST may take up to $O(n\alpha(n))$ time.

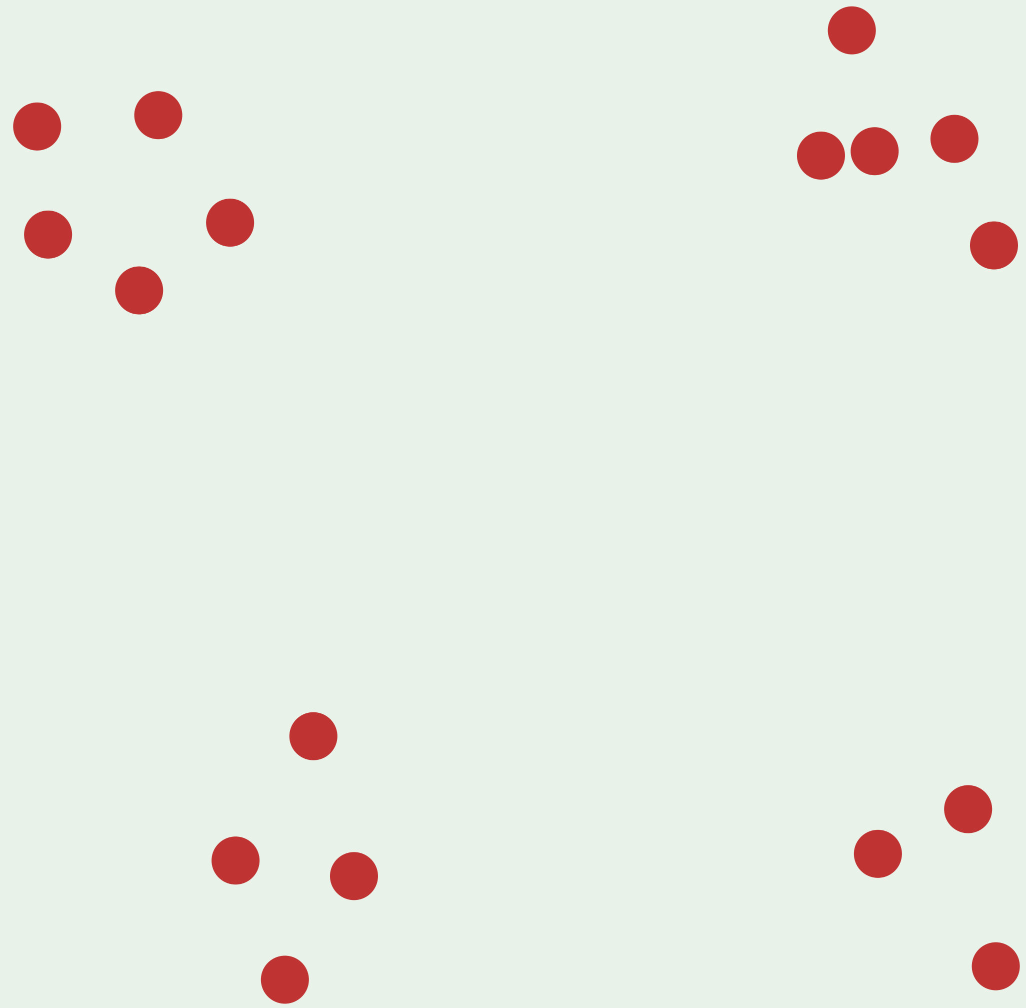
For *planar* graphs, this is only $O(n)$.

Unfortunately, G need not be planar.

Problem: computing a MST may take up to $O(n\alpha(n))$ time.

For *planar* graphs, this is only $O(n)$.

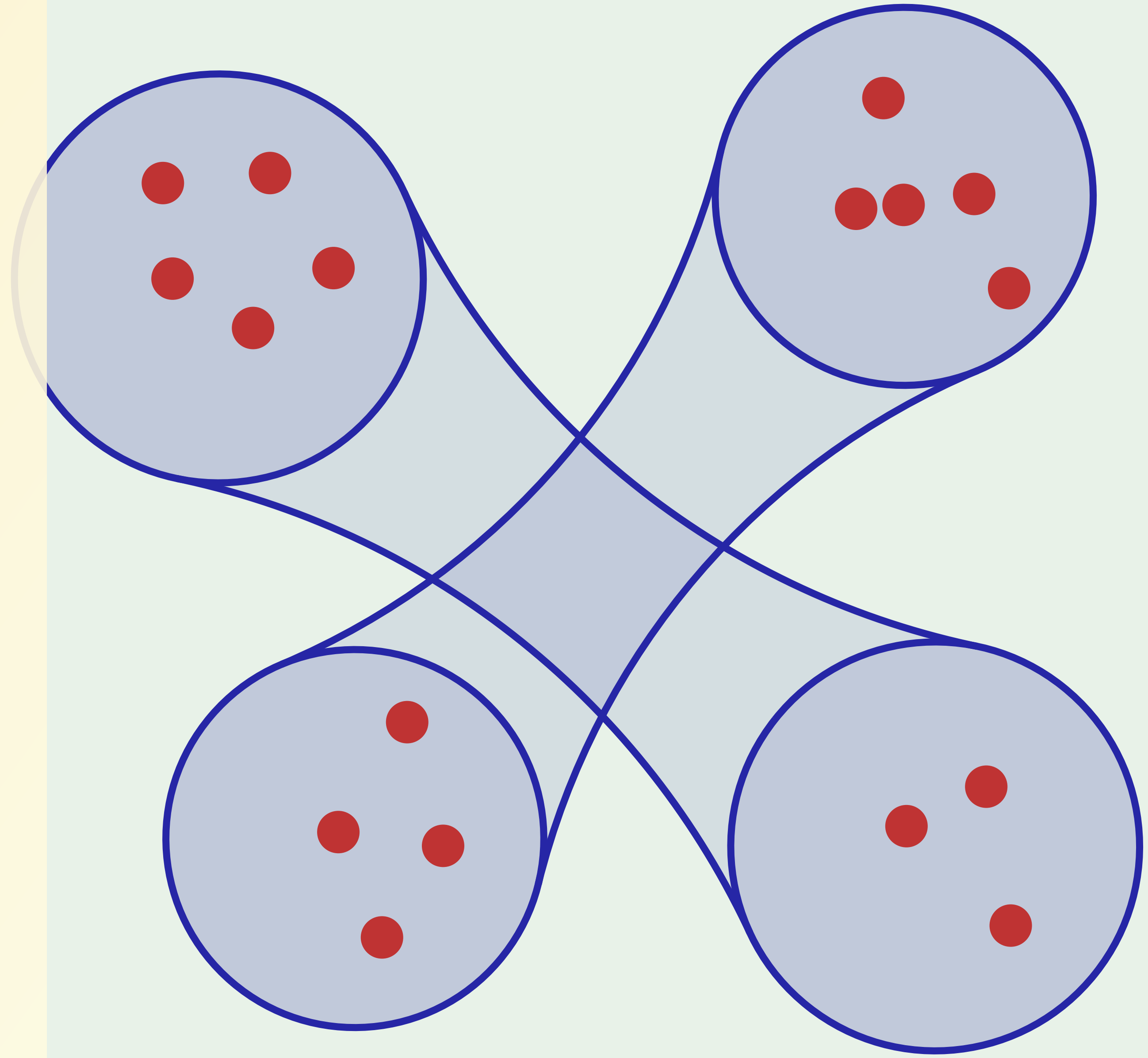
Unfortunately, G need not be planar.



Problem: computing a MST may take up to $O(n\alpha(n))$ time.

For *planar* graphs, this is only $O(n)$.

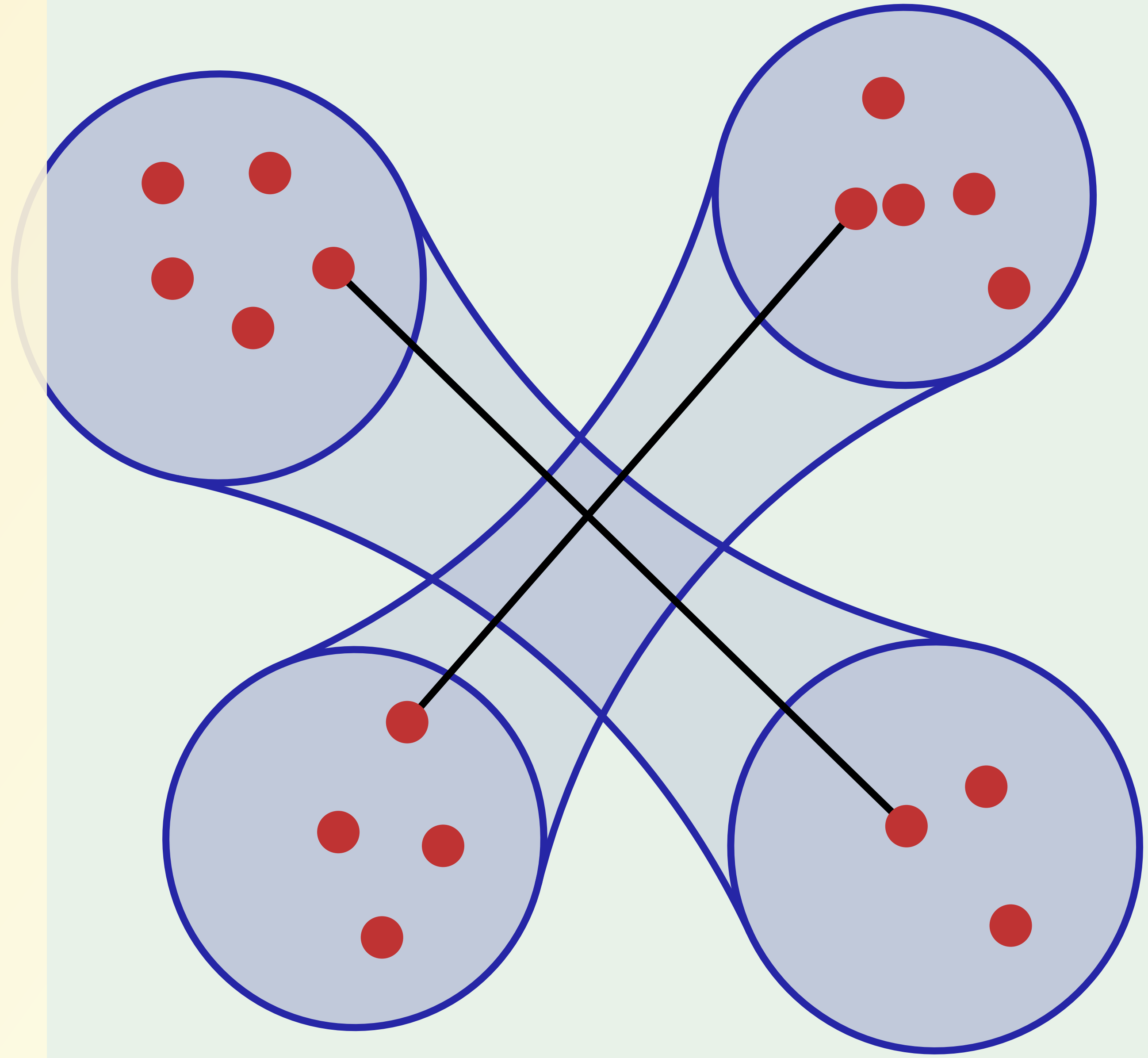
Unfortunately, G need not be planar.



Problem: computing a MST may take up to $O(n\alpha(n))$ time.

For *planar* graphs, this is only $O(n)$.

Unfortunately, G need not be planar.

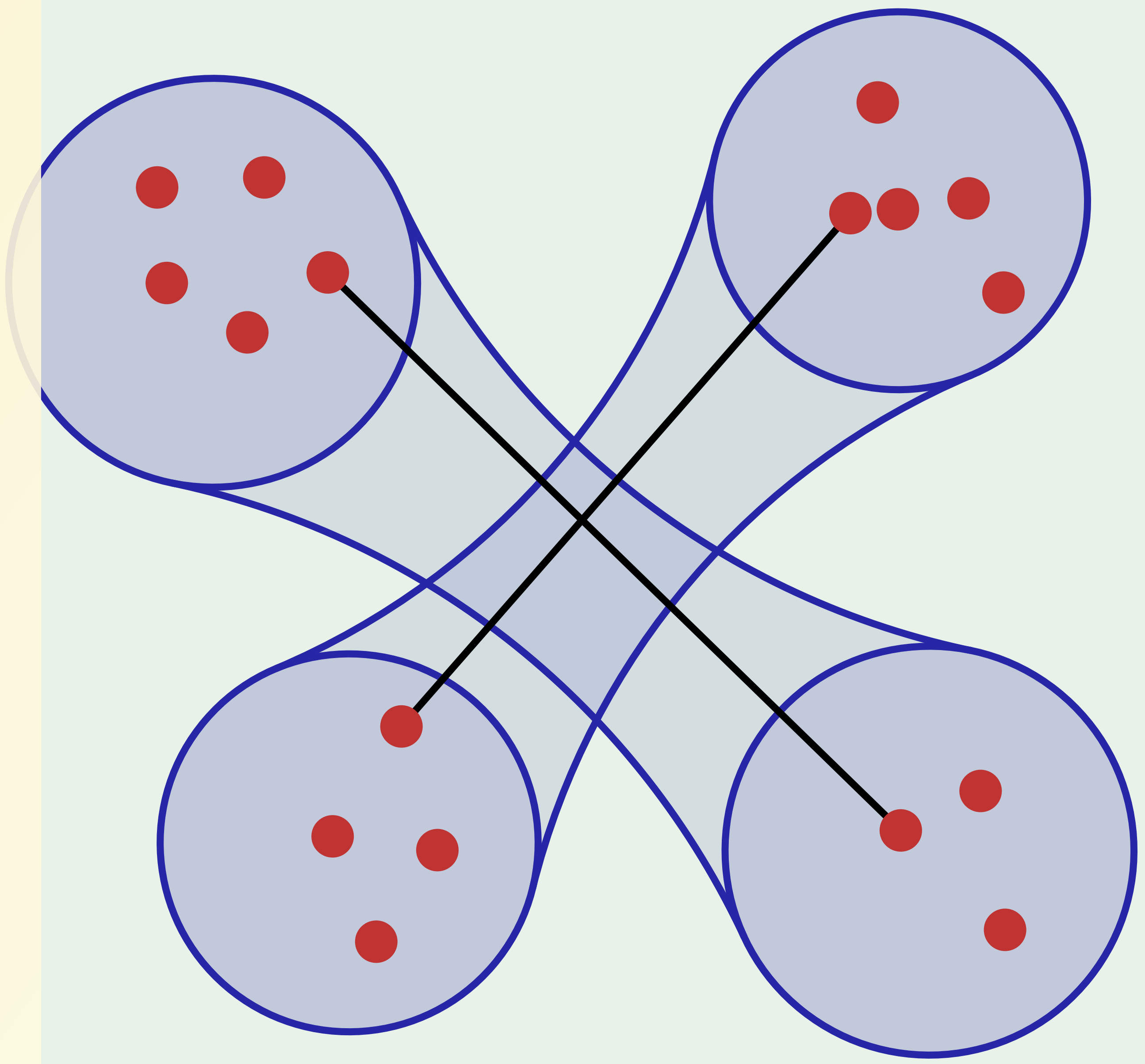


Problem: computing a MST may take up to $O(n\alpha(n))$ time.

For *planar* graphs, this is only $O(n)$.

Unfortunately, G need not be planar.

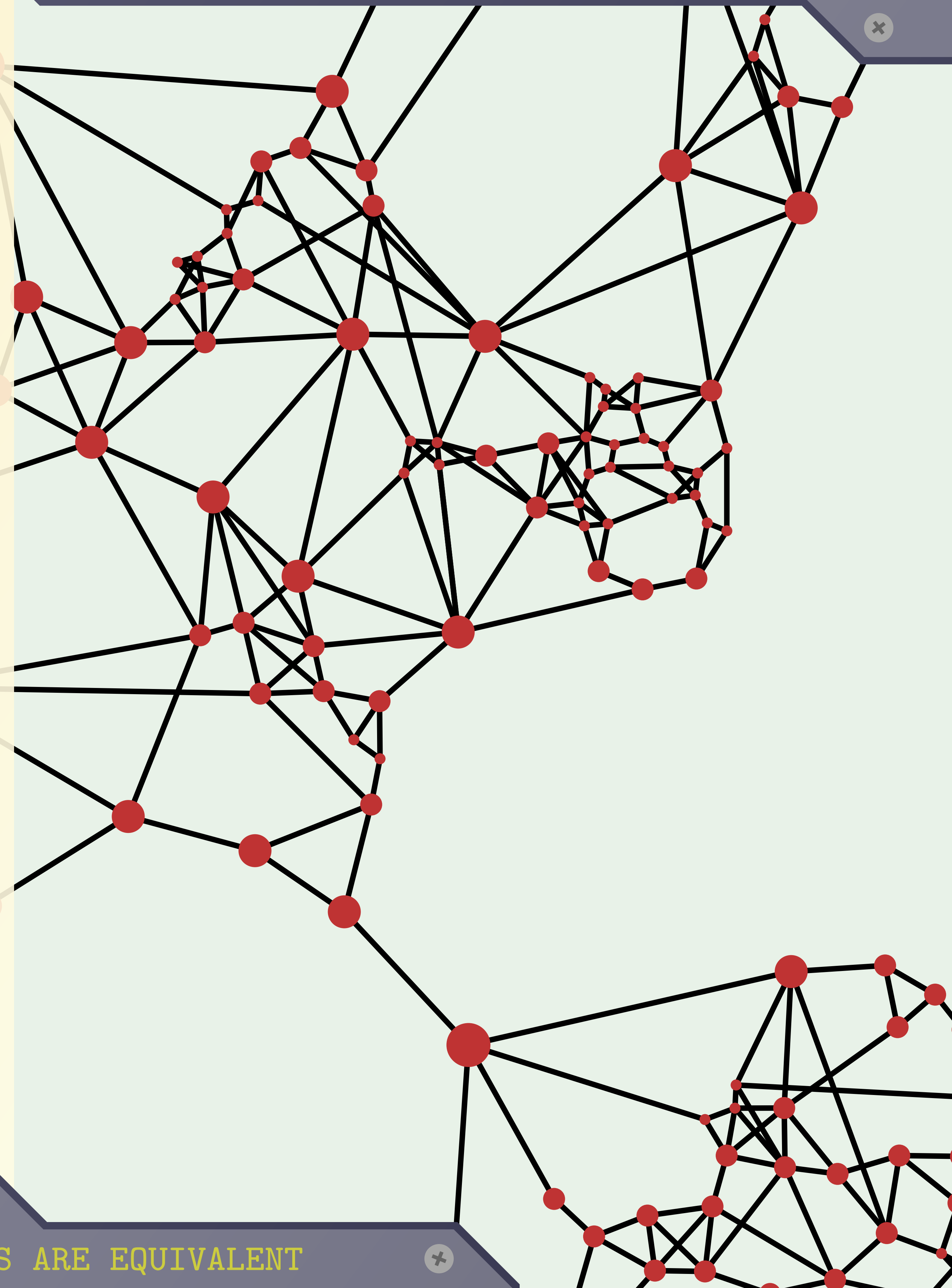
Claim: any edge e of G crosses at most $O(1)$ edges of length $\Omega(|e|)$.



EXTRACTING THE MST

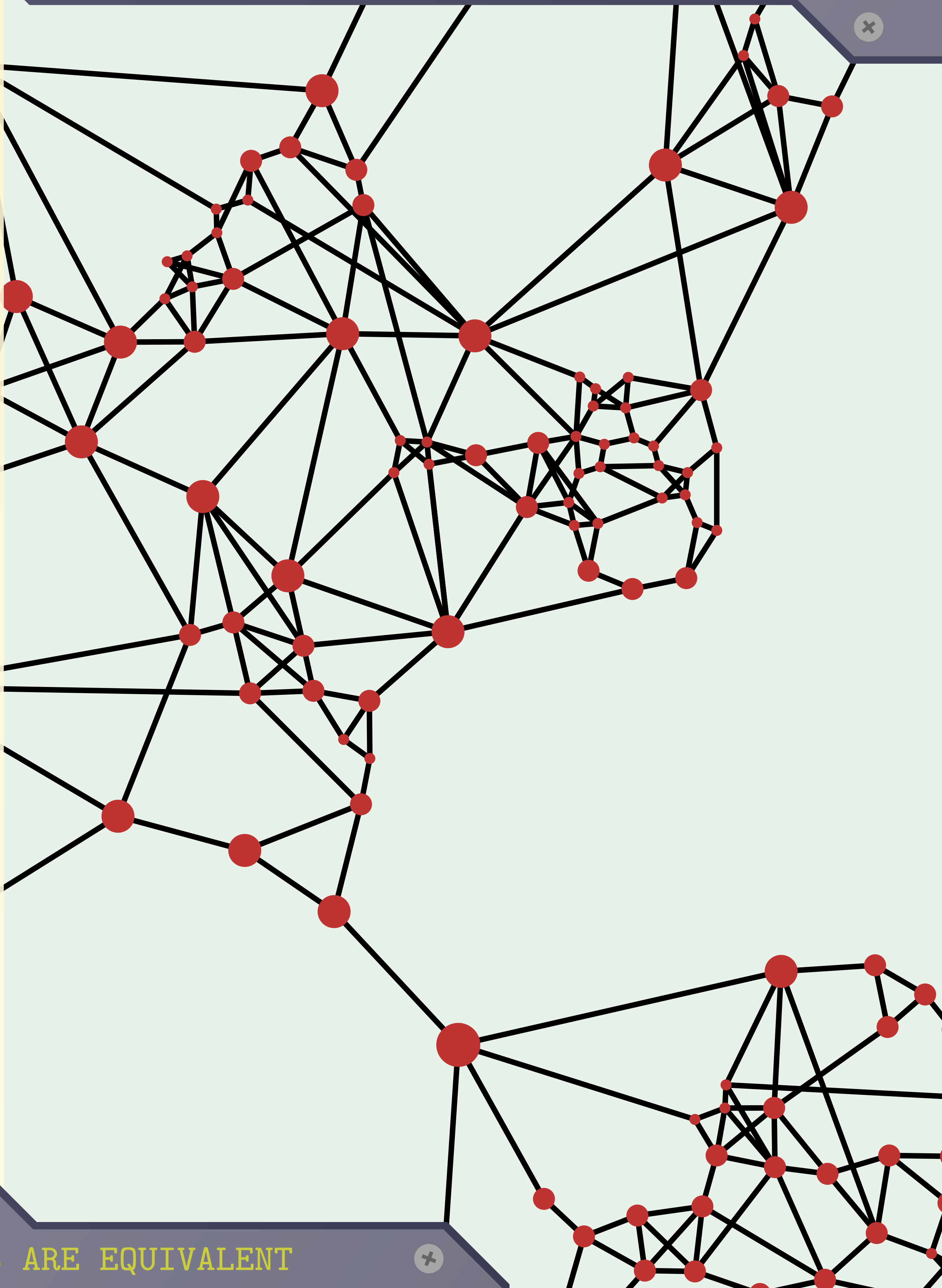
Start with P and G.

Start with P and G.



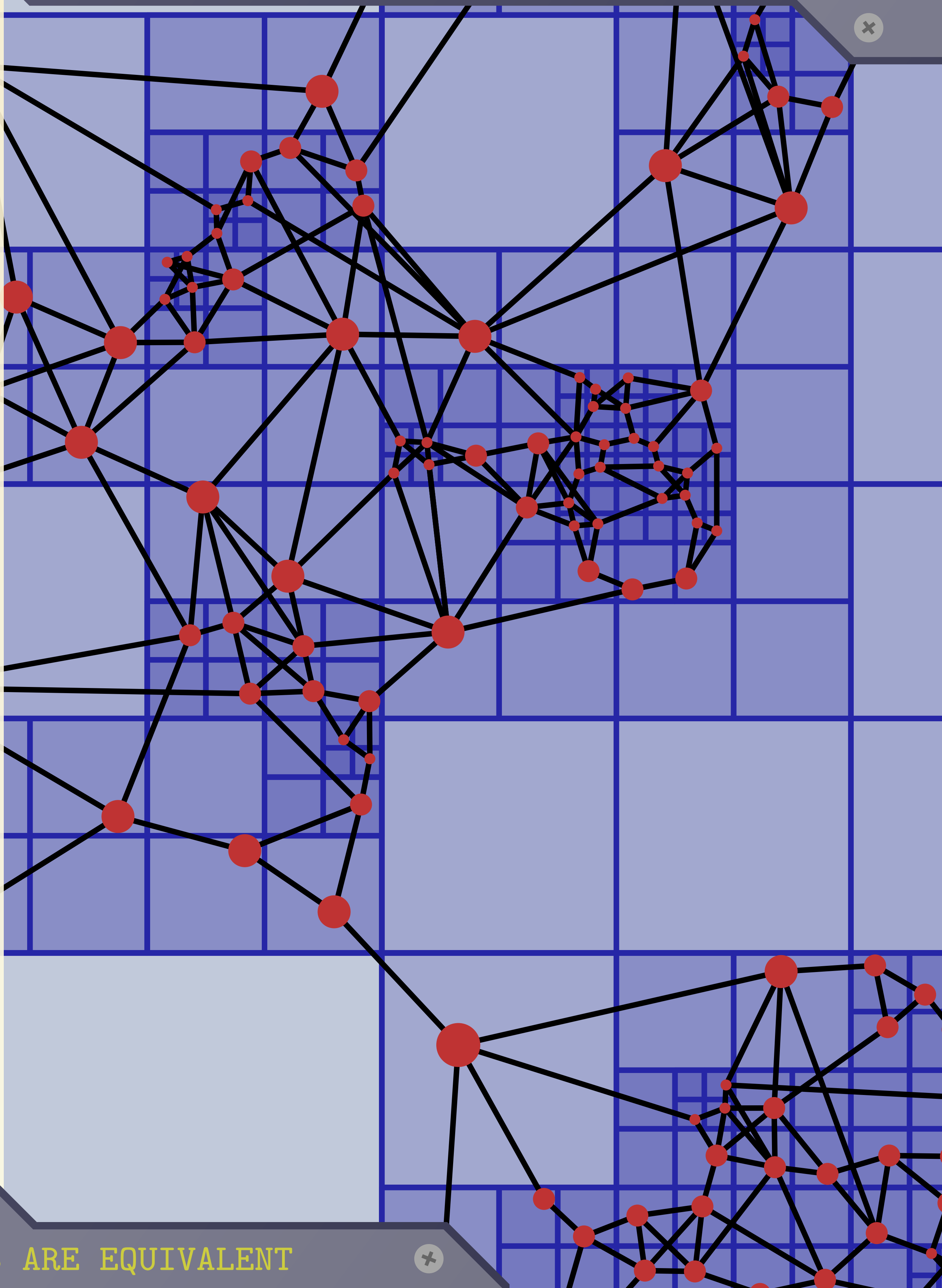
Start with P and G.

Remember we have a quadtree T on P.



Start with P and G.

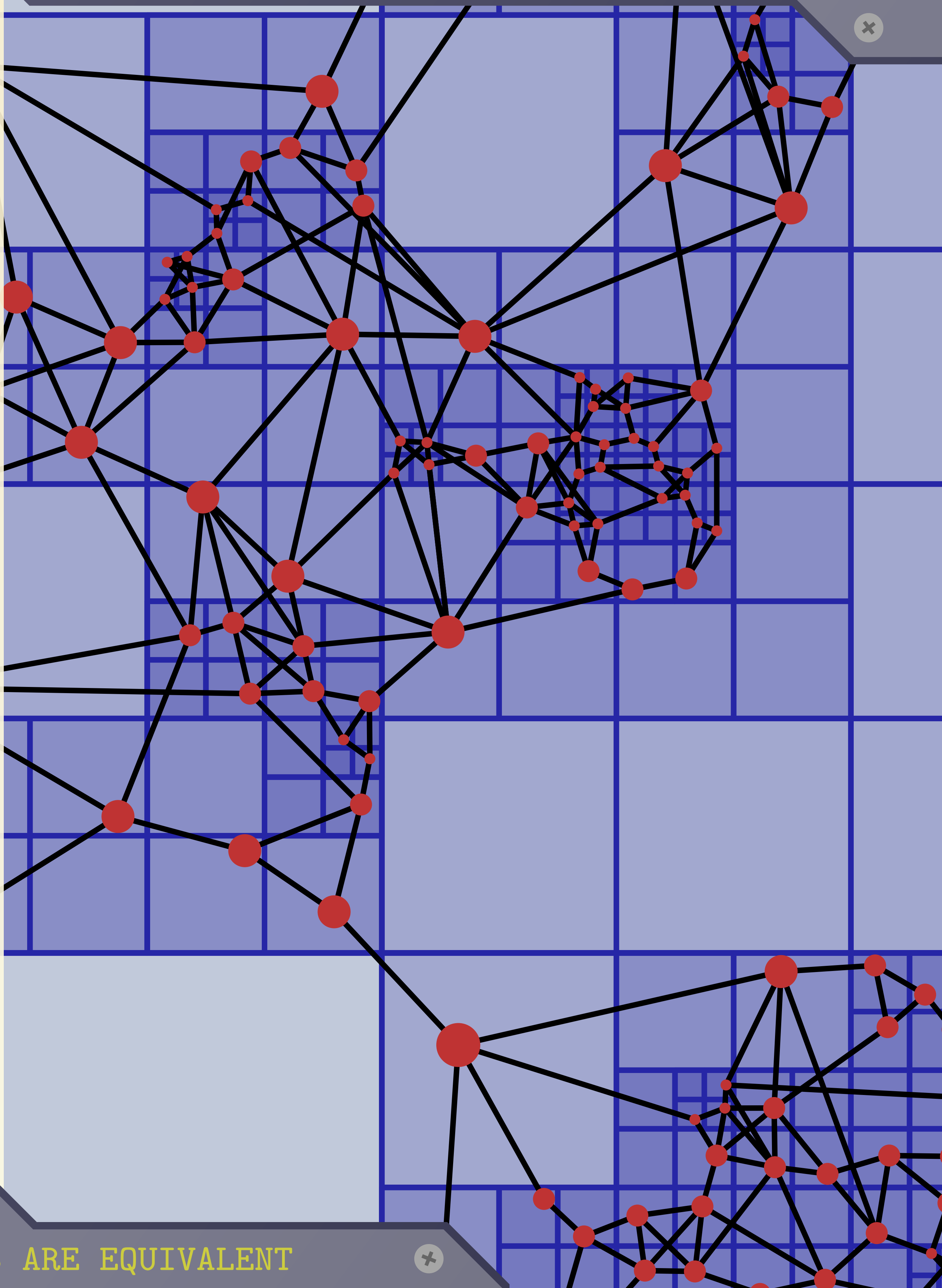
Remember we have a quadtree T on P.



Start with P and G.

Remember we have a quadtree T on P.

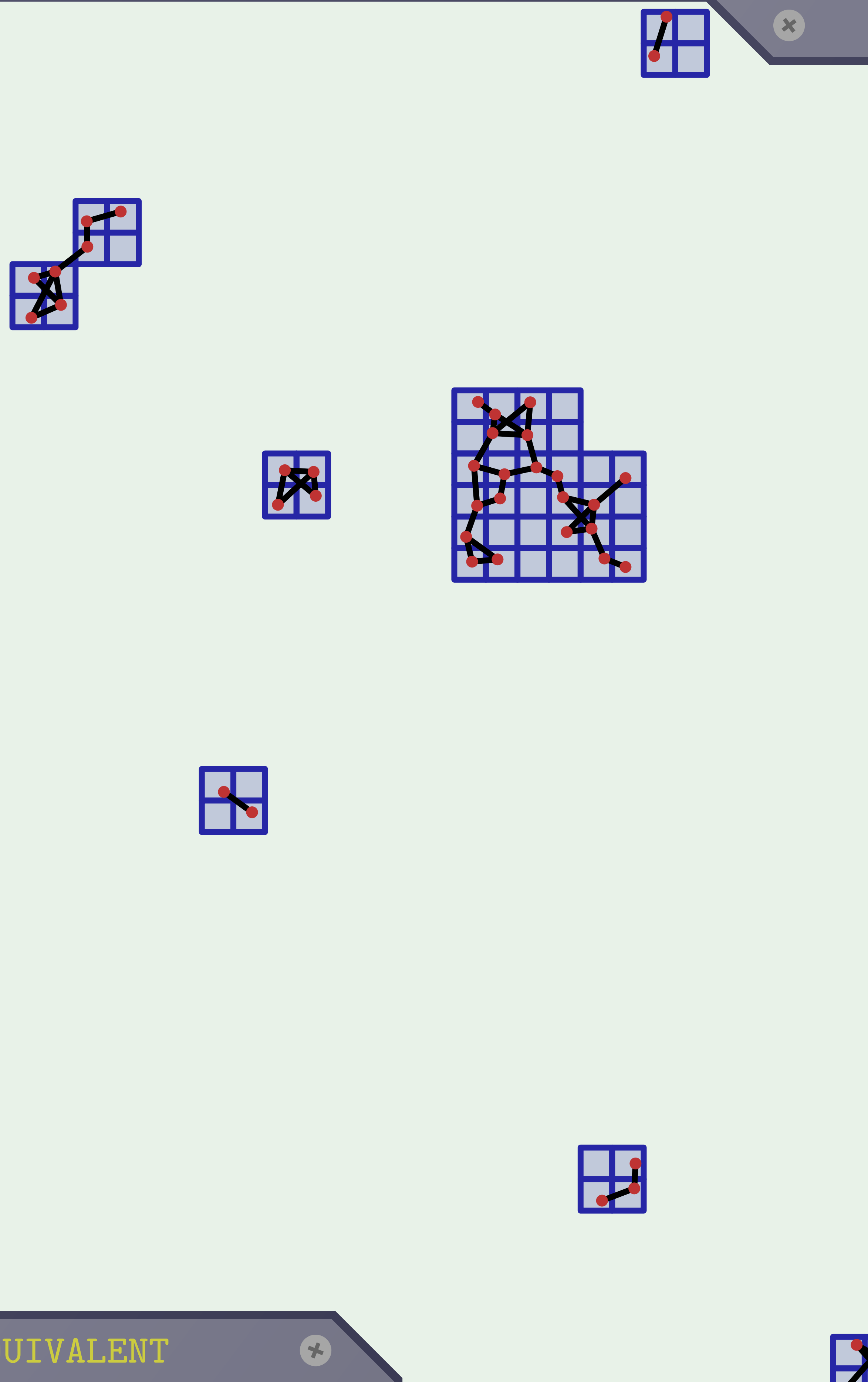
We can process the edges of G by increasing length, Borůvka-style.



Start with P and G.

Remember we have a quadtree T on P.

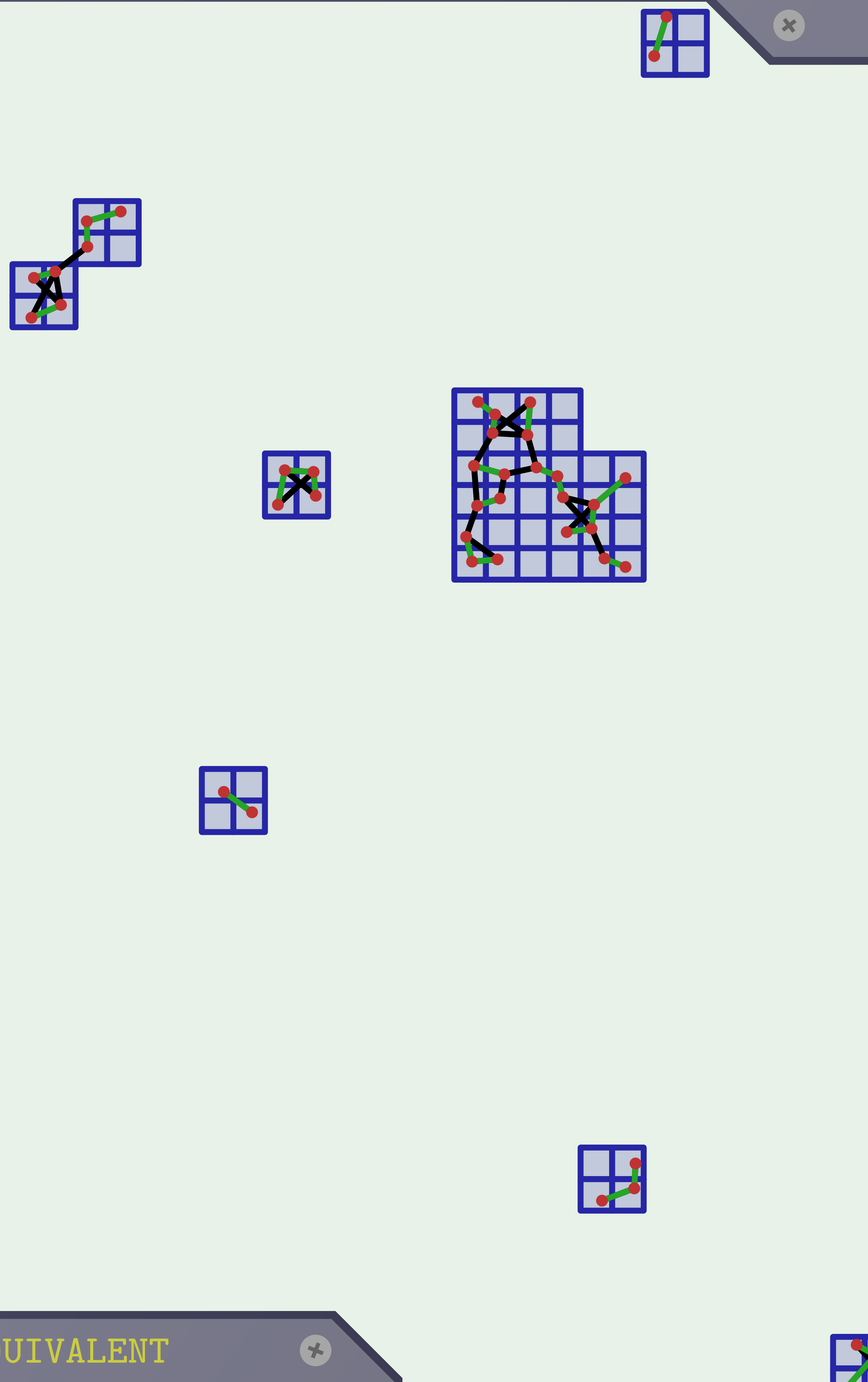
We can process the edges of G by increasing length, Borůvka-style.



Start with P and G.

Remember we have a quadtree T on P.

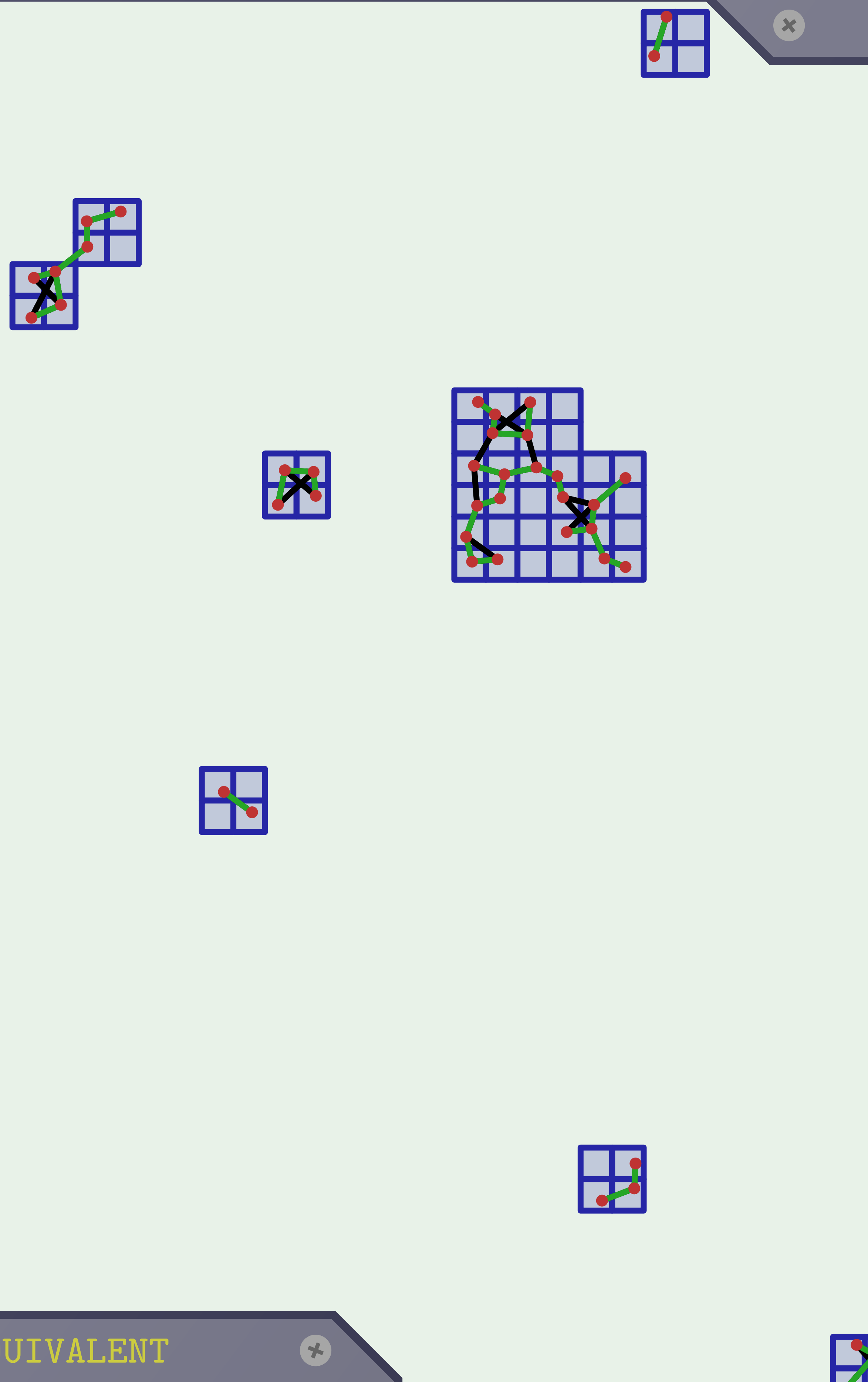
We can process the edges of G by increasing length, Borůvka-style.



Start with P and G.

Remember we have a quadtree T on P.

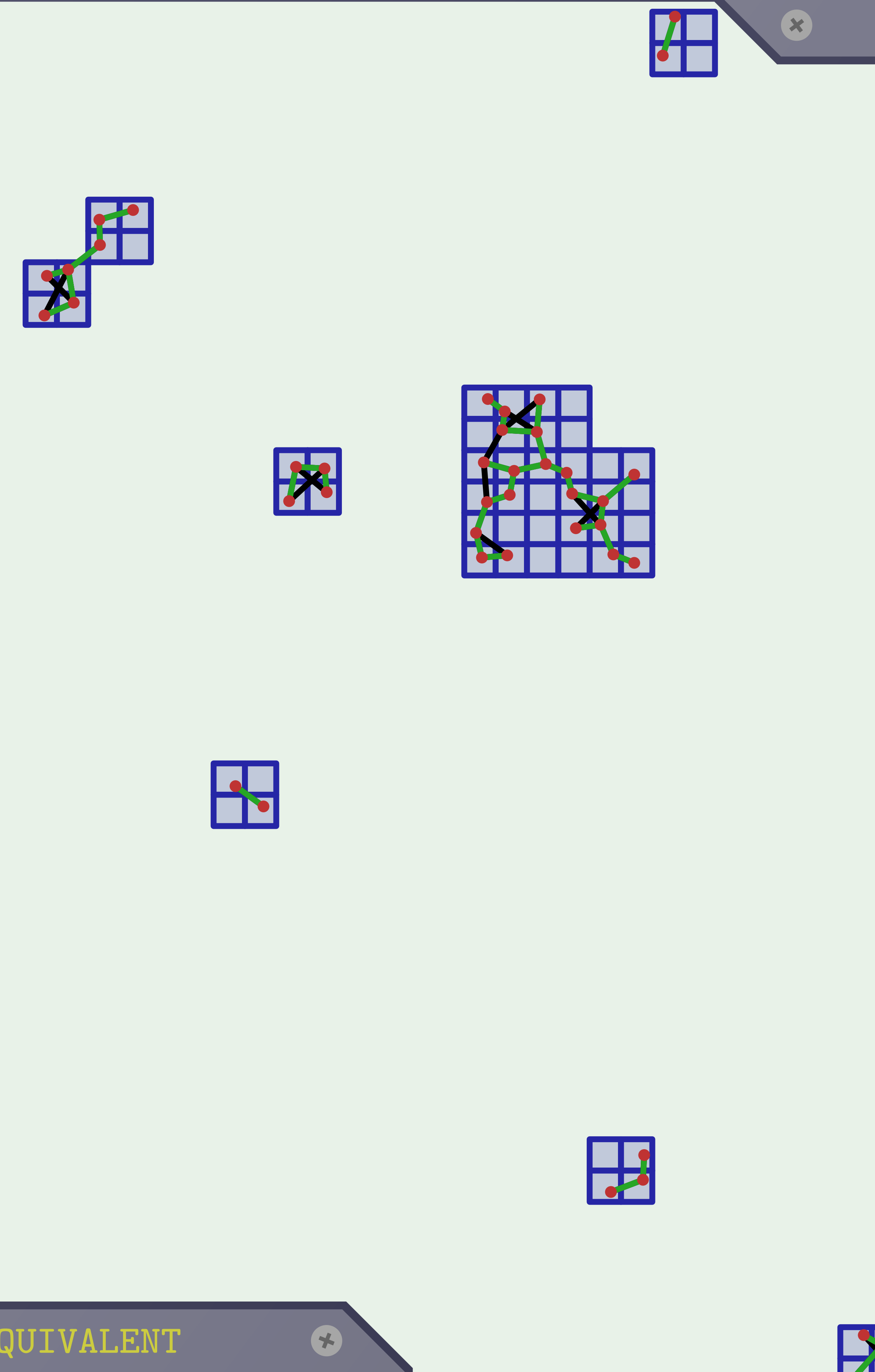
We can process the edges of G by increasing length, Borůvka-style.



Start with P and G.

Remember we have a quadtree T on P.

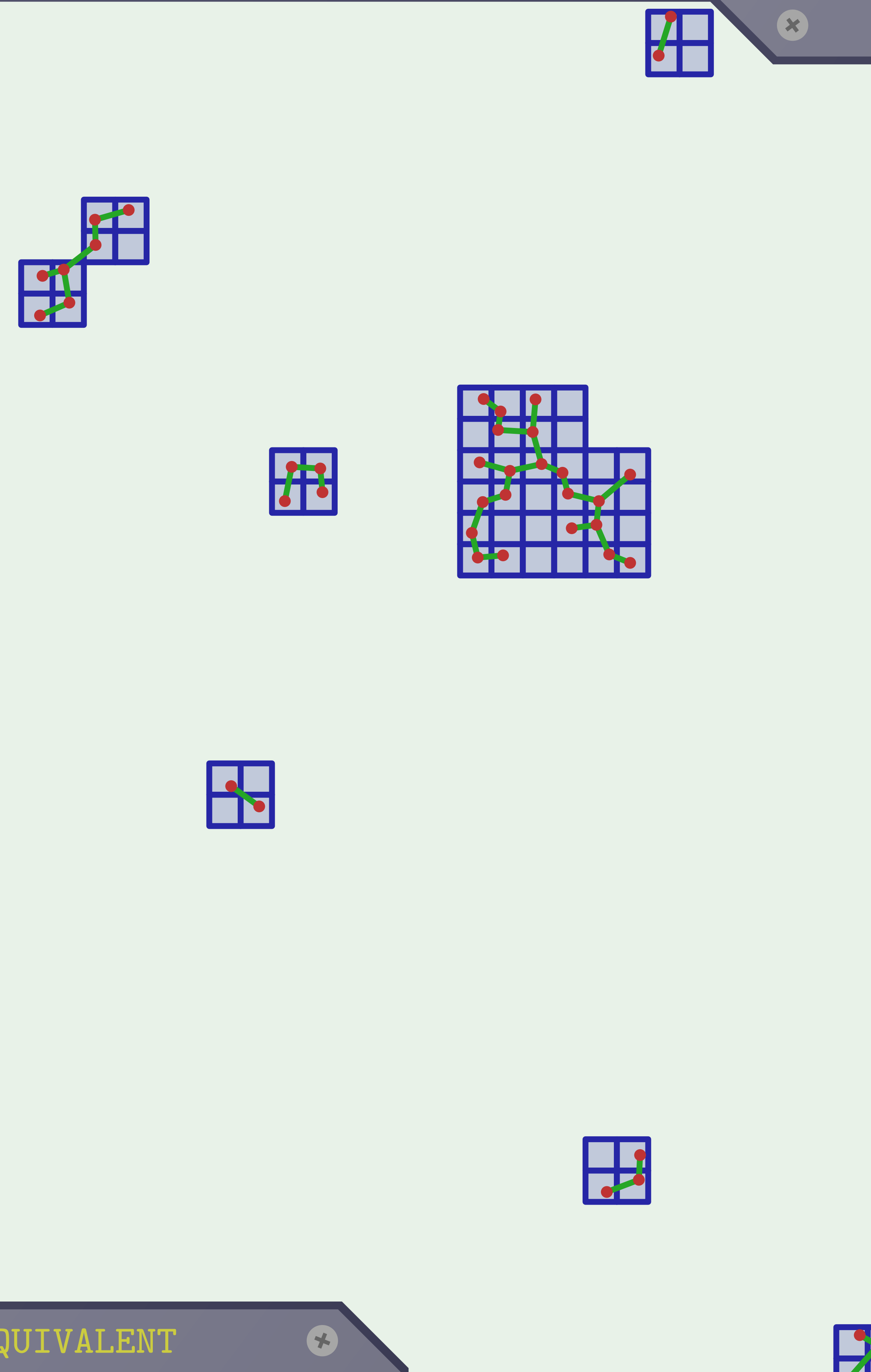
We can process the edges of G by increasing length, Borůvka-style.



Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

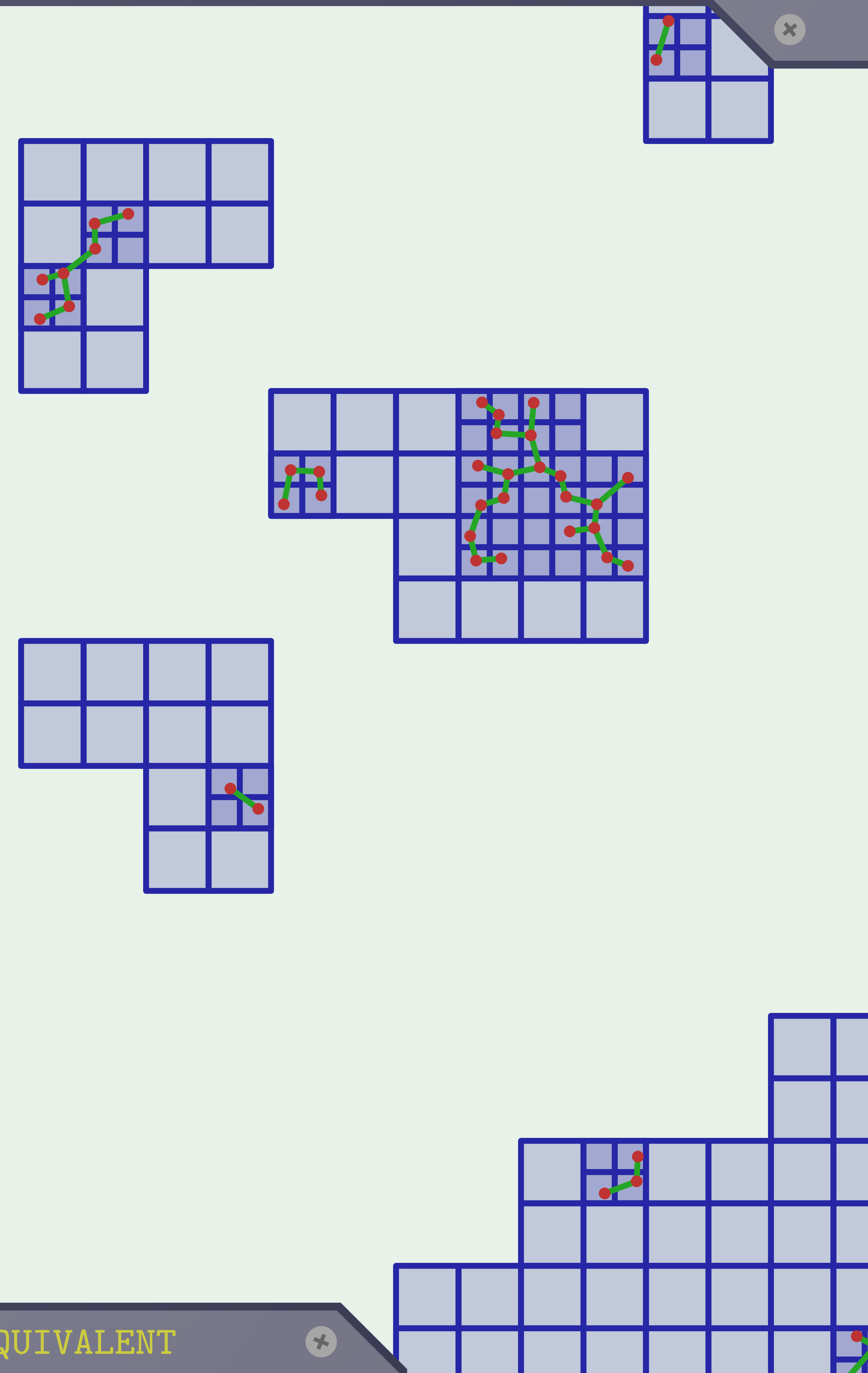


EXTRACTING THE MST

Start with P and G.

Remember we have a quadtree T on P.

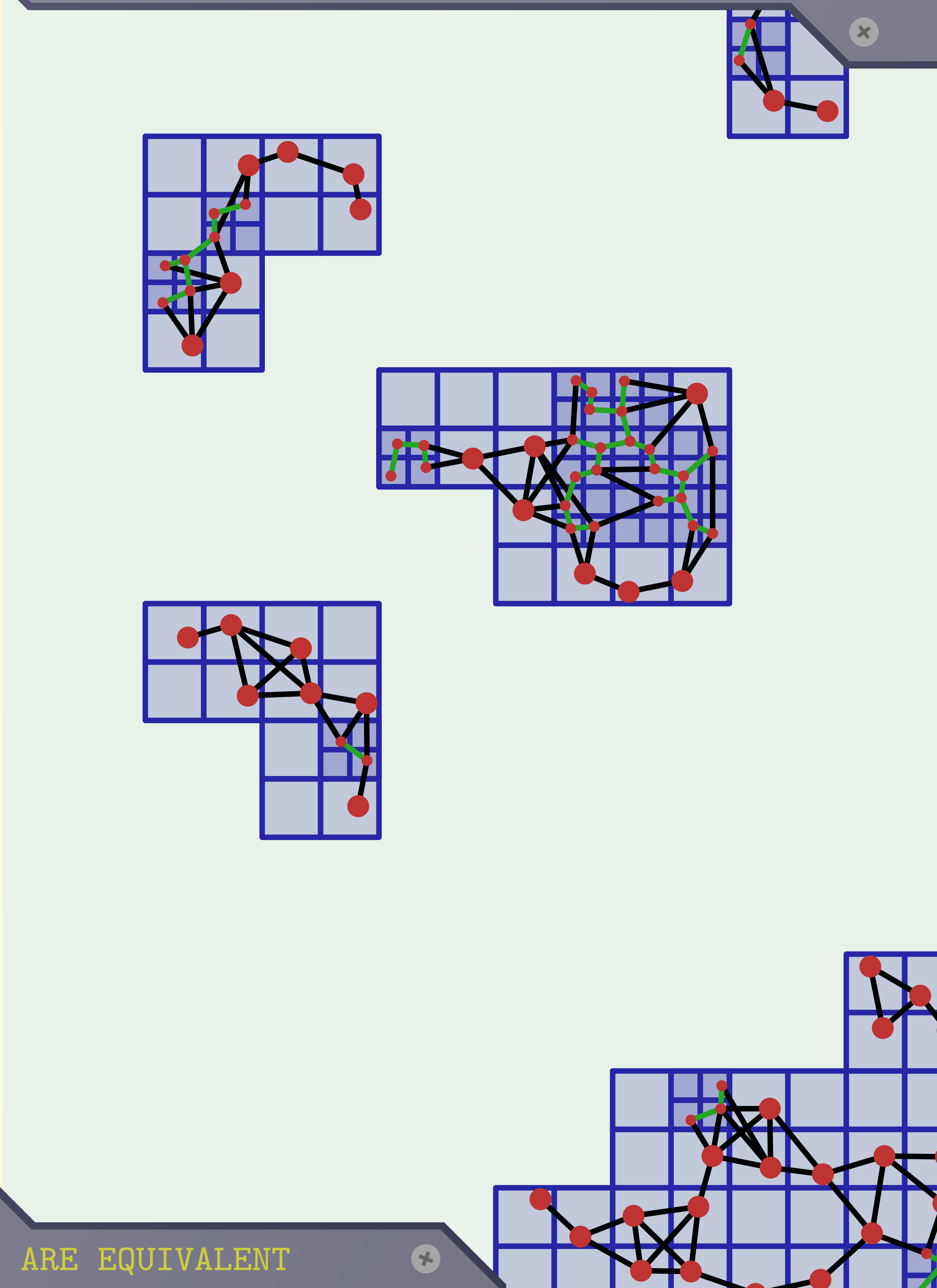
We can process the edges of G by increasing length, Borůvka-style.



Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

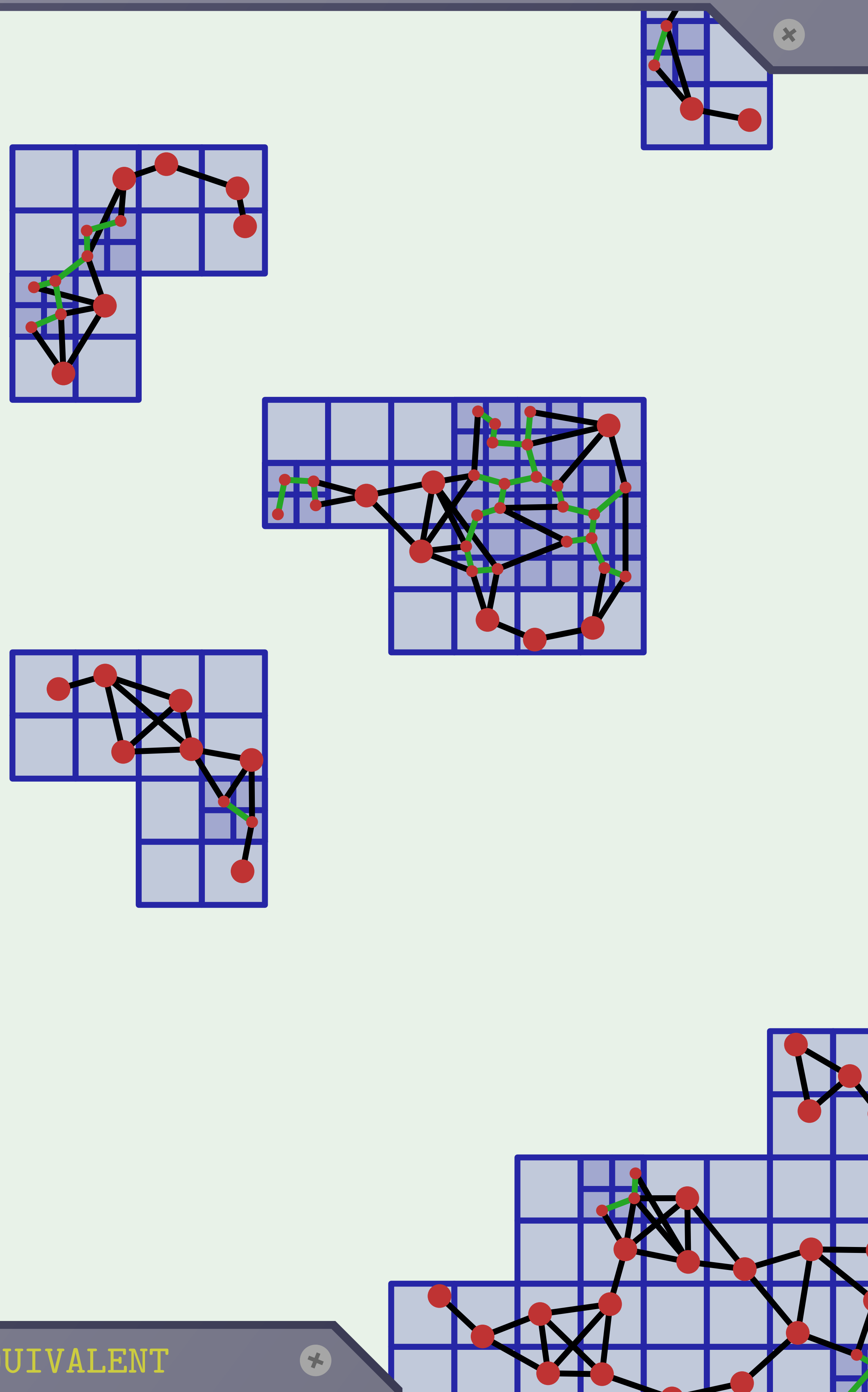


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

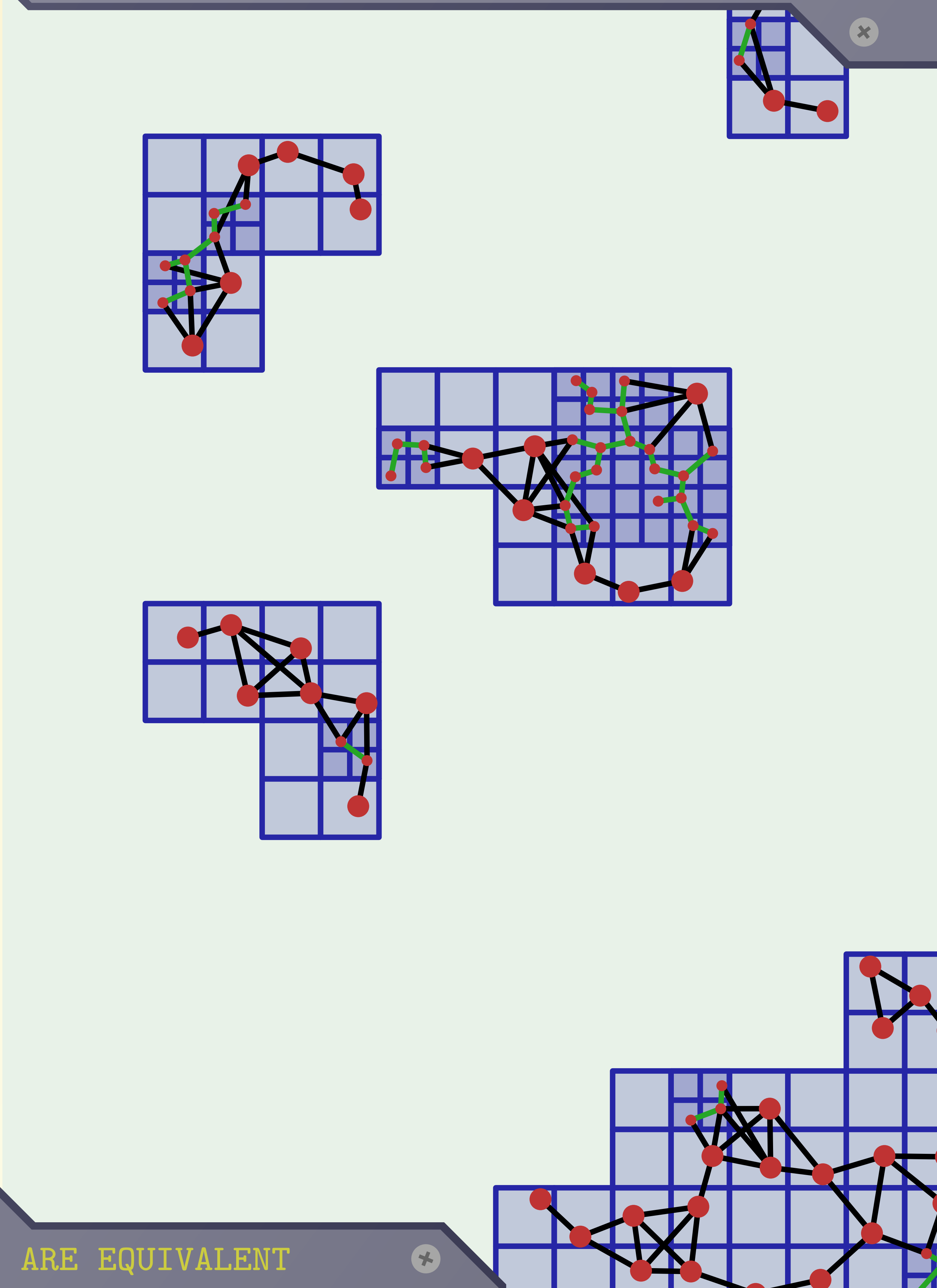


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

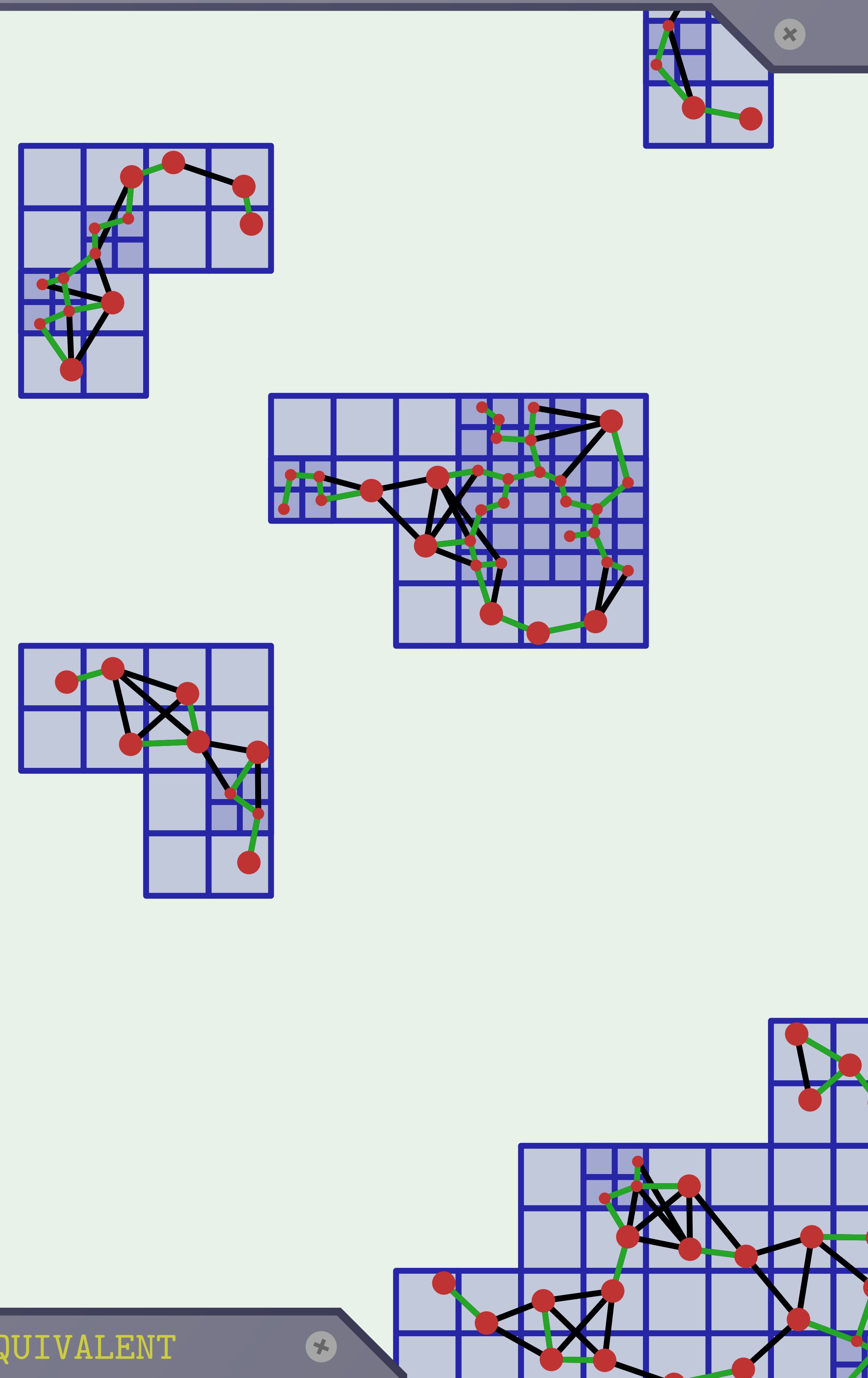


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

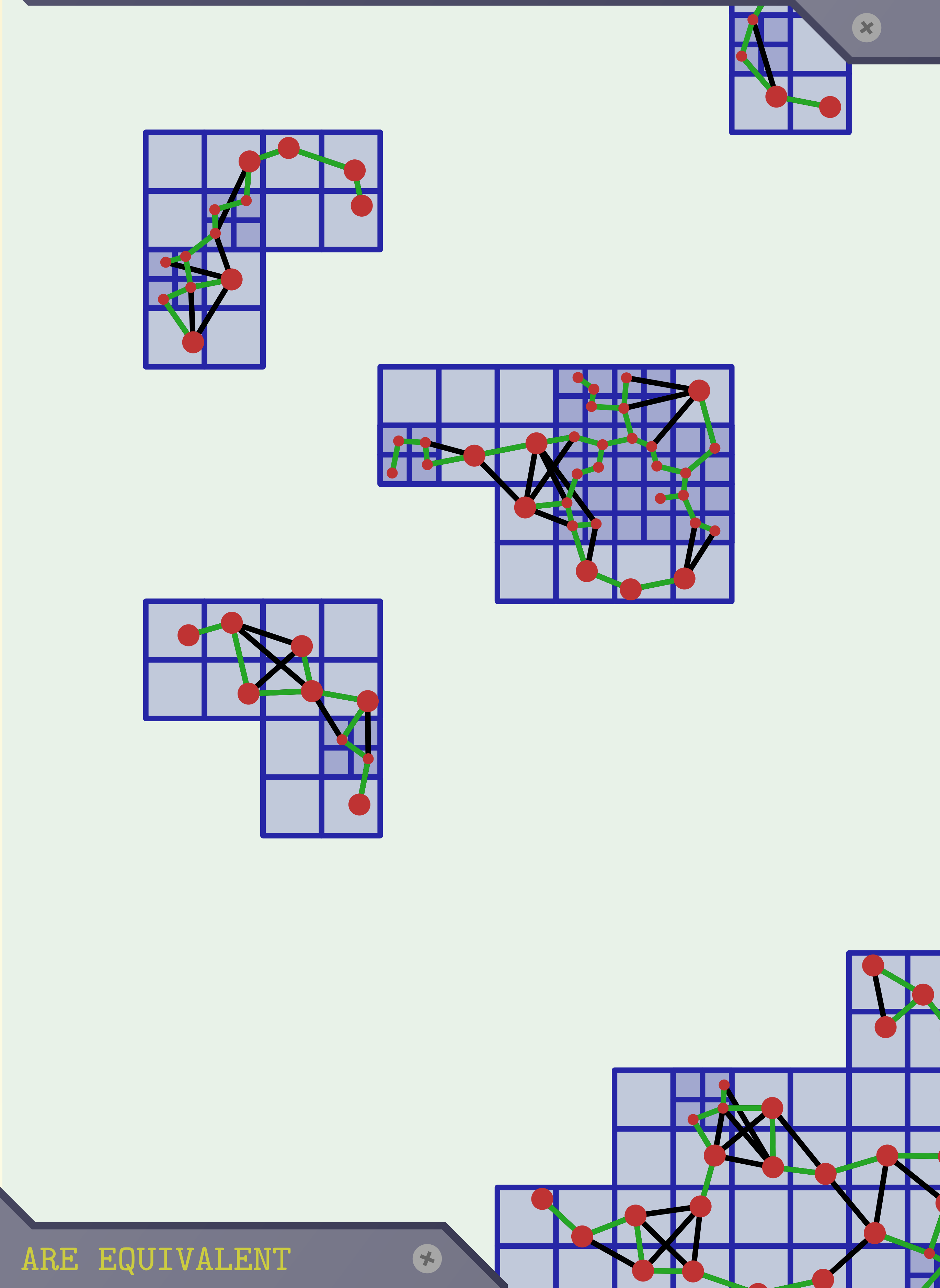


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

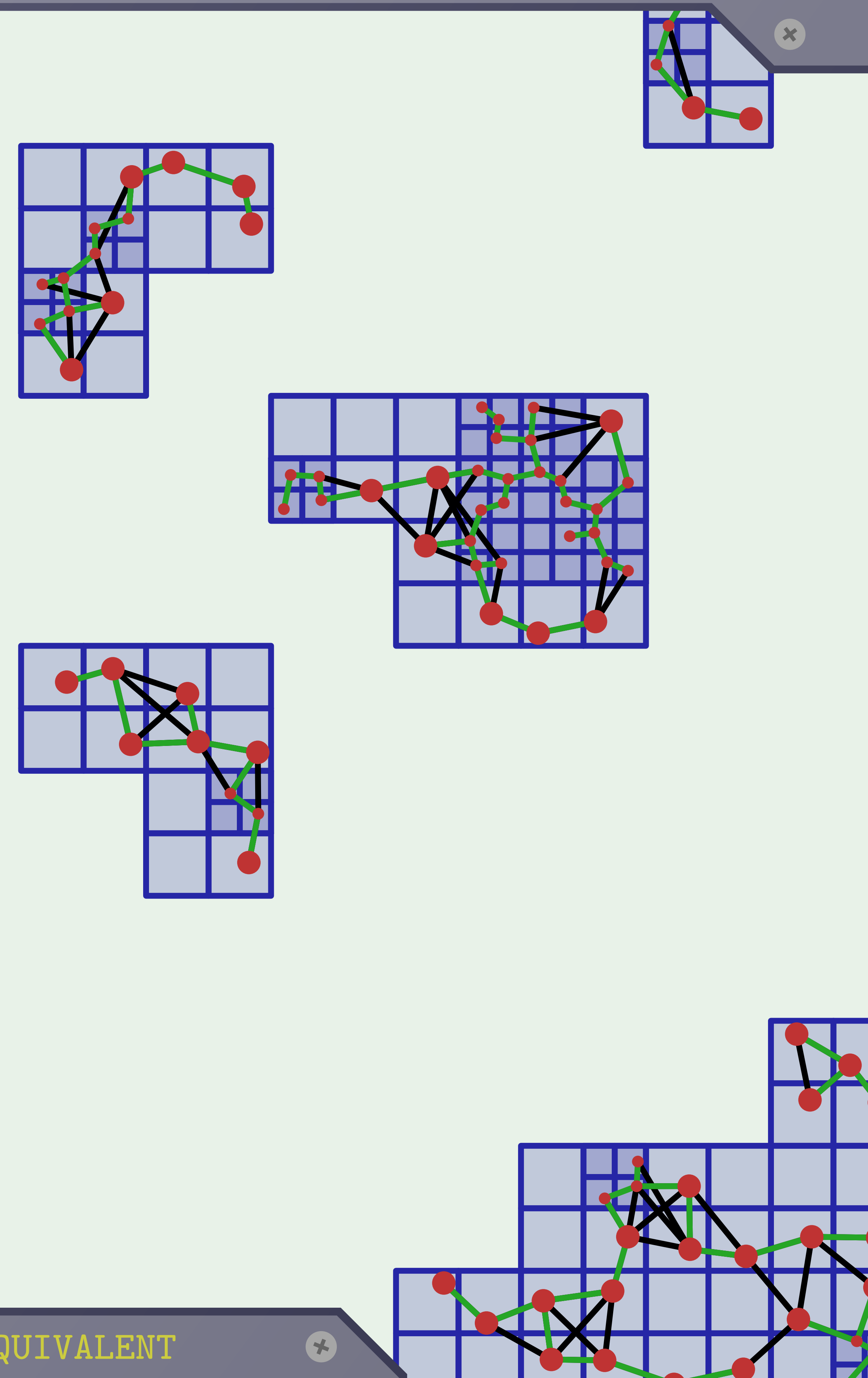


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

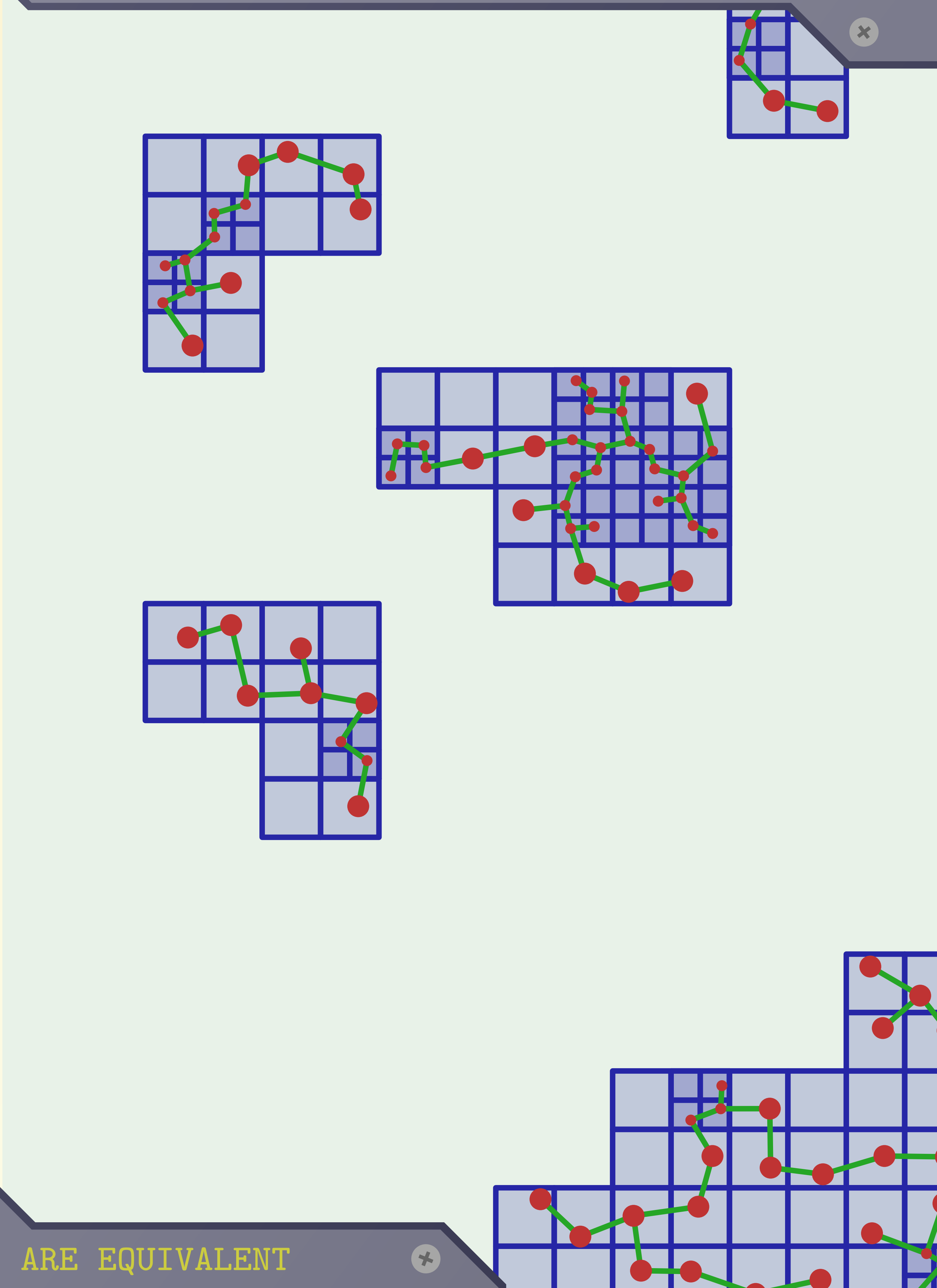


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

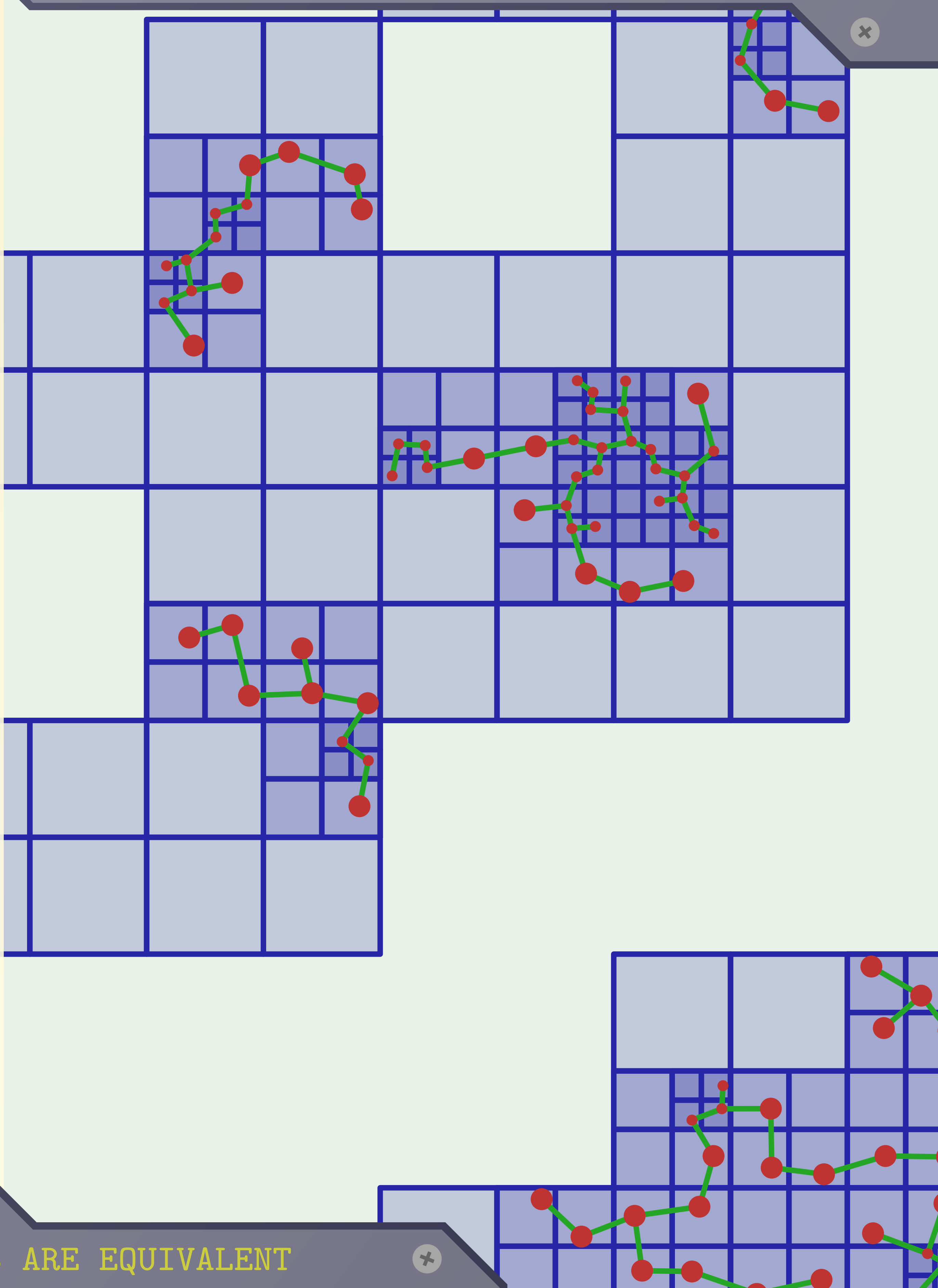


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

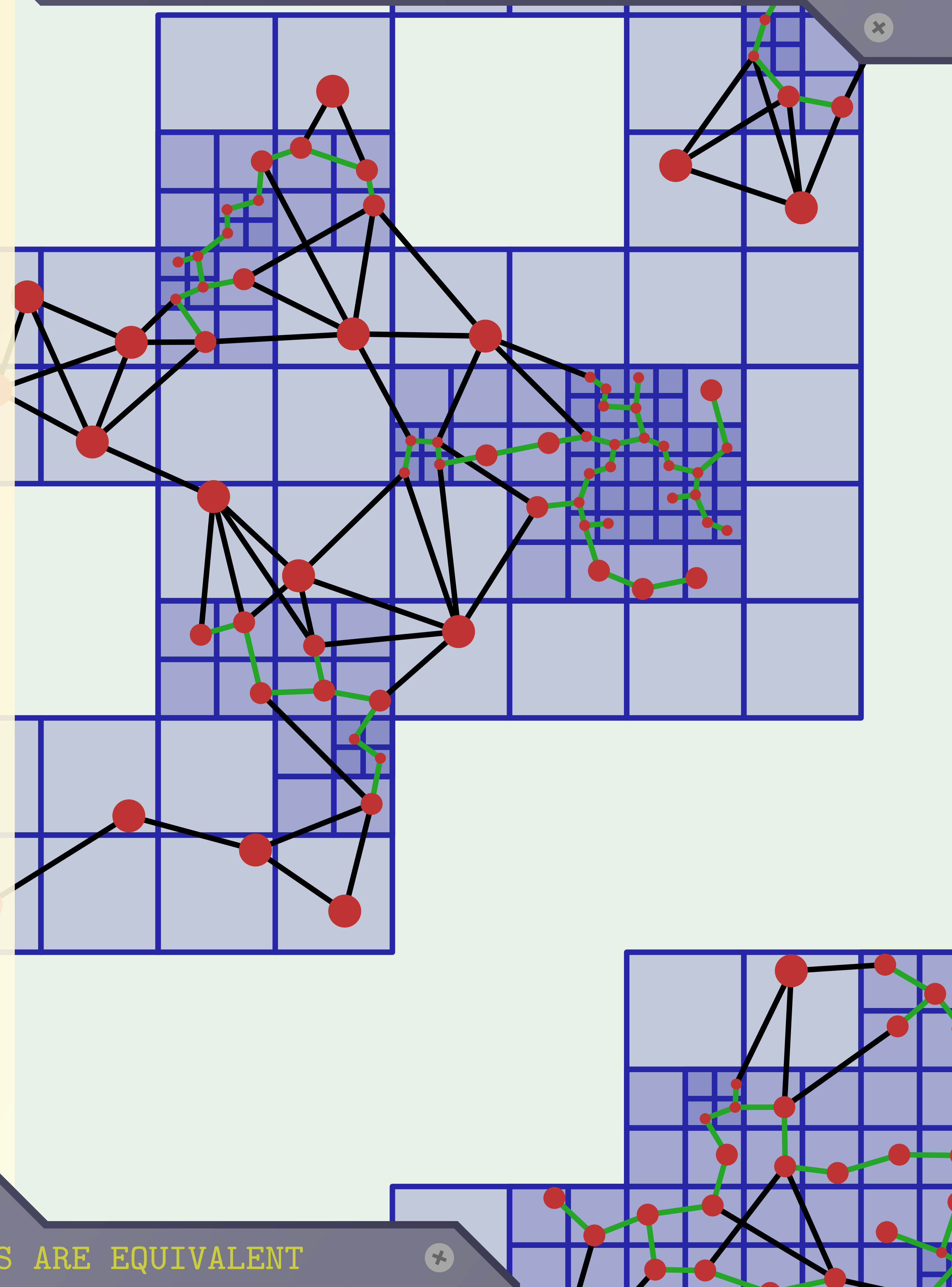


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

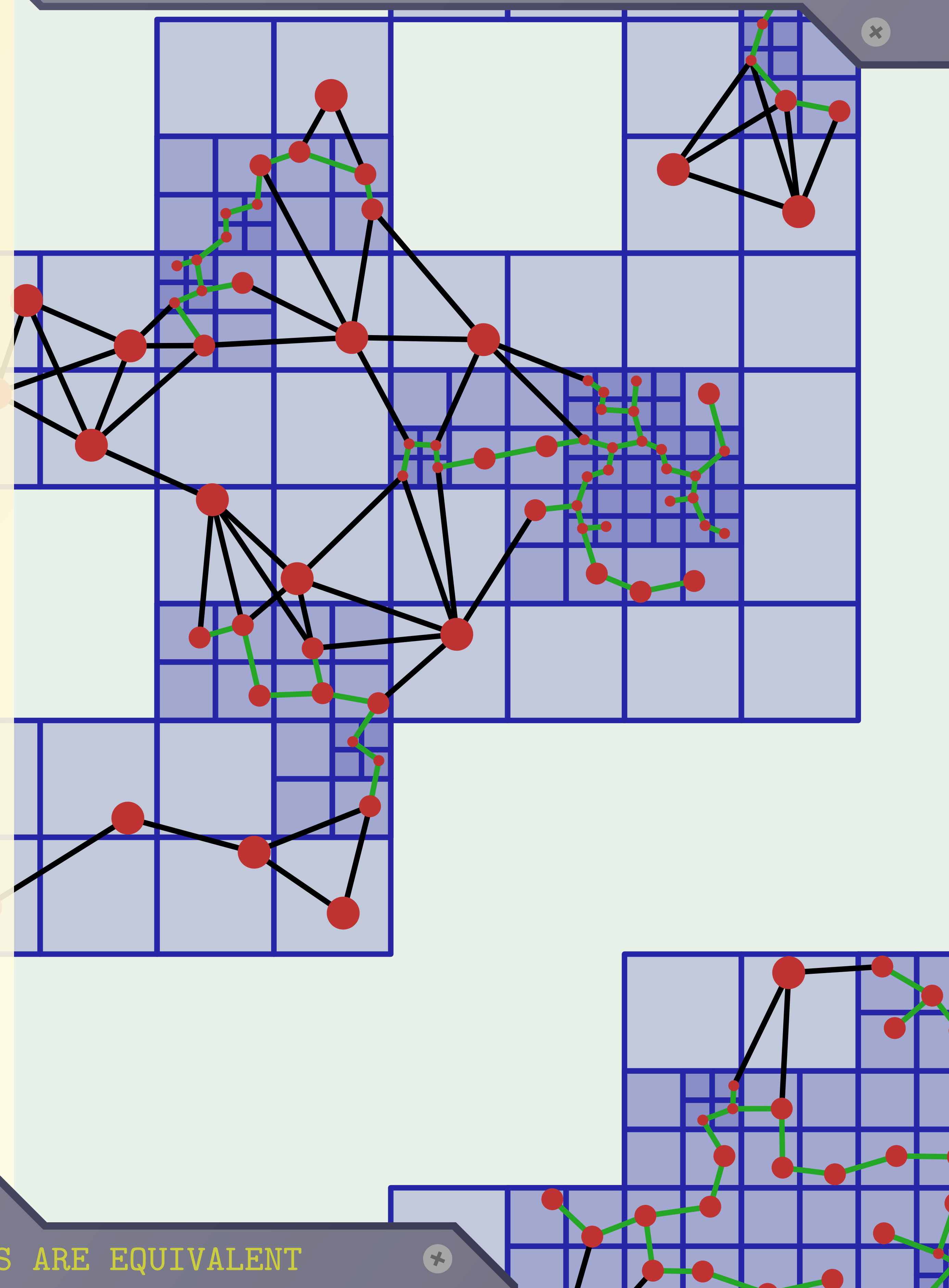


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

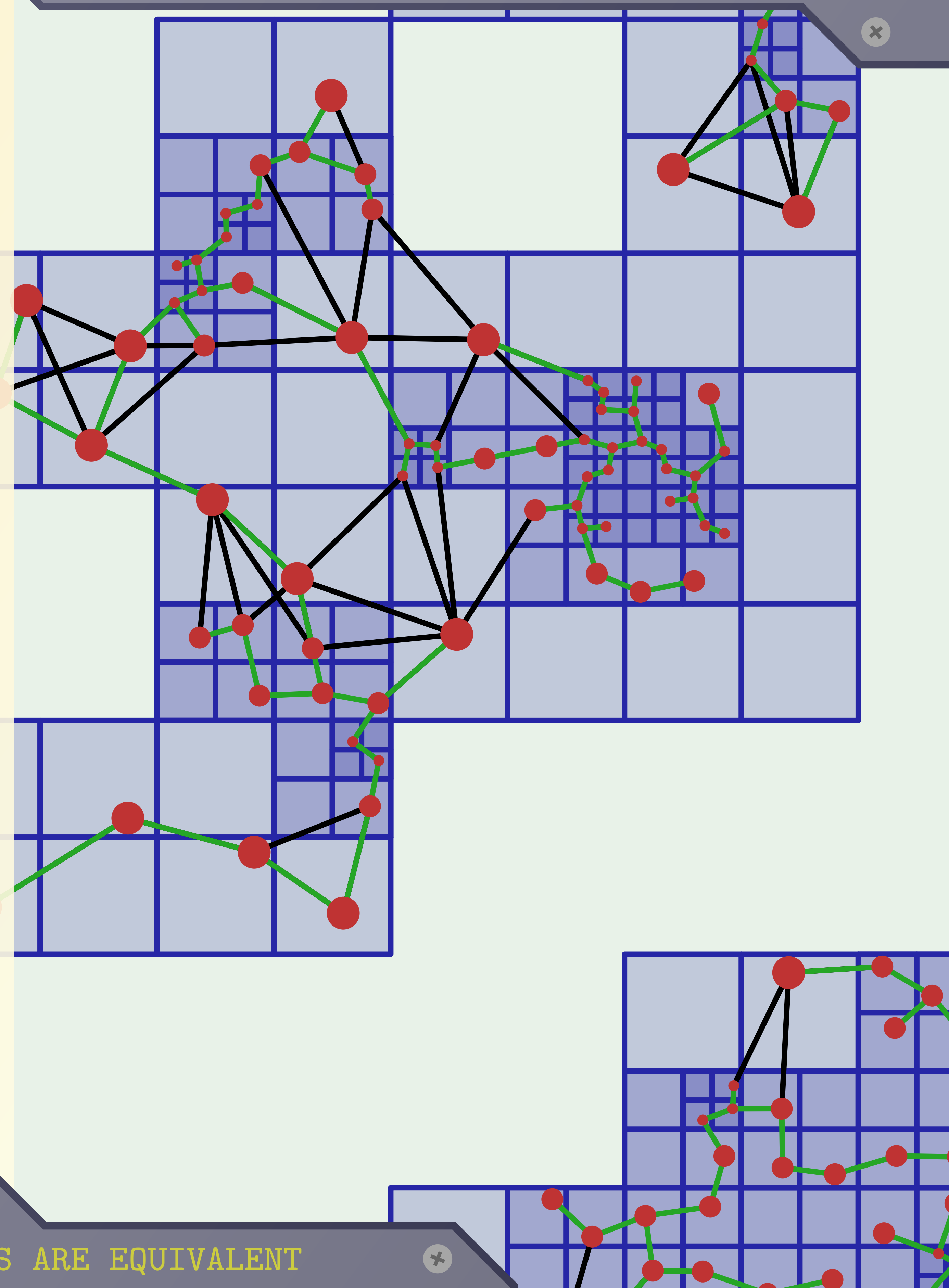


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

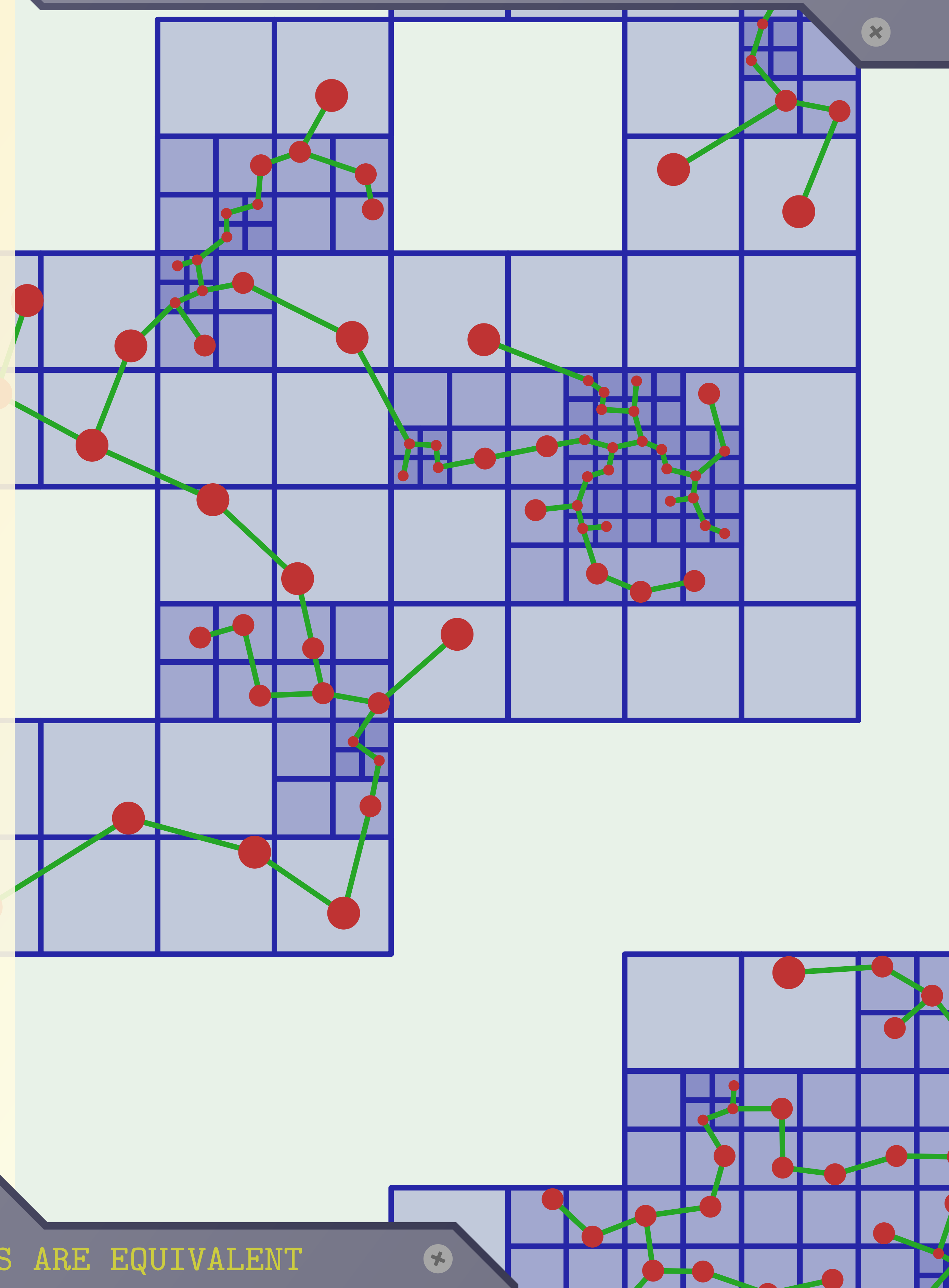


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

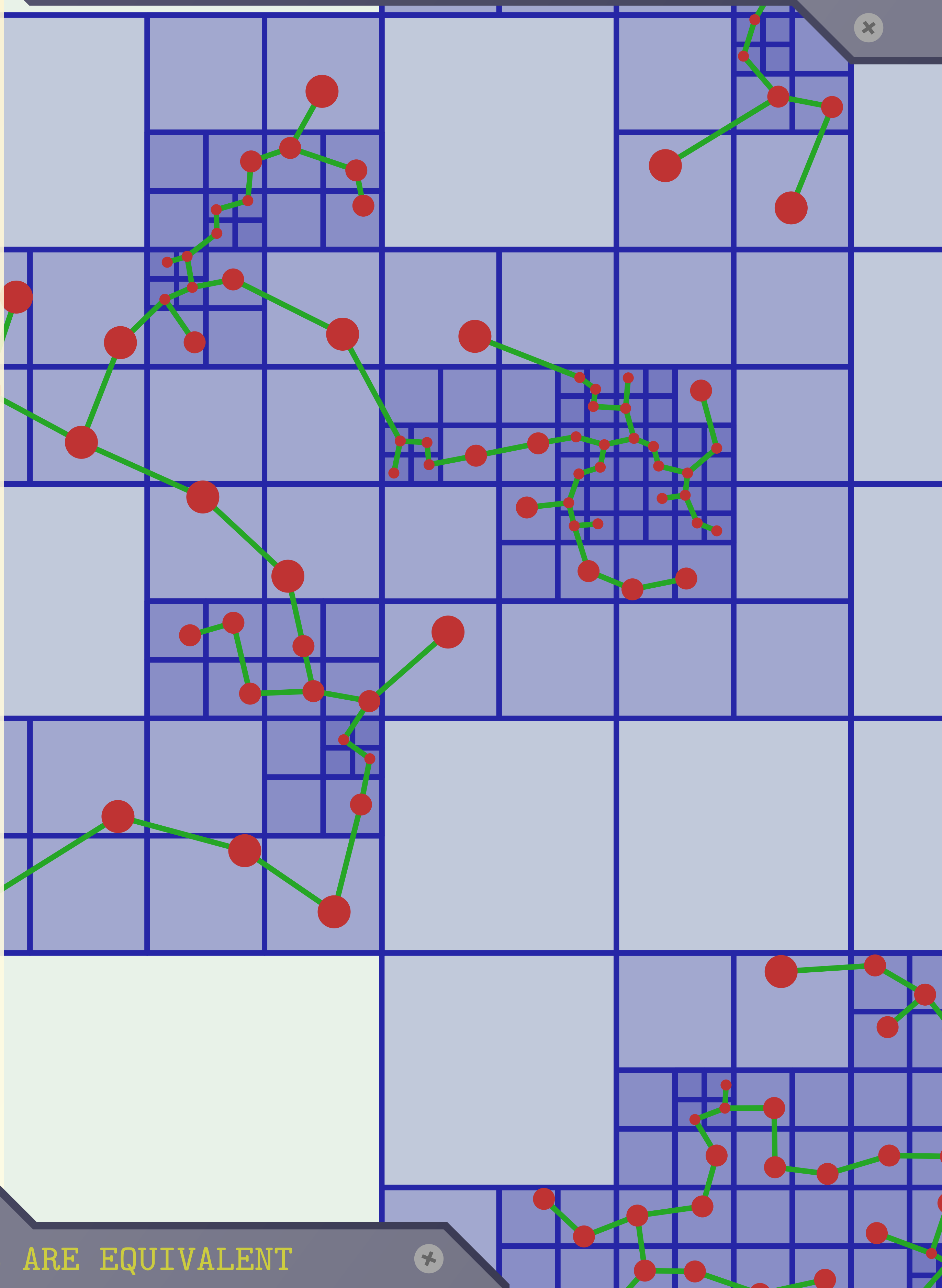


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

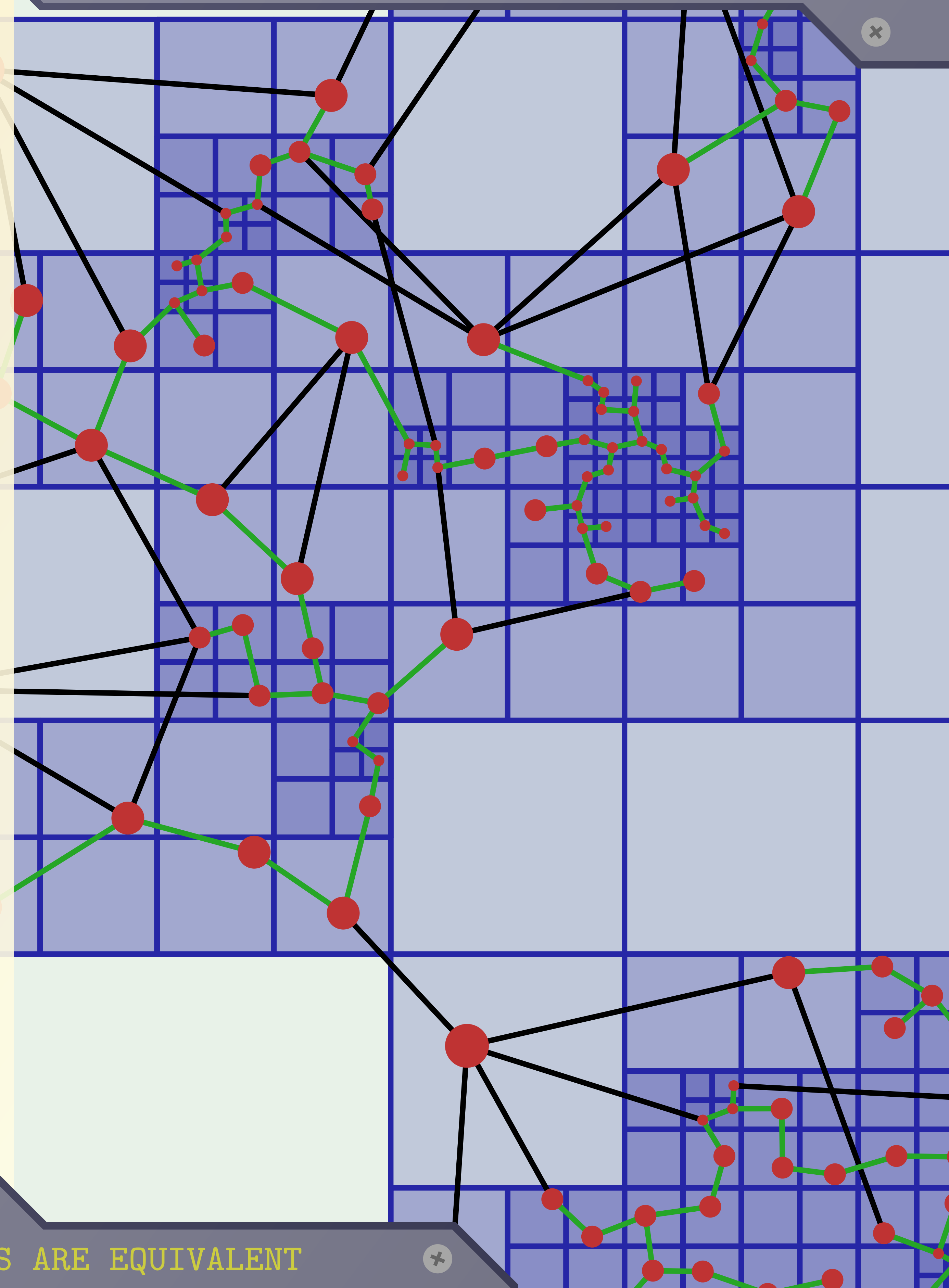


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

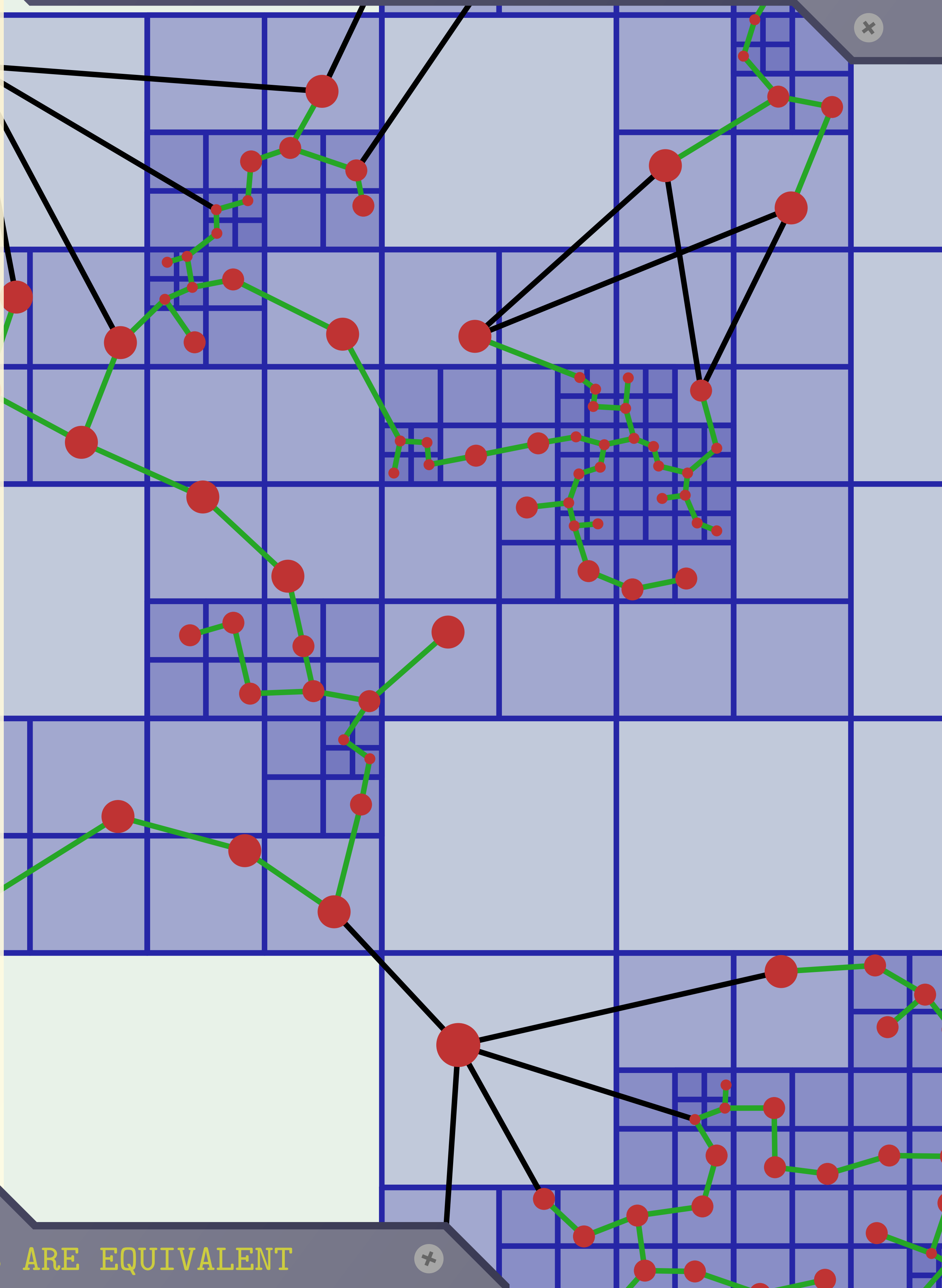


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

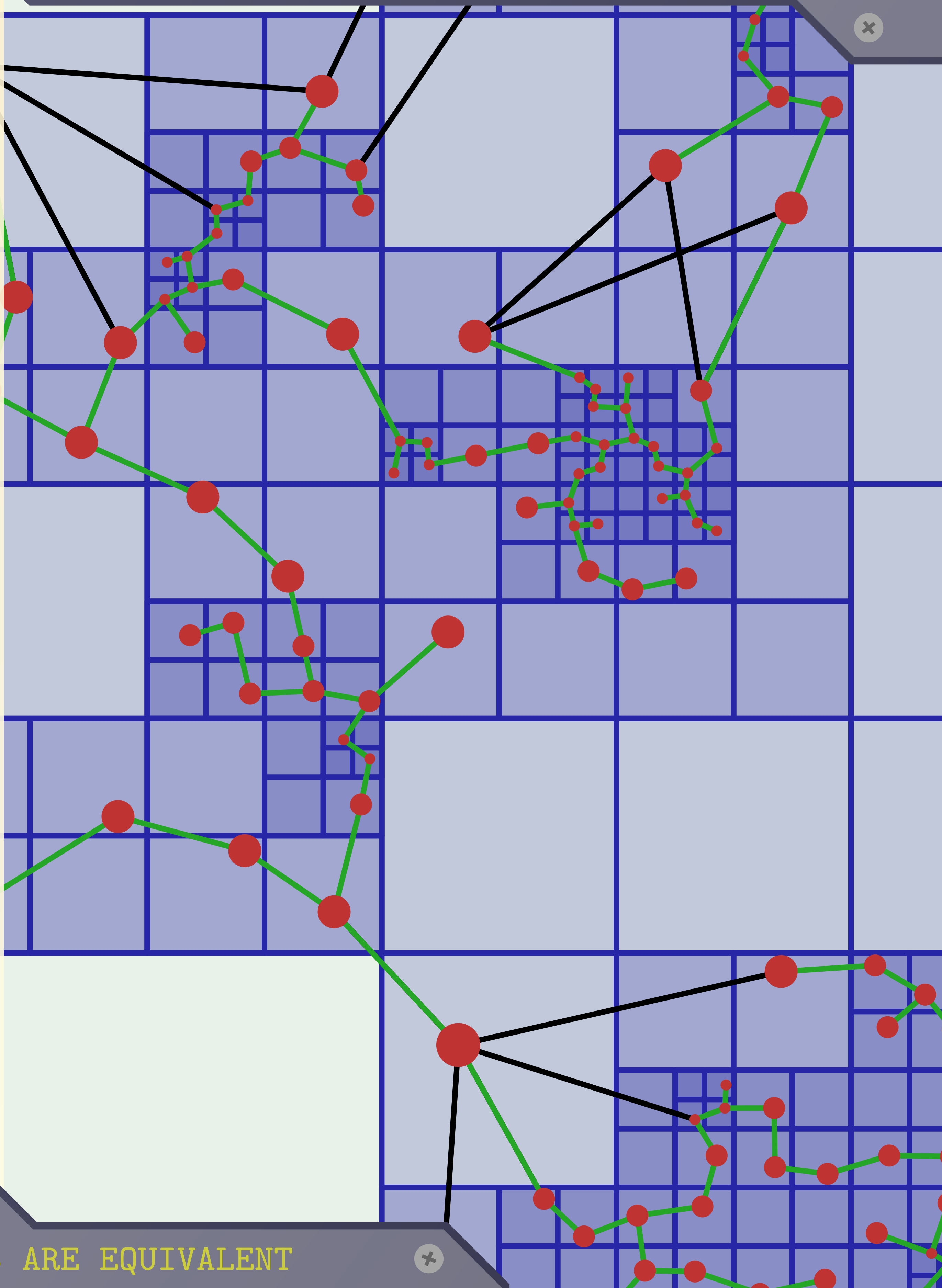


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

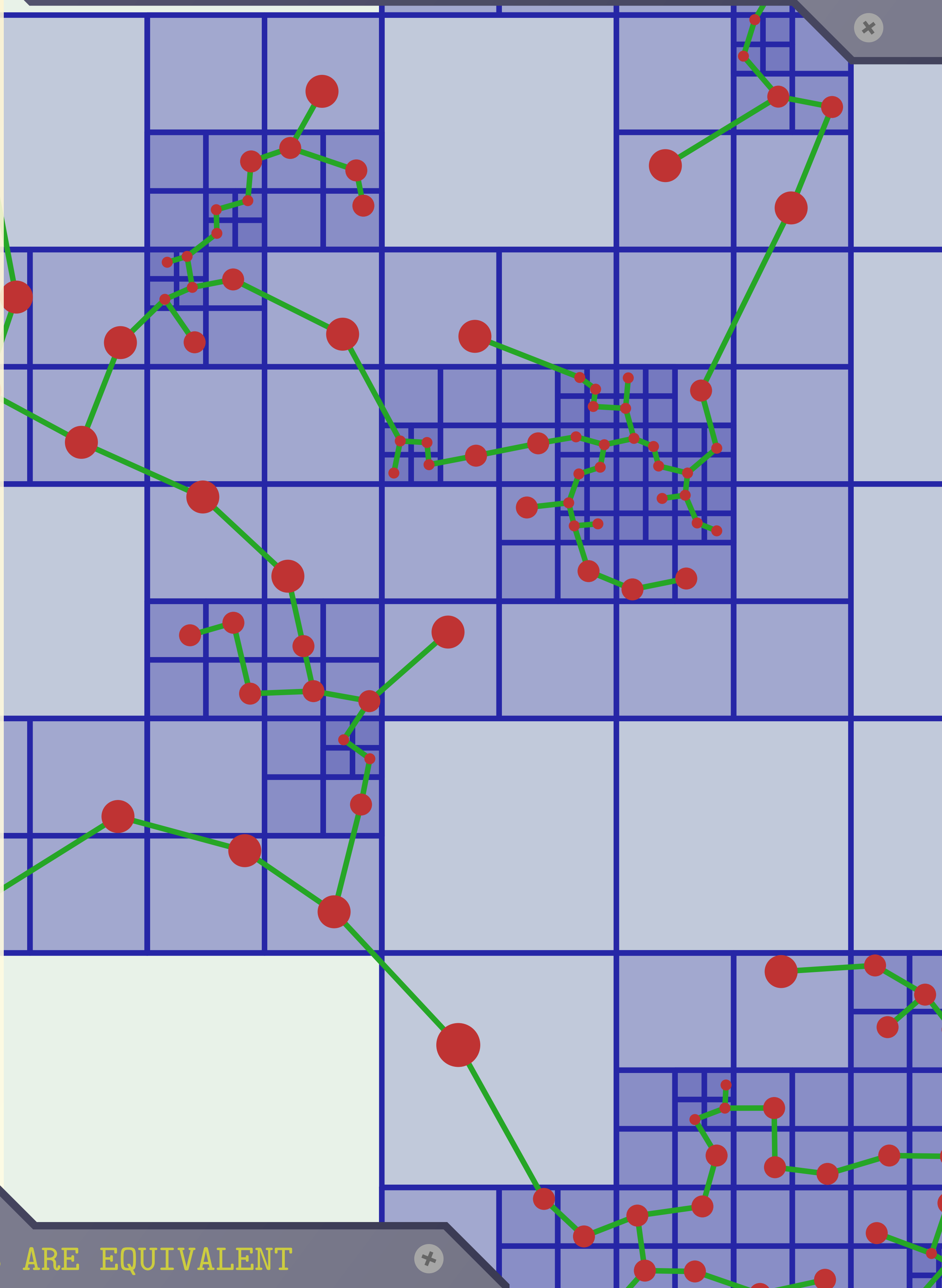


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

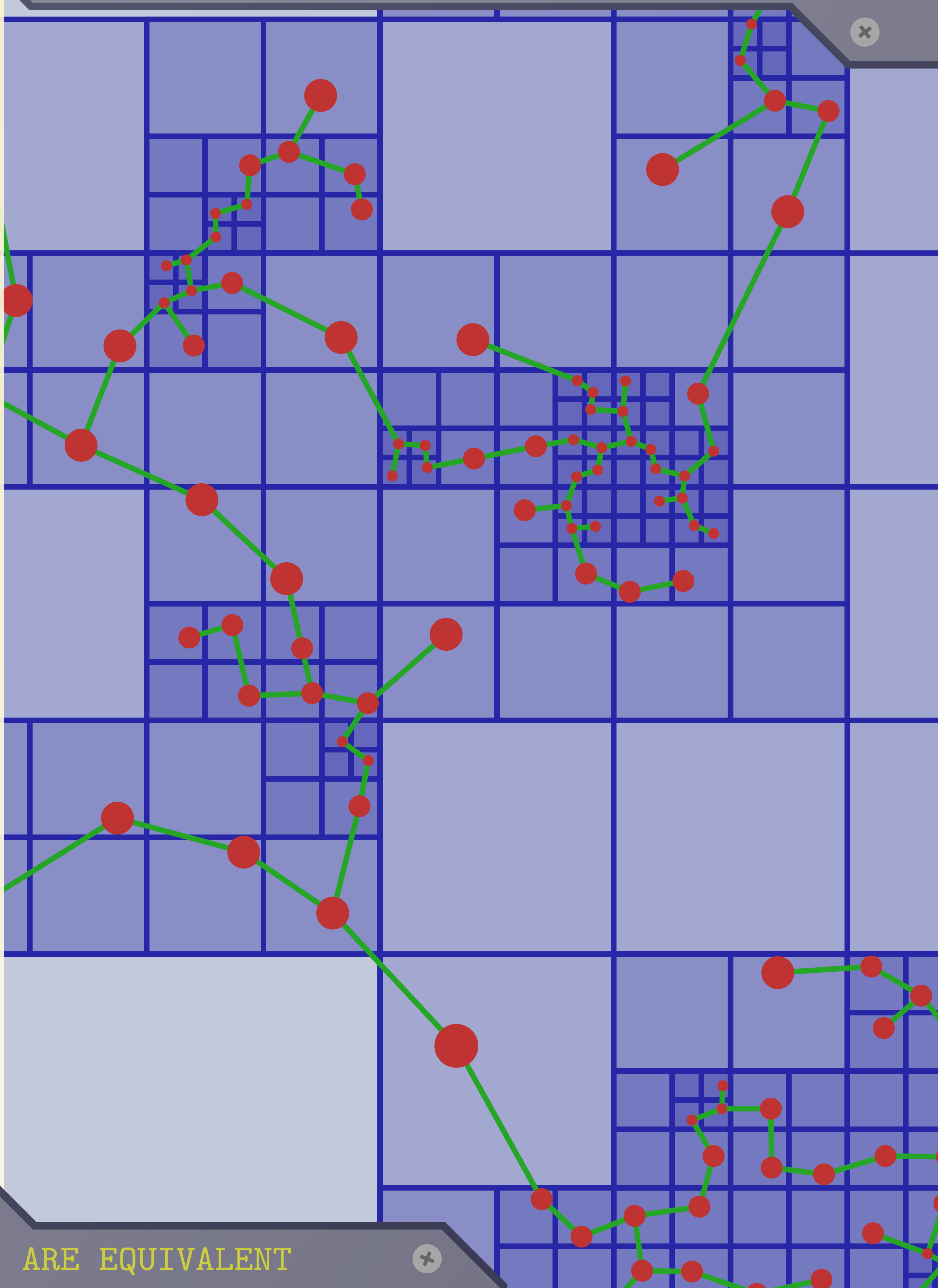


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

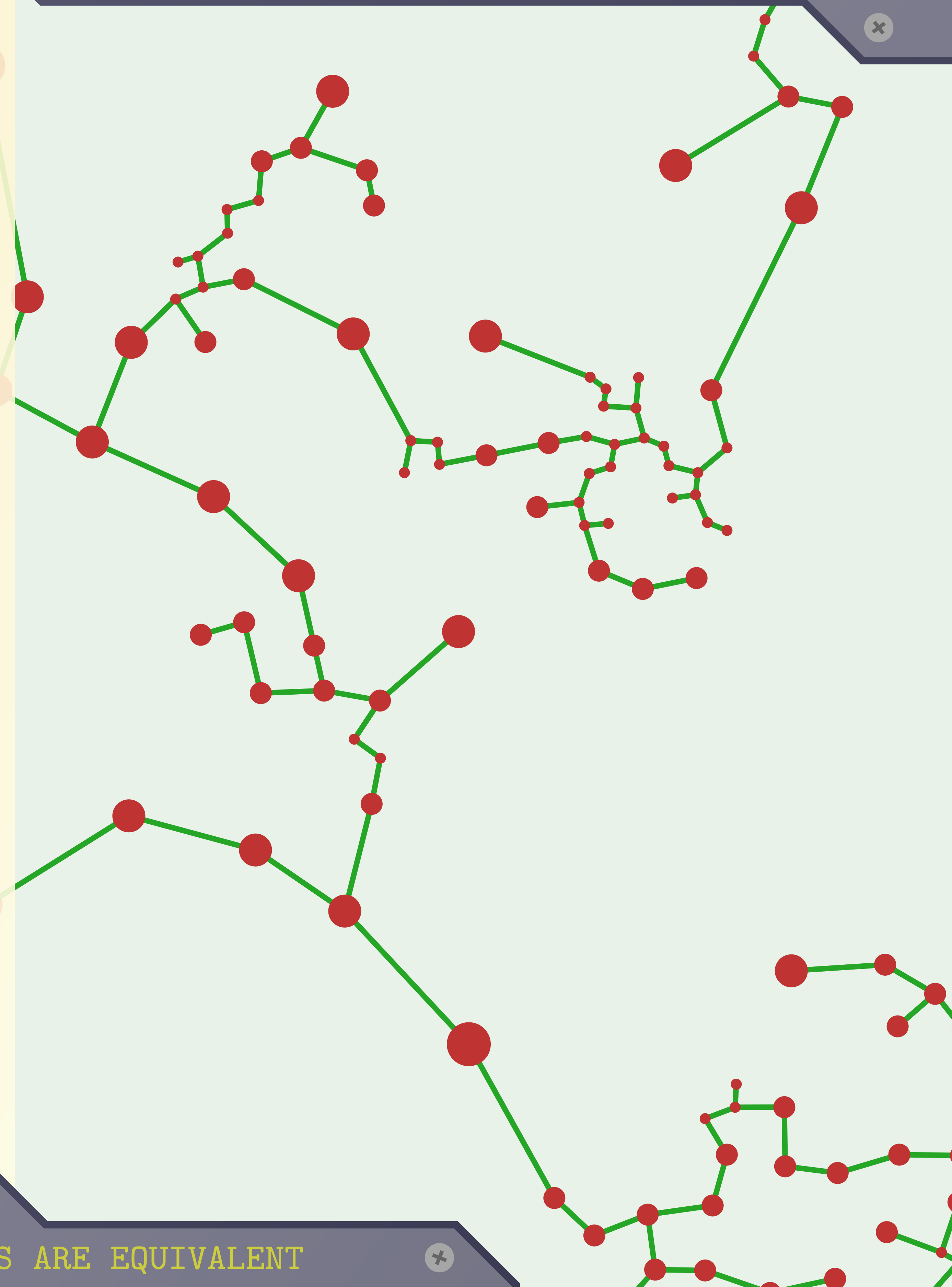


Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.



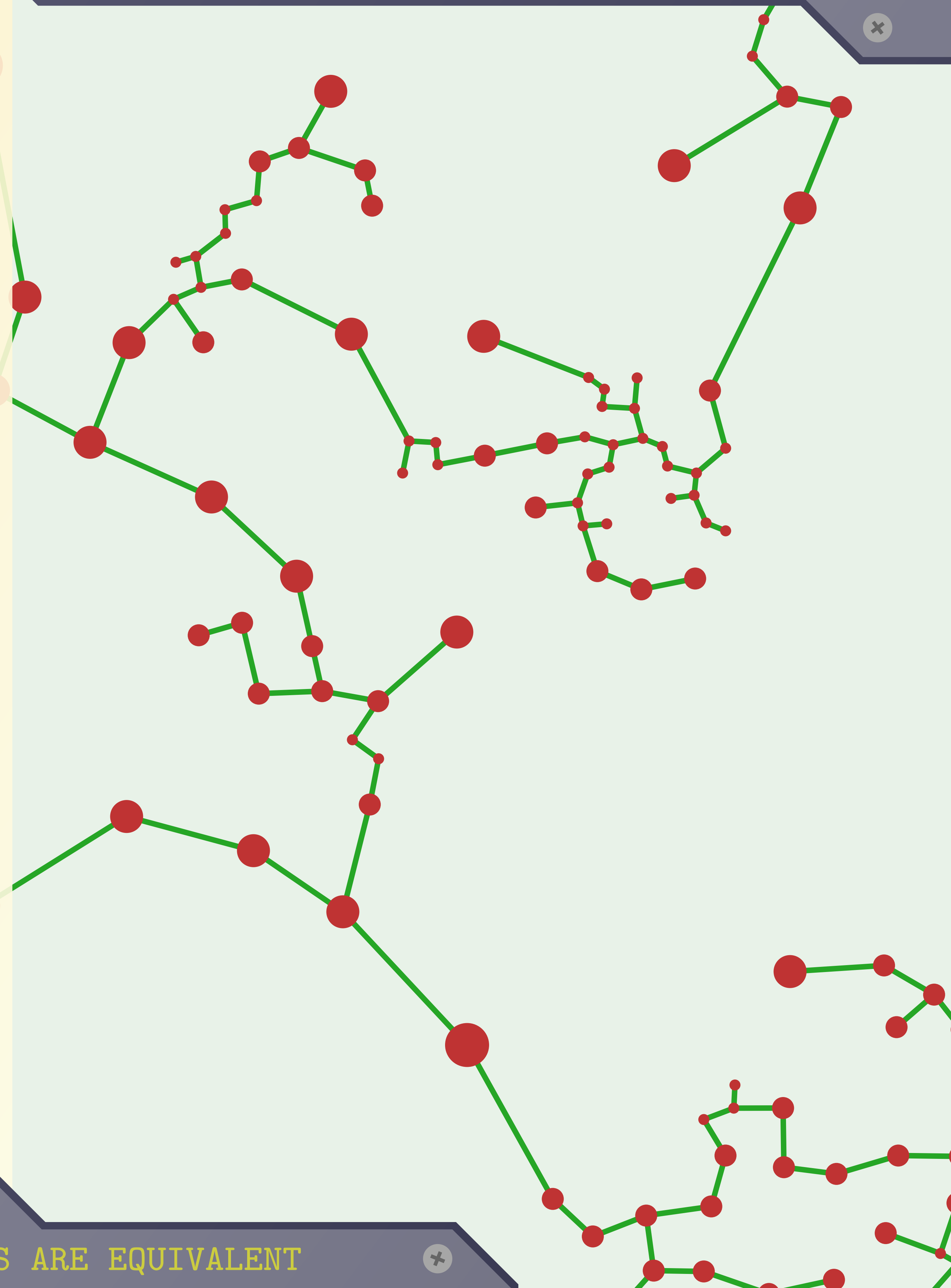
Start with P and G.

Remember we have a quadtree T on P.

We can process the edges of G by increasing length, Borůvka-style.

Ignore edges within components.

Done!

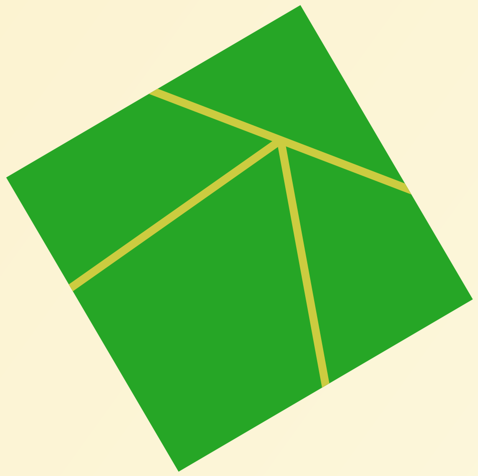


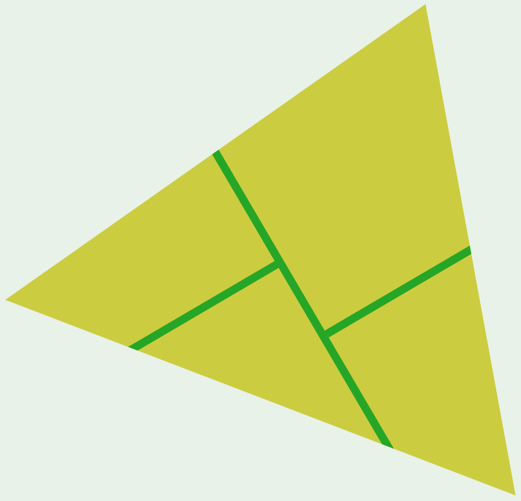
Given a quadtree on a set of points P, we can compute the MST of P in linear time.

Given a quadtree on a set of points P, we can compute the MST of P in linear time.

Many major proximity structures on planar point sets can be derived from each other in linear deterministic time.

THANK *YOU!*





ANY QUE*STIONS?*