Computational Geometry and Topology
Géométrie et topologie algorithmiques

Steve OUDOT *Pooran MEMARI*

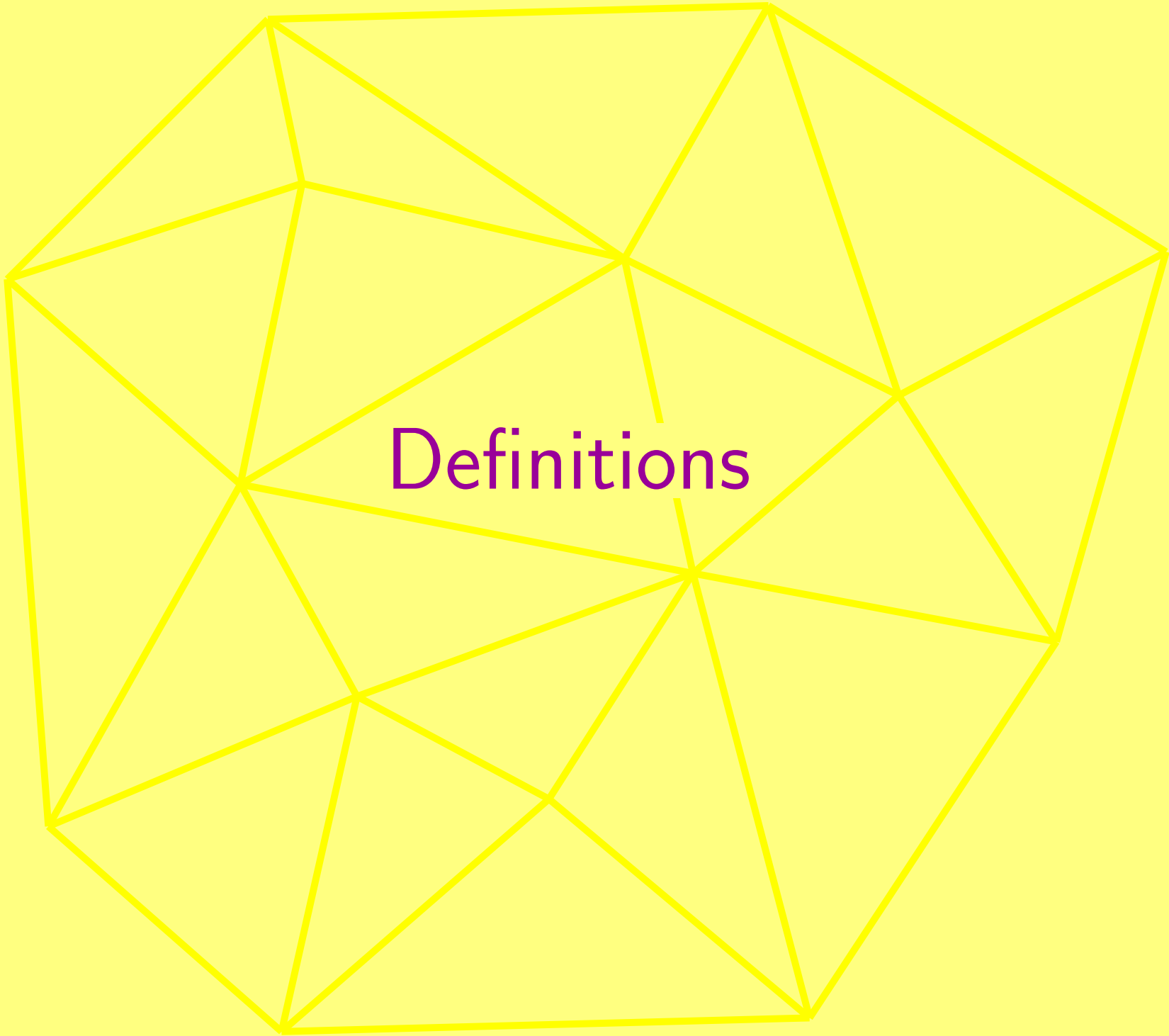
Acknowledgments of Involved Colleagues:

Jean Daniel Boissonnat, Frederic Chazal, Marc Glisse,
As well as Olivier Devillers and Luca Castelli

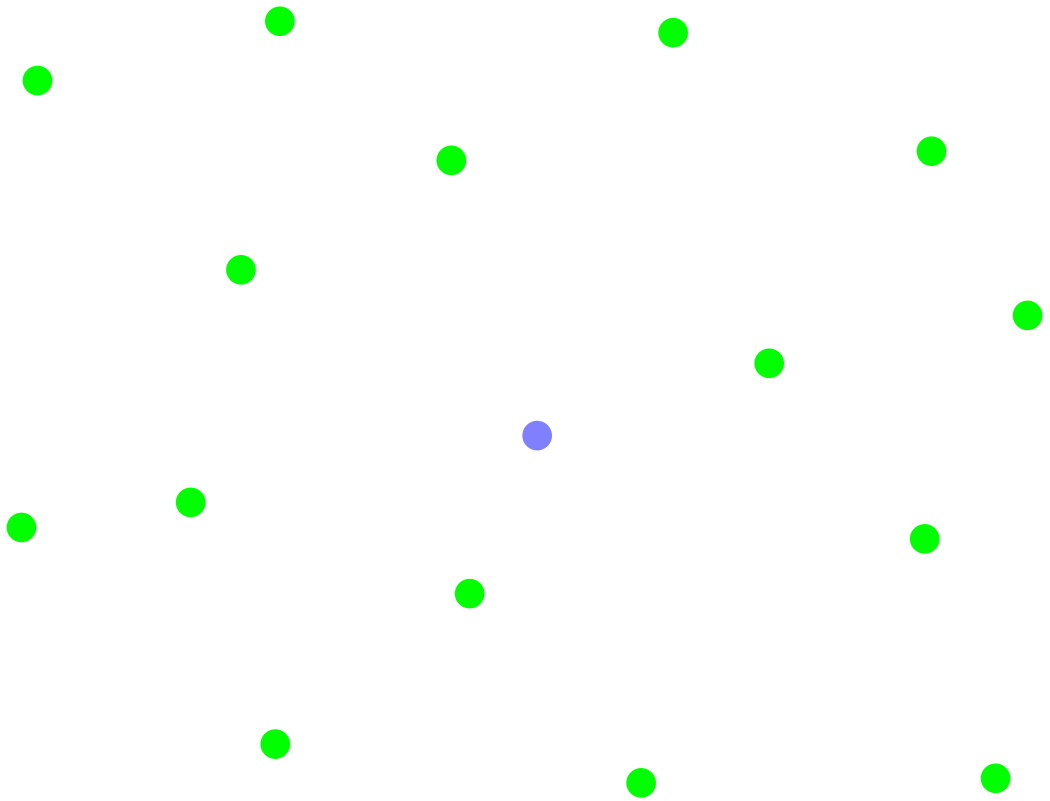
This lecture: slides courtesy of
Olivier Devillers and Jean Daniel Boissonnat

Outline

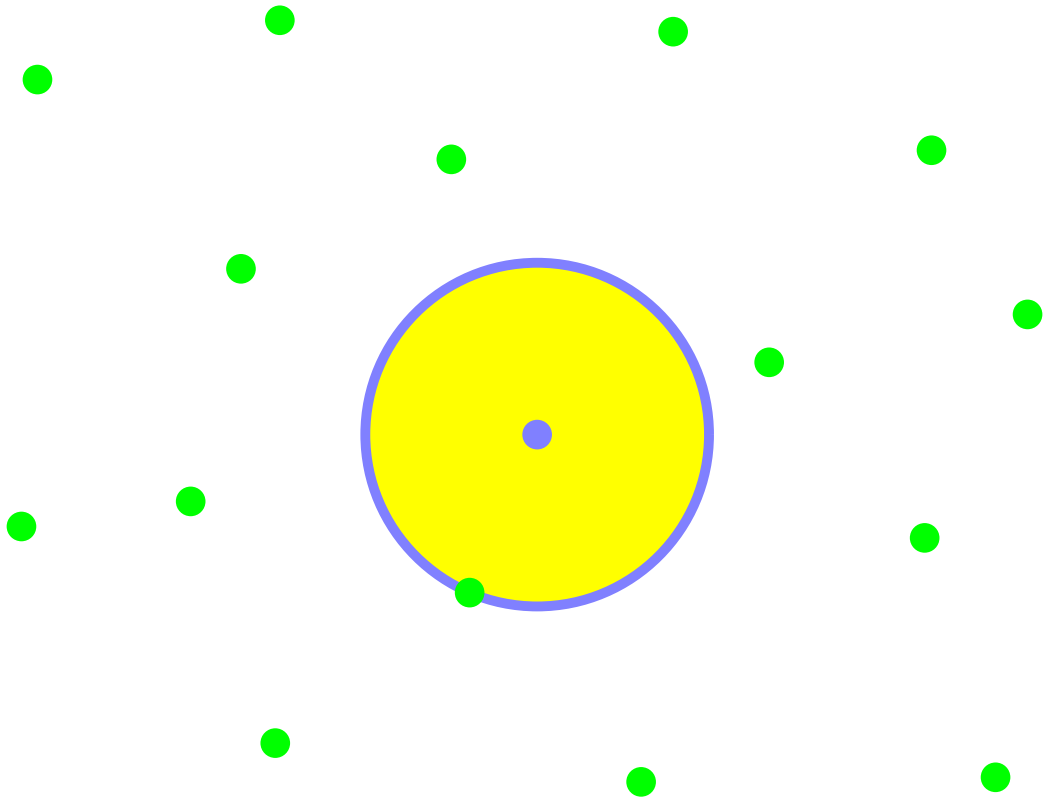
1. Definitions and examples
2. Structral properties and applications
3. Size
4. Construction
5. Generalizations


Definitions

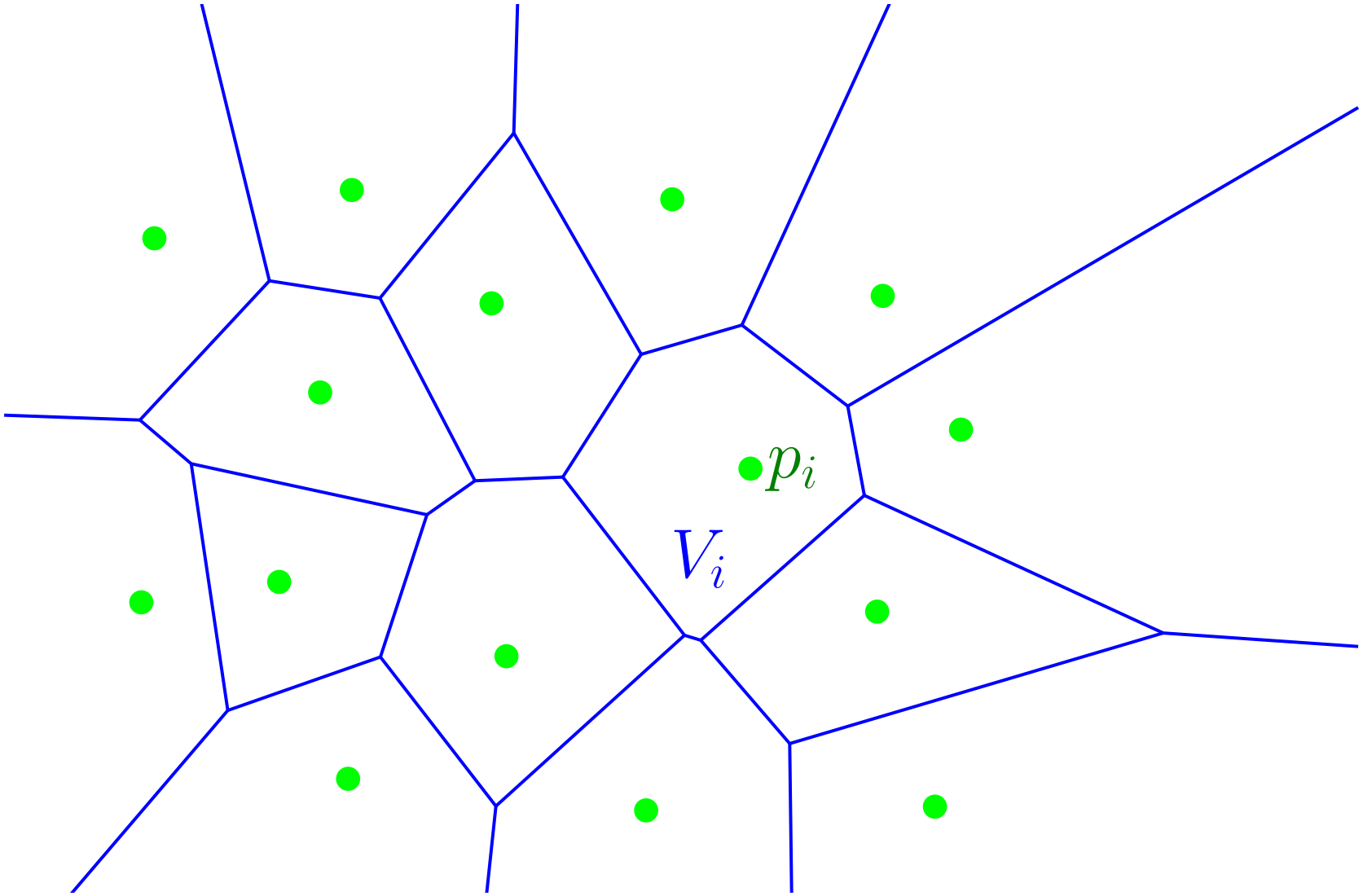
looking for nearest neighbor



looking for nearest neighbor

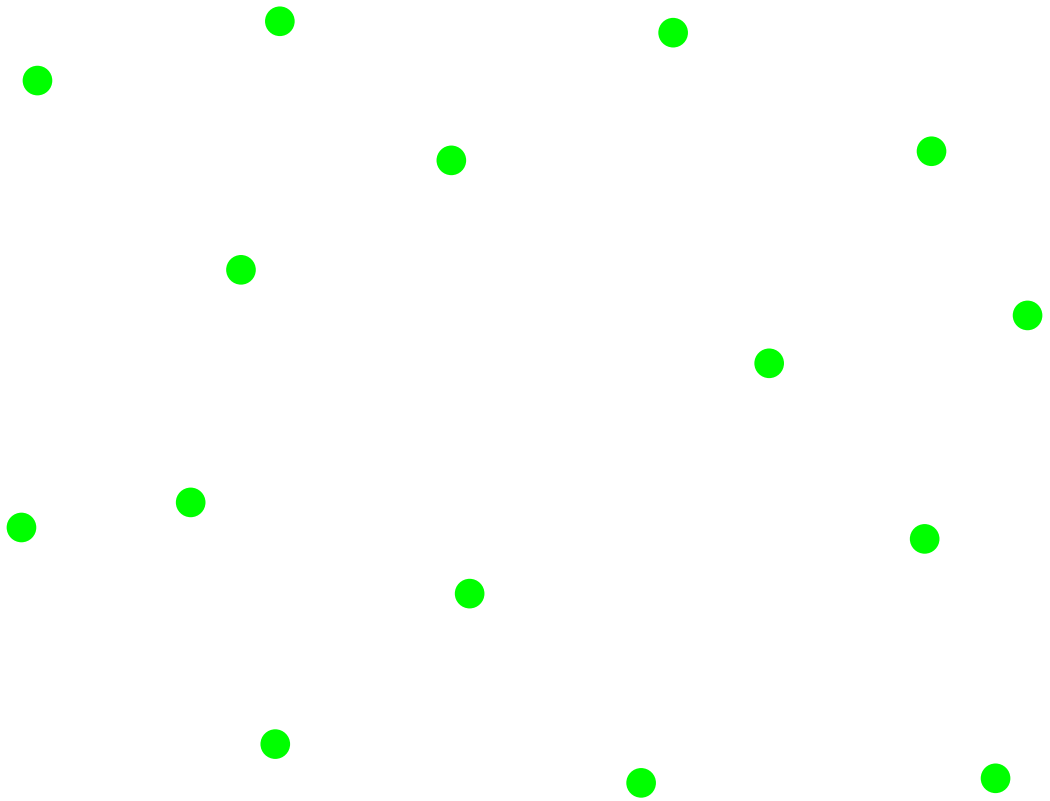


Voronoi diagram of $\{p_1, \cdots, p_n\} \subset \mathbb{R}^d$



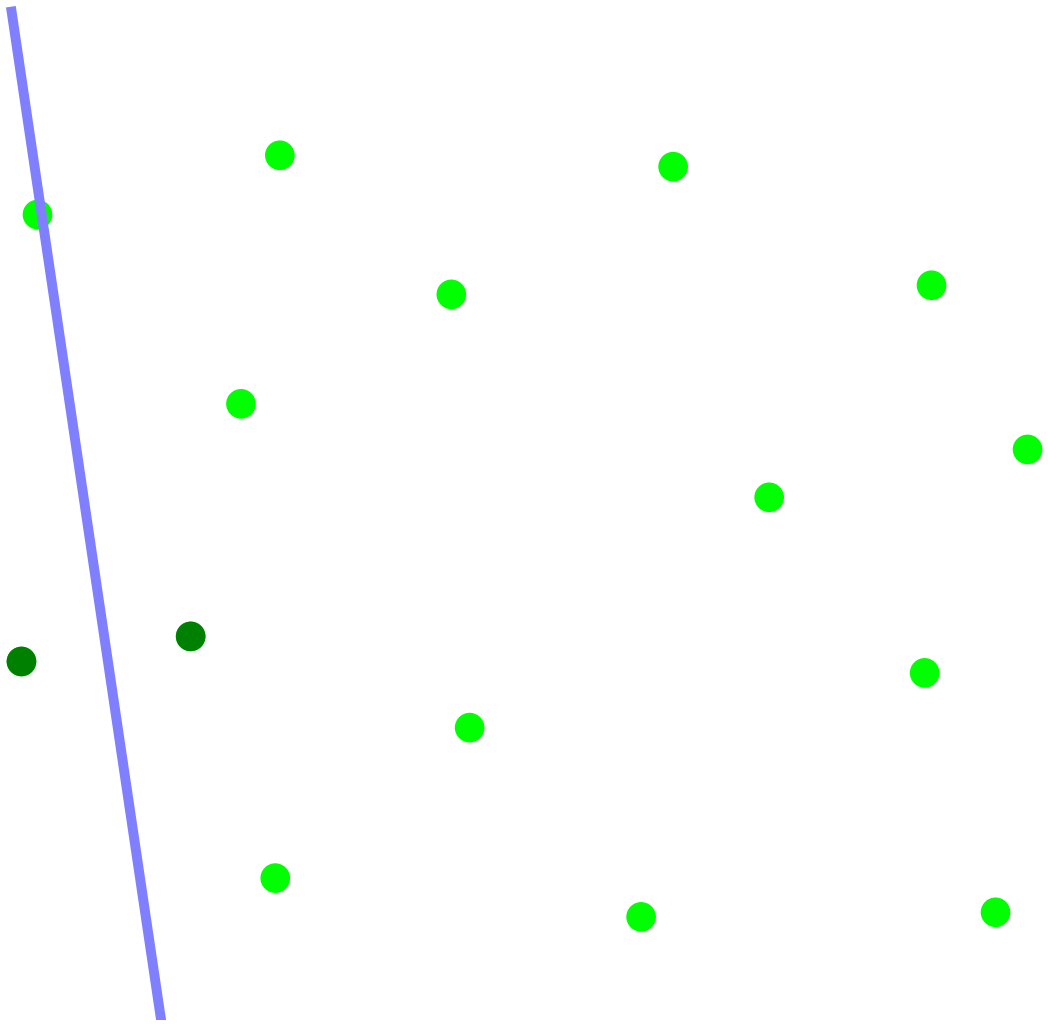
$$V_i := \{q \in \mathbb{R}^d \mid \|q - p_i\| \leq \|q - p_j\| \ \forall j\}$$

Voronoi



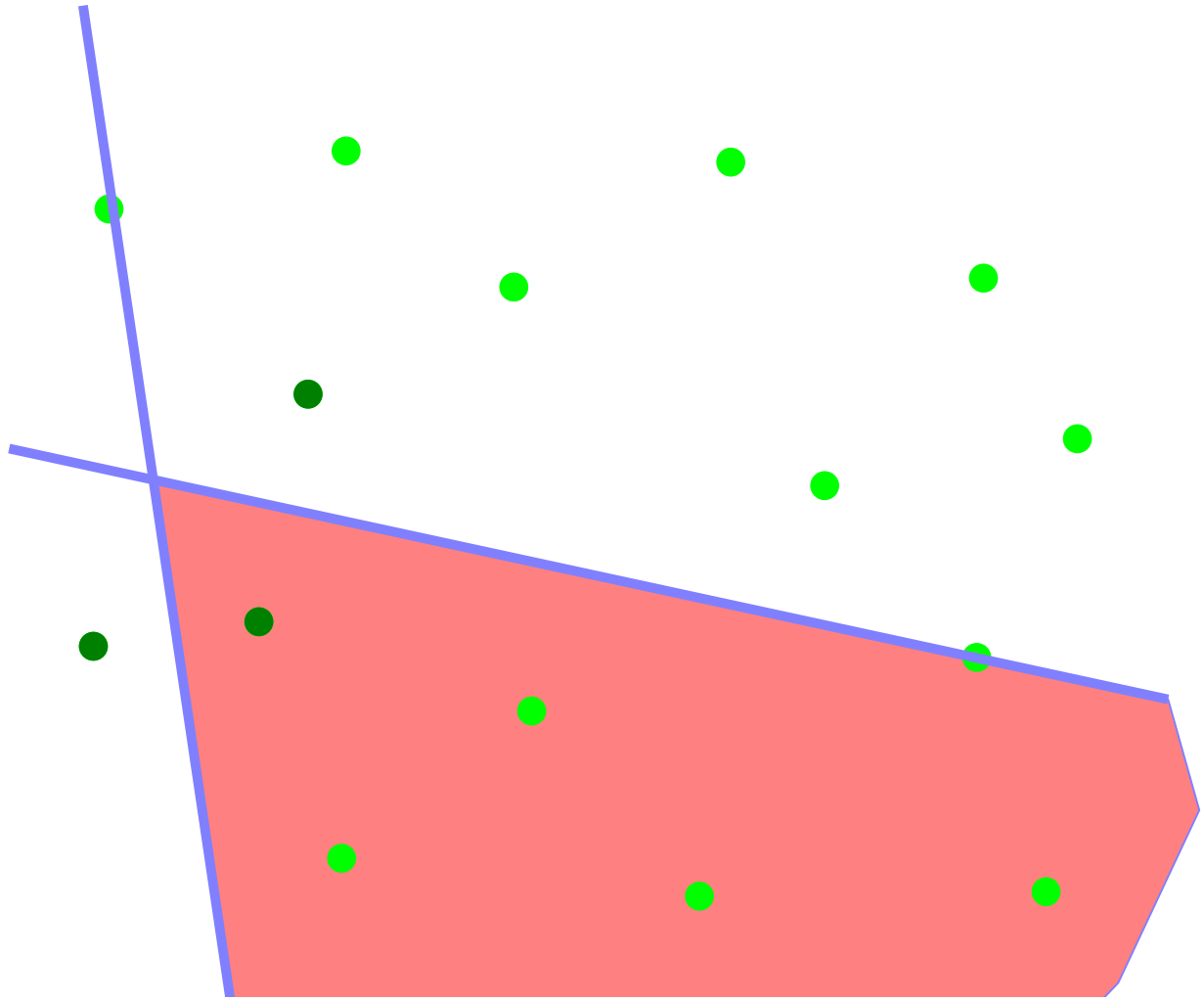
faces of the Voronoi diagram

Voronoi



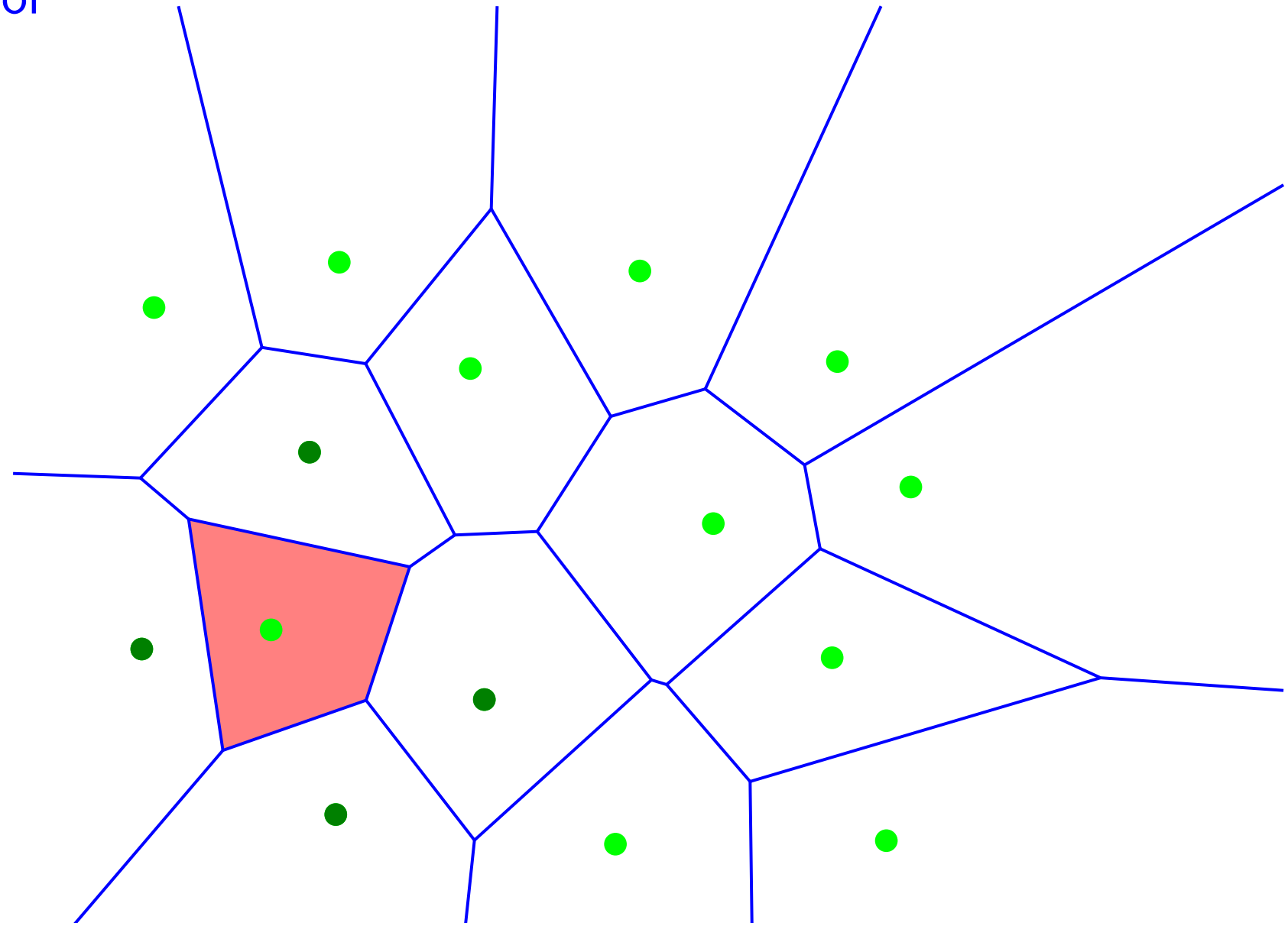
faces of the Voronoi diagram

Voronoi



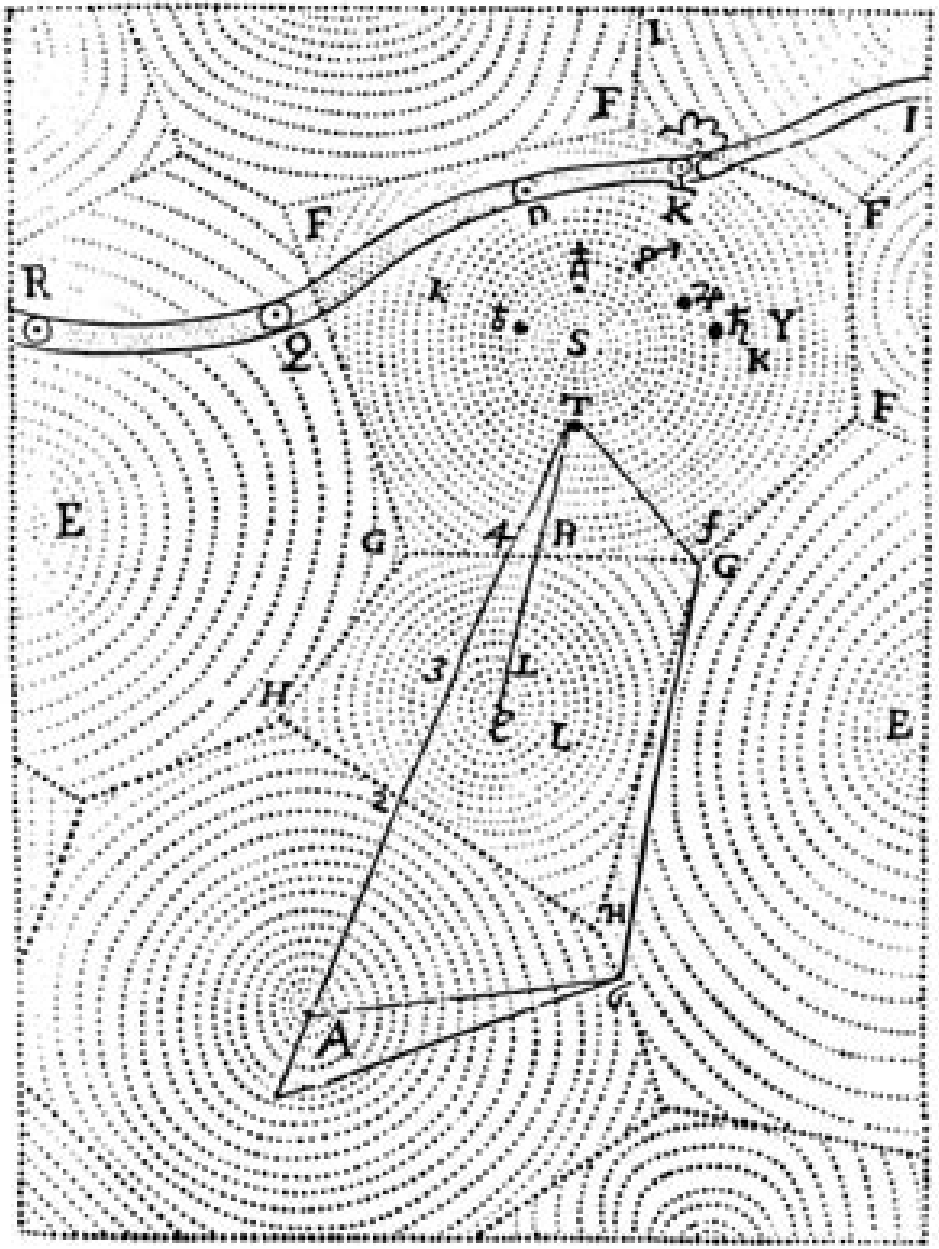
faces of the Voronoi diagram

Voronoi

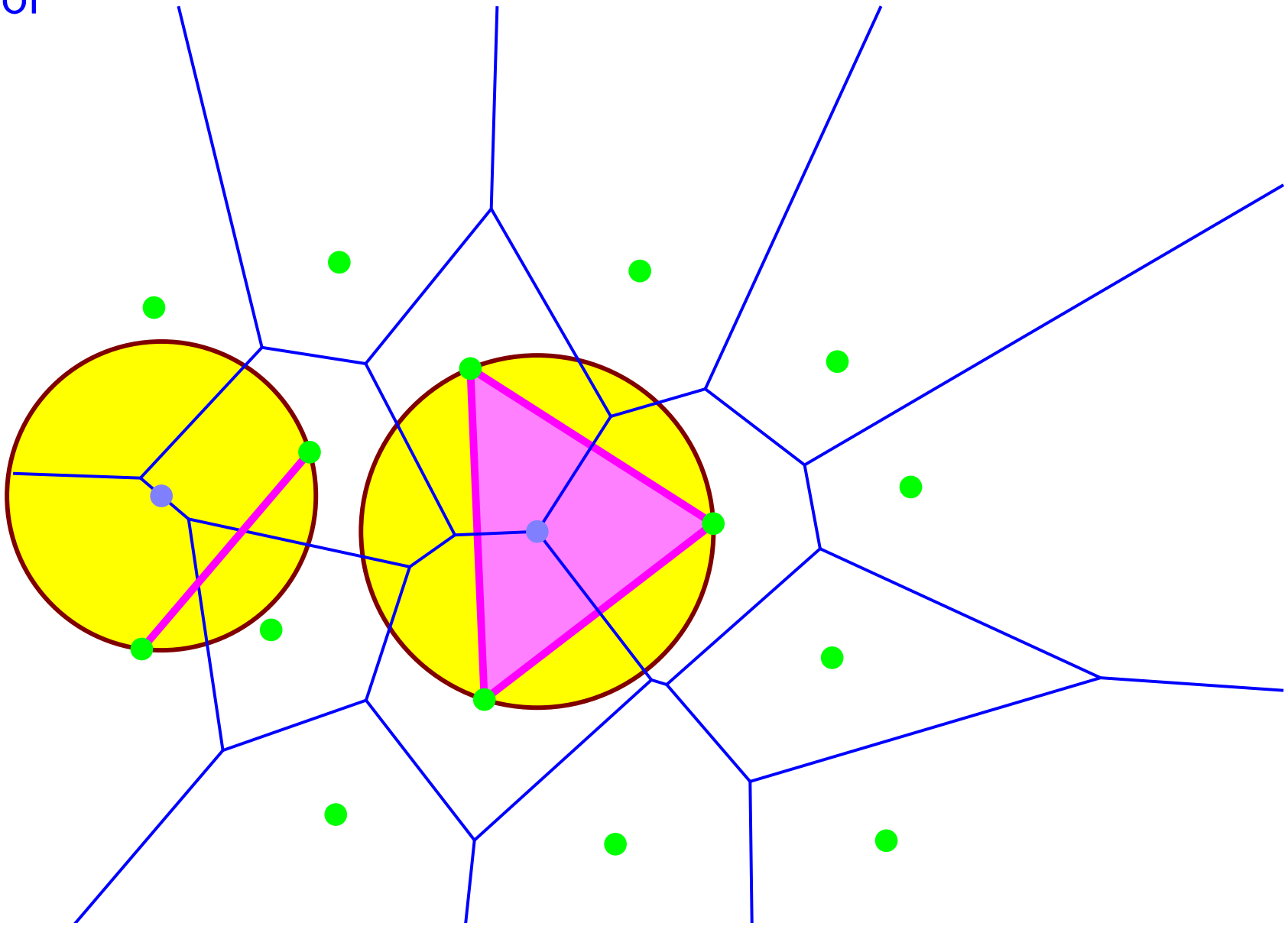


faces of the Voronoi diagram

Voronoi is everywhere



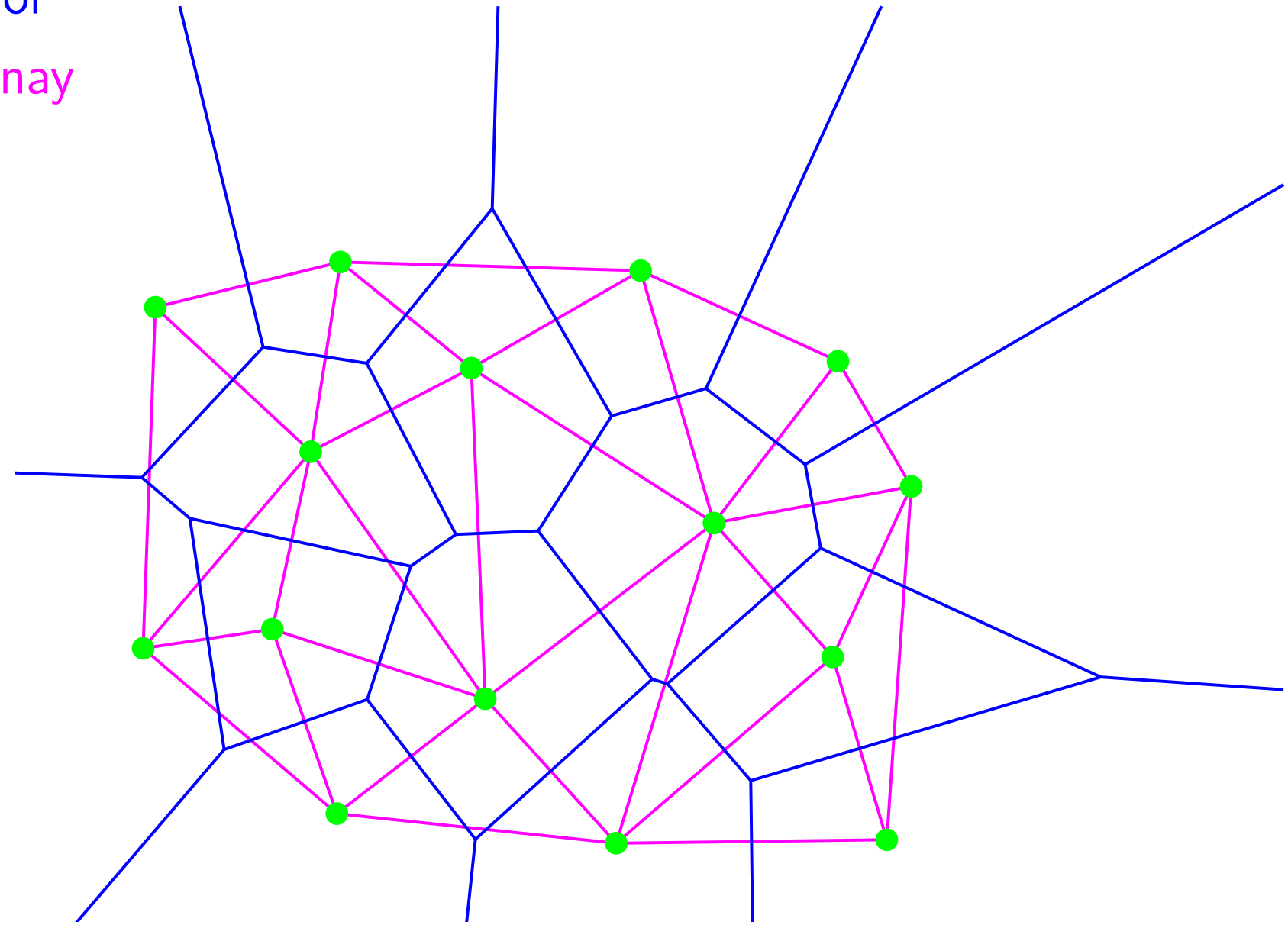
Voronoi



Empty sphere property

Voronoi

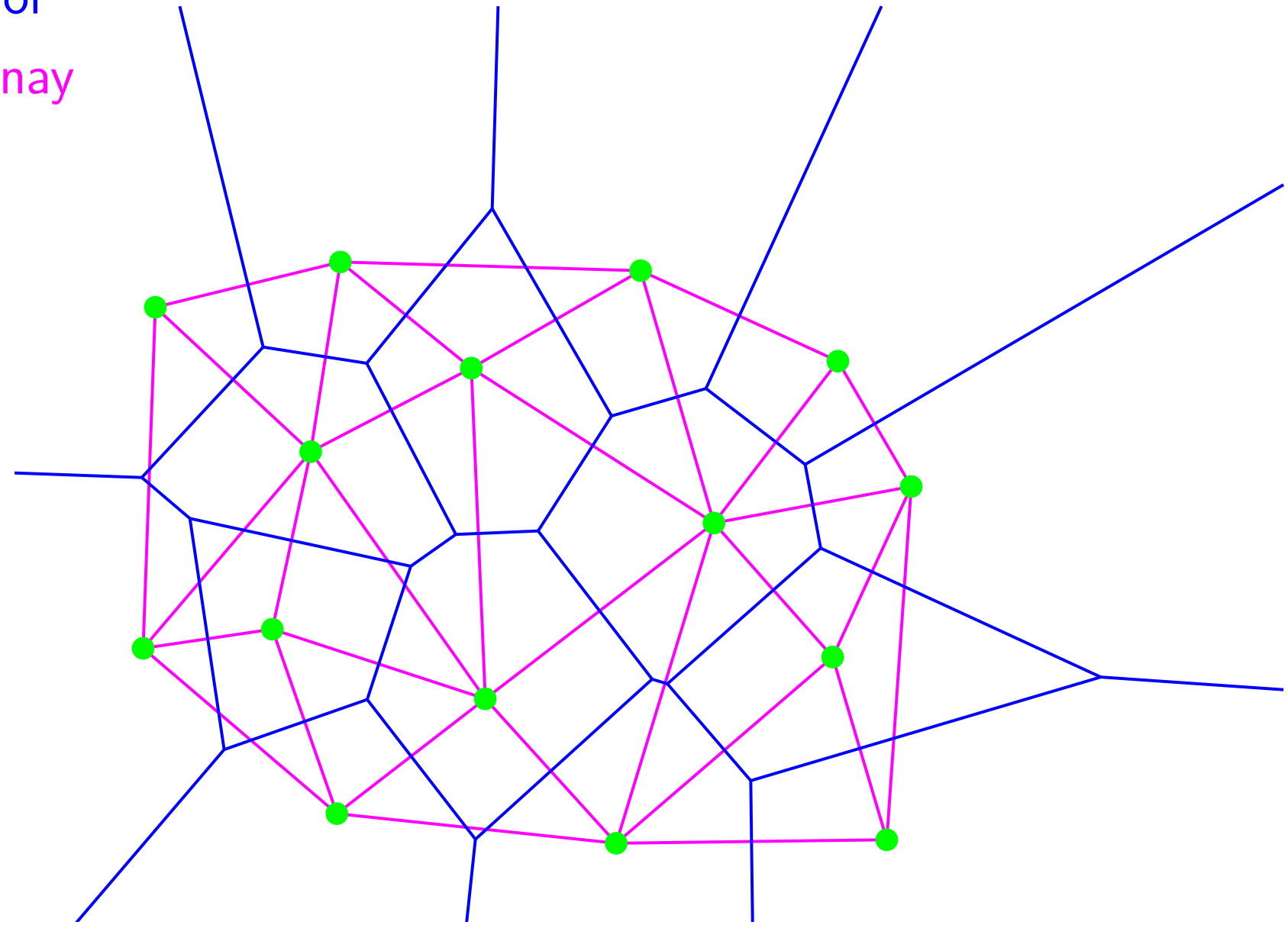
Delaunay



Nerve: $\{p_0, \cdots, p_k\} \in \mathrm{Del}(P) \Leftrightarrow V_0 \cap \cdots \cap V_k \neq \emptyset$

Voronoi

Delaunay



Voronoi $\leftrightarrow$ geometry

Delaunay $\leftrightarrow$ connectivity (nerve)

Geometric simplicial complexes

vertex set: $V = \{v_0, v_1, \ldots, v_{n-1}\} \subset \mathbb{R}^d$

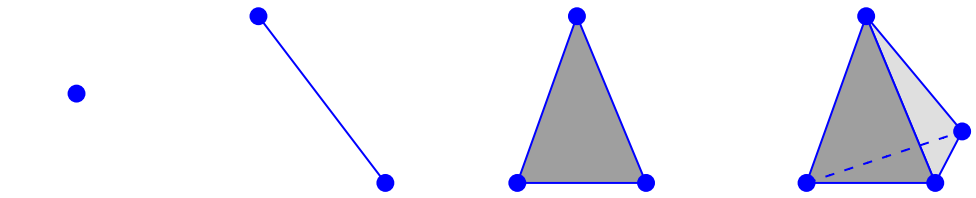
k-simplex: $\sigma = \mathrm{Conv}\{v_{i_0}, v_{i_1}, \cdots, v_{i_k}\}$

inclusion property (τ *face* of σ):

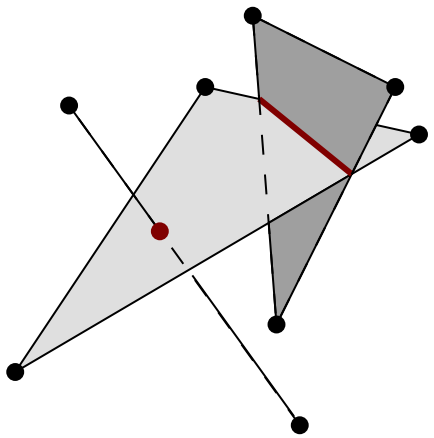
$\sigma \in K$ and $V(\tau) \subseteq V(\sigma) \Longrightarrow \tau \in K$

intersection property:

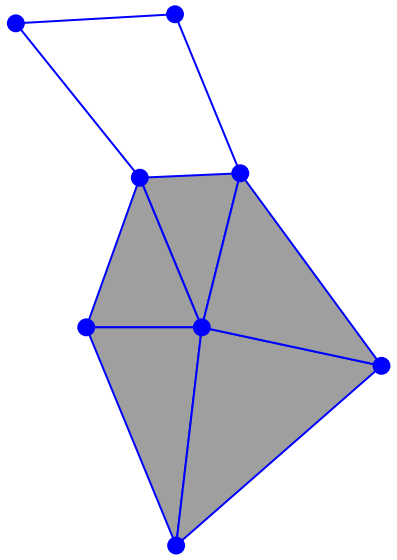
$\sigma_1, \sigma_2 \in K$ and $\sigma_1 \cap \sigma_2 \neq \emptyset \Longrightarrow$
$\sigma_1 \cap \sigma_2 \in K$ and is a face of both



0-simplex (vertex) 1-simplex (edge) 2-simplex (triangle) 3-simplex (tetrahedron)



invalid simplicial complex



valid simplicial complex

Geometric simplicial complexes

vertex set: $V = \{v_0, v_1, \ldots, v_{n-1}\} \subset \mathbb{R}^d$

k-simplex: $\sigma = \mathrm{Conv}\{v_{i_0}, v_{i_1}, \cdots, v_{i_k}\}$

inclusion property (τ *face* of σ):

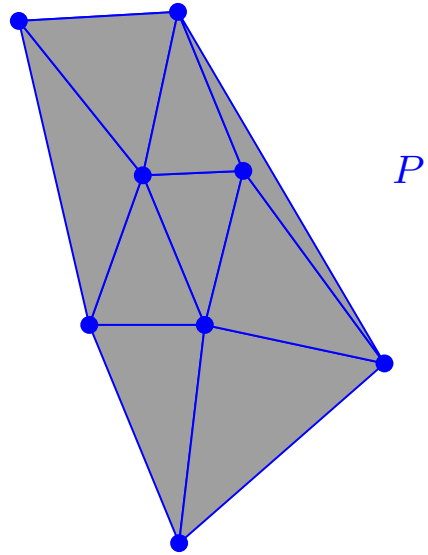
$\sigma \in K$ and $V(\tau) \subseteq V(\sigma) \Longrightarrow \tau \in K$

intersection property:

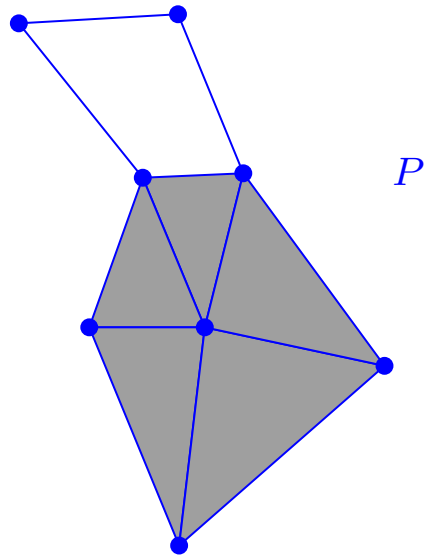
$\sigma_1, \sigma_2 \in K$ and $\sigma_1 \cap \sigma_2 \neq \emptyset \Longrightarrow$
$\sigma_1 \cap \sigma_2 \in K$ and is a face of both

triangulation of P:

simplicial complex T with vertex set P such that $\bigcup_{\sigma \in T} \sigma = \mathrm{Conv}\, P$

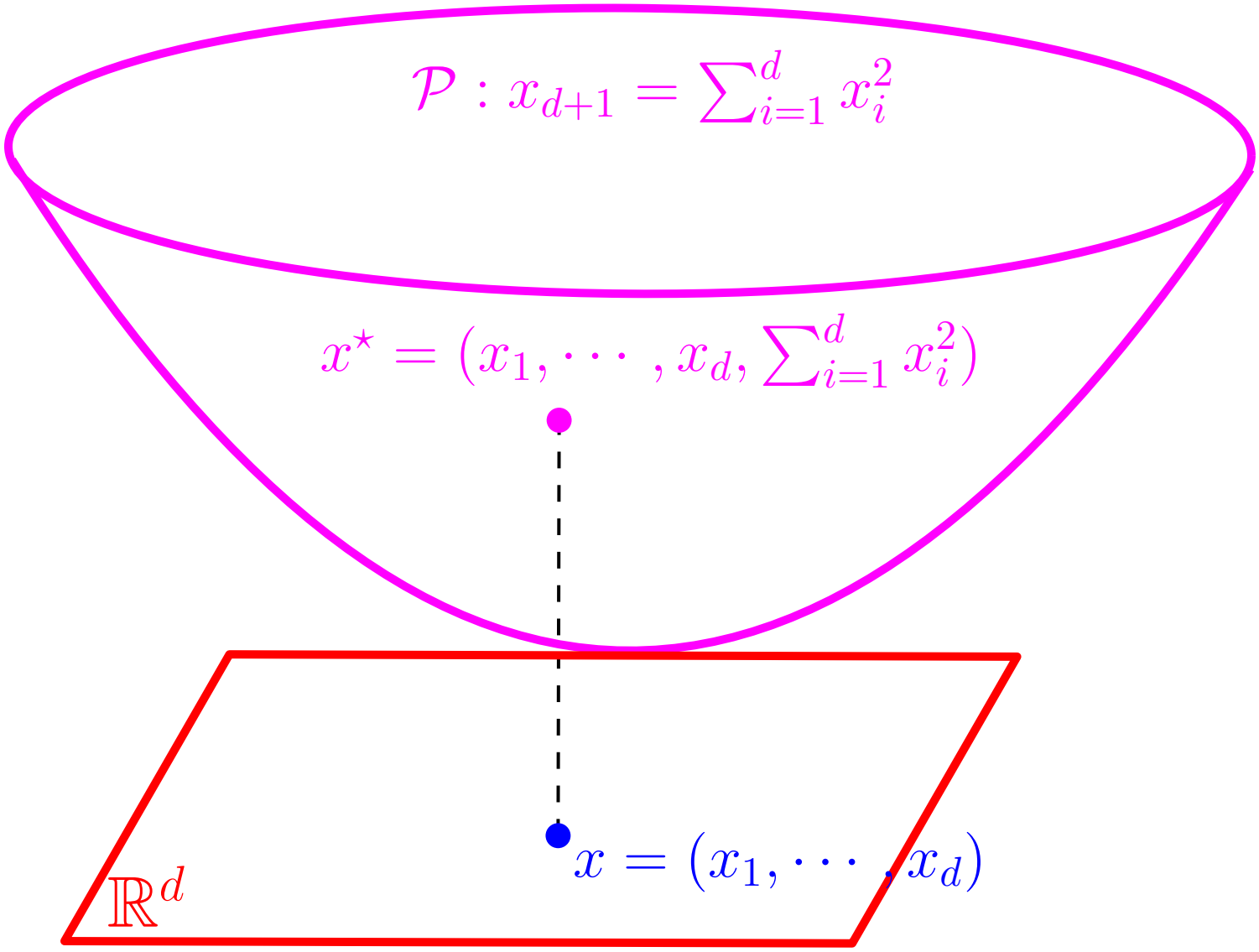


valid triangulation of P

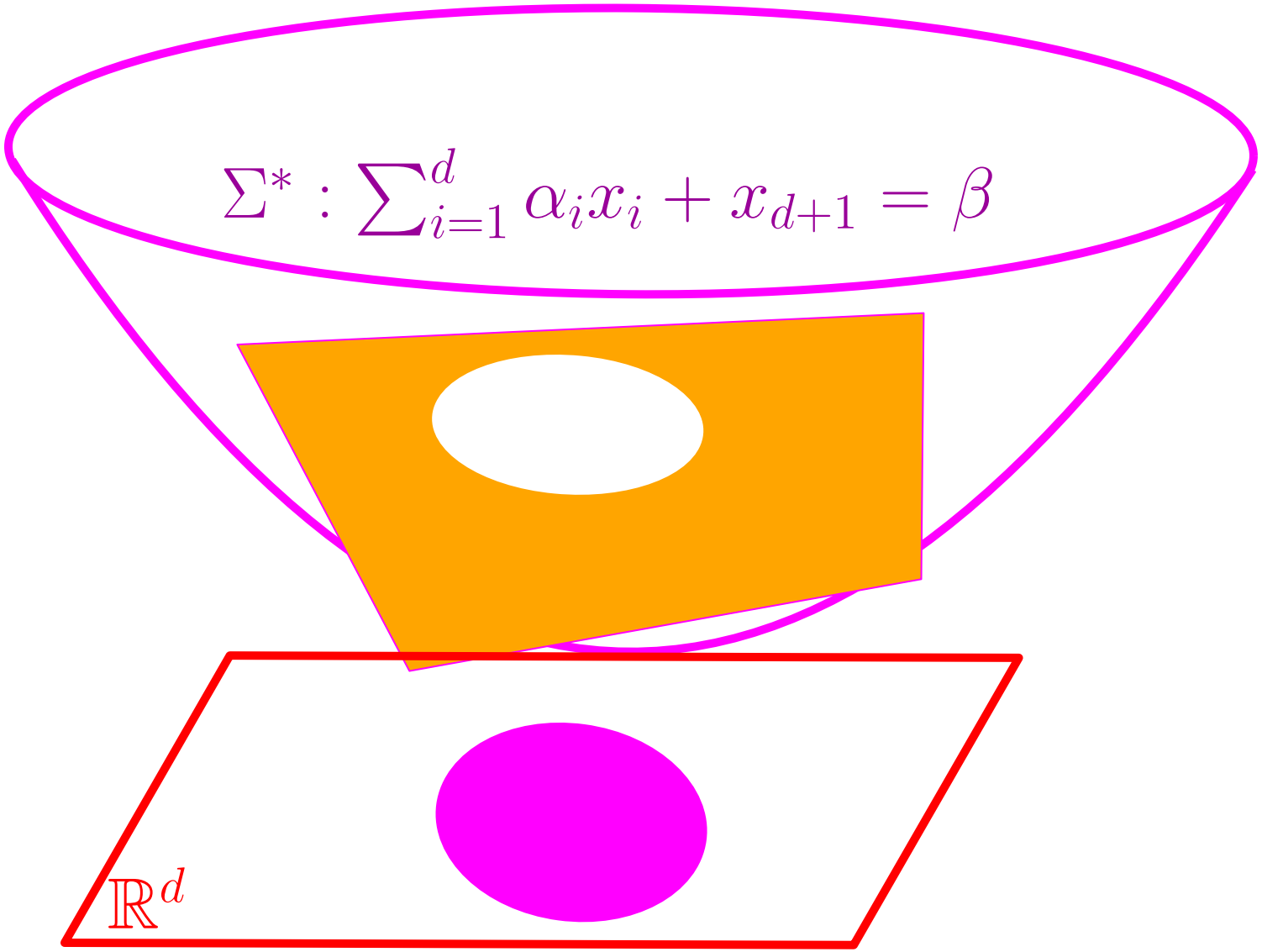


invalid triangulation of P

point / sphere lifting

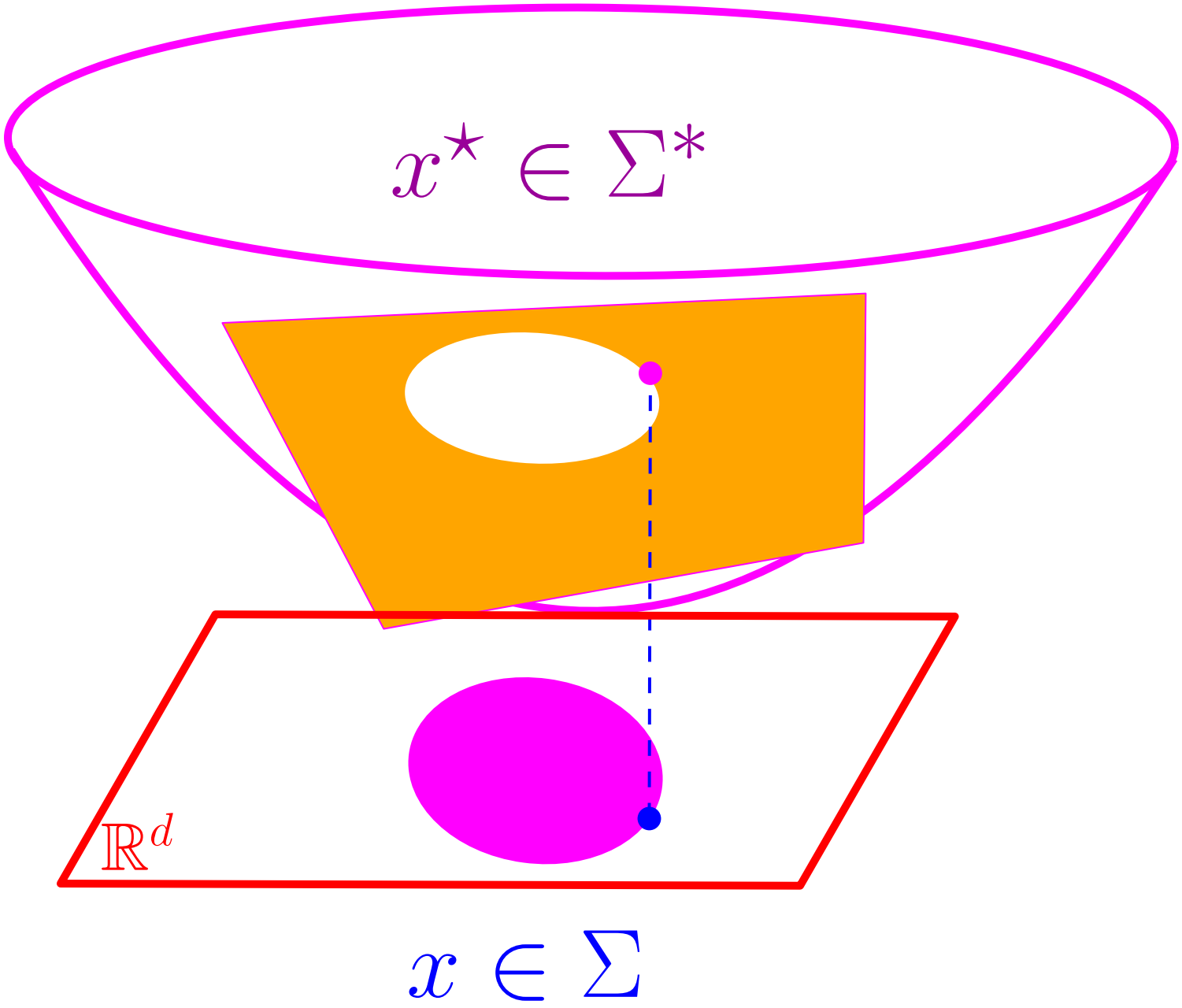


point / sphere lifting

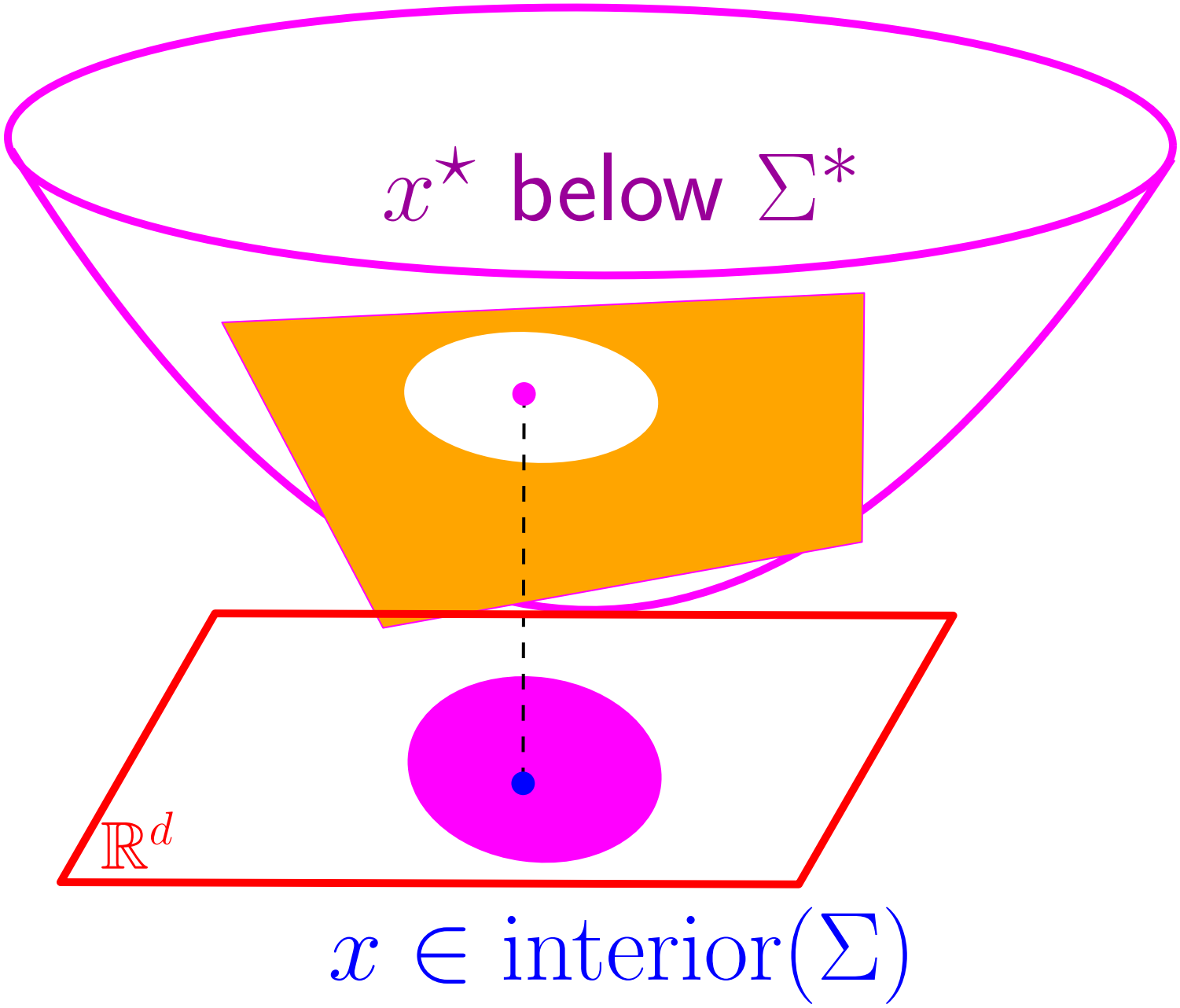


$$\Sigma : x^2 - 2x \cdot (\tfrac{-\alpha_1}{2}, \cdots, \tfrac{-\alpha_d}{2}) + (\tfrac{-\alpha_1}{2}, \cdots, \tfrac{-\alpha_d}{2})^2 = \beta + (\tfrac{-\alpha_1}{2}, \cdots, \tfrac{-\alpha_d}{2})^2$$

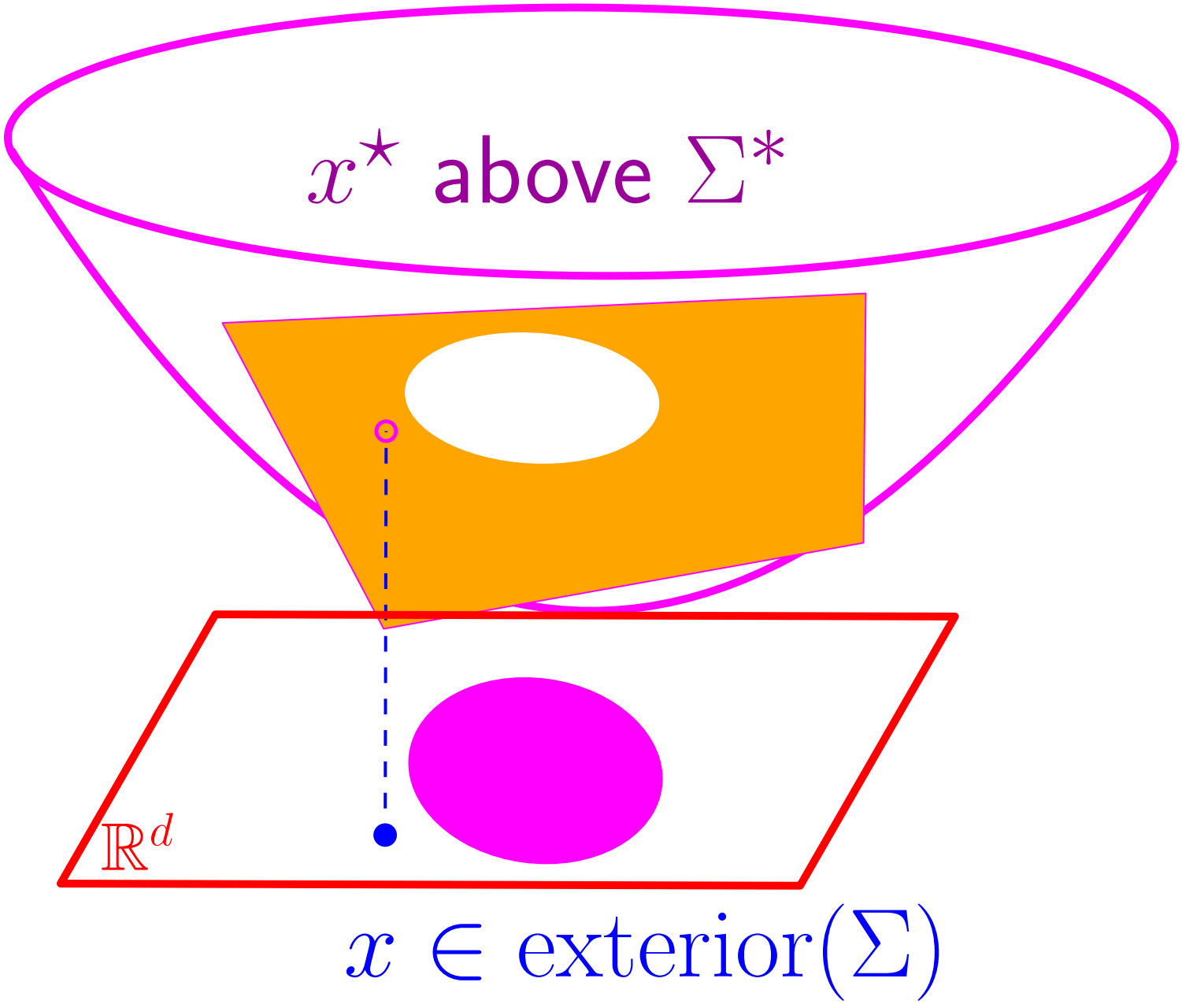
point / sphere lifting



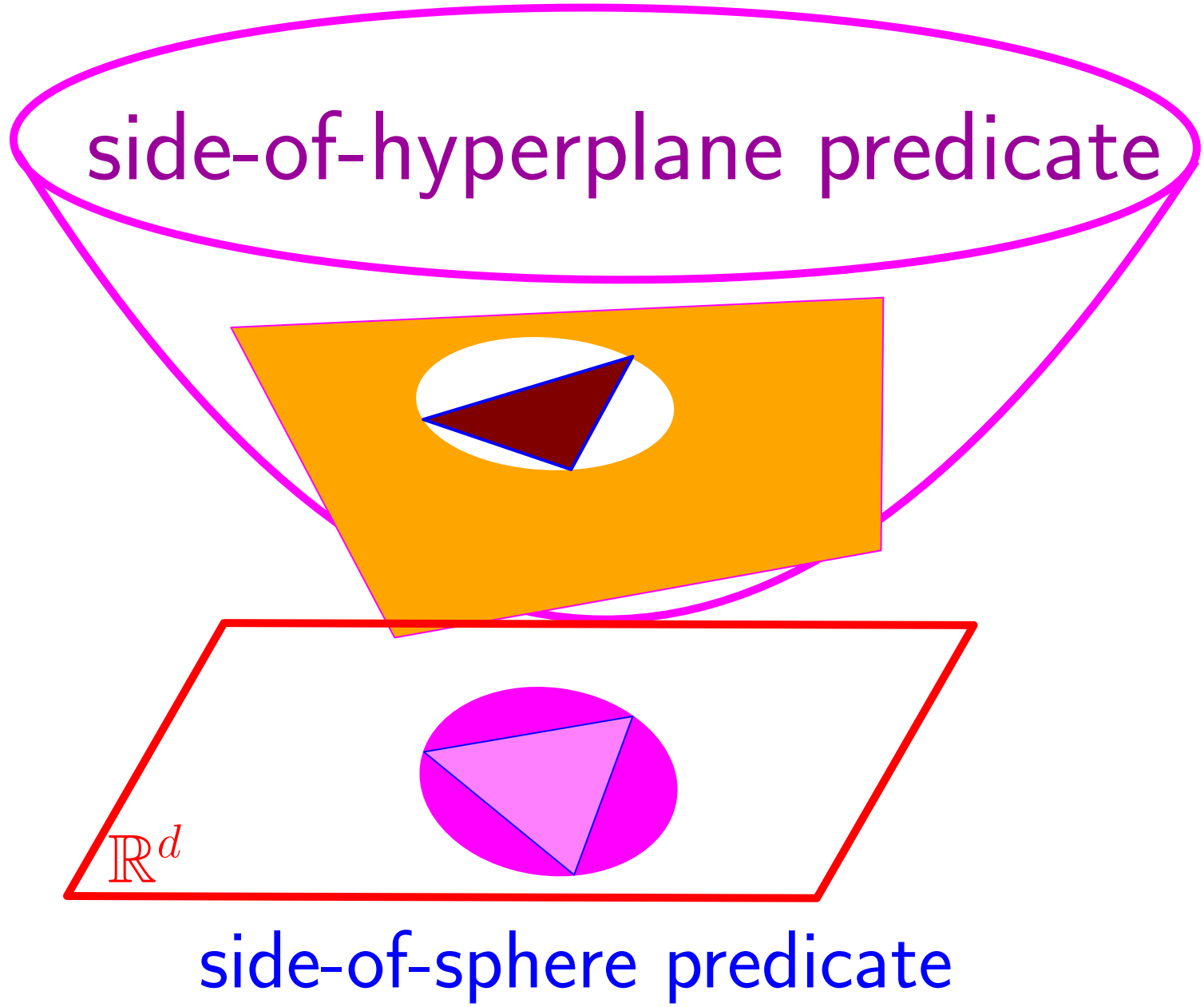
point / sphere lifting



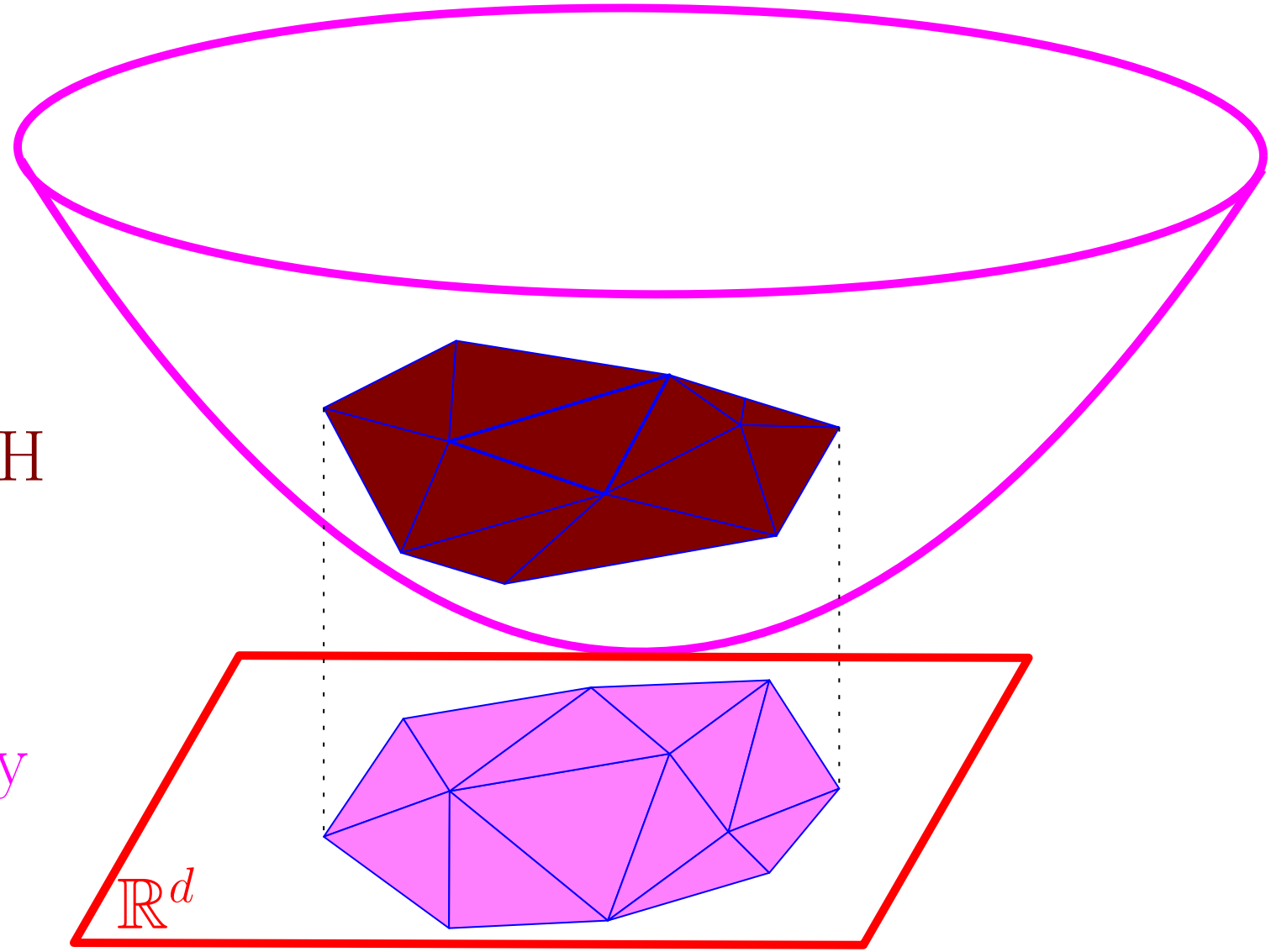
point / sphere lifting



point / sphere lifting



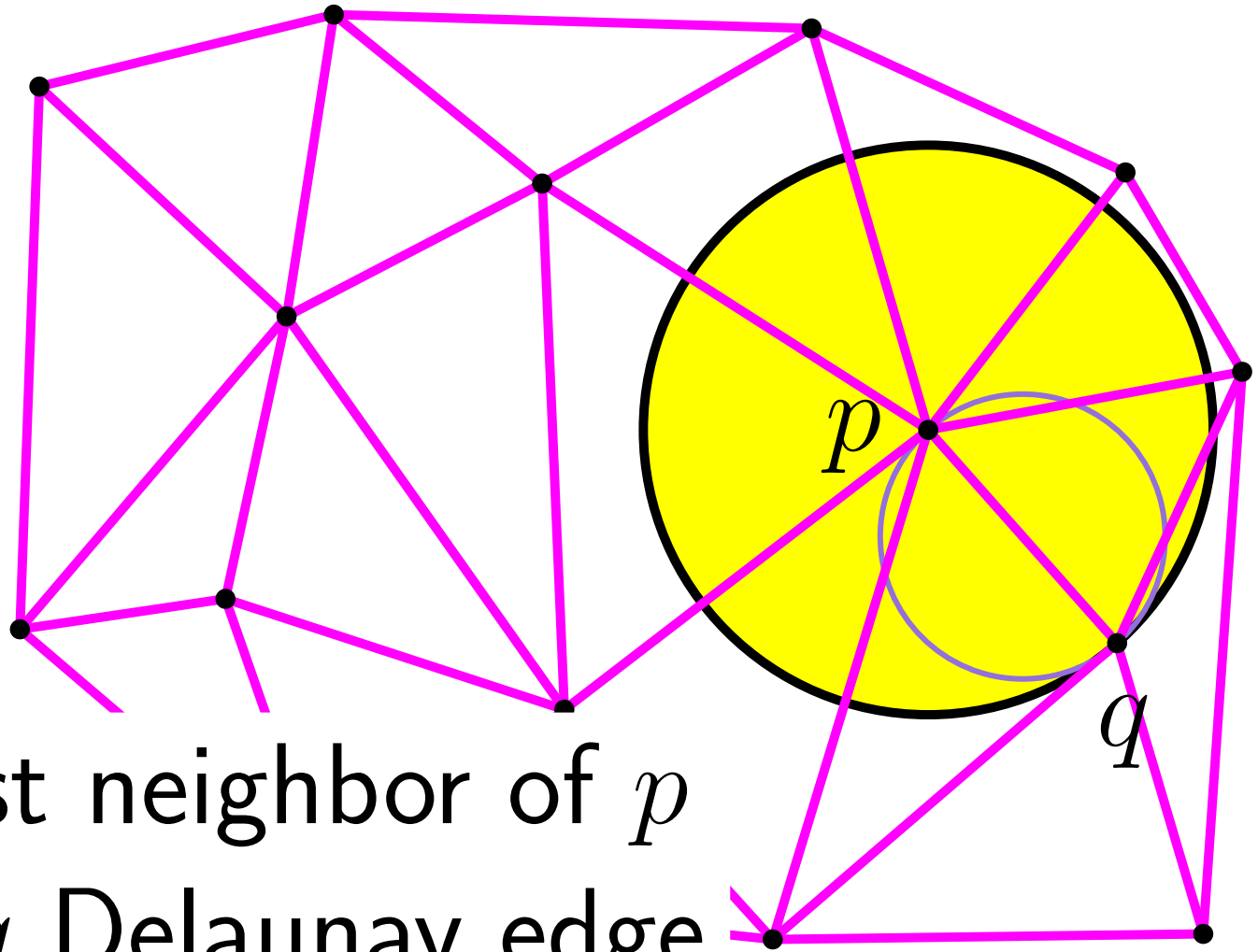
point / sphere lifting



$\Rightarrow$ Delaunay is generically a triangulation (not an abstract complex)

Basic properties and applications

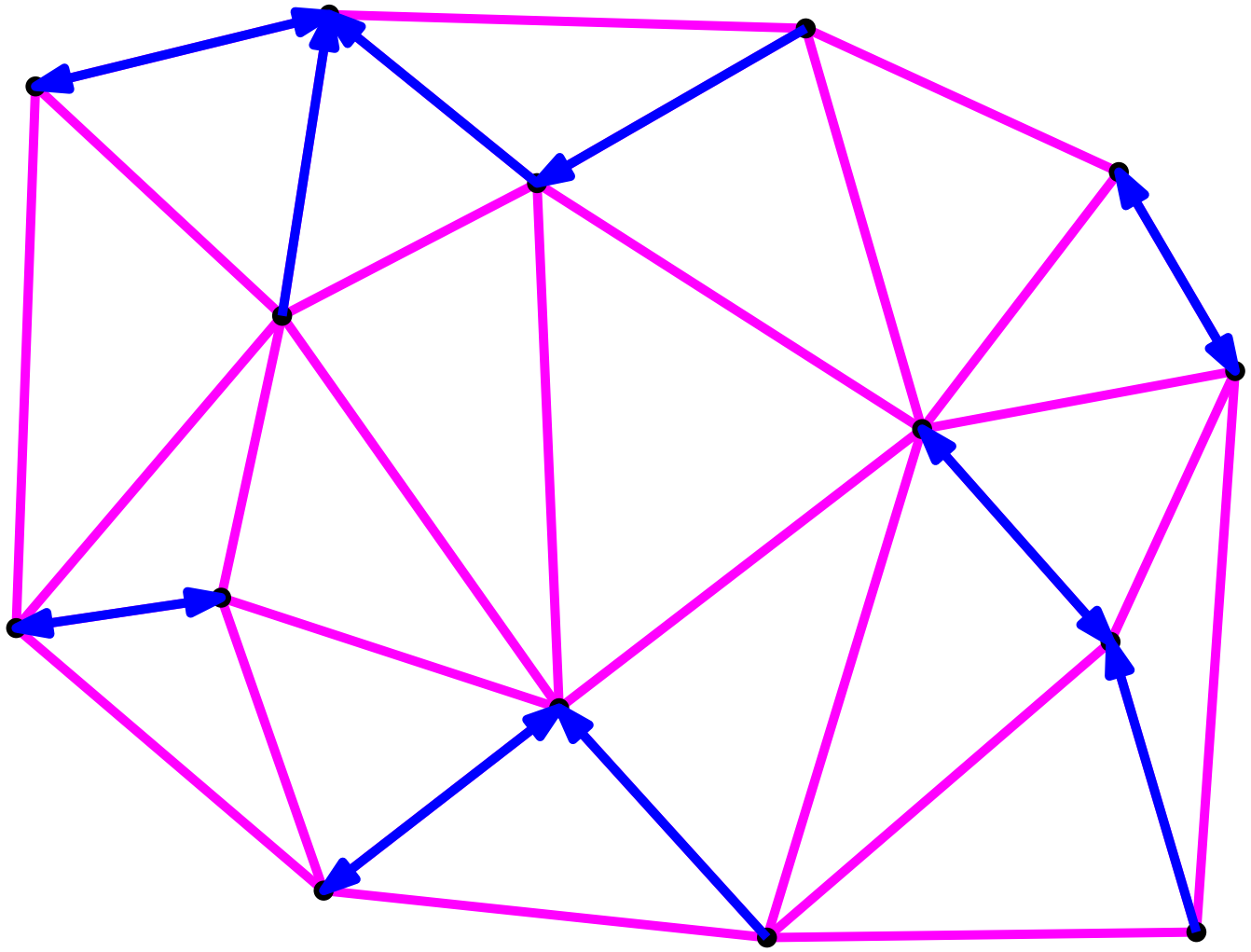
nearest neighbor graph



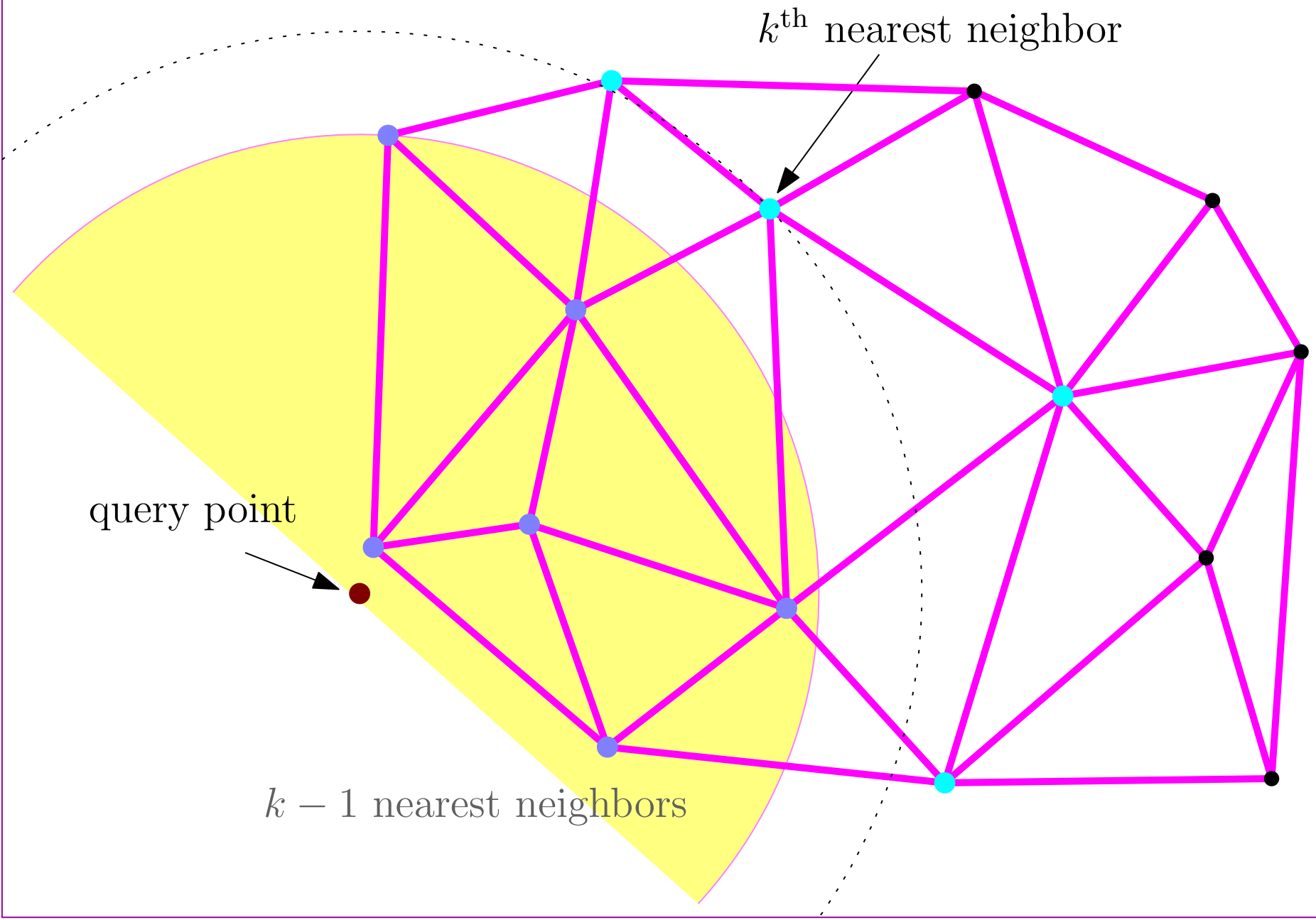
q nearest neighbor of p

$\Rightarrow$ pq Delaunay edge

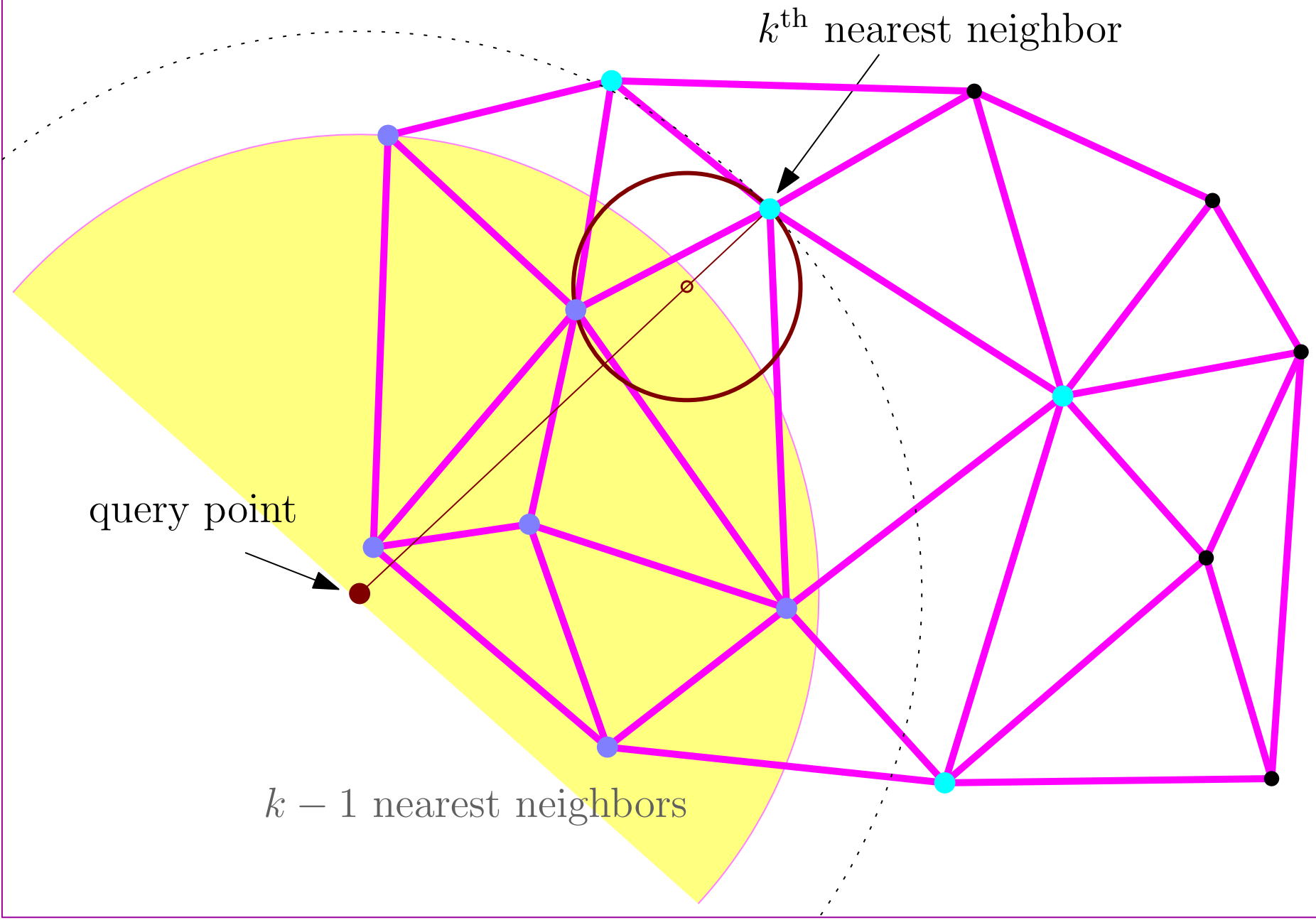
nearest neighbor graph



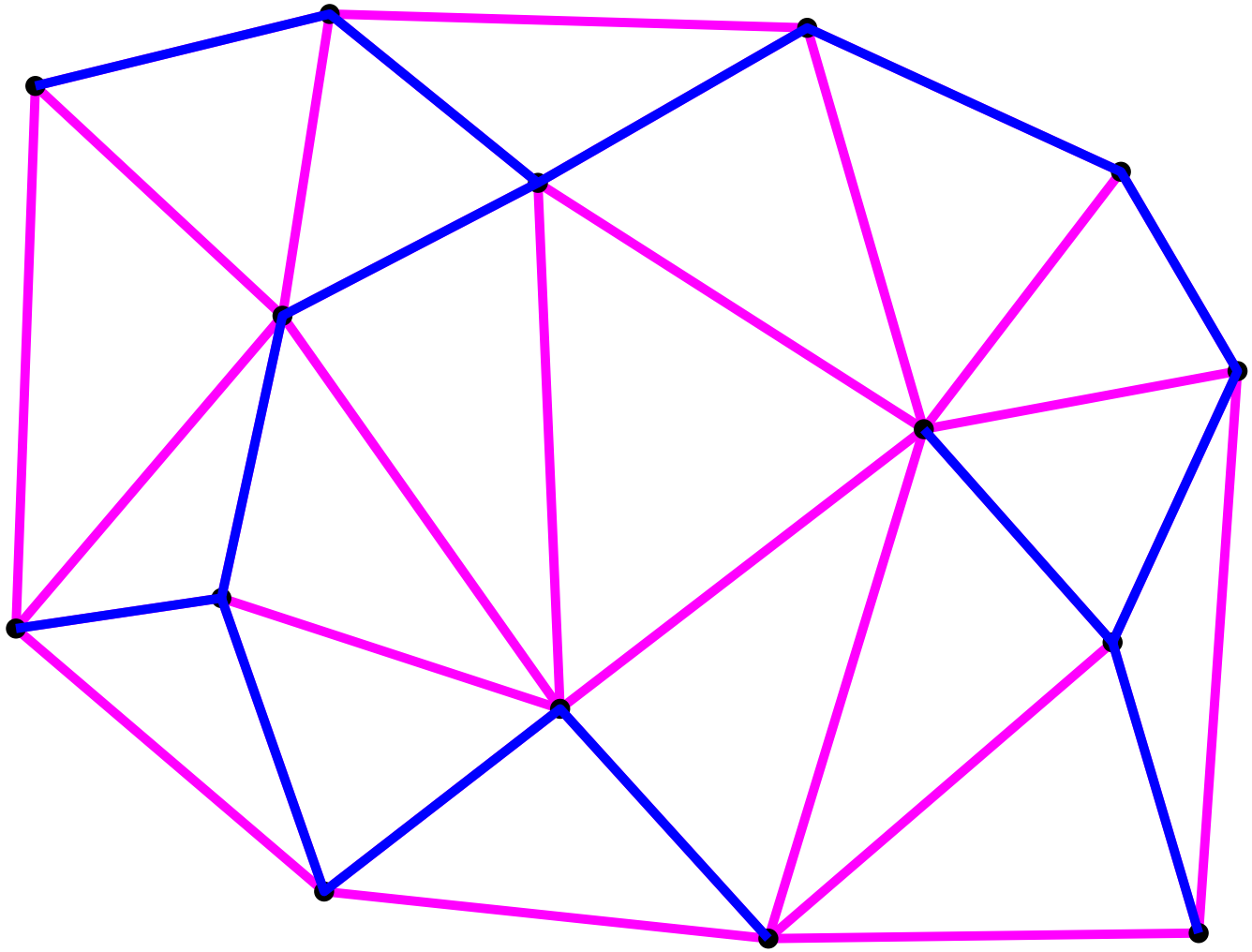
k nearest neighbors



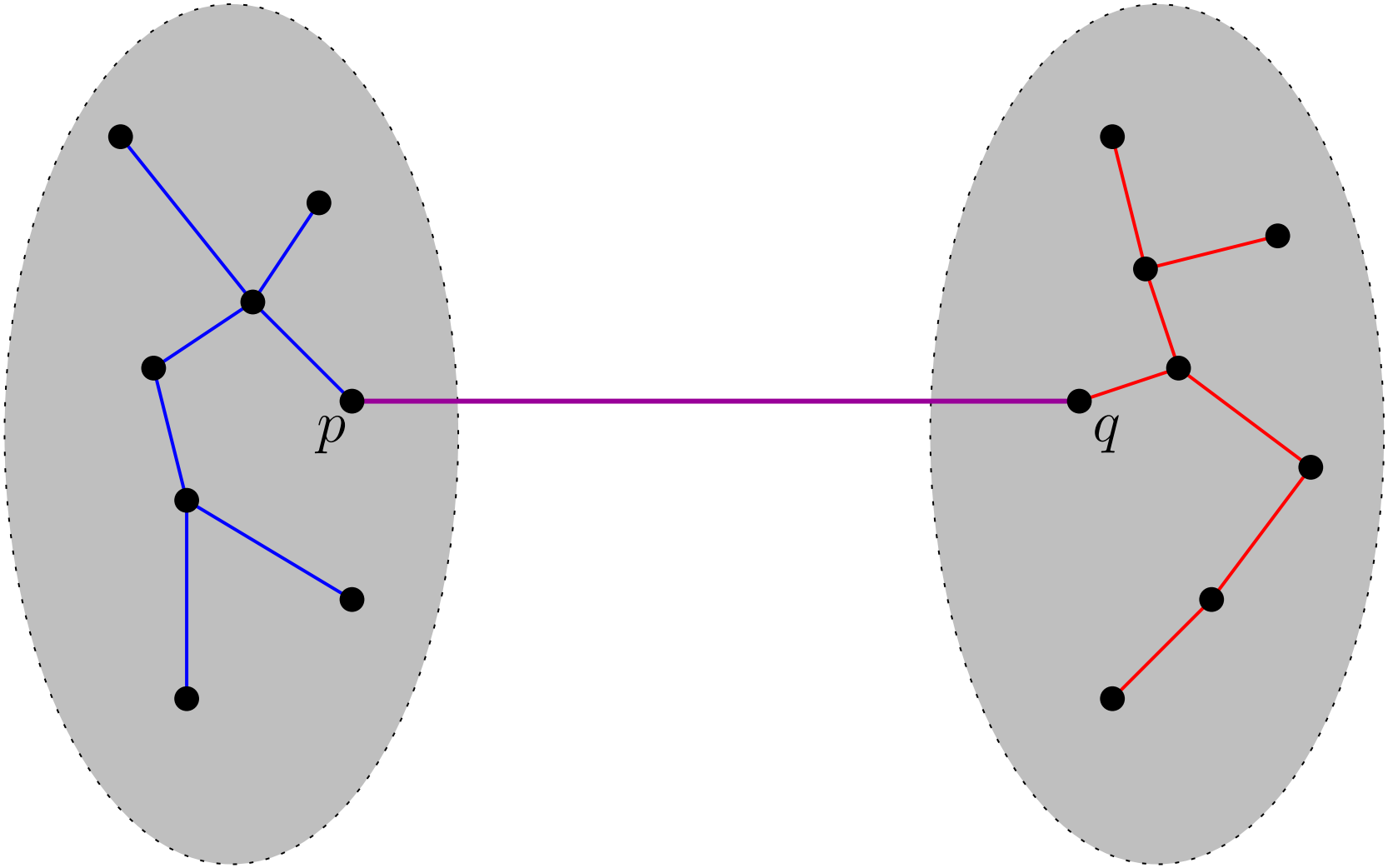
k nearest neighbors



Minimum Spanning Tree

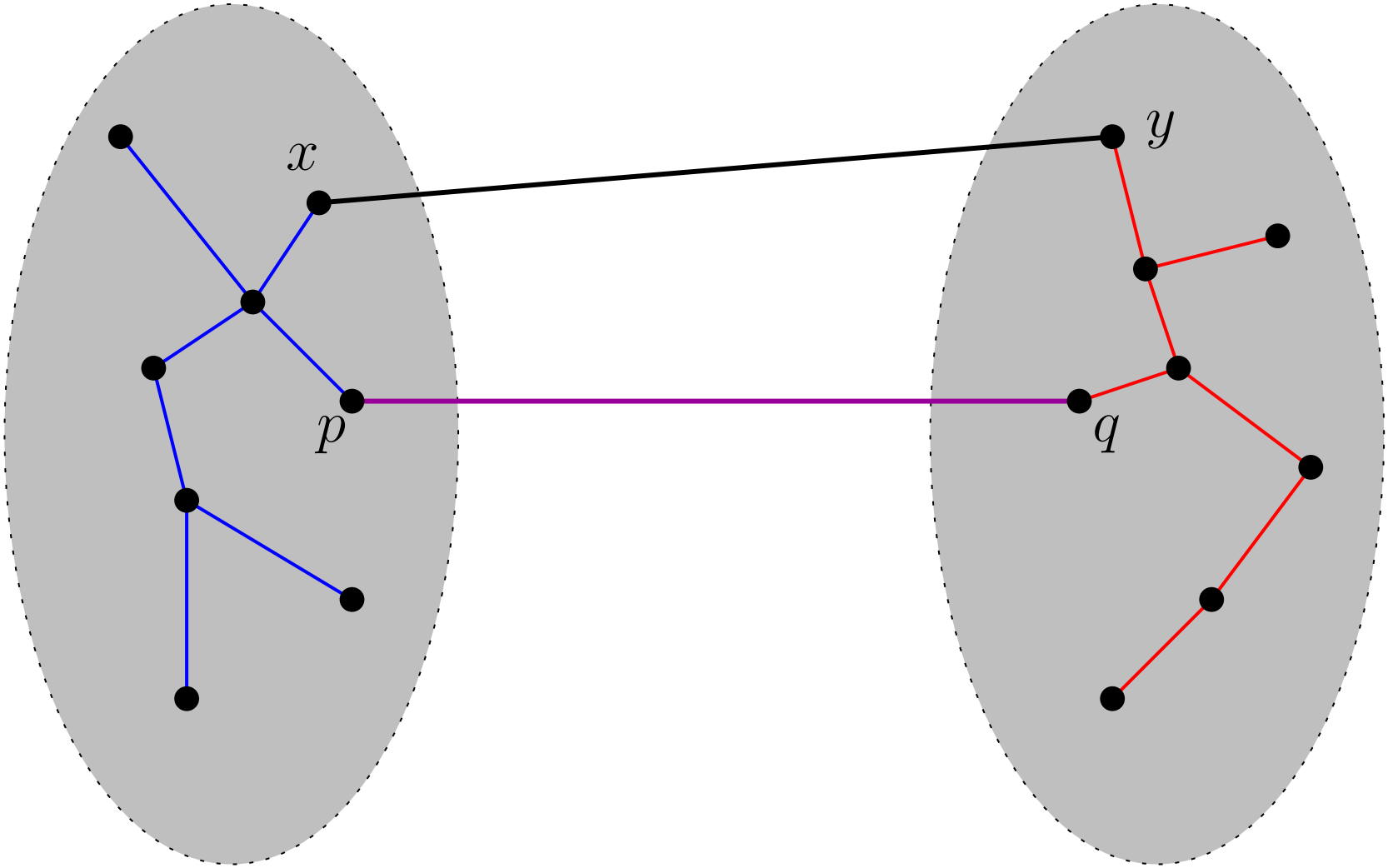


Minimum Spanning Tree

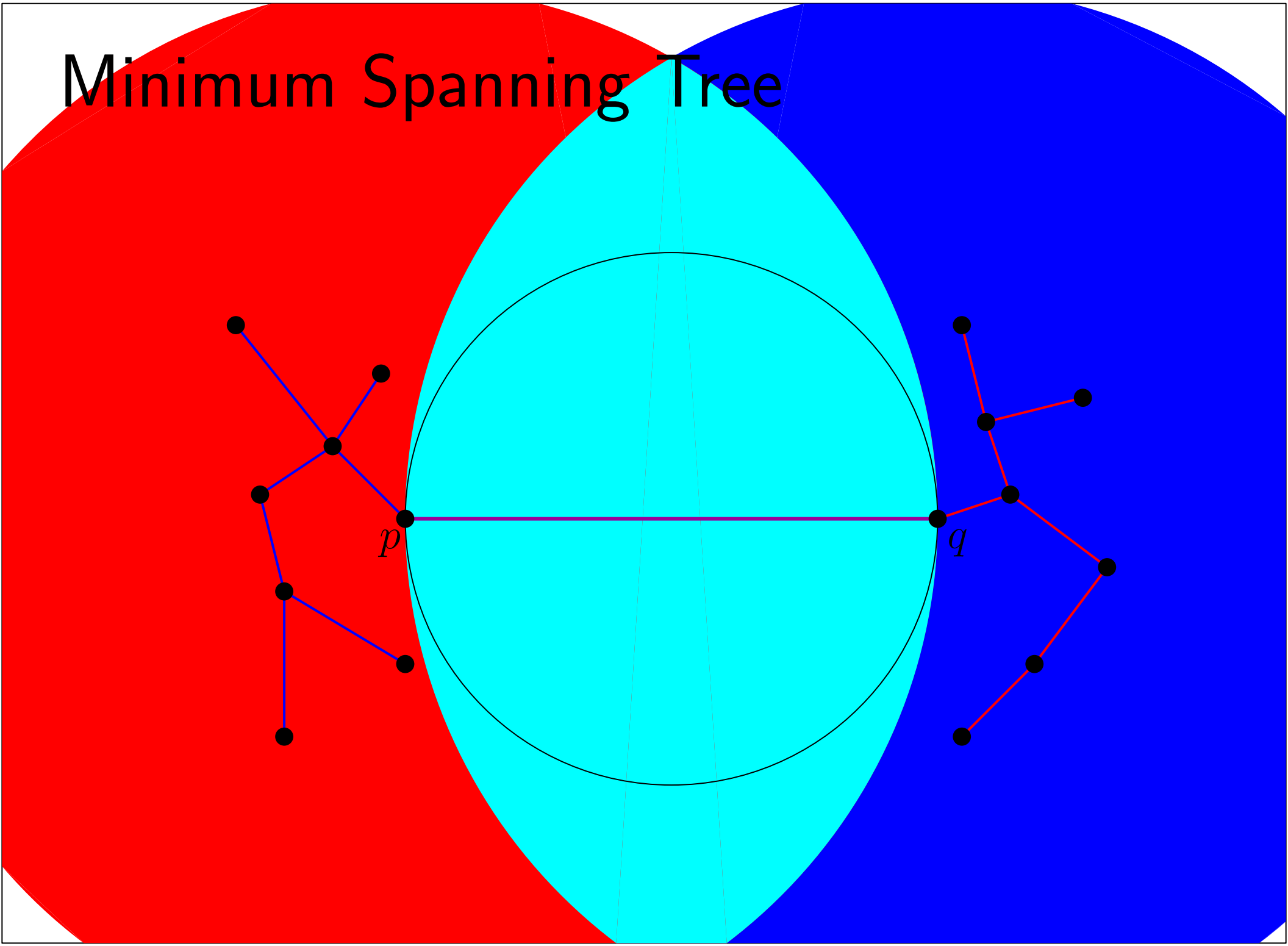


Minimum Spanning Tree

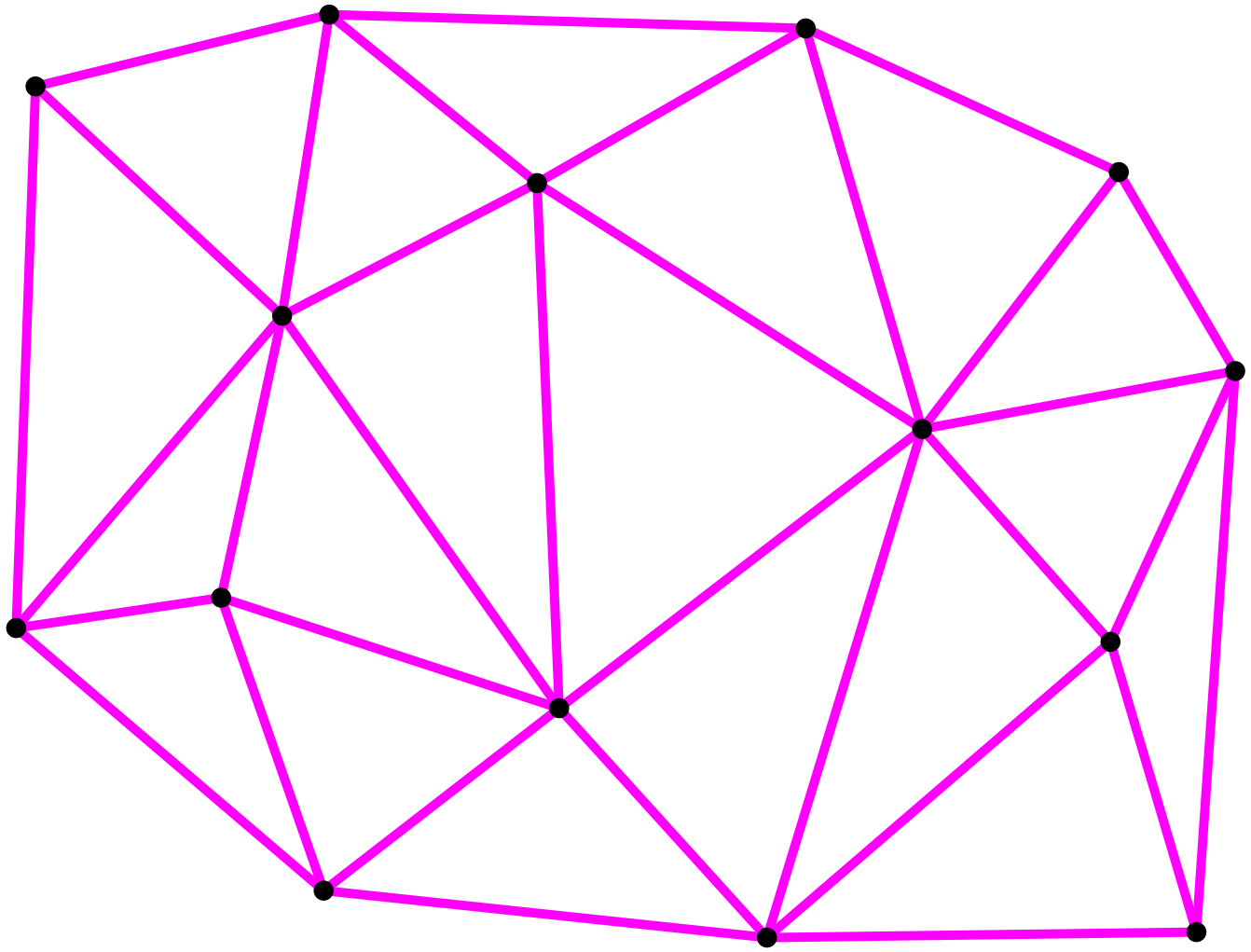
$$\forall [pq] \in A,\ \|p - q\| = \min\{\|x - y\| \mid x \in A_p, y \in A_q\}$$



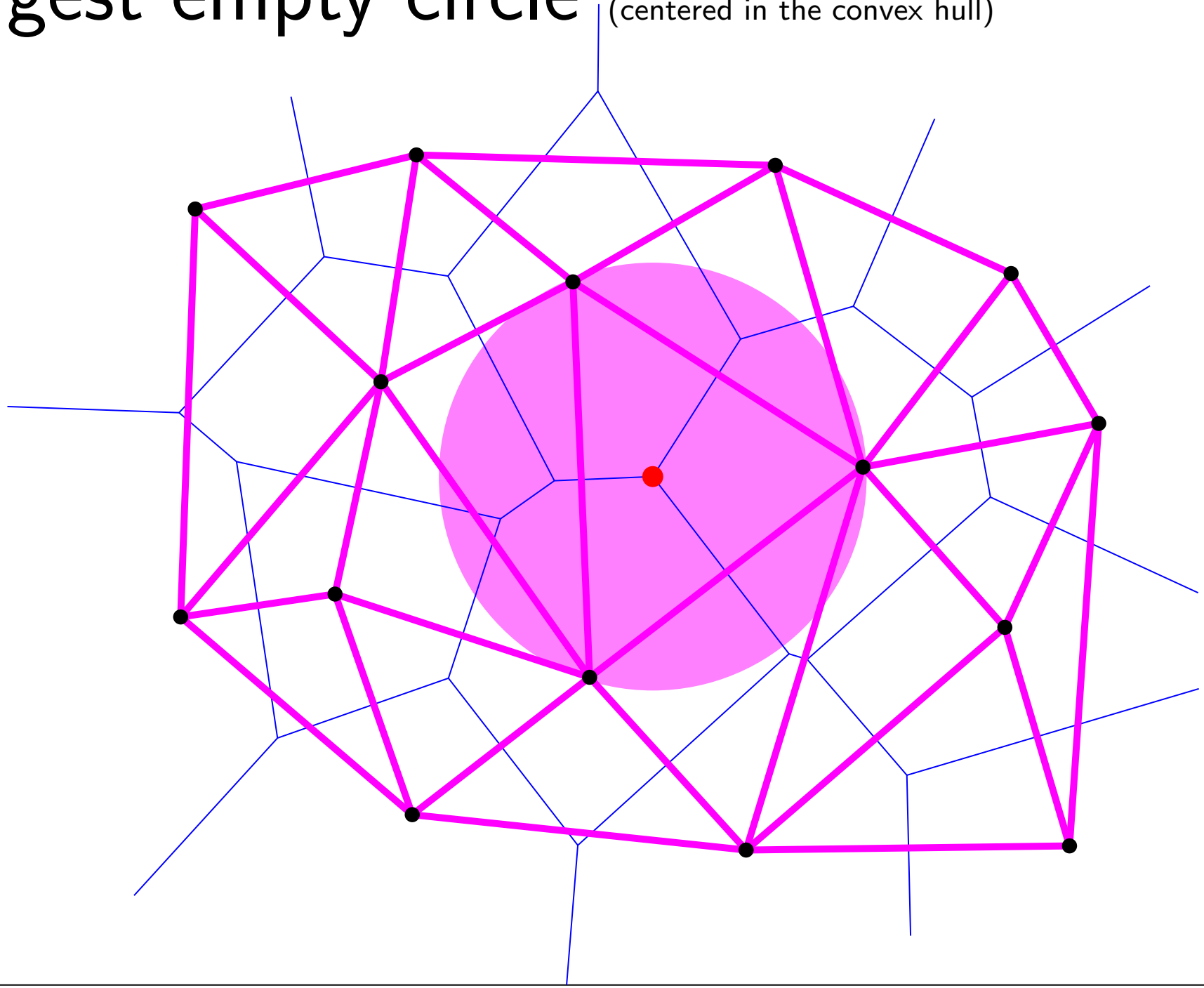
Minimum Spanning Tree



Largest empty circle (centered in the convex hull)

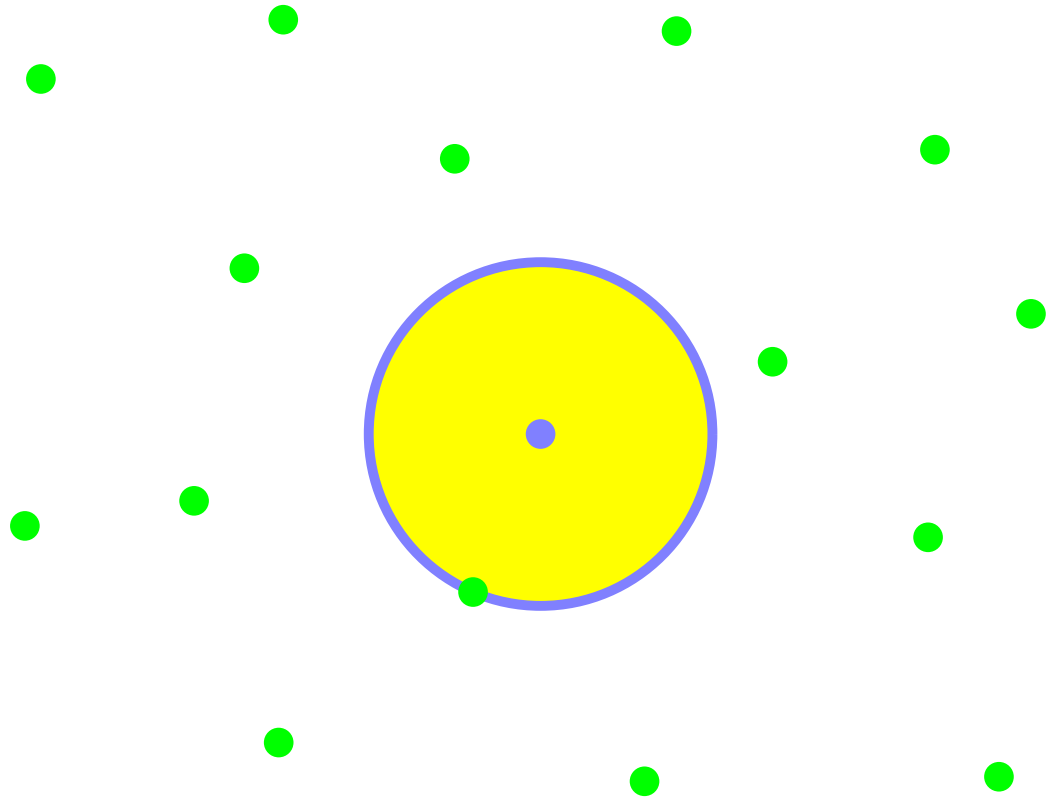


Largest empty circle (centered in the convex hull)



Applications

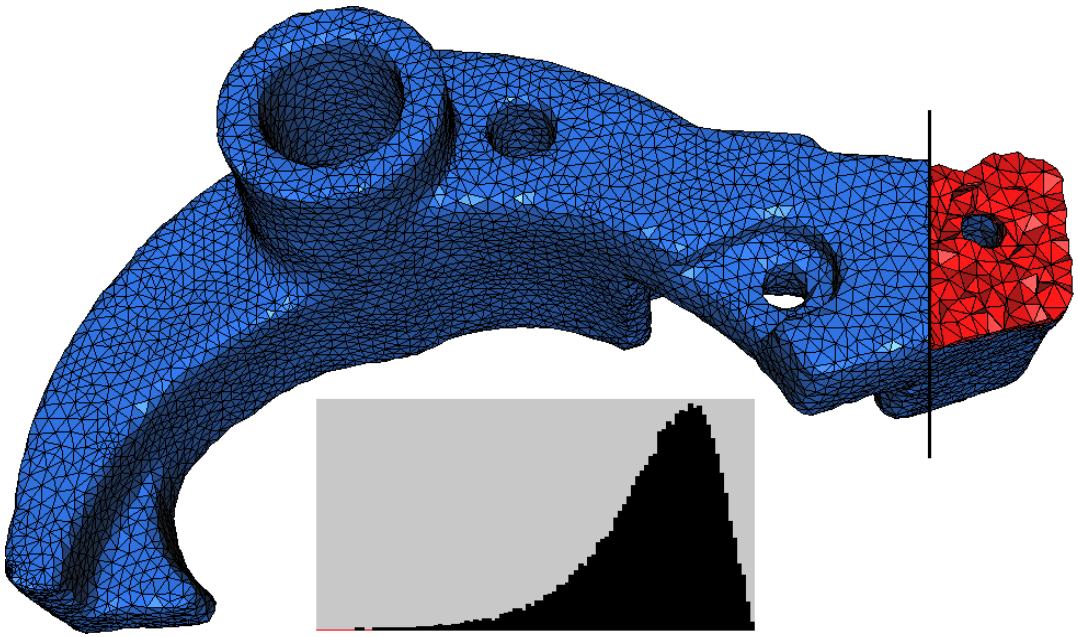
Databases, AI (NN-search)

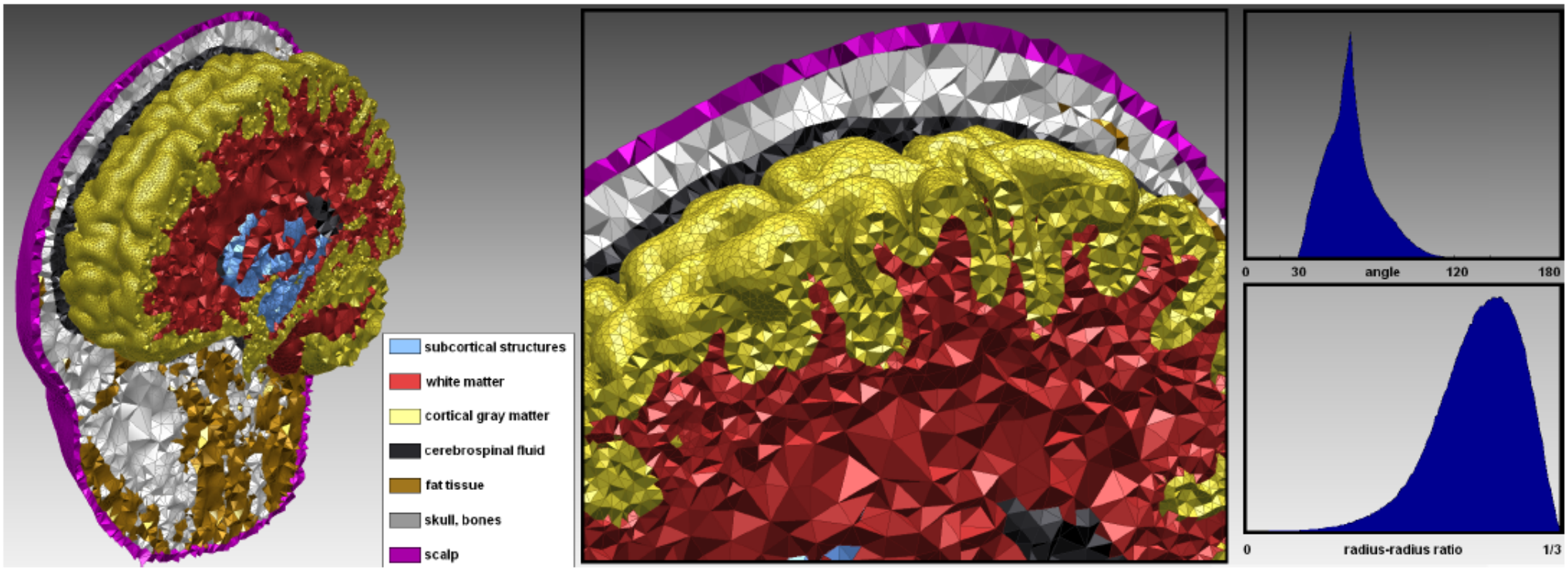


Applications

Databases, AI

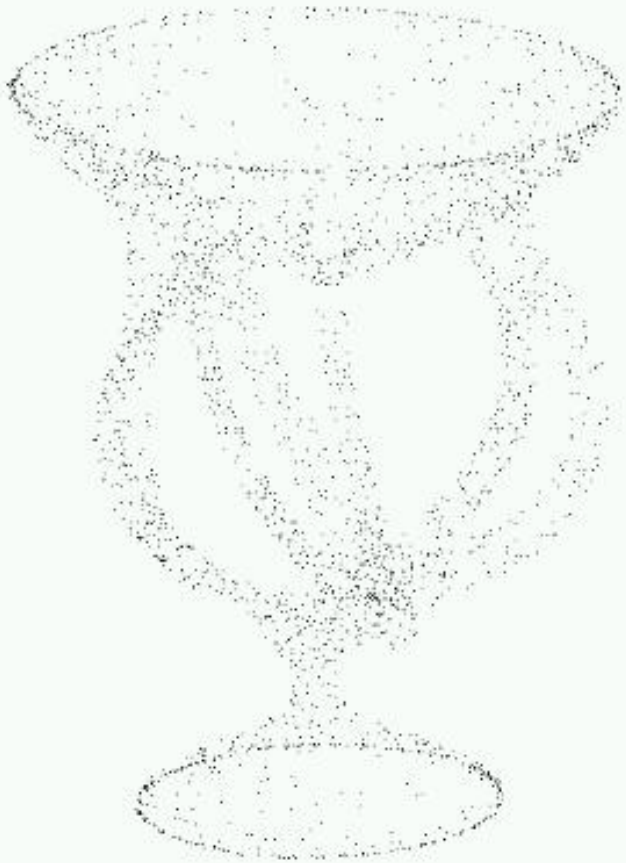
Mesh generation

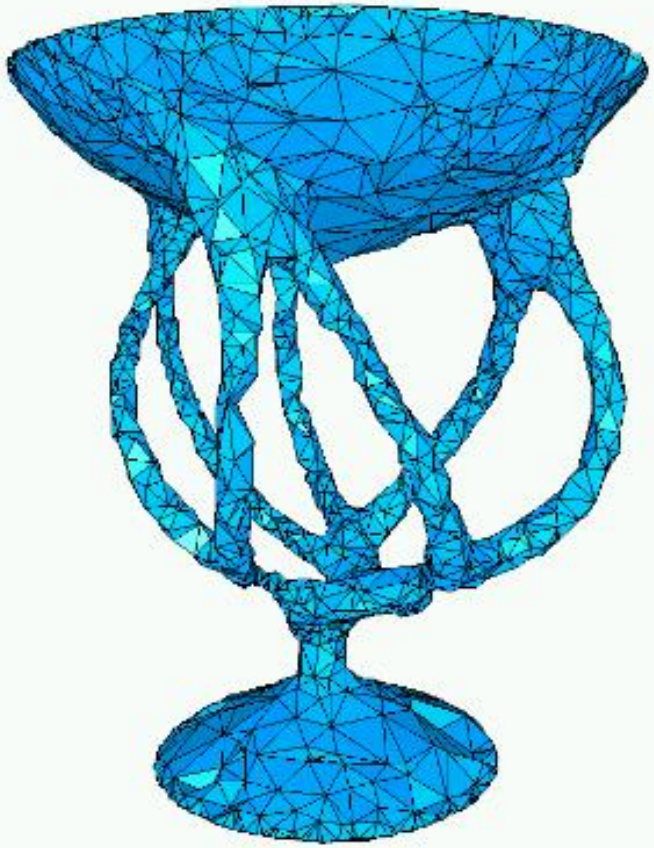




Applications

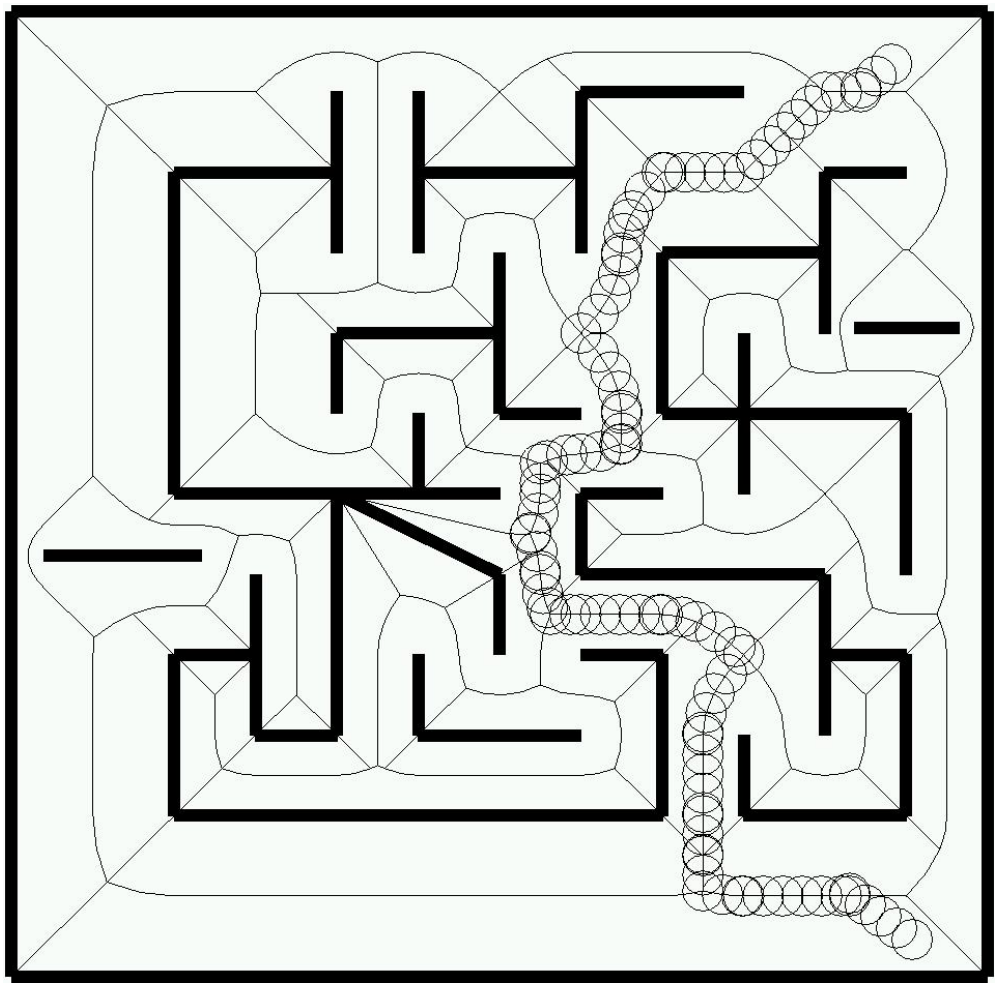
- Databases, AI
  - Mesh generation
- Reconstruction





Applications

- Databases, AI
- Path planning
  - Mesh generation
- Reconstruction



Applications

Databases, AI

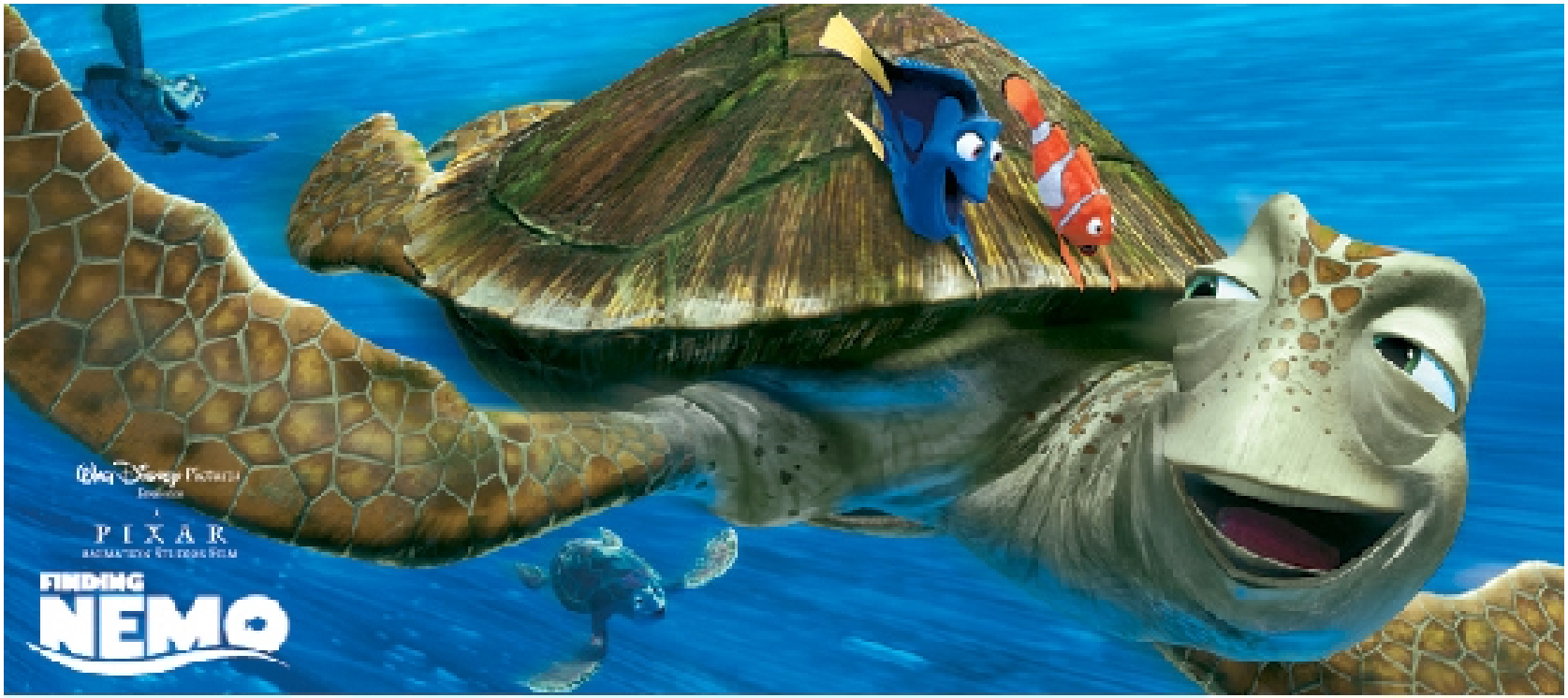
Mesh generation

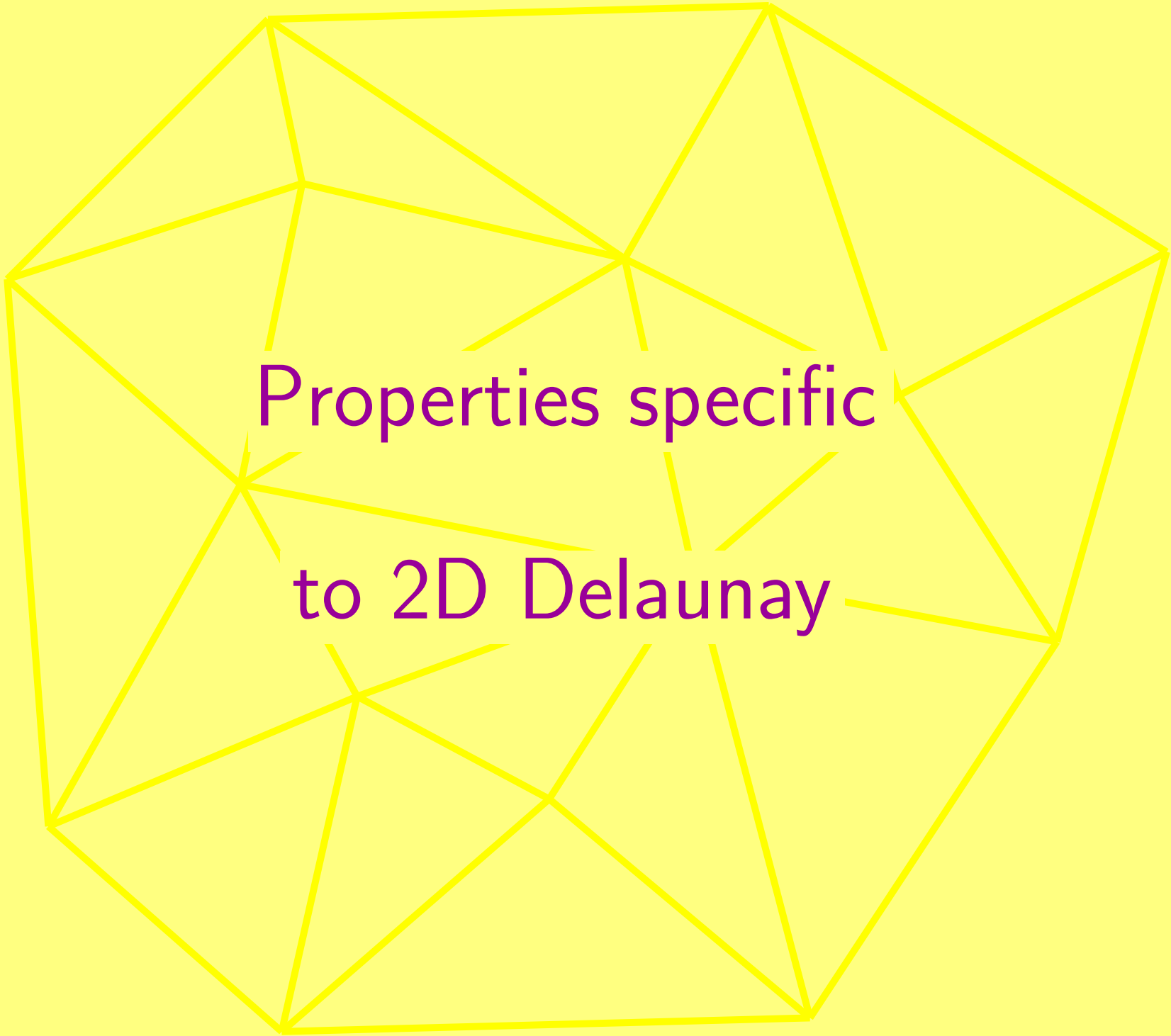
Reconstruction

Path planning

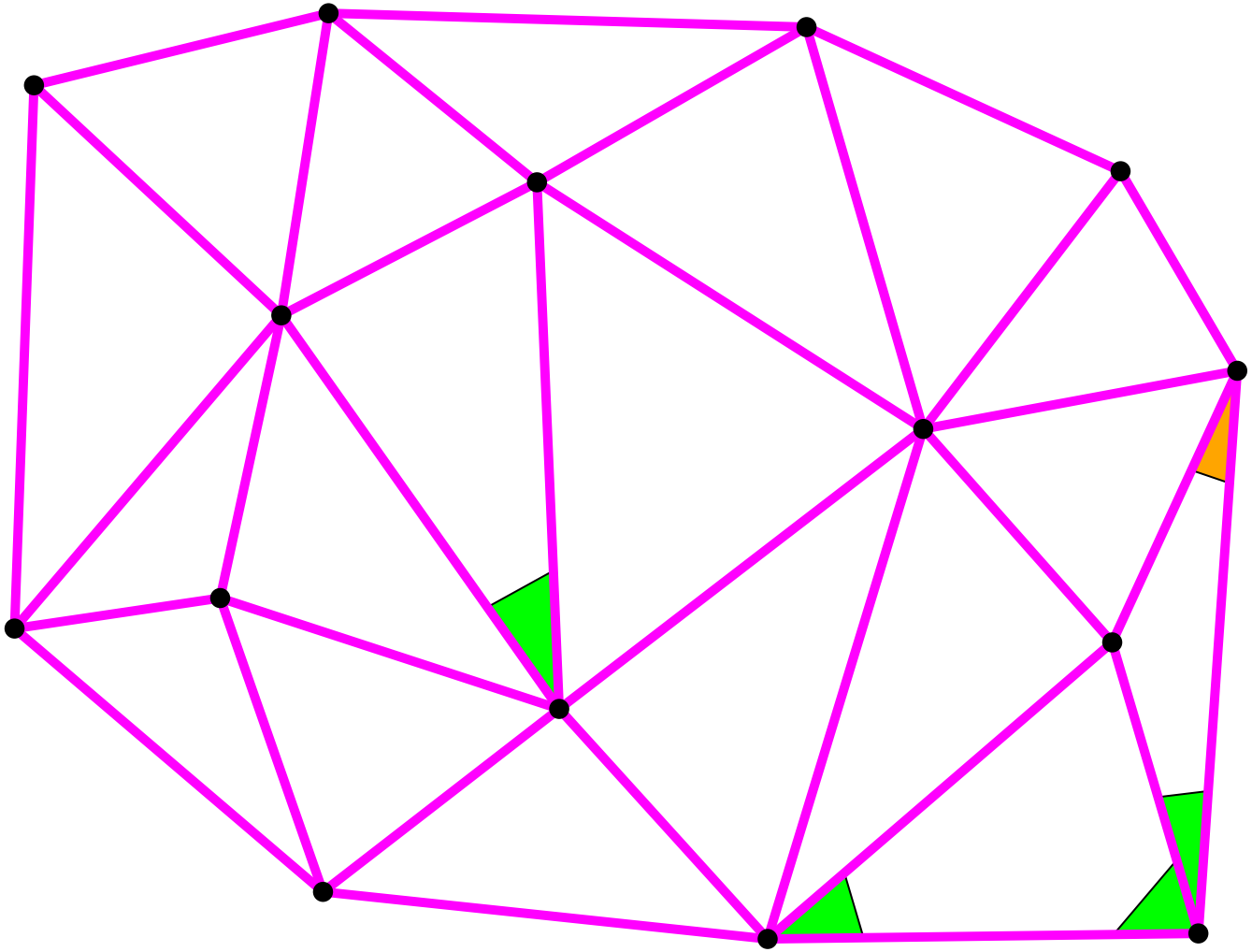
and many more

(e.g. texture synthesis)

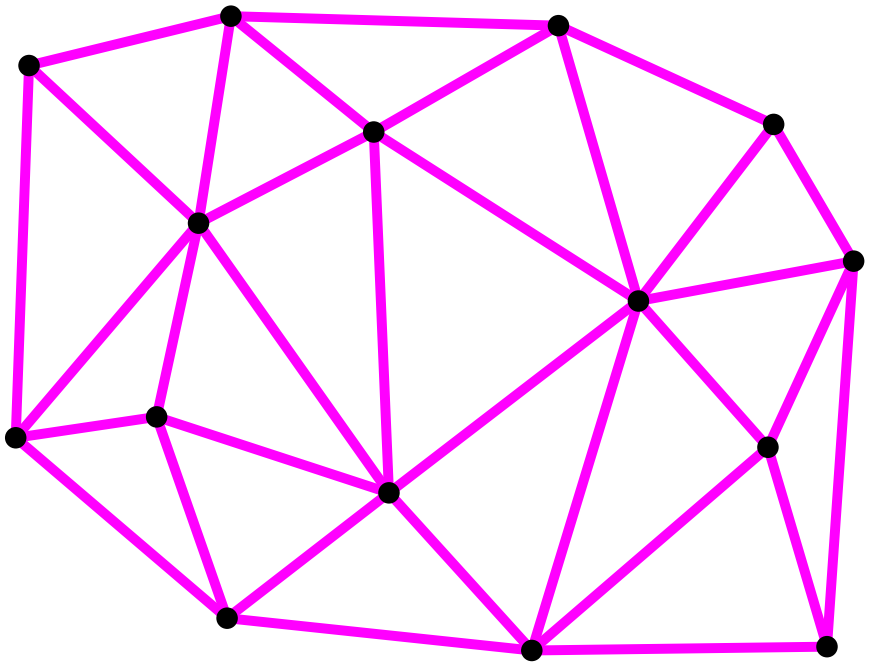


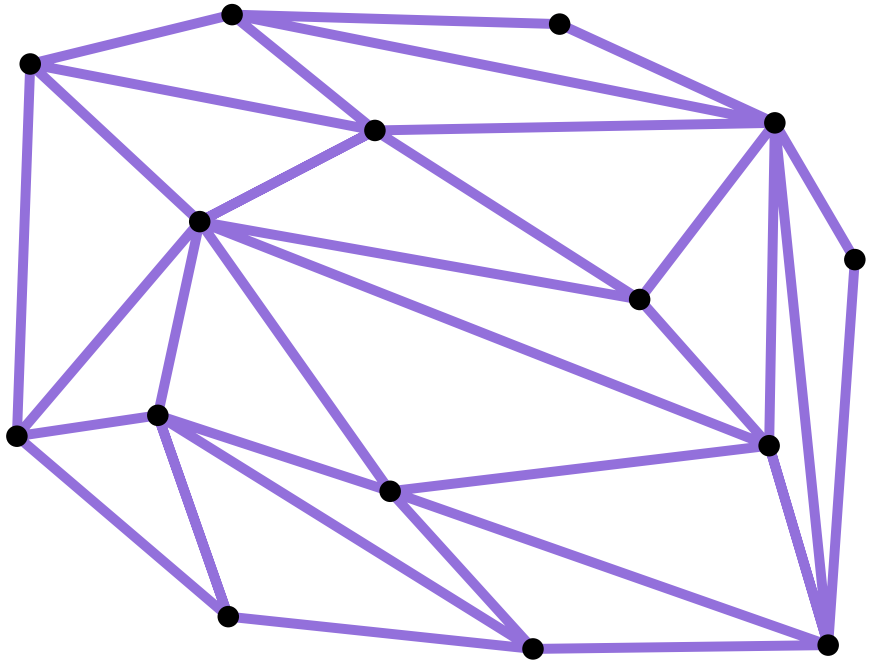

Properties specific
to 2D Delaunay

Delaunay maximizes the smallest angle

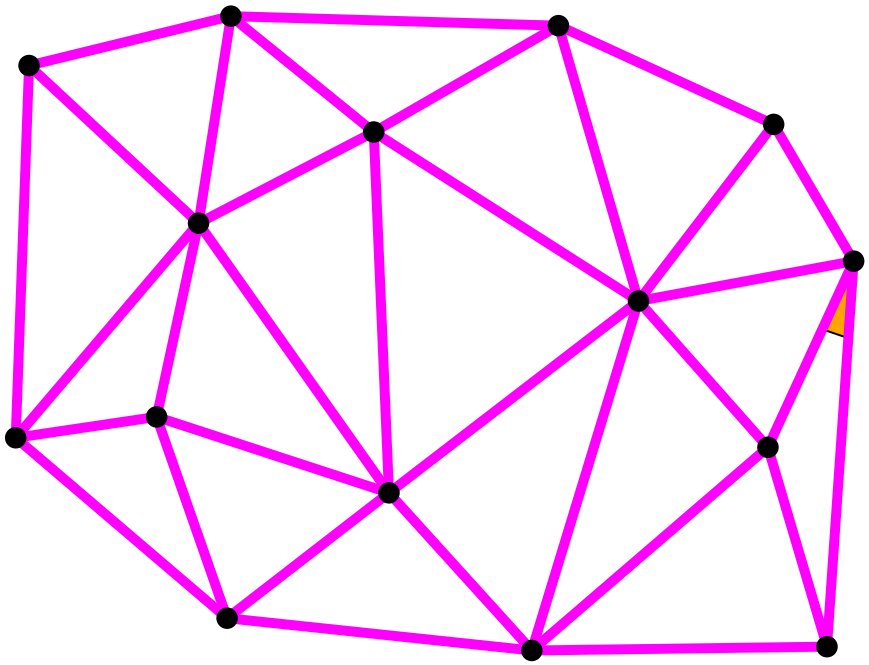


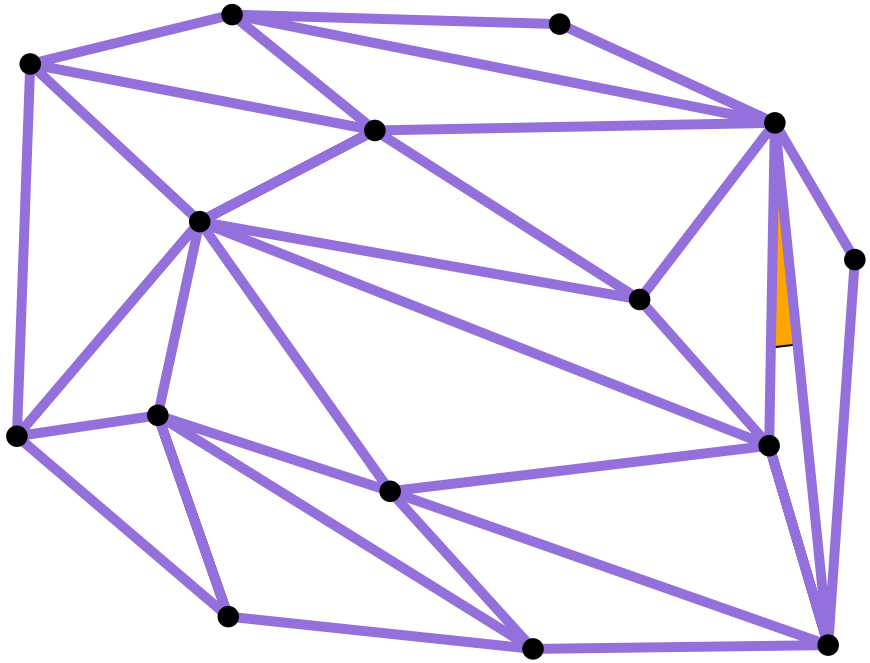
Delaunay maximizes the smallest angle



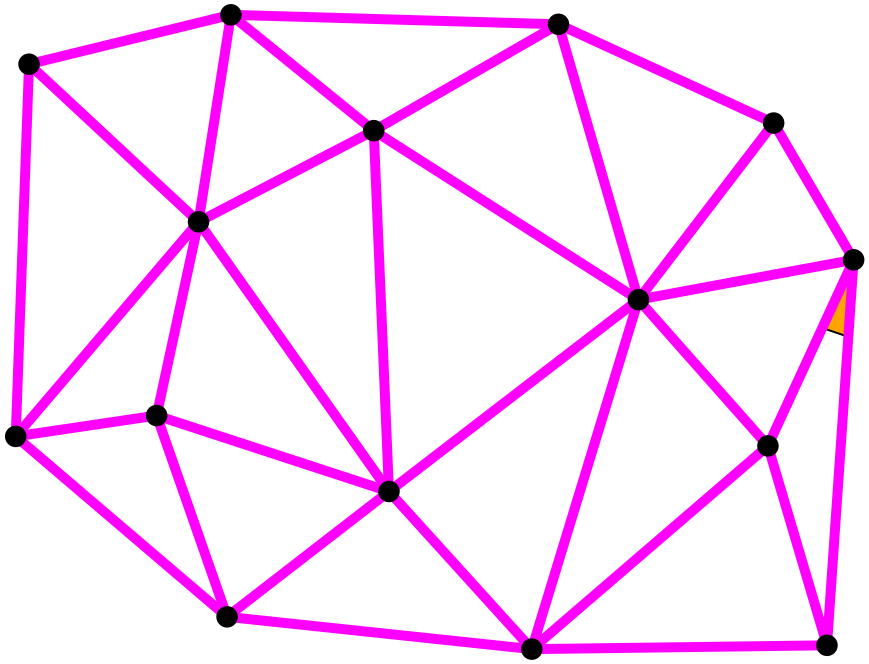


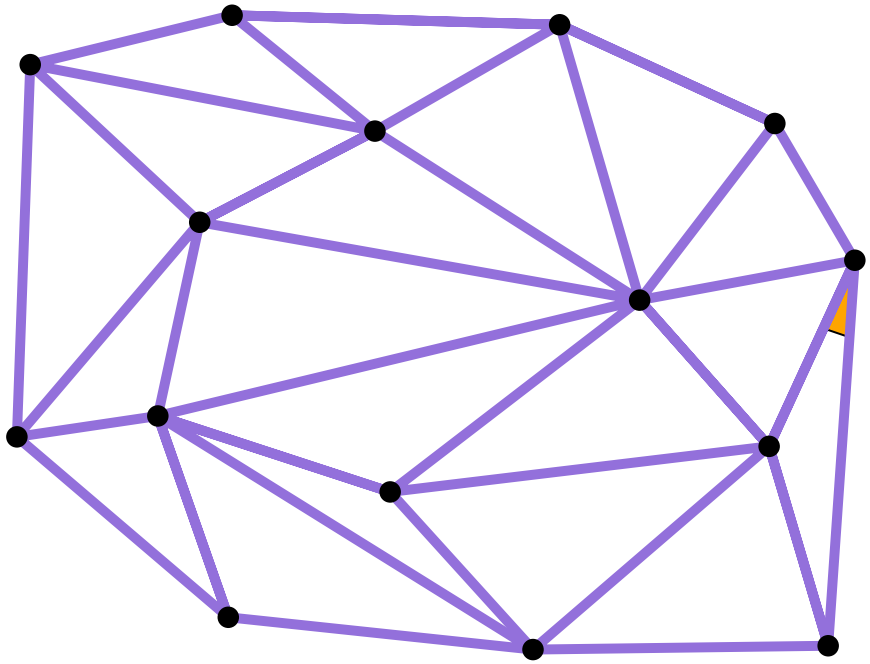
Delaunay maximizes the smallest angle



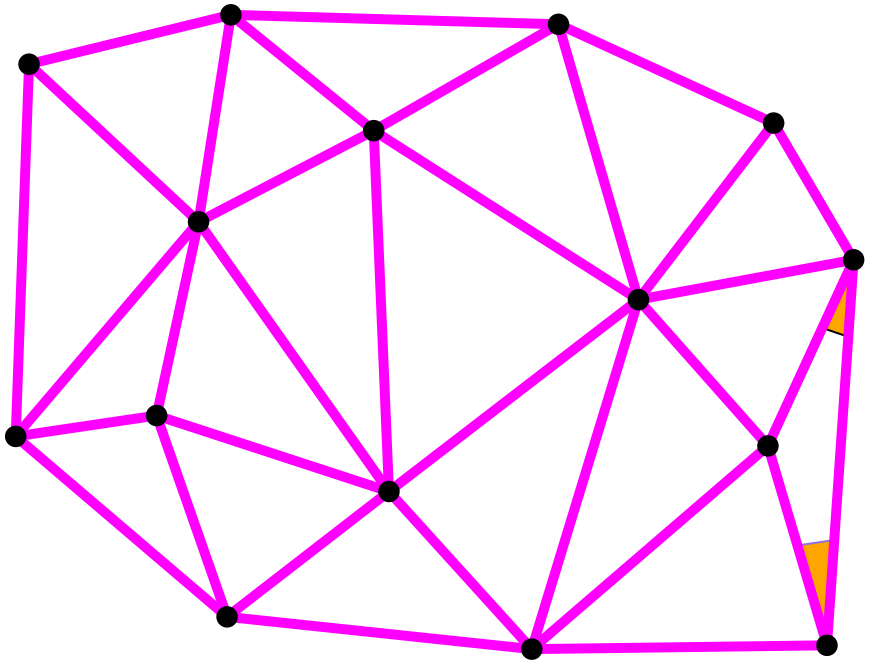


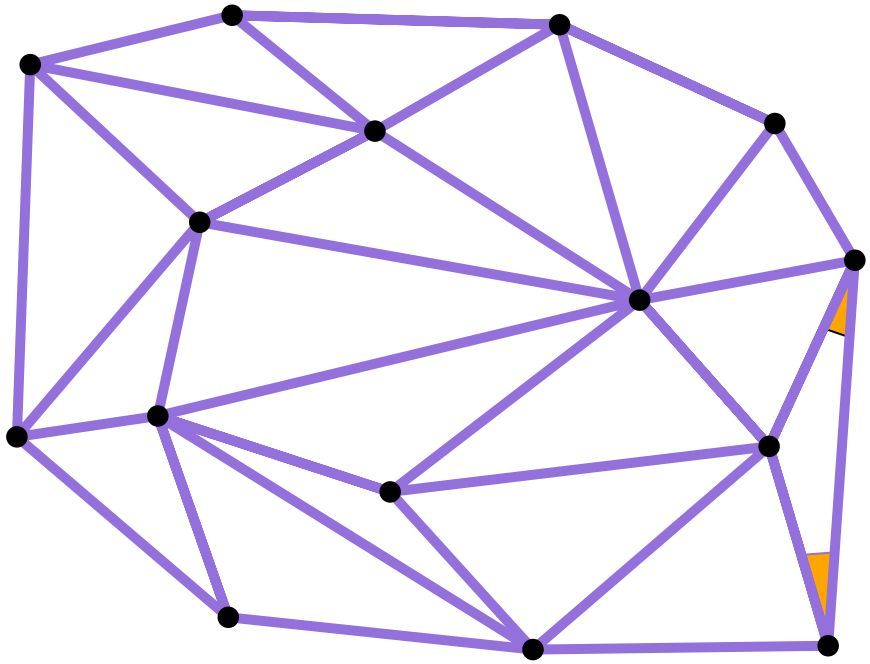
Delaunay maximizes the smallest angle

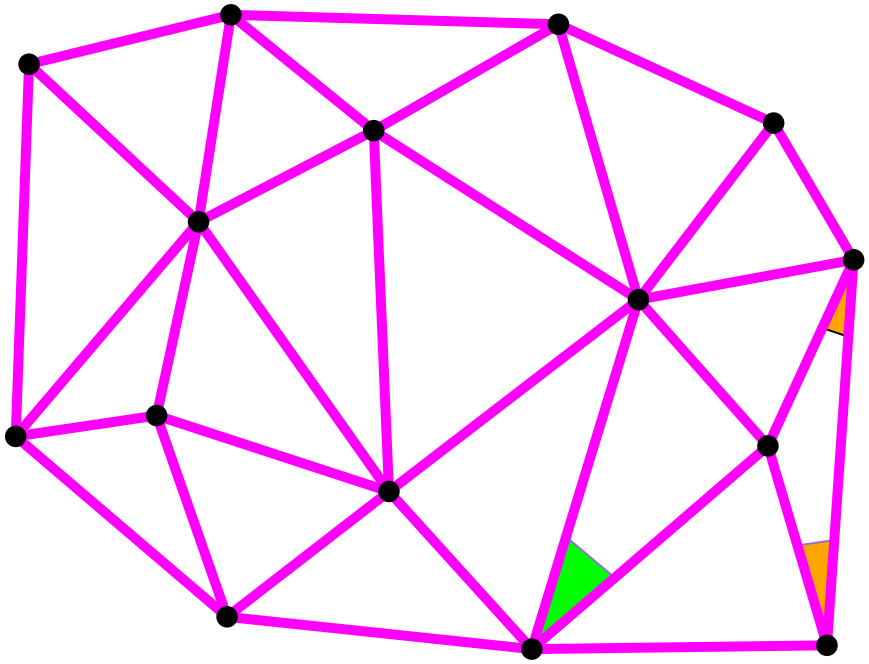


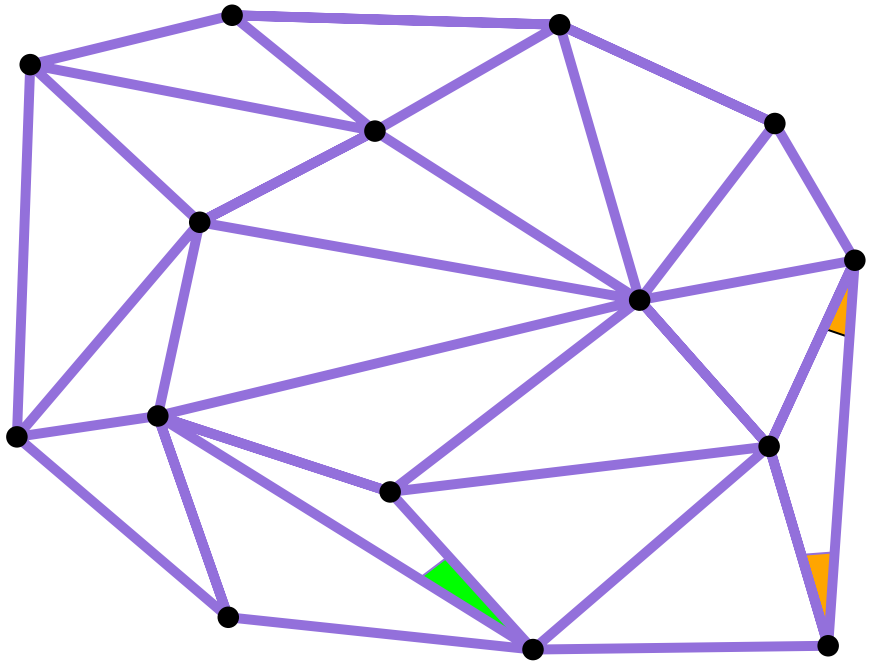


... but the converse is false



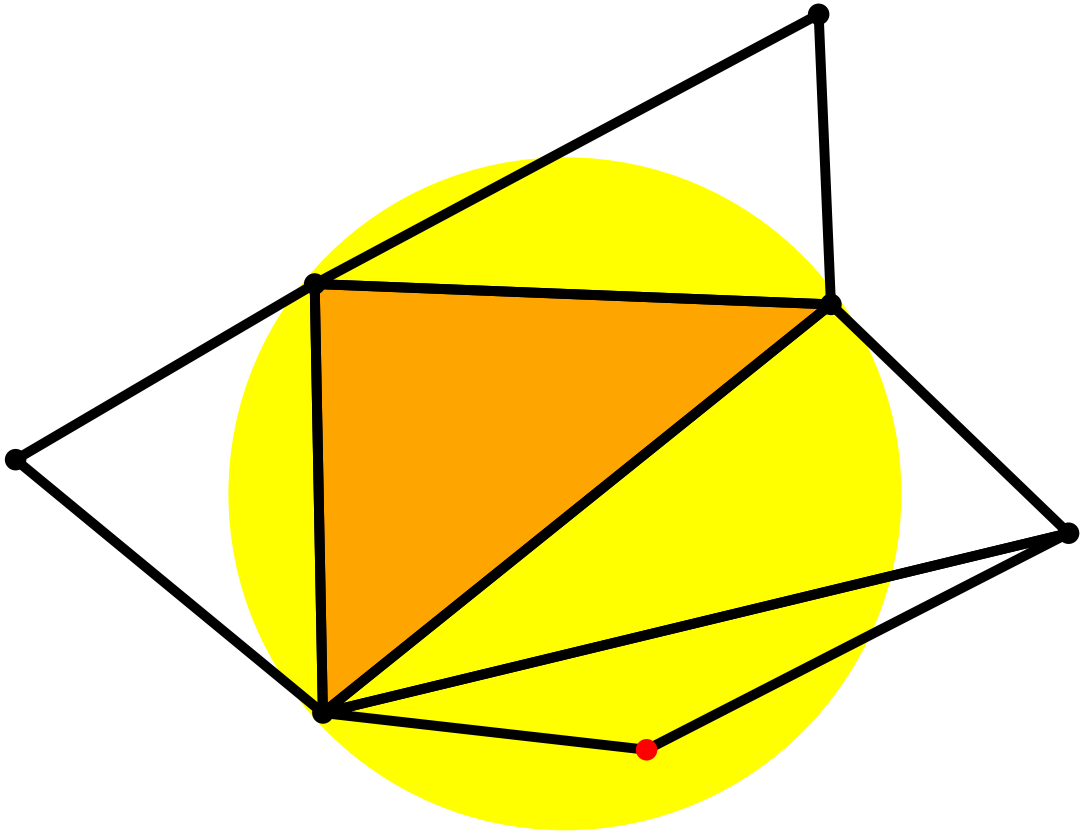






Delaunay maximizes the sequence of angles in lexicographic order

Local optimality vs global optimality



highlighted triangle is only locally Delaunay

Theorem

Locally Delaunay everywhere

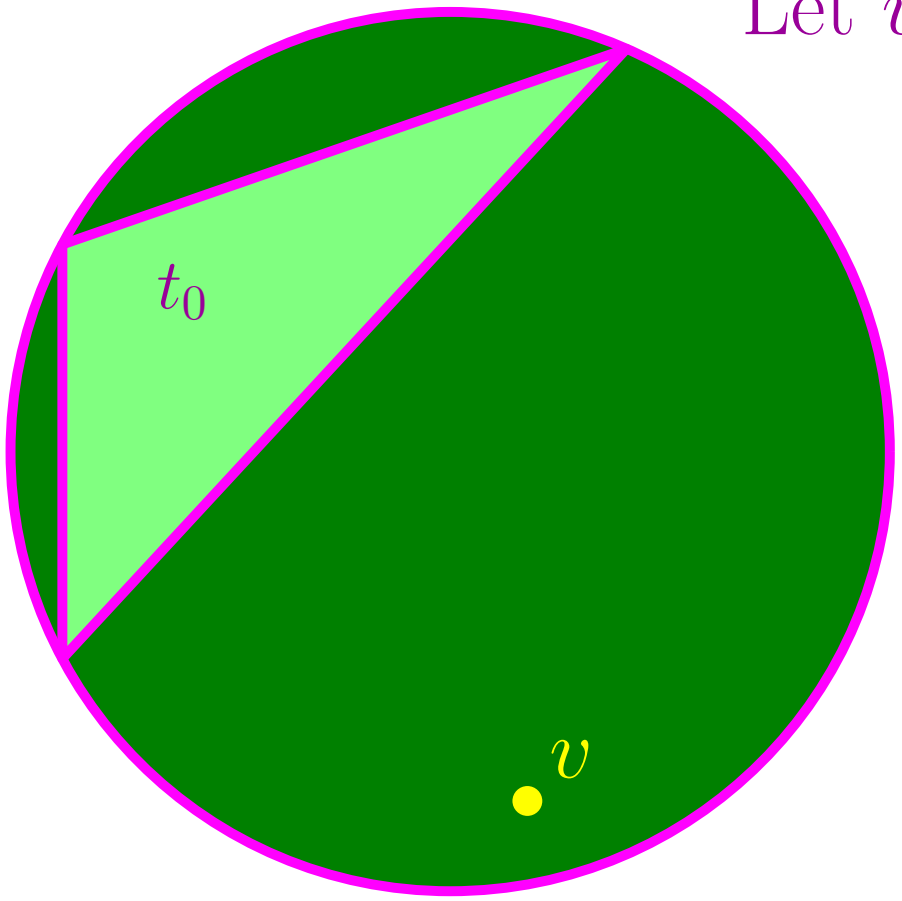
$\Longleftrightarrow$

Globally Delaunay

Proof:

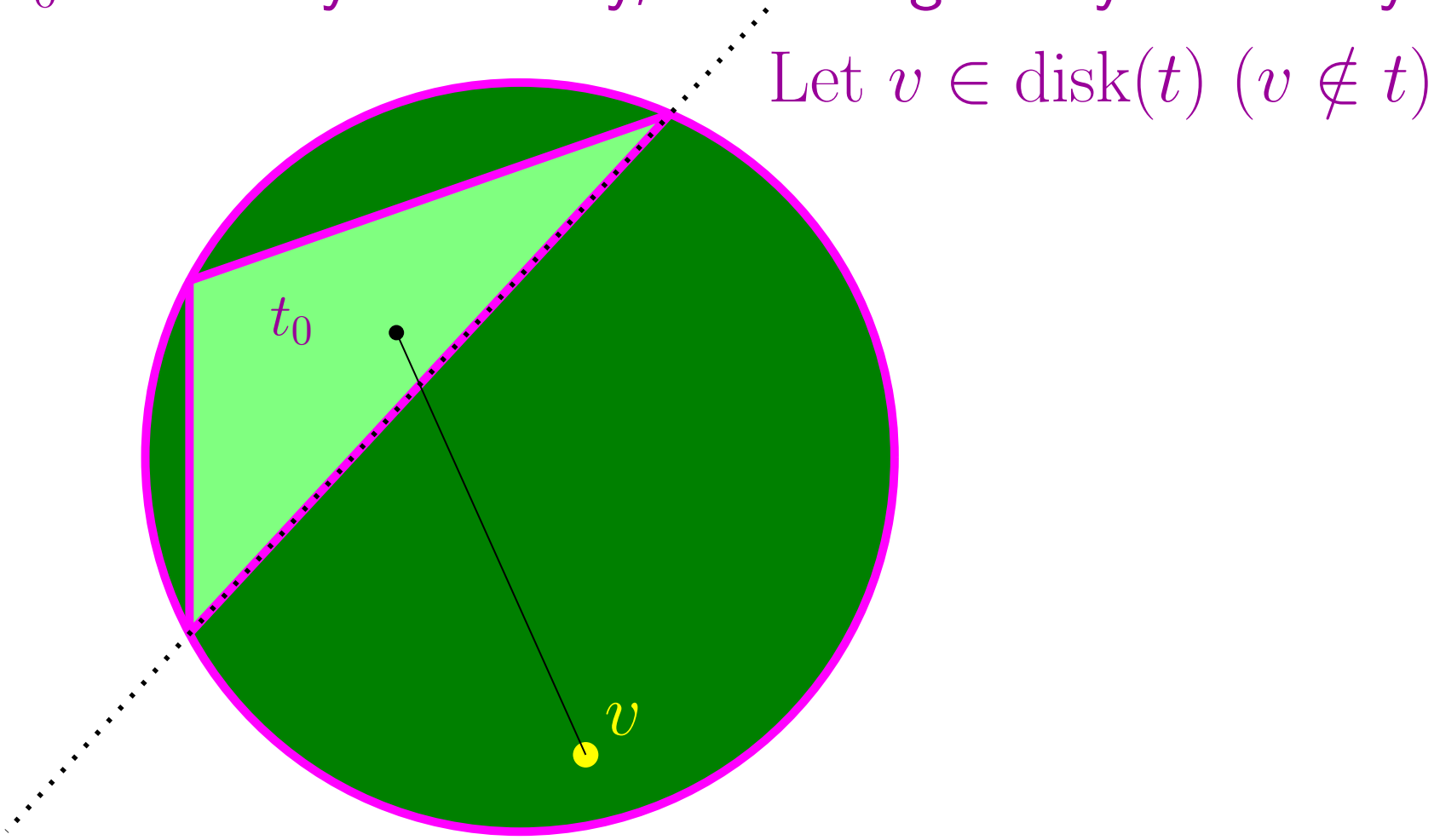
Let t_0 be locally Delaunay, but not globally Delaunay

Let $v \in \text{disk}(t)$ $(v \notin t)$



Proof:

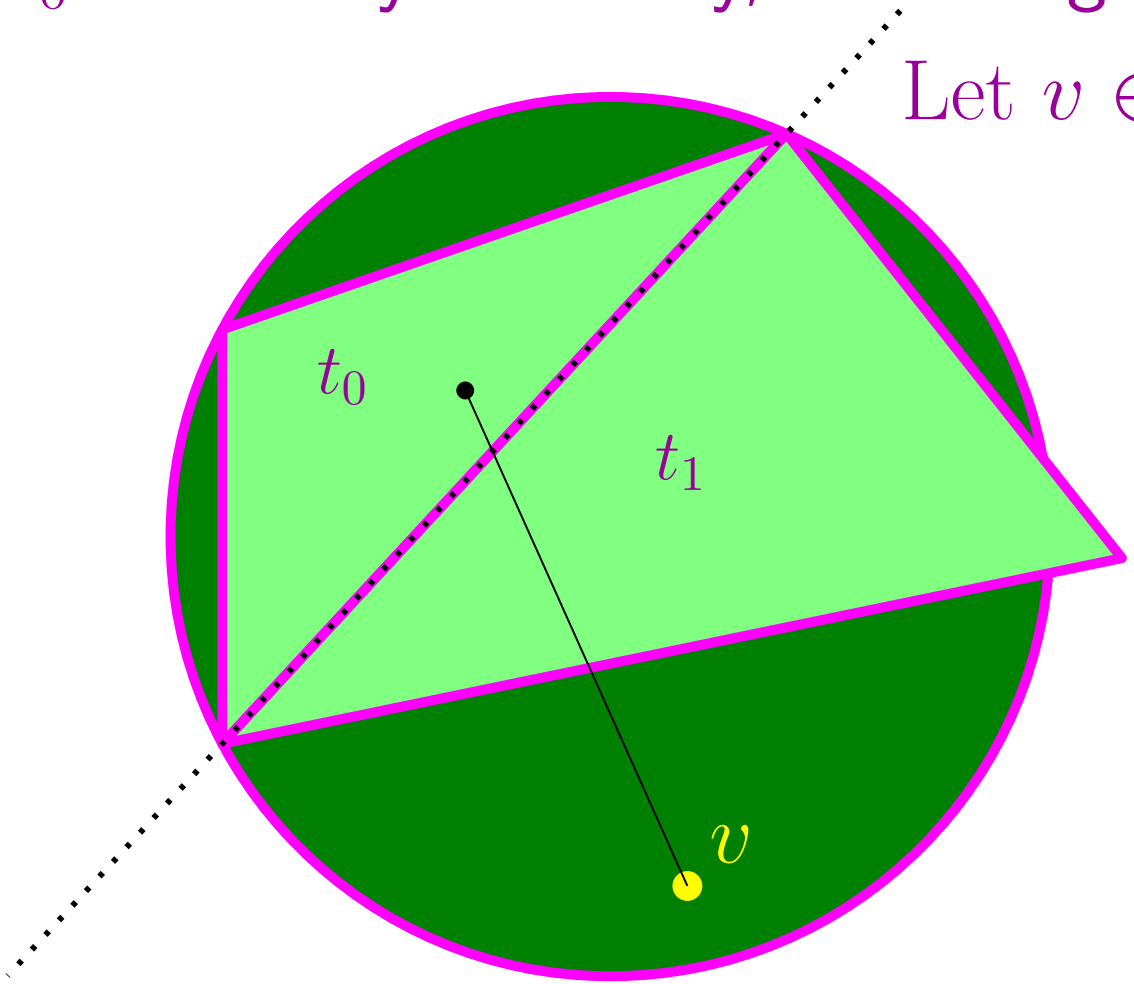
Let t_0 be locally Delaunay, but not globally Delaunay



Proof:

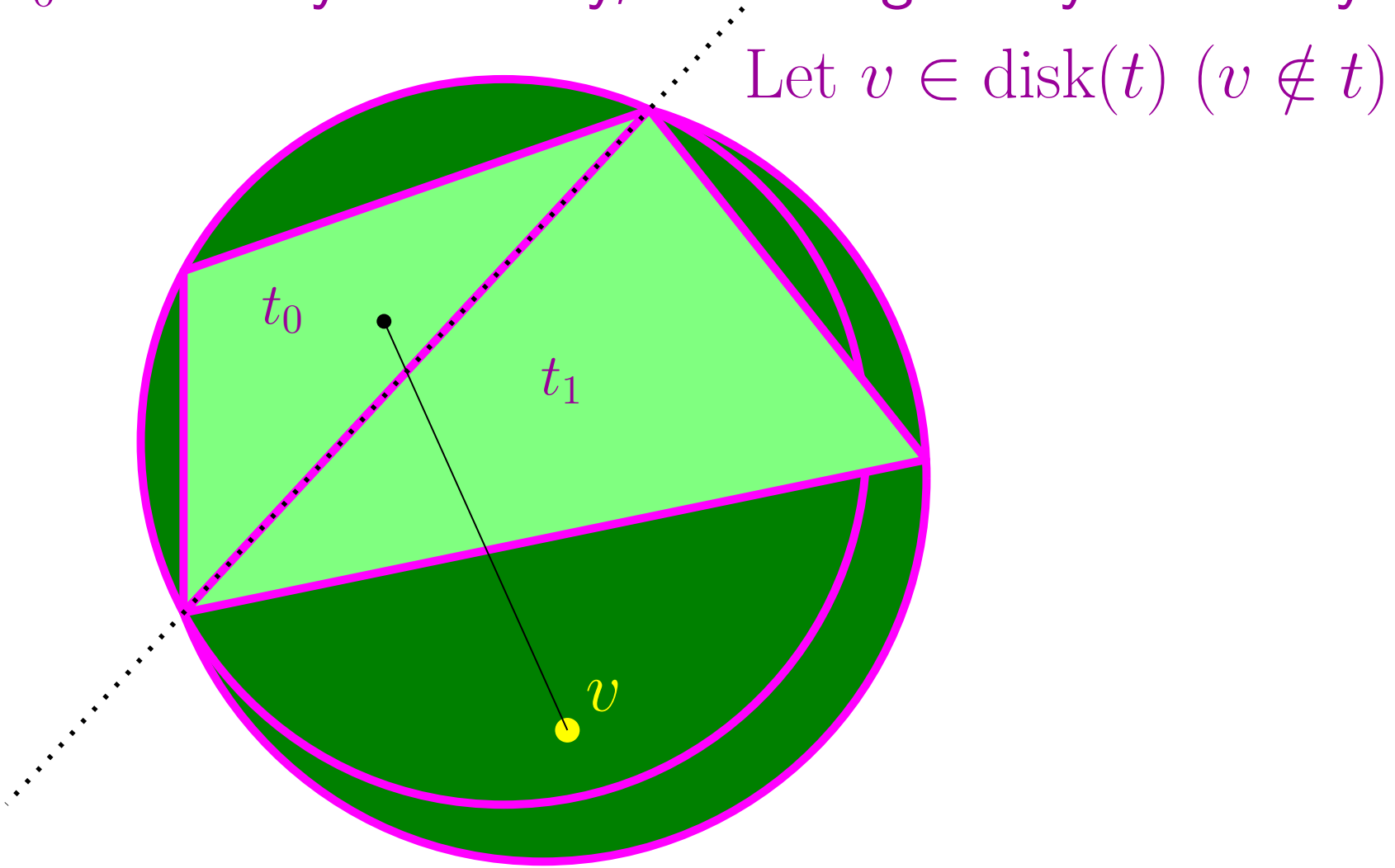
Let t_0 be locally Delaunay, but not globally Delaunay

Let $v \in \text{disk}(t)$ $(v \notin t)$



Proof:

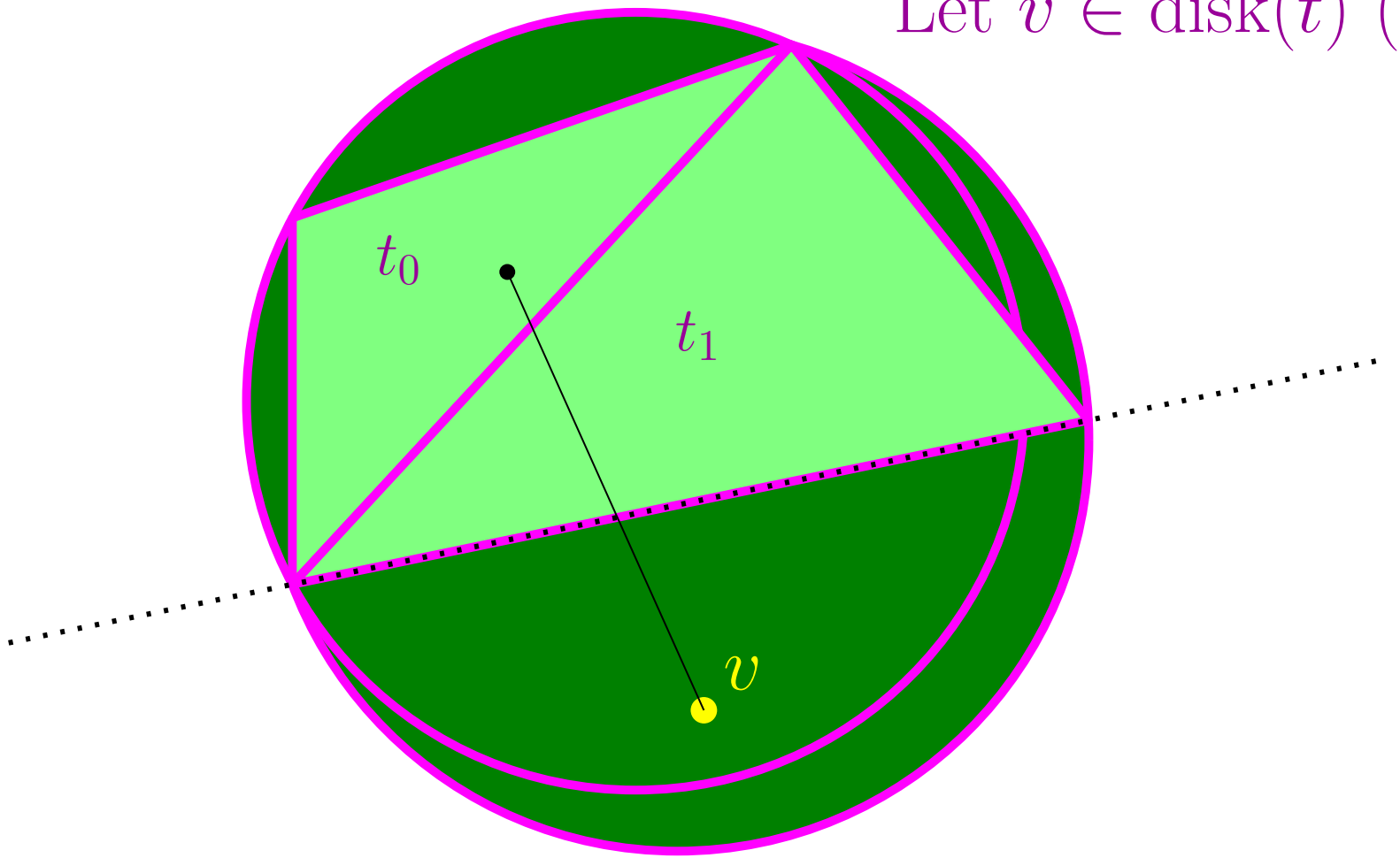
Let t_0 be locally Delaunay, but not globally Delaunay



Proof:

Let t_0 be locally Delaunay, but not globally Delaunay

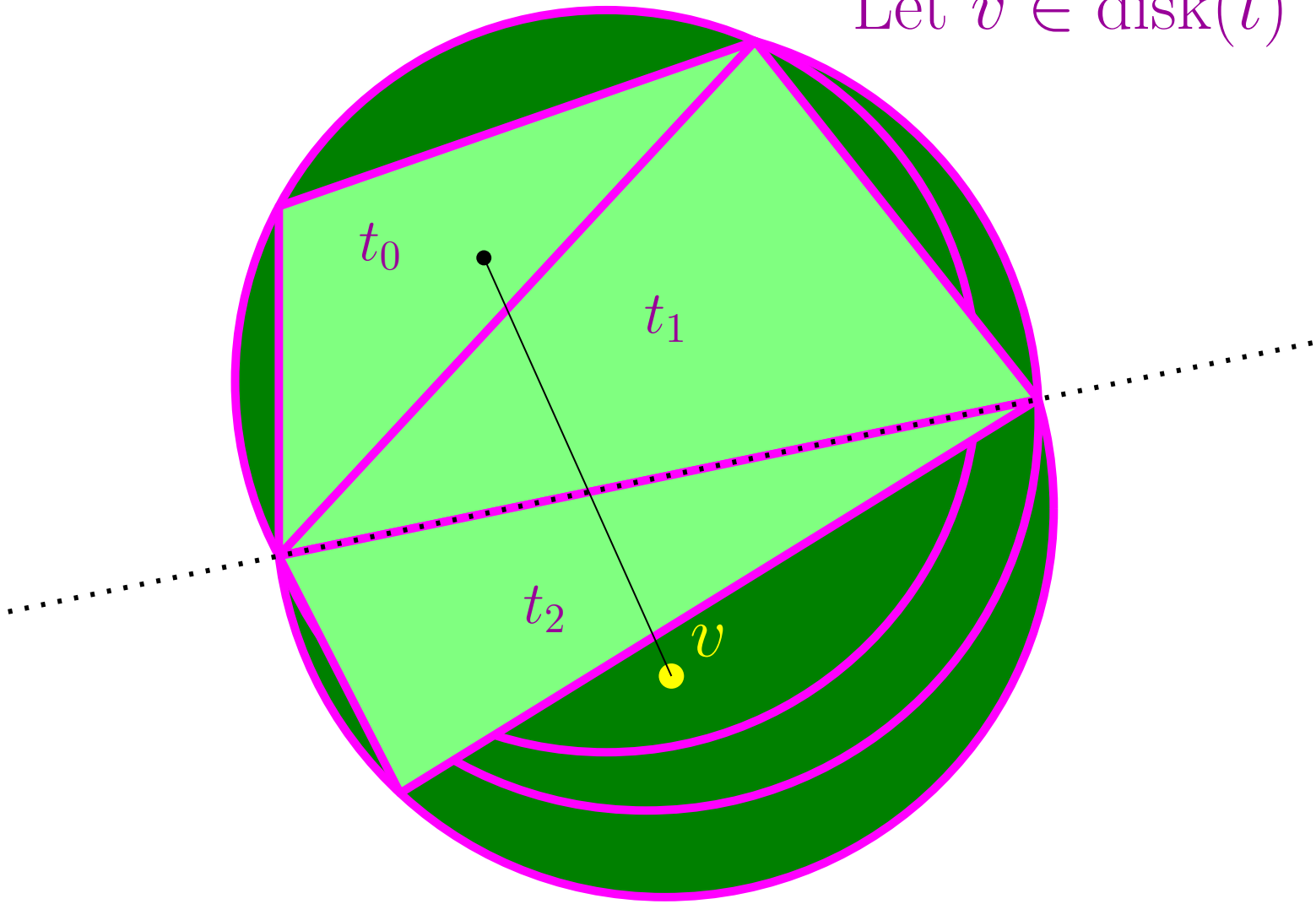
Let $v \in \text{disk}(t)$ $(v \notin t)$



Proof:

Let t_0 be locally Delaunay, but not globally Delaunay

Let $v \in \text{disk}(t)$ $(v \notin t)$



Since $\exists$ finitely many triangles, at some point v is a vertex of t_i

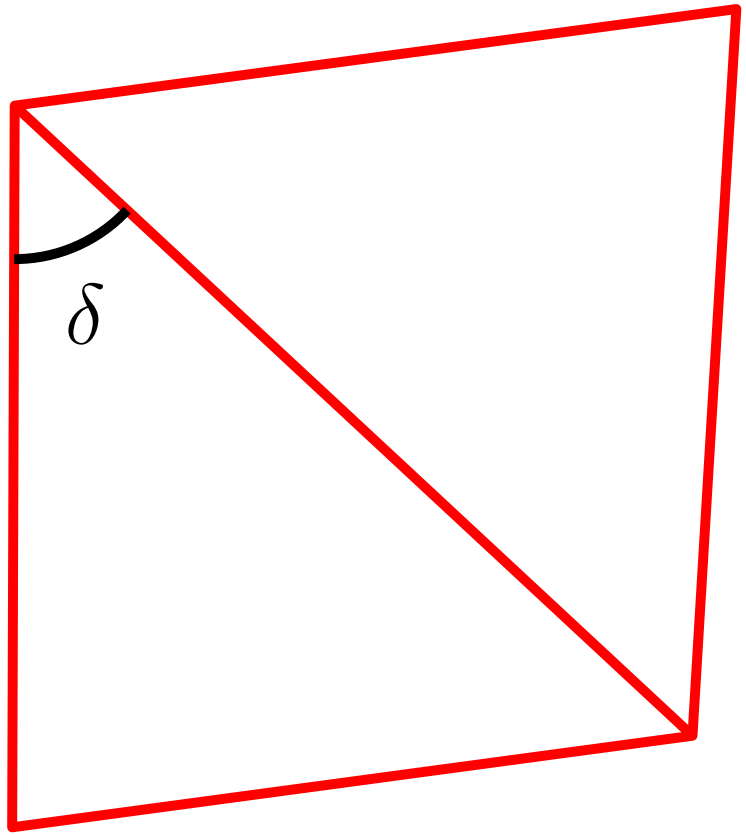
Local optimality and smallest angle

Case of 4 points

Lemma:
For any 4 points in convex position,
Delaunay $\Longleftrightarrow$ smallest angle maximized

Local optimality and smallest angle

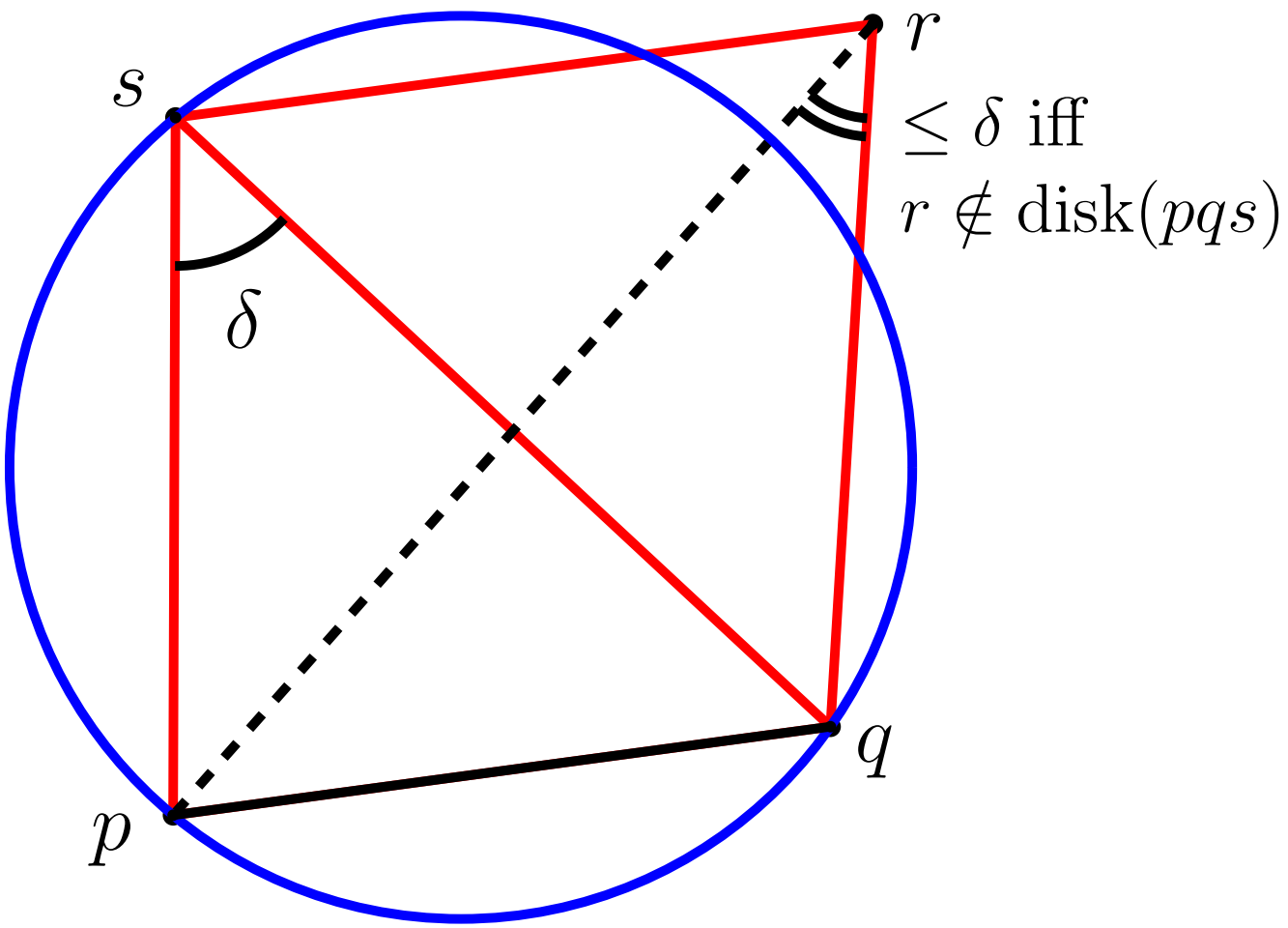
Case of 4 points



Let δ be the smallest angle

Local optimality and smallest angle

Case of 4 points



Let δ be the smallest angle

Algorithm for making a triangulation Delaunay

`while` $\exists$ pairs of adjacent triangles that are not locally Delaunay

pick an arbitrary pair and flip common edge

Algorithm for making a triangulation Delaunay

`while` $\exists$ pairs of adjacent triangles that are not locally Delaunay

pick an arbitrary pair and flip common edge

Theorem: whatever the choice of order on pairs, the algorithm terminates

$\rightarrow$ proof: each flip increases smallest angle in quad $\Rightarrow$ cannot be undone

$\rightarrow$ output is (globally) Delaunay

Algorithm for making a triangulation Delaunay

`while` $\exists$ pairs of adjacent triangles that are not locally Delaunay

pick an arbitrary pair and flip common edge

Theorem: whatever the choice of order on pairs, the algorithm terminates

$\rightarrow$ proof: each flip increases smallest angle in quad $\Rightarrow$ cannot be undone

$\rightarrow$ output is (globally) Delaunay

does not work in higher dimensions (several types of flips possible)

Local optimality and smallest angle

Theorem

Delaunay $\Rightarrow$ maximum smallest angle

Local optimality and smallest angle

Theorem

Delaunay $\Rightarrow$ maximum smallest angle

Proof: Let T triangulation

Local optimality and smallest angle

Theorem
Delaunay $\Rightarrow$ maximum smallest angle

Proof: Let T triangulation

Apply flipping algorithm on T

$\rightarrow$ output is Delaunay

Local optimality and smallest angle

Theorem
Delaunay $\Rightarrow$ maximum smallest angle

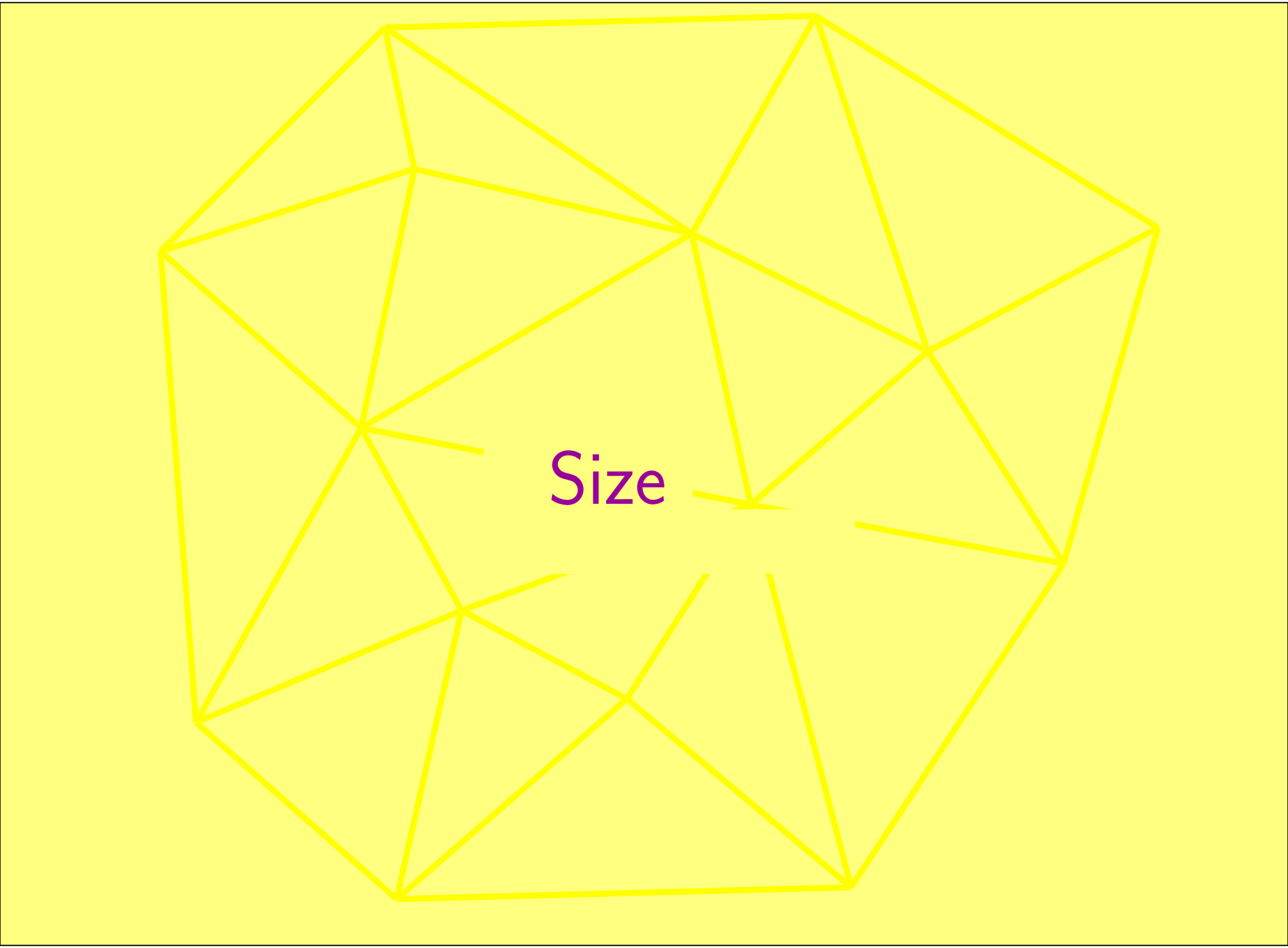
Proof: Let T triangulation

Apply flipping algorithm on T

$\rightarrow$ output is Delaunay

Each flip increases angles within quadrangle

$\rightarrow$ output has larger smallest angle


Size

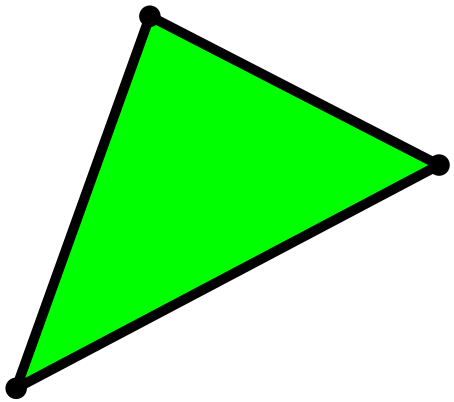
Euler formula

f: number of facets (except ∞)

e: number of edges

v: number of vertices

$$f - e + v = 1$$



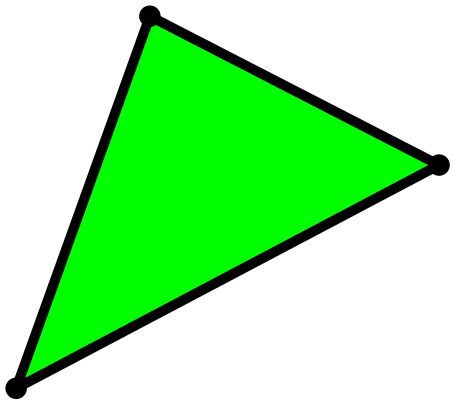
Euler formula

f: number of facets (except ∞)

e: number of edges

v: number of vertices

$$f - e + v = 1$$



$$1 - 3 + 3 = 1$$

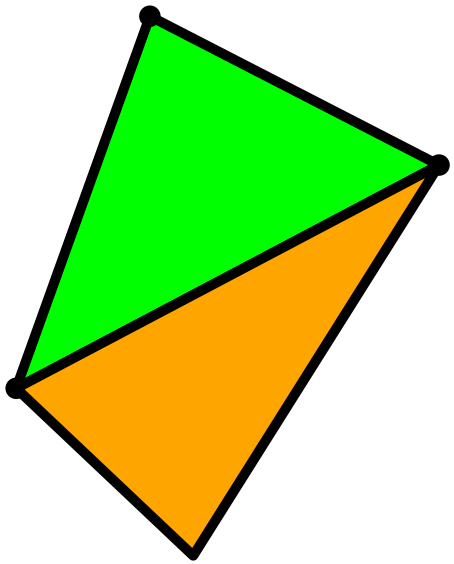
Euler formula

f: number of facets (except ∞)

e: number of edges

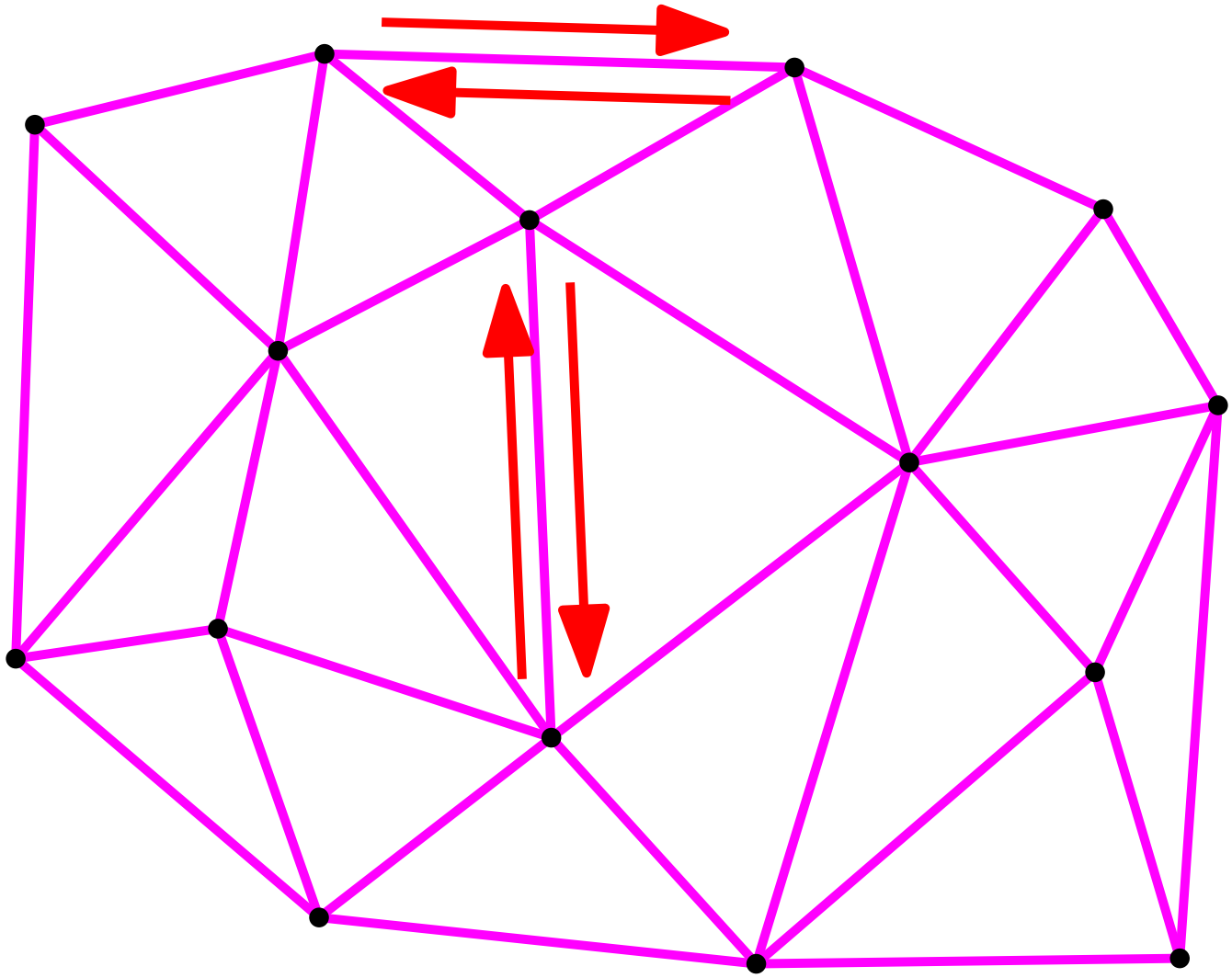
v: number of vertices

$$f - e + v = 1$$



$$+1 - 2 + 1 = +0$$

k: size of ∞ facet



number of oriented edges

in a triangulation: $2e = 3f + k$

Euler formula

$$f - e + v = 1$$

Triangulation

$$2e = 3f + k$$

$$f = 2v - 2 - k = O(v)$$
$$e = 3v - 3 - k = O(v)$$

Euler formula

$$f - e + v = 1$$

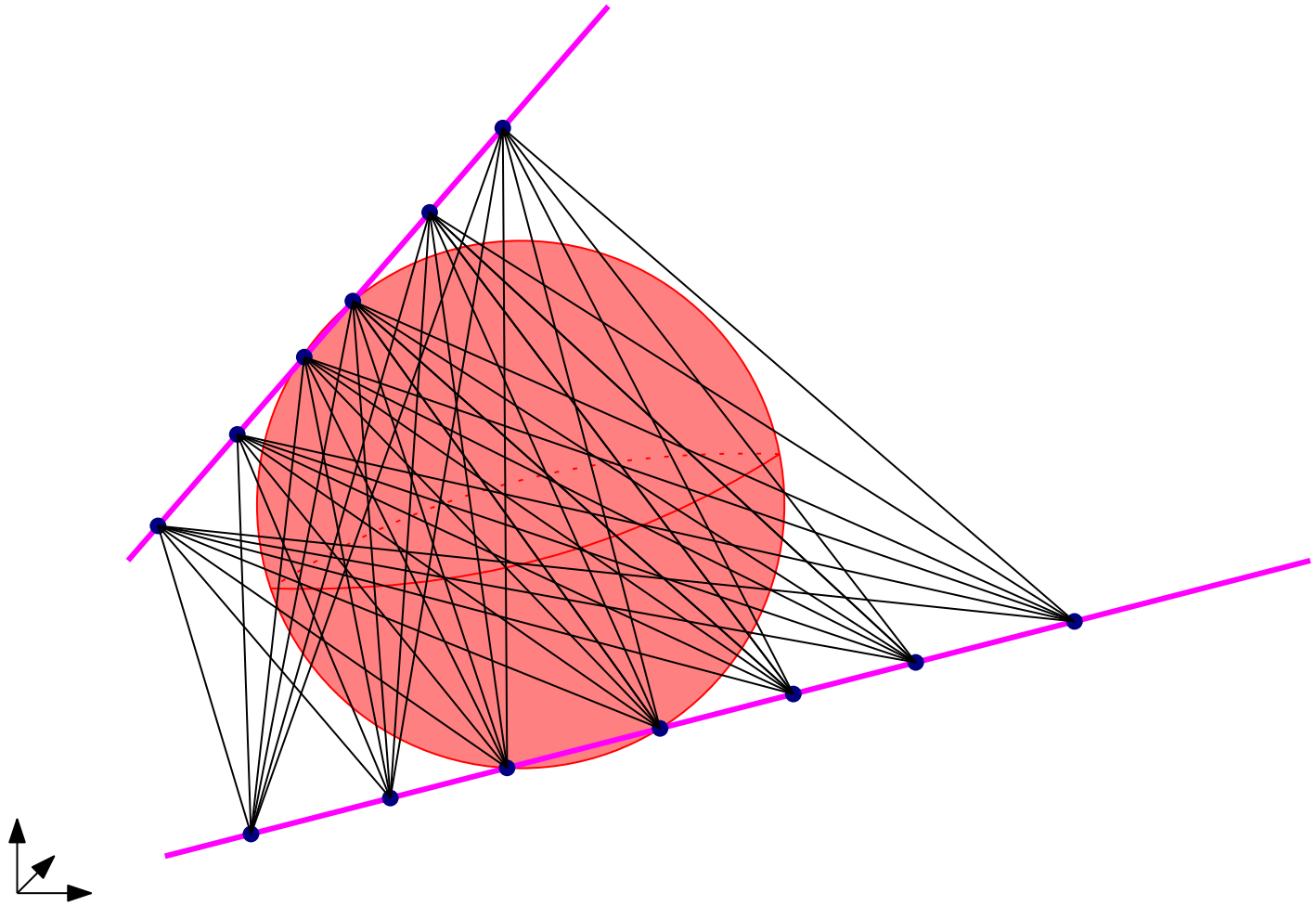
Triangulation

$$2e = 3f + k$$

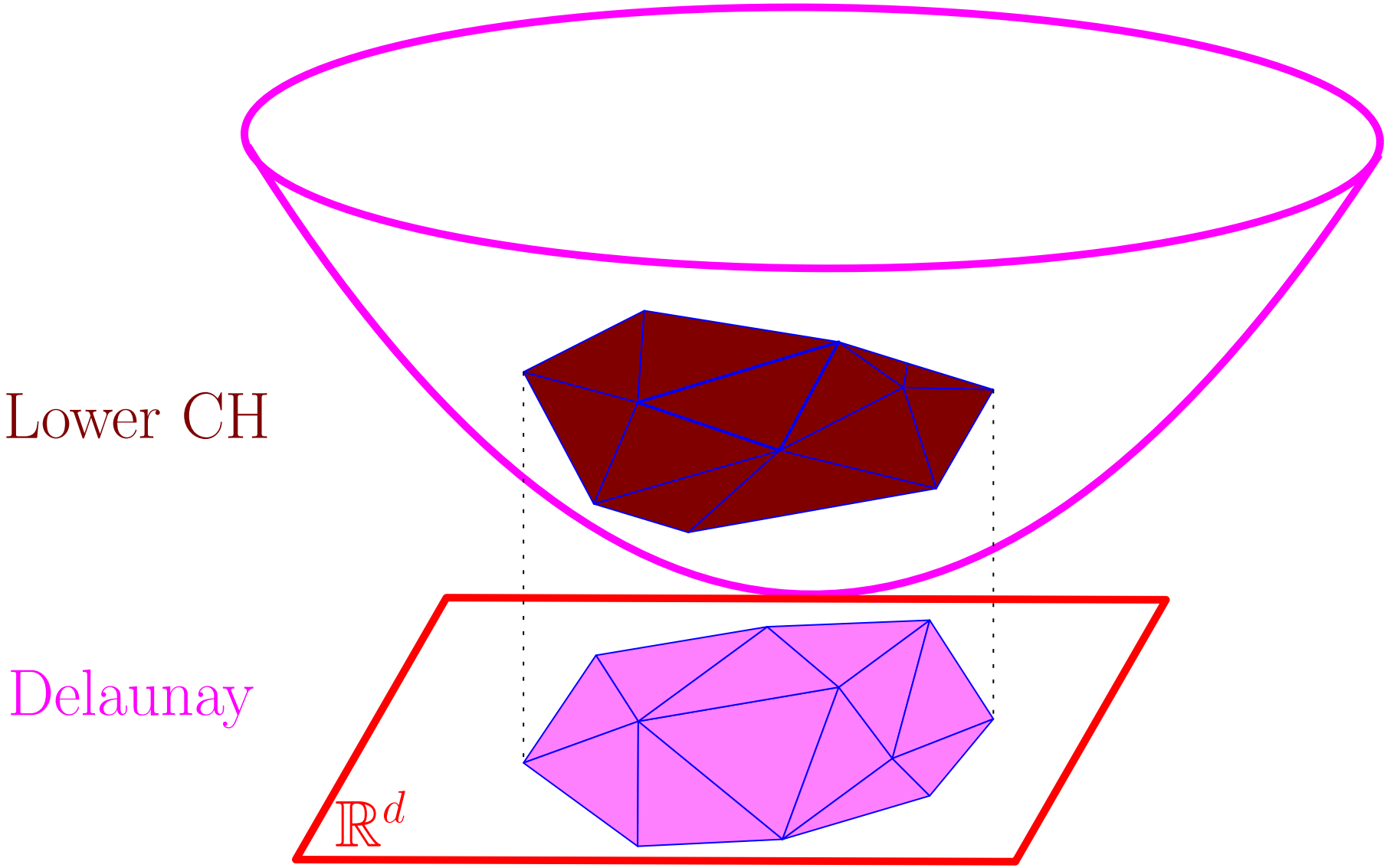
2D Delaunay has linear size

$$f = 2v - 2 - k = O(v)$$
$$e = 3v - 3 - k = O(v)$$

3D Delaunay can have quadratic size



point / sphere lifting



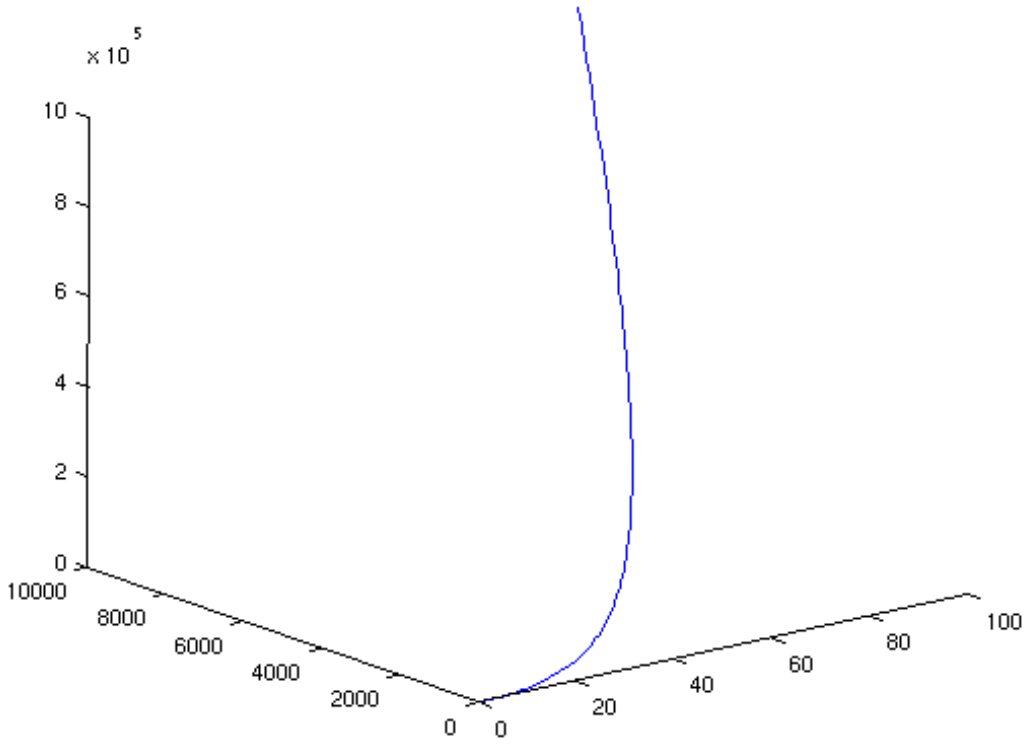
Size of Delaunay in $\mathbb{R}^d$

- By point/sphere lifting, $|\mathrm{Del}(P)| = |\mathrm{Conv}(P^*)| = O(|P|^{\lfloor \frac{d+1}{2} \rfloor}) = O(|P|^{\lceil \frac{d}{2} \rceil})$

Size of Delaunay in $\mathbb{R}^d$

- By point/sphere lifting, $|\mathrm{Del}(P)| = |\mathrm{Conv}(P^*)| = O(|P|^{\lfloor \frac{d+1}{2} \rfloor}) = O(|P|^{\lceil \frac{d}{2} \rceil})$

- When d is even, point set P on moments curve $t \mapsto (t, t^2, t^3, \cdots, t^d)$ yields $|\mathrm{Del}(P)| \geq |\mathrm{Conv}(P)| = \Omega(|P|^{\lfloor \frac{d}{2} \rfloor}) = \Omega(|P|^{\lceil \frac{d}{2} \rceil})$.

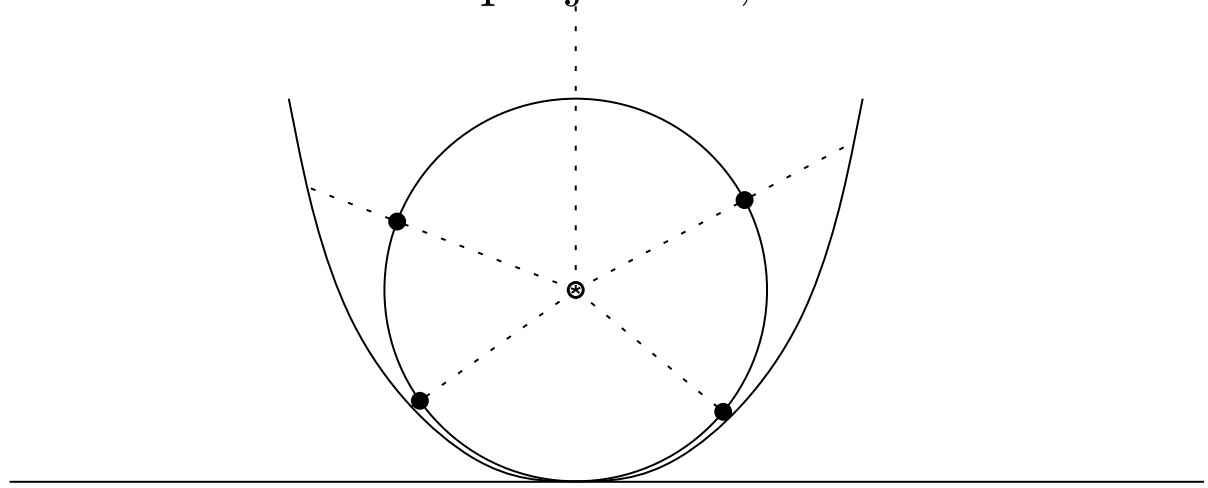


Size of Delaunay in $\mathbb{R}^d$

- By point/sphere lifting, $|\mathrm{Del}(P)| = |\mathrm{Conv}(P^*)| = O(|P|^{\lfloor \frac{d+1}{2} \rfloor}) = O(|P|^{\lceil \frac{d}{2} \rceil})$

- When d is even, point set P on moments curve $t \mapsto (t, t^2, t^3, \cdots, t^d)$ yields $|\mathrm{Del}(P)| \geq |\mathrm{Conv}(P)| = \Omega(|P|^{\lfloor \frac{d}{2} \rfloor}) = \Omega(|P|^{\lceil \frac{d}{2} \rceil})$.

- When d is odd, take point set P^* on trigonometric curve $t \mapsto \frac{2}{d+1}(\cos t, \sin t, \cos 2t, \sin 2t, \cdots, \cos \frac{d+1}{2}t, \sin \frac{d+1}{2}t) \in \mathbb{S}^d \subset \mathbb{R}^{d+1}$ yields $|\mathrm{Conv}(P^*)| = \Omega(|P^*|^{\lfloor \frac{d+1}{2} \rfloor}) = \Omega(|P^*|^{\lceil \frac{d}{2} \rceil})$.
$\rightarrow$ map P^* onto unit paraboloid via radial projection, then down to $P \subset \mathbb{R}^d$.



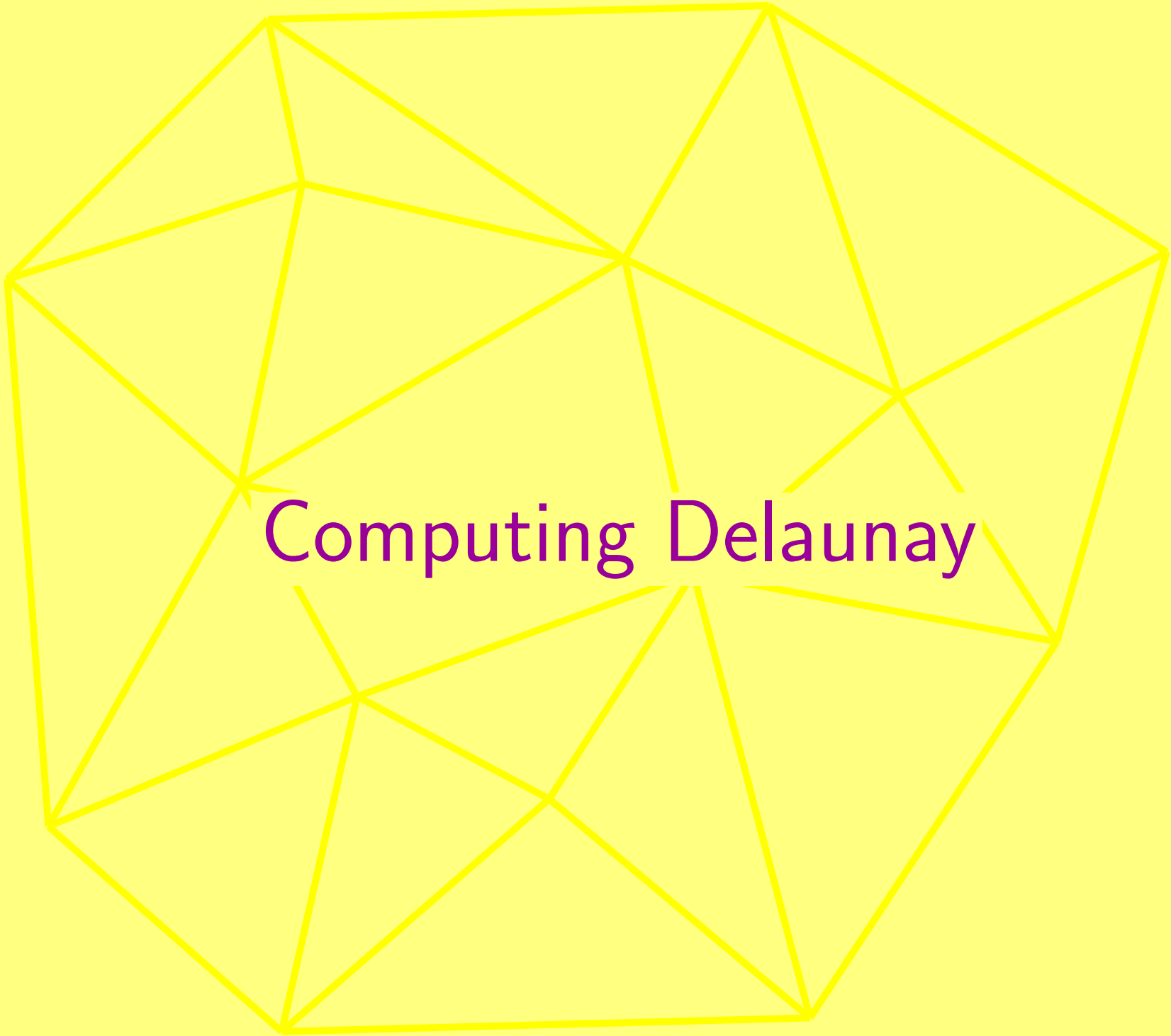
Size of Delaunay in $\mathbb{R}^d$

- By point/sphere lifting, $|\mathrm{Del}(P)| = |\mathrm{Conv}(P^*)| = O(|P|^{\lfloor \frac{d+1}{2} \rfloor}) = O(|P|^{\lceil \frac{d}{2} \rceil})$

- When d is even, point set P on moments curve $t \mapsto (t, t^2, t^3, \cdots, t^d)$ yields $|\mathrm{Del}(P)| \geq |\mathrm{Conv}(P)| = \Omega(|P|^{\lfloor \frac{d}{2} \rfloor}) = \Omega(|P|^{\lceil \frac{d}{2} \rceil})$.

- When d is odd, take point set P^* on trigonometric curve $t \mapsto \frac{2}{d+1}(\cos t, \sin t, \cos 2t, \sin 2t, \cdots, \cos \frac{d+1}{2}t, \sin \frac{d+1}{2}t) \in \mathbb{S}^d \subset \mathbb{R}^{d+1}$ yields $|\mathrm{Conv}(P^*)| = \Omega(|P^*|^{\lfloor \frac{d+1}{2} \rfloor}) = \Omega(|P^*|^{\lceil \frac{d}{2} \rceil})$.
$\rightarrow$ map P^* onto unit paraboloid via radial projection, then down to $P \subset \mathbb{R}^d$.

Size of Delaunay of n points in $\mathbb{R}^d$: $\Theta\left(n^{\lceil \frac{d}{2} \rceil}\right)$


Computing Delaunay

Computing the Delaunay triangulation

1. Lift P to $\mathbb{R}^{d+1}$ and compute lower convex hull there

$\rightarrow$ direct extension of Graham's algorithm ([H.-P. Seidel]): $O(n^{\lceil \frac{d+1}{2} \rceil} + n \log n)$

$\rightarrow$ randomized incremental algorithm ([Clarkson, Shor]): exp. $O(n^{\lceil \frac{d}{2} \rceil} + n \log n)$

$\rightarrow$ de-randomized incremental algorithm ([Chazelle]): $O(n^{\lceil \frac{d}{2} \rceil} + n \log n)$

Computing the Delaunay triangulation

1. Lift P to $\mathbb{R}^{d+1}$ and compute lower convex hull there

2. Incremental algorithm ([Boissonnat et al.])

   $\rightarrow O(n^{\lceil \frac{d+1}{2} \rceil} + n \log n)$ with deterministic point insertion order

   $\rightarrow$ exp. $O(n^{\lceil \frac{d}{2} \rceil} + n \log n)$ with randomized point insertion order

Computing the Delaunay triangulation

1. Lift P to $\mathbb{R}^{d+1}$ and compute lower convex hull there

2. Incremental algorithm ([Boissonnat et al.])

3. Divide-and-conquer algorithm [Guibas, Stolfi]

   $\rightarrow$ only in the plane or in 3-space

   $\rightarrow$ optimal $O(n \log n)$ in the plane and $O(n^2)$ in $\mathbb{R}^3$

Computing the Delaunay triangulation

1. Lift P to $\mathbb{R}^{d+1}$ and compute lower convex hull there

2. Incremental algorithm ([Boissonnat et al.])

3. Divide-and-conquer algorithm [Guibas, Stolfi]

4. Plane-sweep algorithm [Fortune]

   $\rightarrow$ in the plane only

   $\rightarrow$ computes Voronoi diagram

   $\rightarrow$ optimal $O(n \log n)$ time

Computing the Delaunay triangulation

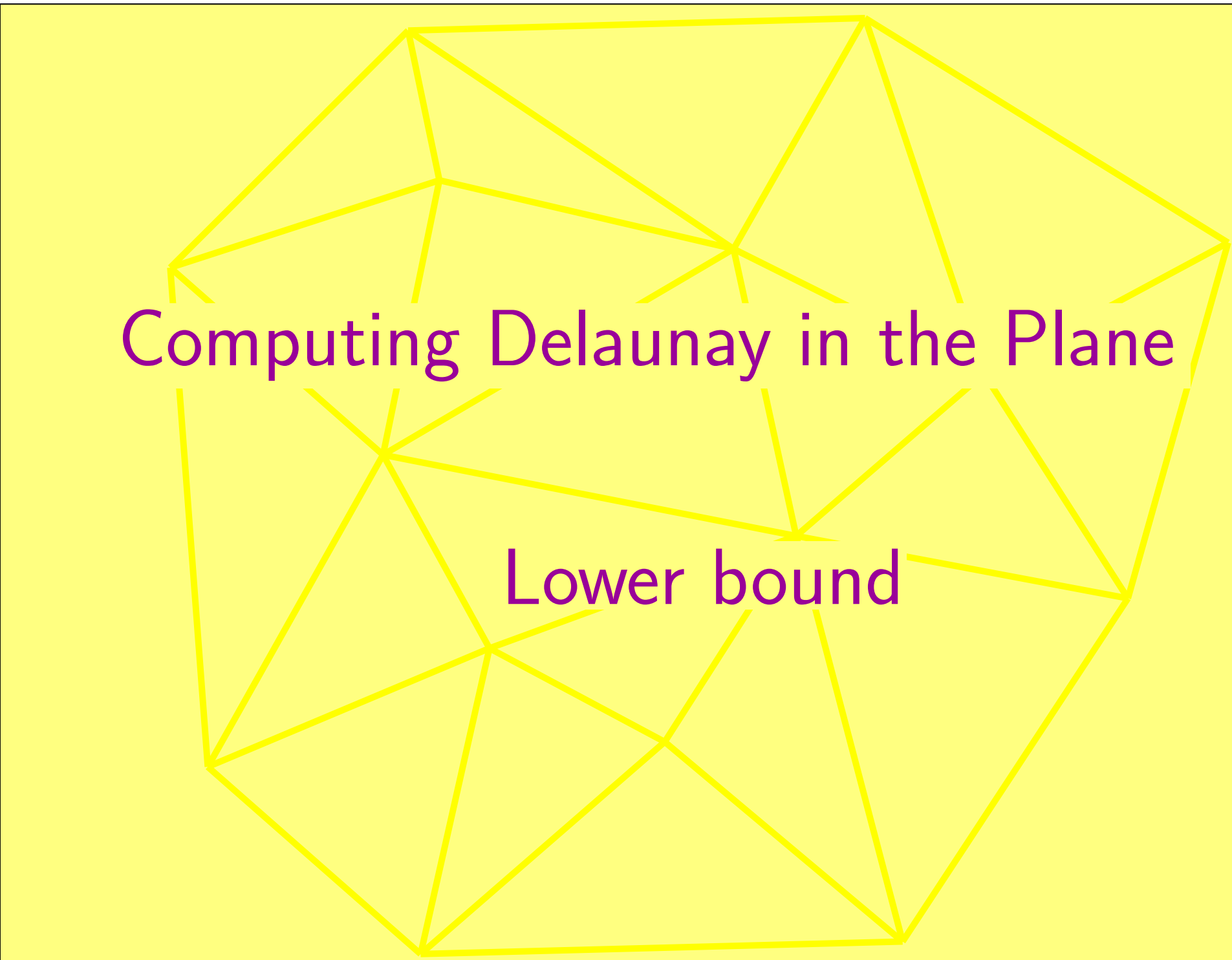
1. Lift P to $\mathbb{R}^{d+1}$ and compute lower convex hull there

2. Incremental algorithm ([Boissonnat et al.])

3. Divide-and-conquer algorithm [Guibas, Stolfi]

(today)

4. Plane-sweep algorithm [Fortune]



Computing Delaunay in the Plane

Lower bound

Lower bound for Delaunay

Delaunay can be used to sort numbers

Lower bound for Delaunay

Delaunay can be used to sort numbers

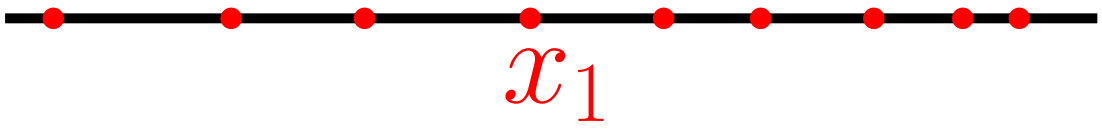
Take an instance of sort

Assume one can compute Delaunay in $\mathbb{R}^2$

Use Delaunay to solve this instance of sort

Lower bound for Delaunay

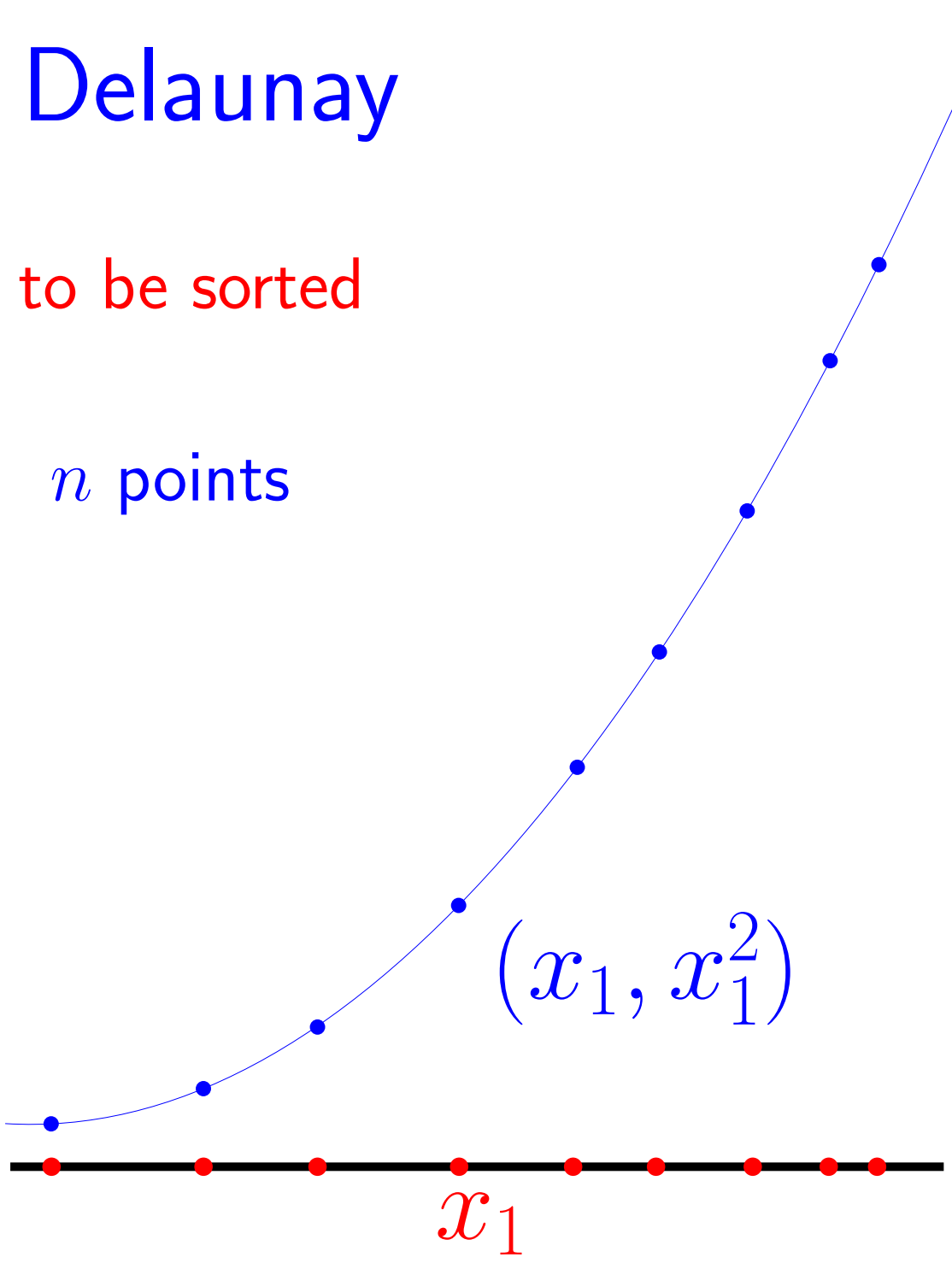
Let $x_1, x_2, \ldots, x_n \in \mathbb{R}$, to be sorted



Lower bound for Delaunay

Let $x_1, x_2, \ldots, x_n \in \mathbb{R}$, to be sorted

$(x_1, x_1^2), \ldots, (x_n, x_n^2)$ n points



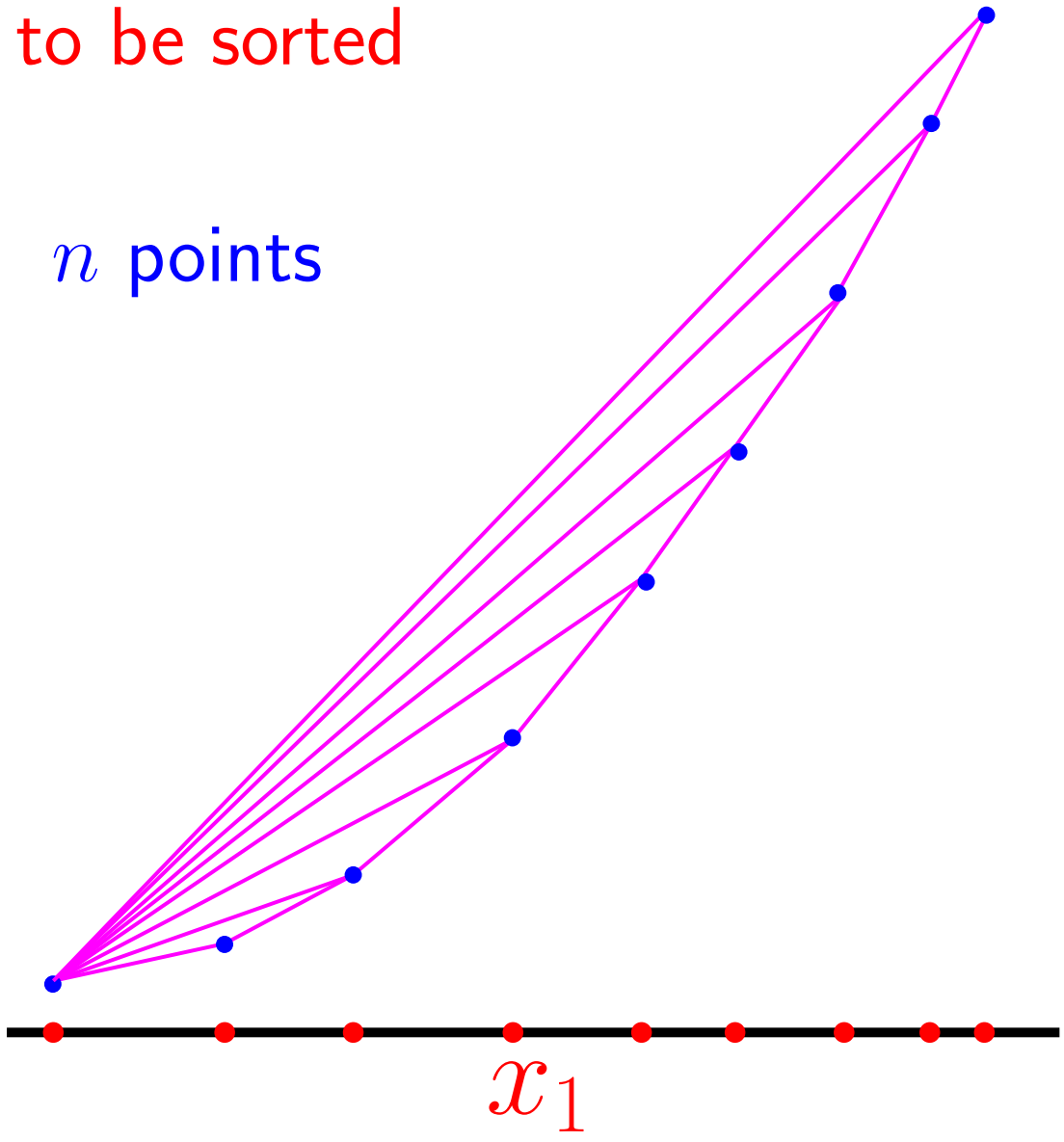
Lower bound for Delaunay

Let $x_1, x_2, \ldots, x_n \in \mathbb{R}$, to be sorted

$(x_1, x_1^2), \ldots, (x_n, x_n^2)$ n points

Delaunay

$\rightarrow$ order in x



Lower bound for Delaunay

Let $x_1, x_2, \ldots, x_n \in \mathbb{R}$, to be sorted

n points $\downarrow O(n)$

$(x_1, x_1^2), \ldots, (x_n, x_n^2)$

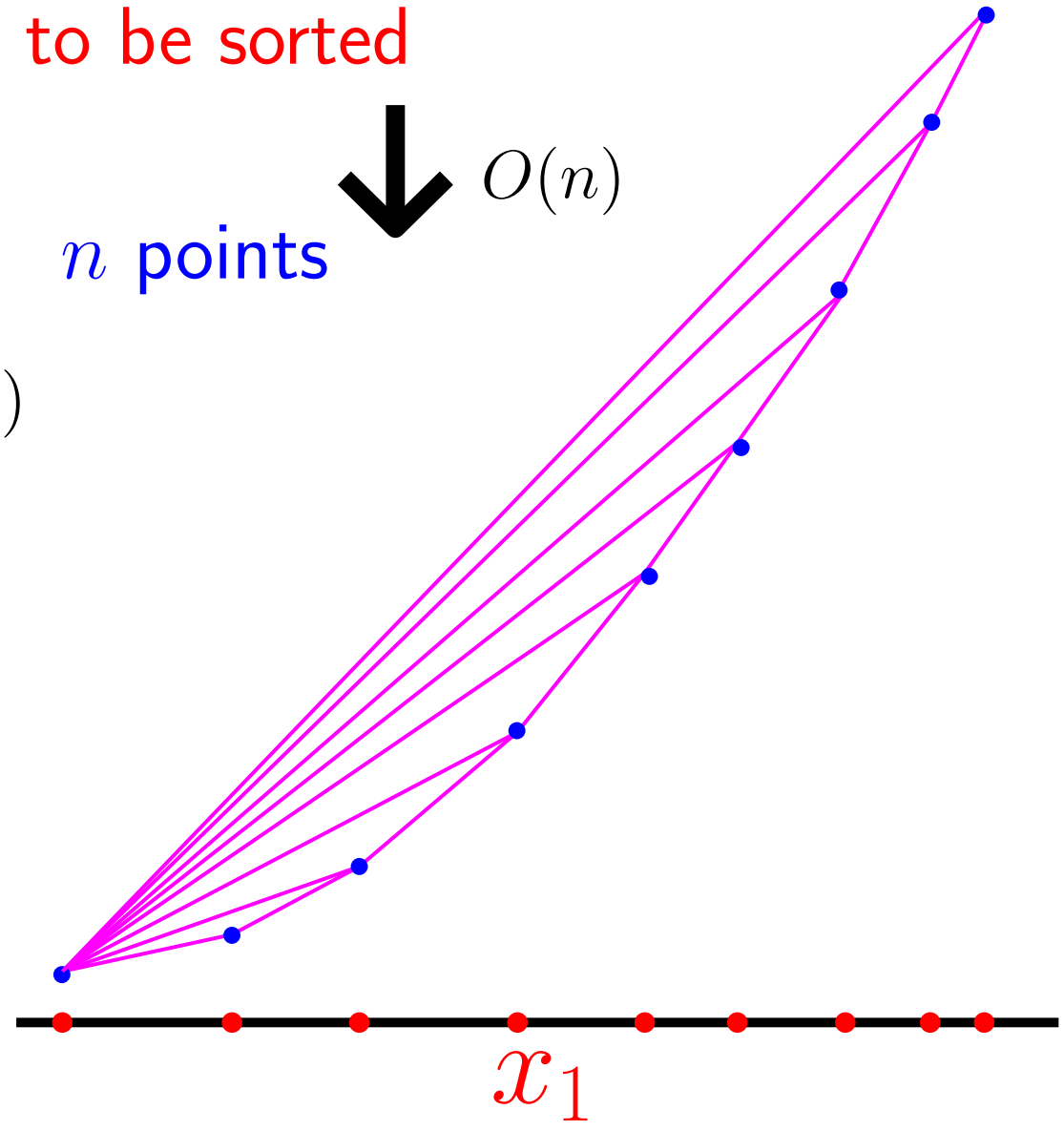
$\downarrow f(n)$

Delaunay

$\downarrow O(n)$

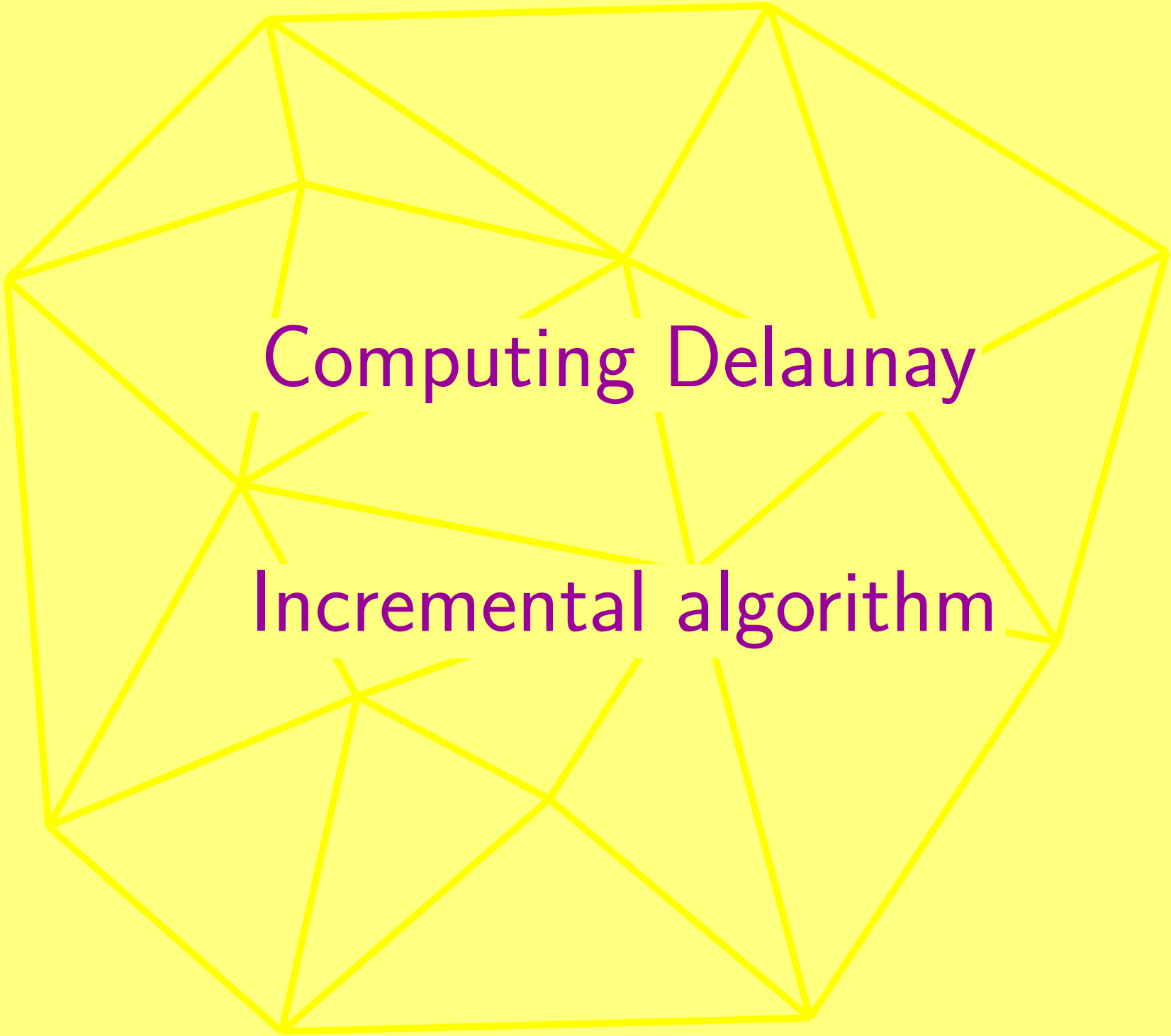
$\rightarrow$ order in x

$O(n) + f(n) \in \Omega(n \log n)$

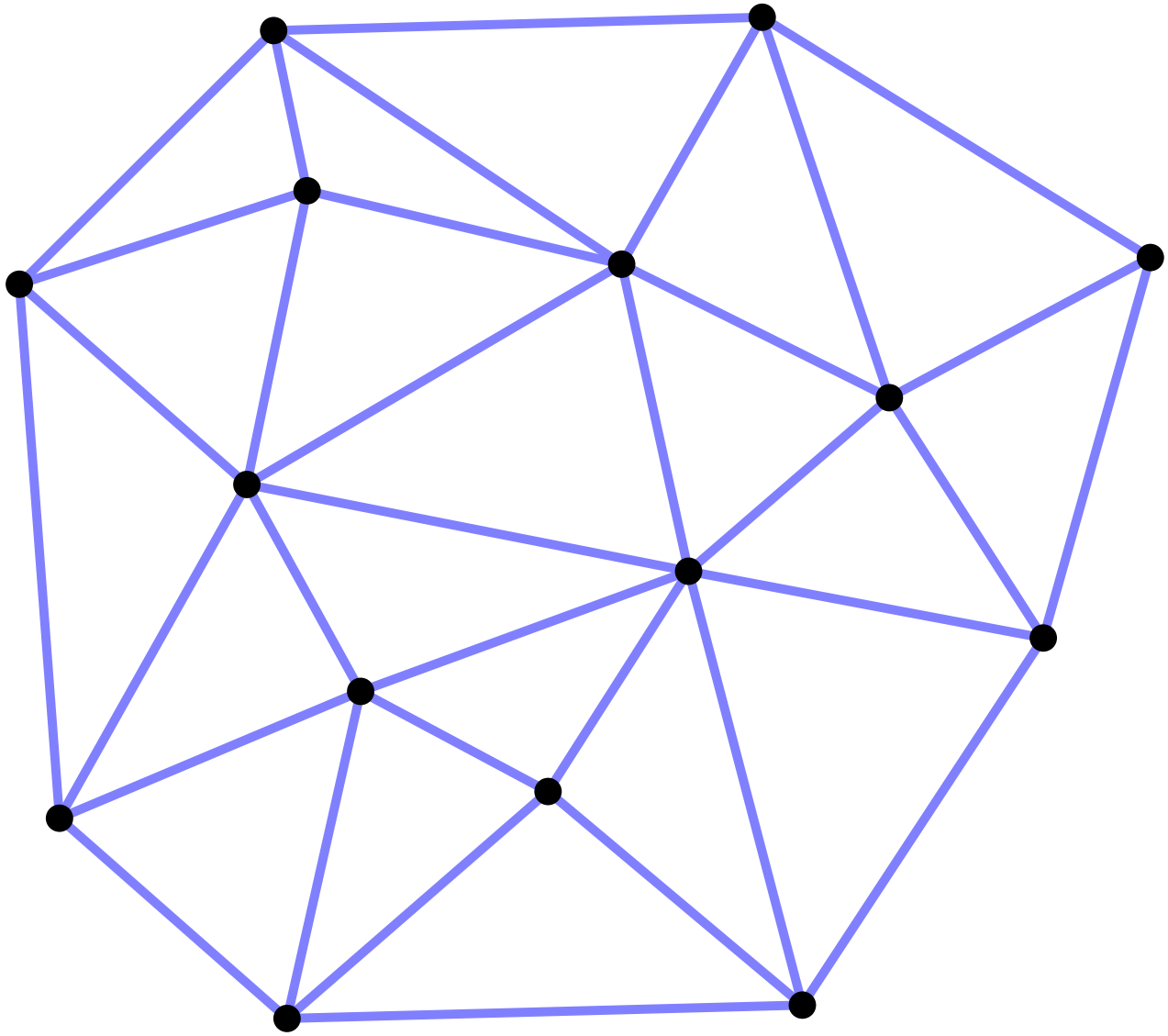


Lower bound for Delaunay

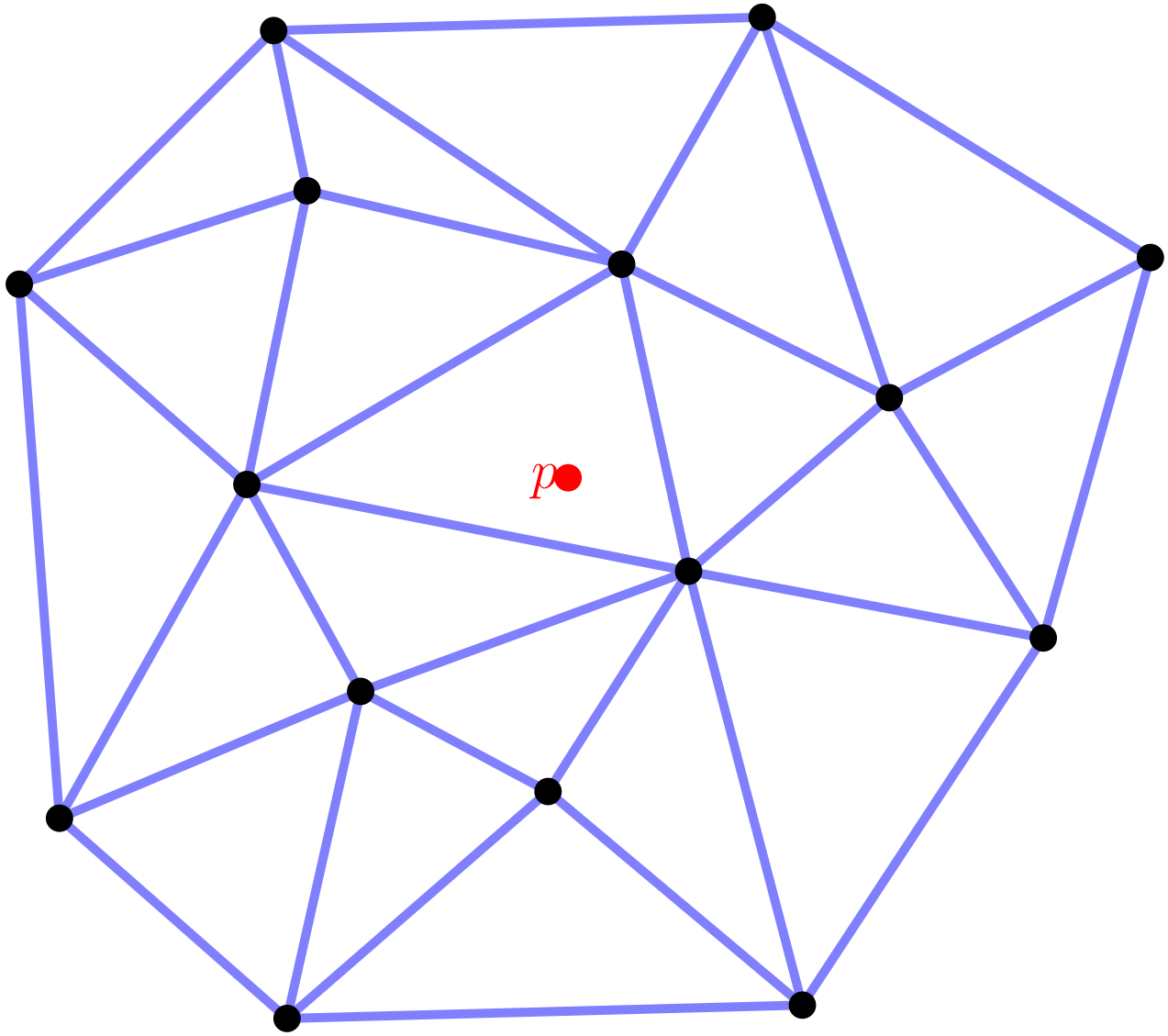
$$\Rightarrow f(n) \in \Omega(n \log n)$$


Computing Delaunay
Incremental algorithm

Algorithm overview

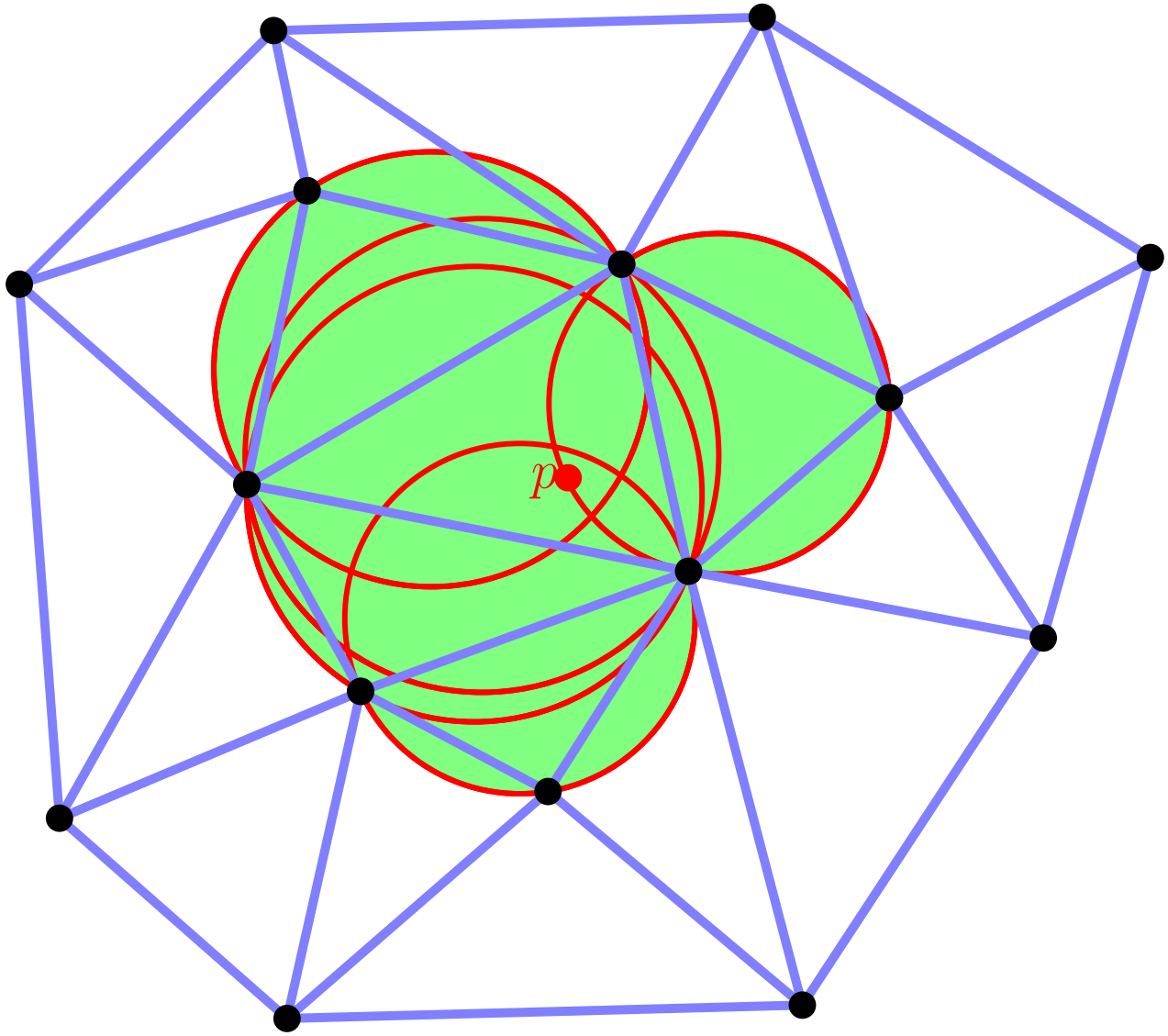


Algorithm overview



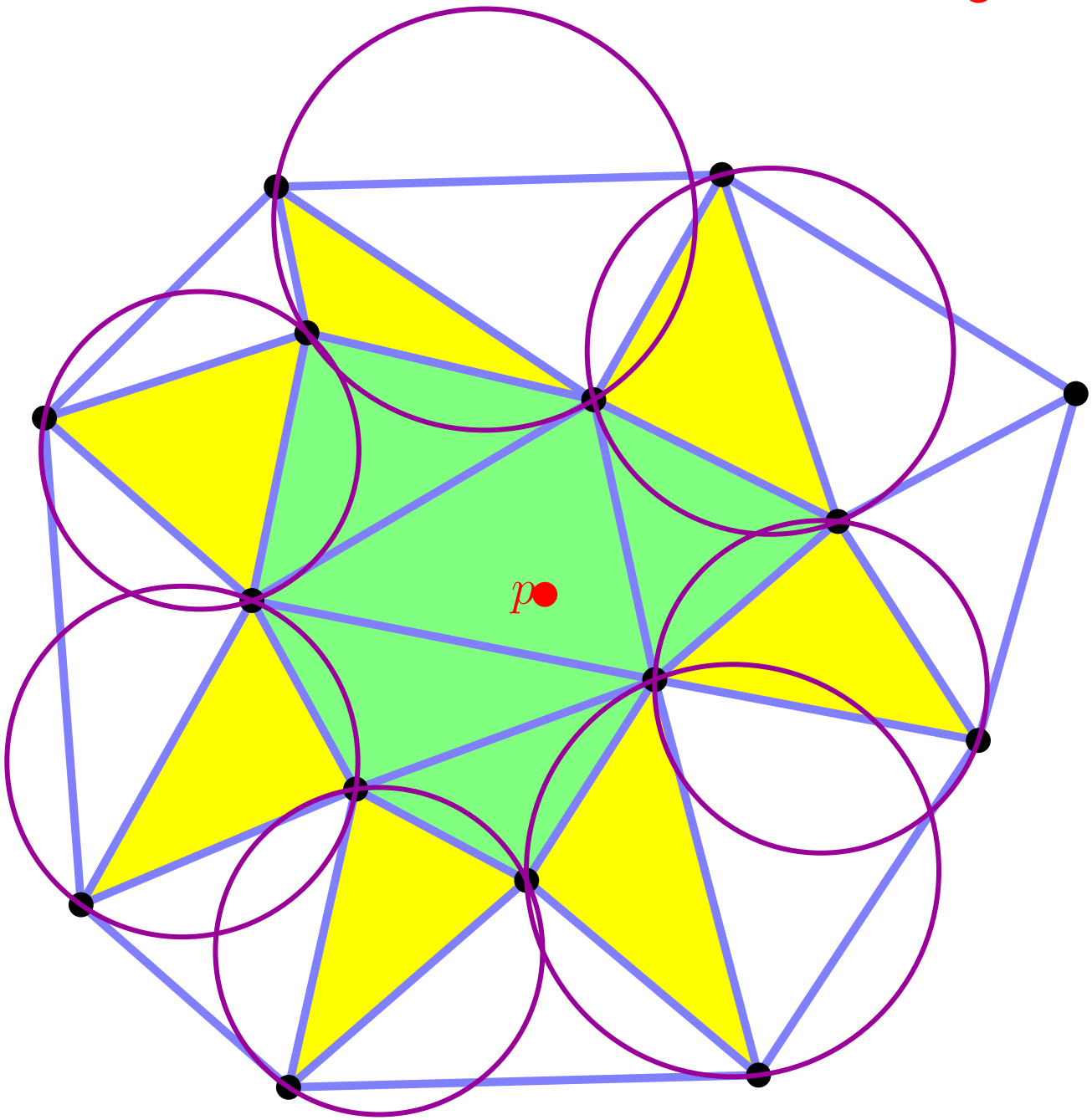
Algorithm overview

- Find triangles in conflict with p



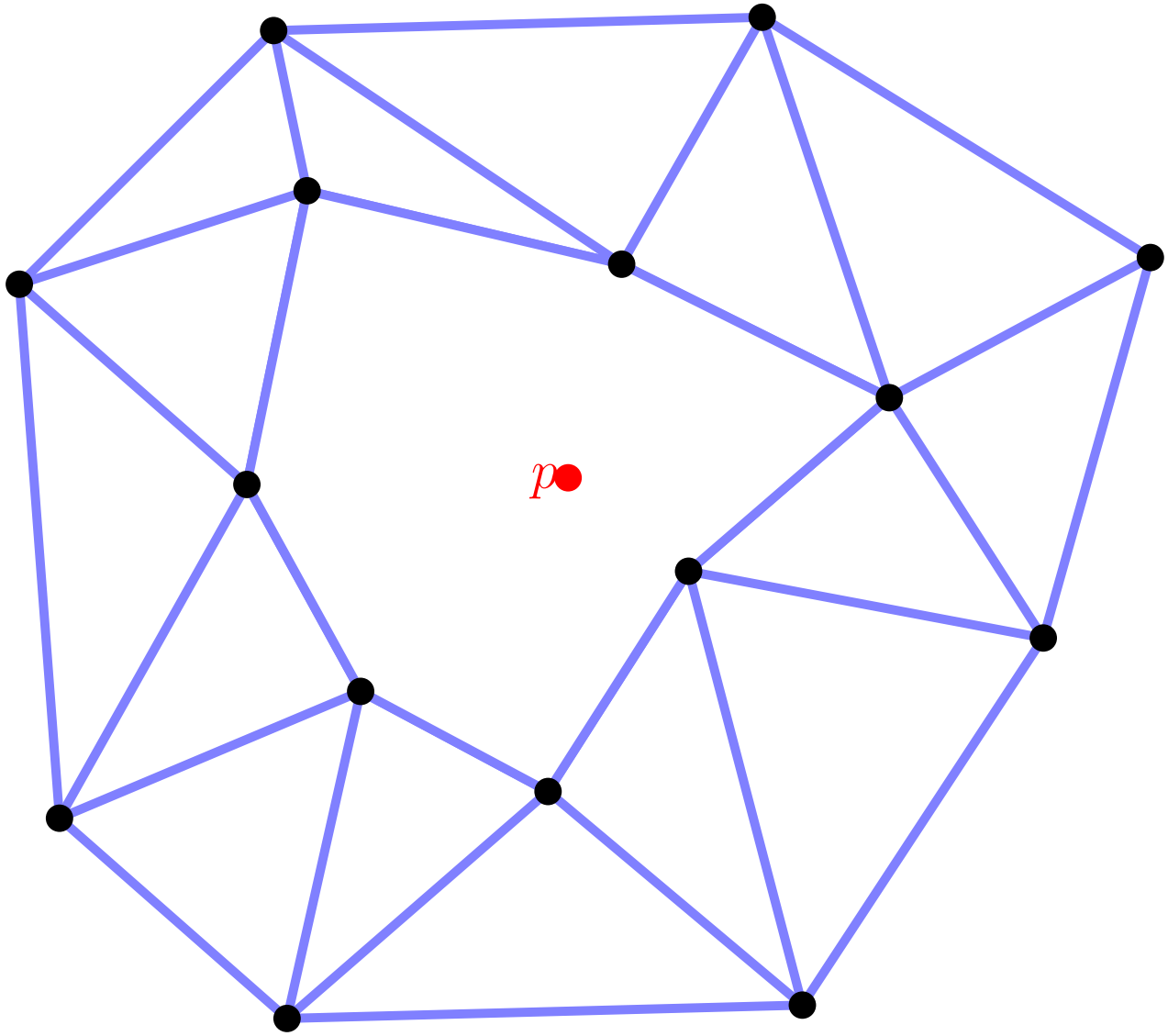
Algorithm overview

- Find triangles in conflict with p



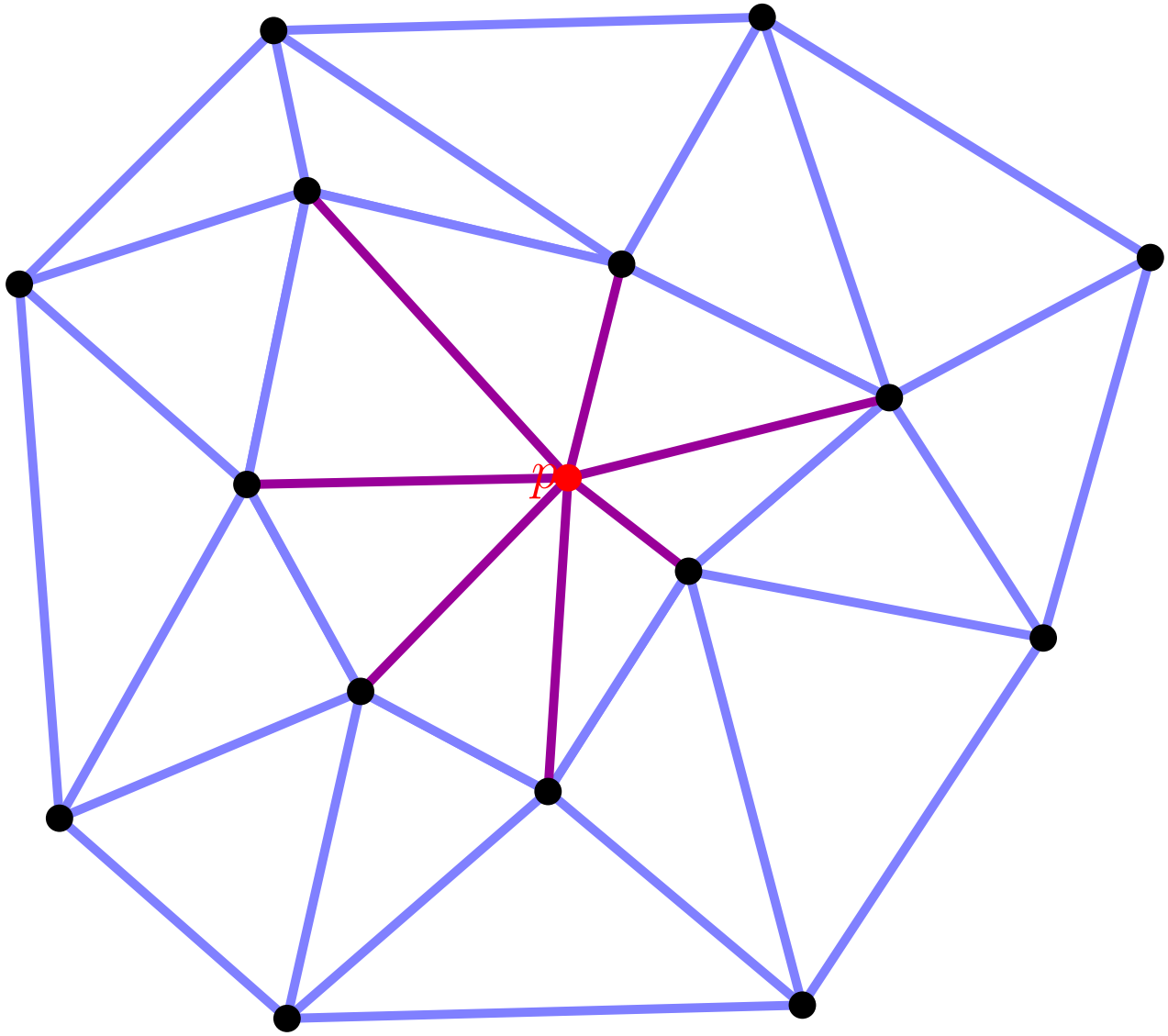
Algorithm overview

- Find triangles in conflict with p
- Delete triangles in conflict



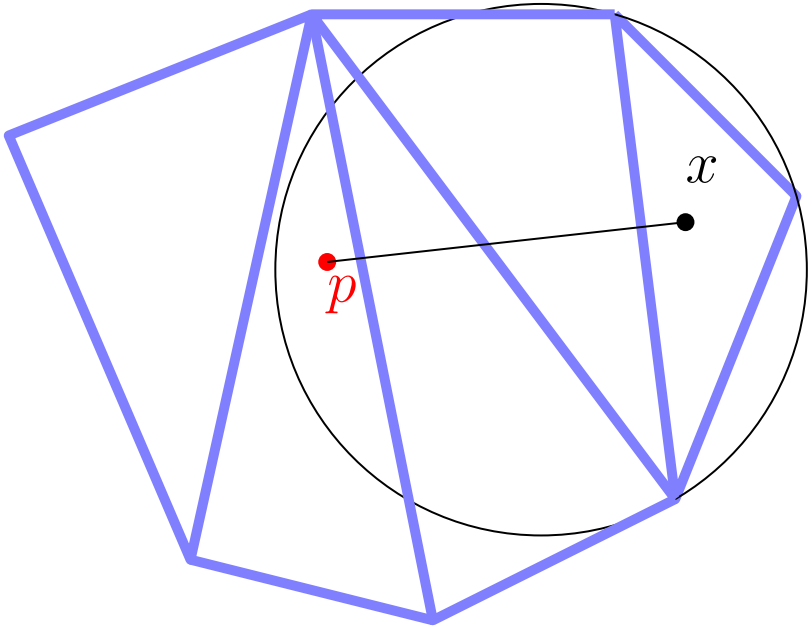
Algorithm overview

- Find triangles in conflict with p
- Delete triangles in conflict
- Re-triangulate hole w.r.t. p



Why it works

Property 1: the conflict zone is starred with respect to p (hence connected)



$\forall x \in$ conflict zone, all triangles intersected by $[p, x]$ are in conflict with p

(same proof as for locally Del. $\Rightarrow$ globally Del.)

Why it works

Property 1: the conflict zone is starred with respect to p (hence connected)

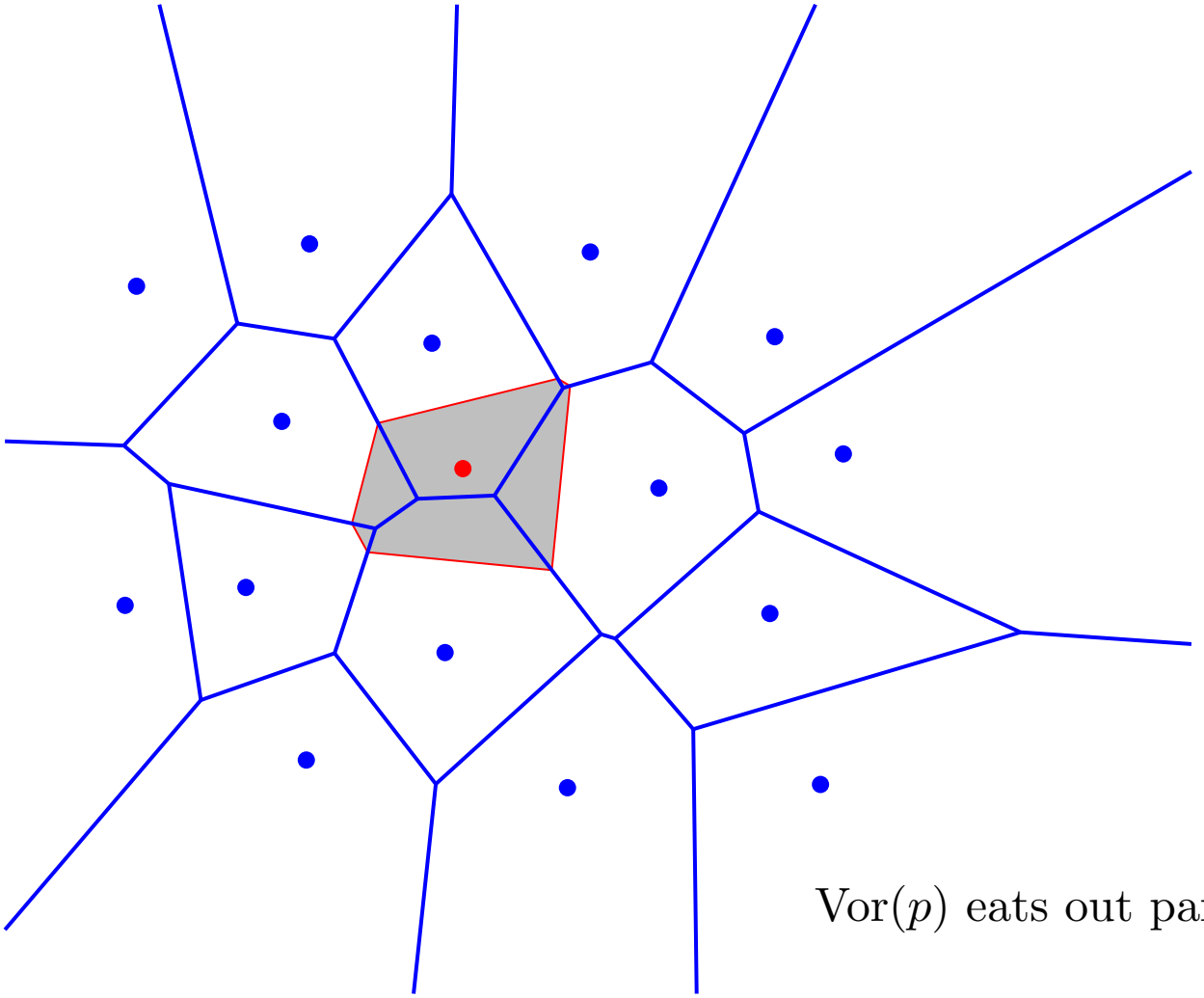
$\rightarrow$ can be computed by a traversal in the dual graph from some $\sigma \ni p$

$\rightarrow$ can be re-triangulated by join products $p * \sigma$ for each σ on its boundary

Why it works

Property 3: every new Delaunay simplex is incident to p

$\rightarrow$ re-triangulation by join products with p is Delaunay



Vor(p) eats out parts of the other Voronoi regions

Complexity analysis

n points $\Rightarrow$ n insertions, each of which is composed of:

- locate: $O(n)$ naive, $O(n^{1/d})$ with random line walk, $O(\log n)$ with hierarchy.
- bfs in conflict zone: $O(d_i)$, where d_i is the number of deleted cells at i-th iteration.
- star conflict zone: $O(c_i)$, where c_i is the number of created cells at i-th iteration.

$\Rightarrow$ total complexity $= O(n \log n + \sum_{i=1}^{n}(c_i + d_i)$

Complexity analysis

n points $\Rightarrow$ n insertions, each of which is composed of:

- locate: $O(n)$ naive, $O(n^{1/d})$ with random line walk, $O(\log n)$ with hierarchy.
- bfs in conflict zone: $O(d_i)$, where d_i is the number of deleted cells at i-th iteration.
- star conflict zone: $O(c_i)$, where c_i is the number of created cells at i-th iteration.

$\Rightarrow$ total complexity $= O(n \log n + \sum_{i=1}^{n}(c_i + d_i)$

boundary of conflicts zone is homeomorphic to a $(d-1)$-sphere since the conflict zone is starred w.r.t. $p \Rightarrow c_i, d_i = O(i^{\lceil \frac{d-1}{2} \rceil})$ by a variant of Upper Bound Theorem [Stanley 75].

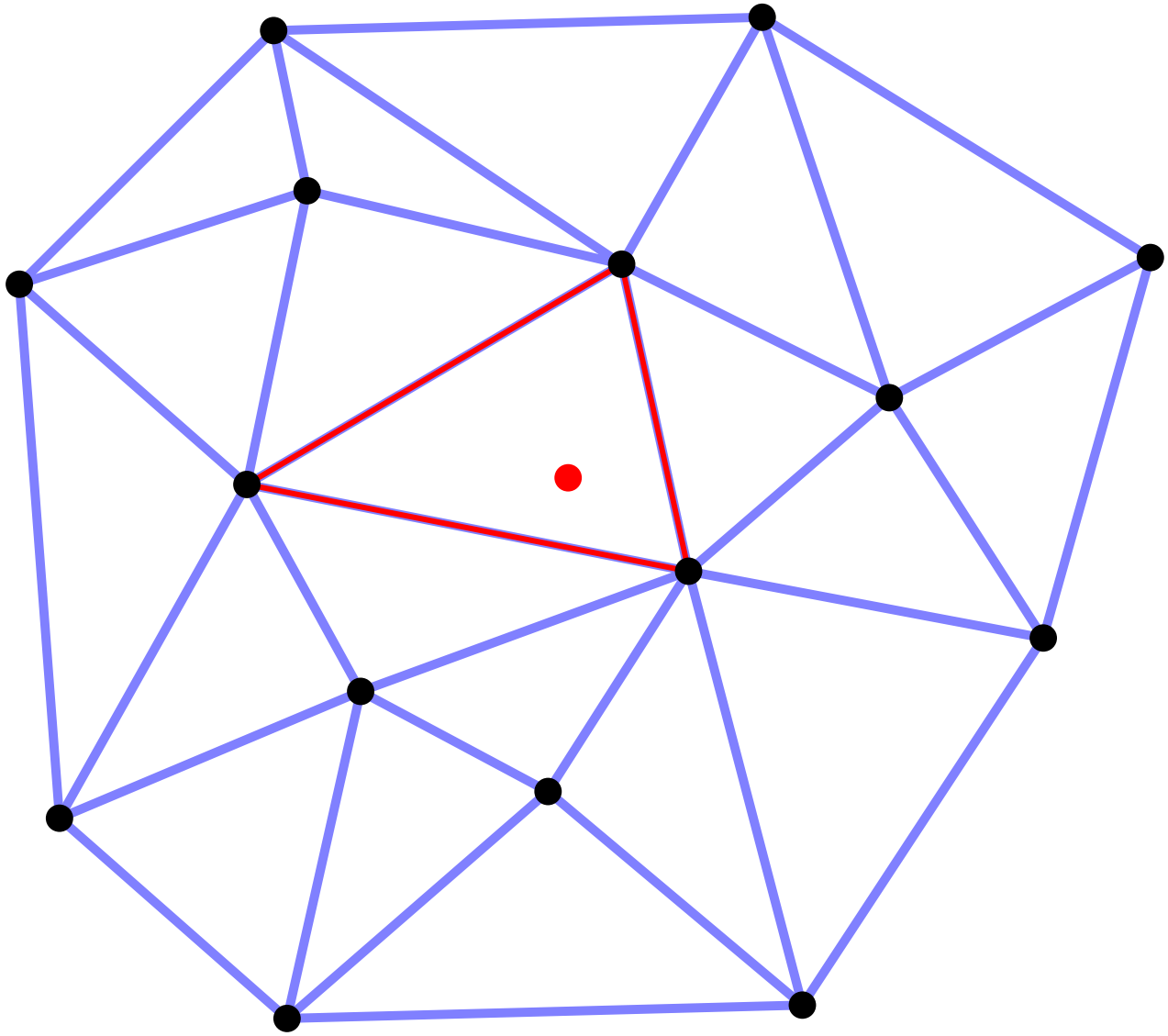
$$\Rightarrow \text{total complexity} = O(n \log n + n^{\lceil \frac{d+1}{2} \rceil})$$

(sub-optimal in even dimensions only)

(can be improved to exp. $O(n \log n + n^{\lceil \frac{d}{2} \rceil})$ if random insertion order can be used)

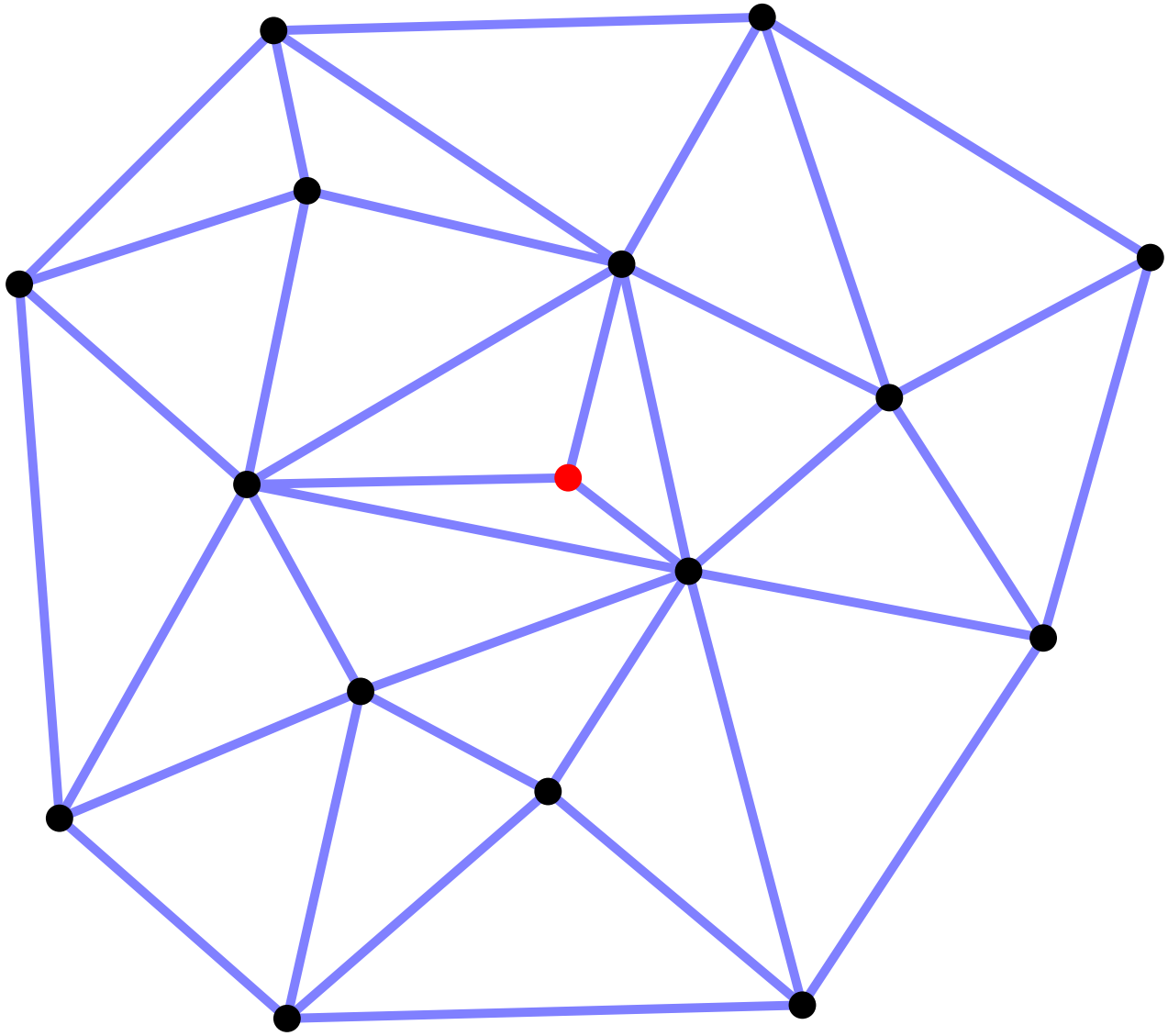
The Guibas/Stolfi variant in 2D

- Locate point in triangulation



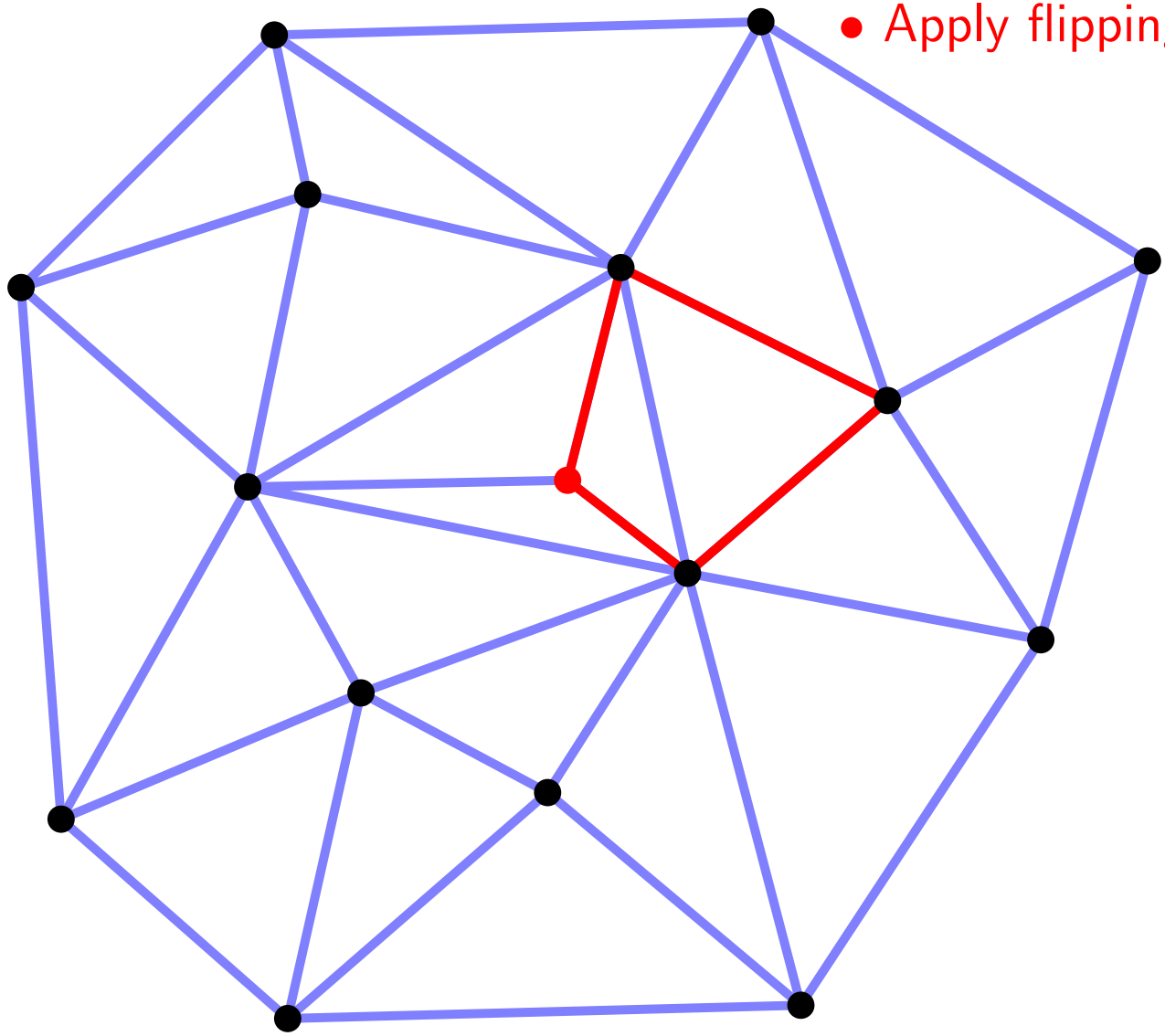
The Guibas/Stolfi variant in 2D

- Locate point in triangulation
- Star triangle



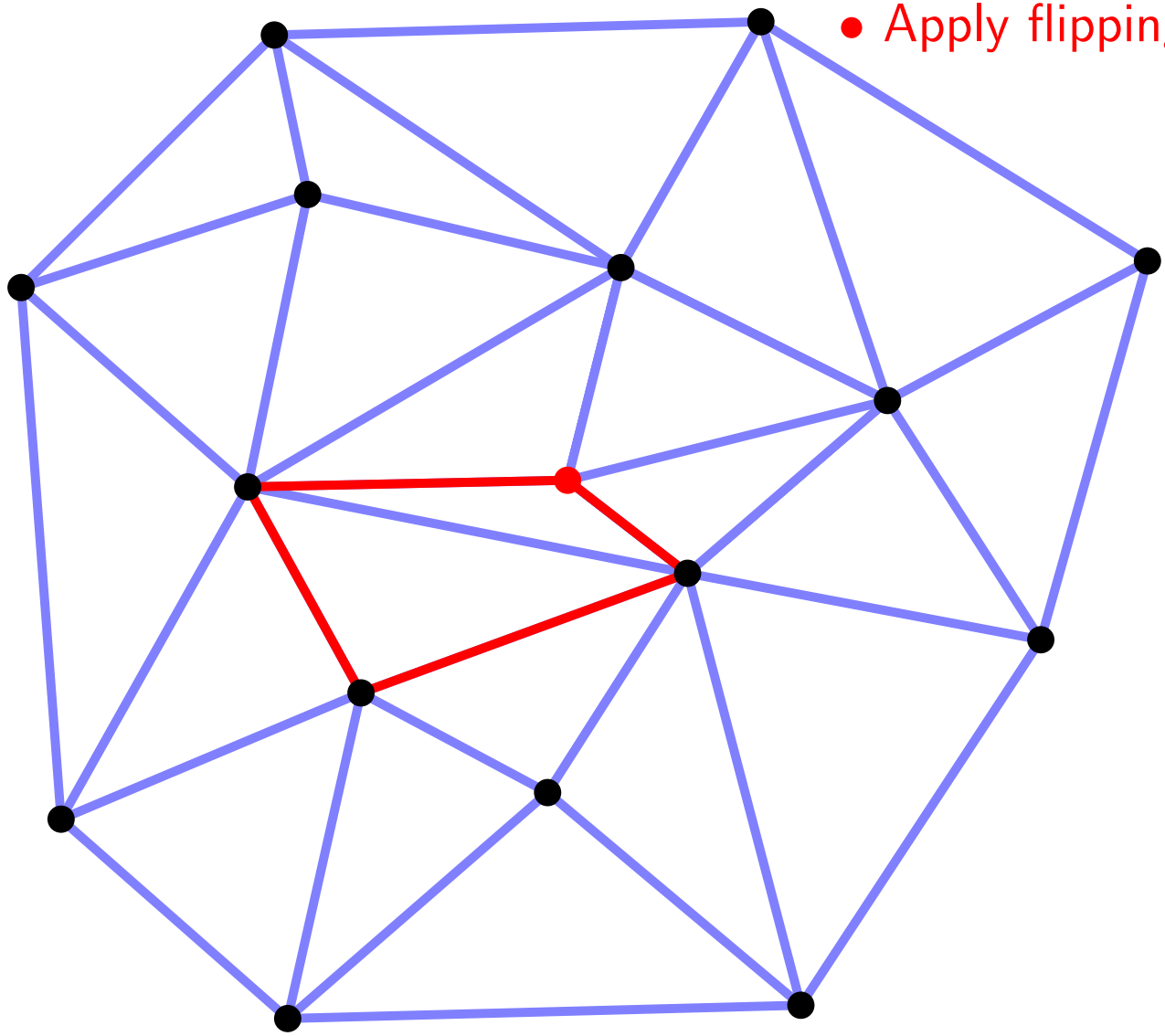
The Guibas/Stolfi variant in 2D

- Locate point in triangulation
- Star triangle
- Apply flipping algorithm



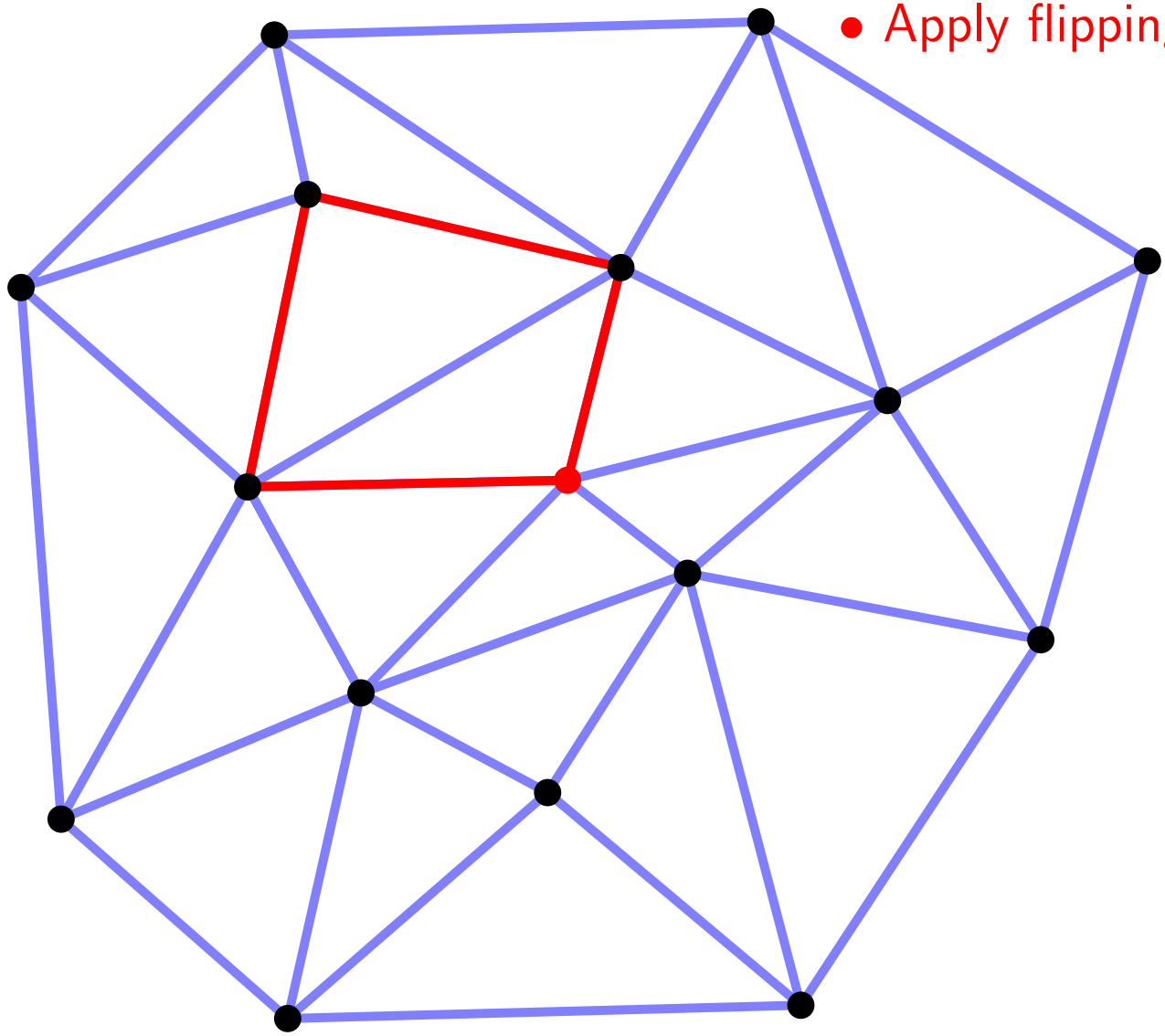
The Guibas/Stolfi variant in 2D

- Locate point in triangulation
- Star triangle
- Apply flipping algorithm



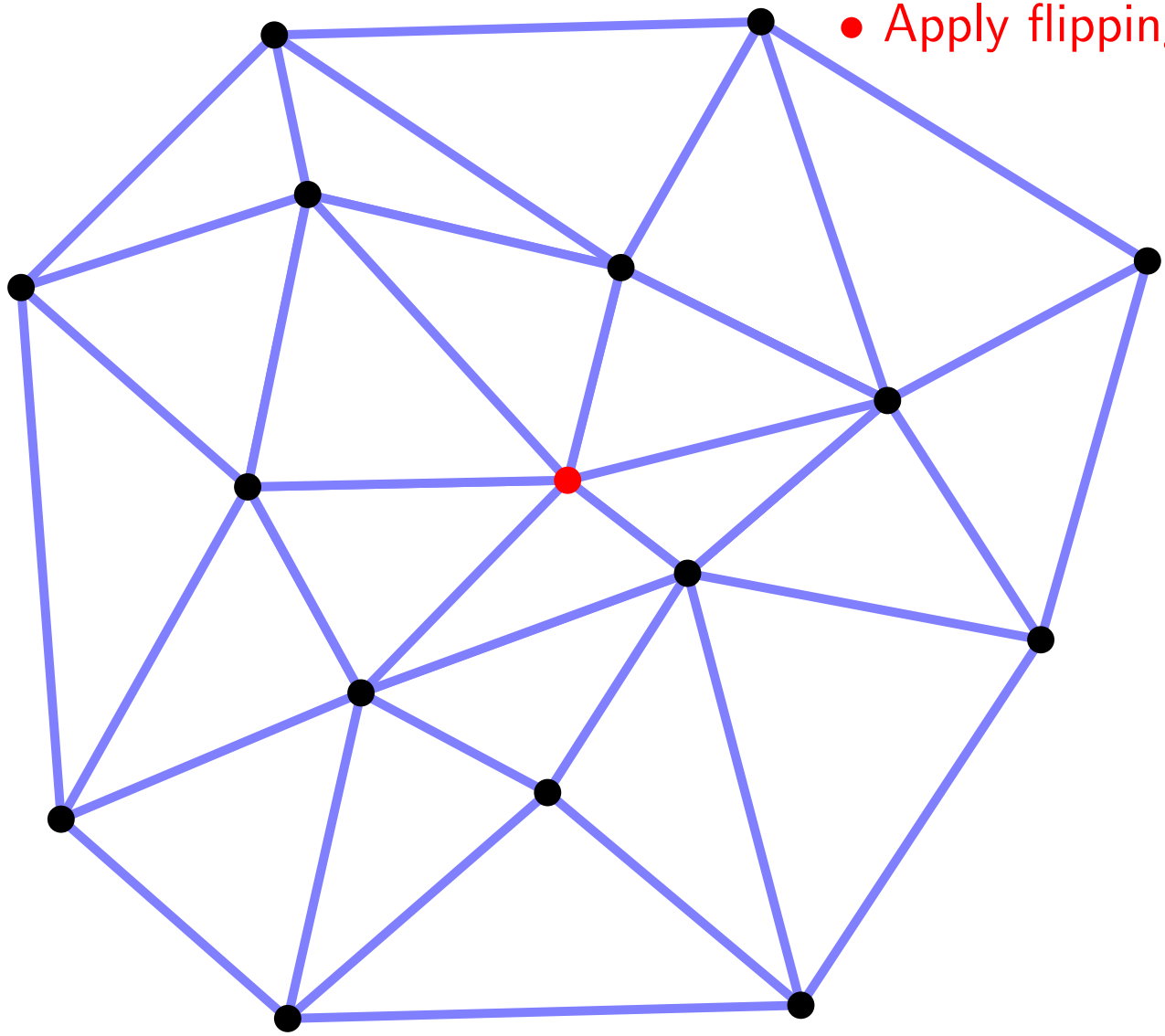
The Guibas/Stolfi variant in 2D

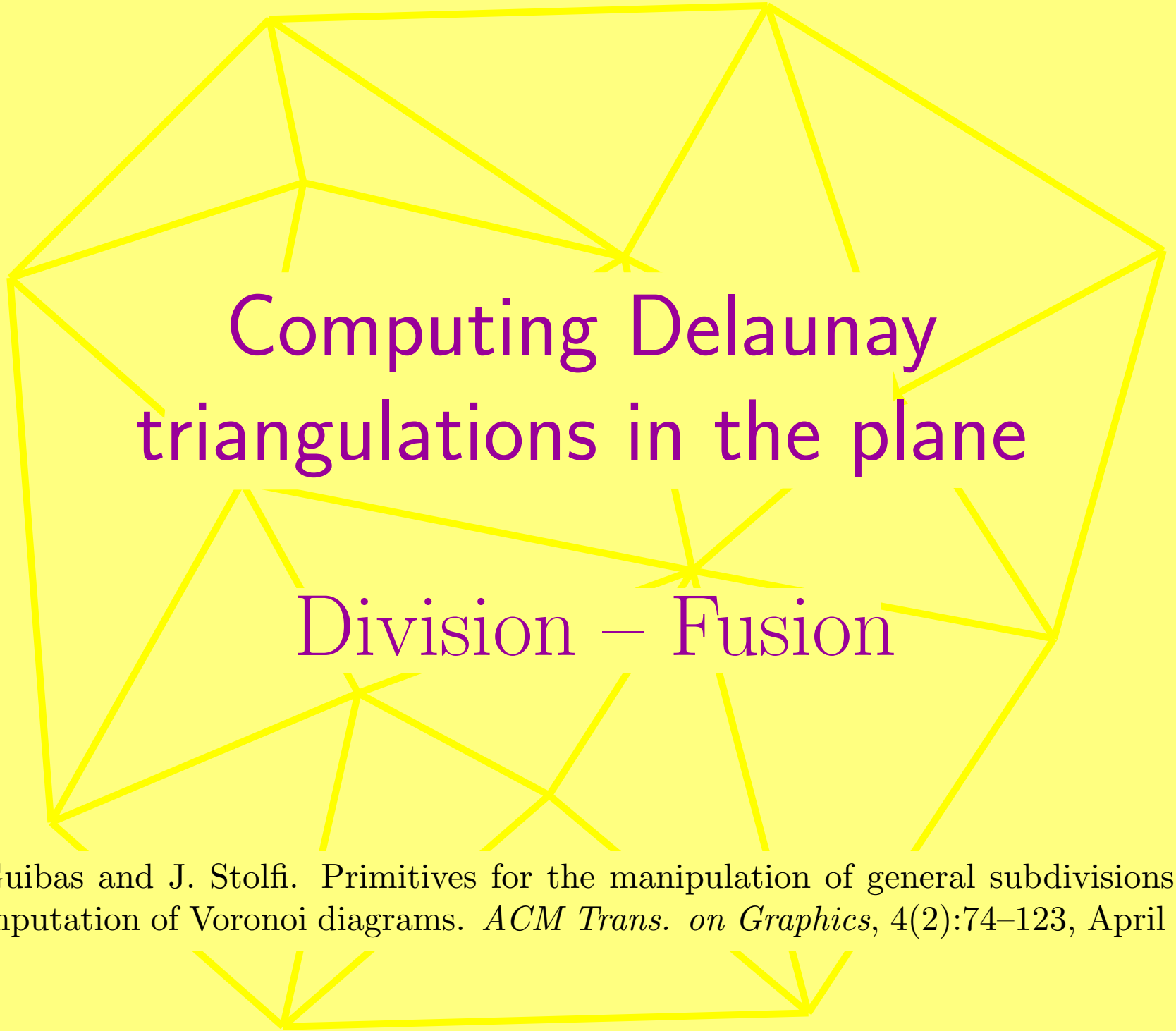
- Locate point in triangulation
- Star triangle
- Apply flipping algorithm



The Guibas/Stolfi variant in 2D

- Locate point in triangulation
- Star triangle
- Apply flipping algorithm





Computing Delaunay triangulations in the plane

Division – Fusion

L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. on Graphics*, 4(2):74–123, April 1985

Division-Fusion

Classical approach example: sort

Problem of size n

$\rightarrow$ division into 2 pbs of size $O\left(n/2\right)$

$\rightarrow$ recursive call on sub-problems

$\rightarrow$ fusion

Division-Fusion

Classical approach example: sort

Problem of size n

$\rightarrow$ division into 2 pbs of size $O\left(n/2\right)$

$O(n)$

$\rightarrow$ recursive call on sub-problems

$2\ f\left(\frac{n}{2}\right)$

$\rightarrow$ fusion

$O(n)$

Division-Fusion

Classical approach example: sort

Problem of size n

$$f(n) = O(n) + 2f\left(\frac{n}{2}\right) = O(n \log n)$$

$\rightarrow$ division into 2 pbs of size $O\left(n/2\right)$

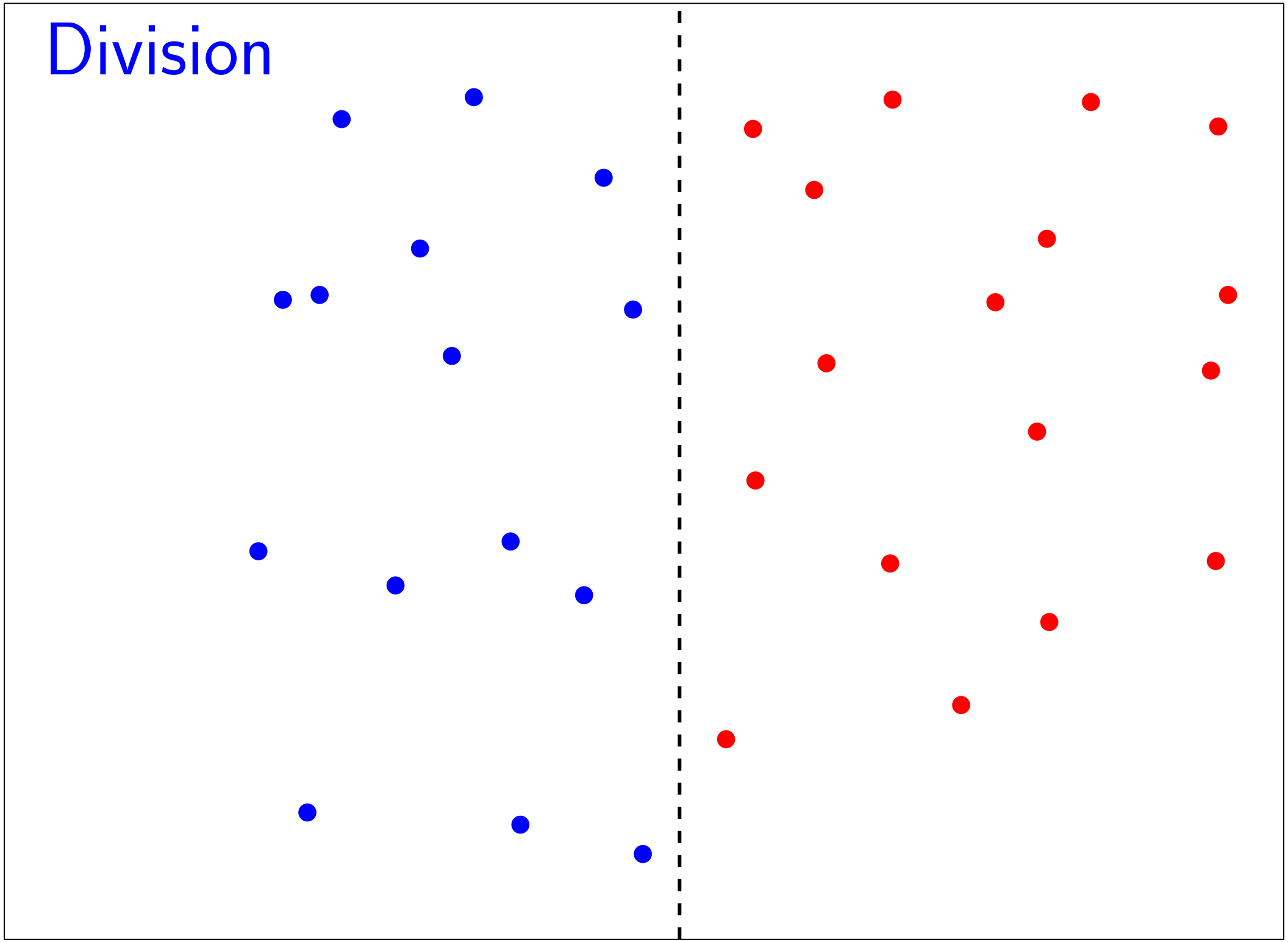
$O(n)$

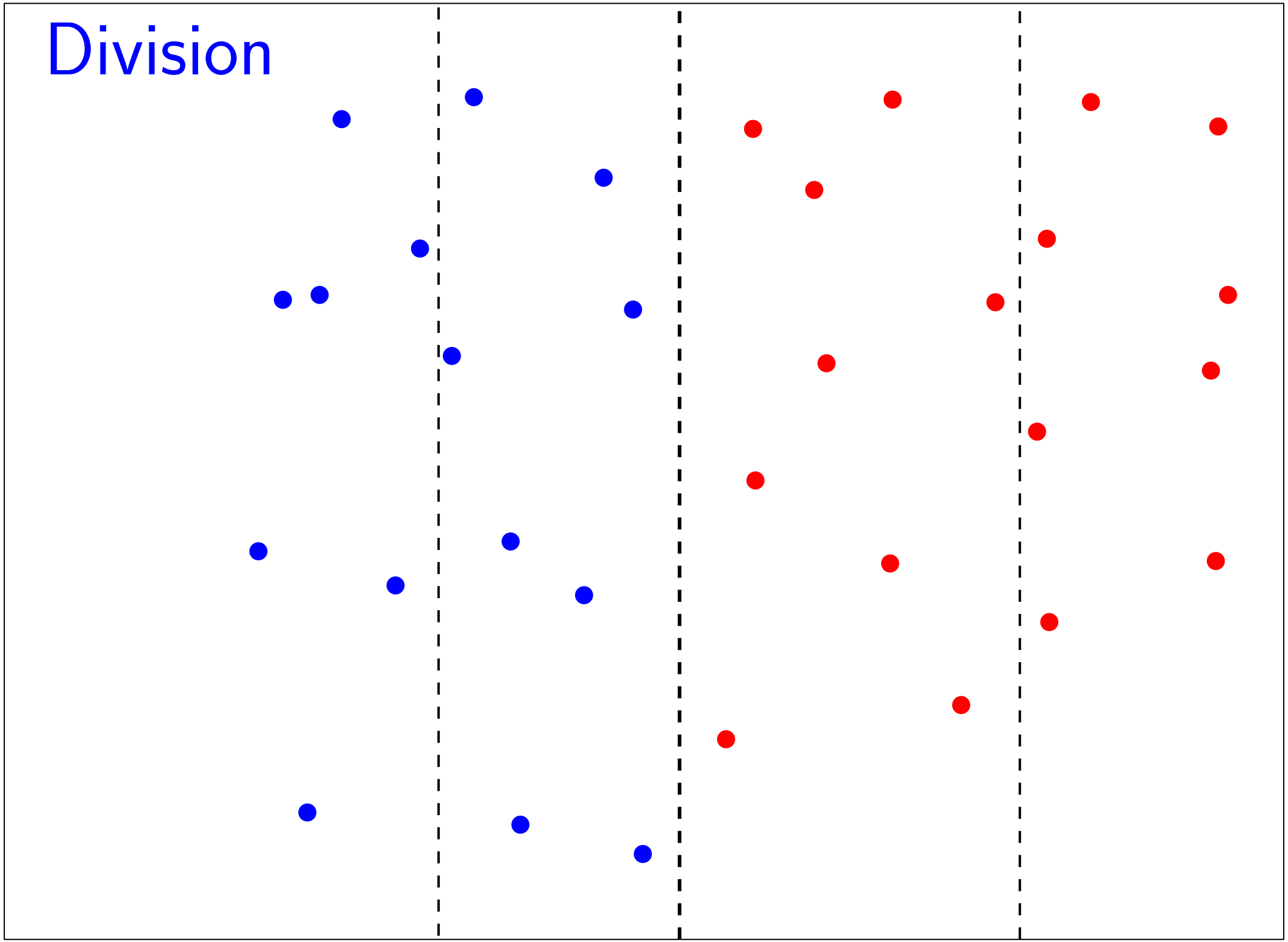
$\rightarrow$ recursive call on sub-problems

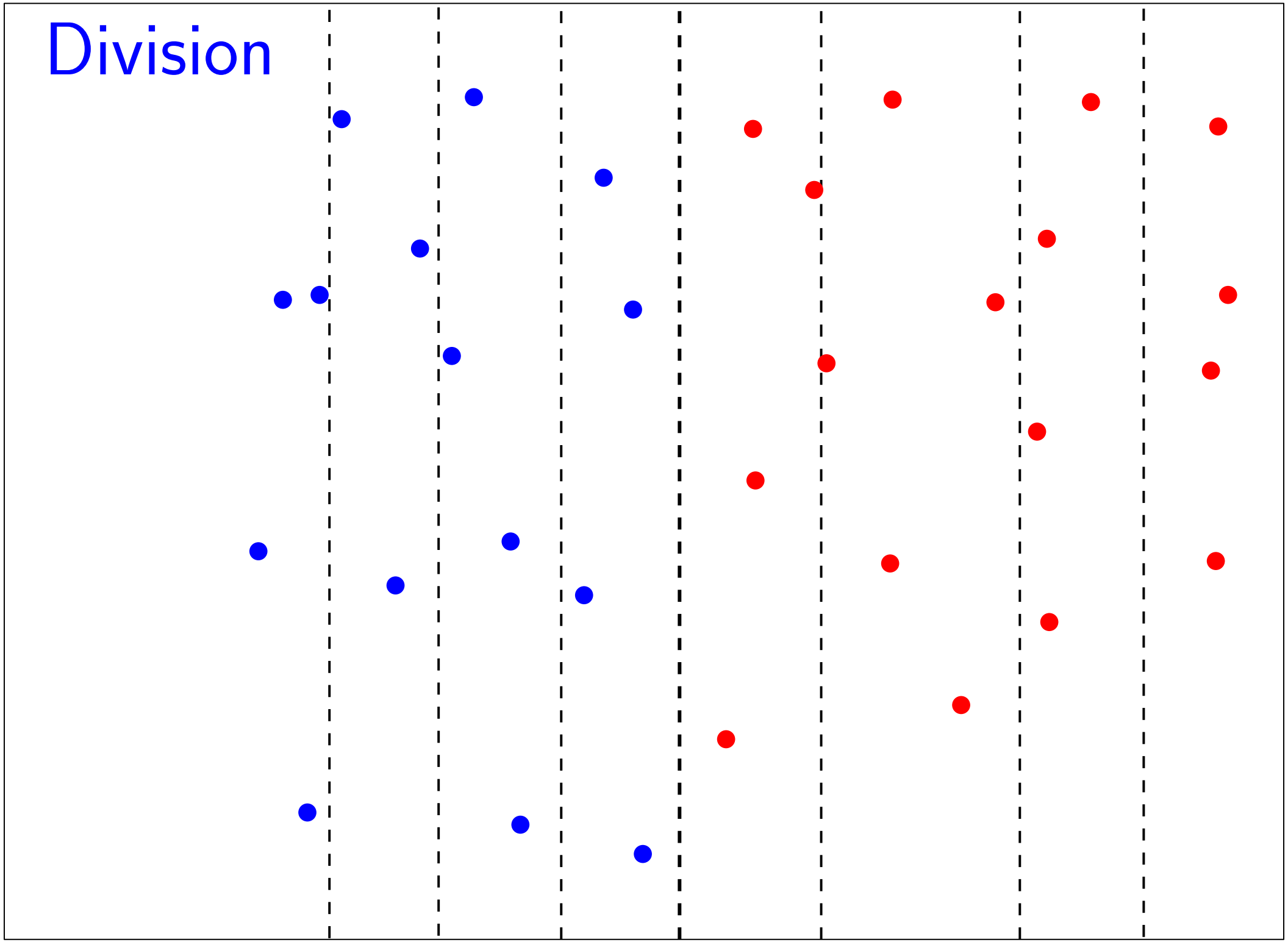
$2\ f\left(\frac{n}{2}\right)$

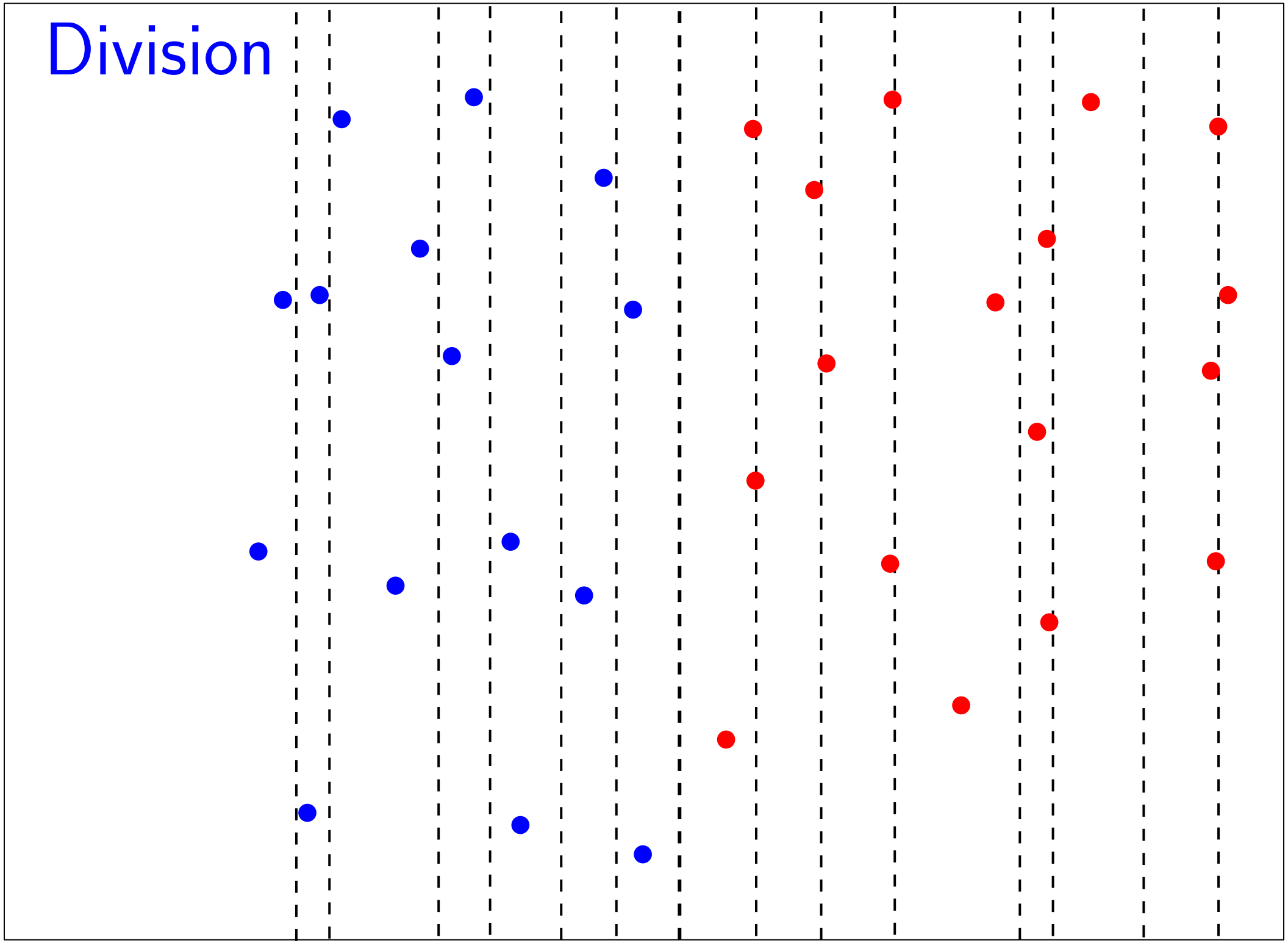
$\rightarrow$ fusion

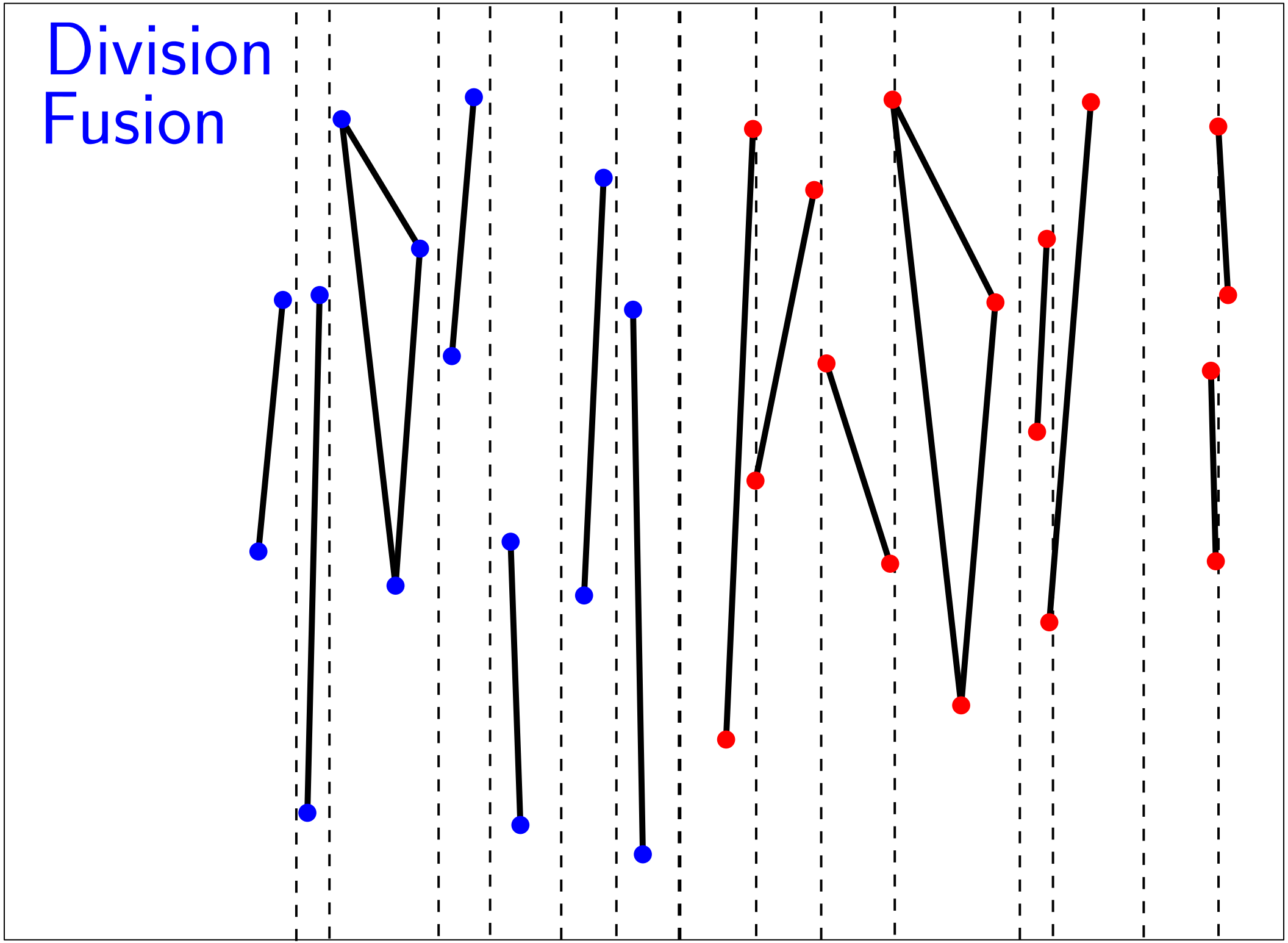
$O(n)$

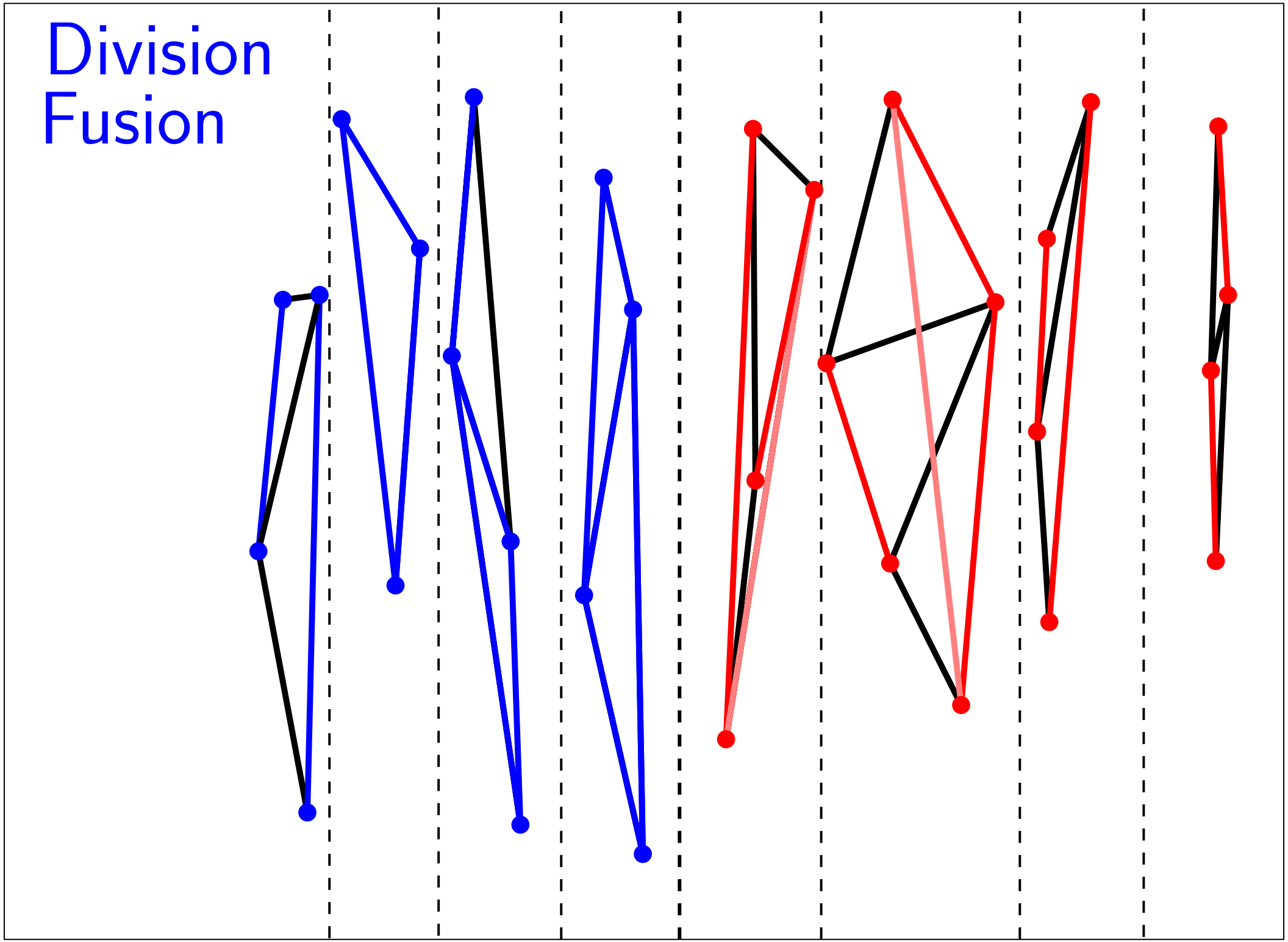

Division

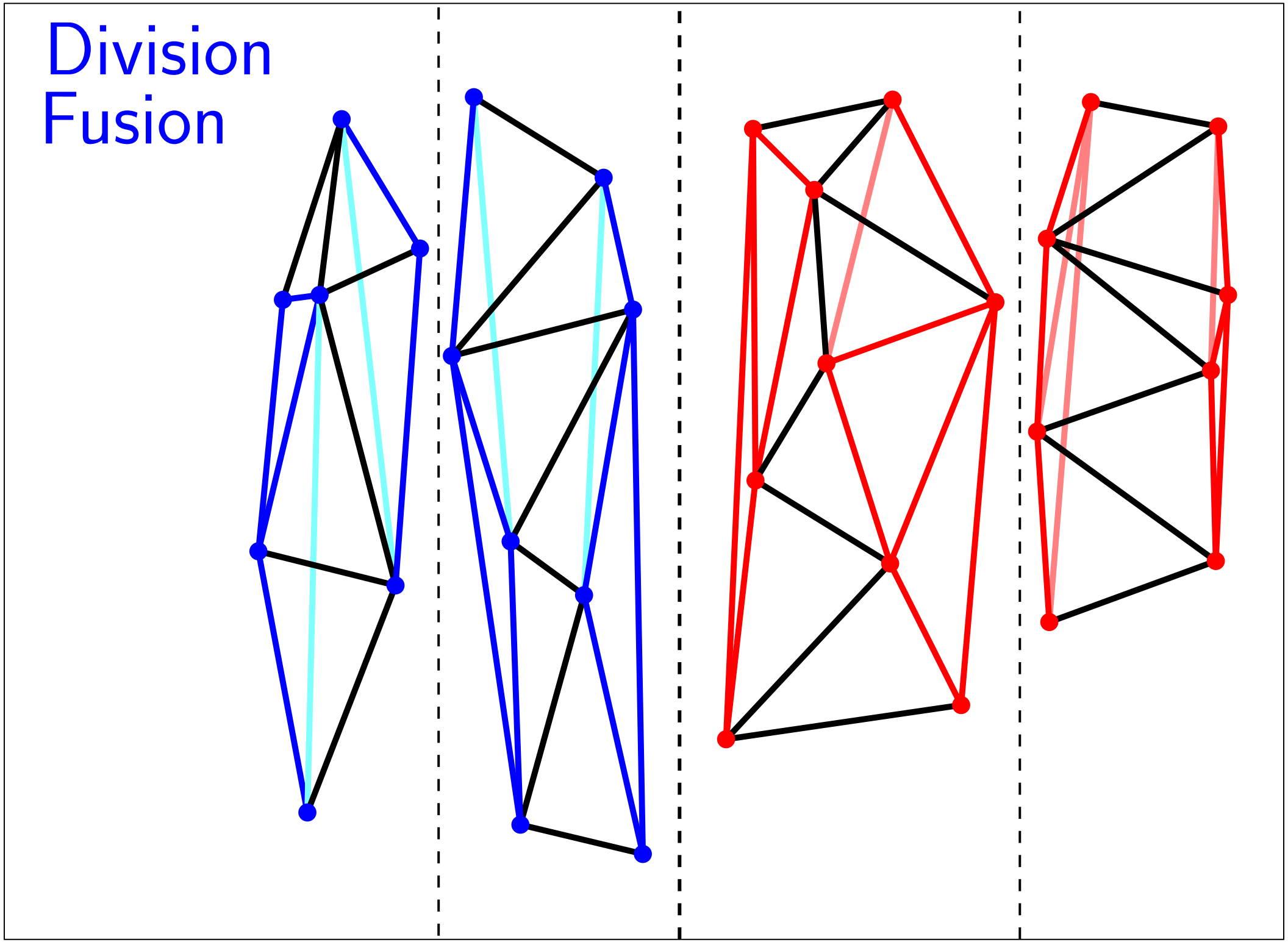

Division


Division

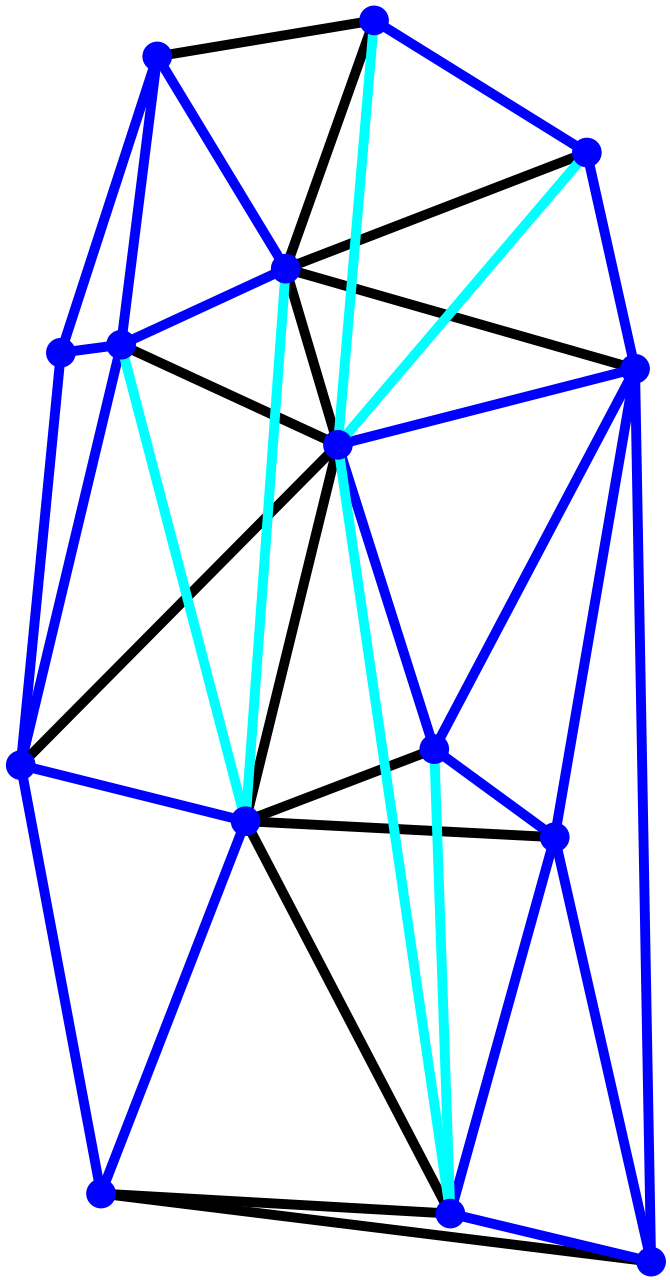

Division

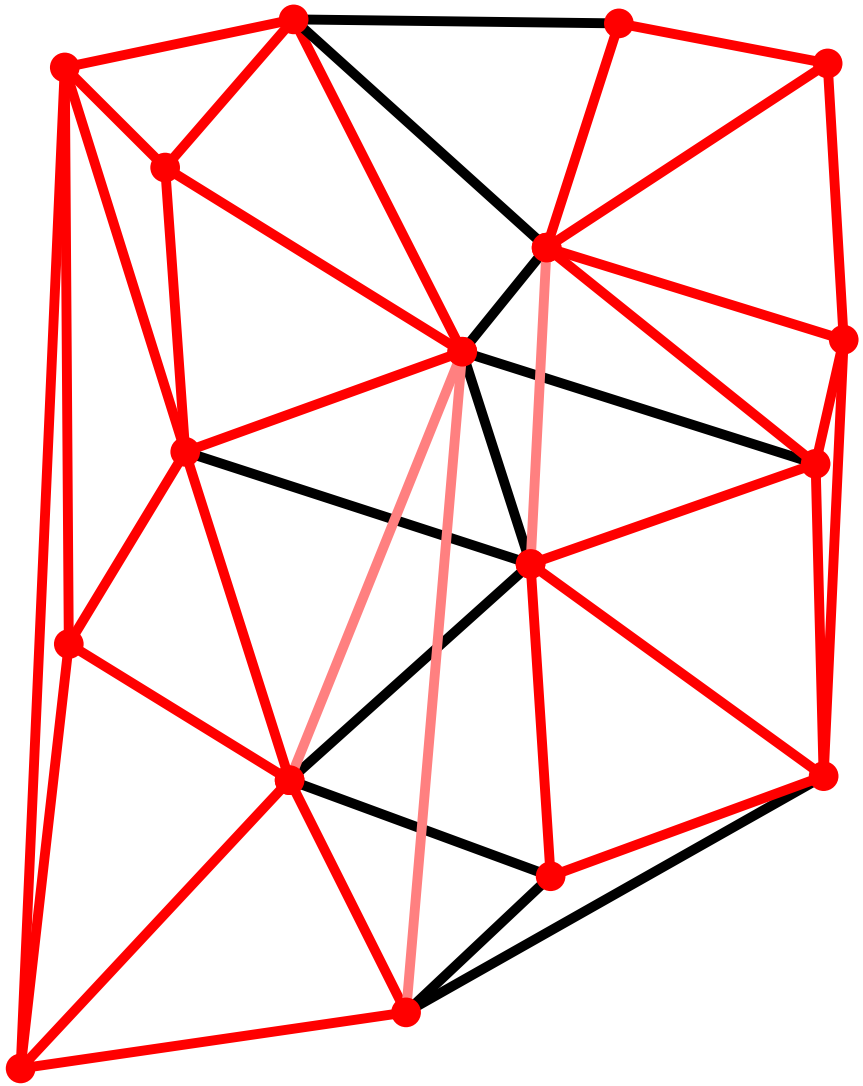

Division
Fusion

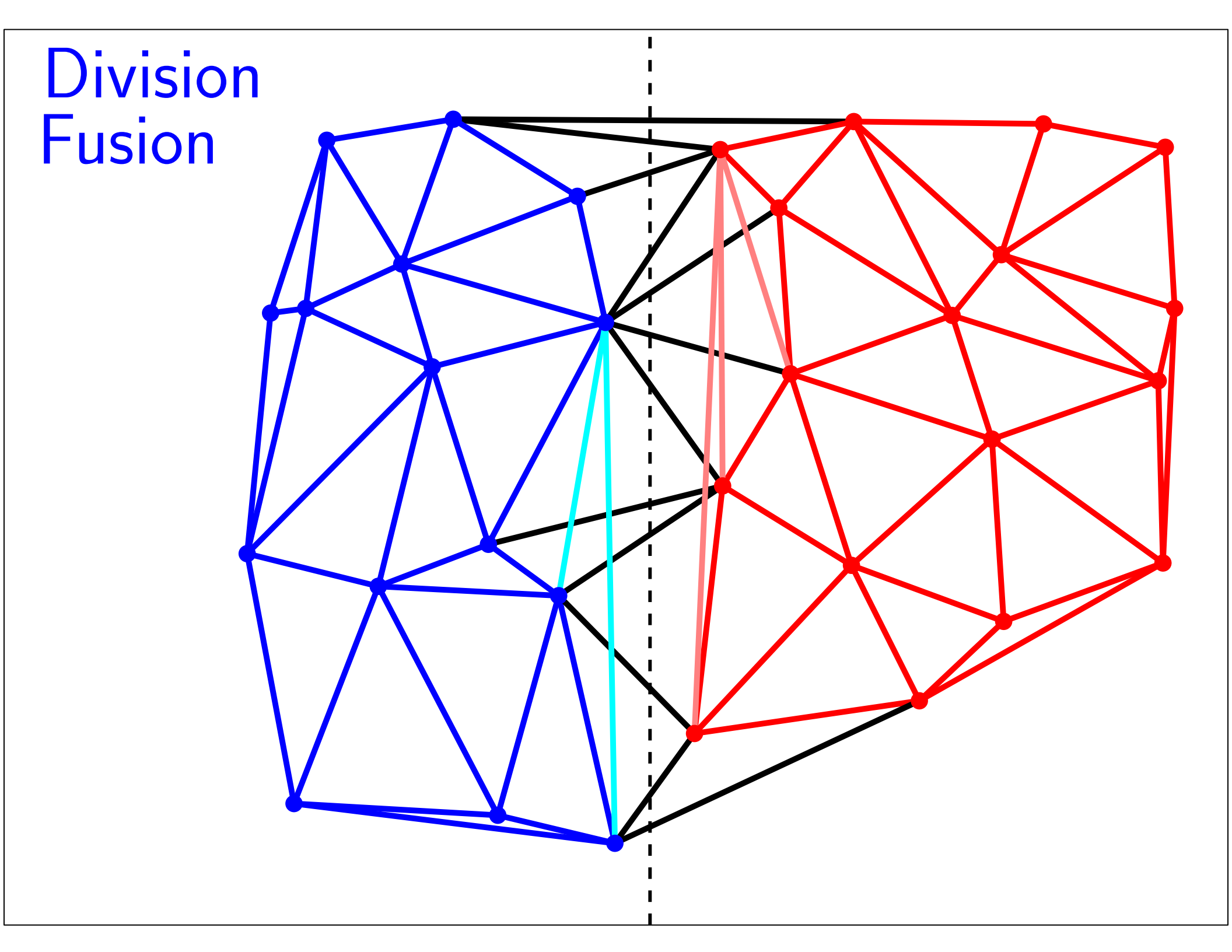

Division
Fusion

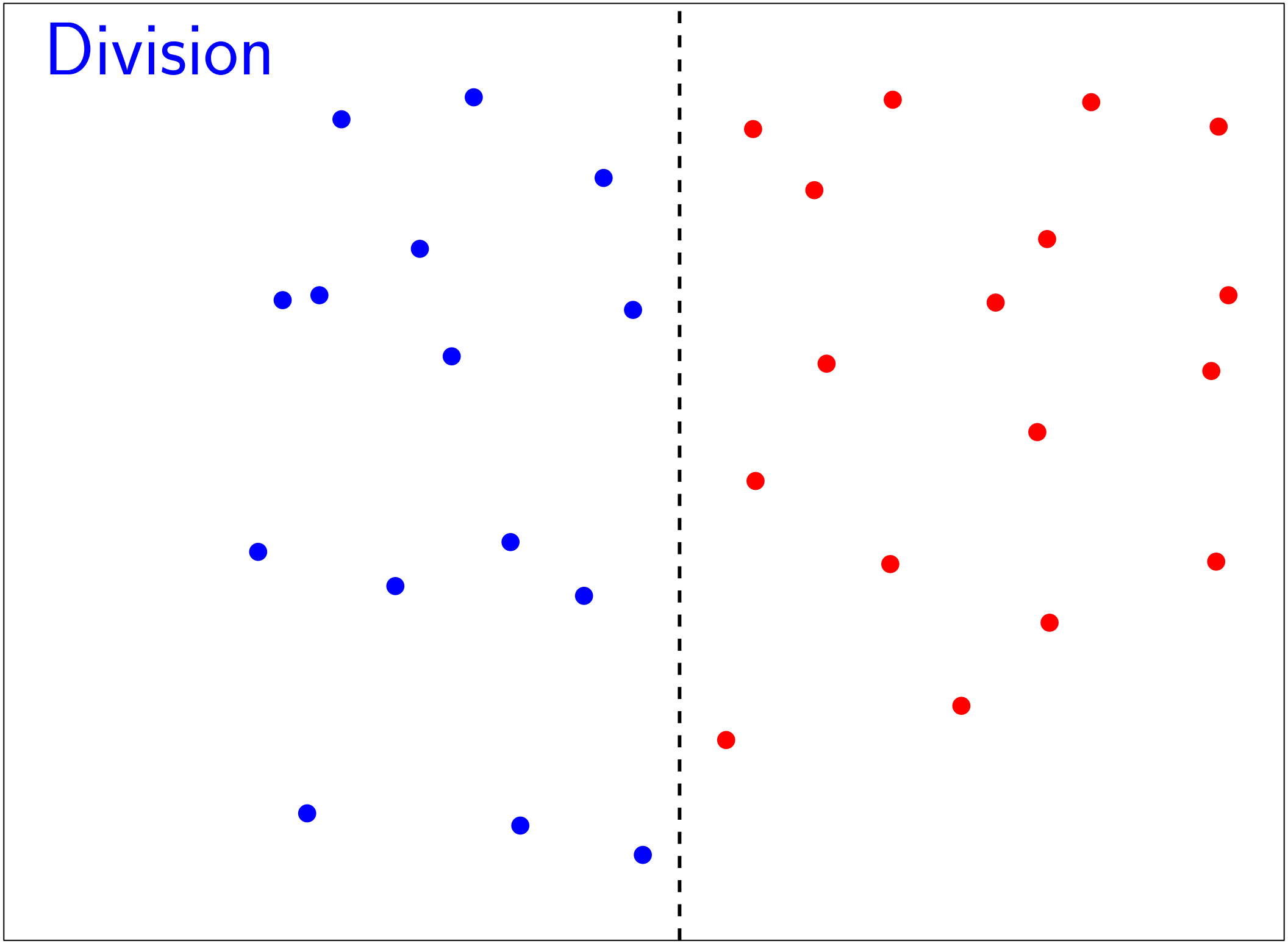

Division
Fusion

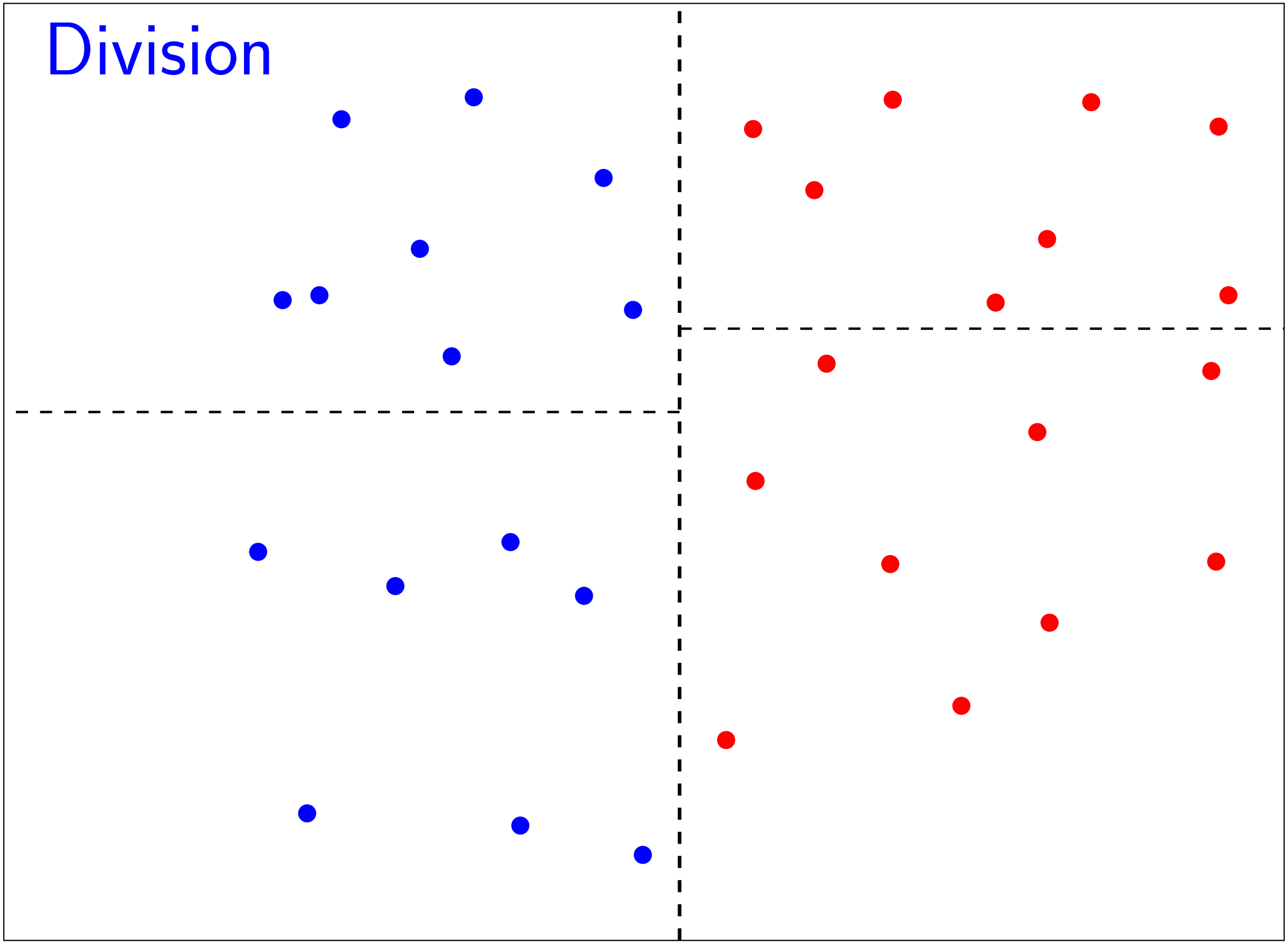
Division
Fusion

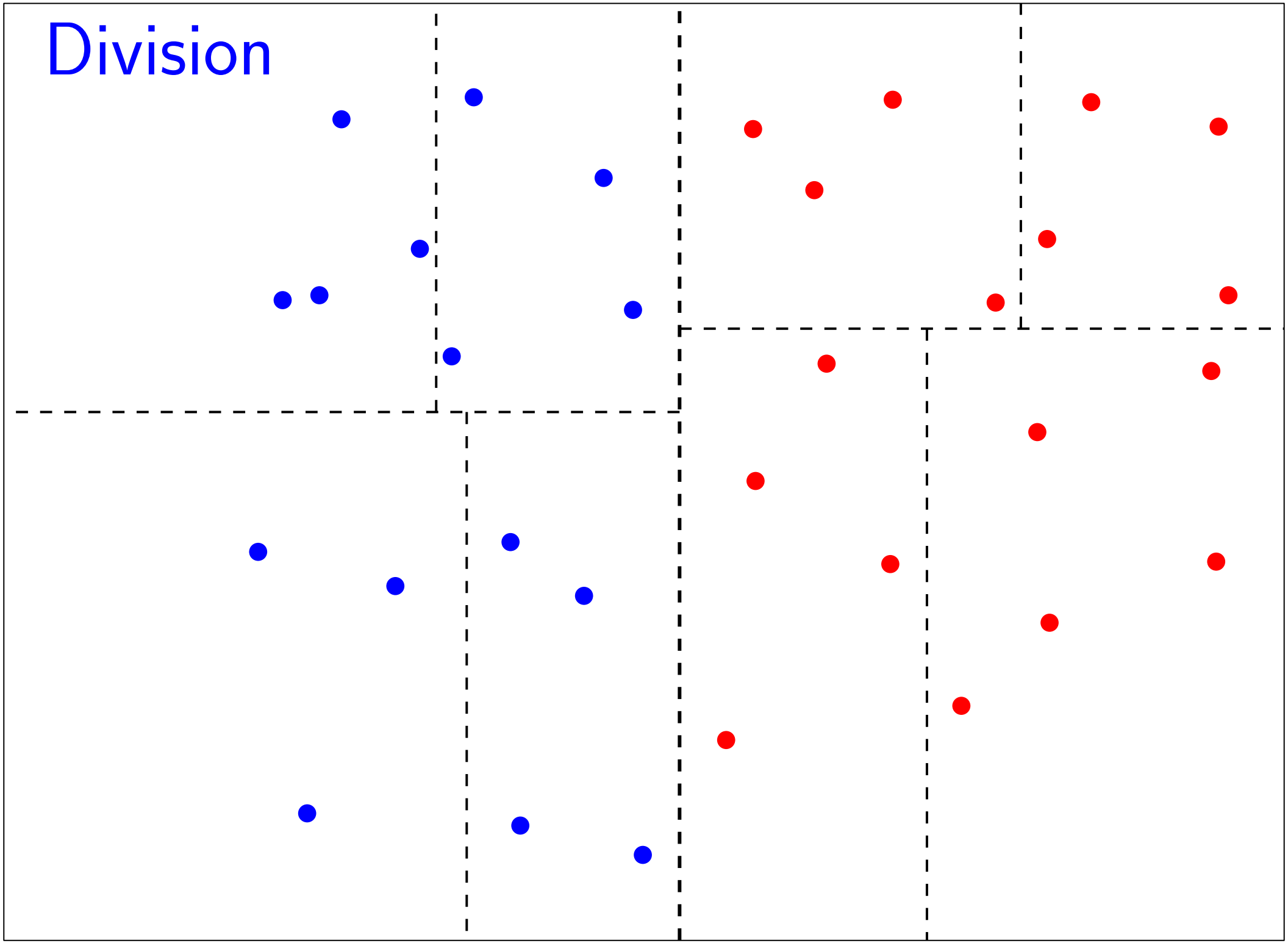


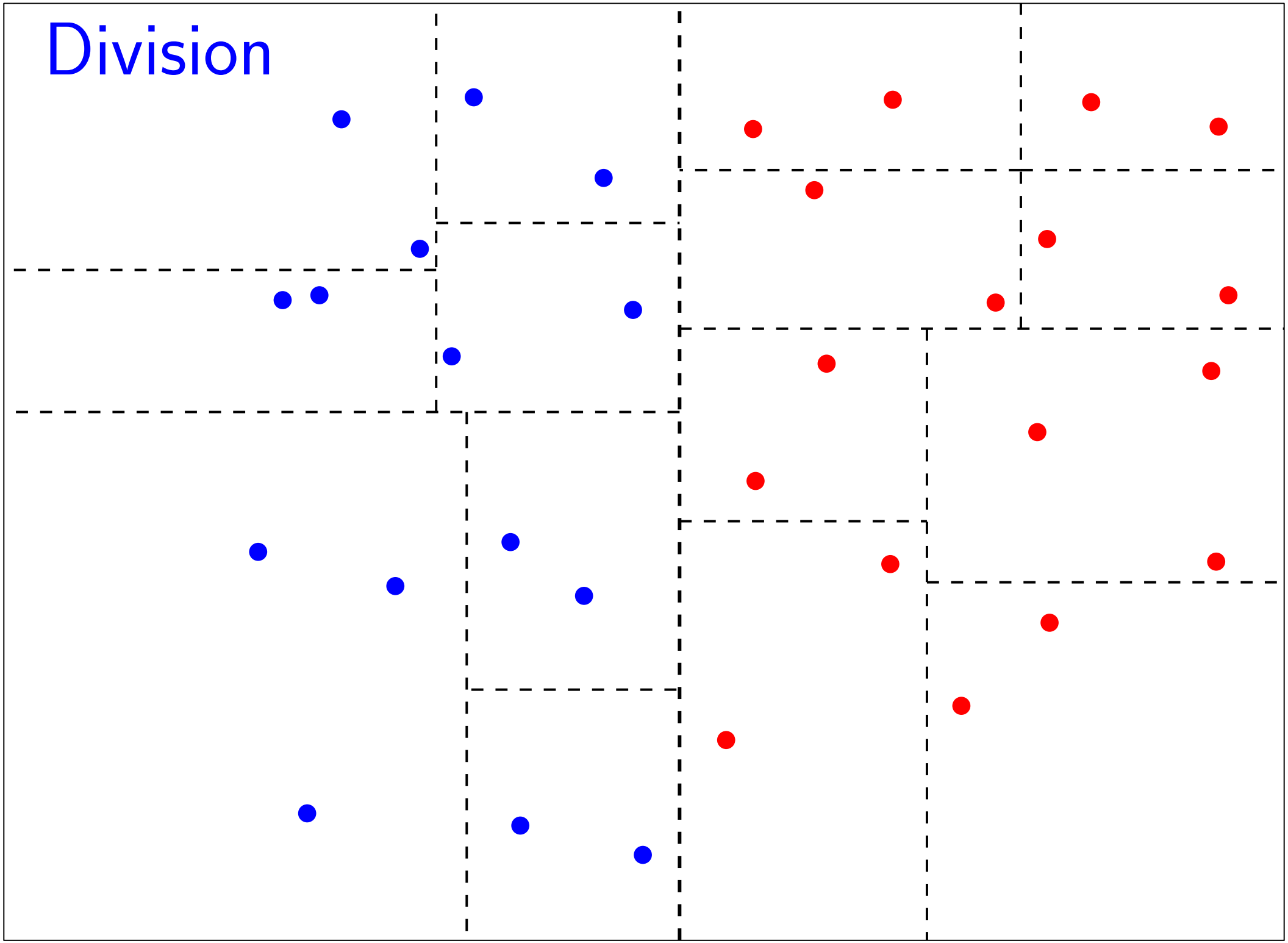


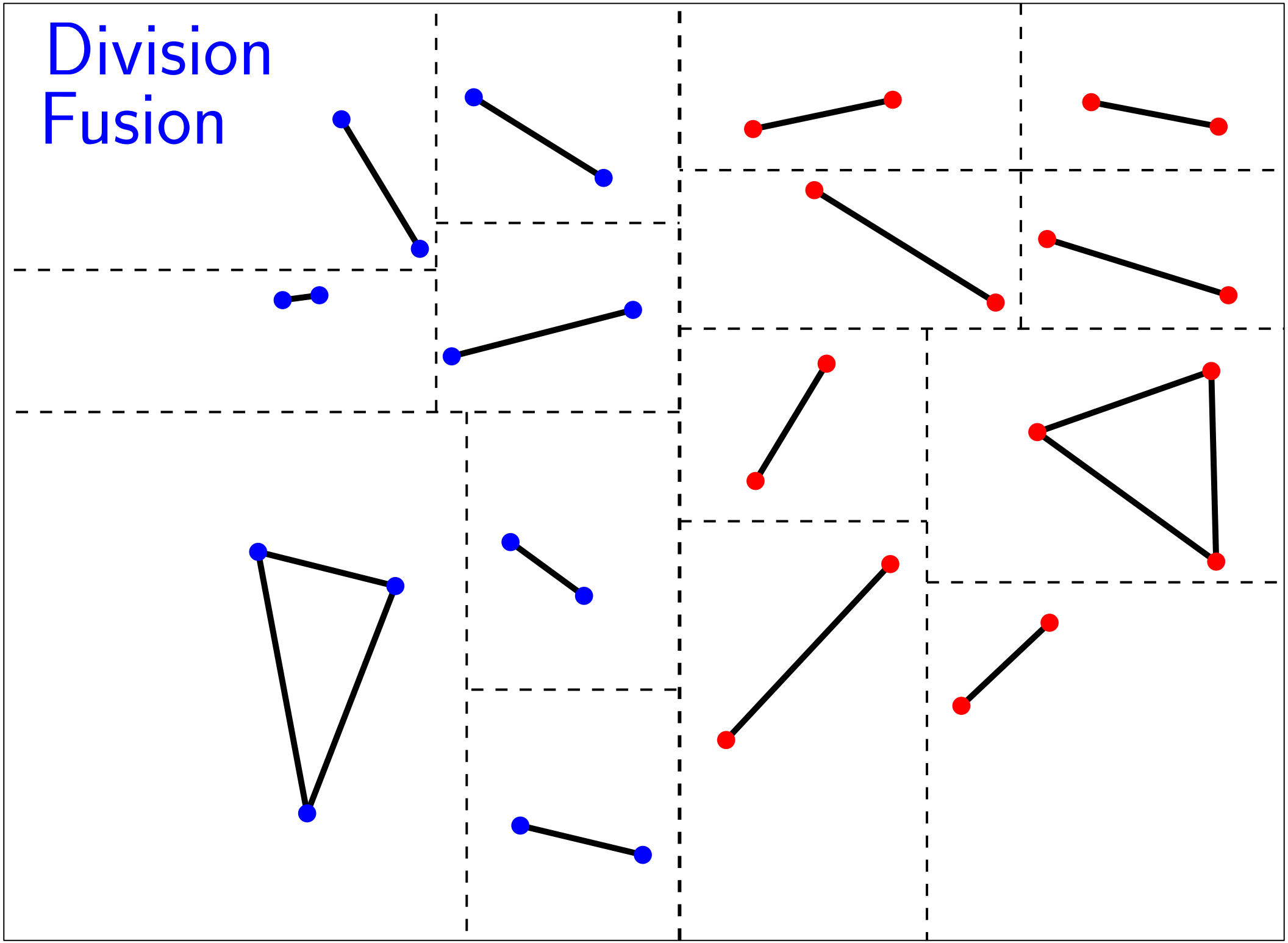

Division
Fusion

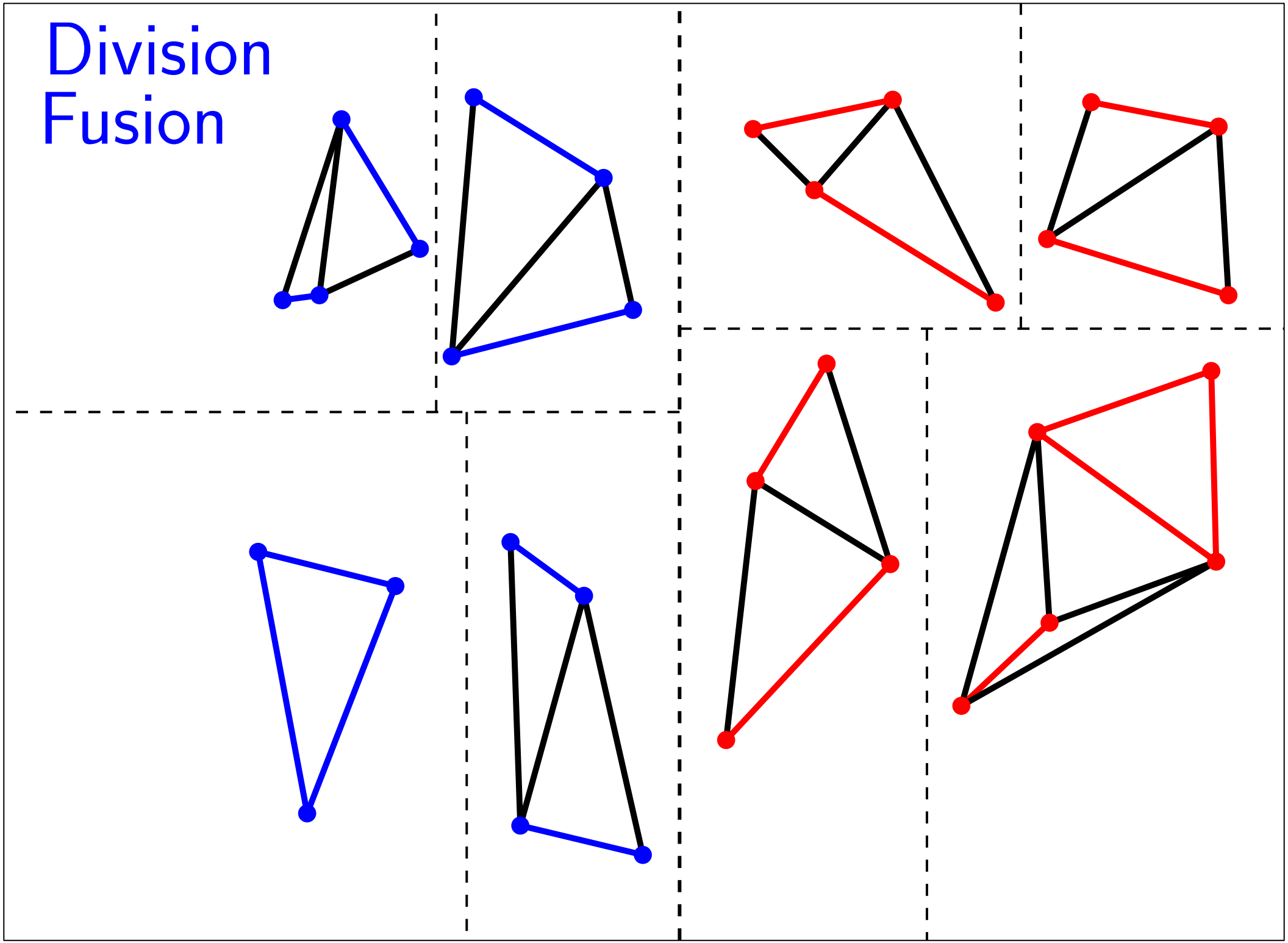

Division


Division

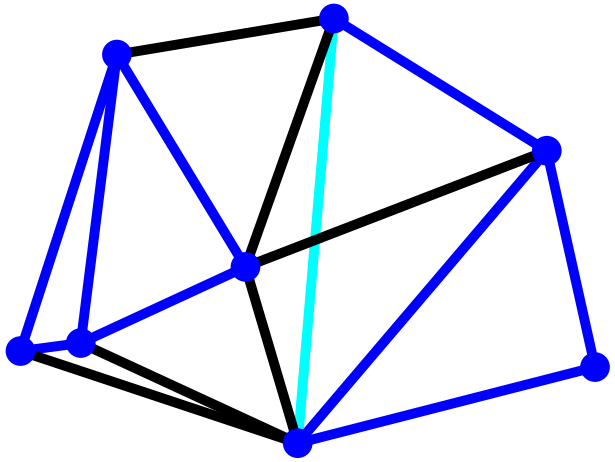

Division

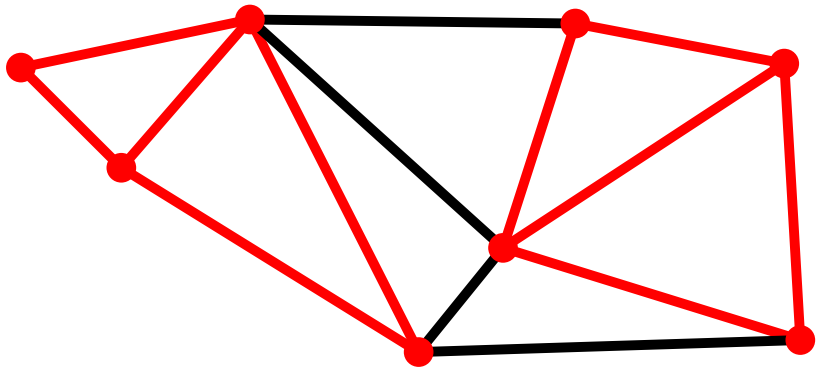

Division

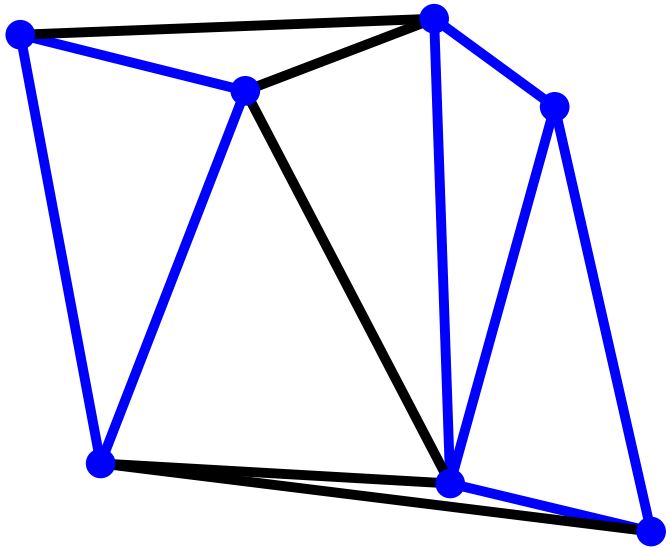

Division
Fusion

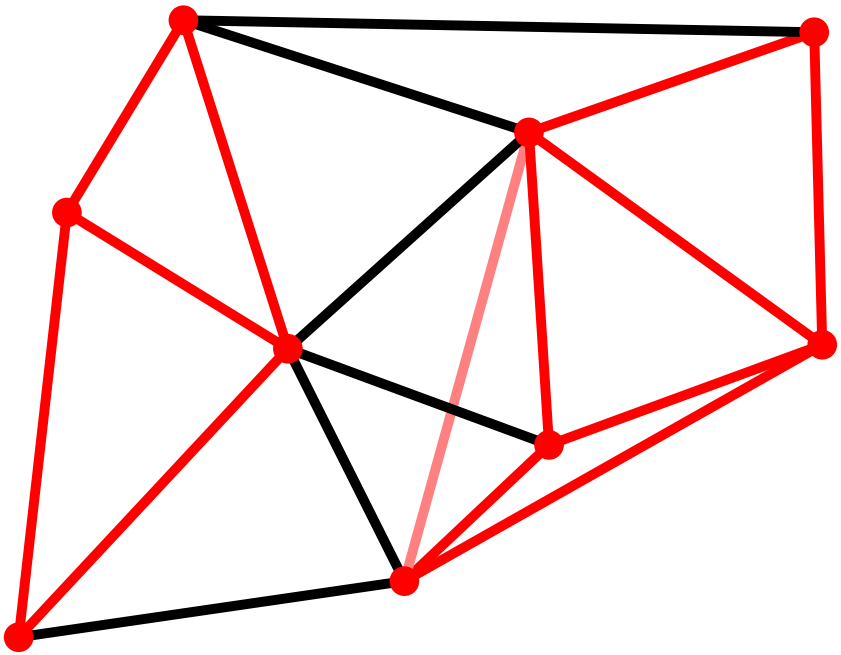

Division
Fusion

Division
Fusion

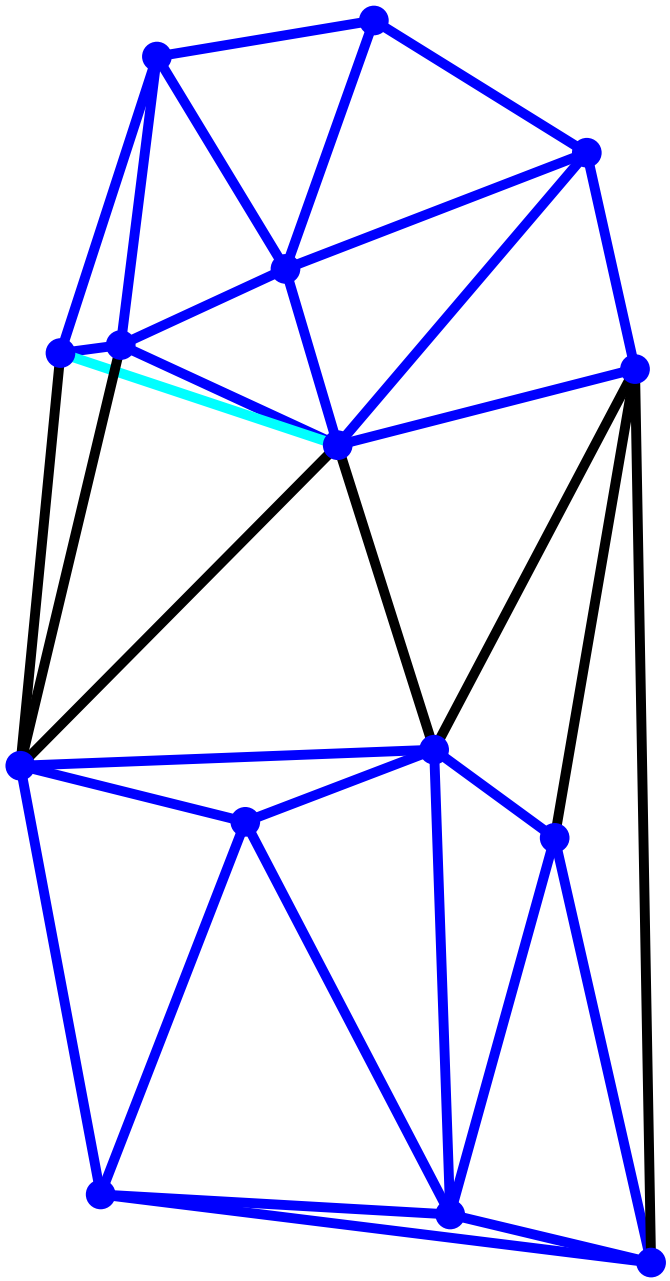


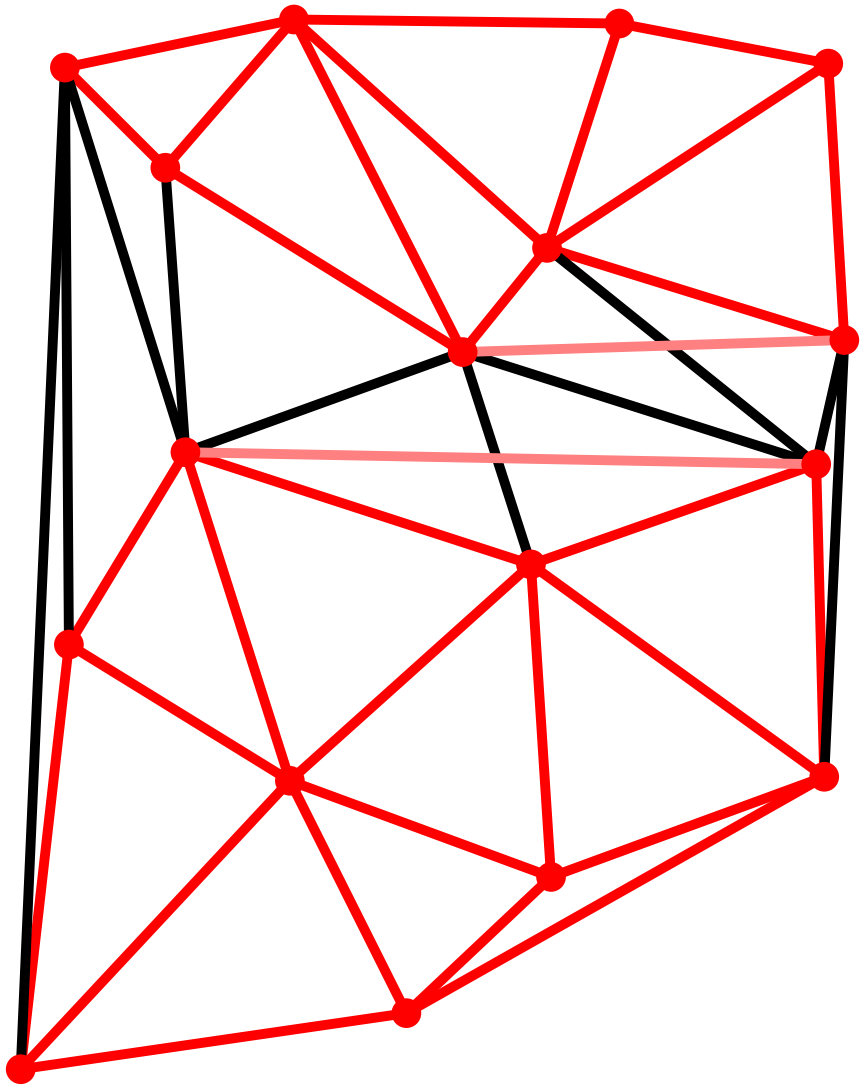


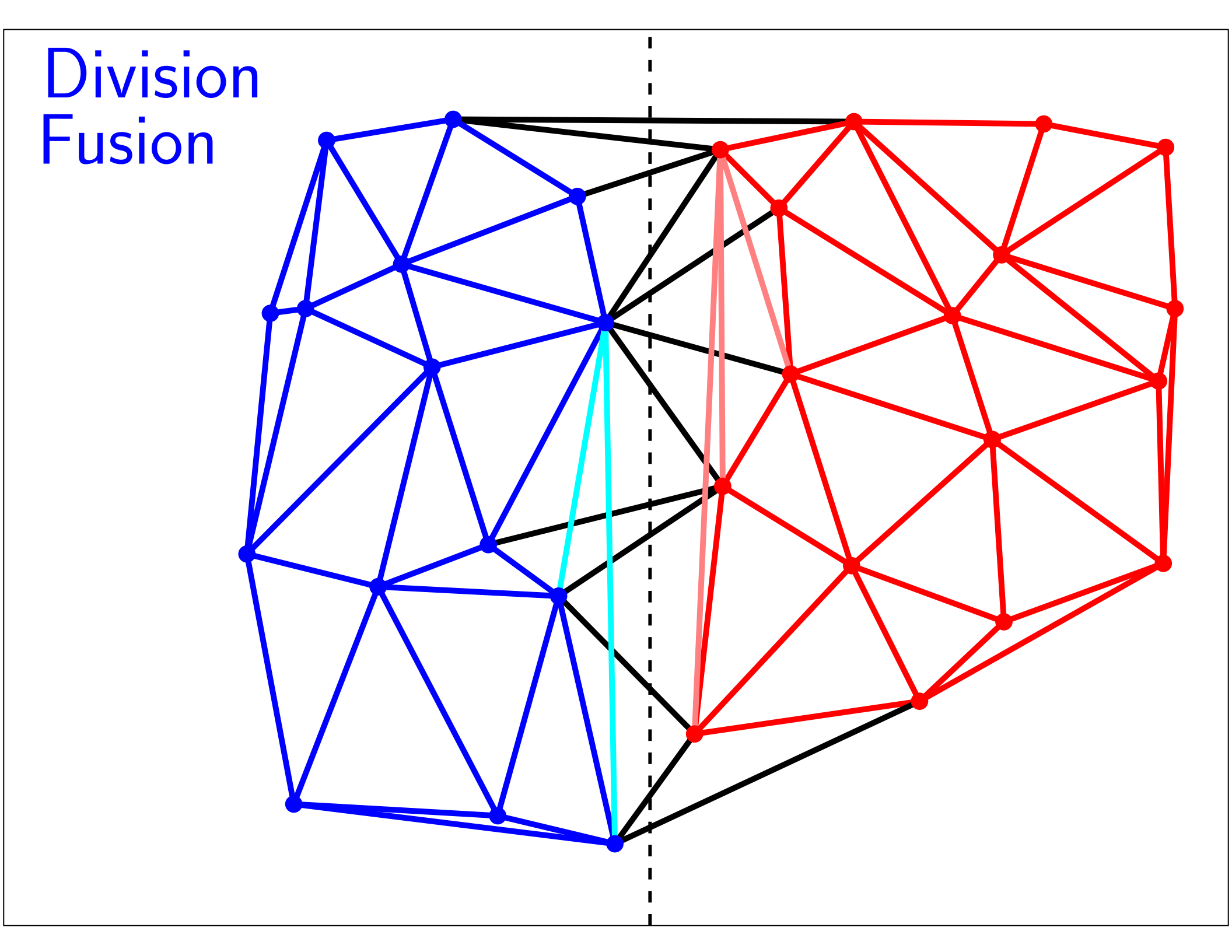


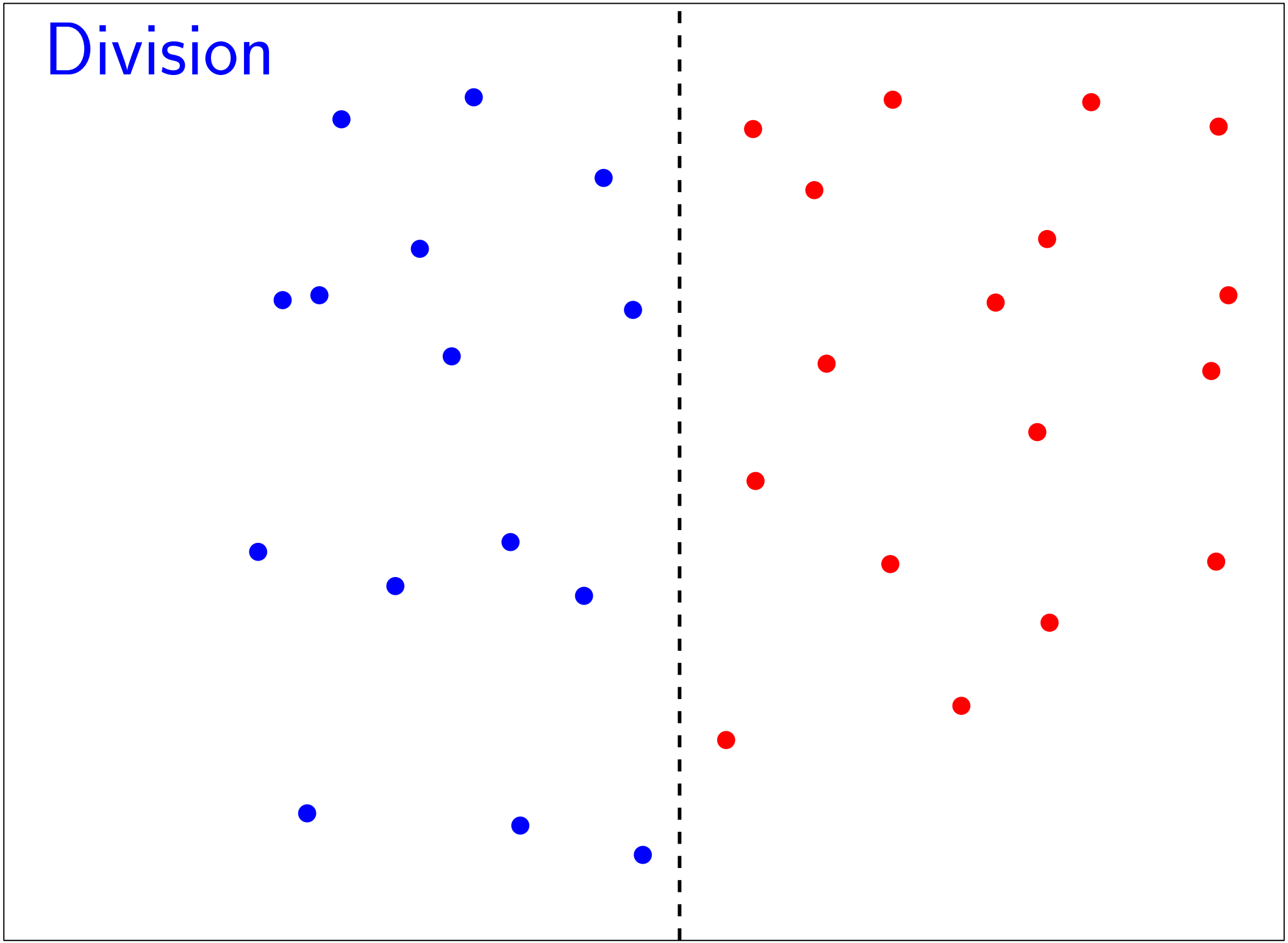


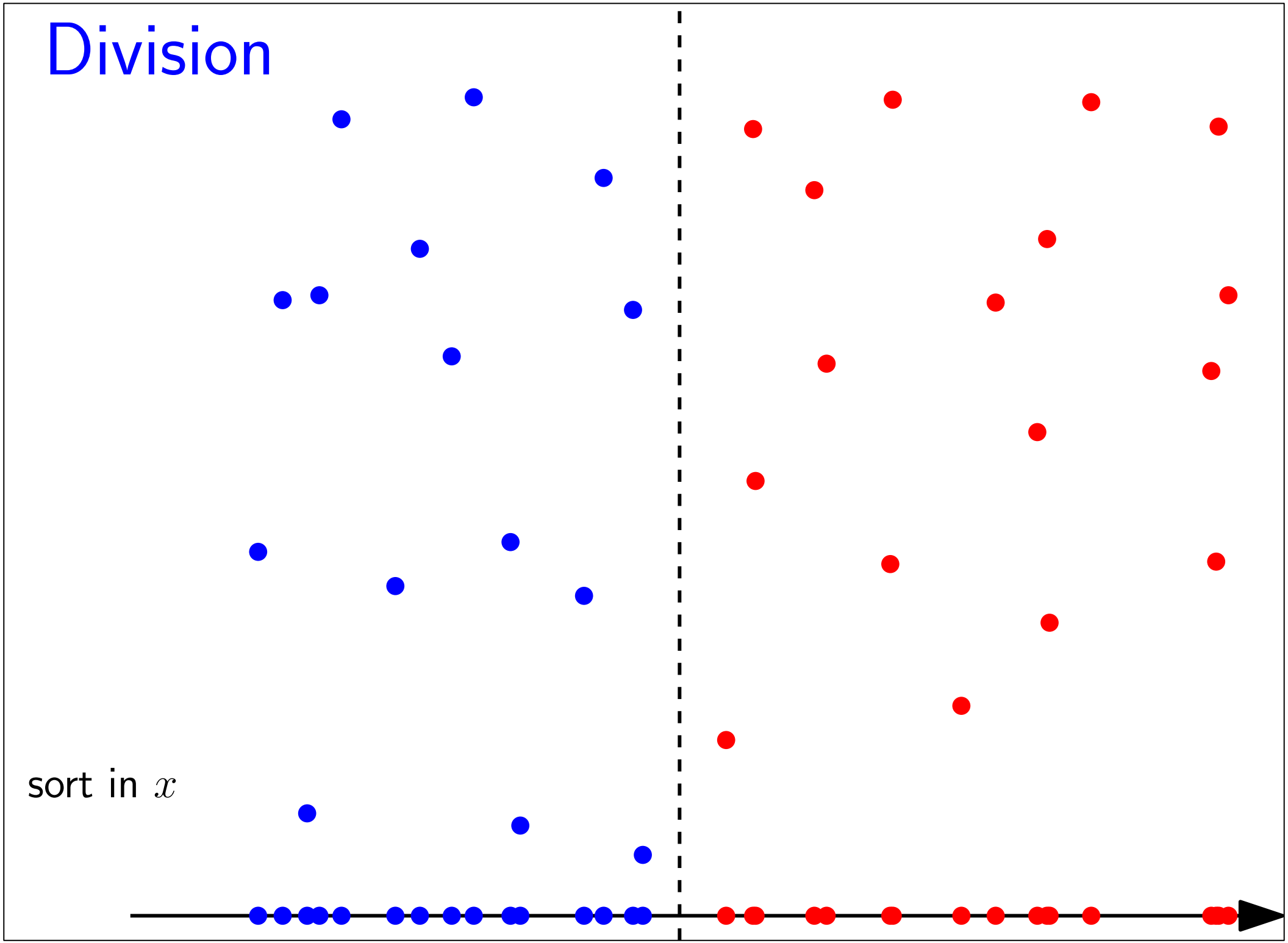
Division
Fusion

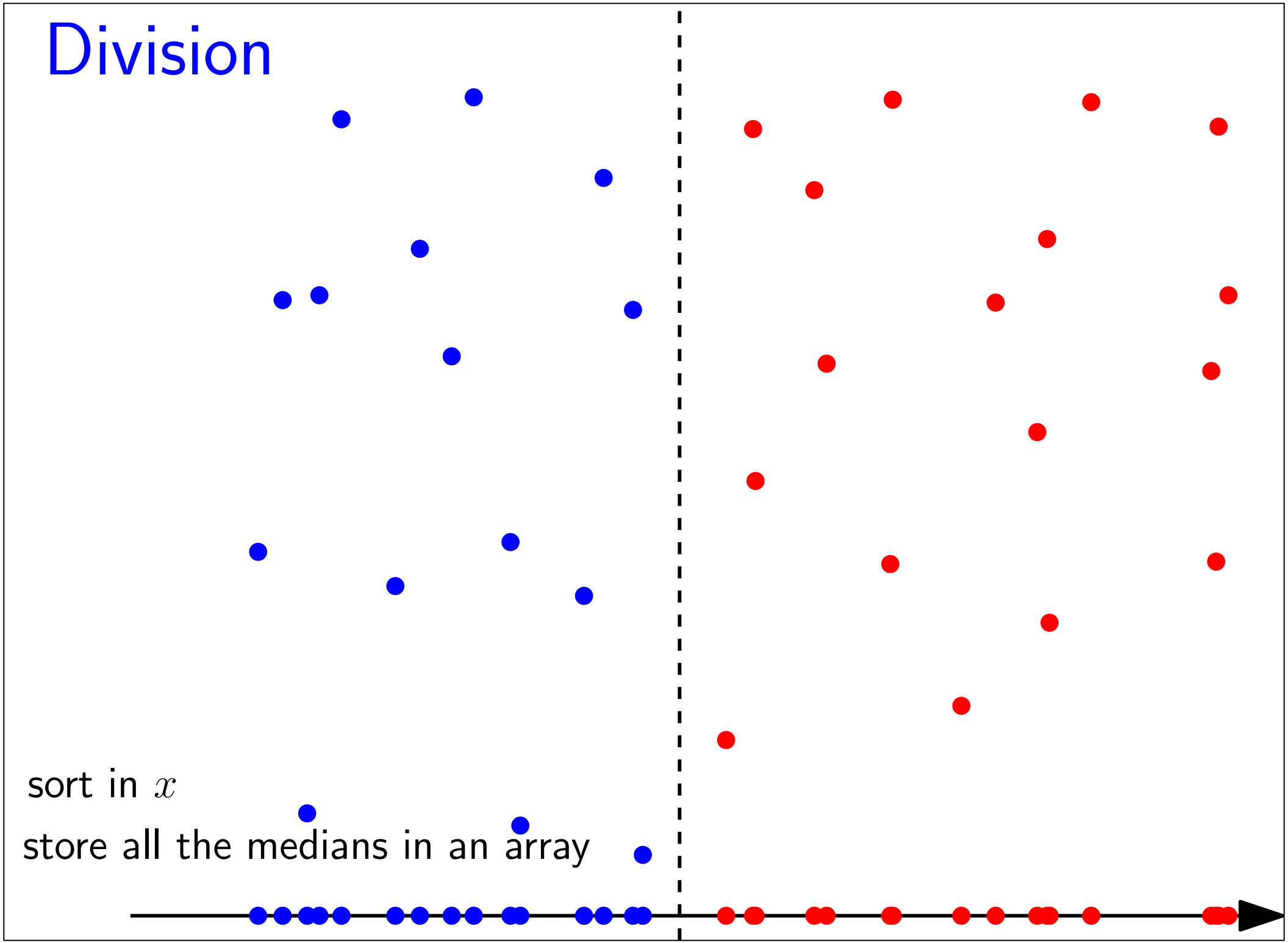


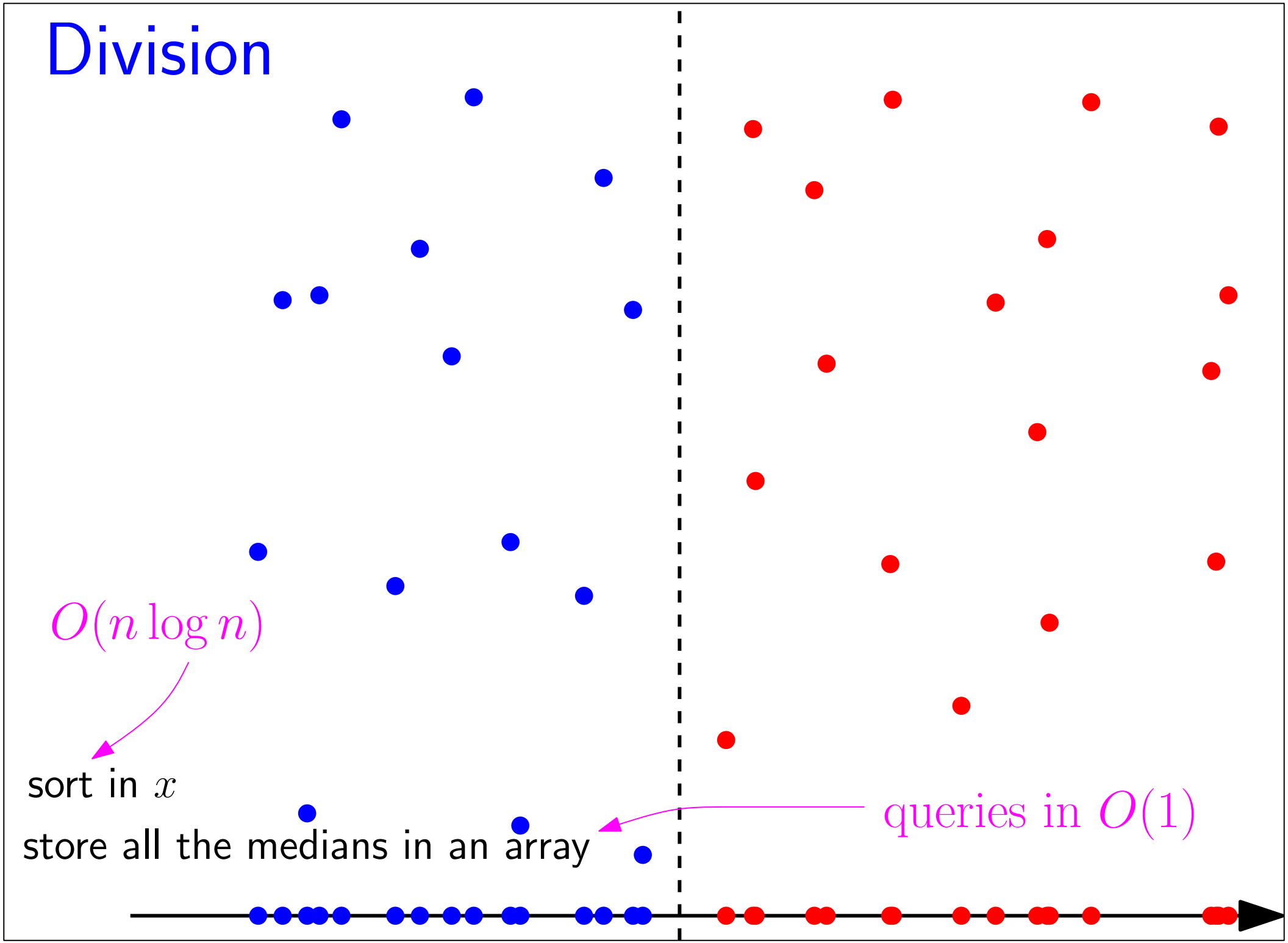



Division
Fusion

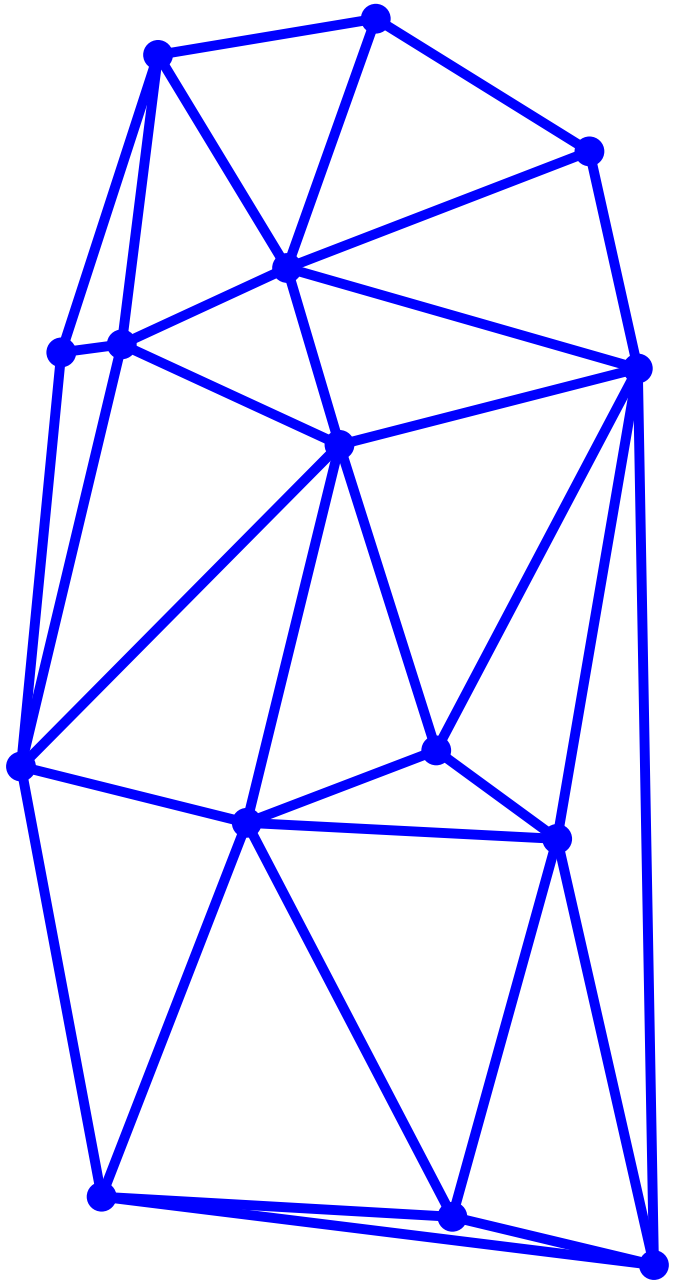

Division

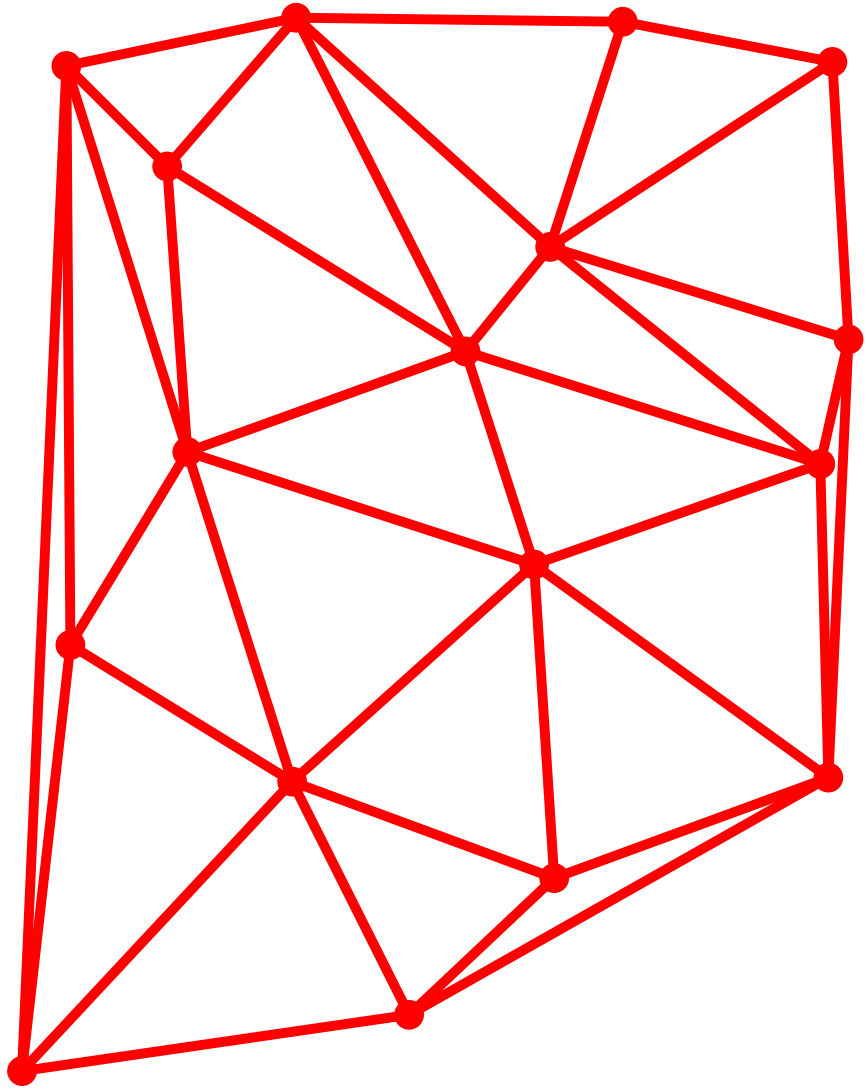

Division
sort in x


Division
sort in x
store all the medians in an array

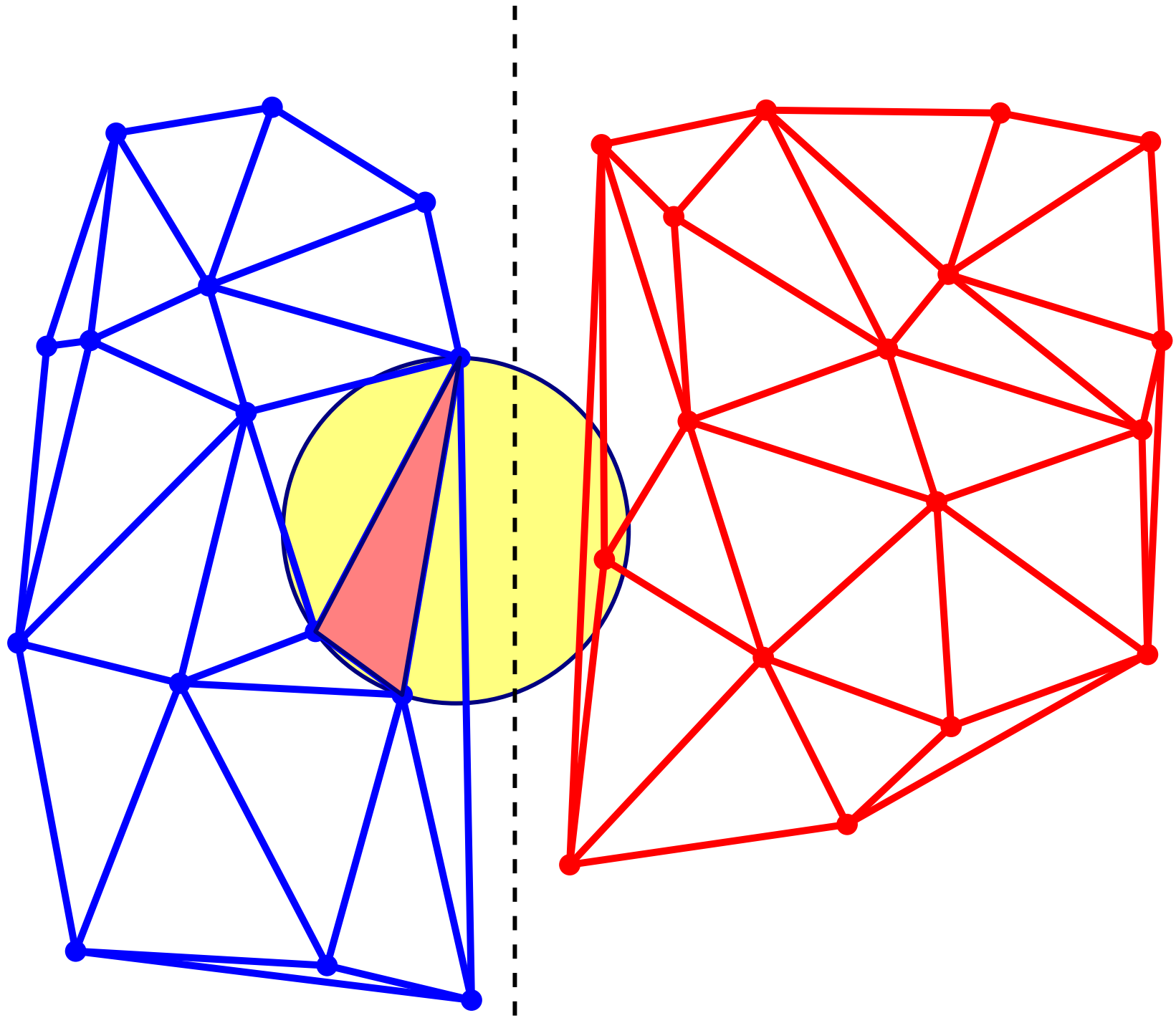

Division
$O(n \log n)$
sort in x
store all the medians in an array
queries in $O(1)$

Fusion



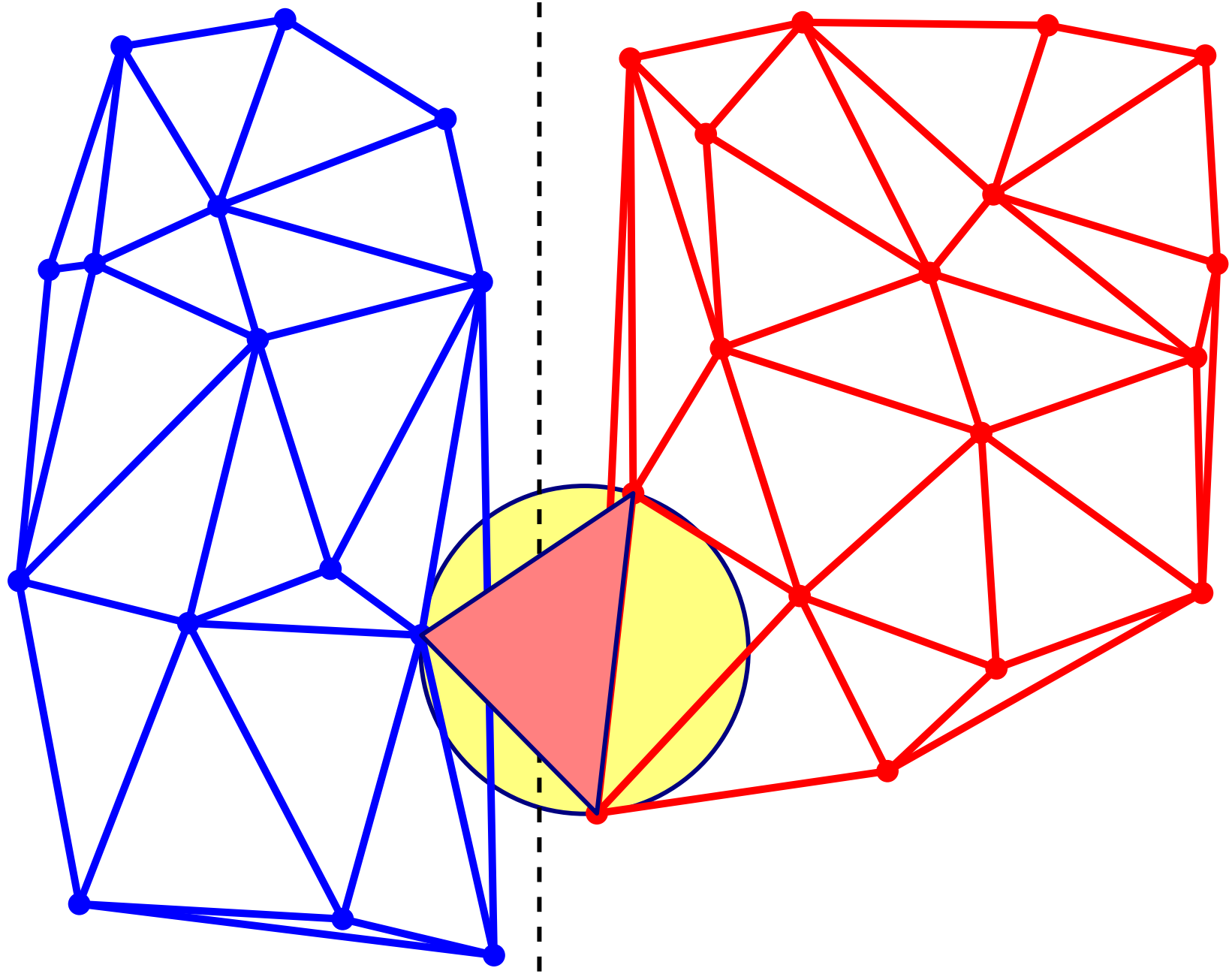


Fusion

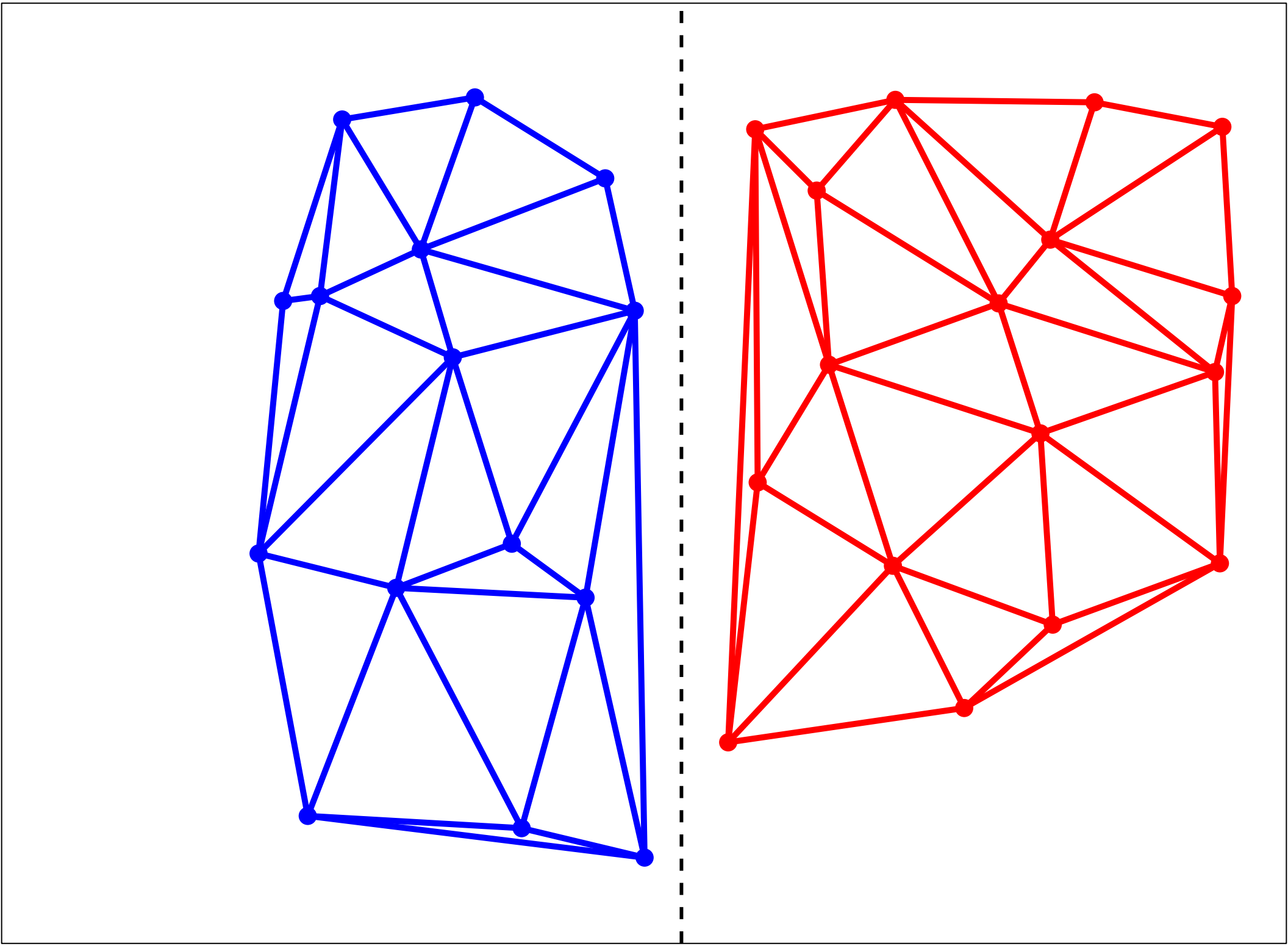


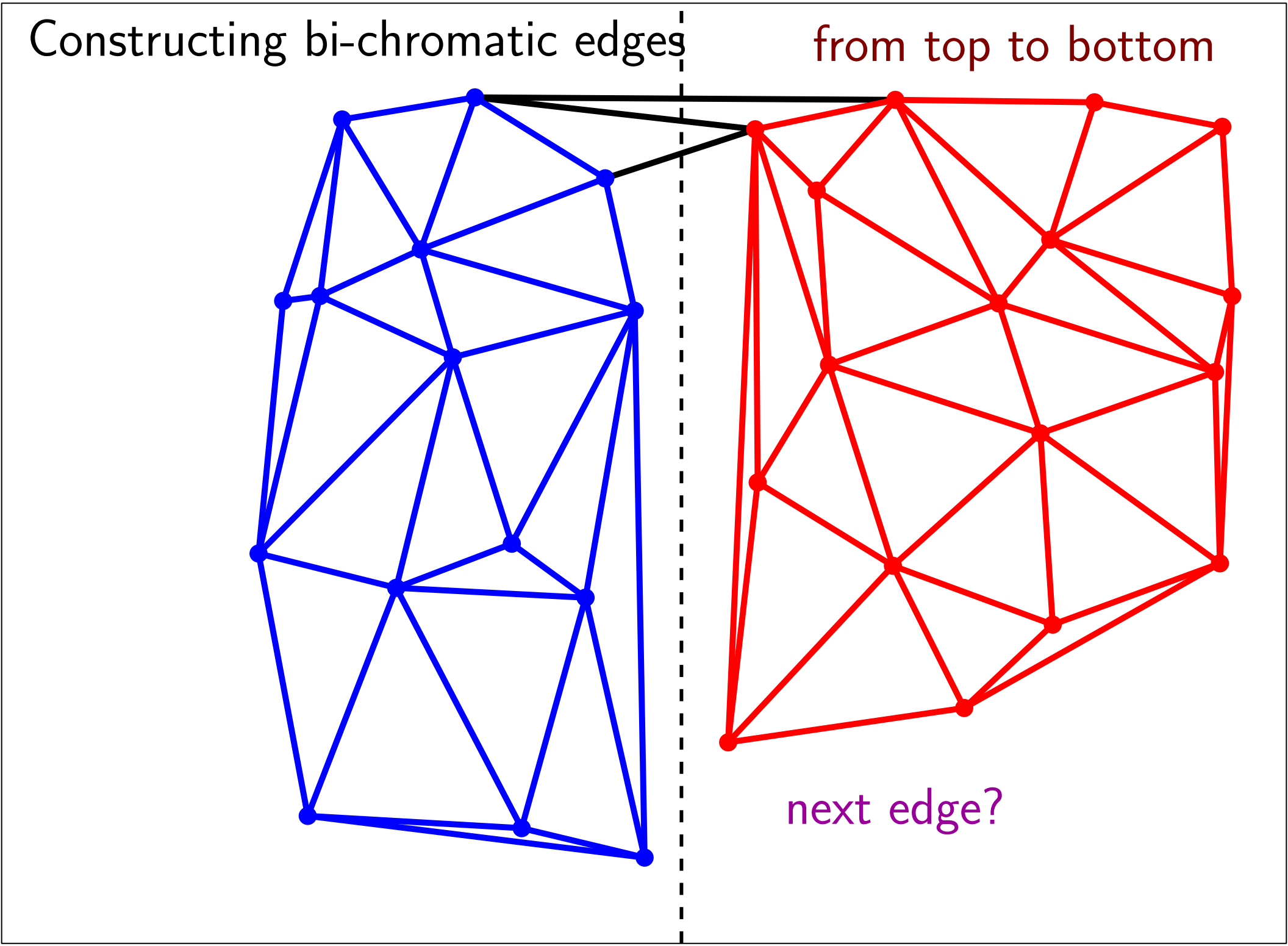
Monochromatic triangles to be deleted

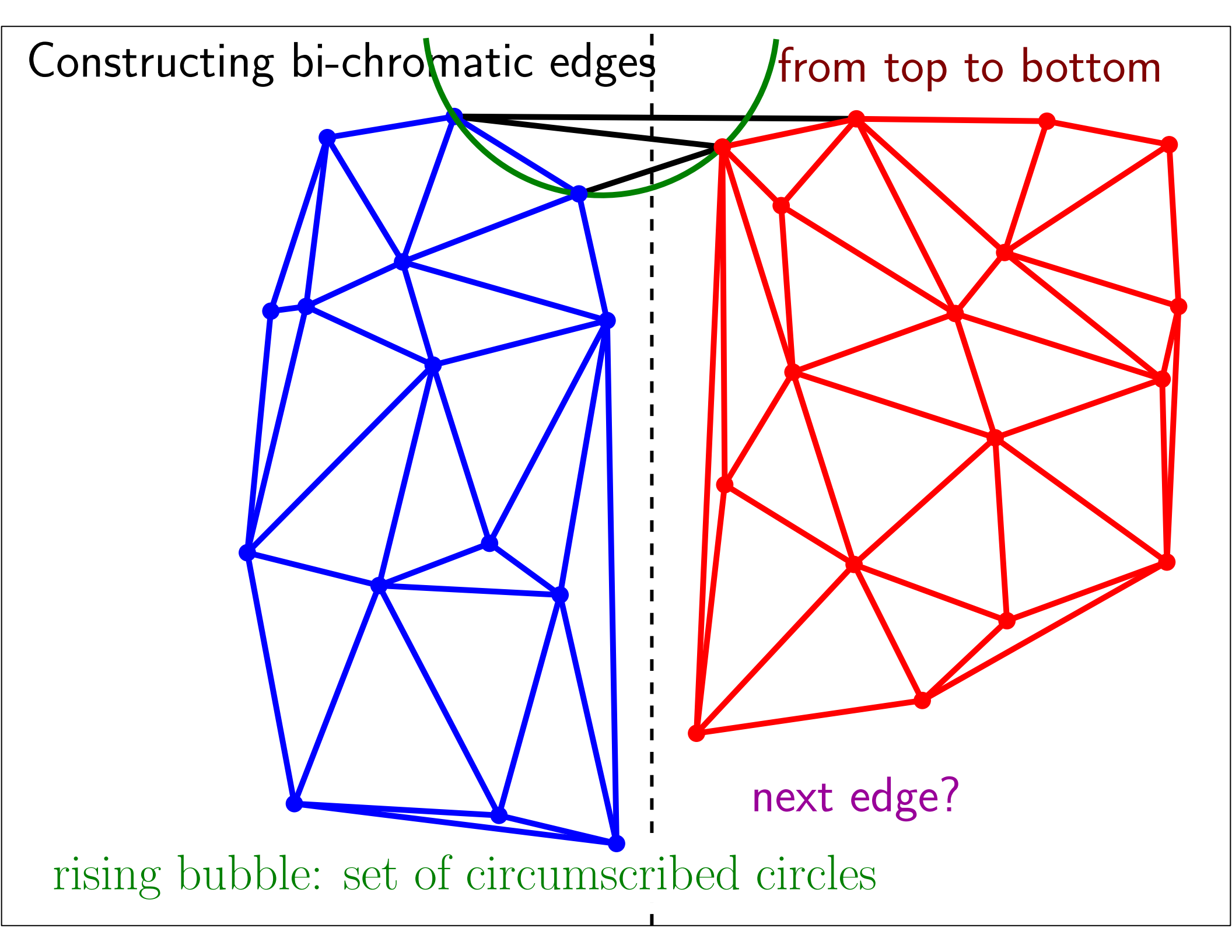
Fusion

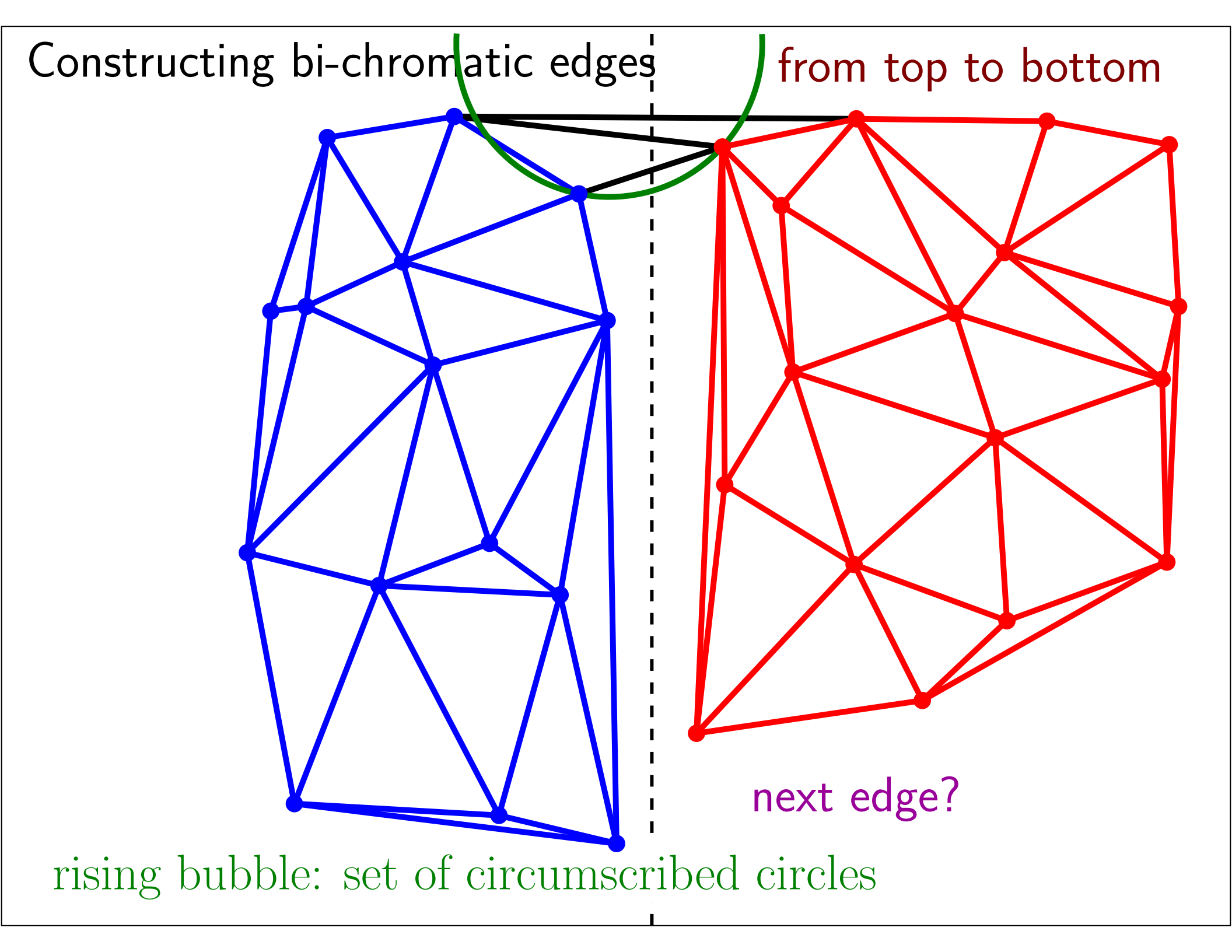


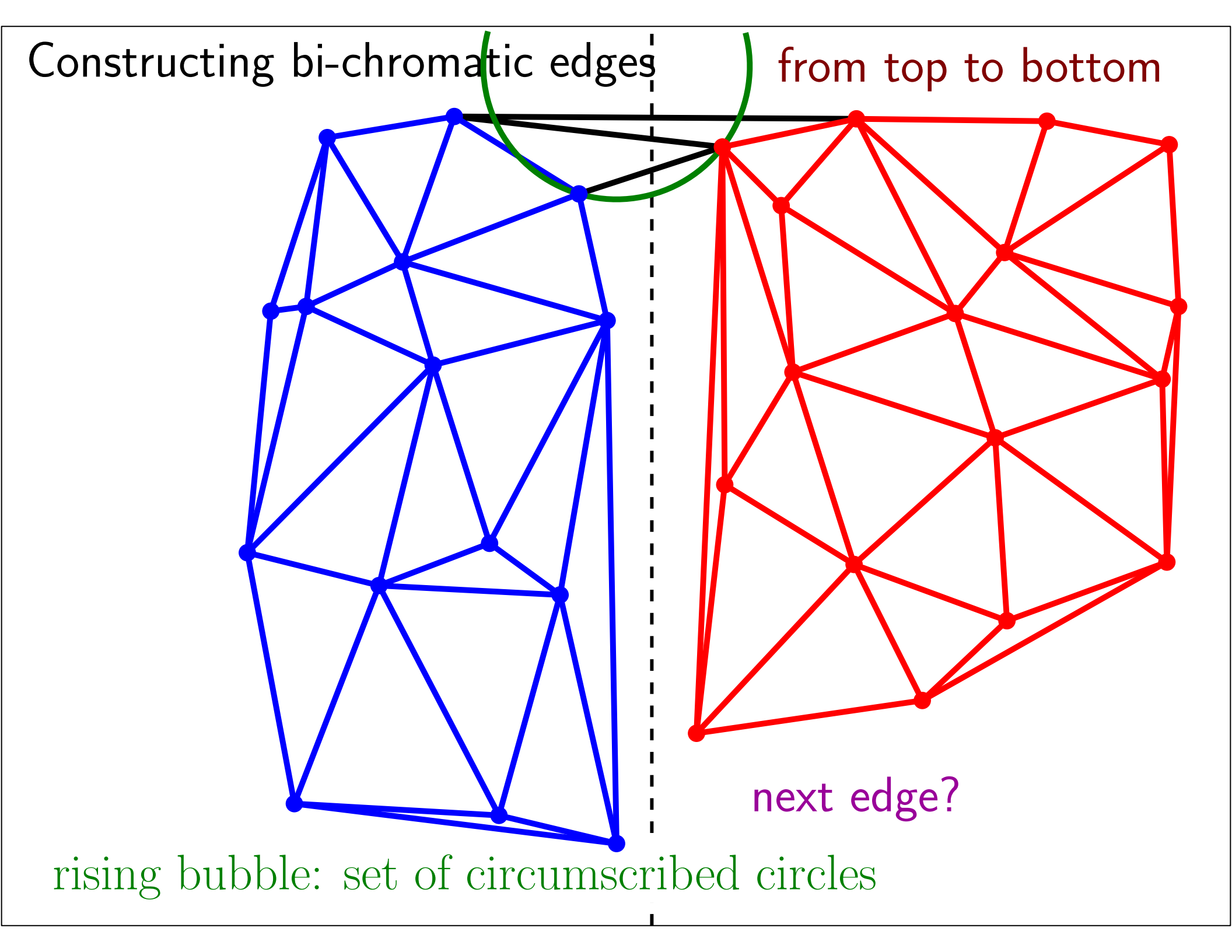
Bi-chromatic triangles to be construced

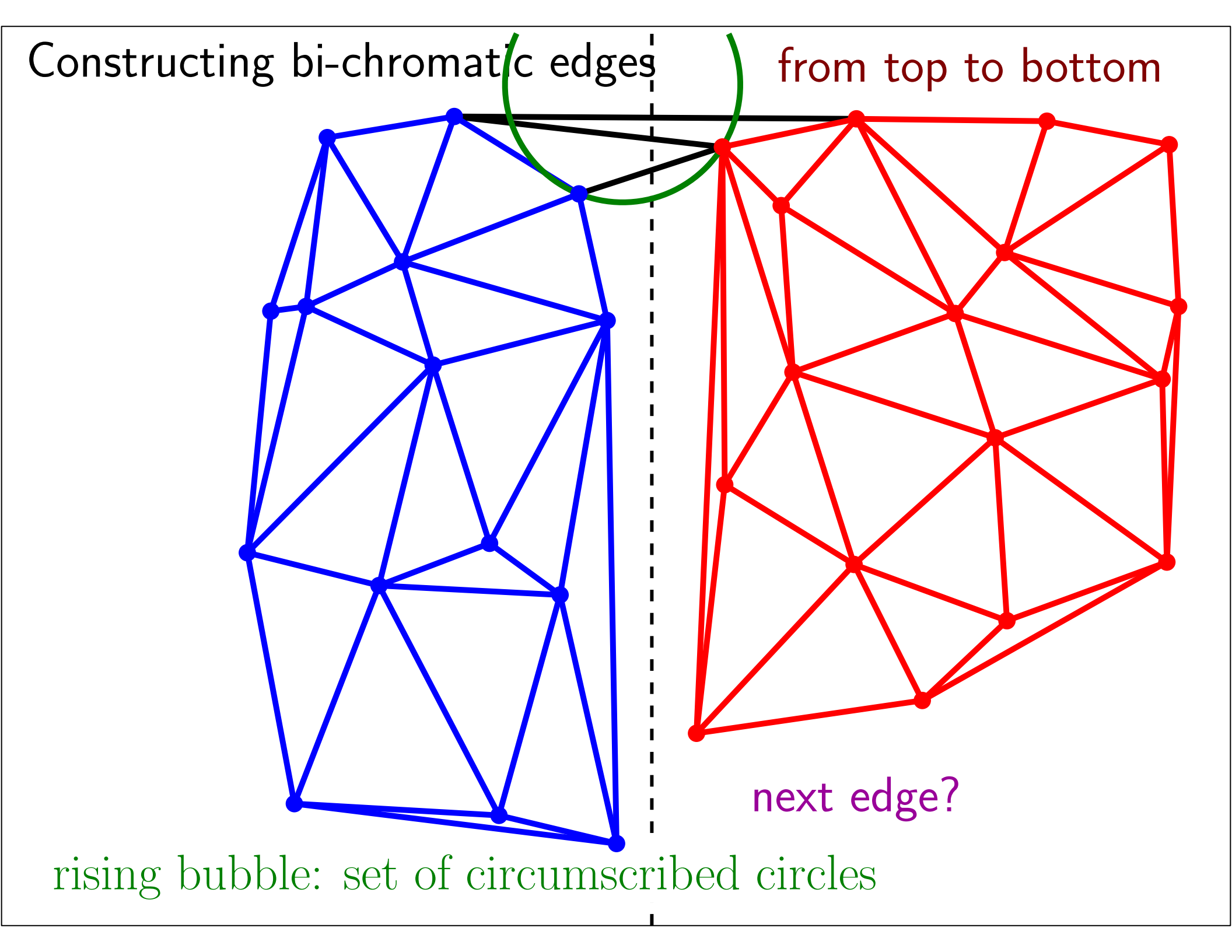


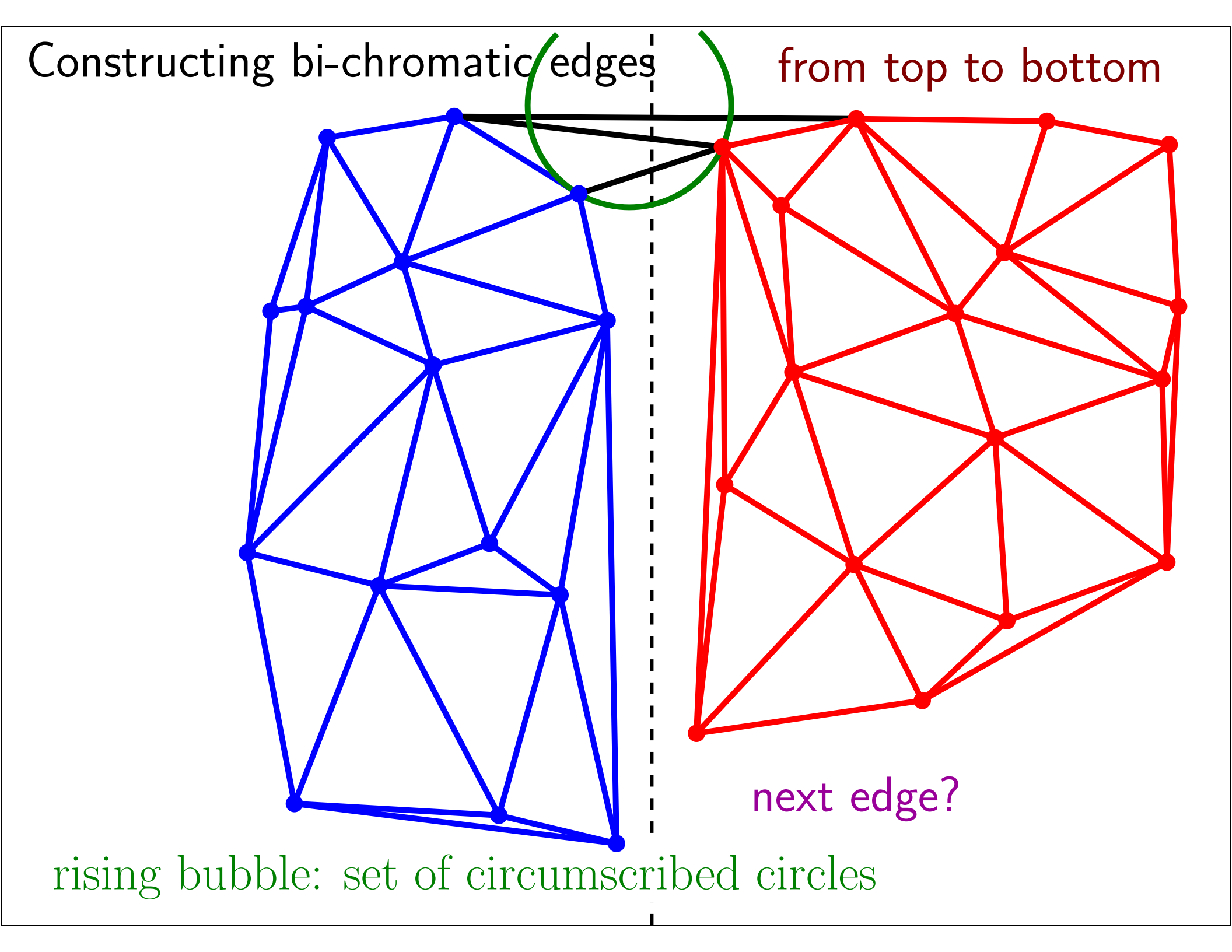

Constructing bi-chromatic edges
from top to bottom
next edge?

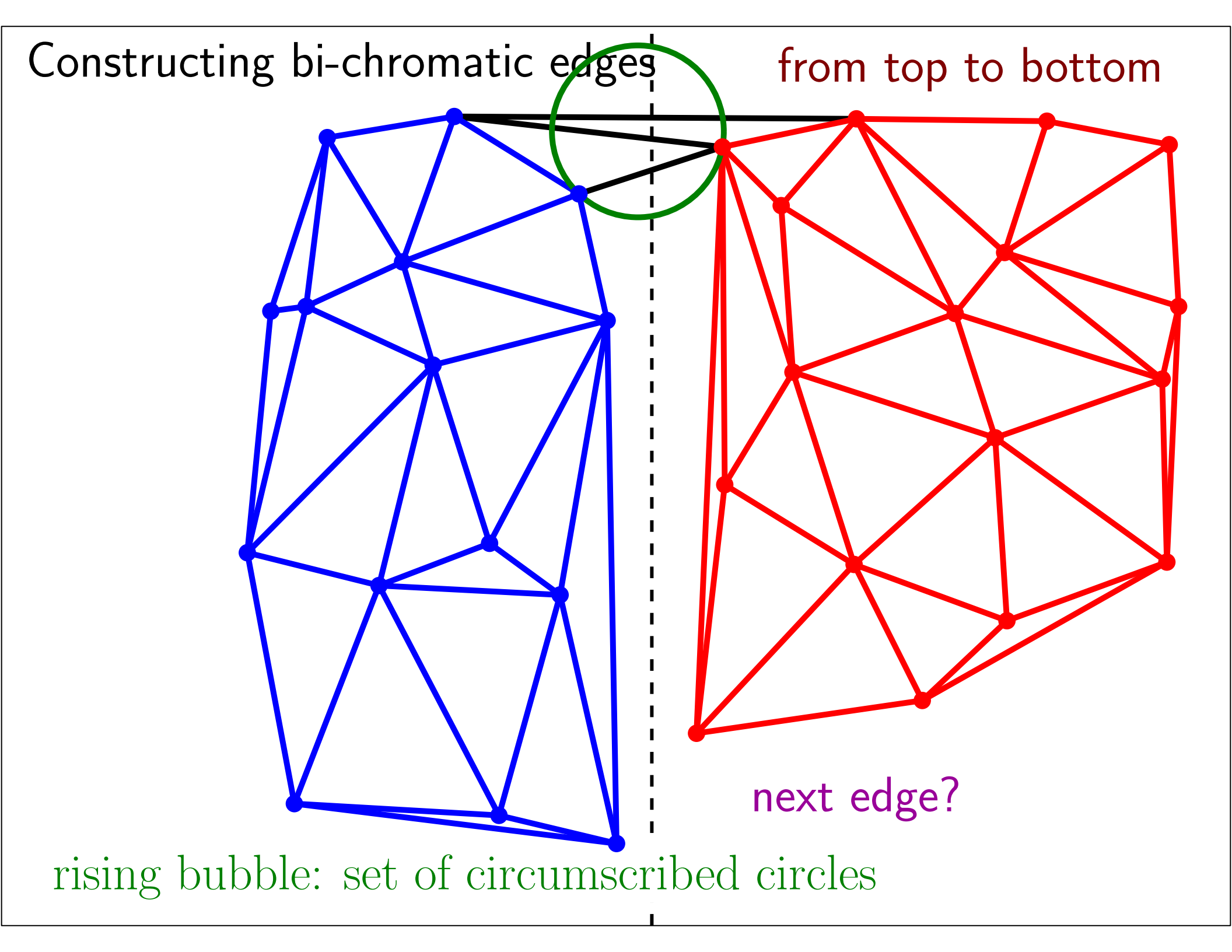

Constructing bi-chromatic edges
from top to bottom
next edge?
rising bubble: set of circumscribed circles

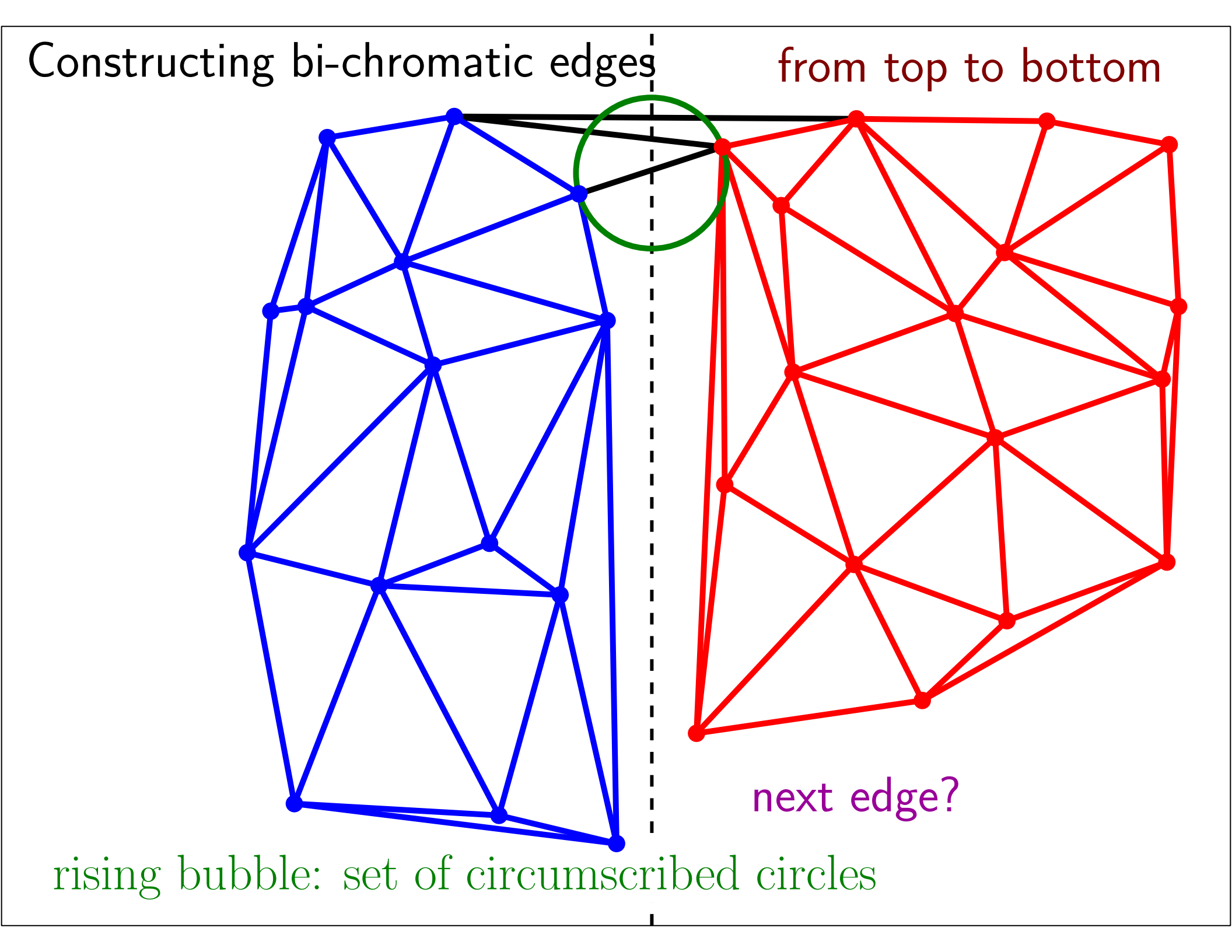

Constructing bi-chromatic edges
from top to bottom
next edge?
rising bubble: set of circumscribed circles

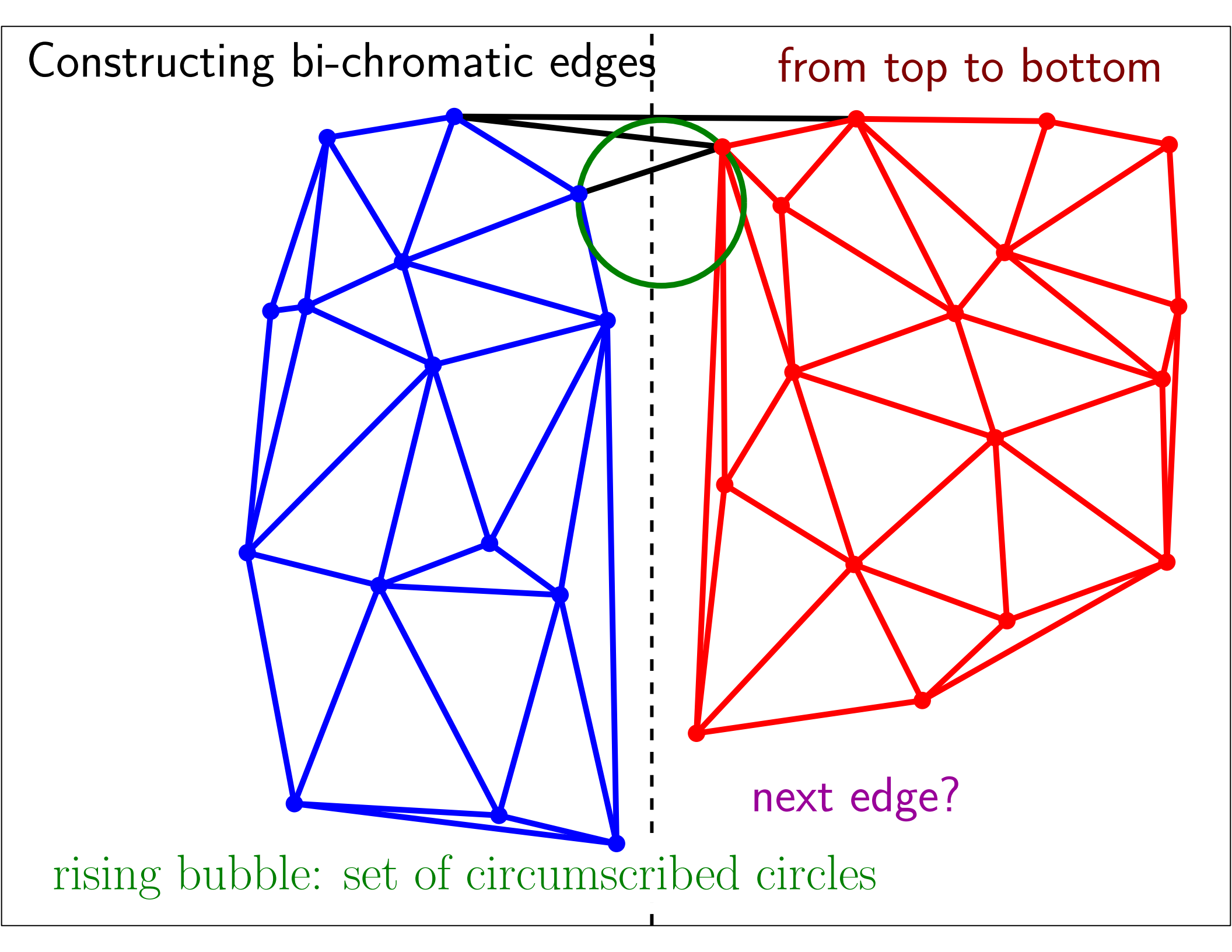

Constructing bi-chromatic edges
from top to bottom
next edge?
rising bubble: set of circumscribed circles

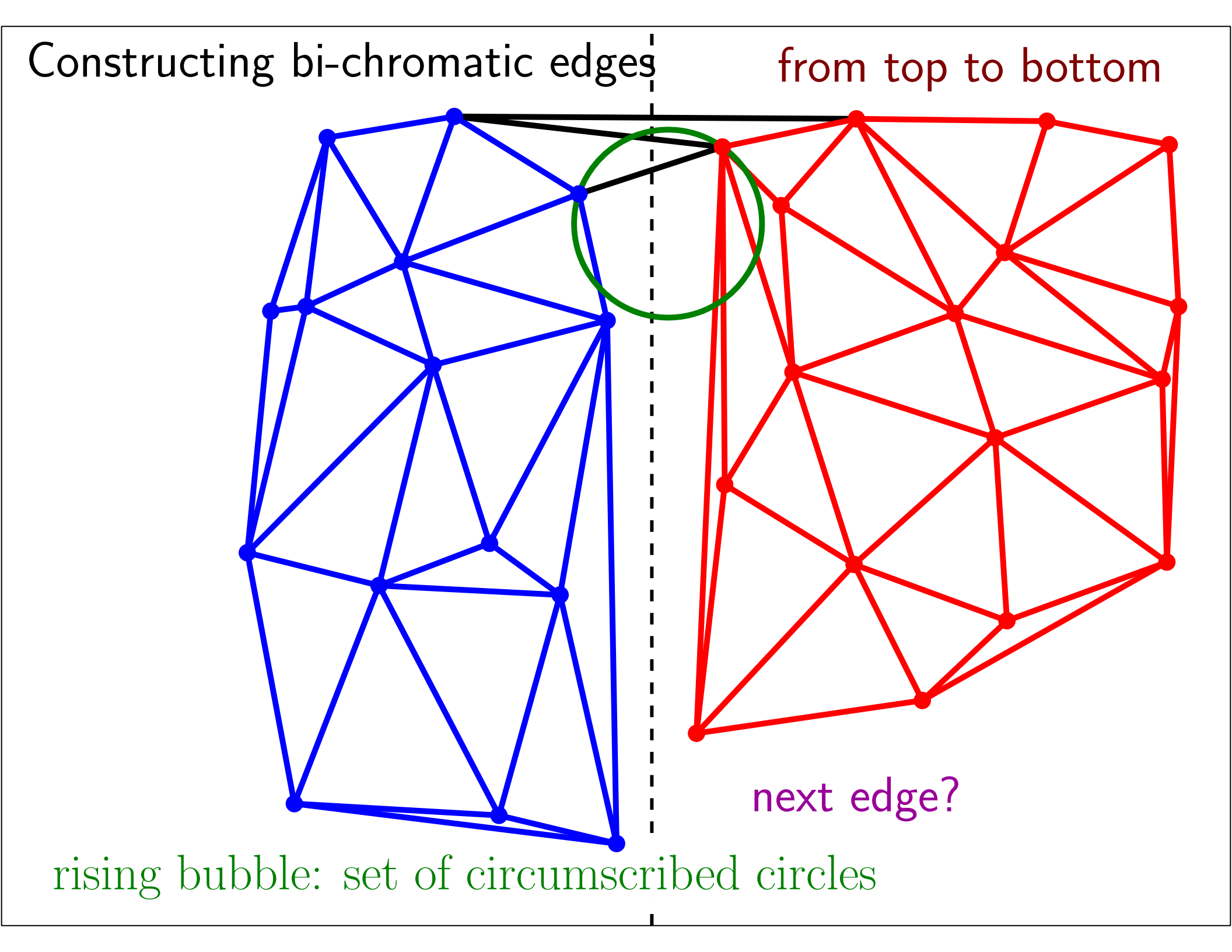

Constructing bi-chromatic edges
from top to bottom
next edge?
rising bubble: set of circumscribed circles

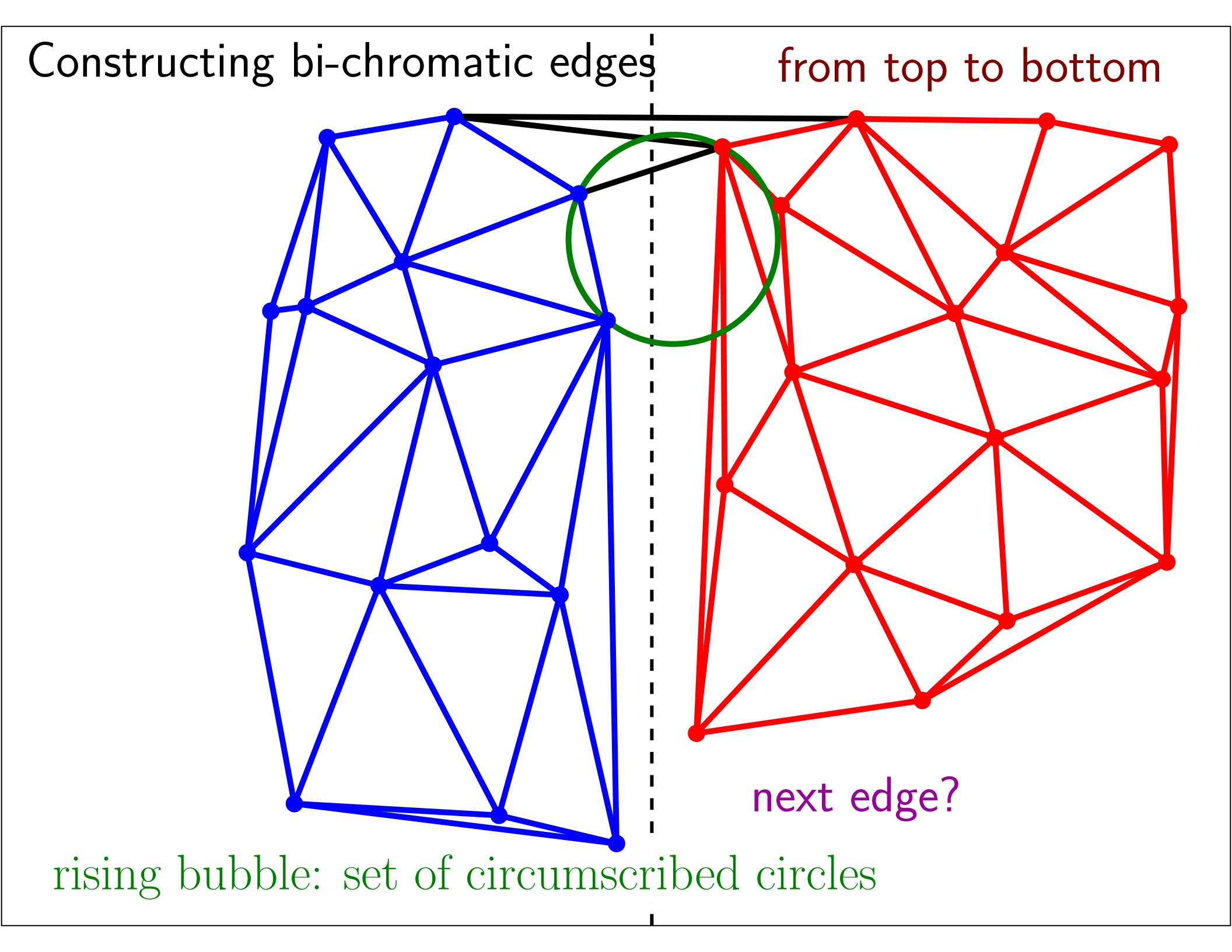

Constructing bi-chromatic edges
from top to bottom
next edge?
rising bubble: set of circumscribed circles

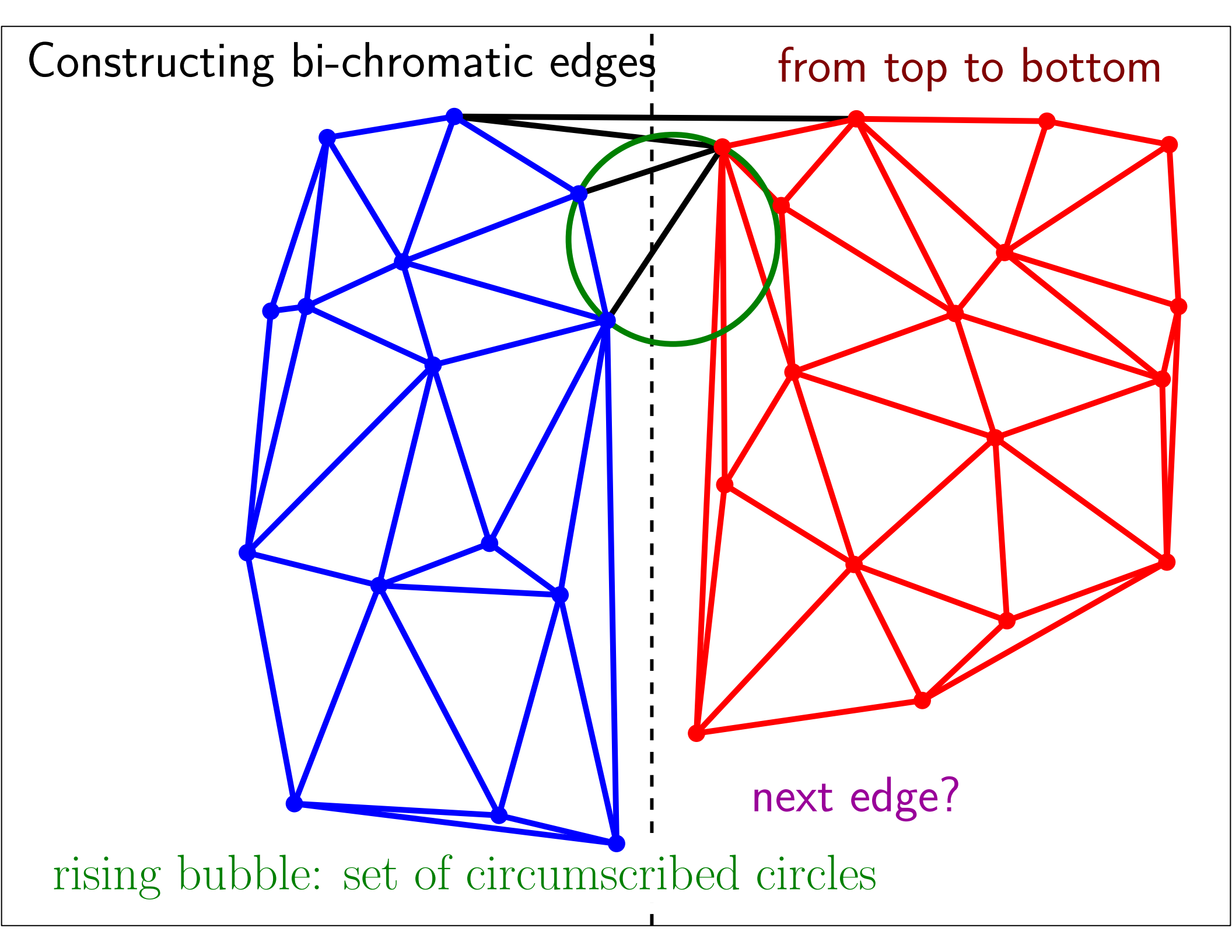

Constructing bi-chromatic edges
from top to bottom
next edge?
rising bubble: set of circumscribed circles

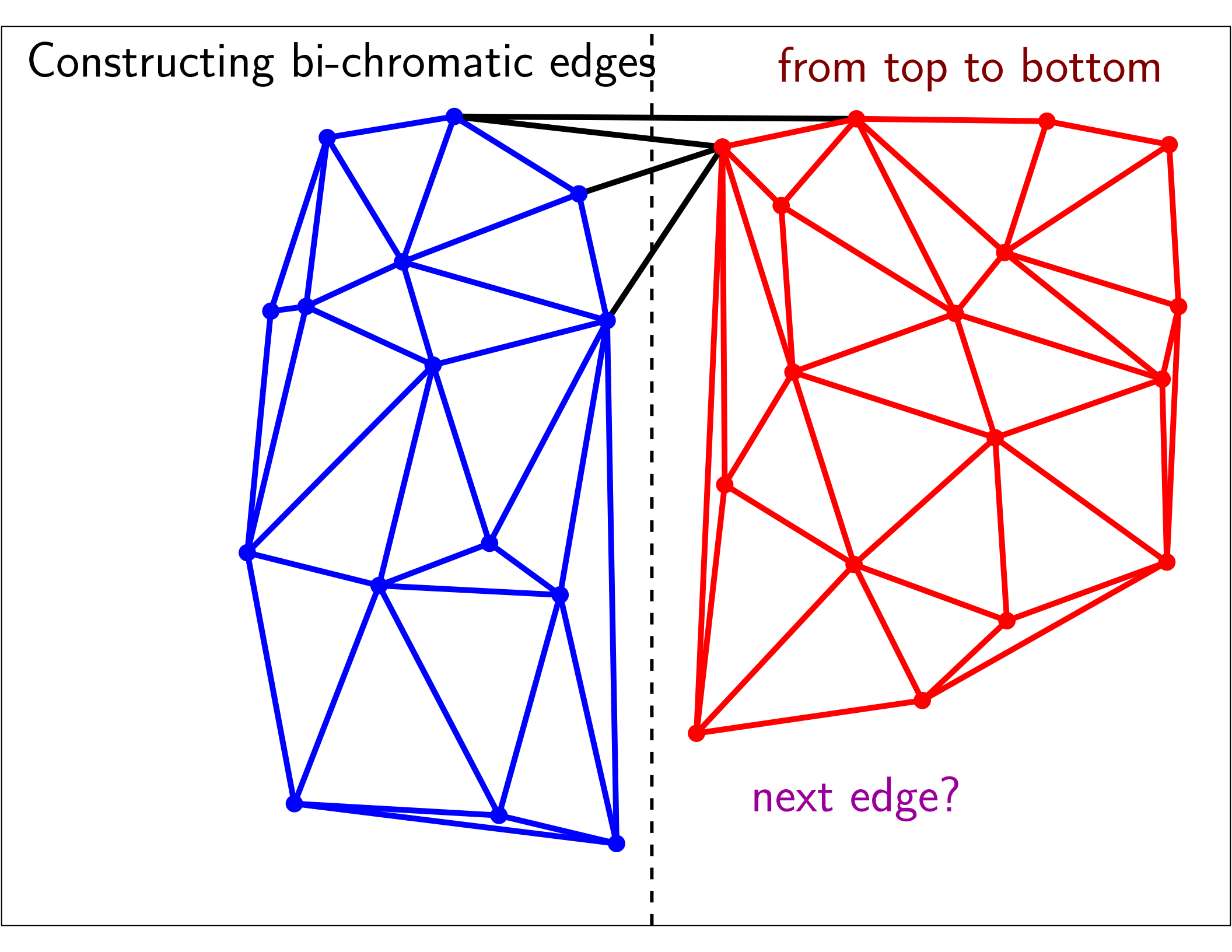

Constructing bi-chromatic edges
from top to bottom
next edge?
rising bubble: set of circumscribed circles

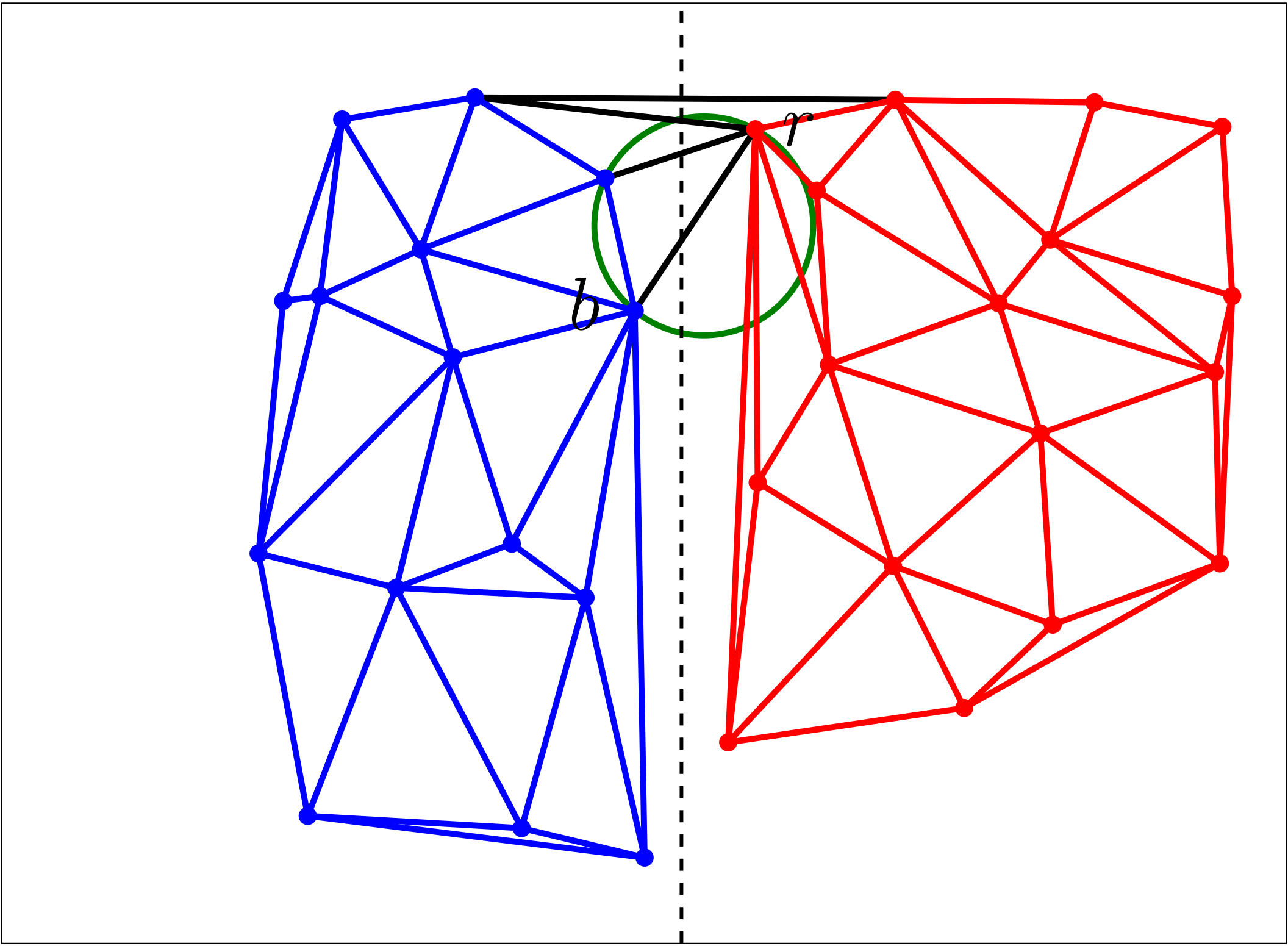

Constructing bi-chromatic edges
from top to bottom
next edge?
rising bubble: set of circumscribed circles

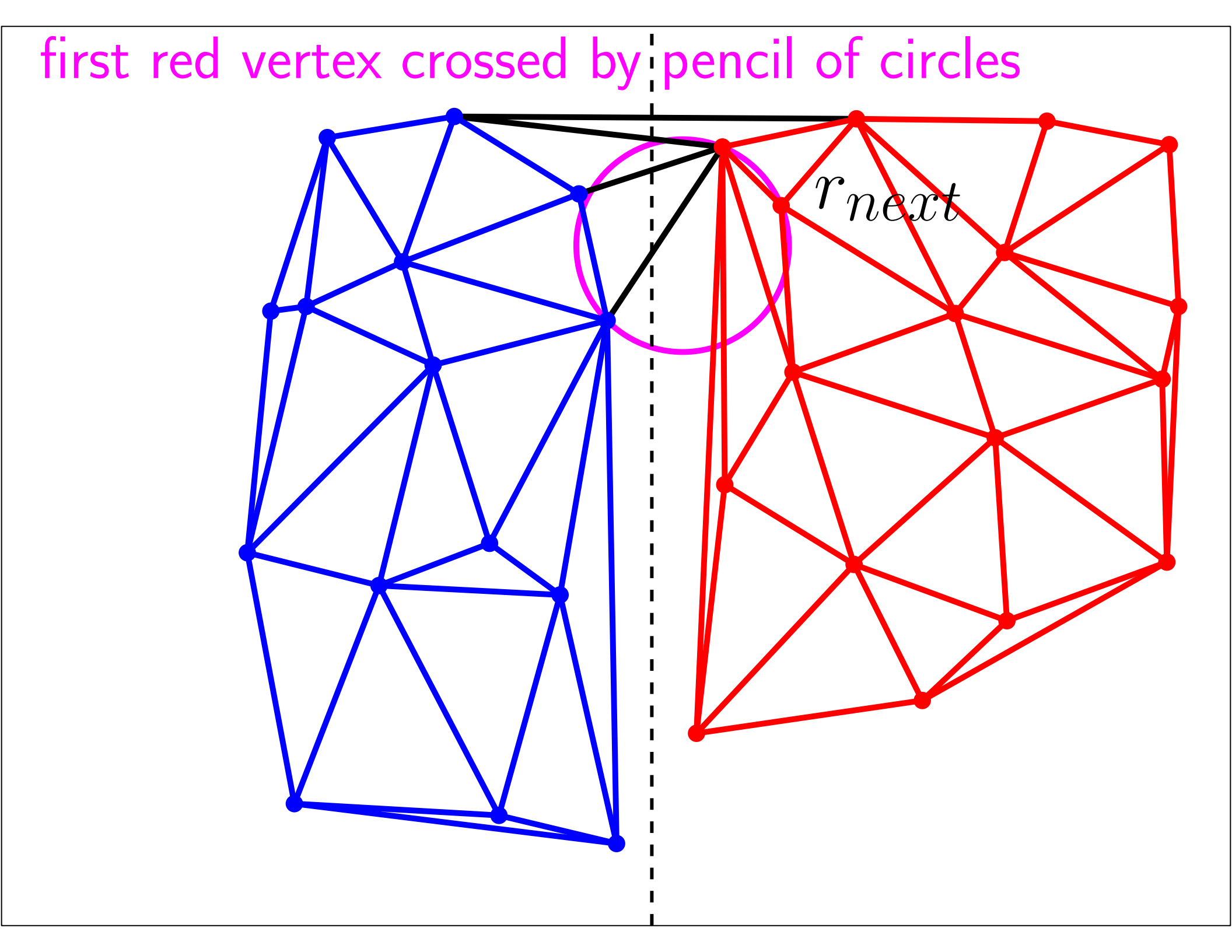

Constructing bi-chromatic edges
from top to bottom
next edge?
rising bubble: set of circumscribed circles

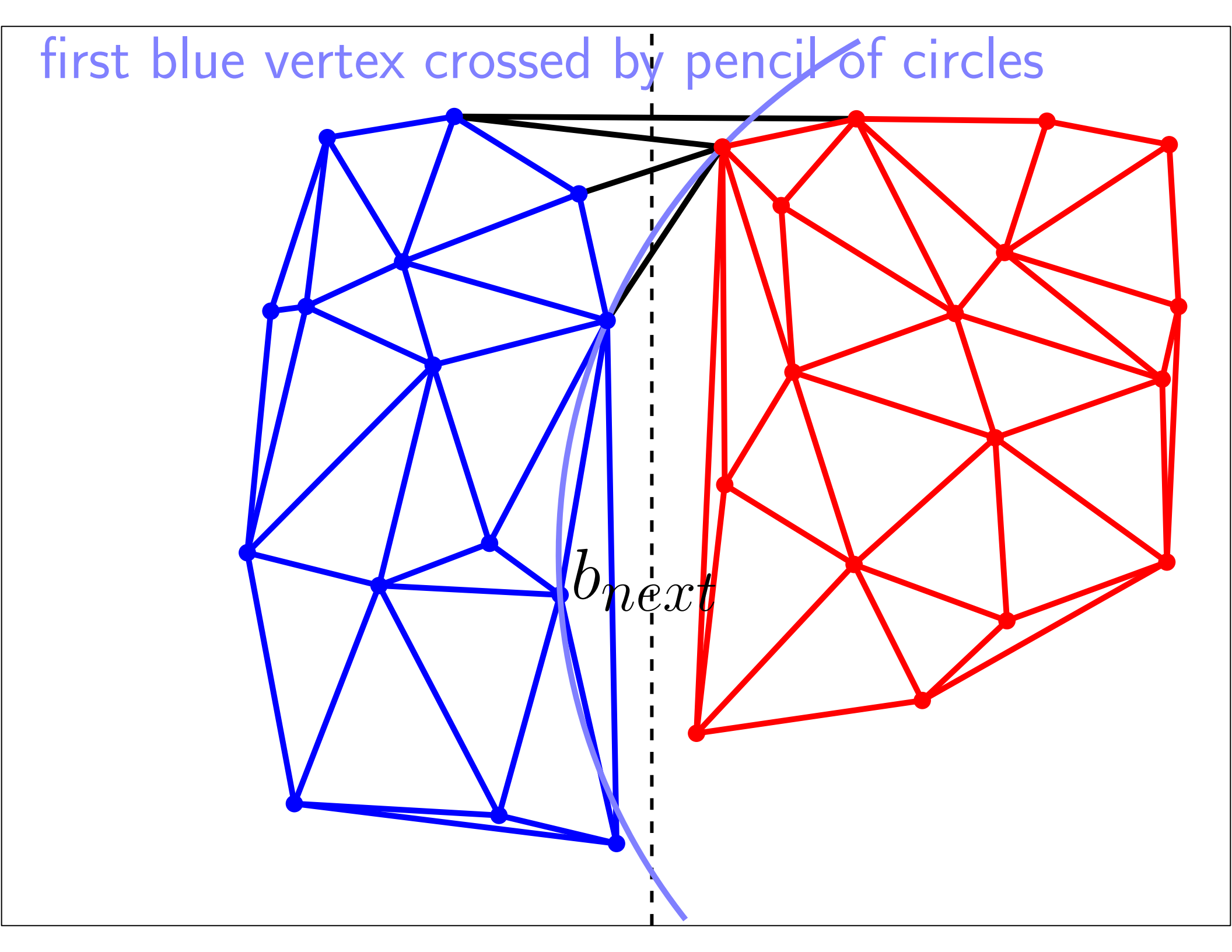

Constructing bi-chromatic edges
from top to bottom
next edge?
rising bubble: set of circumscribed circles


Constructing bi-chromatic edges
from top to bottom
next edge?
rising bubble: set of circumscribed circles

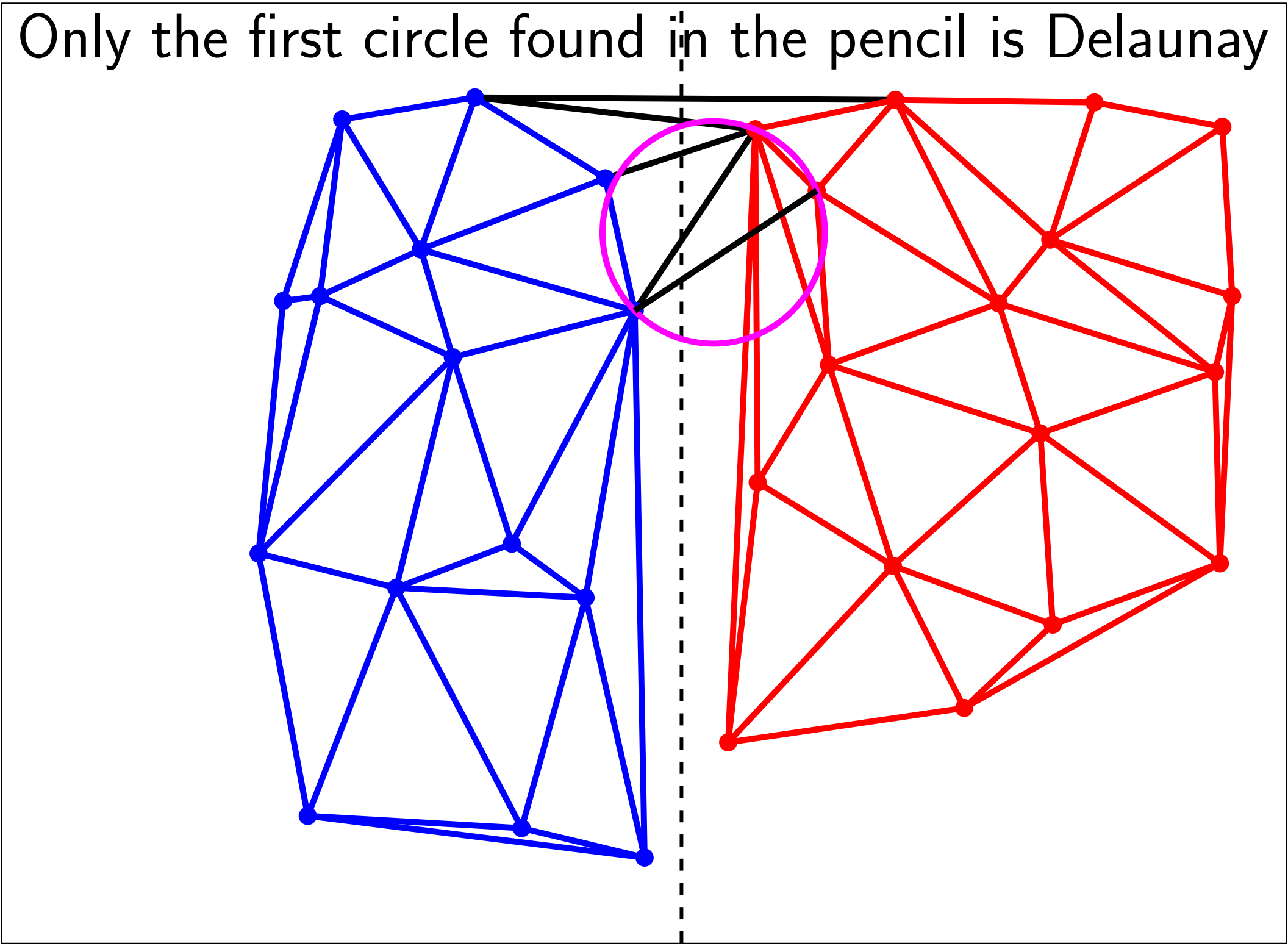

Constructing bi-chromatic edges
from top to bottom
next edge?

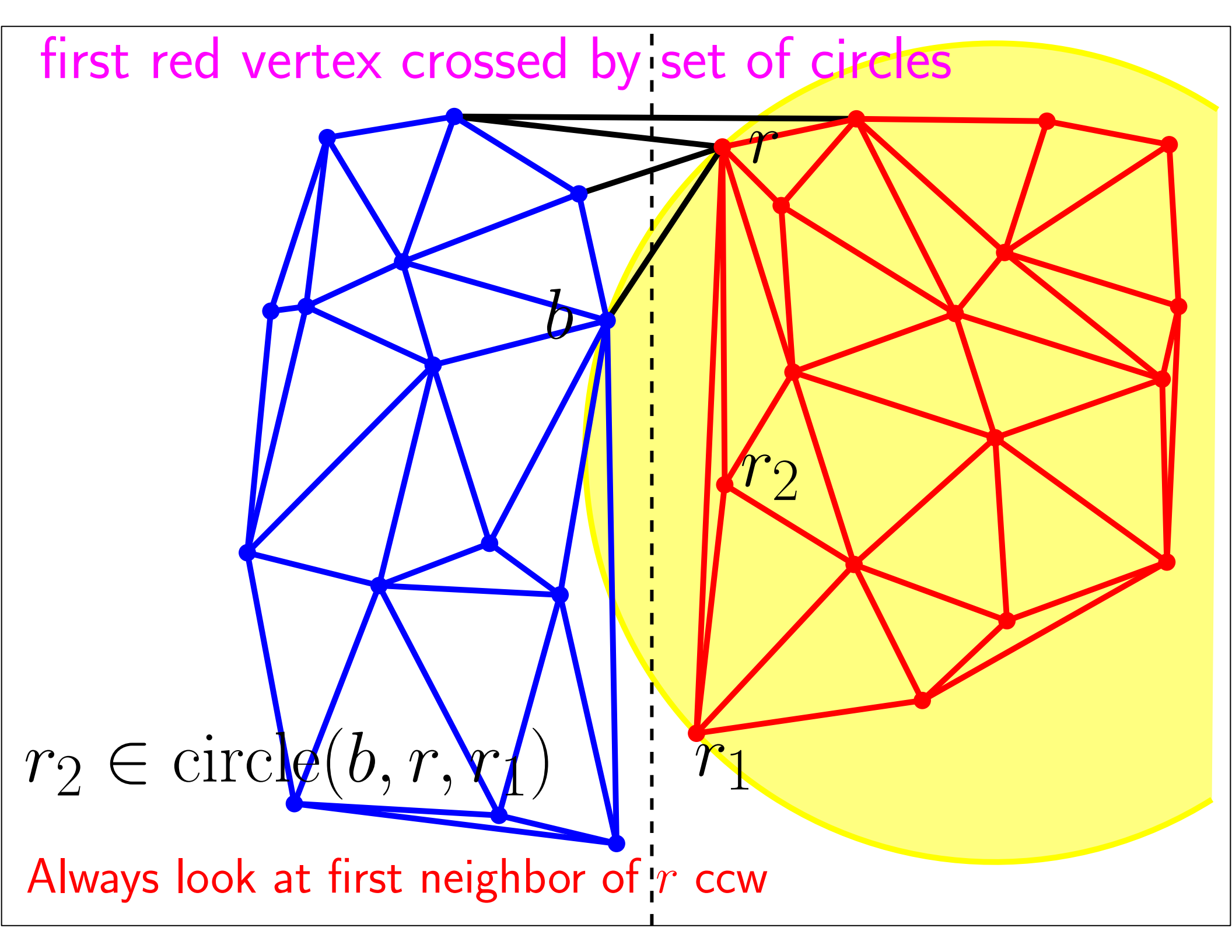

r
b


first red vertex crossed by pencil of circles
r_{next}

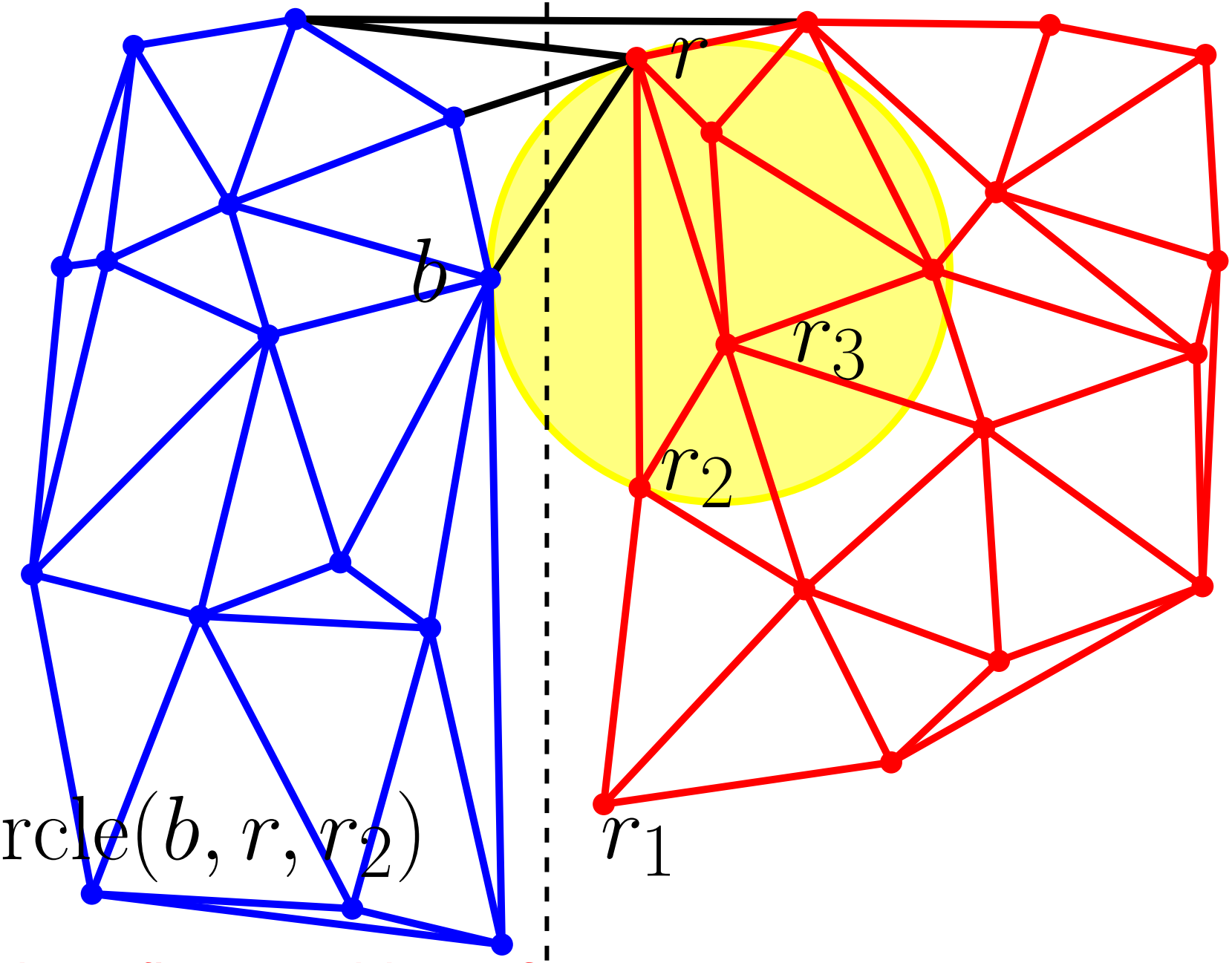

first blue vertex crossed by pencil of circles
b_{next}

Only the first circle found in the pencil is Delaunay




first red vertex crossed by set of circles
r
b
r_2
r_1
$r_2 \in \text{circle}(b, r, r_1)$
Always look at first neighbor of r ccw

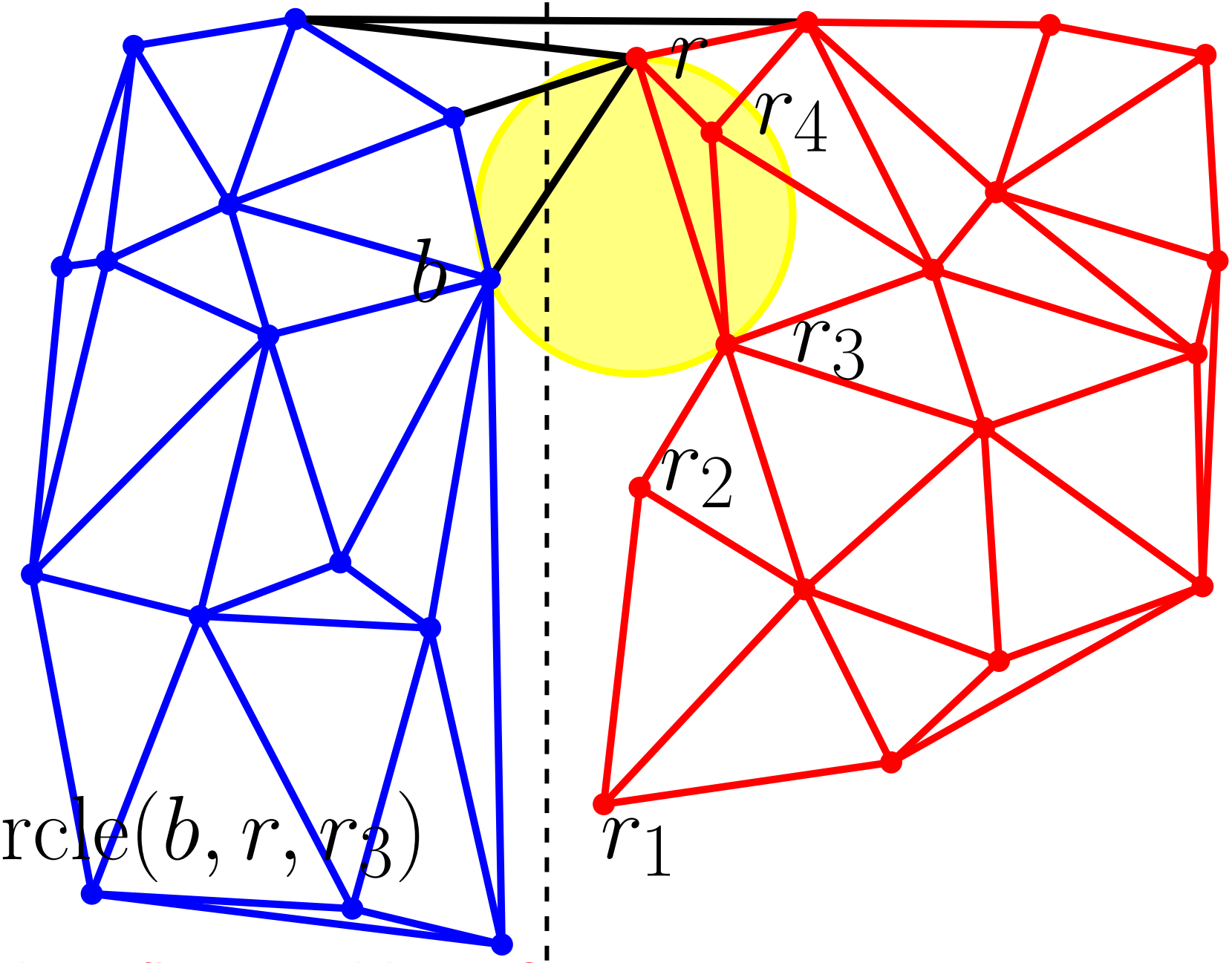
first red vertex crossed by set of circles



$r_3 \in \text{circle}(b, r, r_2)$

Always look at first neighbor of r ccw

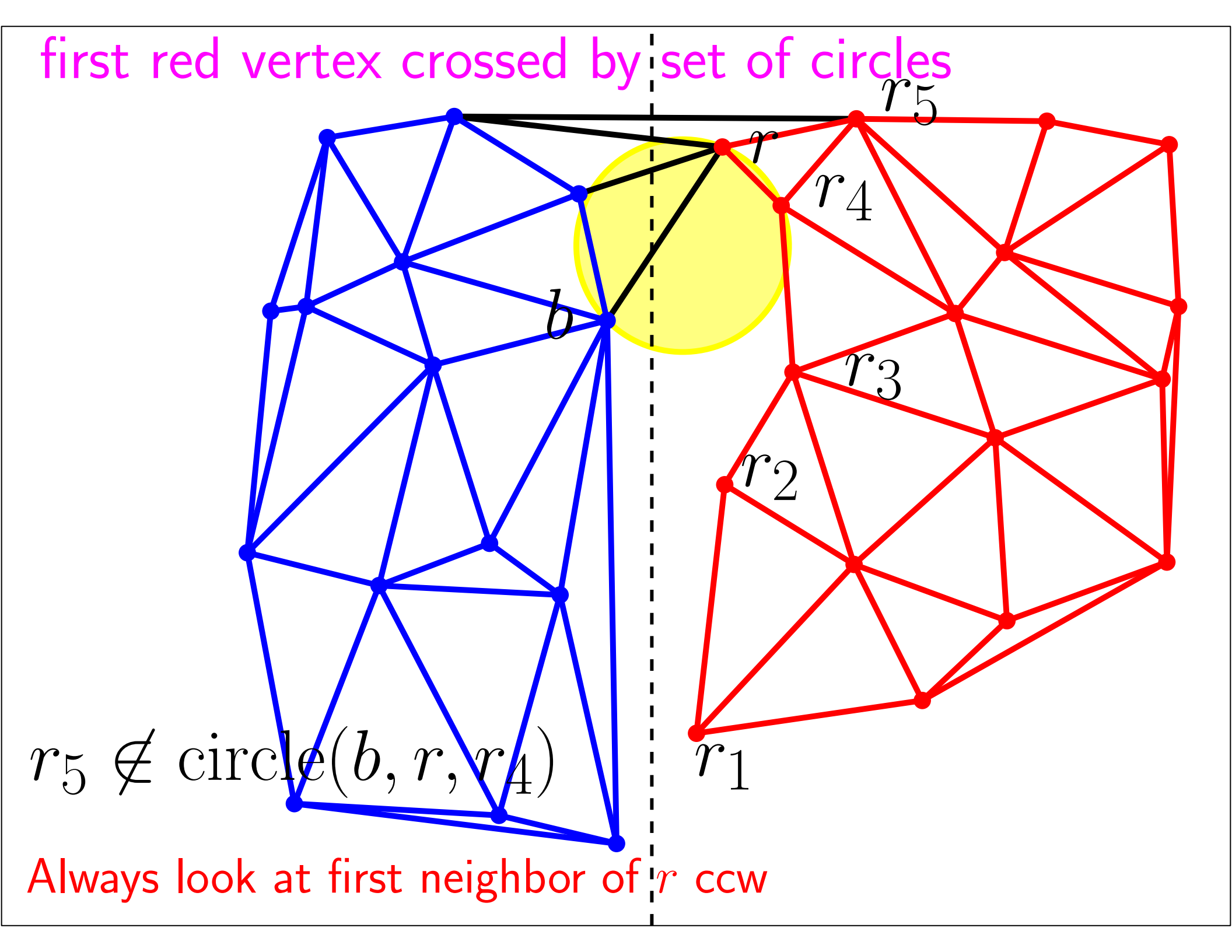
first red vertex crossed by set of circles



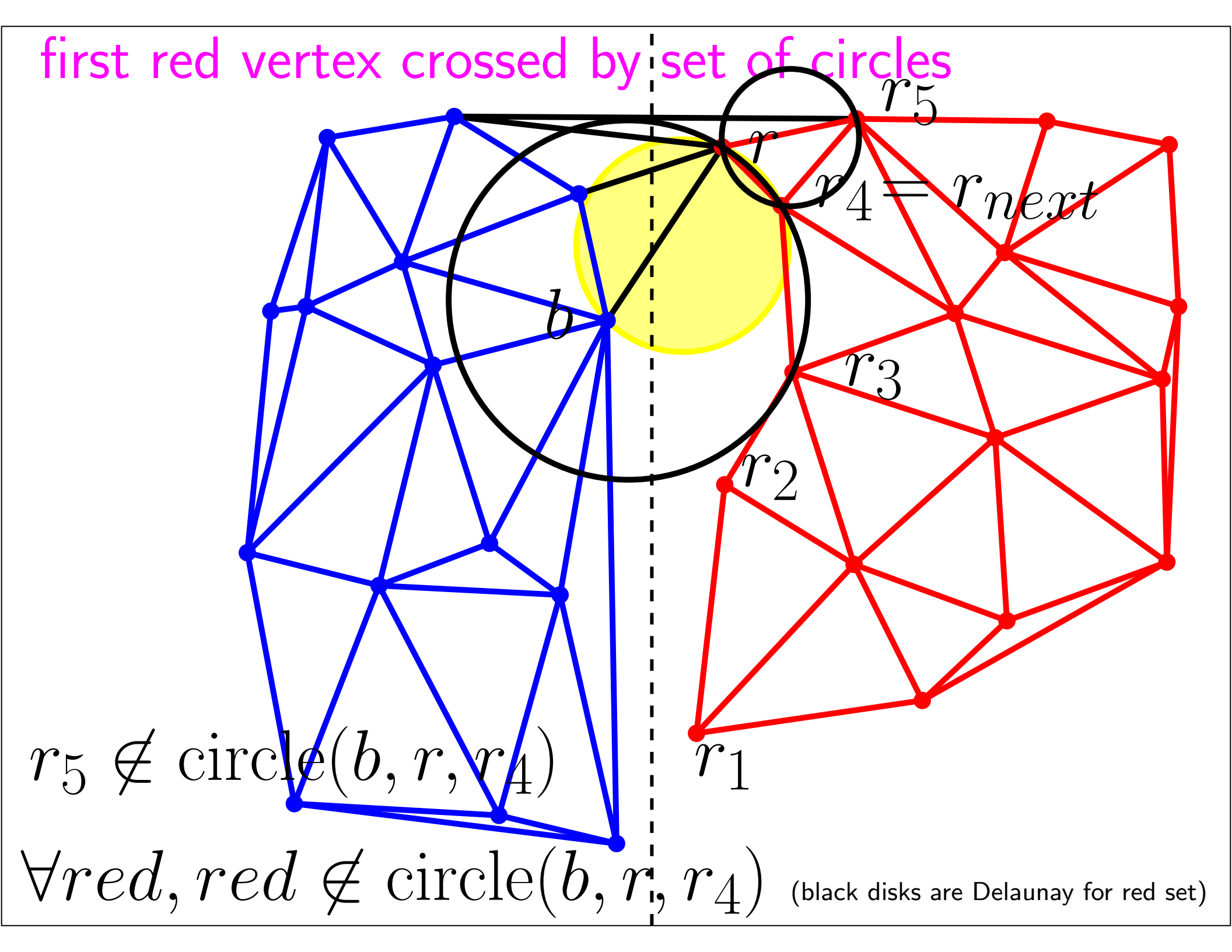
$r_4 \in \text{circle}(b, r, r_3)$

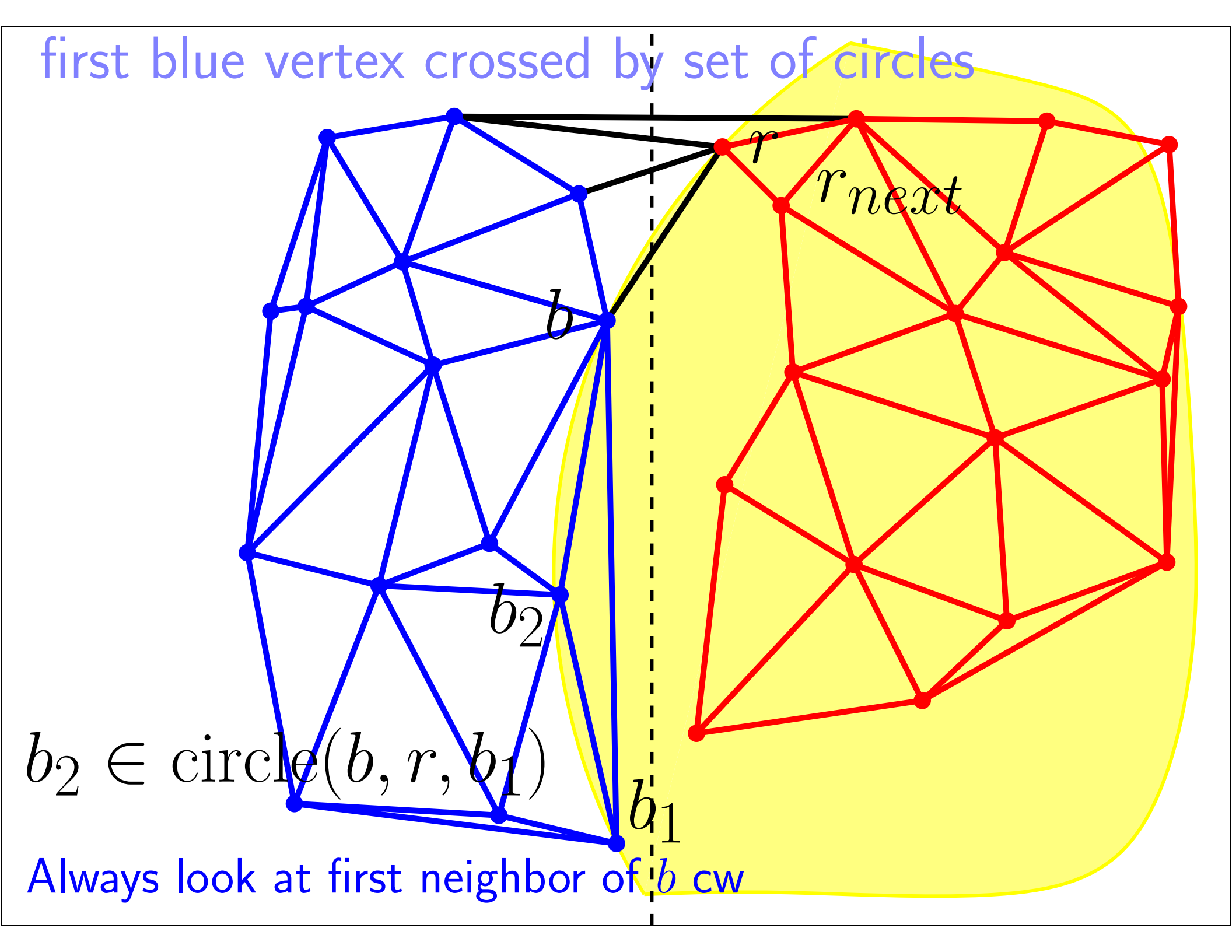
Always look at first neighbor of r ccw

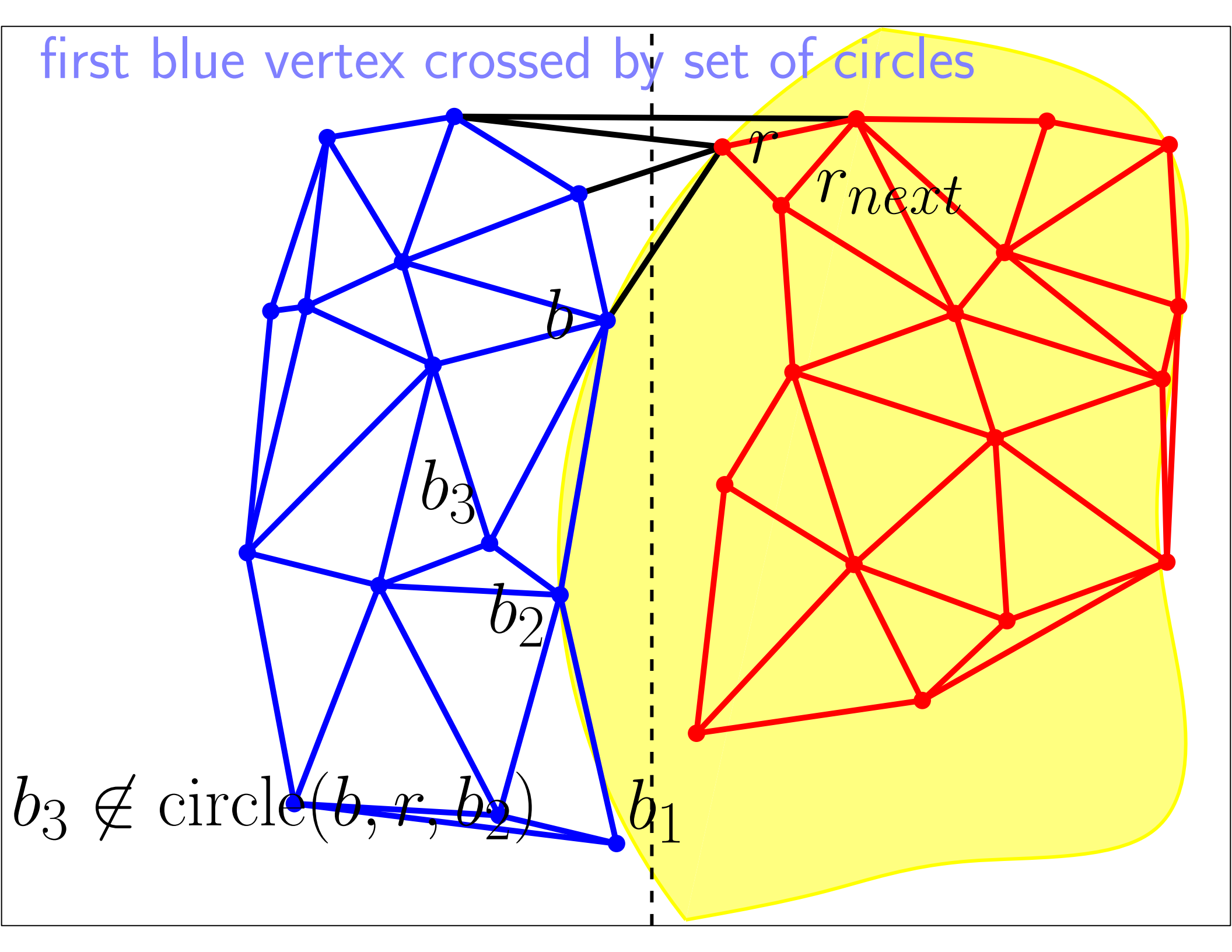
first red vertex crossed by set of circles

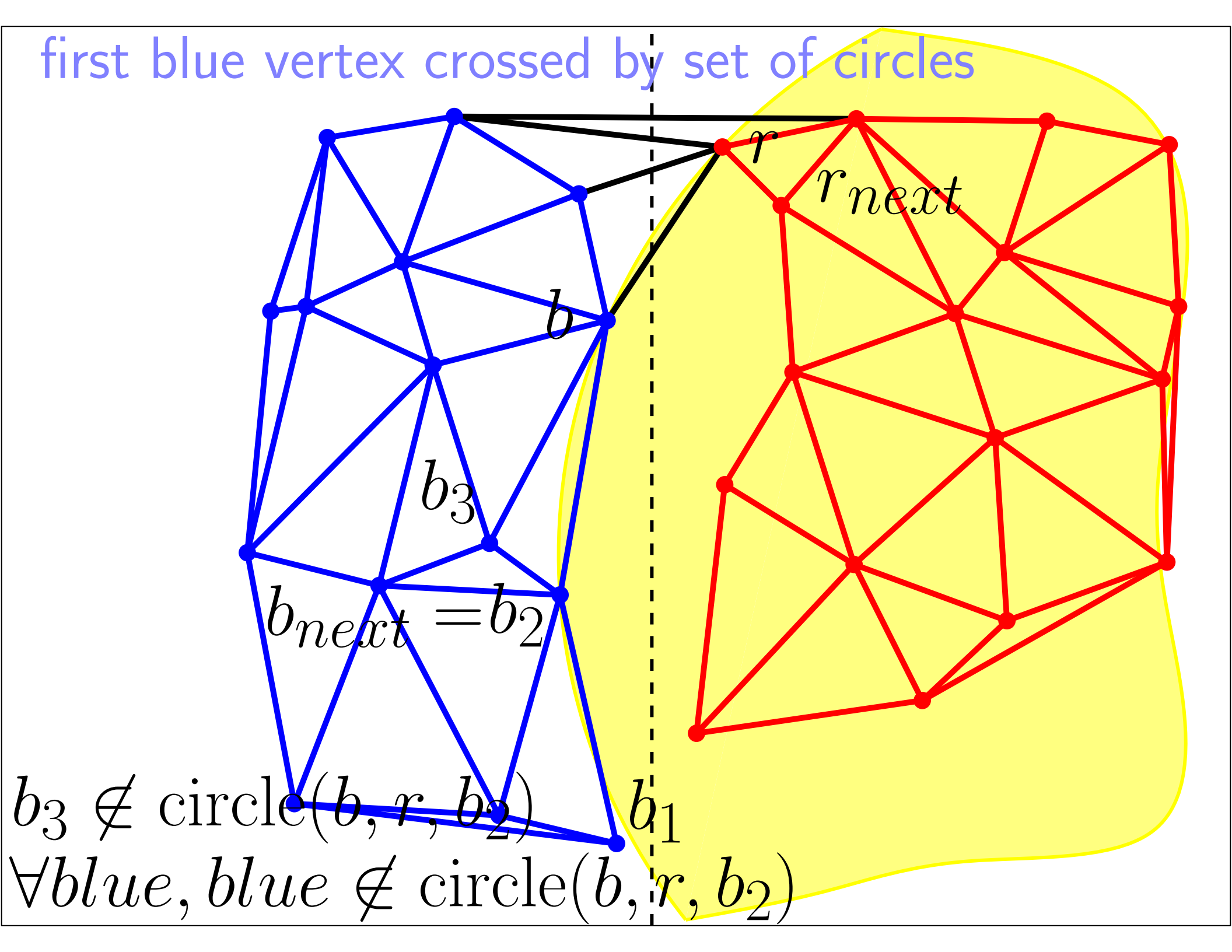


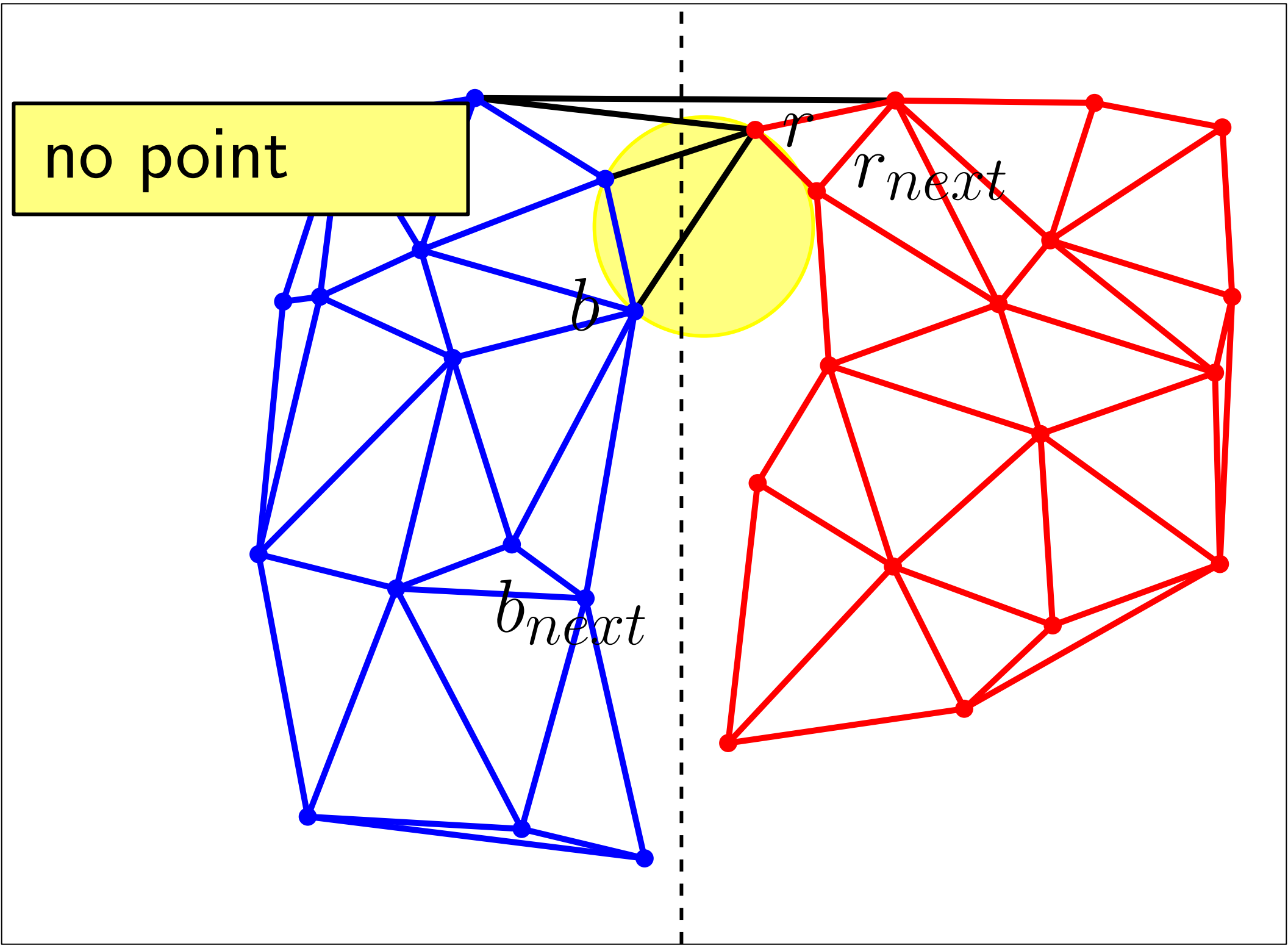
Always look at first neighbor of r ccw

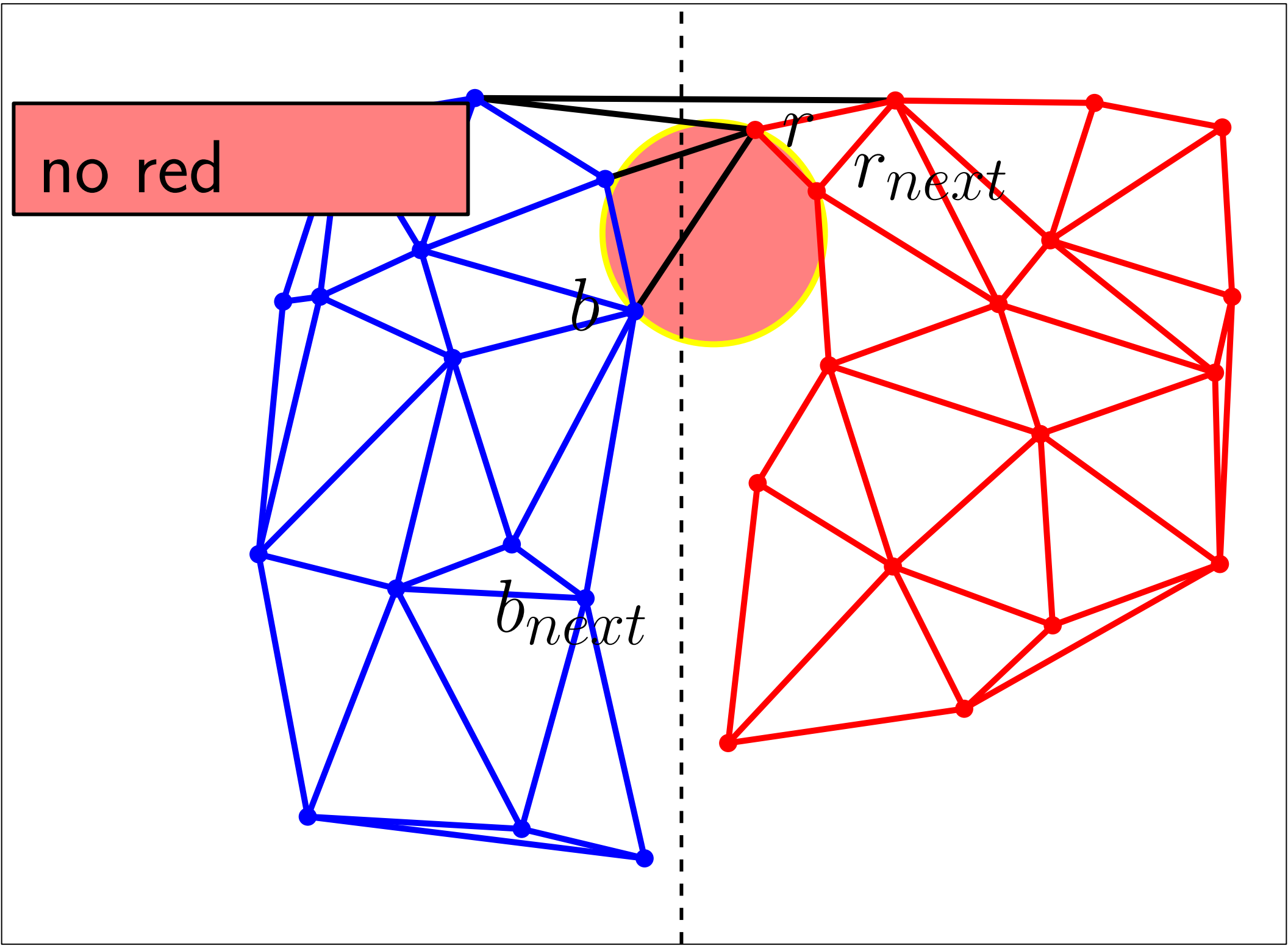

first red vertex crossed by set of circles
r_5
r
$r_4 = r_{next}$
b
r_3
r_2
r_1
$r_5 \notin \text{circle}(b, r, r_4)$
$\forall red, red \notin \text{circle}(b, r, r_4)$
(black disks are Delaunay for red set)

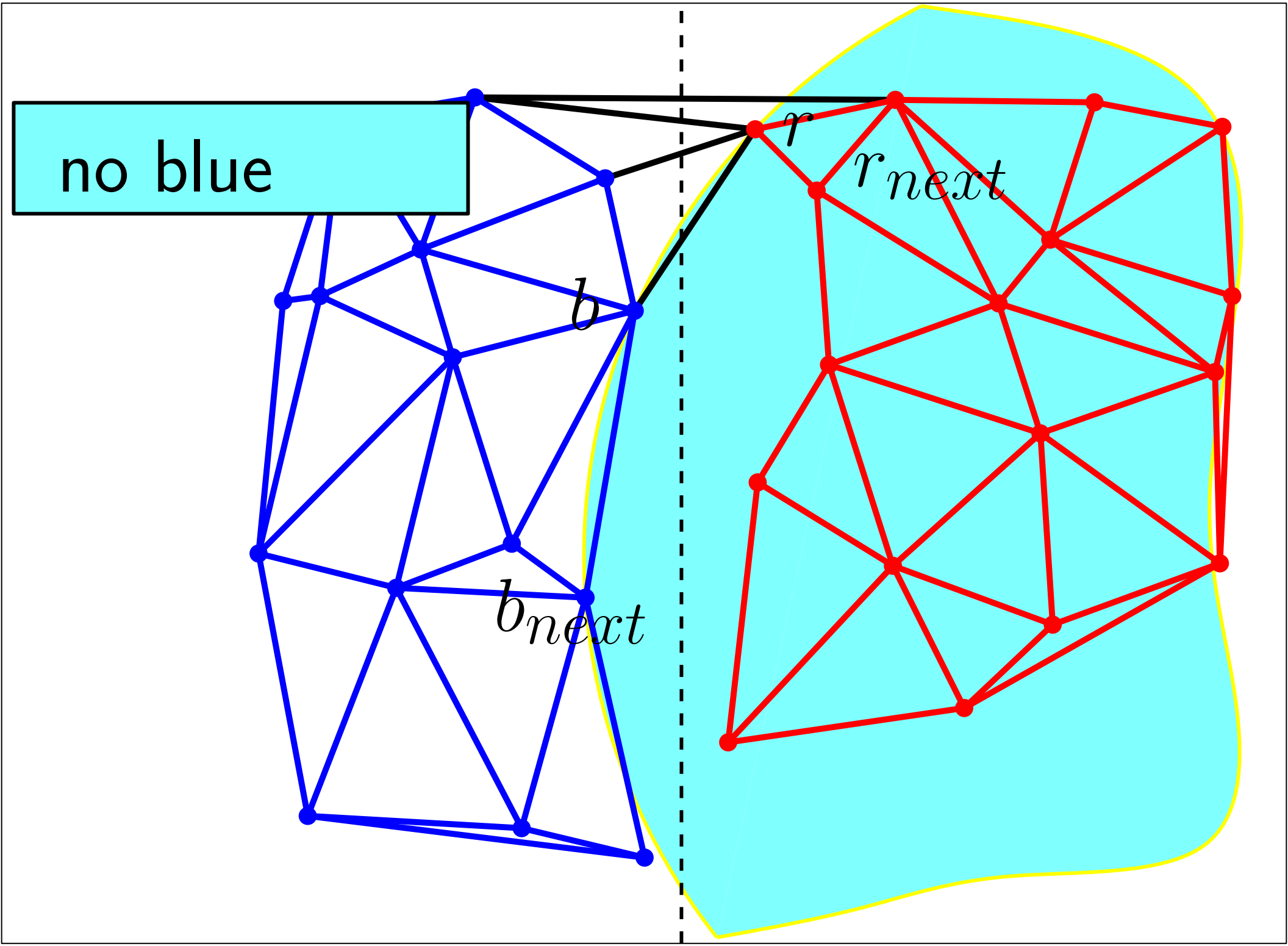

first blue vertex crossed by set of circles
r
r_{next}
b
b_2
$b_2 \in \text{circle}(b, r, b_1)$
b_1
Always look at first neighbor of b cw

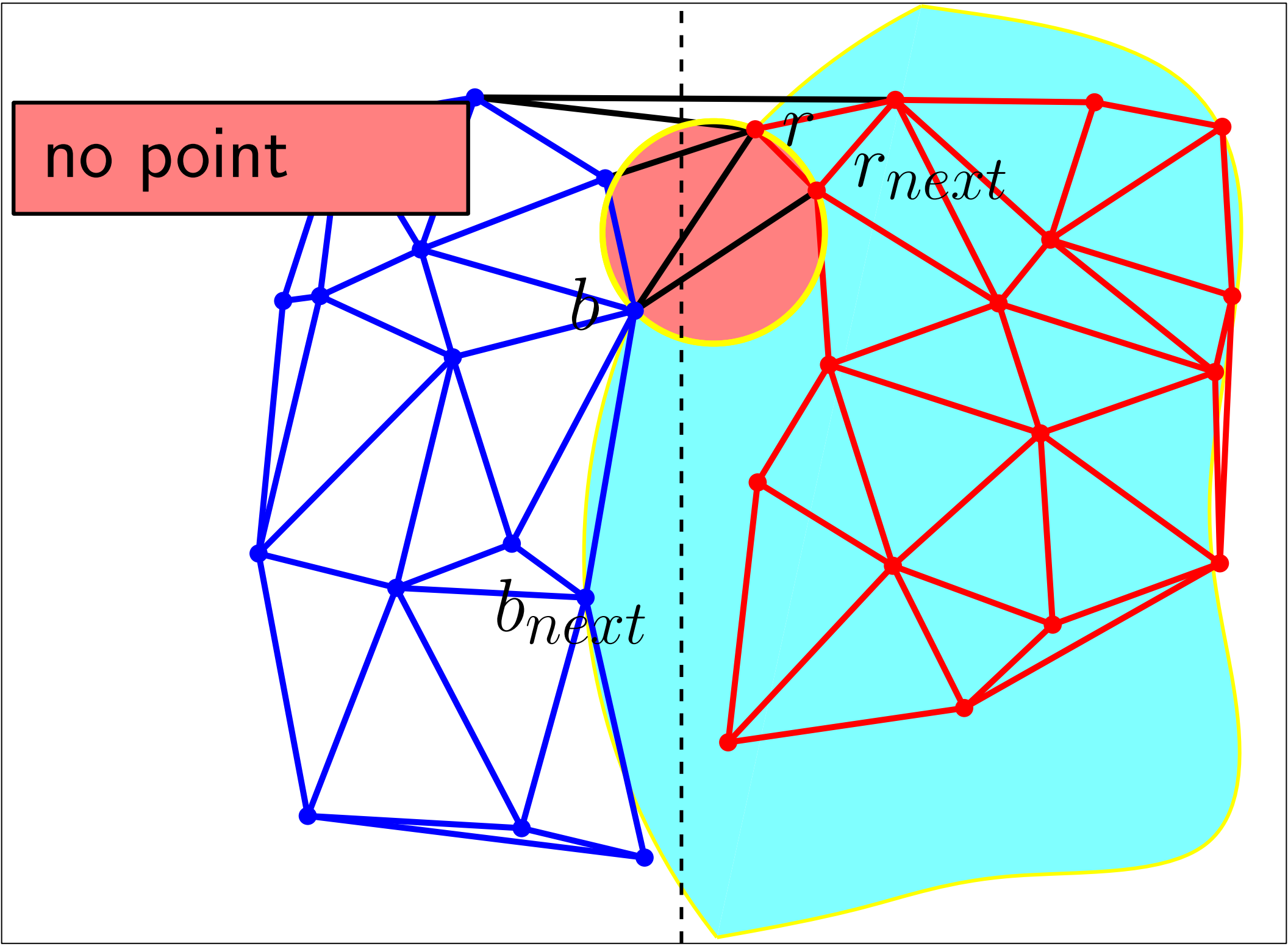

first blue vertex crossed by set of circles
r
r_{next}
b
b_3
b_2
b_1
$b_3 \notin \text{circle}(b, r, b_2)$

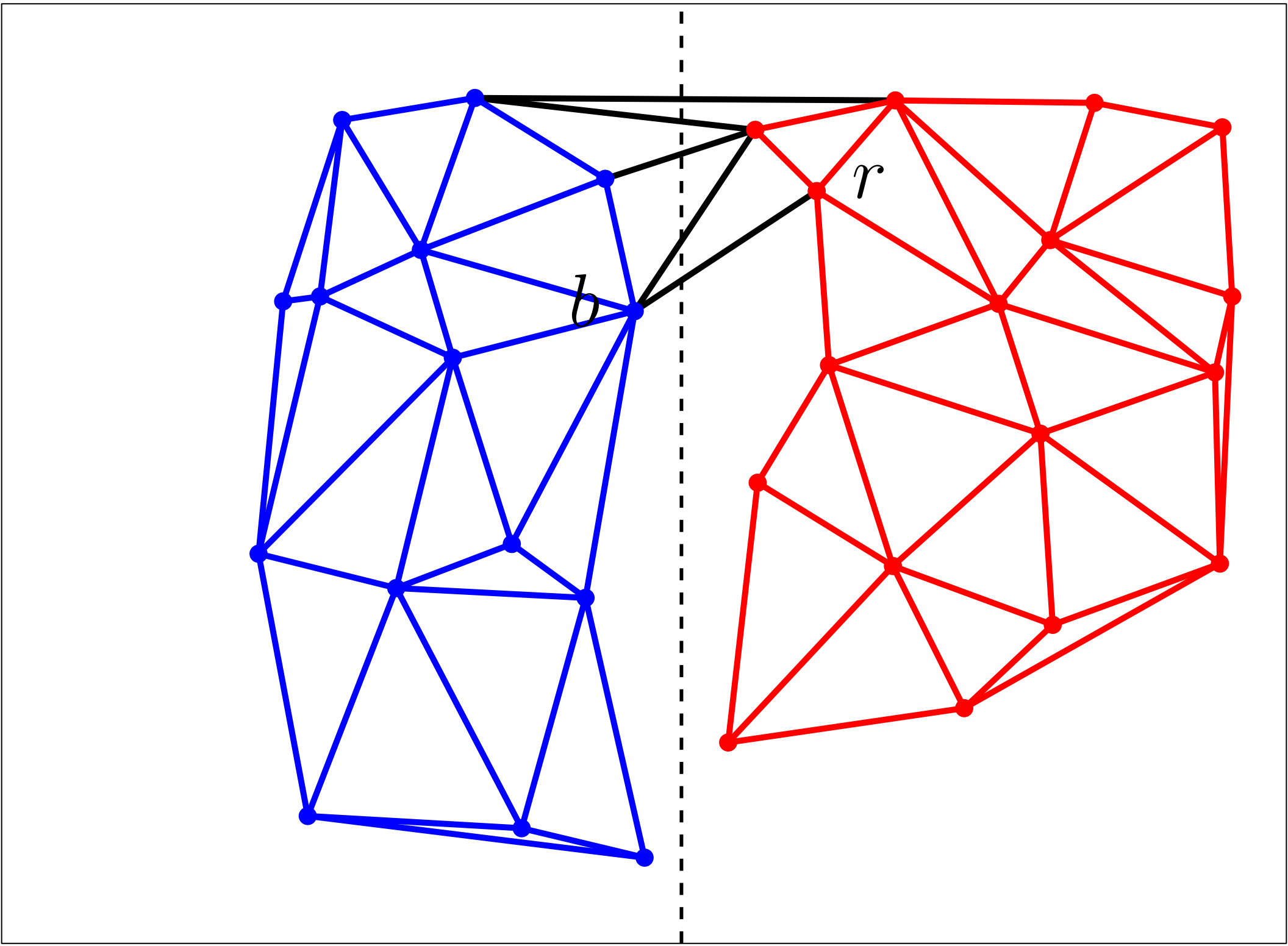

first blue vertex crossed by set of circles
r
r_{next}
b
b_3
$b_{next} = b_2$
b_1
$b_3 \notin \text{circle}(b, r, b_2)$
$\forall blue, blue \notin \text{circle}(b, r, b_2)$

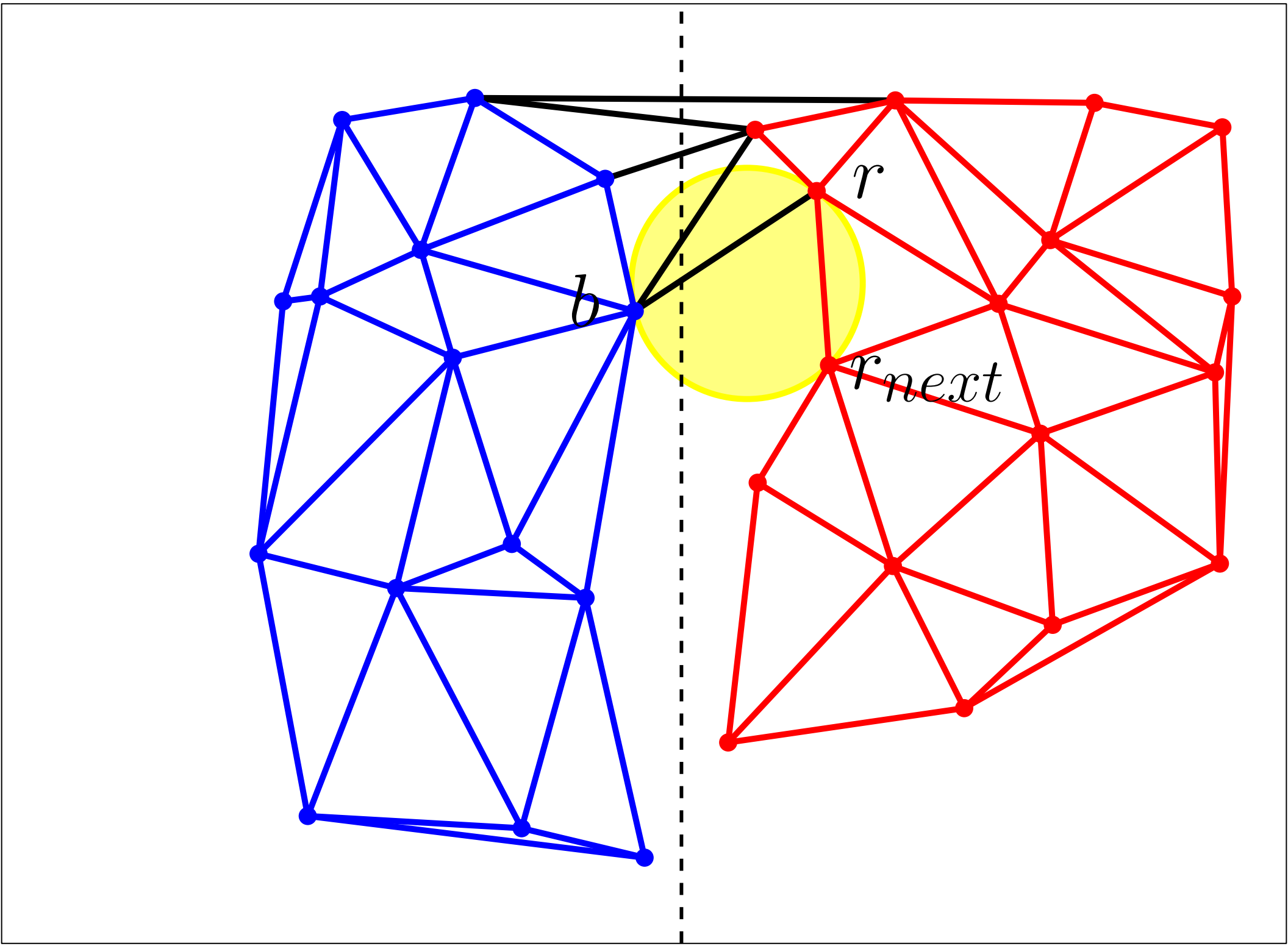

no point
r
r_{next}
b
b_{next}

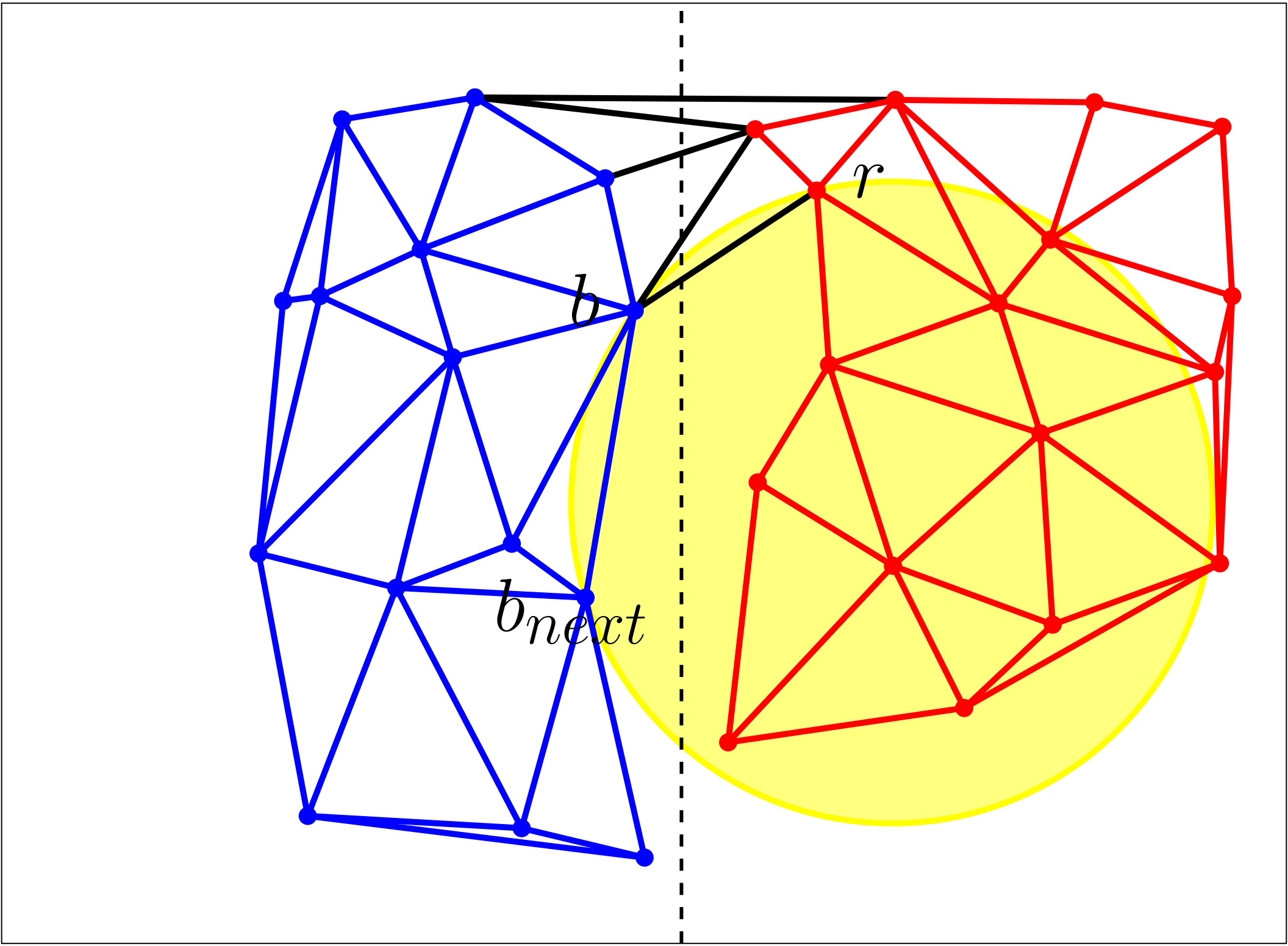

no red
r
r_{next}
b
b_{next}

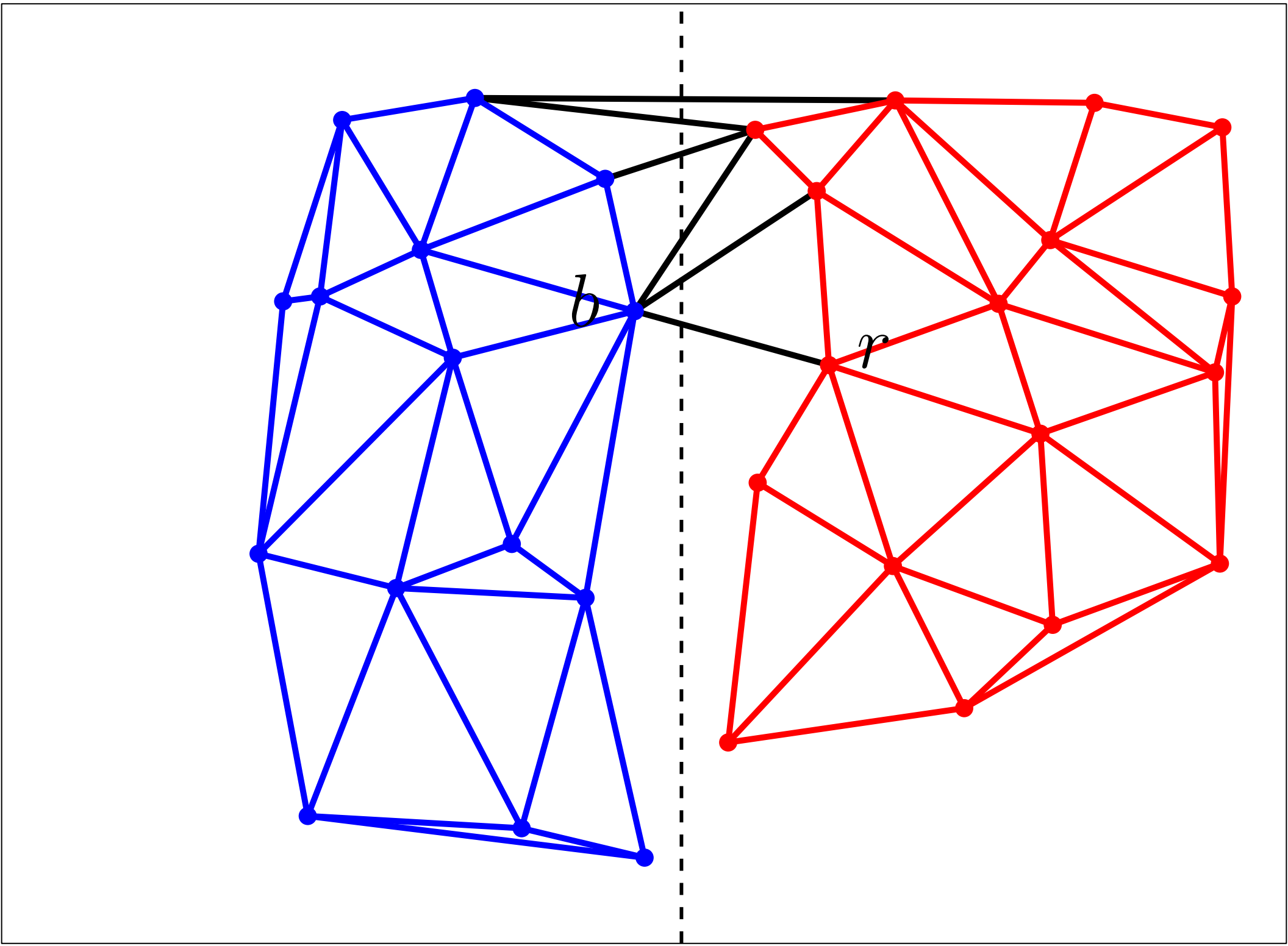

no blue
r
r_{next}
b
b_{next}

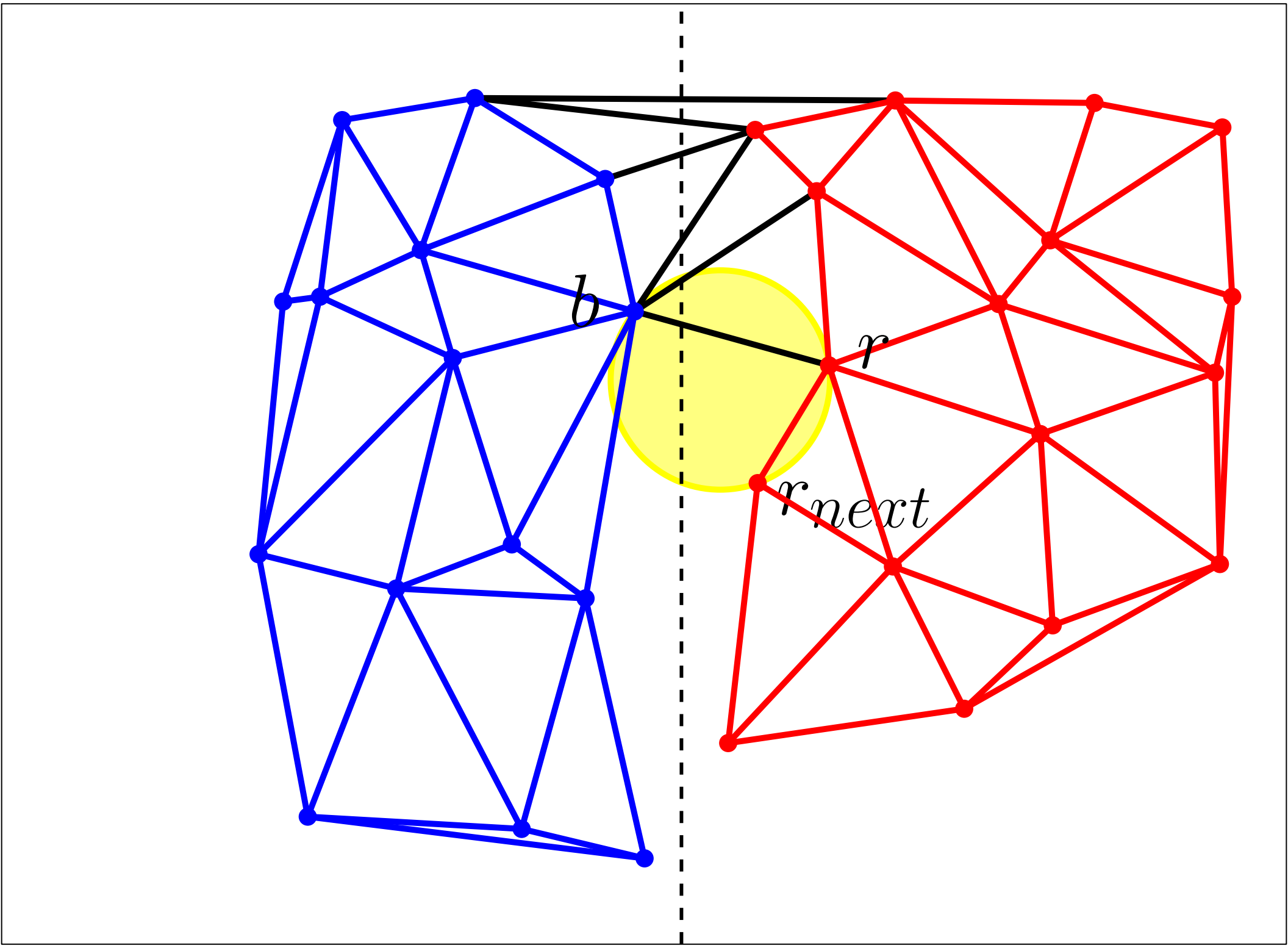

no point
r
r_{next}
b
b_{next}

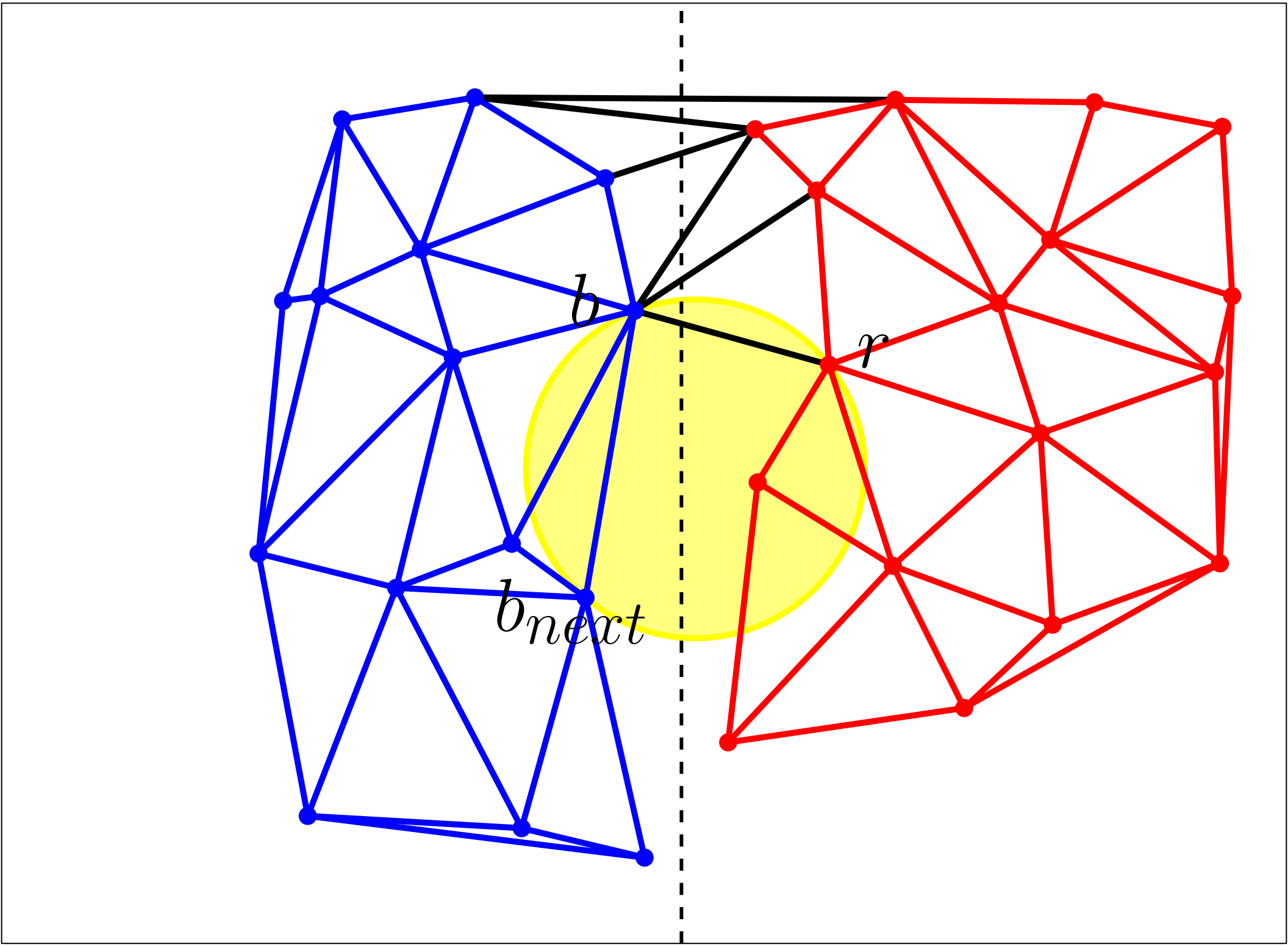

r
b

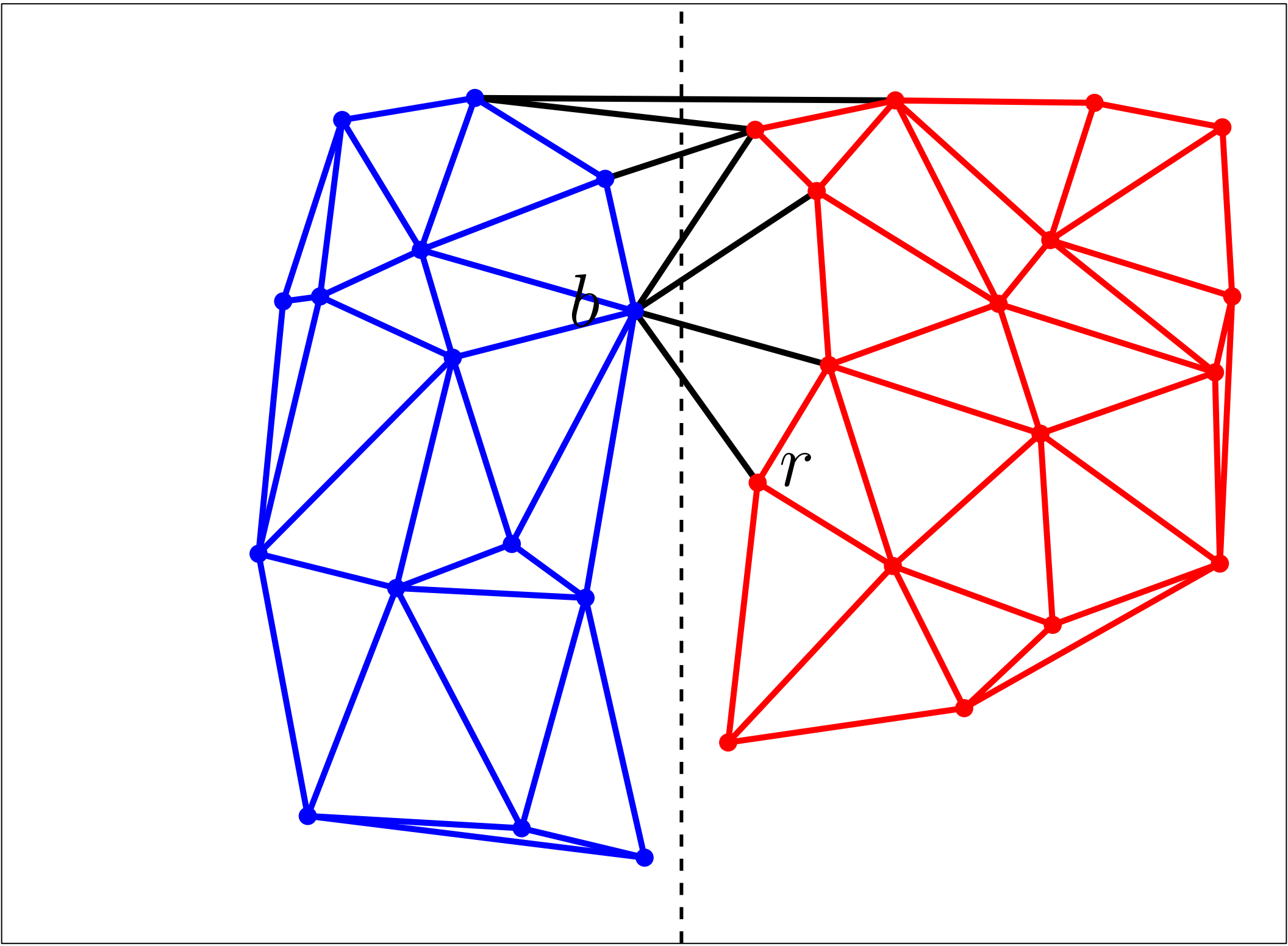

r
b
r_{next}

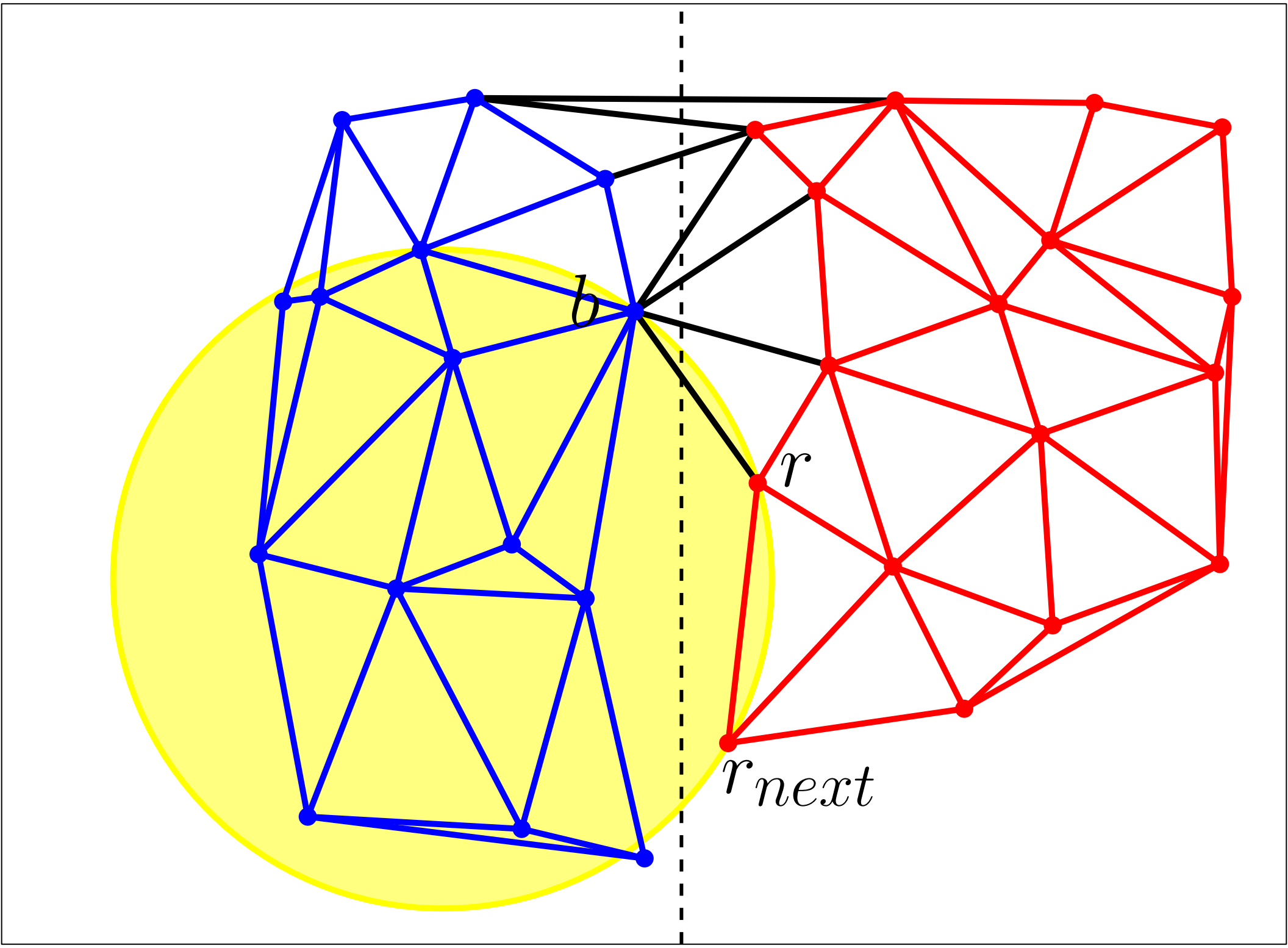

r
b
b_{next}

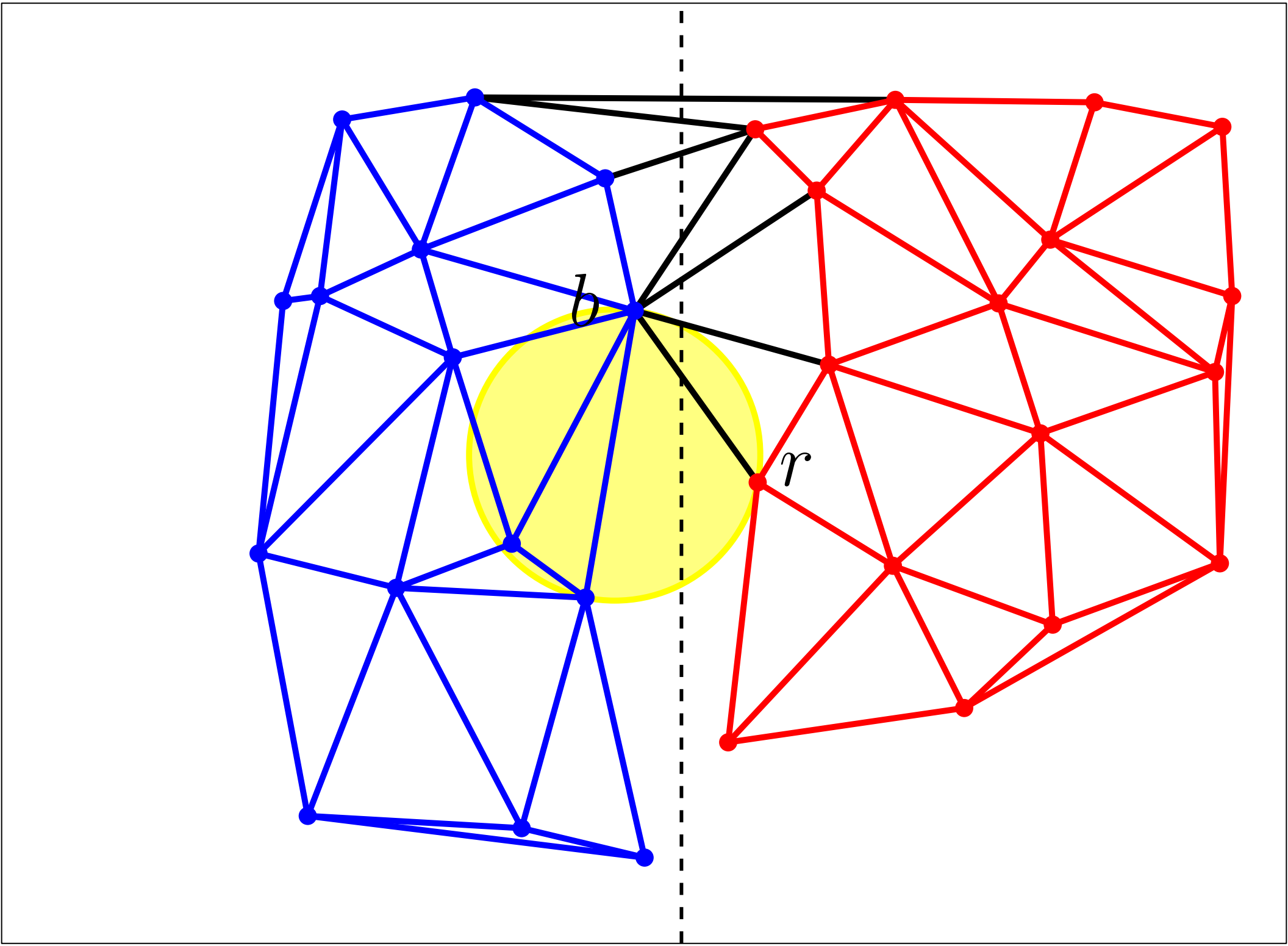

b
r

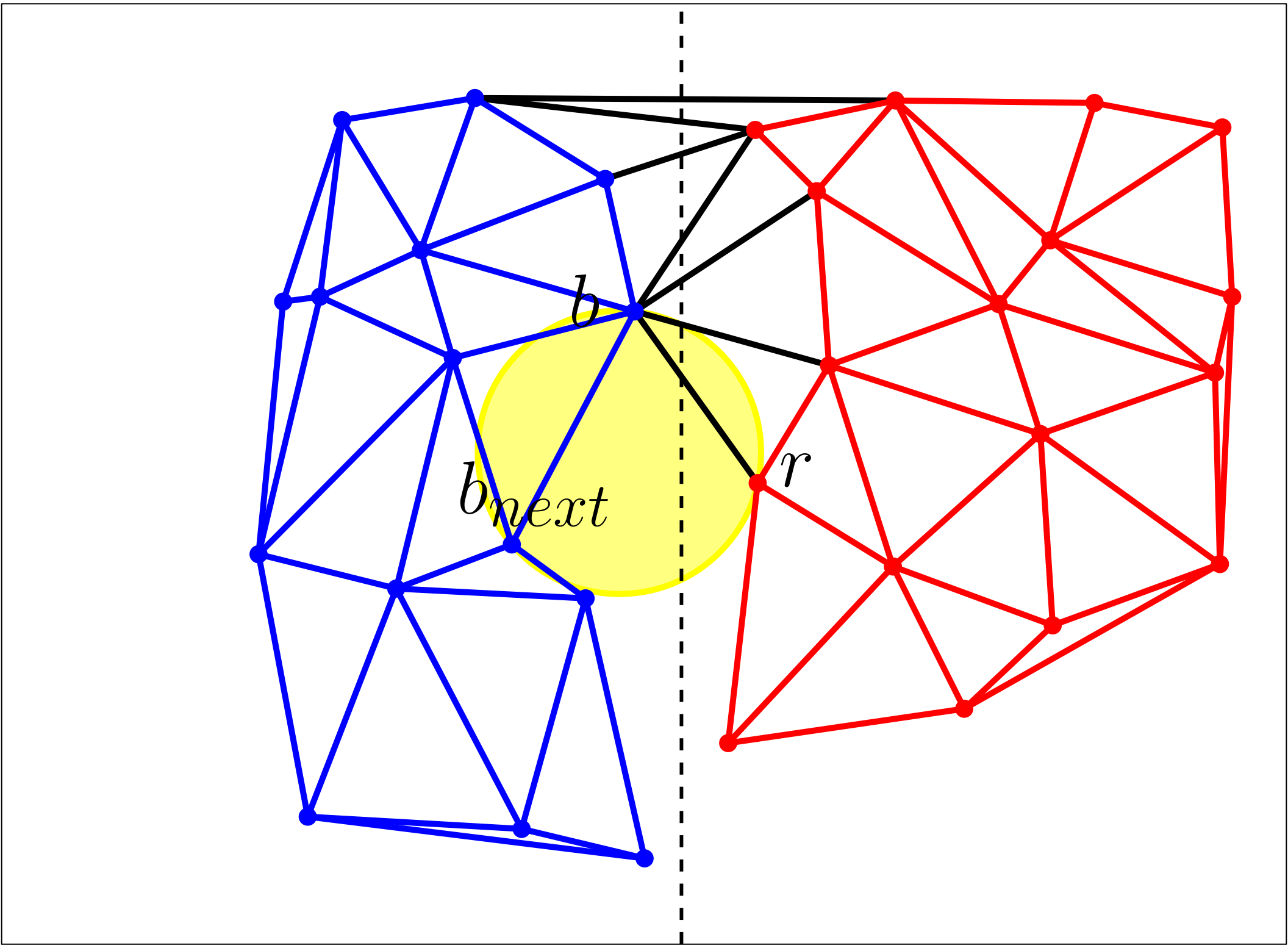

b
r
r_{next}

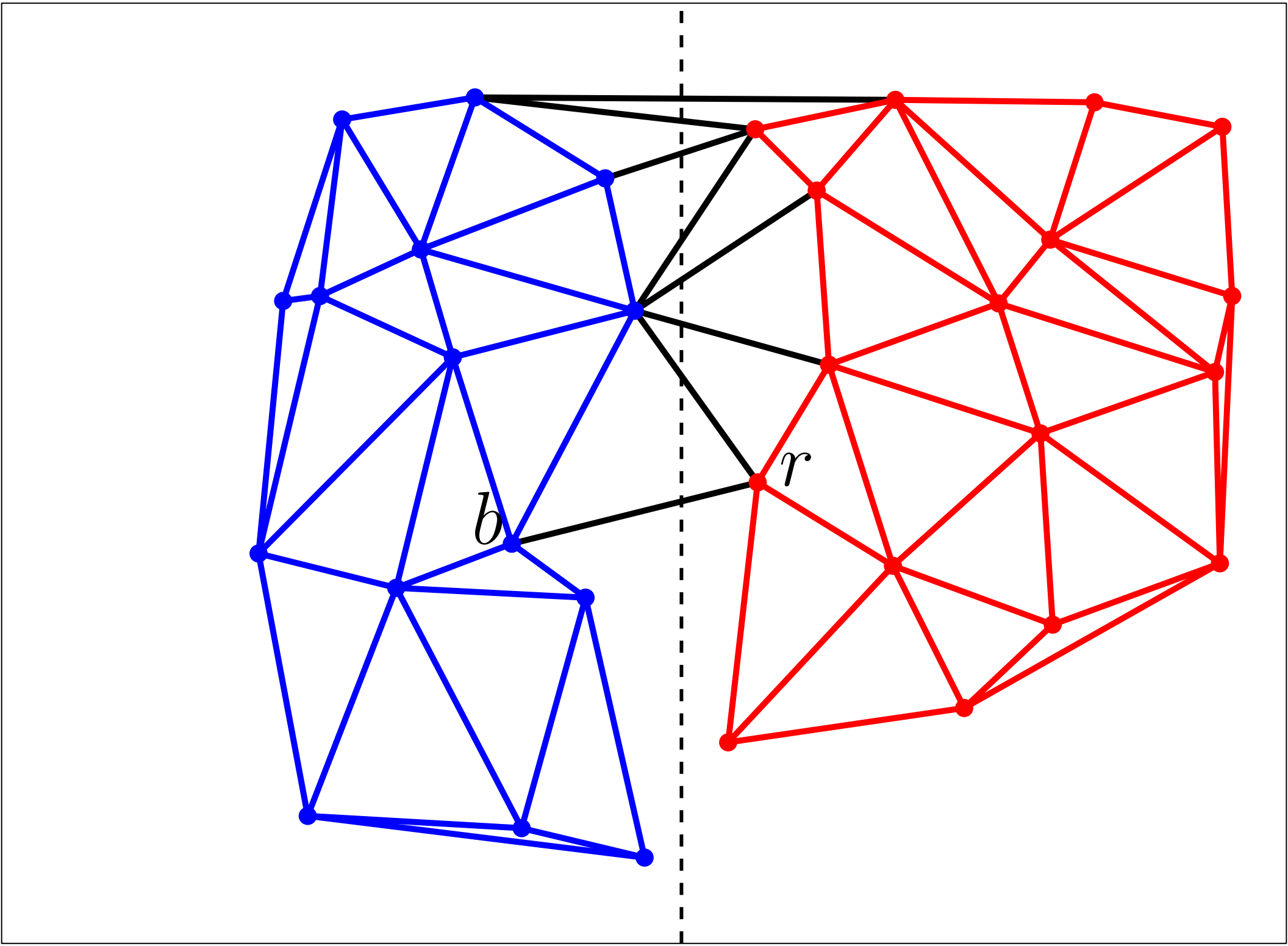

b
r
b_{next}

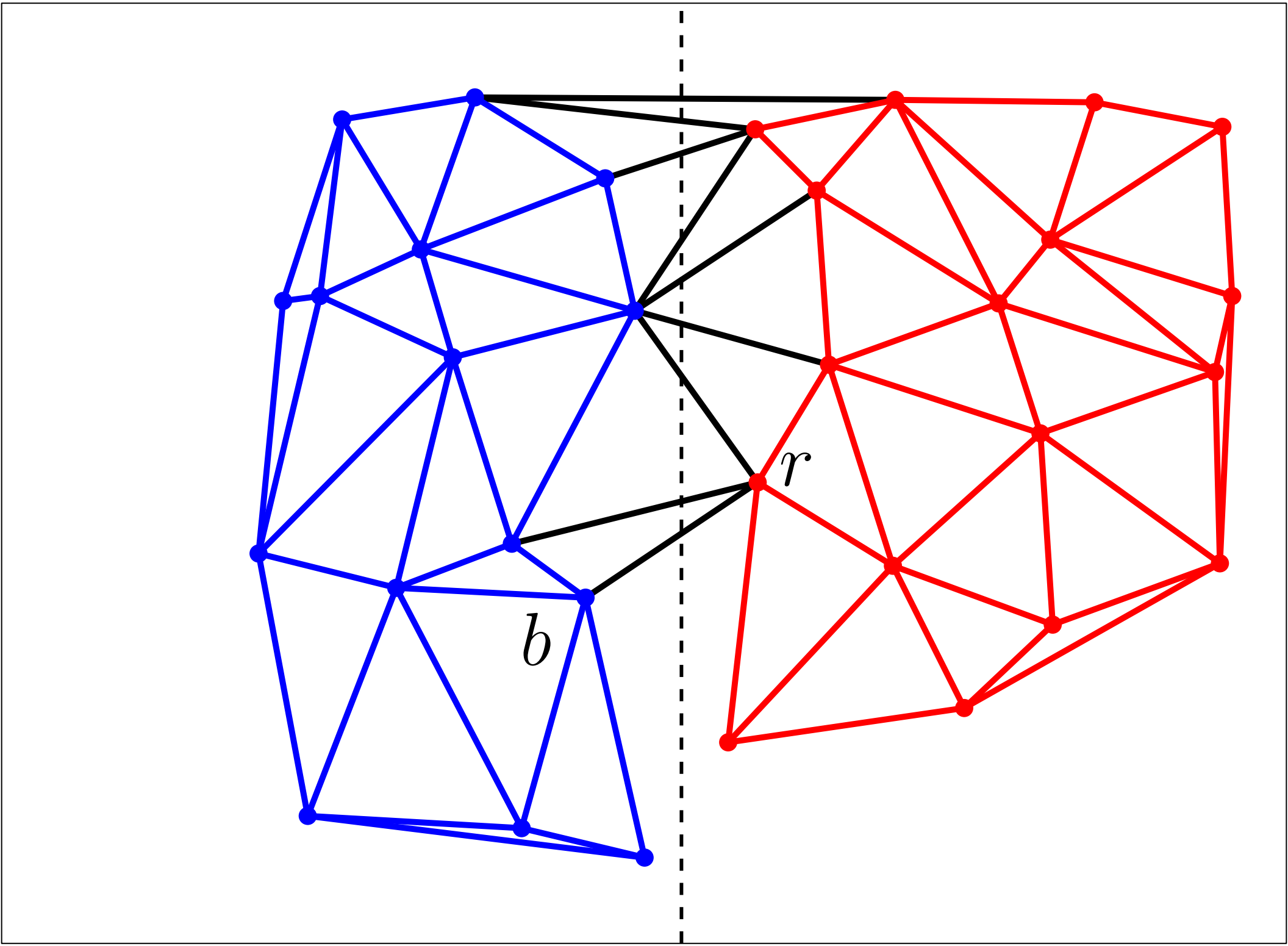

b
r

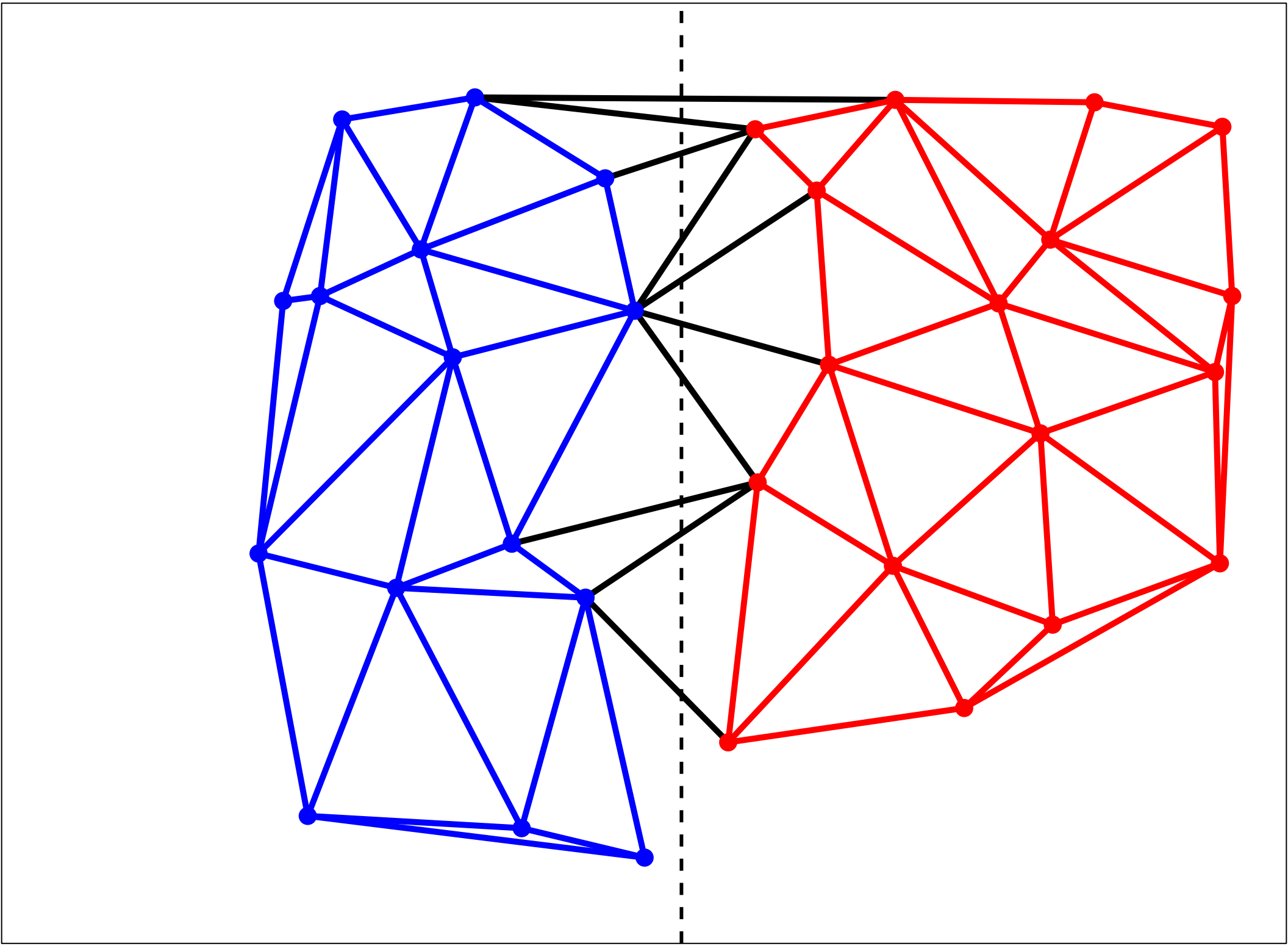

b
r
r_{next}

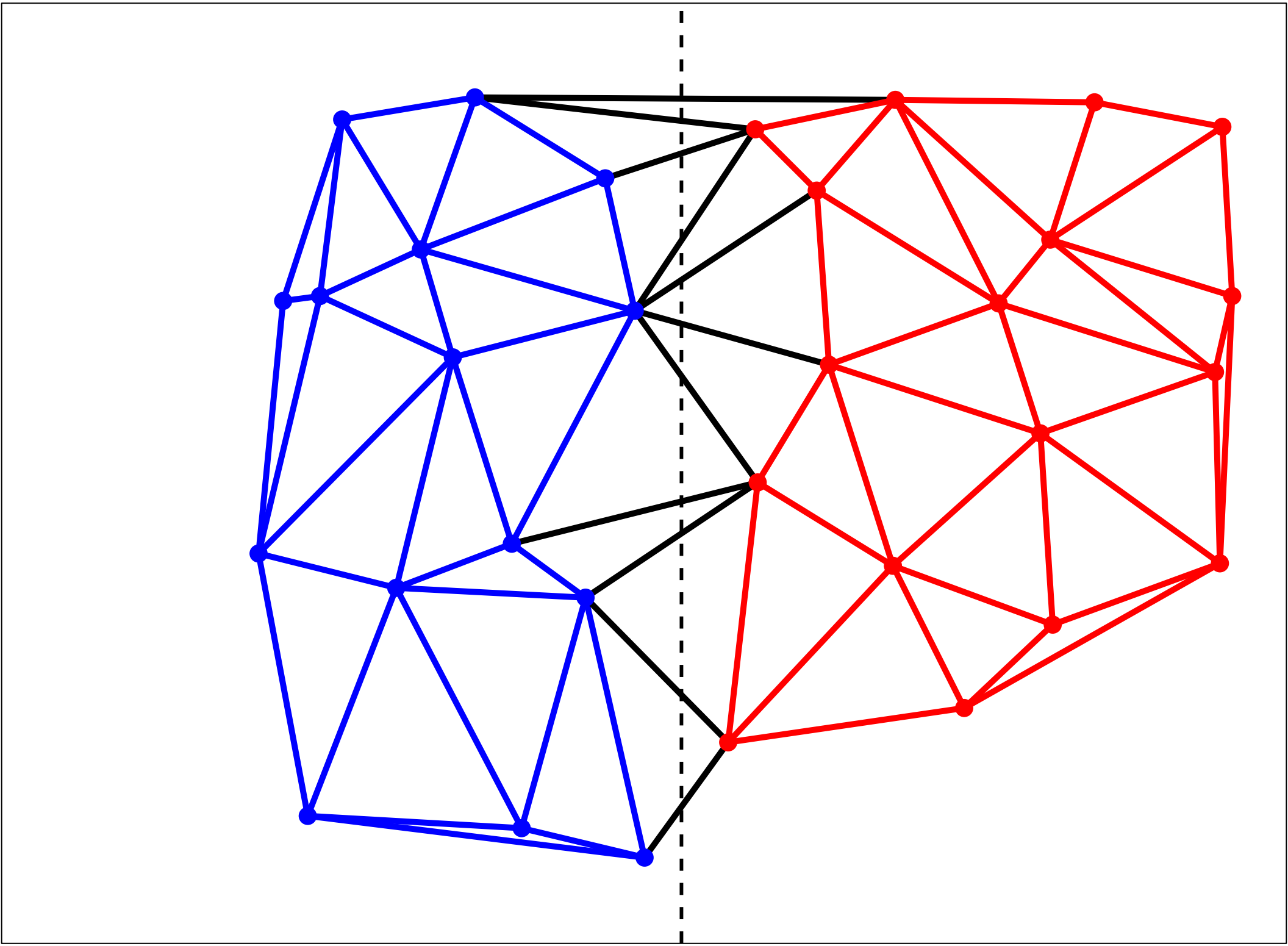

b
r

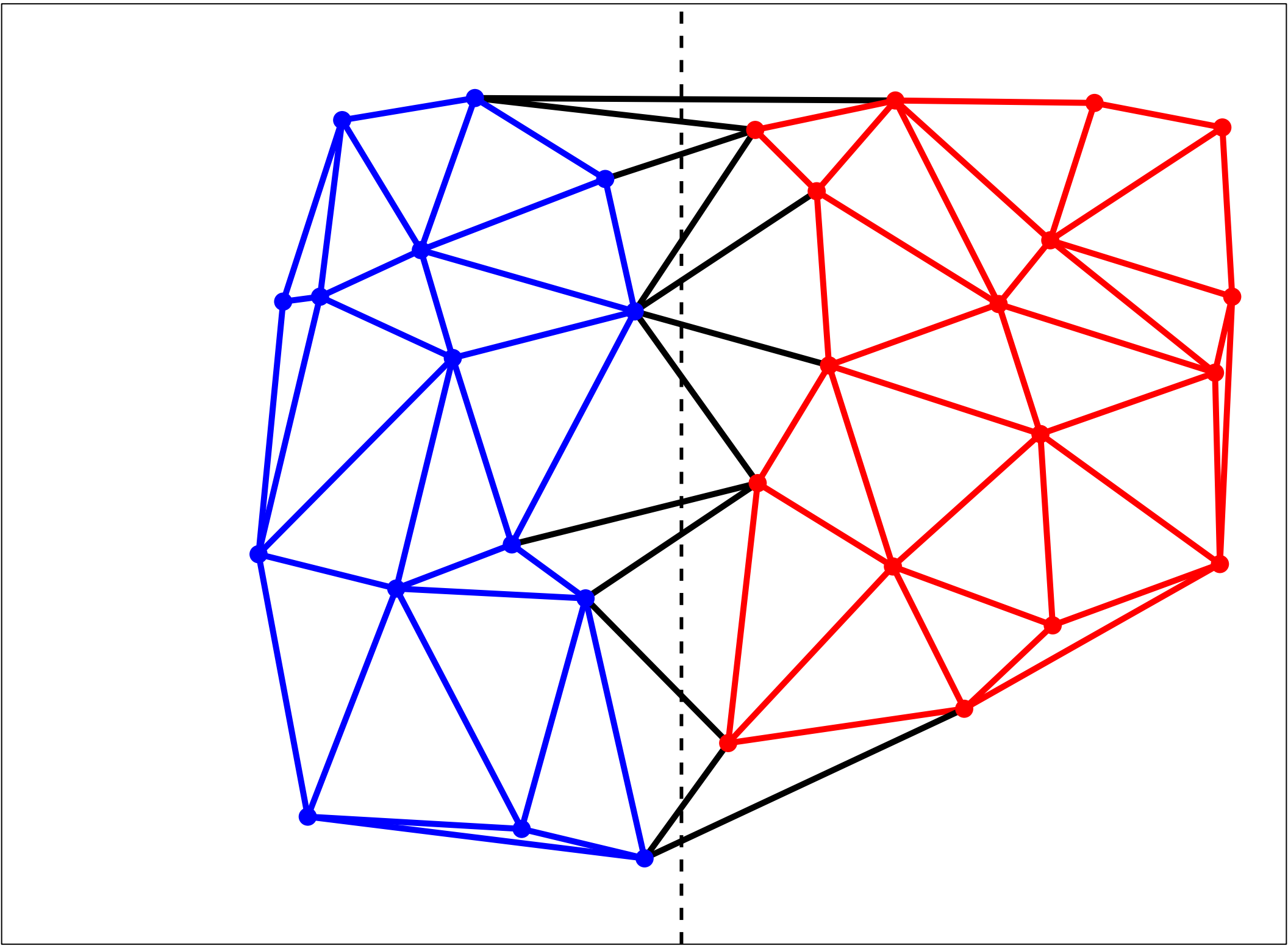

b
b_{next}
r


r
b


r
b







Complexity of Fusion

At each step of the search for r_{next}

A red edge is deleted

At each step of the search for b_{next}

A blue edge is deleted

After the choice between r_{next} and b_{next}

A black edge is created

Complexity of Fusion

Complexity $\leq$ $\sharp$ red edges
$+\sharp$ blue edges
$+\sharp$ black edges

Complexity of Fusion

Complexity $\leq$ $\sharp$ red edges
$+\sharp$ blue edges
$+\sharp$ black edges

$$\leq 3\frac{n}{2} + 3\frac{n}{2} + 3n = O(n)$$

each colored triangulation has $\leq 3k$ edges, where k is the size of the subset of vertices

the black edges are Delaunay $\Rightarrow$ there are at most $3n$ of them

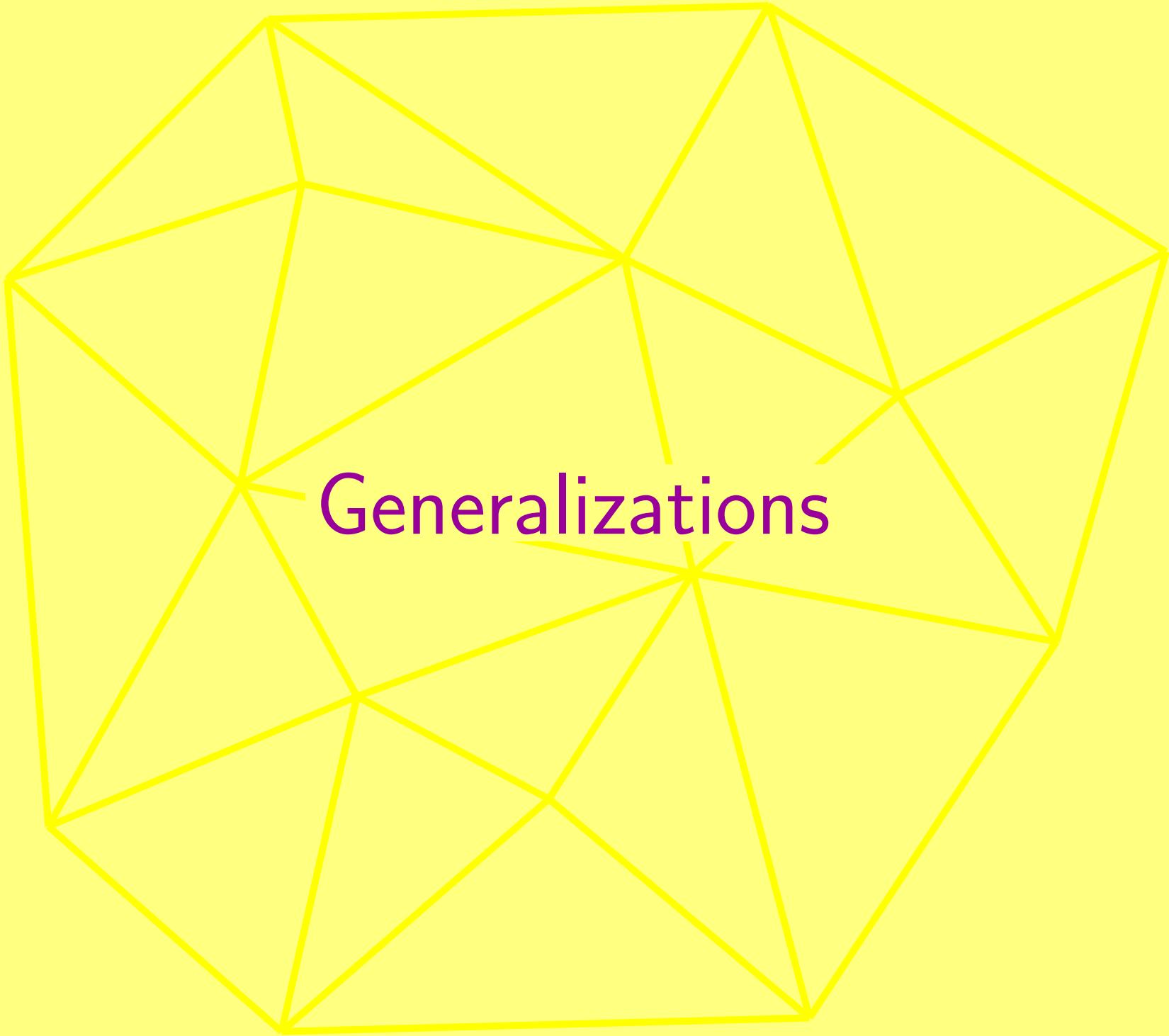
Overall Complexity

Division $= O(k)$ on sub-problem of size k

$$+ O(n \log n) \text{ preprocessing}$$

Fusion $= O(k)$ on sub-problem of size k

Division-Fusion $\Longrightarrow O(n \log n)$


Generalizations

Voronoi diagram

Q Nearest neighbor of q among $\mathcal{S}$

Voronoi diagram

Q Nearest neighbor of q among $\mathcal{S}$

Change

ambient space (for q)

$\mathbb{R}^2$ $\mathbb{R}^3$ $\mathbb{R}^d$

Voronoi diagram

Q Nearest neighbor of q among $\mathcal{S}$

Change

metrics

Euclidean L_2

L_1, L_∞, L_p

hyperbolic

additive weights

multiplicative weights

Voronoi diagram

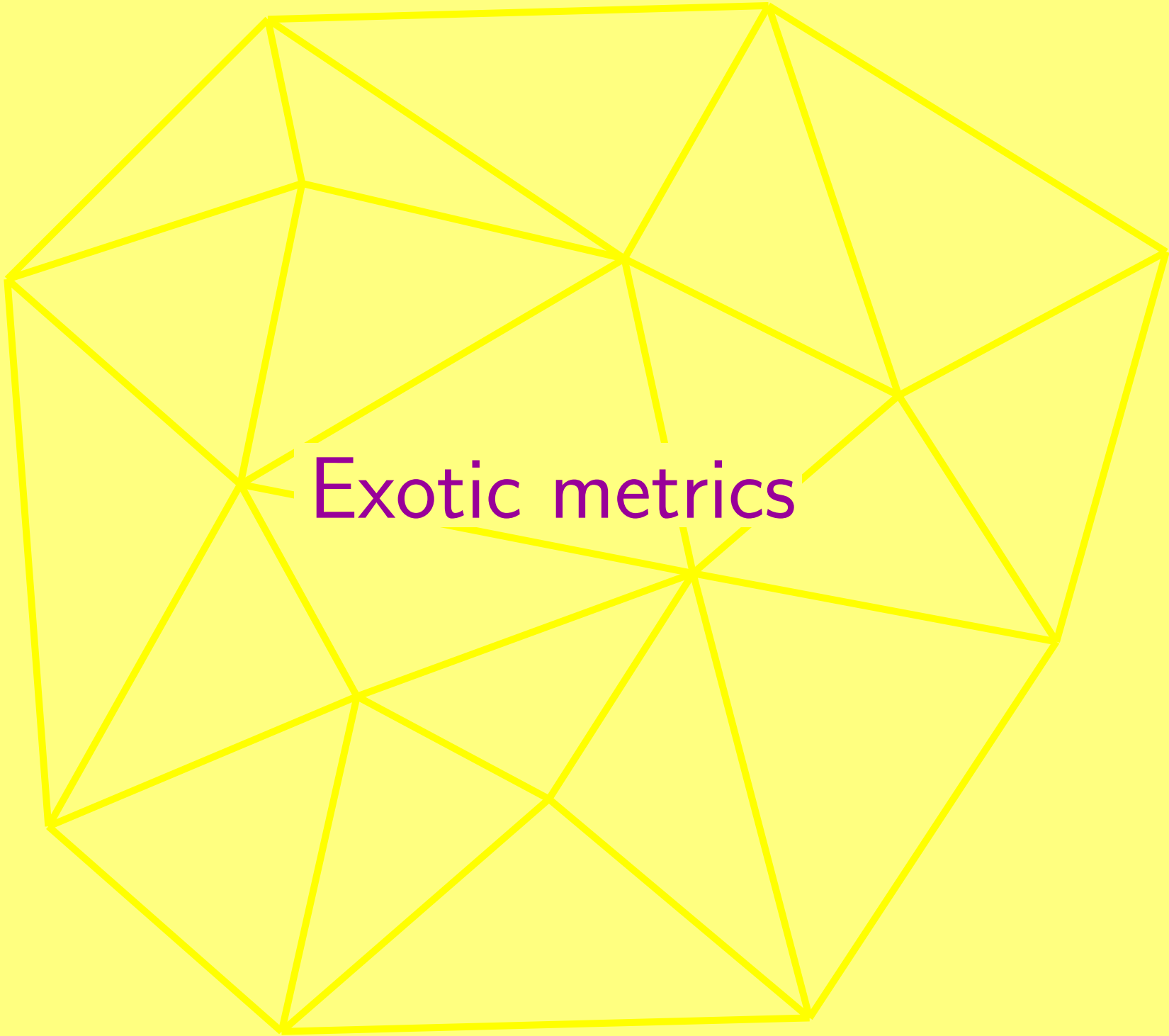
Q Nearest neighbor of q among $\mathcal{S}$

Change

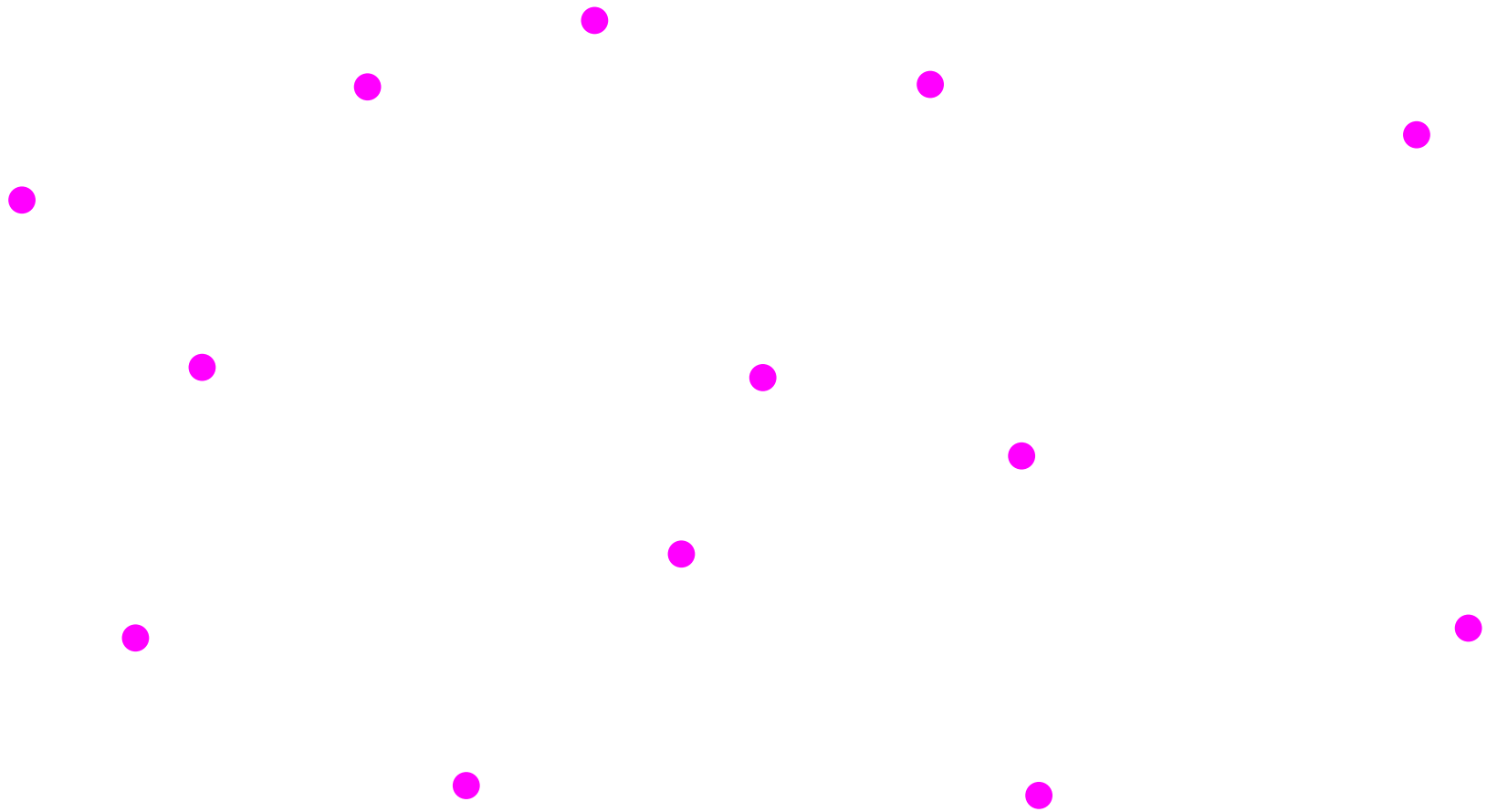
universal set $\supset \mathcal{S}$

points of $\mathbb{R}^d$ segments of $\mathbb{R}^d$

spheres of $\mathbb{R}^d$

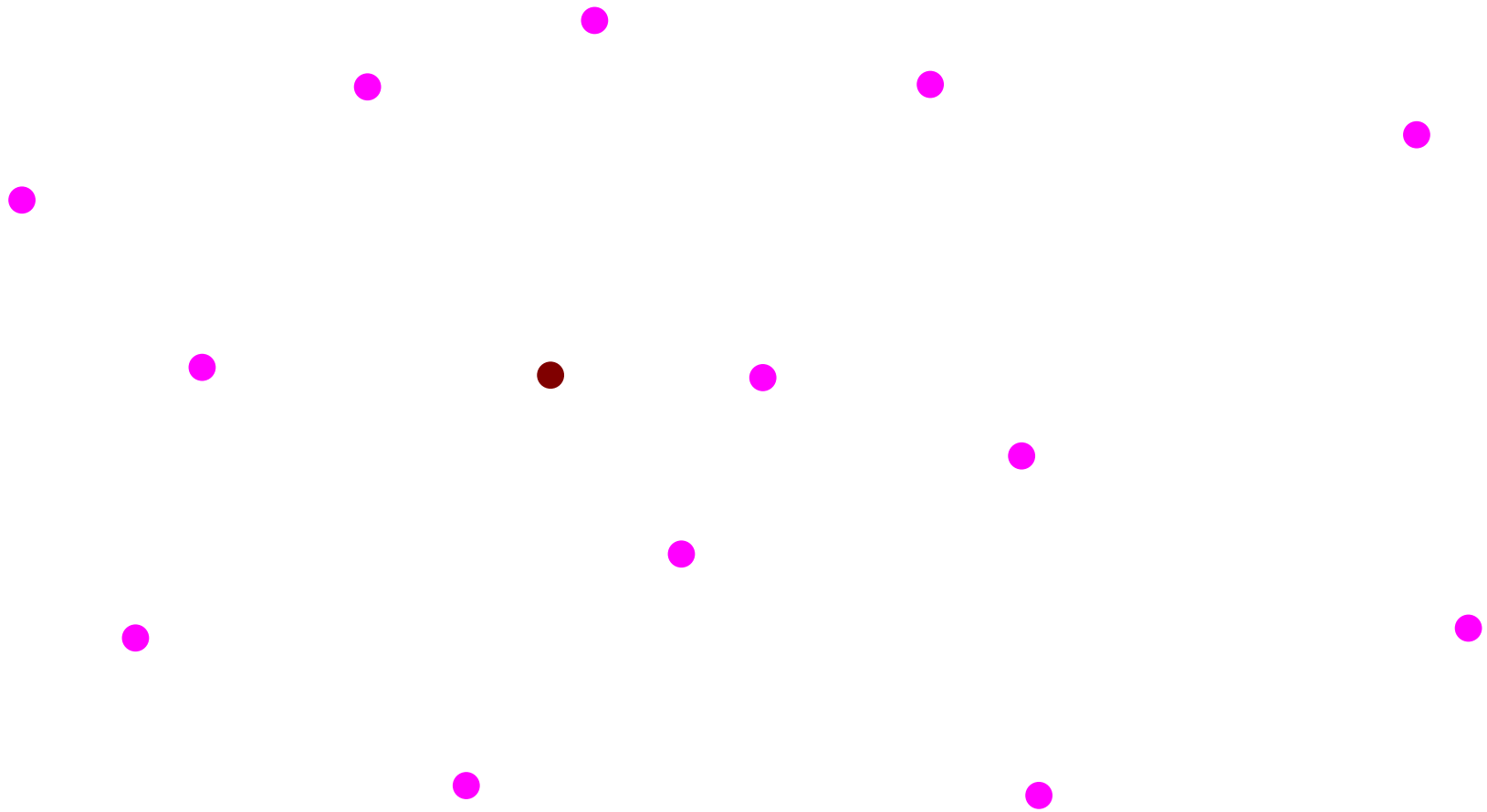

Exotic metrics

Norm L_∞: max($|x|, |y|$)



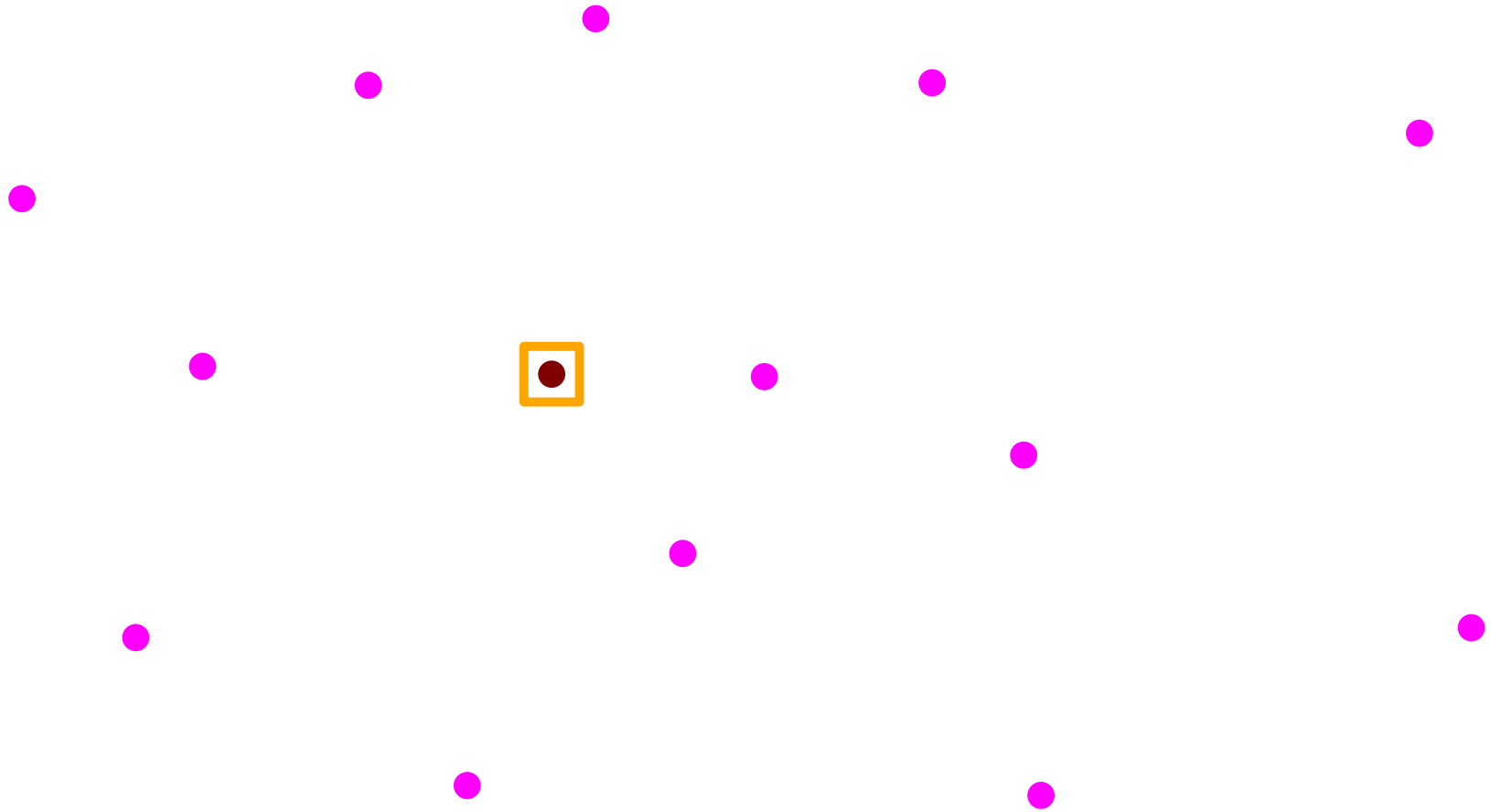
Norm L_∞: max($|x|, |y|$)

query



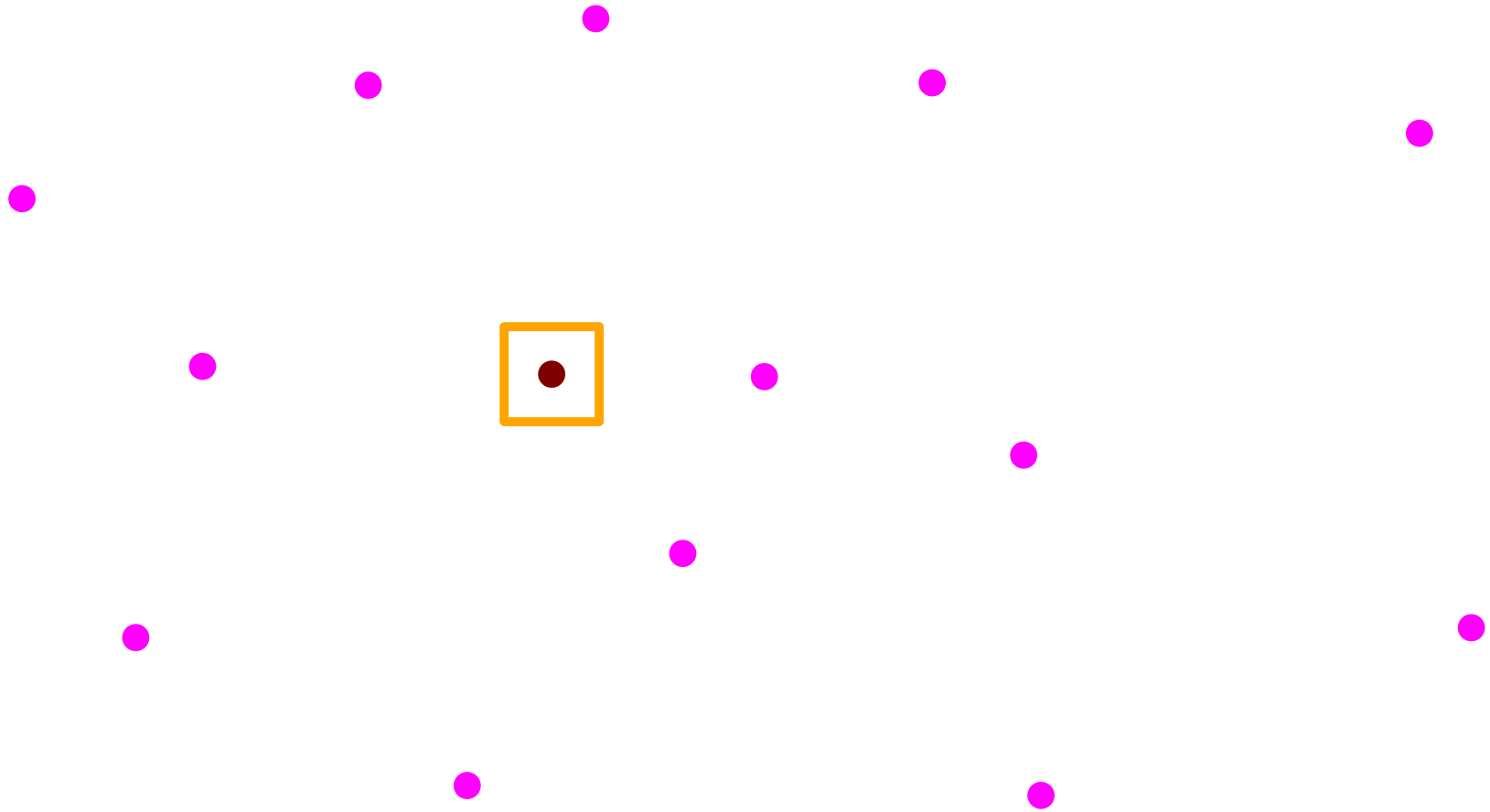
Norm L_∞: max($|x|, |y|$)

query



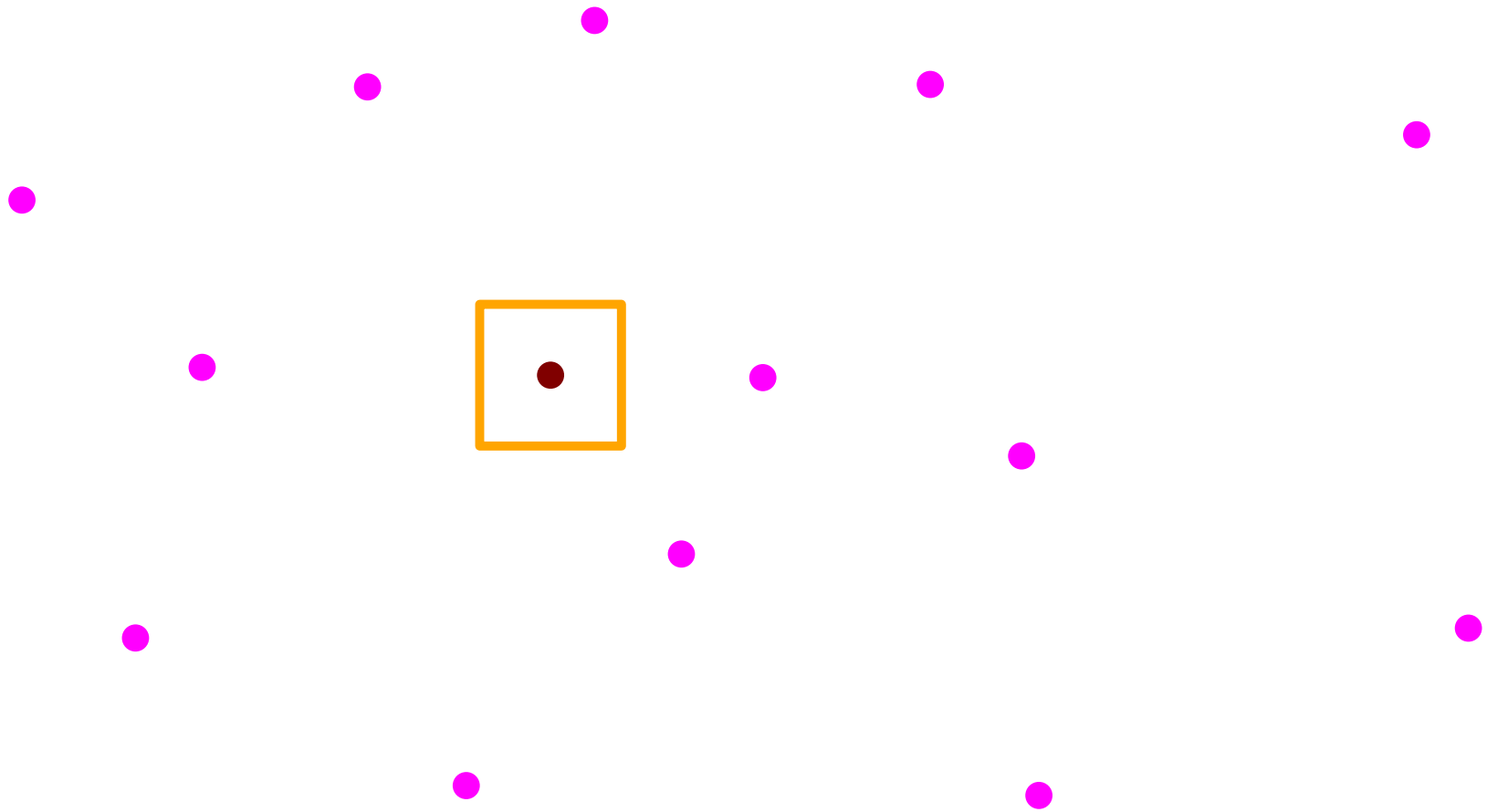
Norm L_∞: max($|x|, |y|$)

query



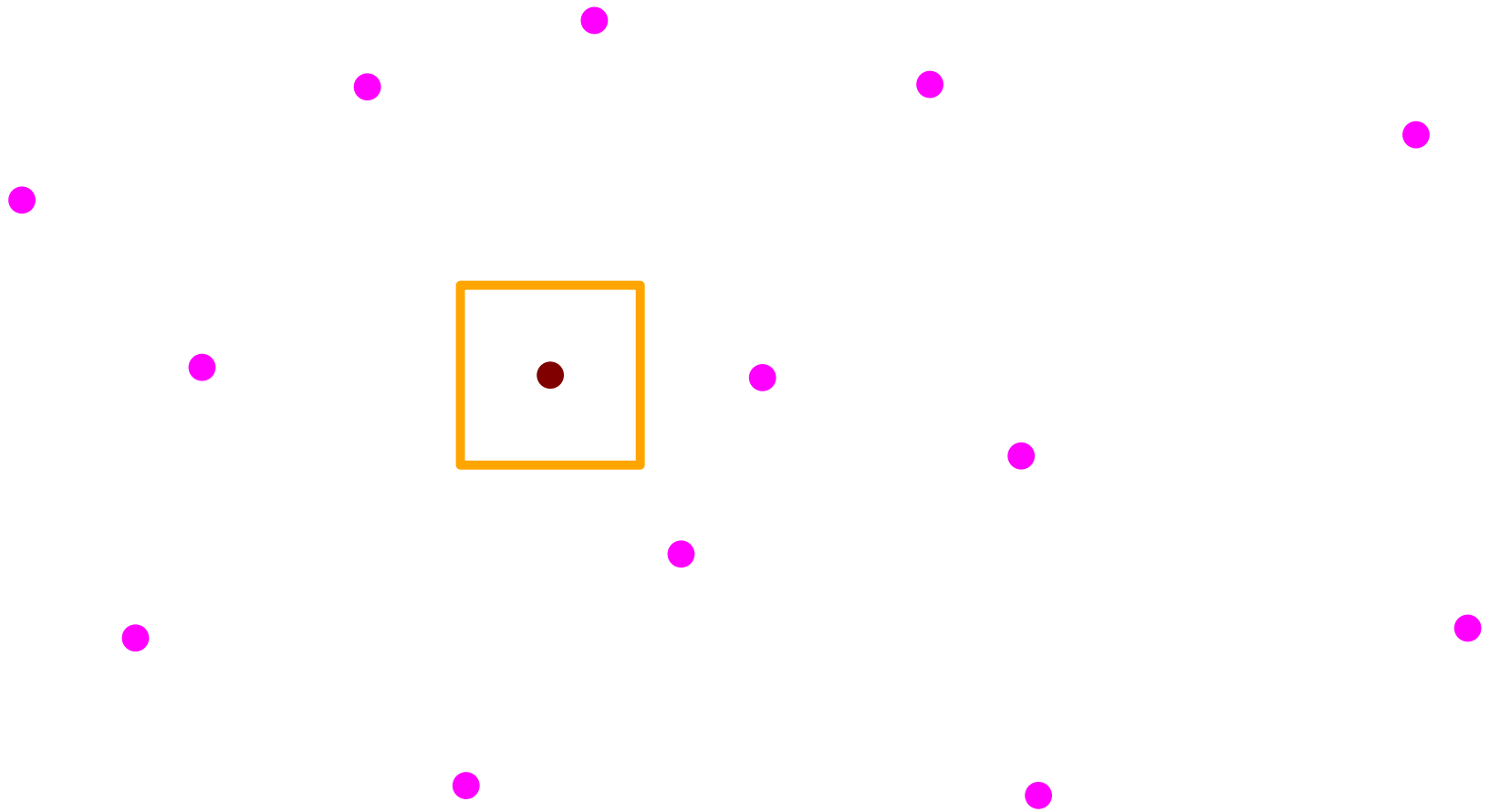
Norm L_∞: max($|x|, |y|$)

query



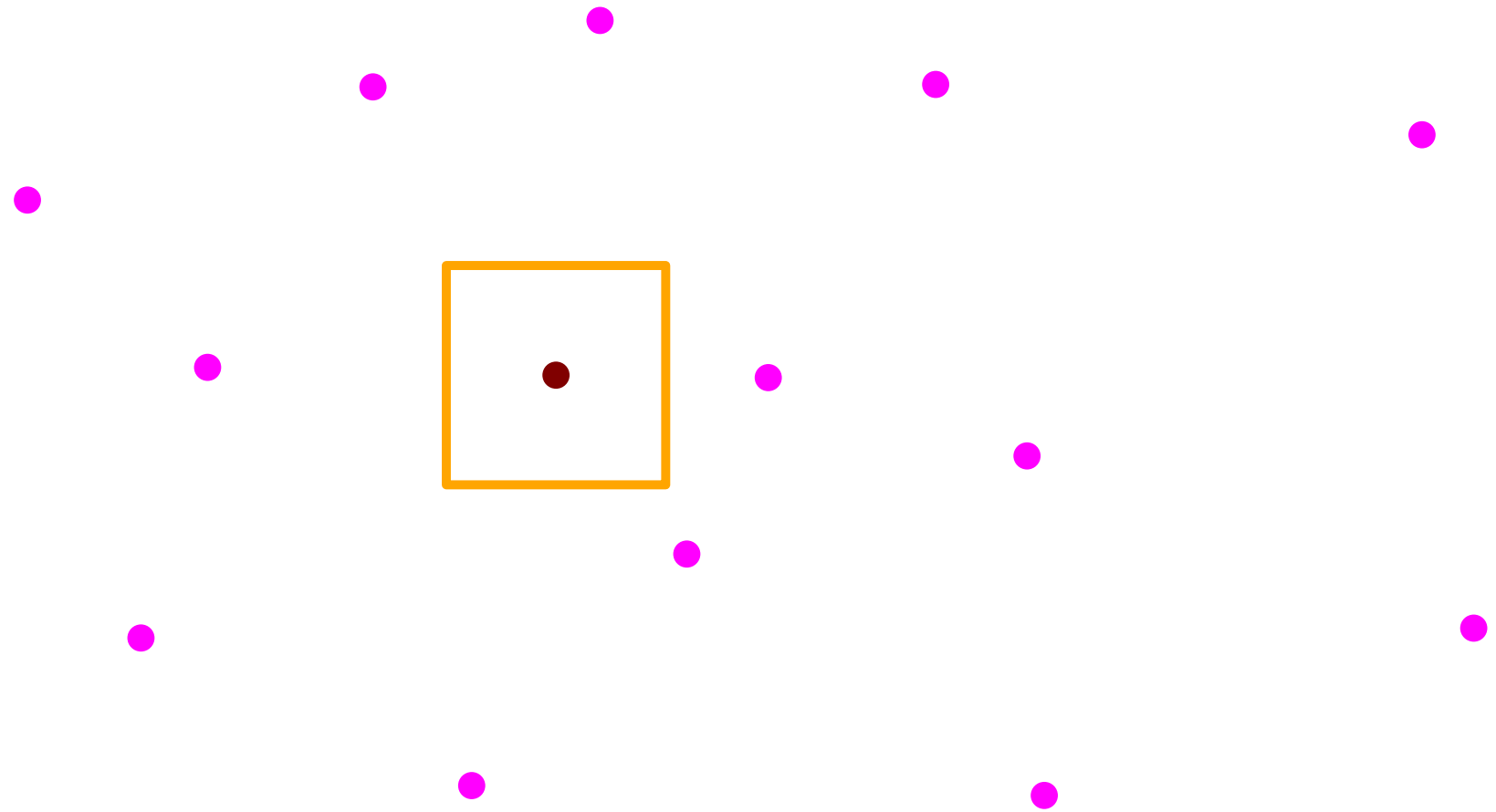
Norm L_∞: max($|x|, |y|$)

query



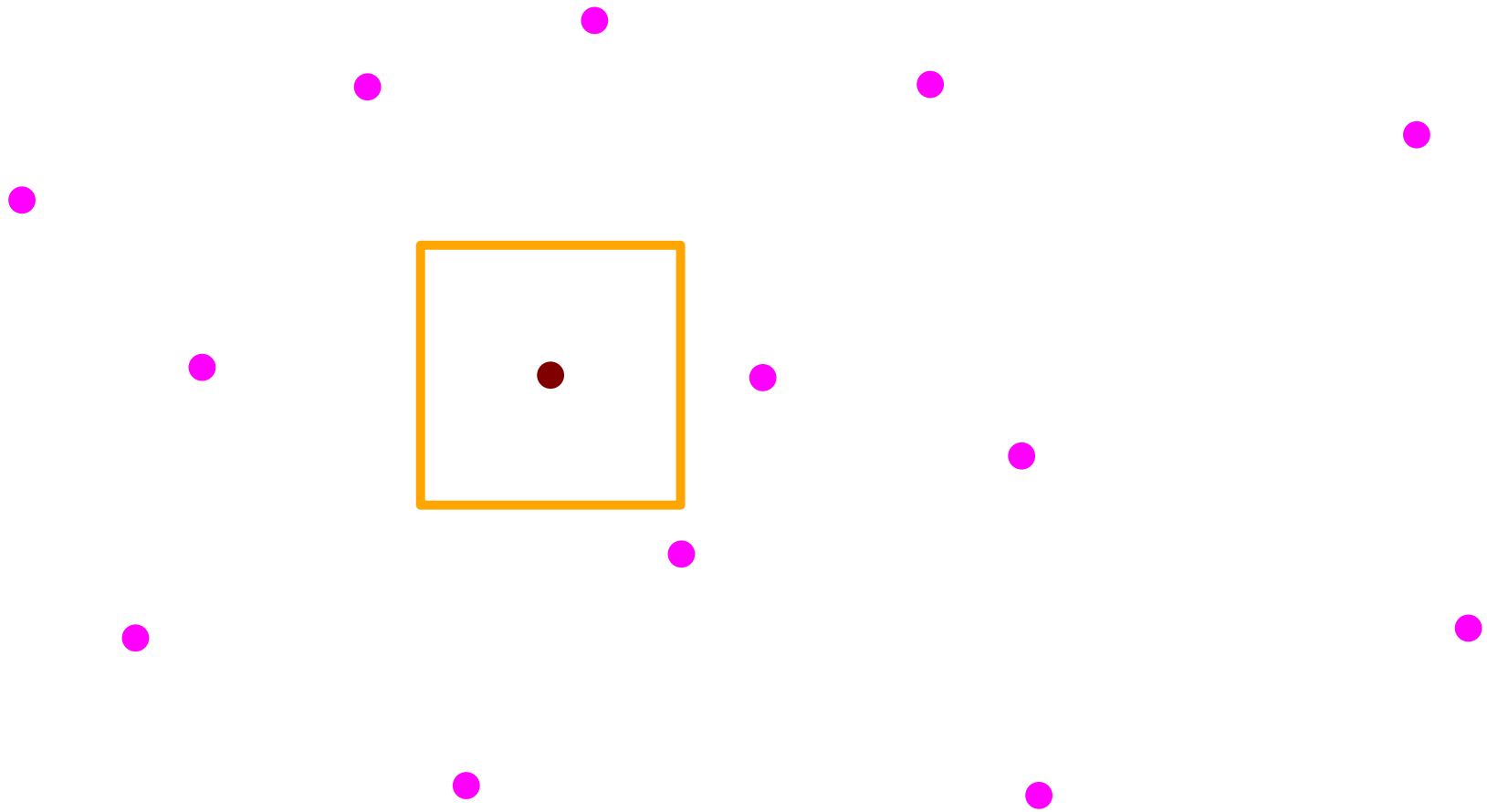
Norm L_∞: max($|x|, |y|$)

query



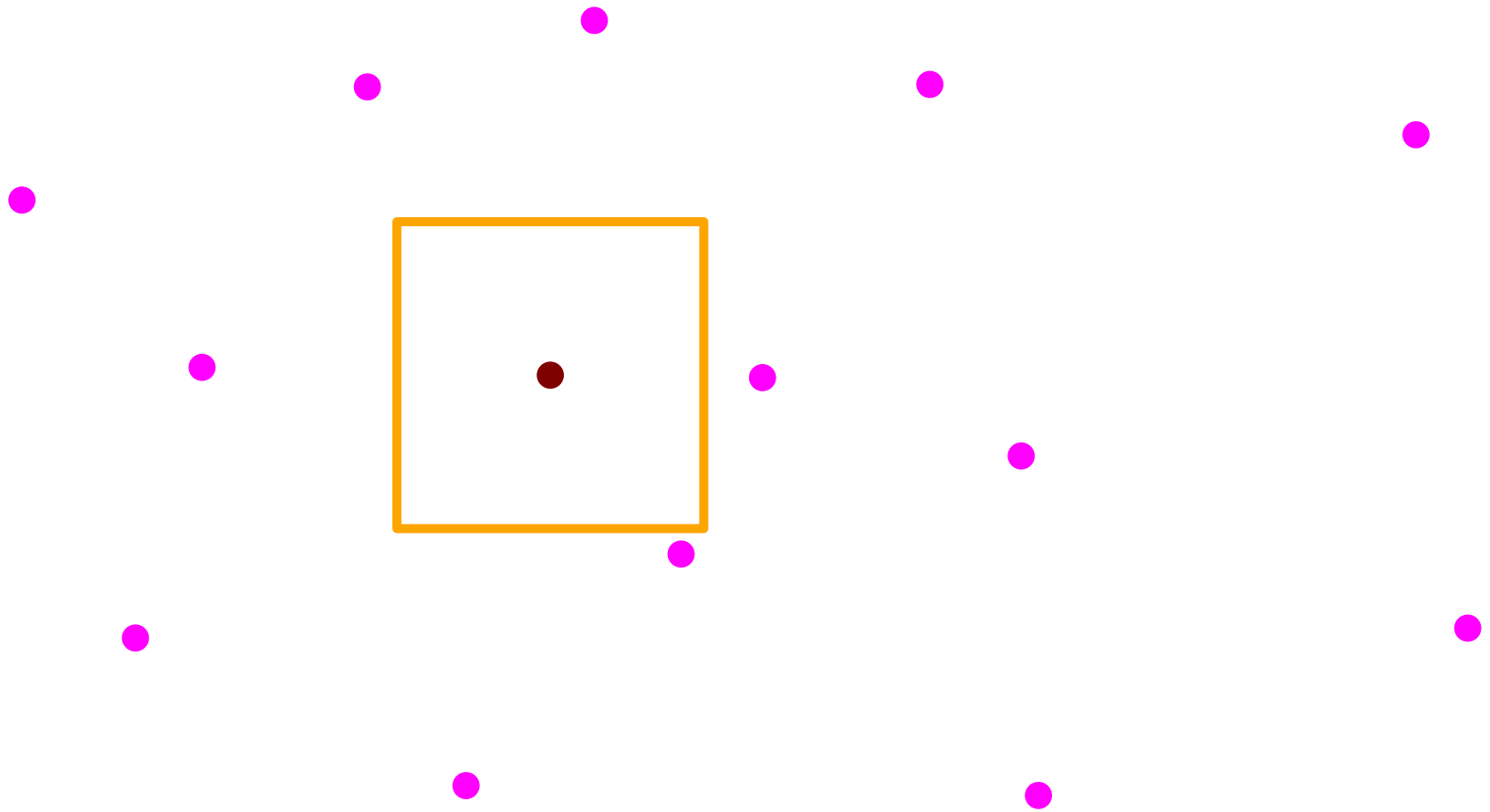
Norm L_∞: max($|x|, |y|$)

query



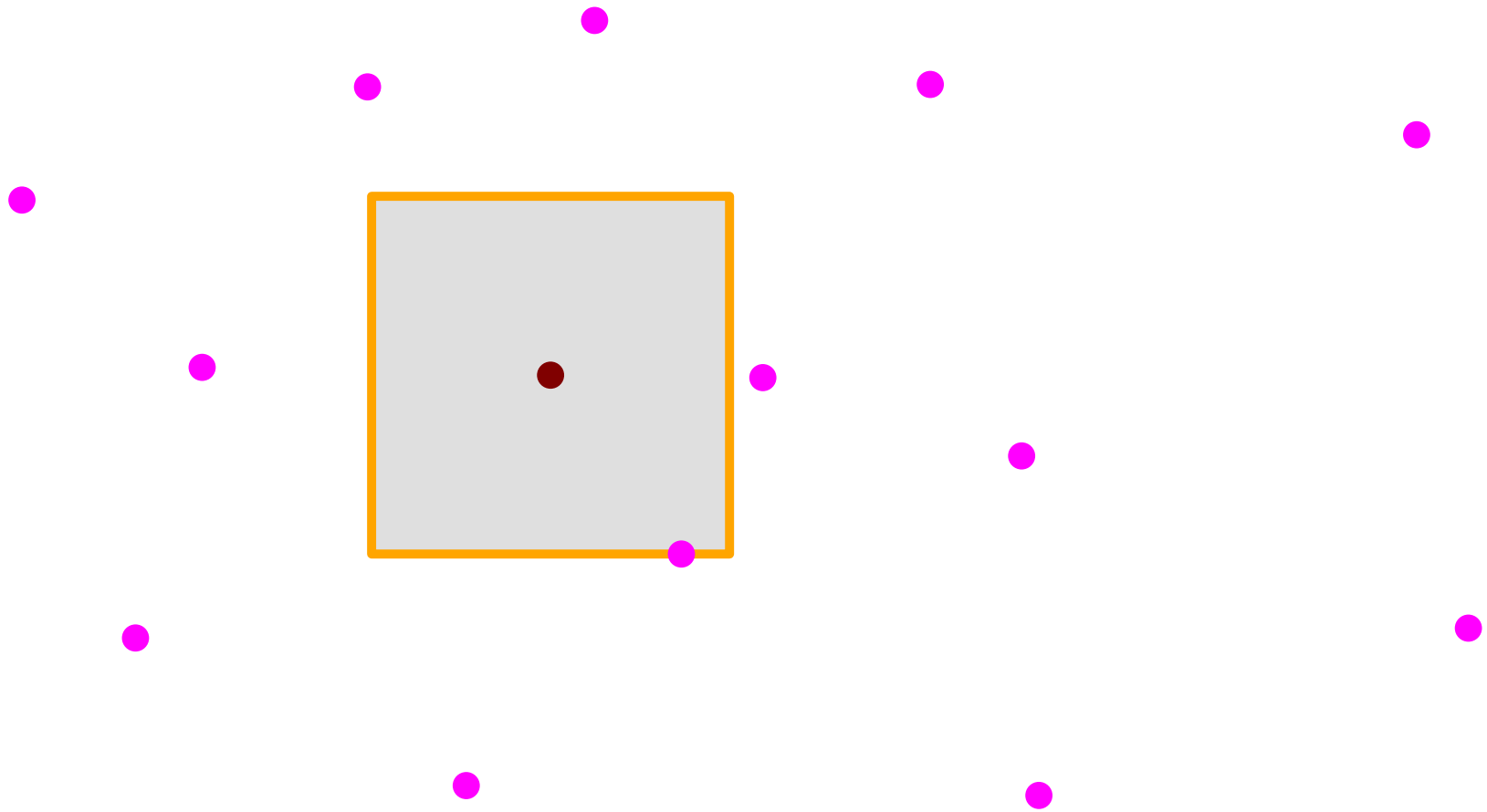
Norm L_∞: max($|x|, |y|$)

query

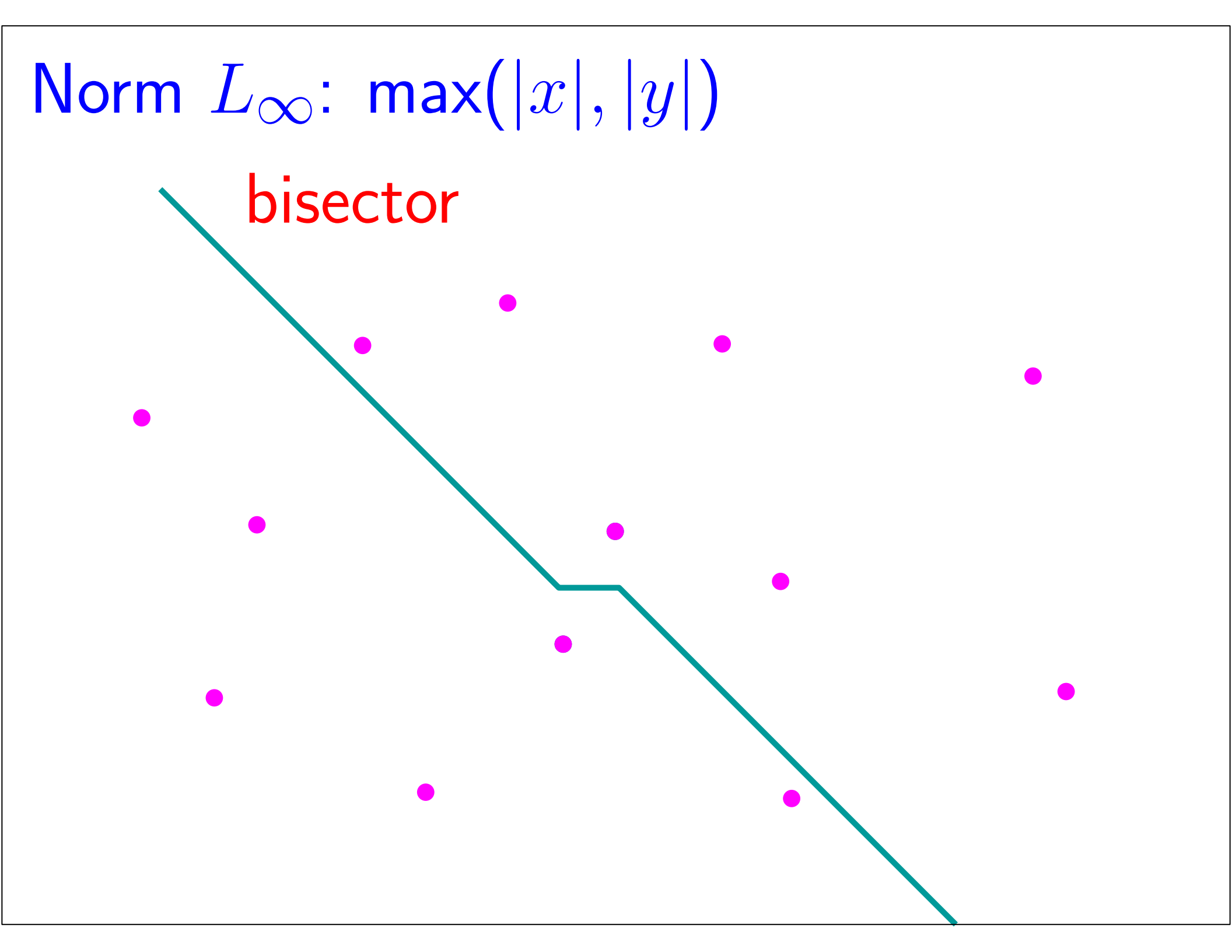


Norm L_∞: max($|x|, |y|$)

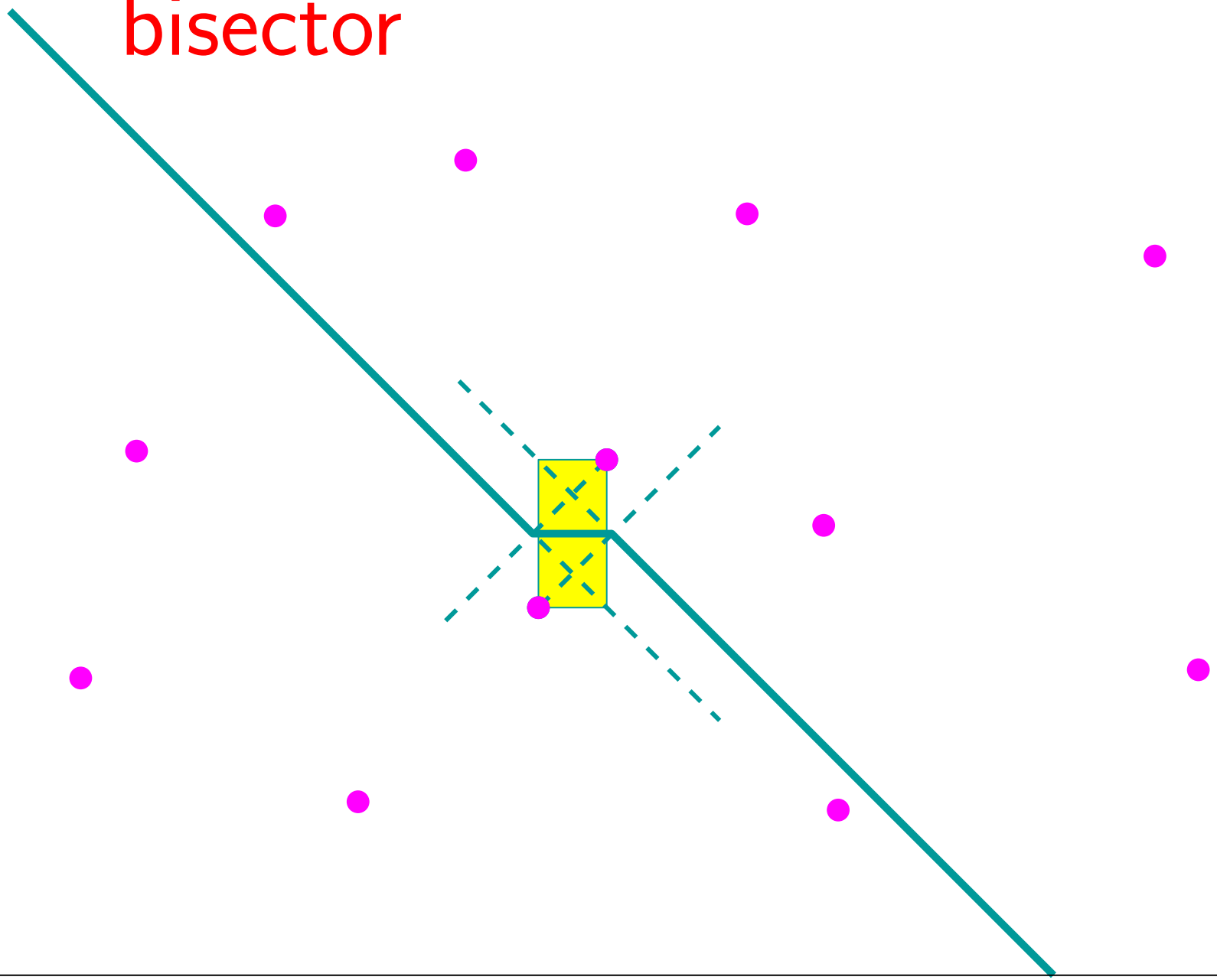
query



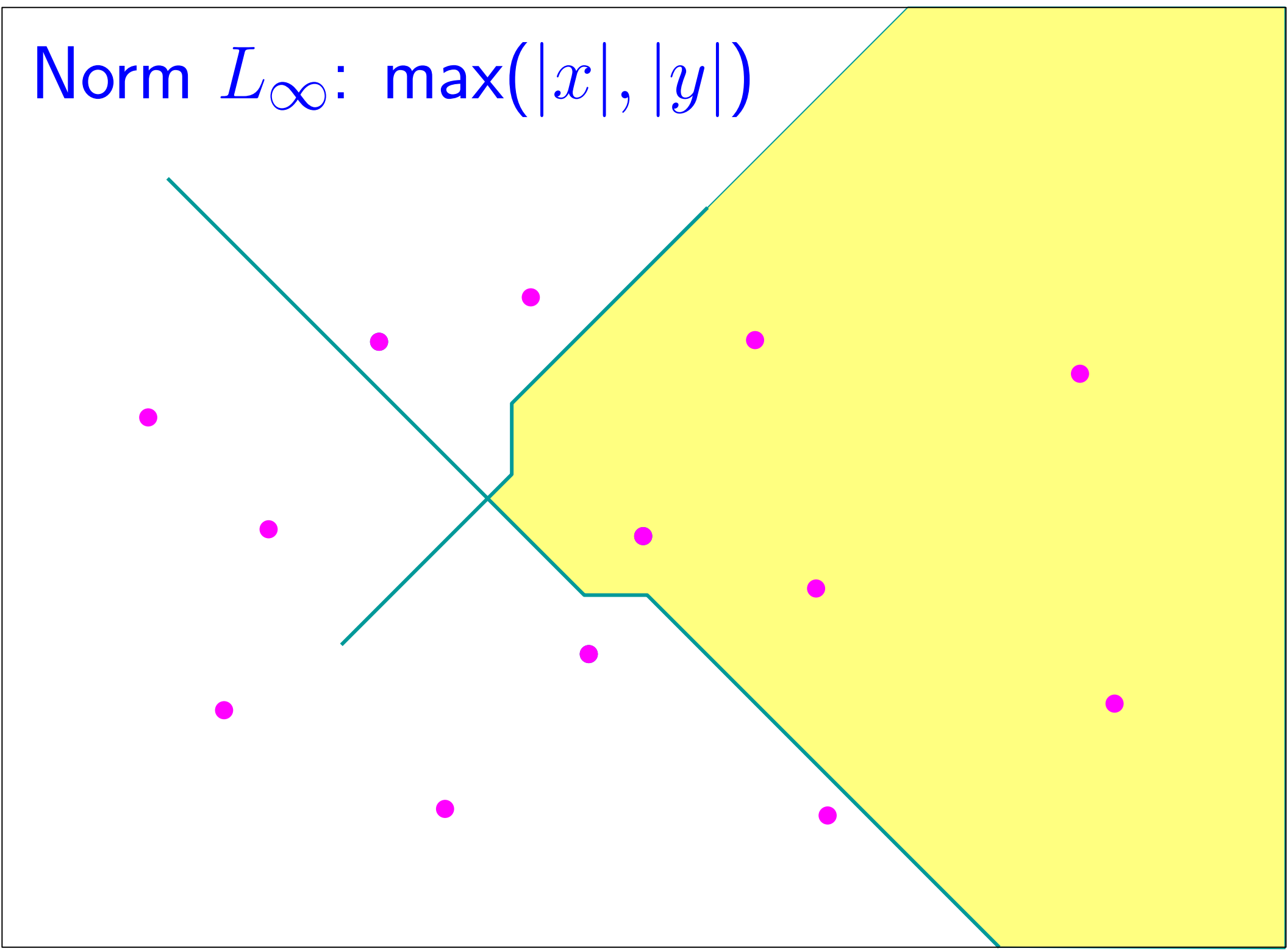
Norm L_∞: max($|x|, |y|$)



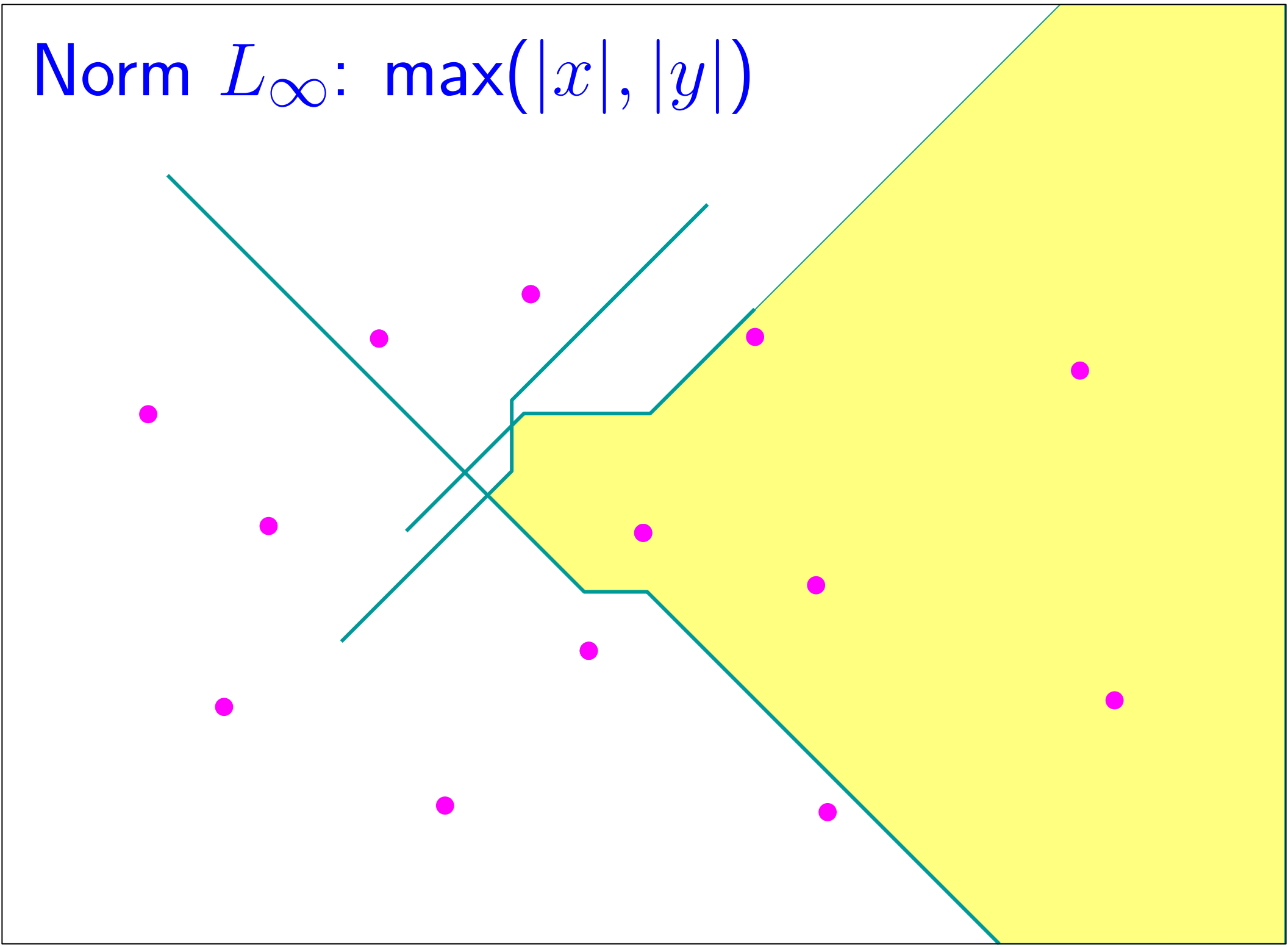
Norm L_∞: max($|x|, |y|$)



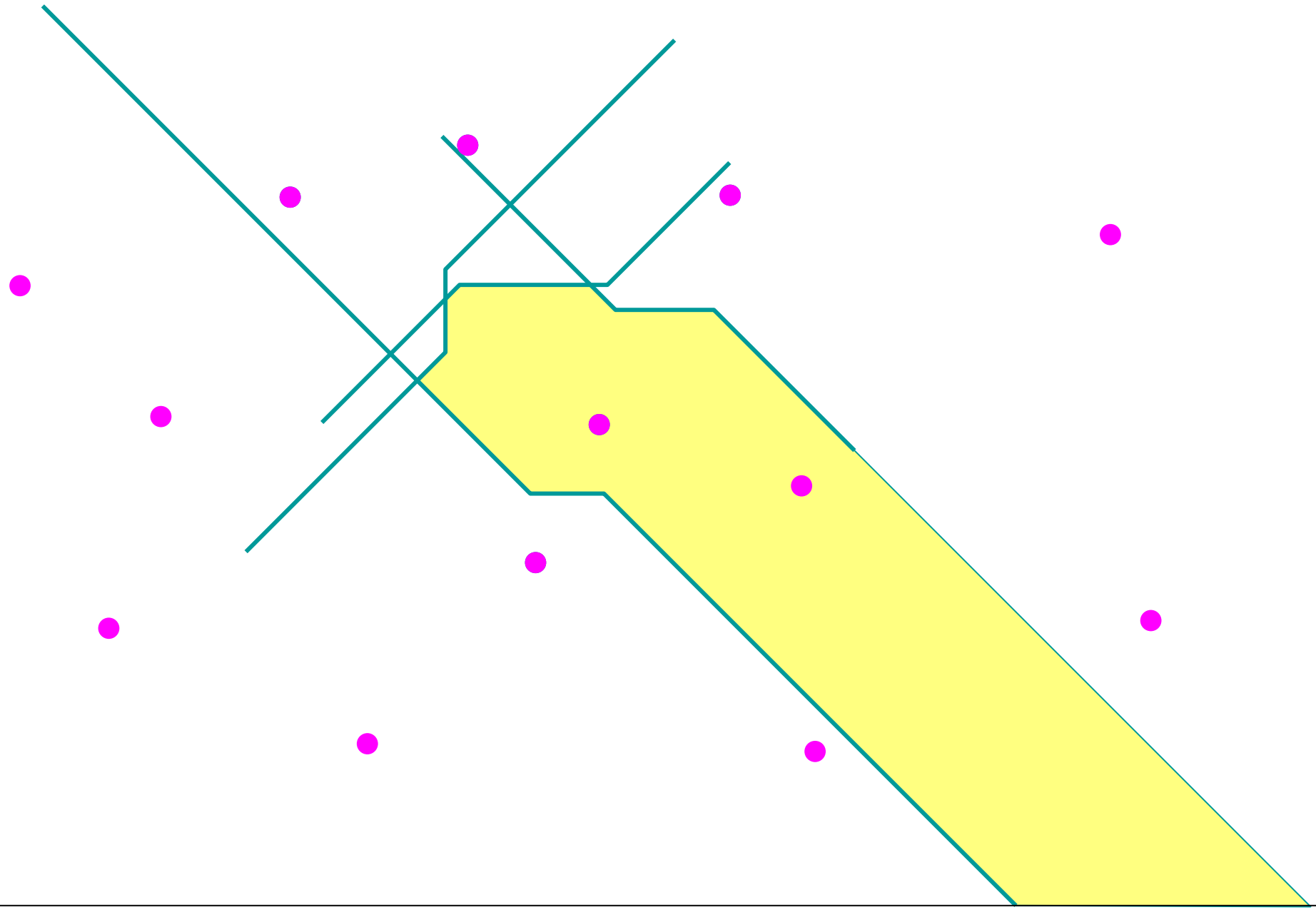
Norm L_∞: max($|x|, |y|$)



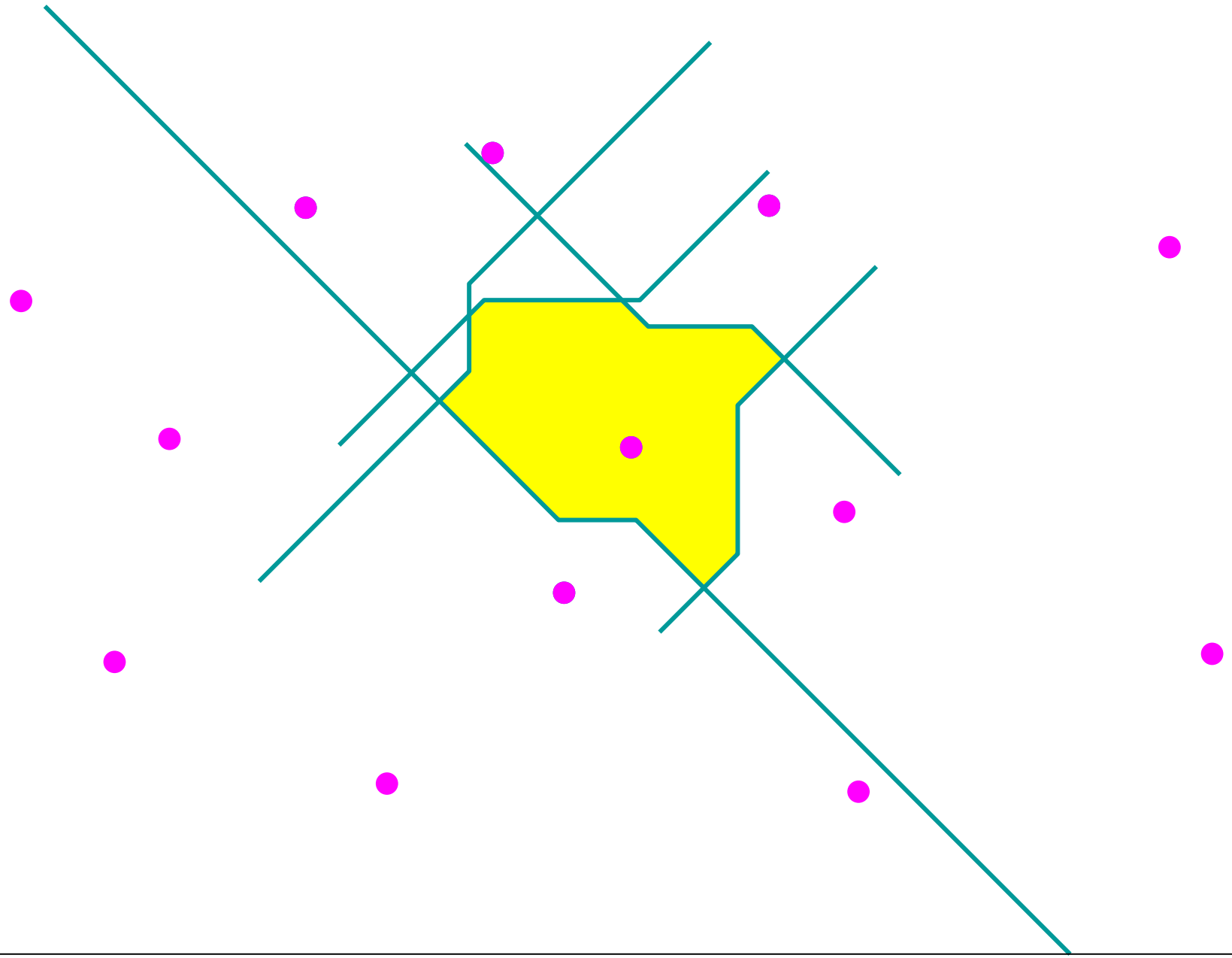
Norm L_∞: max($|x|, |y|$)

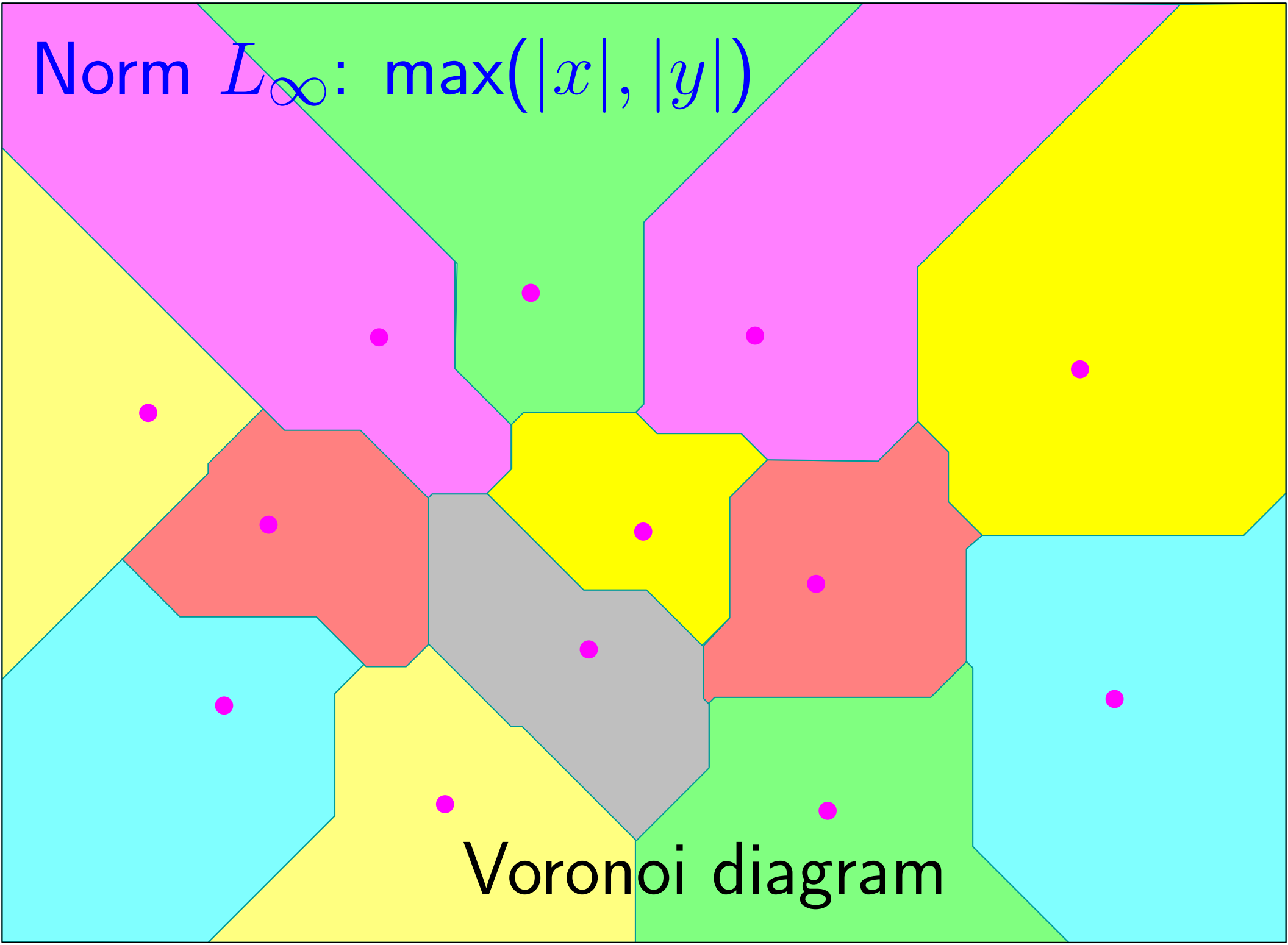


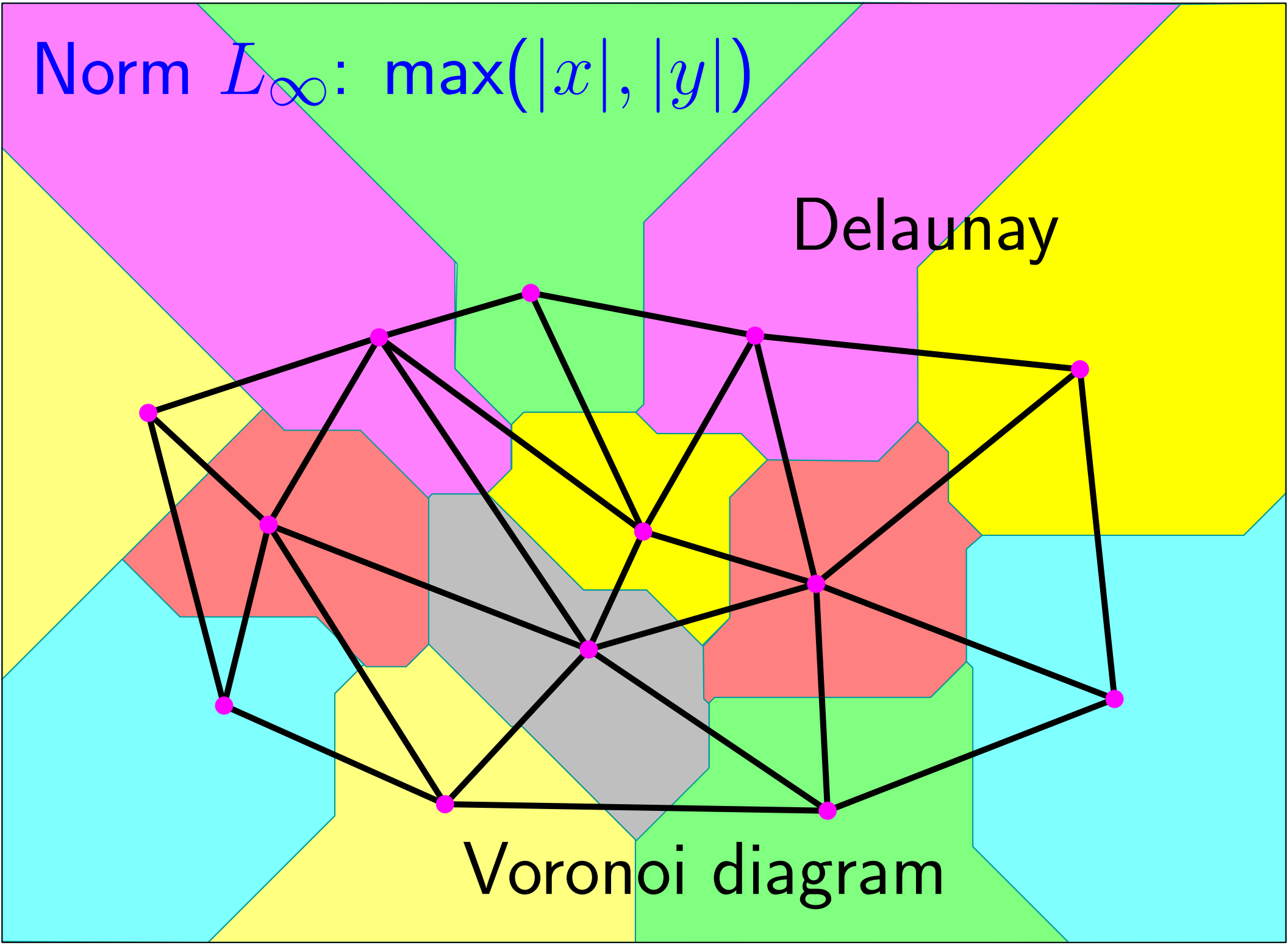
Norm L_∞: max($|x|, |y|$)



Norm L_∞: max($|x|, |y|$)




Norm L_∞: max($|x|, |y|$)
Voronoi diagram

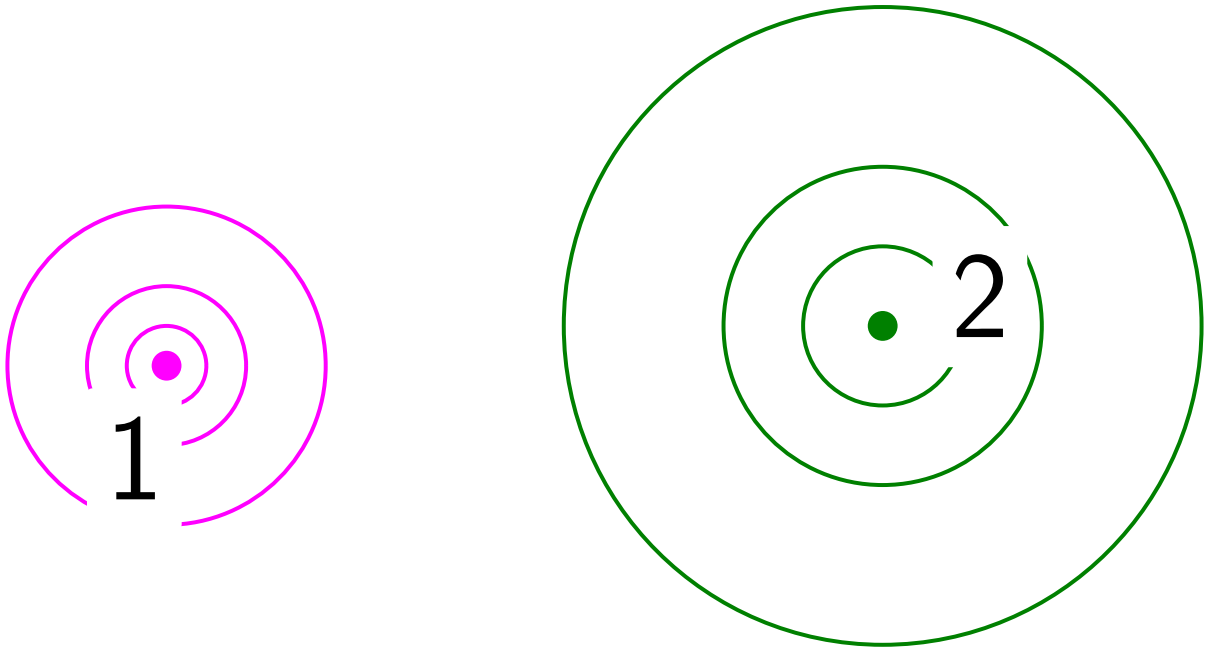

Norm L_∞: max($|x|, |y|$)
Delaunay
Voronoi diagram

multiplicatively-weighted points

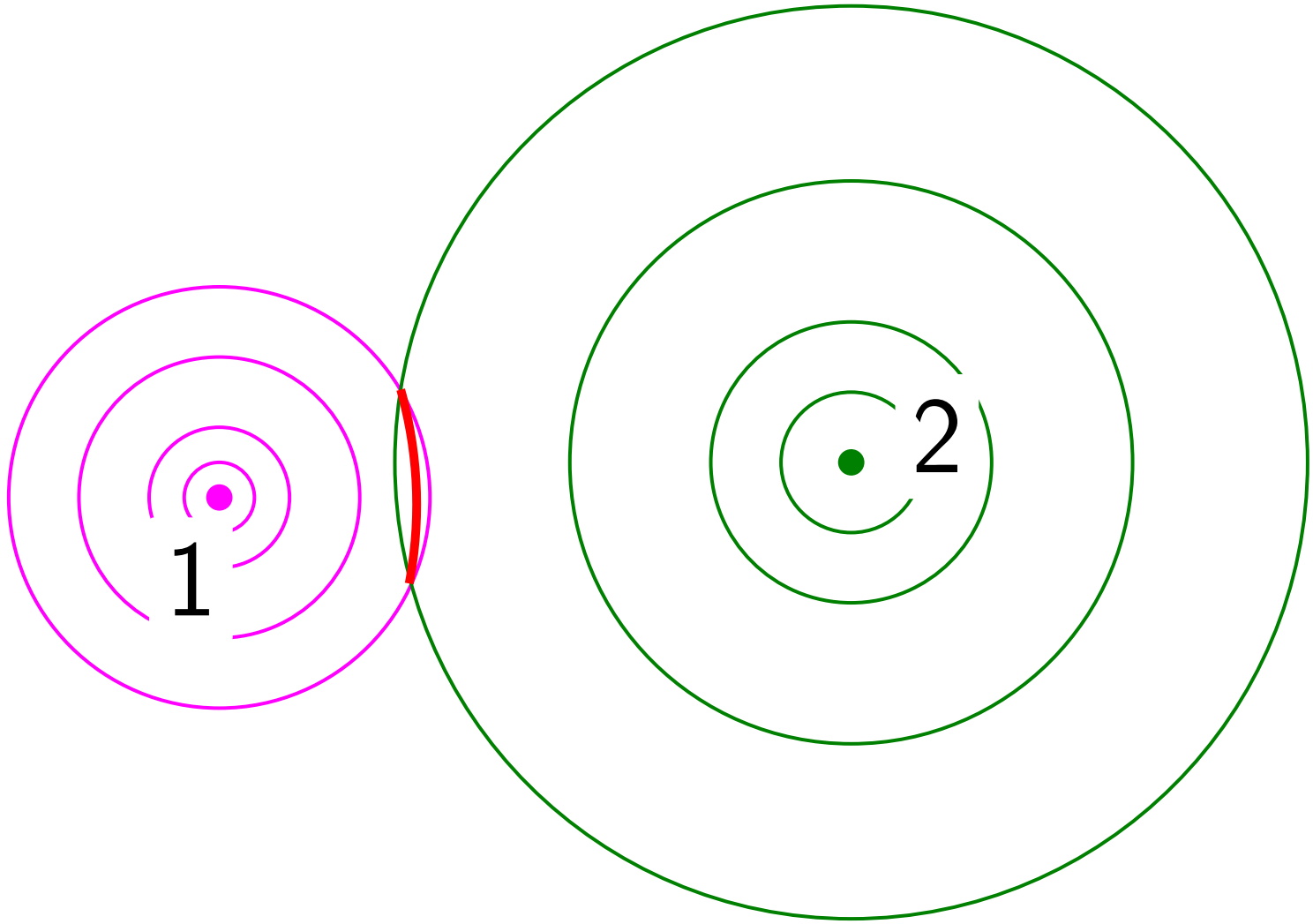
multiplicatively-weighted points

multiplicatively-weighted points

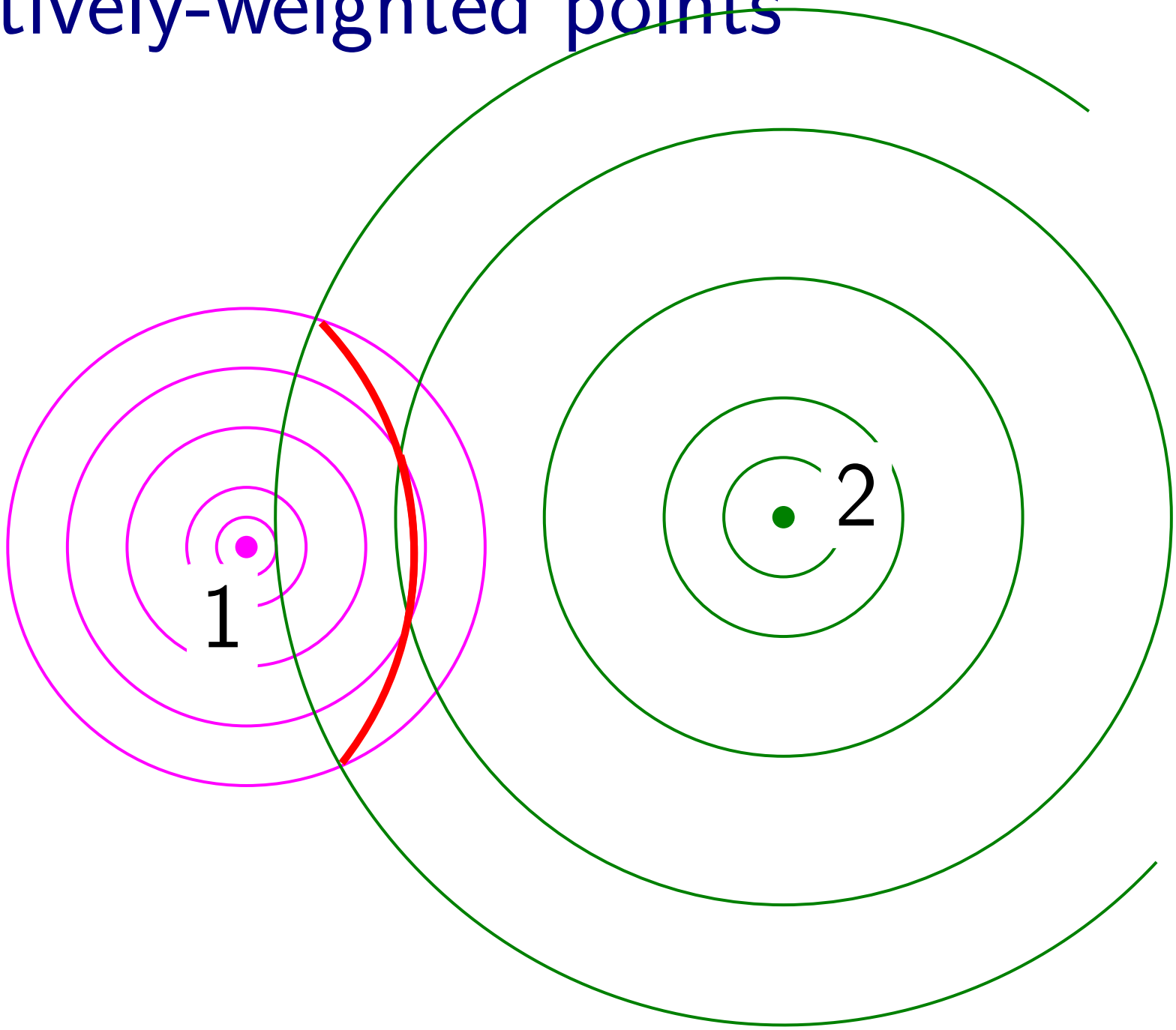
multiplicatively-weighted points



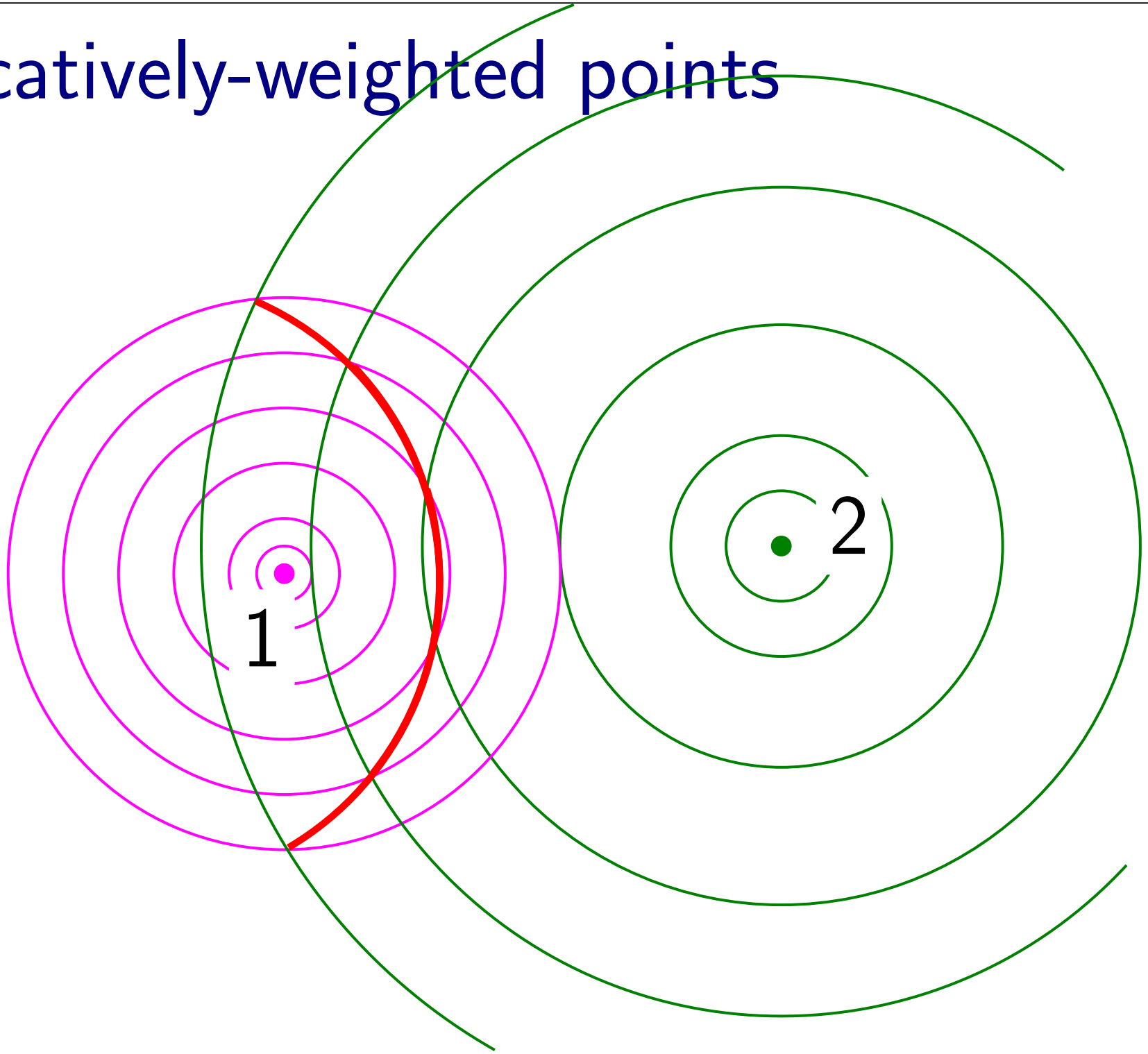
multiplicatively-weighted points

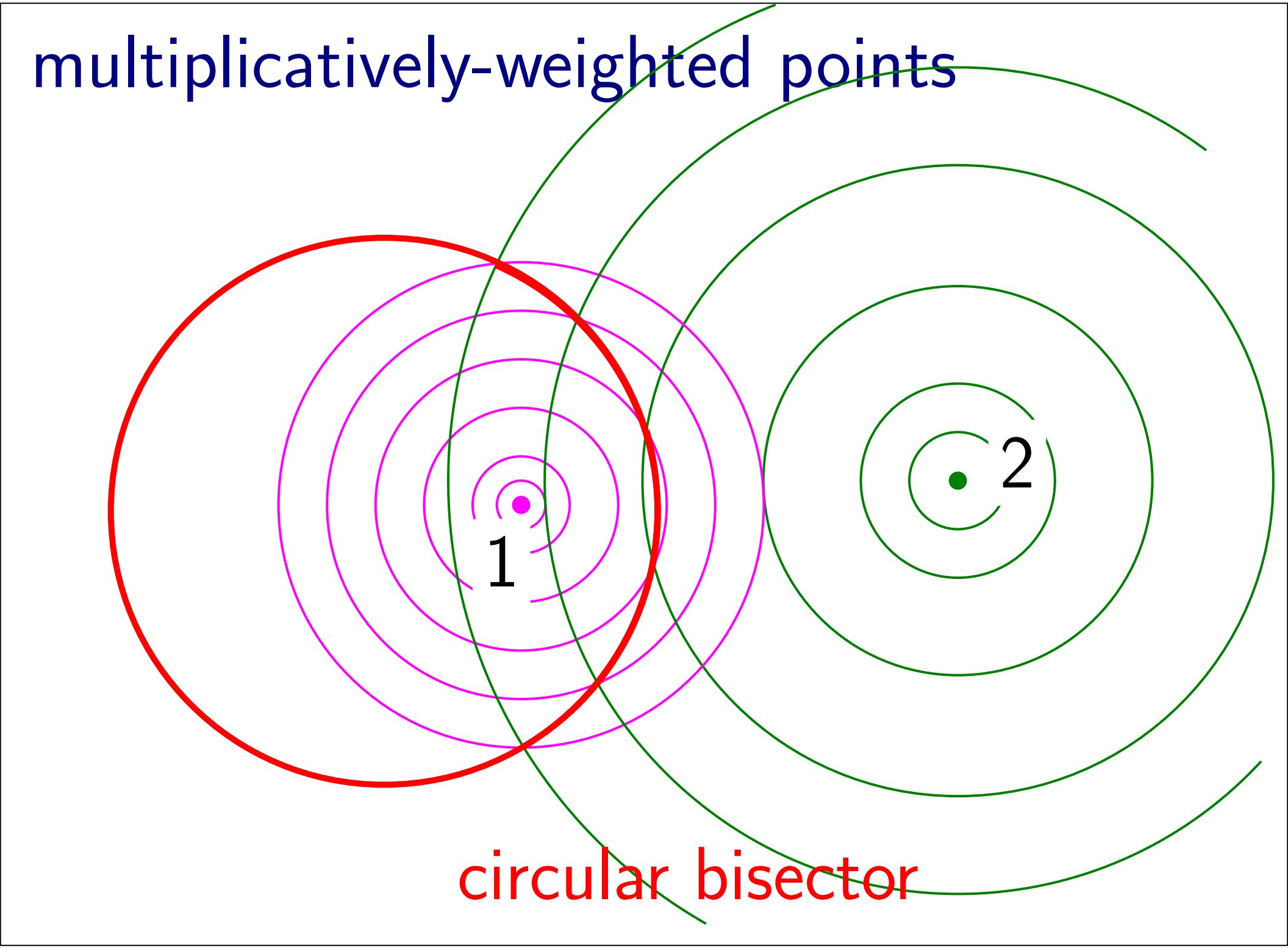


multiplicatively-weighted points

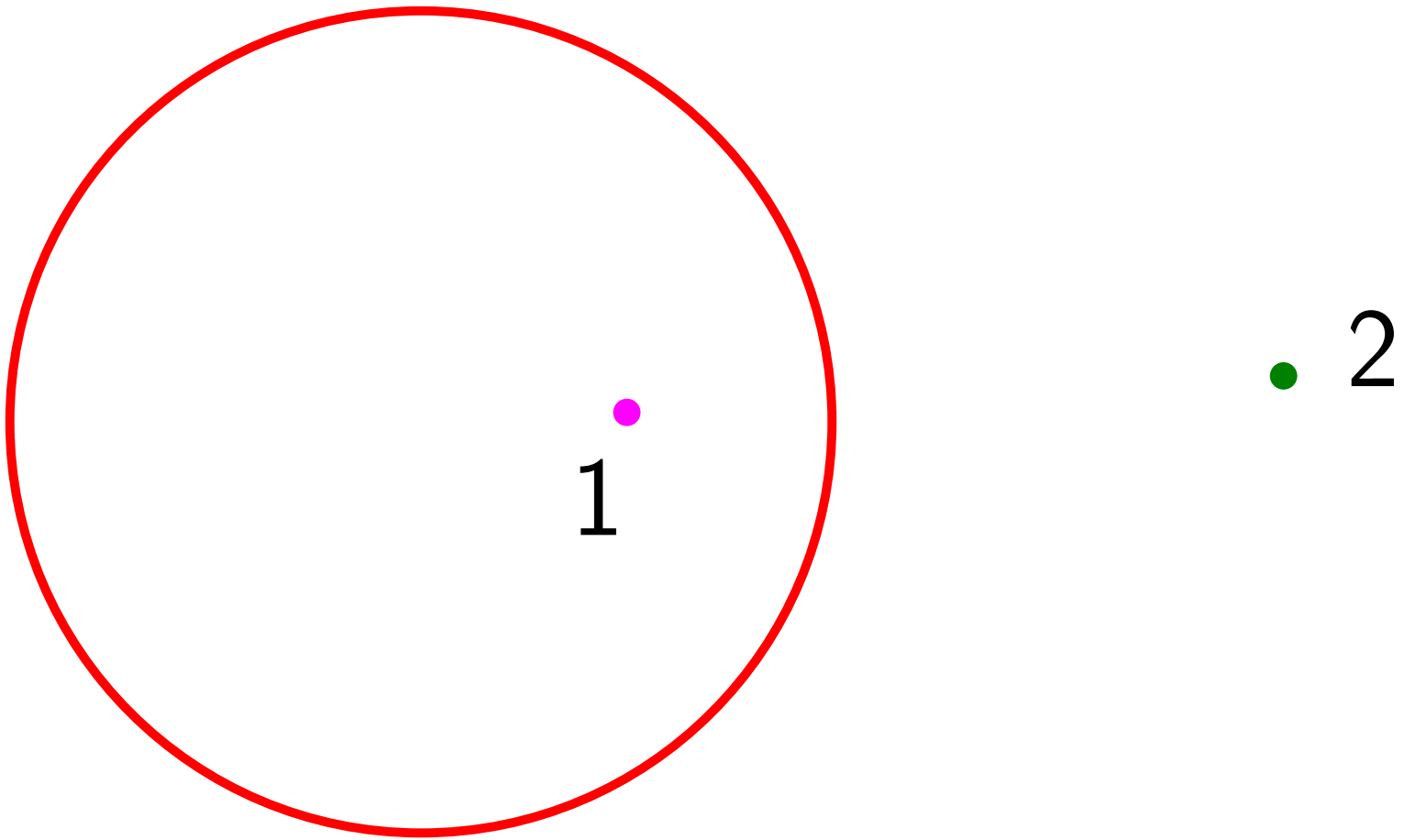


multiplicatively-weighted points



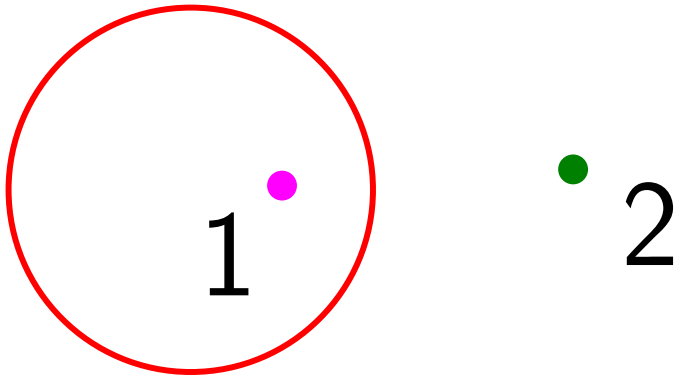

multiplicatively-weighted points
1
2
circular bisector

multiplicatively-weighted points

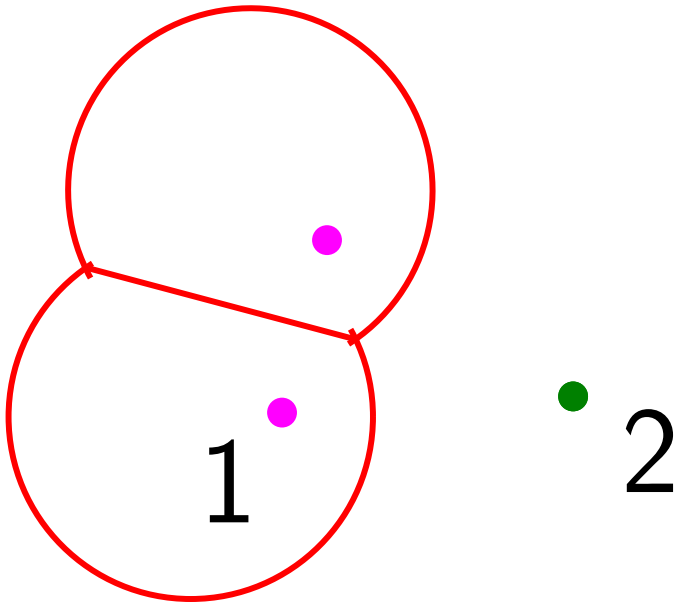


circular bisector

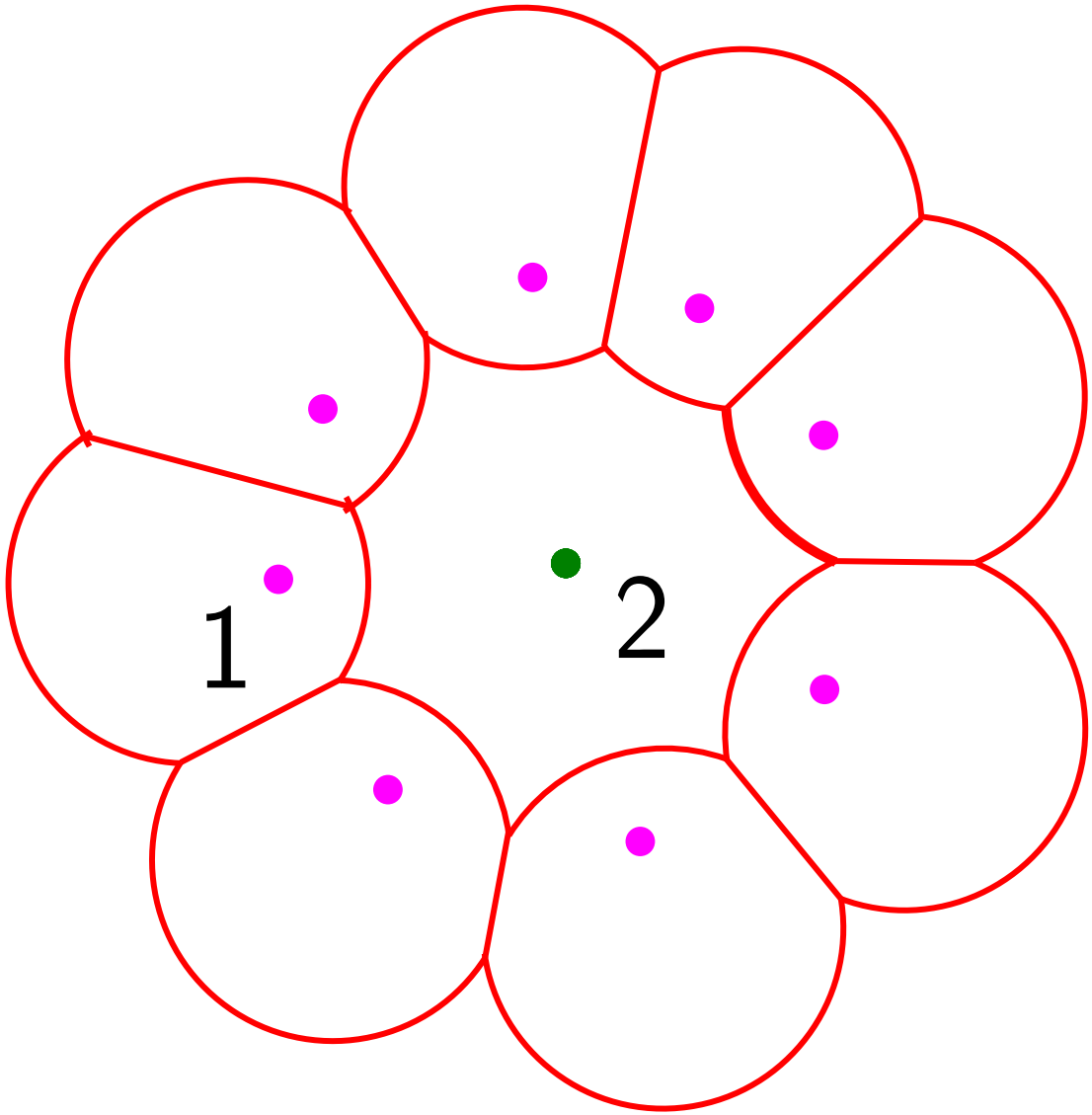
multiplicatively-weighted points



multiplicatively-weighted points



multiplicatively-weighted points

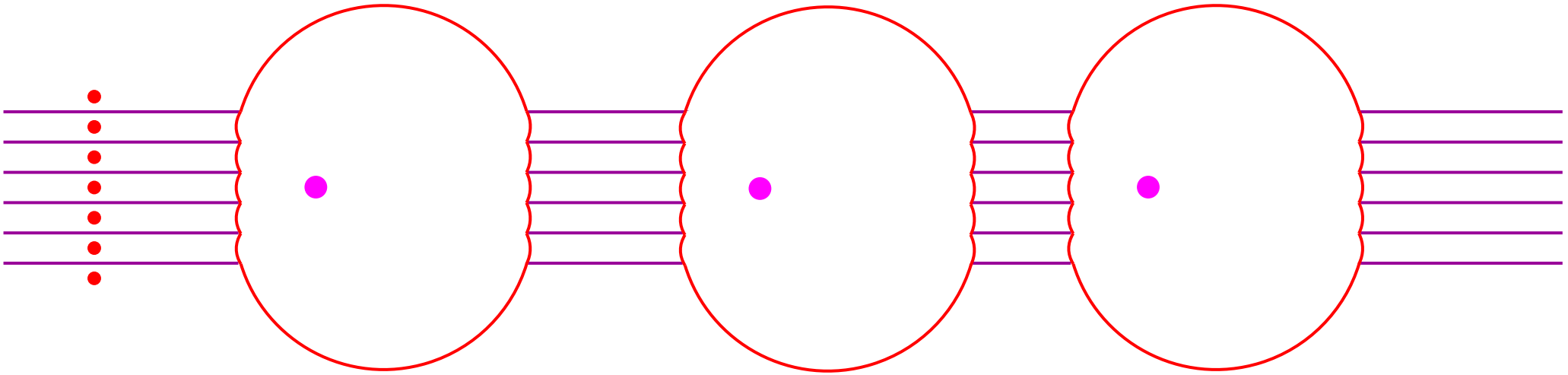


disconnected cell

multiplicatively-weighted points

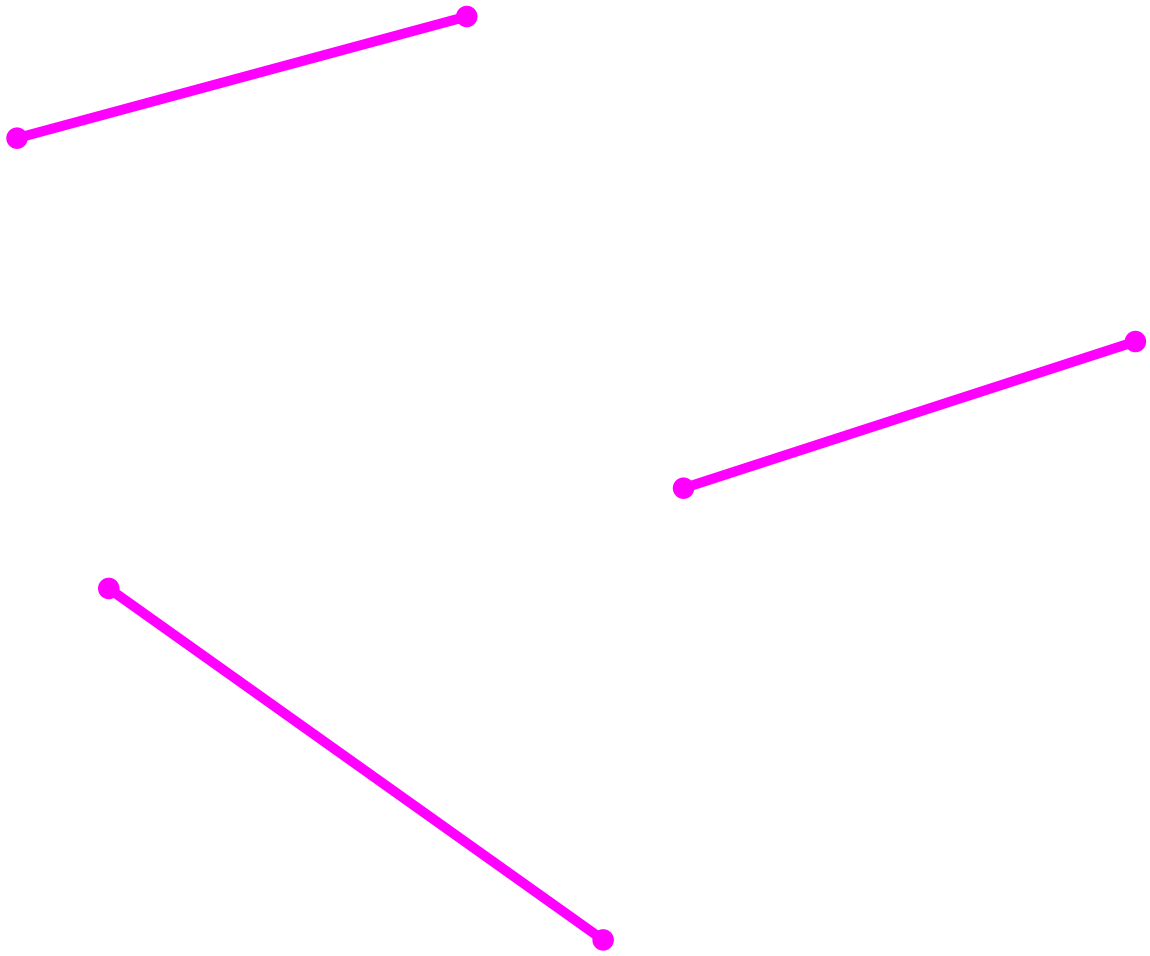
multiplicatively-weighted points

multiplicatively-weighted points

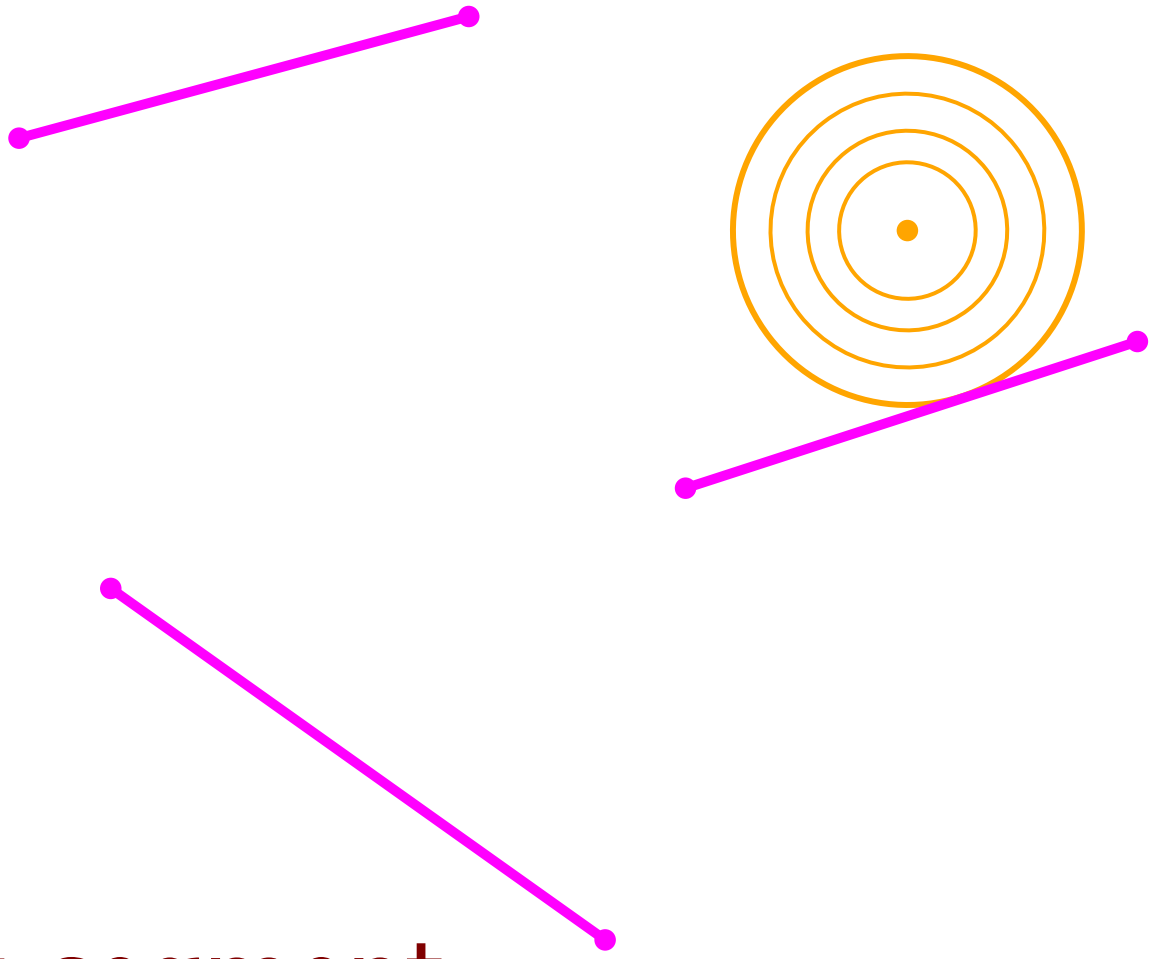


quadratic size

Voronoi diagram of segments

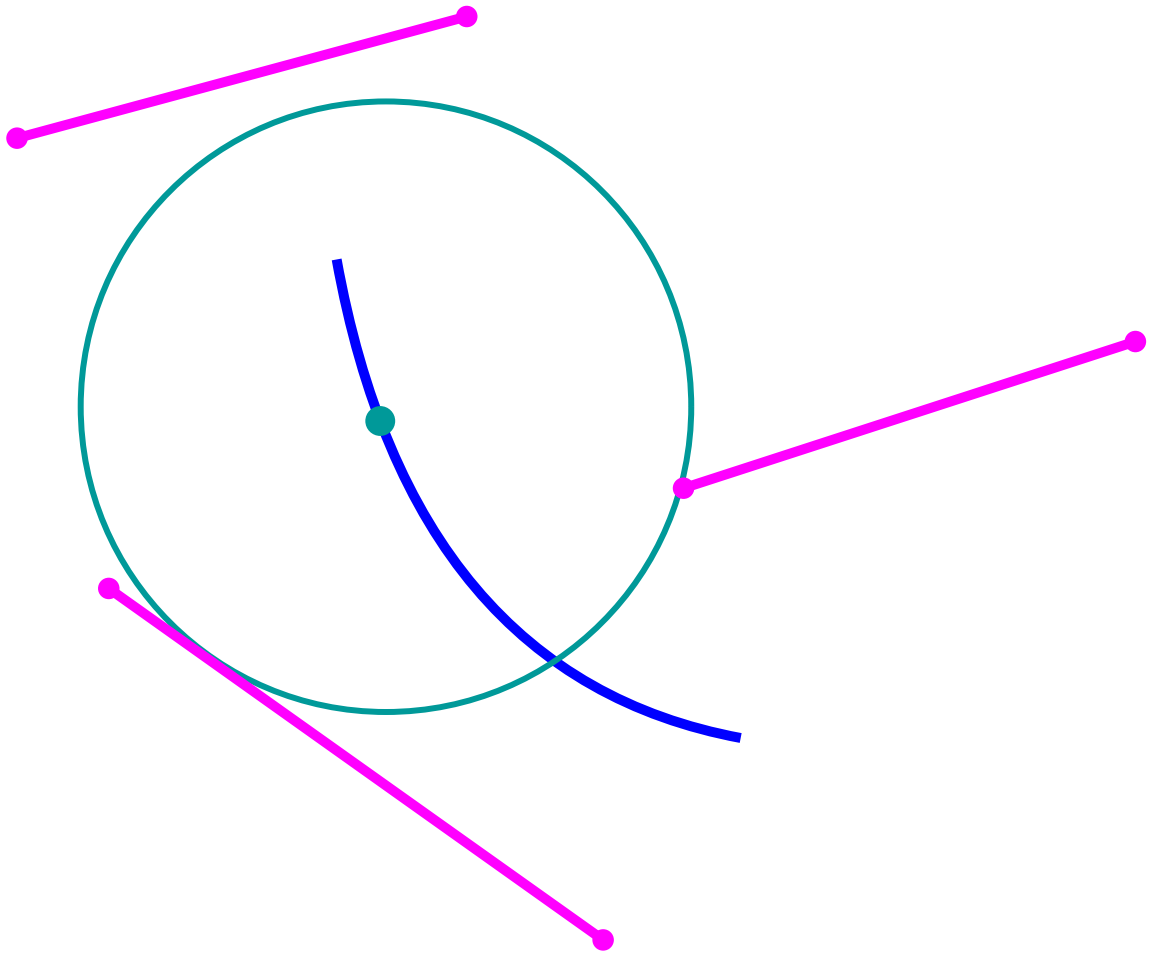


Voronoi diagram of segments



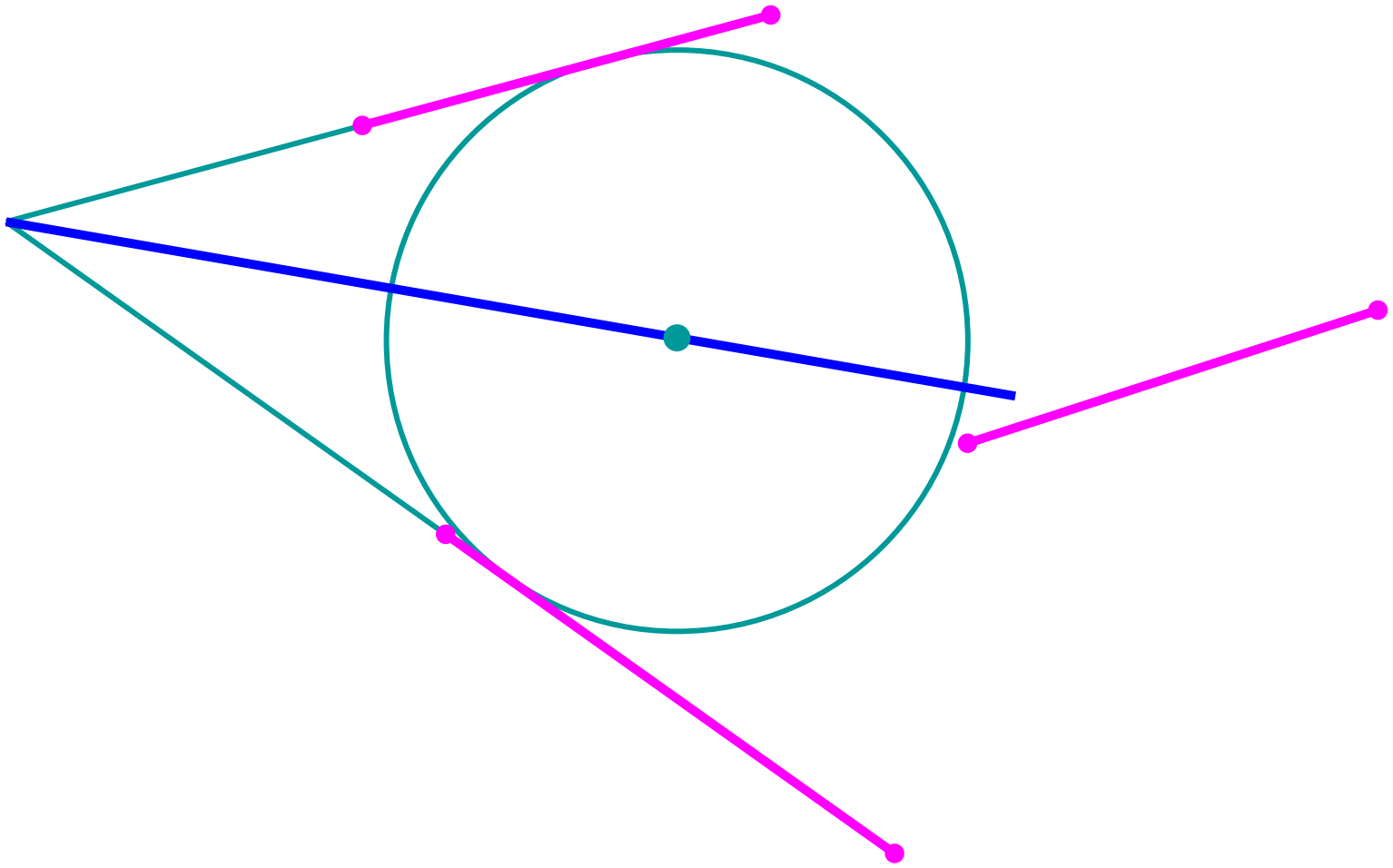
Q nearest segment

Voronoi diagram of segments



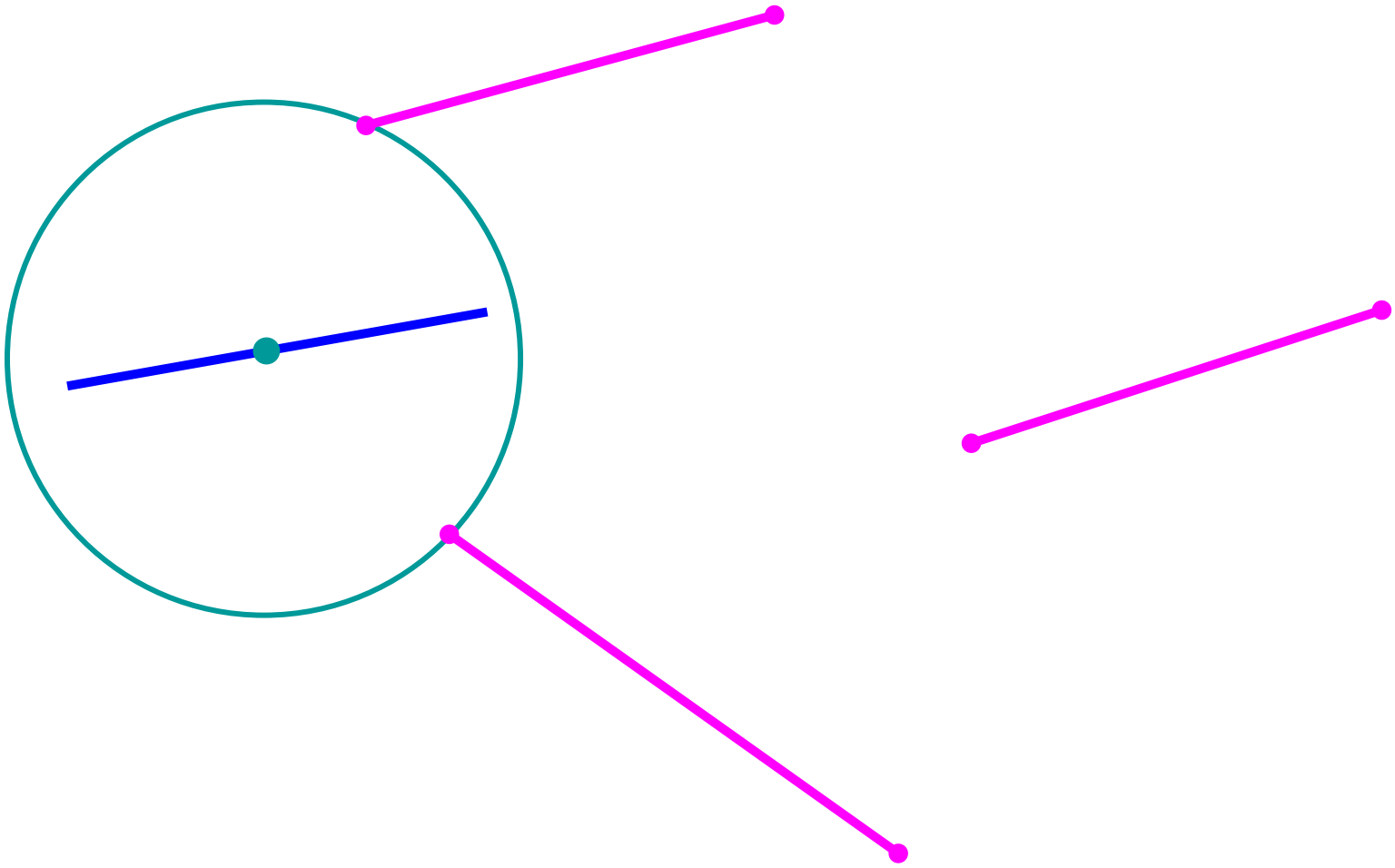
parabolic bisector

Voronoi diagram of segments



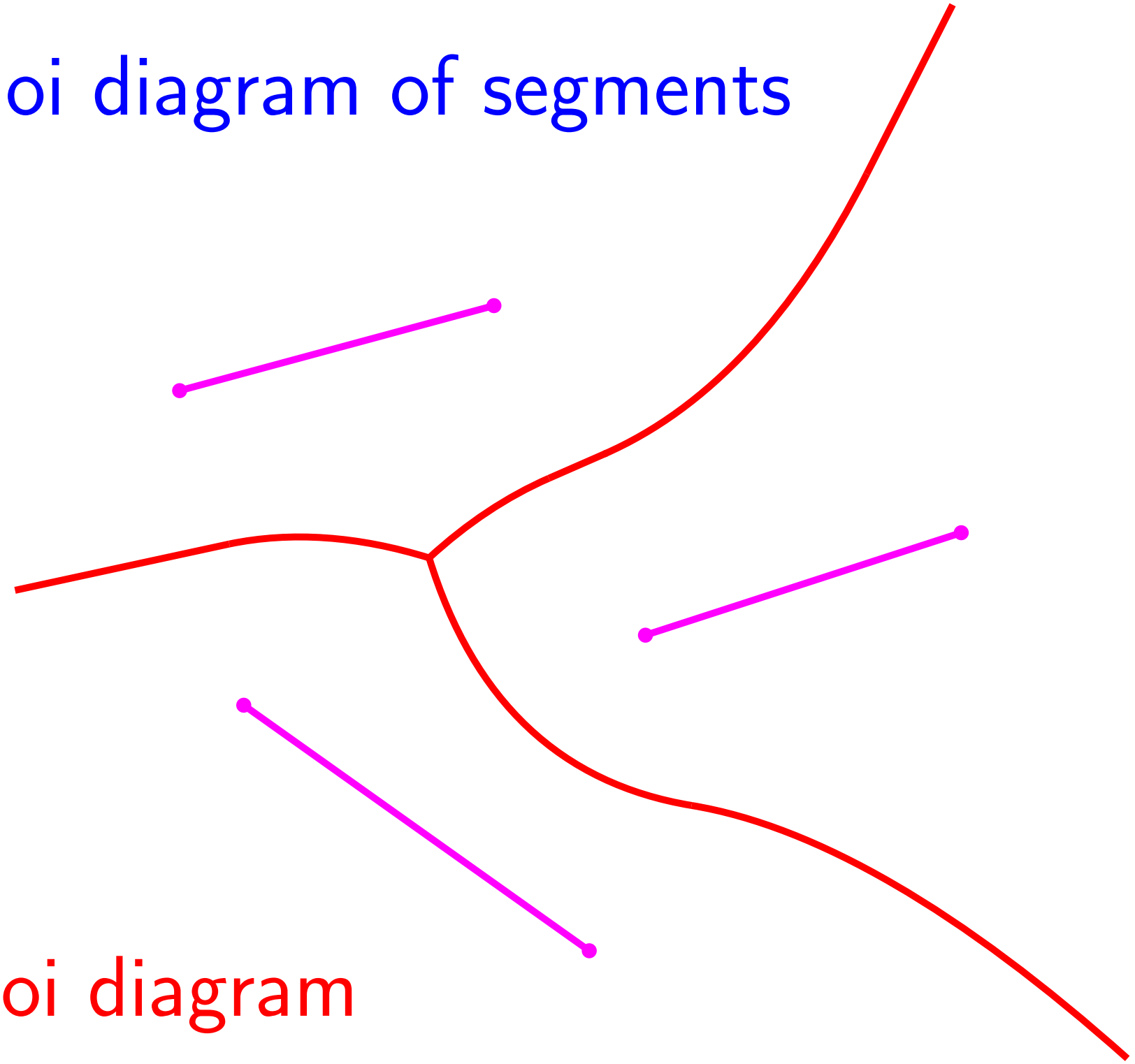
angle bisector

Voronoi diagram of segments



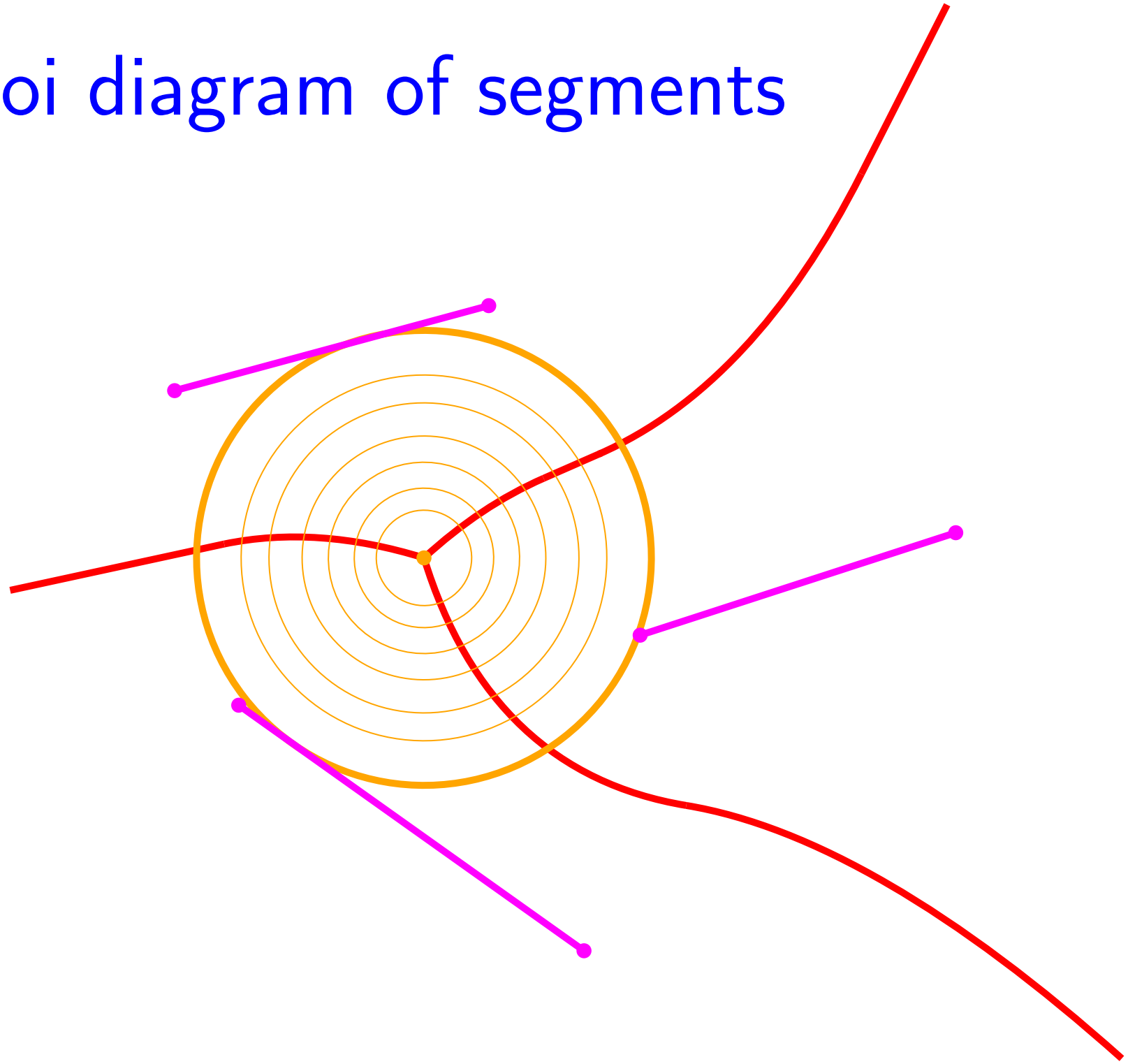
points bisector

Voronoi diagram of segments

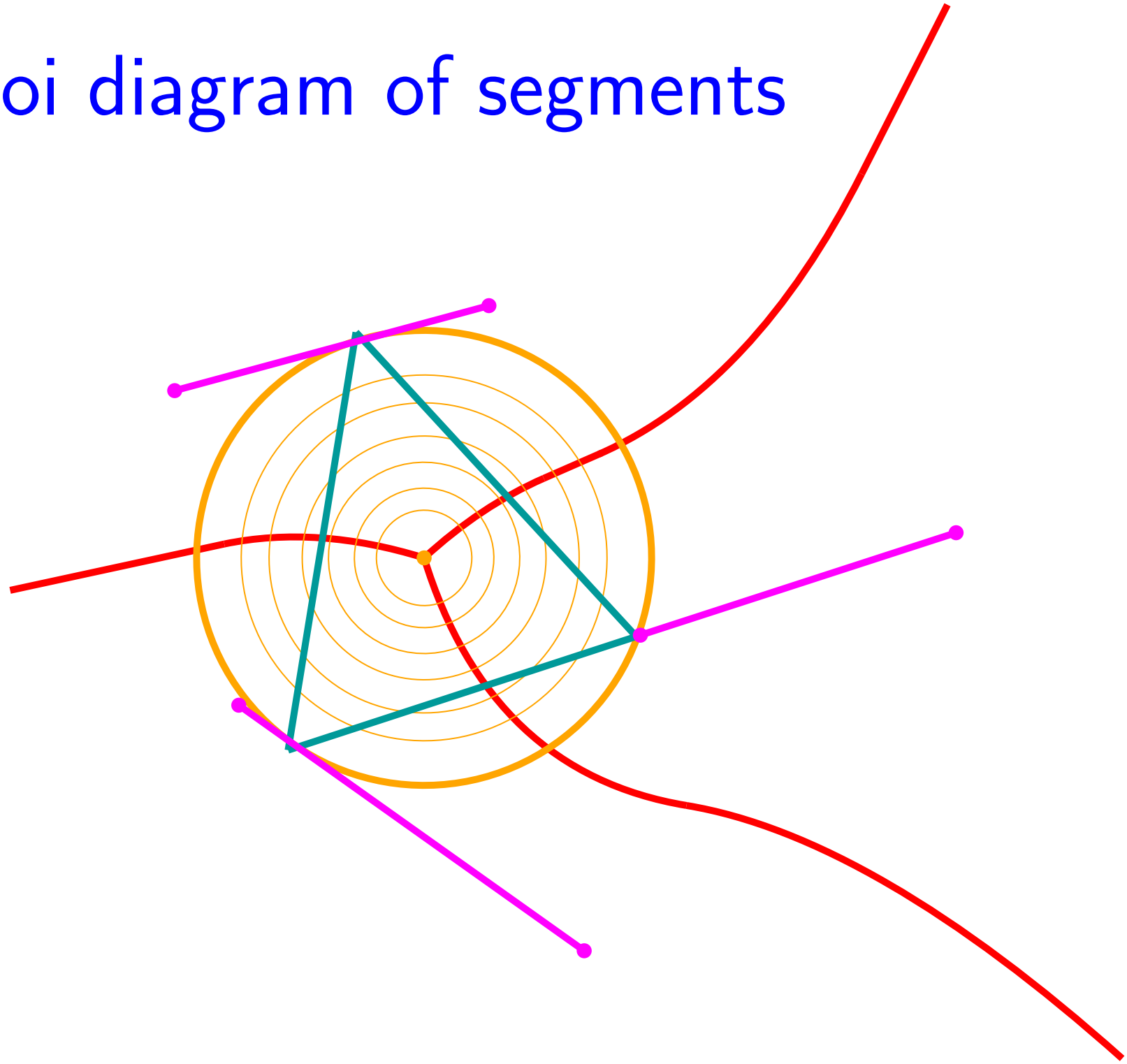


Voronoi diagram

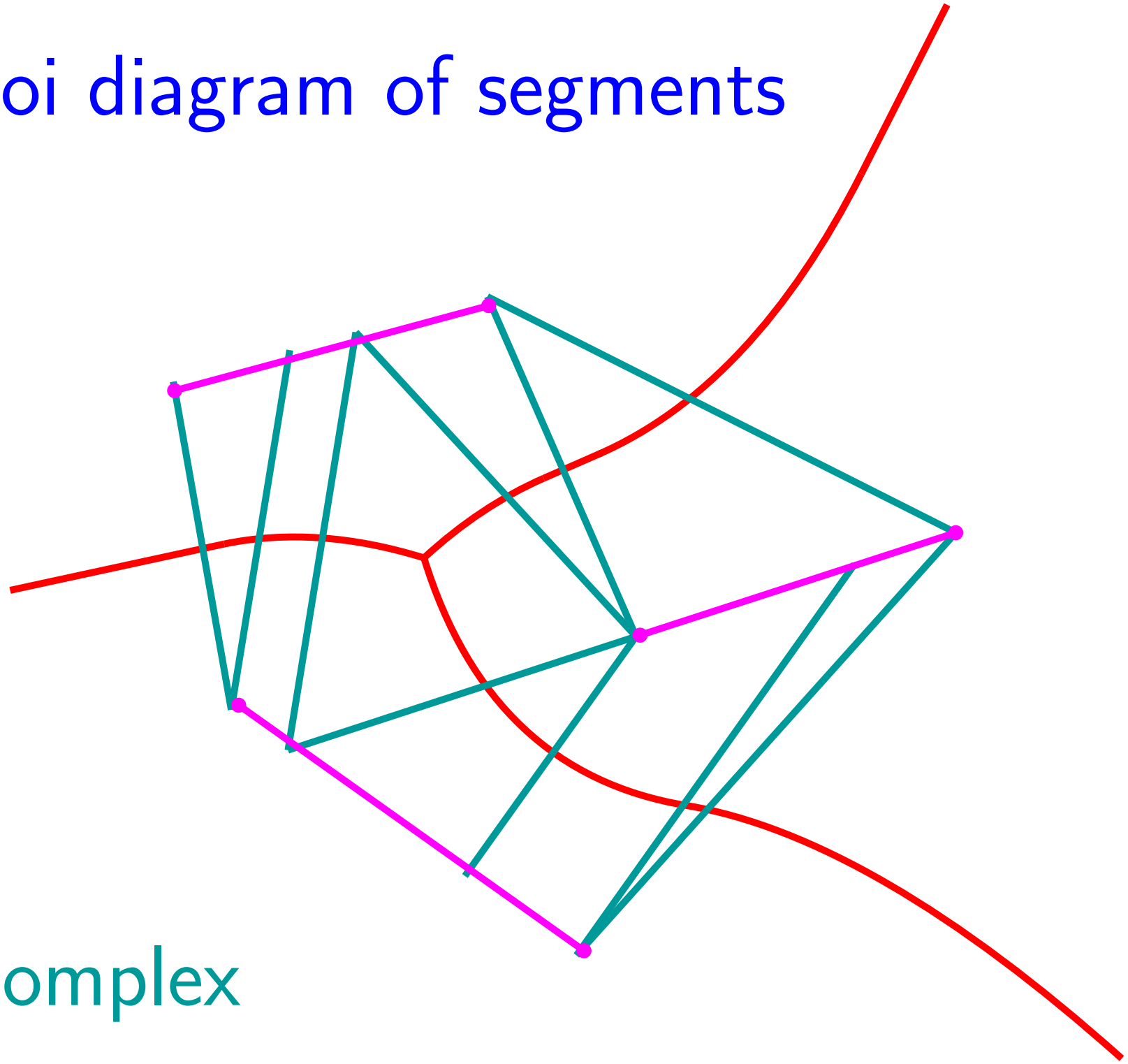
Voronoi diagram of segments



Voronoi diagram of segments



Voronoi diagram of segments



Dual complex

Laguerre geometry

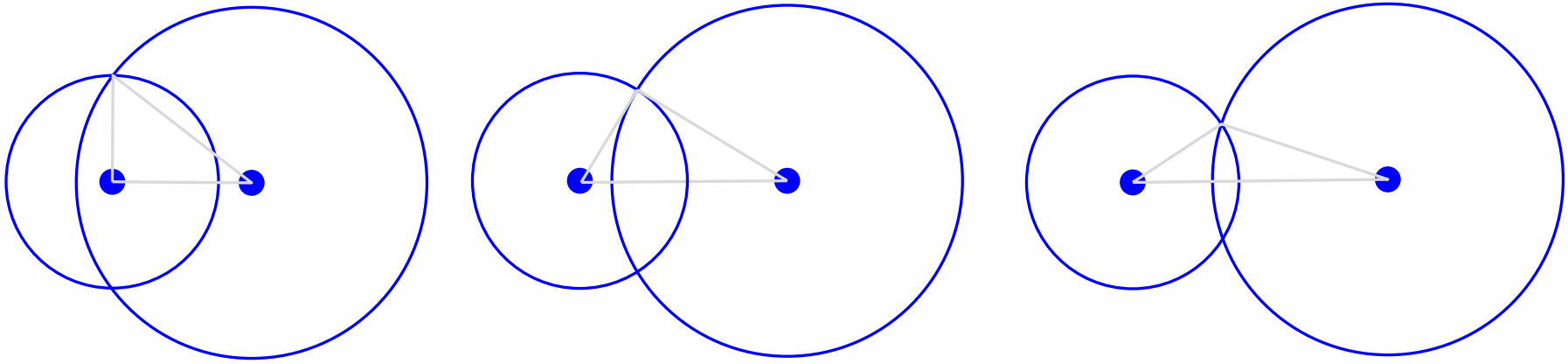
Power distance of two balls or of two weighted points.

ball $b_1(p_1, r_1)$, center p_1 radius r_1 $\longleftrightarrow$ weigthed point $(p_1, r_1^2) \in \mathbb{R}^d$
ball $b_2(p_2, r_2)$, center p_2 radius r_2 $\longleftrightarrow$ weigthed point $(p_2, r_2^2) \in \mathbb{R}^d$

$$\pi(b_1, b_2) = (p_1 - p_2)^2 - r_1^2 - r_2^2$$

Orthogonal balls

$$b_1, b_2 \text{ closer} \iff \pi(b_1, b_2) < 0 \iff (p_1 - p_2)^2 \leq r_1^2 + r_2^2$$
$$b_1, b_2 \text{ orthogonal} \iff \pi(b_1, b_2) = 0 \iff (p_1 - p_2)^2 = r_1^2 + r_2^2$$
$$b_1, b_2 \text{ further} \iff \pi(b_1, b_2) > 0 \iff (p_1 - p_2)^2 \leq r_1^2 + r_2^2$$



Laguerre geometry

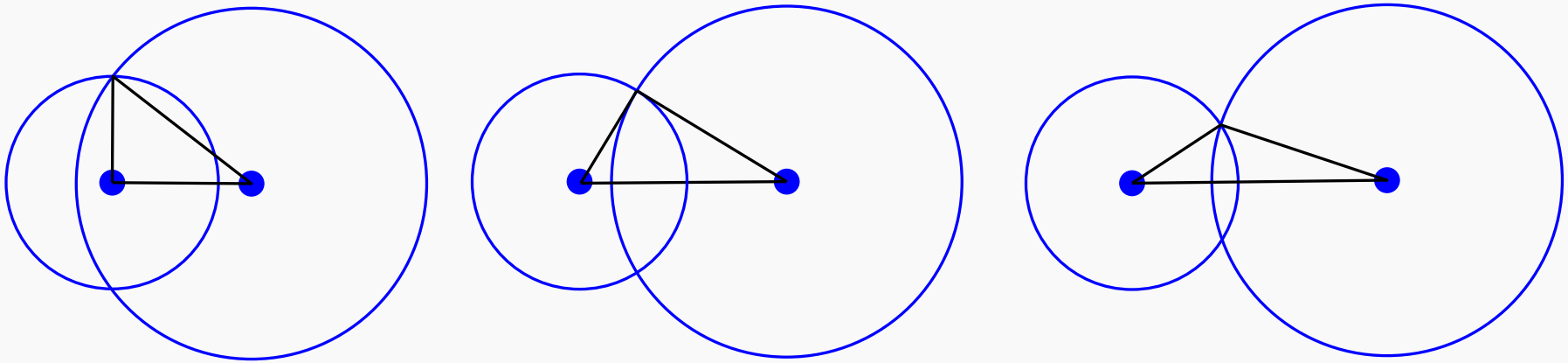
Power distance of two balls or of two weighted points.

ball $b_1(p_1, r_1)$, center p_1 radius r_1 $\longleftrightarrow$ weigthed point $(p_1, r_1^2) \in \mathbb{R}^d$
ball $b_2(p_2, r_2)$, center p_2 radius r_2 $\longleftrightarrow$ weigthed point $(p_2, r_2^2) \in \mathbb{R}^d$

$$\pi(b_1, b_2) = (p_1 - p_2)^2 - r_1^2 - r_2^2$$

Orthogonal balls

$$\begin{aligned} b_1, b_2 \text{ closer} &\iff \pi(b_1, b_2) < 0 \iff (p_1 - p_2)^2 \leq r_1^2 + r_2^2 \\ b_1, b_2 \text{ orthogonal} &\iff \pi(b_1, b_2) = 0 \iff (p_1 - p_2)^2 = r_1^2 + r_2^2 \\ b_1, b_2 \text{ further} &\iff \pi(b_1, b_2) > 0 \iff (p_1 - p_2)^2 \leq r_1^2 + r_2^2 \end{aligned}$$

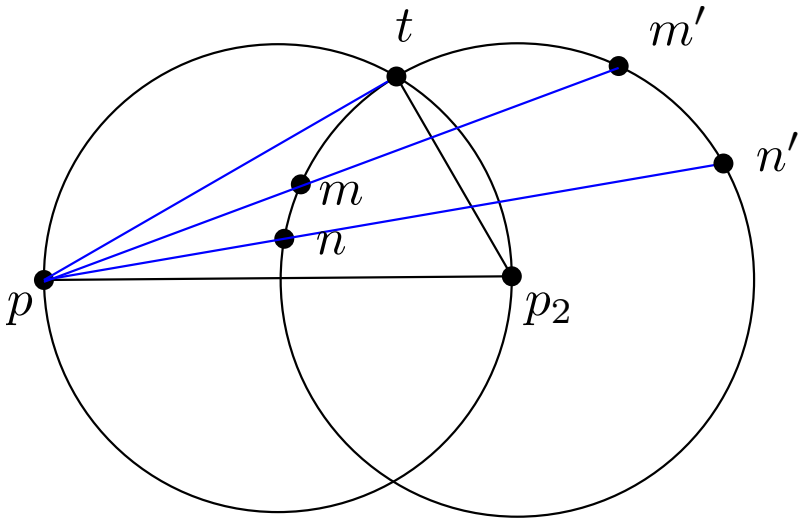


Power distance of a point wrt a ball

If b_1 is reduced to a point p : $\pi(p, b_2) = (p - p_2)^2 - r_2^2$

Normalized equation of bounding sphere :
$p \in \partial b_2 \iff \pi(p, b_2) = 0$

$$\begin{aligned} p \in \text{int} b_2 &\iff \pi(p, b) < 0 \\ p \in \partial b_2 &\iff \pi(p, b) = 0 \\ p \notin b_2 &\iff \pi(p, b) > 0 \end{aligned}$$

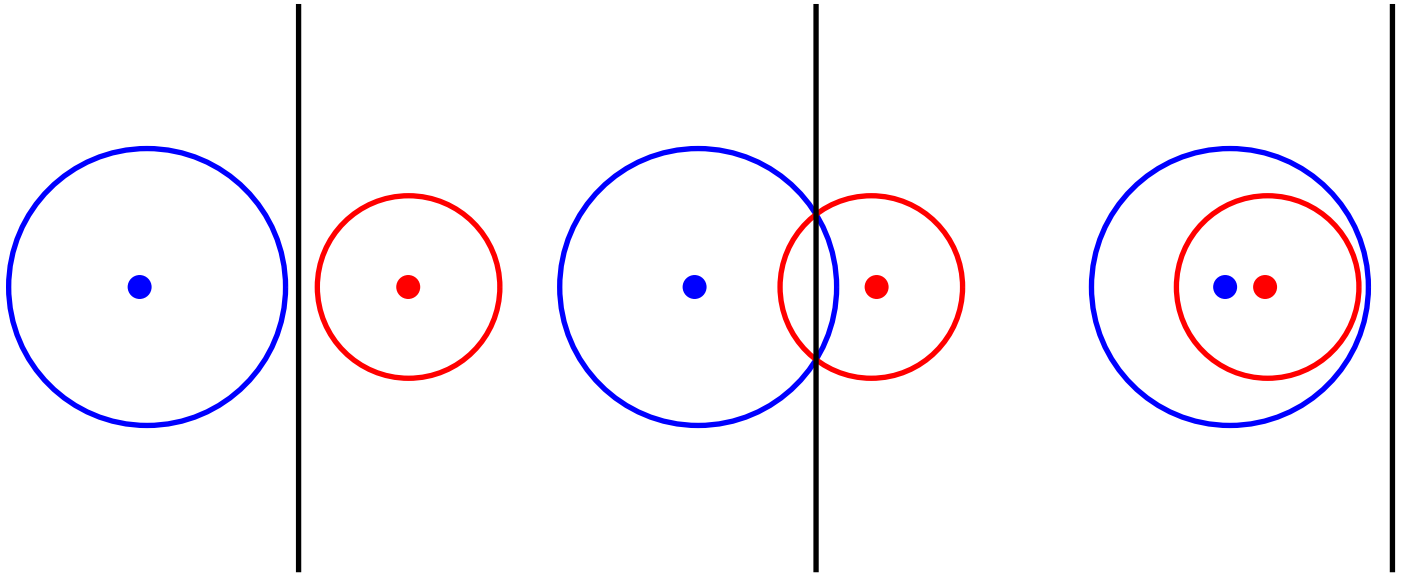


Tangents and secants through p
$\pi(p, b) = pt^2 = \overline{pm} \cdot \overline{pm'} = \overline{pn} \cdot \overline{pn'}$

Radical Hyperplane

The locus of point $\in \mathbb{R}^d$ with same power distance
to balls $b_1(p_1, r_1)$ and $b_2(p_2, r_2)$ is a hyperplane of $\mathbb{R}^d$

$$\begin{aligned}
\pi(x, b_1) = \pi(x, b_2) &\iff (x - p_1)^2 - r_1^2 = (x - p_2)^2 - r_2^2 \\
&\iff -2p_1x + p_1^2 - r_1^2 = -2p_2x + p_2^2 - r_2^2 \\
&\iff 2(p_2 - p_1)x + (p_1^2 - r_1^2) - (p_2^2 - r_2^2) = 0
\end{aligned}$$

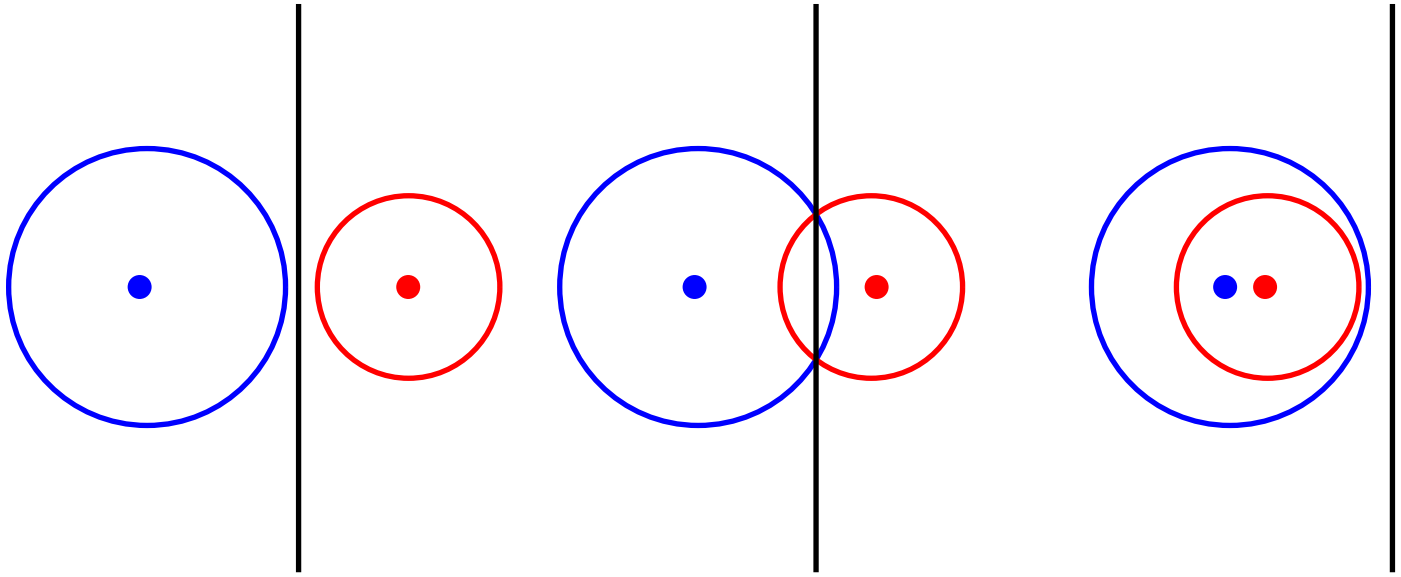


A point in h_{12} is the center of a ball orthogonal to b_1 and b_2

Radical Hyperplane

The locus of point $\in \mathbb{R}^d$ with same power distance
to balls $b_1(p_1, r_1)$ and $b_2(p_2, r_2)$ is a hyperplane of $\mathbb{R}^d$

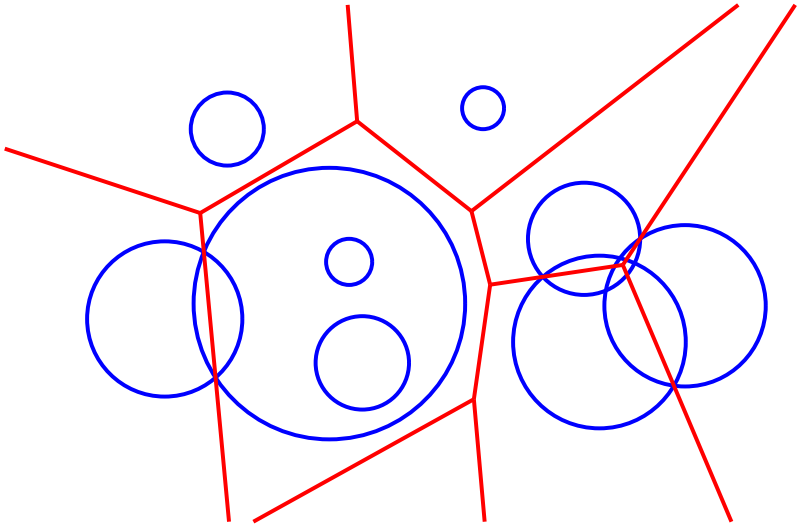
$$\begin{aligned}
\pi(x, b_1) = \pi(x, b_2) &\iff (x - p_1)^2 - r_1^2 = (x - p_2)^2 - r_2^2 \\
&\iff -2p_1x + p_1^2 - r_1^2 = -2p_2x + p_2^2 - r_2^2 \\
&\iff 2(p_2 - p_1)x + (p_1^2 - r_1^2) - (p_2^2 - r_2^2) = 0
\end{aligned}$$



A point in h_{12} is the center of a ball orthogonal to b_1 and b_2

Power Diagrams

also named Laguerre diagrams or weighted Voronoi diagrams



Sites : n balls $B = \{b_i(p_i, r_i), i = 1, \dots n\}$

Power distance: $\pi(x, b_i) = (x - p_i)^2 - r_i^2$

Power Diagram: Vor(B)
One cell $V(b_i)$ for each site
$V(b_i) = \{x : \pi(x, b_i) \leq \pi(x, b_j). \forall j \neq i\}$

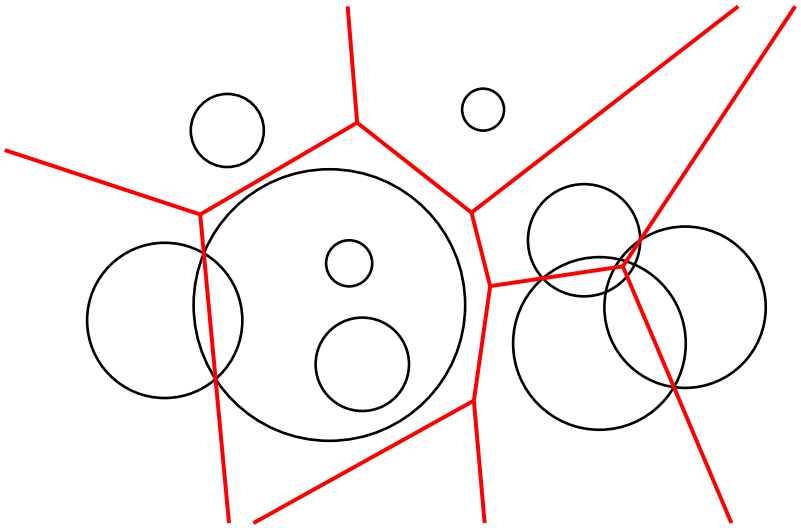
- Each cell is a polytope
- $V(b_i)$ may be empty
- p_i may not belong to $V(b_i)$

Weighted Delaunay triangulations

$B = \{b_i(p_i, r_i)\}$ a set of balls

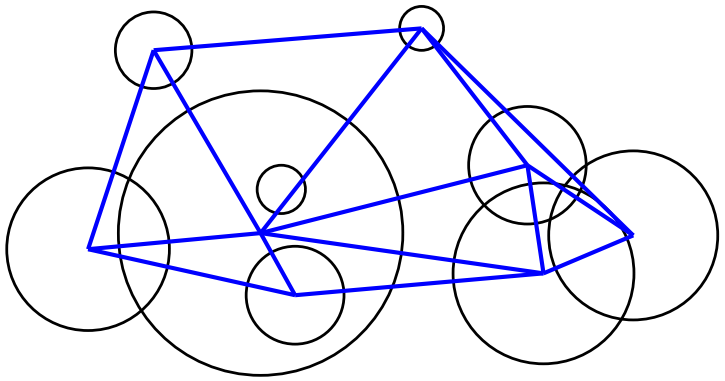
Del(B) = nerve of Vor(B):

$B_\tau = \{b_i(p_i, r_i), i = 0, \dots k\}\} \subset B$
$B_\tau \in \mathsf{Del}(B) \iff \bigcap_{b_i \in B_\tau} V(b_i) \neq \emptyset$

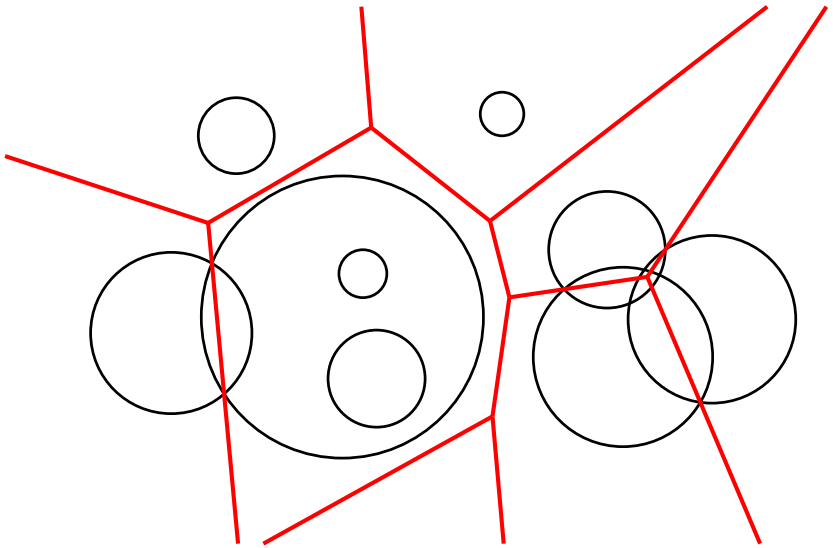


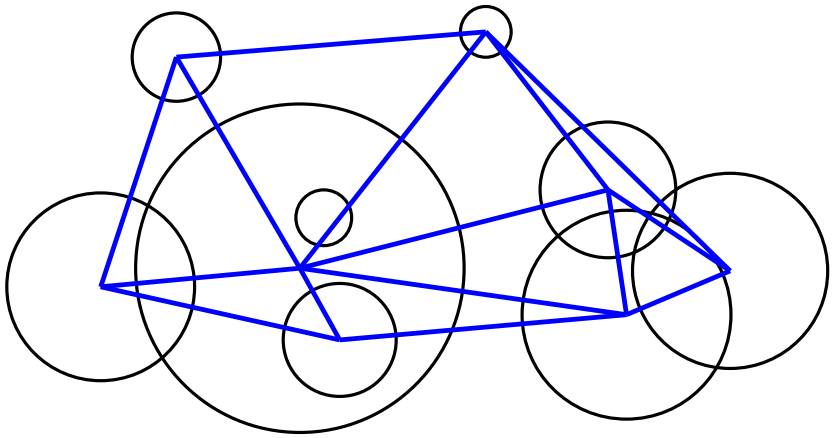
To be proved (next slides):

under a general position condition on B,
$B_\tau \longrightarrow \tau = \mathsf{conv}(\{p_i, i = 0 \dots k\})$
embeds Del(B) as a triangulation in $\mathbb{R}^d$,
called the weighted Delaunay triangulation



Characteristic property of weighted Delaunay complexes





$$\tau \in \mathsf{Del}(B) \iff \bigcap_{b_i \in B_\tau} V(b_i) \neq \emptyset$$

$$\iff \exists\, x \in \mathbb{R}^d \text{ s.t. } \forall b_i, b_j \in B_\tau,\ b_l \in B \setminus B_\tau$$
$$\pi(x, b_i) = \pi(x, b_j) < \pi(x, b_l)$$

$$\iff \exists \text{ ball } b(x, \omega) \text{ s.t. } \forall b_i \in B_\tau,\ b_l \in B \setminus B_\tau$$
$$0 = \pi(b, b_i) < \pi(b, b_l)$$

The space of spheres

$b(p, r)$ ball of $\mathbb{R}^d$

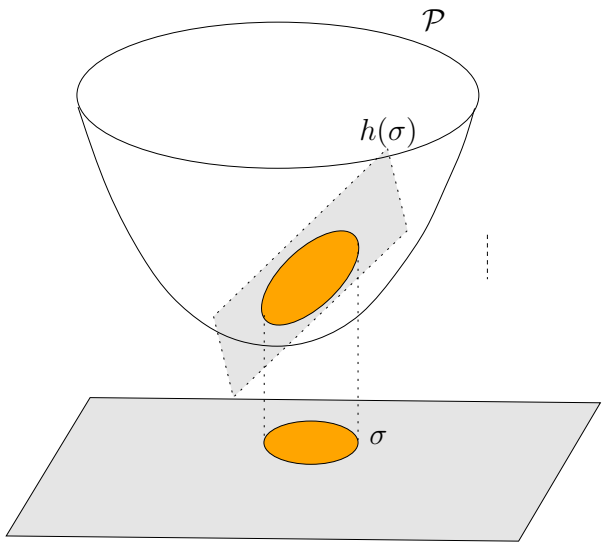
$\rightarrow$ point $\phi(b) \in \mathbb{R}^{d+1}$

$\phi(b) = (p, s = p^2 - r^2)$

$\rightarrow$ polar hyperplane $h_b = \phi(b)^* \in \mathbb{R}^{d+1}$

$\mathcal{P} = \{\hat{x} \in \mathbb{R}^{d+1} : x_{d+1} = x^2\}$

$h_b = \{\hat{x} \in \mathbb{R}^{d+1} : x_{d+1} = 2p \cdot x - s\}$



- Balls will null radius are mapped onto $\mathcal{P}$

  h_p is tangent to $\mathcal{P}$ at $\phi(p)$.

- The vertical projection of $h_b \cap \mathcal{P}$ onto $x_{d+1} = 0$ is ∂b

The space of spheres

$b(p, r)$ ball of $\mathbb{R}^d$

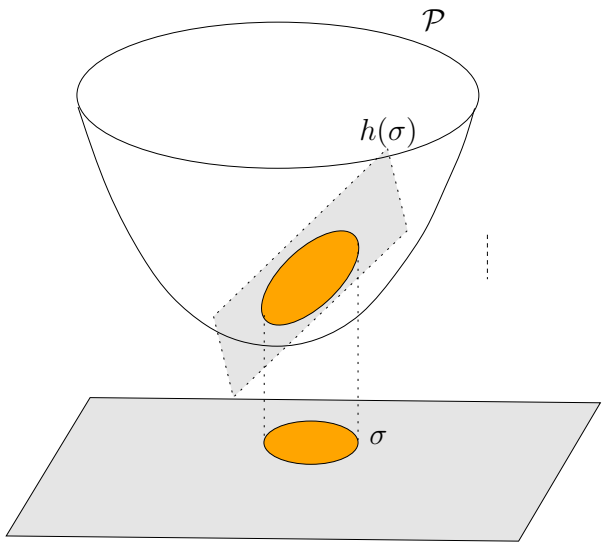
$\rightarrow$ point $\phi(b) \in \mathbb{R}^{d+1}$

$\phi(b) = (p, s = p^2 - r^2)$

$\rightarrow$ polar hyperplane $h_b = \phi(b)^* \in \mathbb{R}^{d+1}$

$\mathcal{P} = \{\hat{x} \in \mathbb{R}^{d+1} : x_{d+1} = x^2\}$

$h_b = \{\hat{x} \in \mathbb{R}^{d+1} : x_{d+1} = 2p \cdot x - s\}$



- Balls will null radius are mapped onto $\mathcal{P}$

  h_p is tangent to $\mathcal{P}$ at $\phi(p)$.

- The vertical projection of $h_b \cap \mathcal{P}$ onto $x_{d+1} = 0$ is ∂b

The space of spheres

$b(p, r)$ ball of $\mathbb{R}^d$

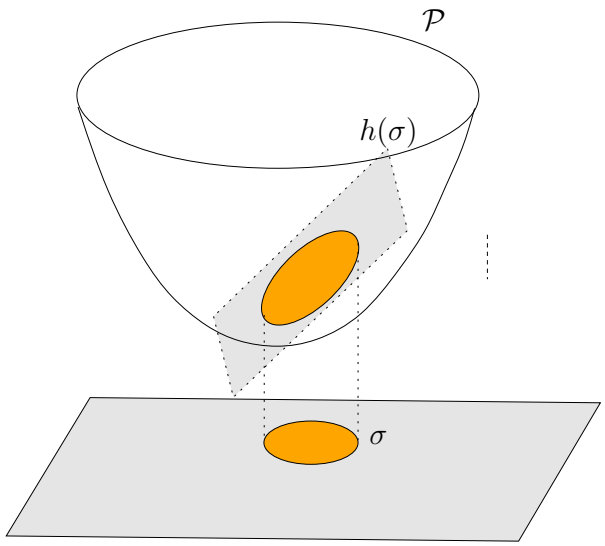
$\rightarrow$ point $\phi(b) \in \mathbb{R}^{d+1}$

$\phi(b) = (p, s = p^2 - r^2)$

$\rightarrow$ polar hyperplane $h_b = \phi(b)^* \in \mathbb{R}^{d+1}$

$\mathcal{P} = \{\hat{x} \in \mathbb{R}^{d+1} : x_{d+1} = x^2\}$

$h_b = \{\hat{x} \in \mathbb{R}^{d+1} : x_{d+1} = 2p \cdot x - s\}$



- Balls will null radius are mapped onto $\mathcal{P}$

  h_p is tangent to $\mathcal{P}$ at $\phi(p)$.

- The vertical projection of $h_b \cap \mathcal{P}$ onto $x_{d+1} = 0$ is ∂b

The space of spheres

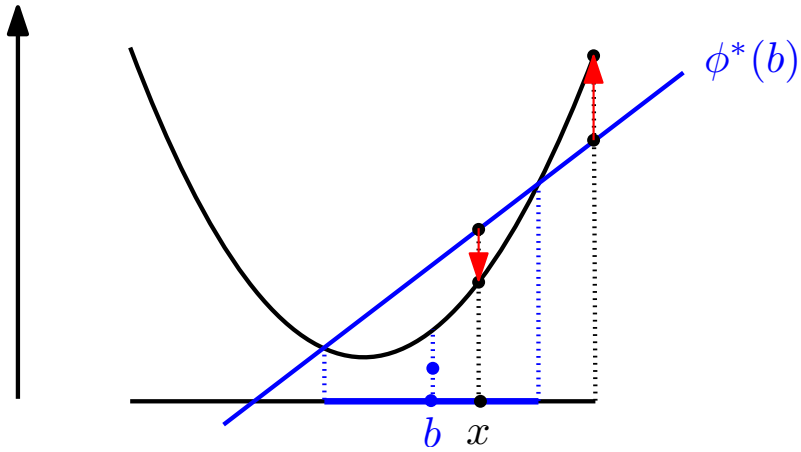
$b(p, r)$ ball of $\mathbb{R}^d$

$\rightarrow$ point $\phi(b) \in \mathbb{R}^{d+1}$

$$\phi(b) = (p, s = p^2 - r^2)$$

$\rightarrow$ polar hyperplane $h_b = \phi(b)^* \in \mathbb{R}^{d+1}$

$$h_b = \{\hat{x} \in \mathbb{R}^{d+1} : x_{d+1} = 2p \cdot x - s\}$$



- The vertical distance between $\hat{x} = (x, x^2)$ and h_b is equal to

$$x^2 - 2p \cdot x + s = \pi(x, b)$$

- The faces of the power diagram of B are the vertical projections onto $x_{d+1} = 0$ of the faces of the polytope $\mathcal{V}(B) = \bigcap_i h_b^+$ of $\mathbb{R}^{d+1}$

The space of spheres

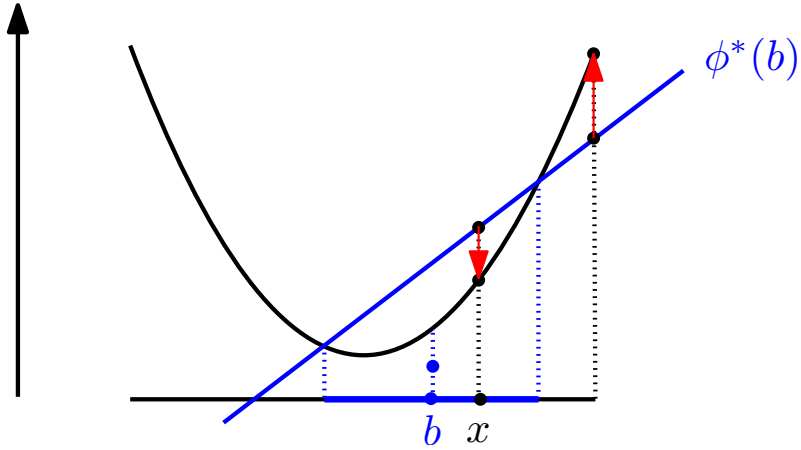
$b(p, r)$ ball of $\mathbb{R}^d$

$\rightarrow$ point $\phi(b) \in \mathbb{R}^{d+1}$

$$\phi(b) = (p, s = p^2 - r^2)$$

$\rightarrow$ polar hyperplane $h_b = \phi(b)^* \in \mathbb{R}^{d+1}$

$$h_b = \{\hat{x} \in \mathbb{R}^{d+1} : x_{d+1} = 2p \cdot x - s\}$$



- The vertical distance between $\hat{x} = (x, x^2)$ and h_b is equal to

$$x^2 - 2p \cdot x + s = \pi(x, b)$$

- The faces of the power diagram of B are the vertical projections onto $x_{d+1} = 0$ of the faces of the polytope $\mathcal{V}(B) = \bigcap_i h_b^+$ of $\mathbb{R}^{d+1}$

Weighted points in general position wrt spheres

$B = \{b_1, b_2 \dots b_n\}$ is said to be in general position wrt spheres if

$$\nexists\, x \in \mathbb{R}^d \text{ with equal power to } d+2 \text{ balls of } B$$

$P = \{p_1, ..., p_n\}$: set of centers of the balls of B

Theorem

If B is in general position wrt spheres, the natural mapping

$$f : \mathrm{vert}(\mathrm{Del}(B)) \to P$$

provides a realization of $\mathrm{Del}(B)$

$\mathrm{Del}(B)$ is a triangulation of $P' \subseteq P$ called the Delaunay triangulation of B

Weighted points in general position wrt spheres

$B = \{b_1, b_2 \dots b_n\}$ is said to be in general position wrt spheres if

$$\nexists\, x \in \mathbb{R}^d \text{ with equal power to } d+2 \text{ balls of } B$$

$P = \{p_1, ..., p_n\}$: set of centers of the balls of B

Theorem

If B is in general position wrt spheres, the natural mapping

$$f : \mathrm{vert}(\mathrm{Del}(B)) \to P$$

provides a realization of $\mathrm{Del}(B)$

$\mathrm{Del}(B)$ is a triangulation of $P' \subseteq P$ called the Delaunay triangulation of B

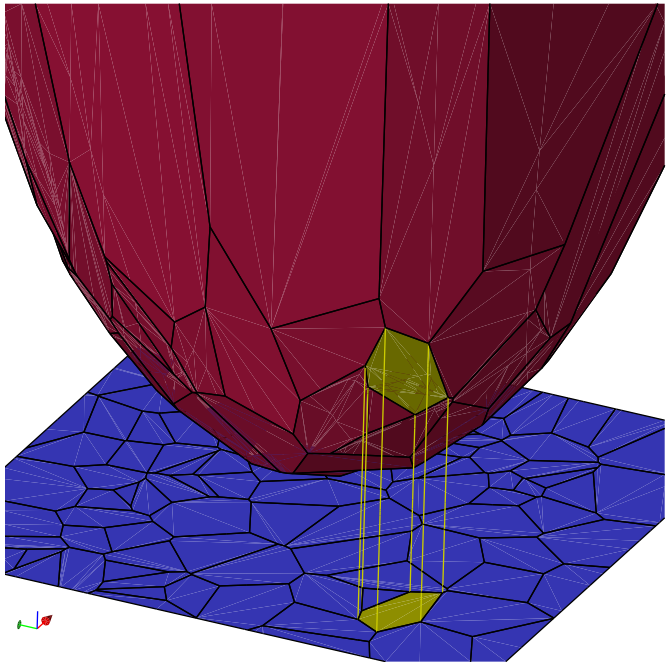
Proof of the theorem

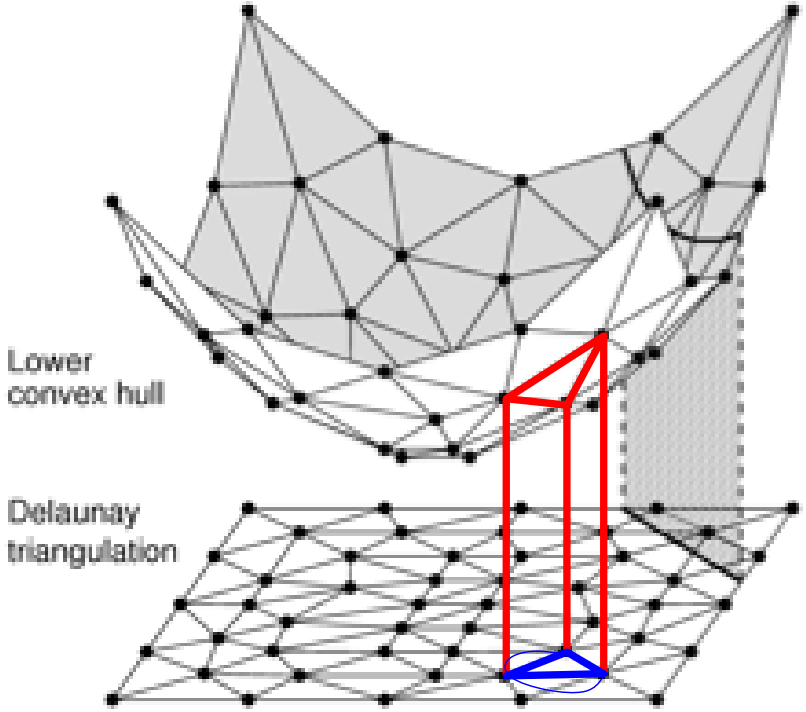
$$B_\tau \subset B, |B_\tau| = d+1,\ \tau = \mathsf{conv}(\{p_i, b_i(p_i, r_i) \in B_\tau\}),$$
$$\phi(\tau) = \mathsf{conv}(\{\phi(b_i), b_i \in B_\tau\})$$

$$\exists\ b(p,r) \text{ s.t. } h_b = \phi(b)^* = \mathsf{aff}(\{\phi(b_i), b_i \in B_\tau\})$$

$$\begin{aligned}
\phi(\tau) \in \mathcal{D}(B) \quad &= \quad \mathsf{conv}^-(\{\phi(b_i)\}) \\
&\iff \quad \forall b_i \in B_\tau, \phi(b_i) \in h_b \quad \forall b_j \notin B_\tau, \phi(b_j) \in h_b^{*+} \\
&\iff \quad \forall b_i \in B_\tau, \pi(b, b_i) = 0 \quad \forall b_j \notin B_\tau, \pi(b, b_j) > 0 \\
&\iff \quad p \in \bigcap_{b_i \in B_\tau} V(b_i) \\
&\iff \quad \tau \in \mathsf{Del}(B)
\end{aligned}$$

Duality





$$\mathcal{V}(B) = \cap_i \, \phi(b_i)^{*+}$$

$$\mathcal{D}(B) = \mathsf{conv}^{-}(\hat{P})$$

Weighted Voronoi diagrams and Delaunay triangulations, and polytopes

If B is a set of balls in general position wrt spheres :

$$\mathcal{V}(B) = h^+_{b_1} \cap \ldots \cap h^+_{b_n} \quad \overset{\text{duality}}{\longrightarrow} \quad \mathcal{D}(B) = \mathsf{conv}^-(\{\phi(b_1), \ldots, \phi(b_n)\})$$

$$\uparrow \qquad\qquad\qquad\qquad\qquad\qquad \downarrow$$

Voronoi Diagram of B $\quad \overset{\text{nerve}}{\longrightarrow} \quad$ Delaunay Complex of B

Complexity and algorithm for weighted VD and DT

Number of faces $= \Theta\left(n^{\lfloor \frac{d+1}{2} \rfloor}\right)$ (Upper Bound Th.)

Construction can be done in time $\Theta\left(n \log n + n^{\lfloor \frac{d+1}{2} \rfloor}\right)$ (Convex hull)

Main predicate

$$\texttt{power\_test}(b_0, \ldots, b_{d+1}) = \operatorname{sign} \begin{vmatrix} 1 & \ldots & 1 \\ p_0 & \ldots & p_{d+1} \\ p_0^2 - r_0^2 & \ldots & p_{d+1}^2 - r_{d+1}^2 \end{vmatrix}$$

Complexity and algorithm for weighted VD and DT

Number of faces $= \Theta\left(n^{\lfloor \frac{d+1}{2} \rfloor}\right)$ (Upper Bound Th.)

Construction can be done in time $\Theta\left(n \log n + n^{\lfloor \frac{d+1}{2} \rfloor}\right)$ (Convex hull)

Main predicate

$$\texttt{power\_test}(b_0, \ldots, b_{d+1}) = \operatorname{sign} \begin{vmatrix} 1 & \ldots & 1 \\ p_0 & \ldots & p_{d+1} \\ p_0^2 - r_0^2 & \ldots & p_{d+1}^2 - r_{d+1}^2 \end{vmatrix}$$

Complexity and algorithm for weighted VD and DT

Number of faces $= \Theta\left(n^{\lfloor \frac{d+1}{2} \rfloor}\right)$ (Upper Bound Th.)

Construction can be done in time $\Theta\left(n \log n + n^{\lfloor \frac{d+1}{2} \rfloor}\right)$ (Convex hull)

Main predicate

$$\texttt{power\_test}(b_0, \ldots, b_{d+1}) = \operatorname{sign} \begin{vmatrix} 1 & \ldots & 1 \\ p_0 & \ldots & p_{d+1} \\ p_0^2 - r_0^2 & \ldots & p_{d+1}^2 - r_{d+1}^2 \end{vmatrix}$$

Power diagrams are maximization diagrams

Cell of b_i in the power diagram $\mathrm{Vor}(B)$

$$\begin{aligned} V(b_i) &= \{x \in \mathbb{R}^d : \pi(x, b_i) \leq \pi(x, b_j). \forall j \neq i\} \\ &= \{x \in \mathbb{R}^d : 2p_i x - s_i = \max_{j \in [1,\dots n]} \{2p_j x - s_j\}\} \end{aligned}$$

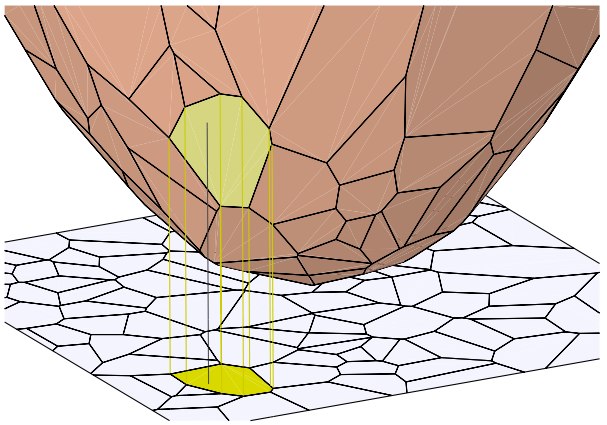
$\mathrm{Vor}(B)$ is the maximization diagram of the set of affine functions

$$\{f_i(x) = 2p_i x - s_i, i = 1, \dots, n\}$$

Affine diagrams (regular subdivisions)

Affine diagrams are defined as the maximization diagrams of a finite set of affine functions

They are equivalently defined as the vertical projections of polyhedra intersection of a finite number of upper half-spaces of $\mathbb{R}^{d+1}$



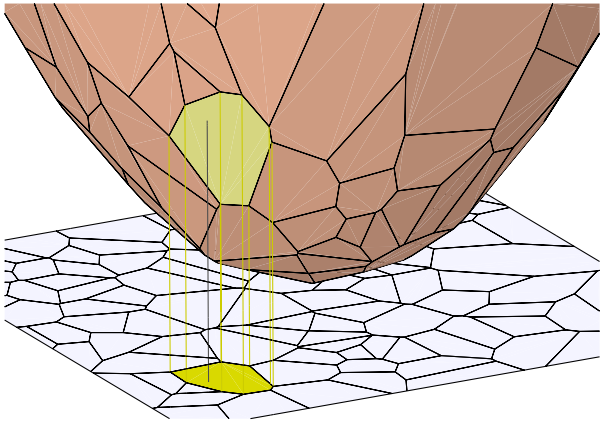
- Voronoi diagrams and power diagrams are affine diagrams.
- Any affine diagram of $\mathbb{R}^d$ is the power diagram of a set of balls.

- Delaunay and weighted Delaunay triangulations are regular triangulations
- Any regular triangulation is a weighted Delaunay triangulation

Affine diagrams (regular subdivisions)

Affine diagrams are defined as the maximization diagrams of a finite set of affine functions

They are equivalently defined as the vertical projections of polyhedra intersection of a finite number of upper half-spaces of $\mathbb{R}^{d+1}$



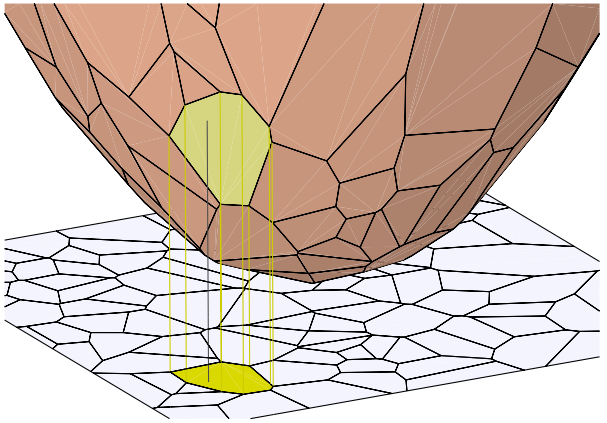
- Voronoi diagrams and power diagrams are affine diagrams.
- Any affine diagram of $\mathbb{R}^d$ is the power diagram of a set of balls.

- Delaunay and weighted Delaunay triangulations are regular triangulations
- Any regular triangulation is a weighted Delaunay triangulation

Affine diagrams (regular subdivisions)

Affine diagrams are defined as the maximization diagrams of a finite set of affine functions

They are equivalently defined as the vertical projections of polyhedra intersection of a finite number of upper half-spaces of $\mathbb{R}^{d+1}$



- Voronoi diagrams and power diagrams are affine diagrams.
- Any affine diagram of $\mathbb{R}^d$ is the power diagram of a set of balls.

- Delaunay and weighted Delaunay triangulations are regular triangulations
- Any regular triangulation is a weighted Delaunay triangulation

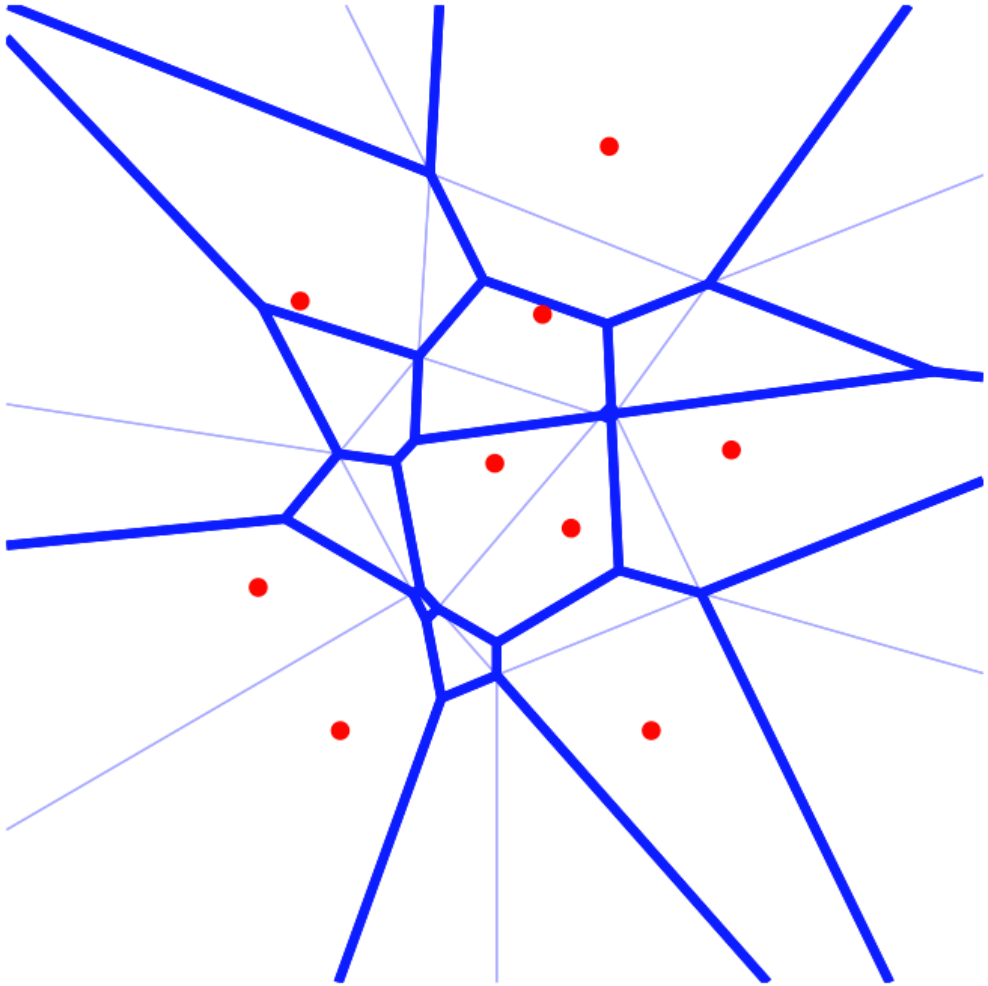
Examples of affine diagrams

1. The intersection of a power diagram with an affine subspace (Exercise)
2. A Voronoi diagram defined with a quadratic distance function

$$\|x-a\|_Q = (x-a)^t Q(x-a) \qquad Q = Q^t$$

3. k order Voronoi diagrams

k-order Voronoi Diagrams



Let P be a set of sites.
Each cell in the k-order Voronoi diagram $\mathrm{Vor}_k(P)$ is the locus of points in $\mathbb{R}^d$ that have the same subset of P as k-nearest neighbors.