MPRI — Computational Geometry and Topology

Voronoi Diagrams and Delaunay Triangulations

Steve Oudot

(steve.oudot@inria.fr)

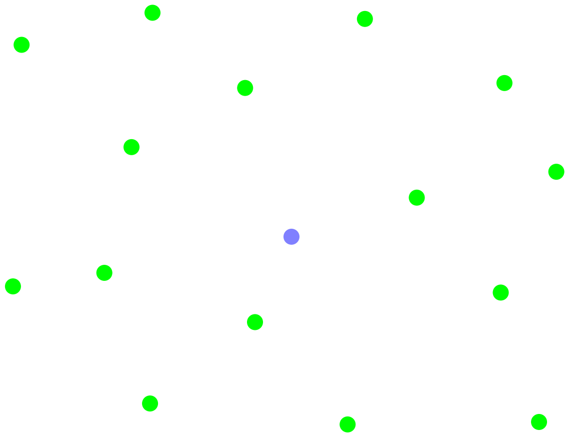
(slides courtesy of O. Devillers for the most part)

Outline

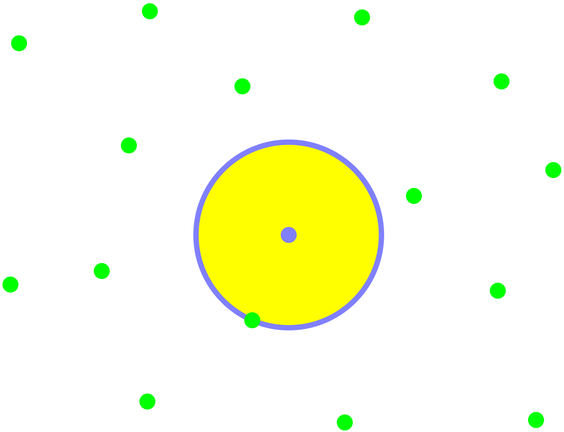
1. Definitions and examples
2. Structral properties and applications
3. Size
4. Construction
5. Generalizations

Definitions

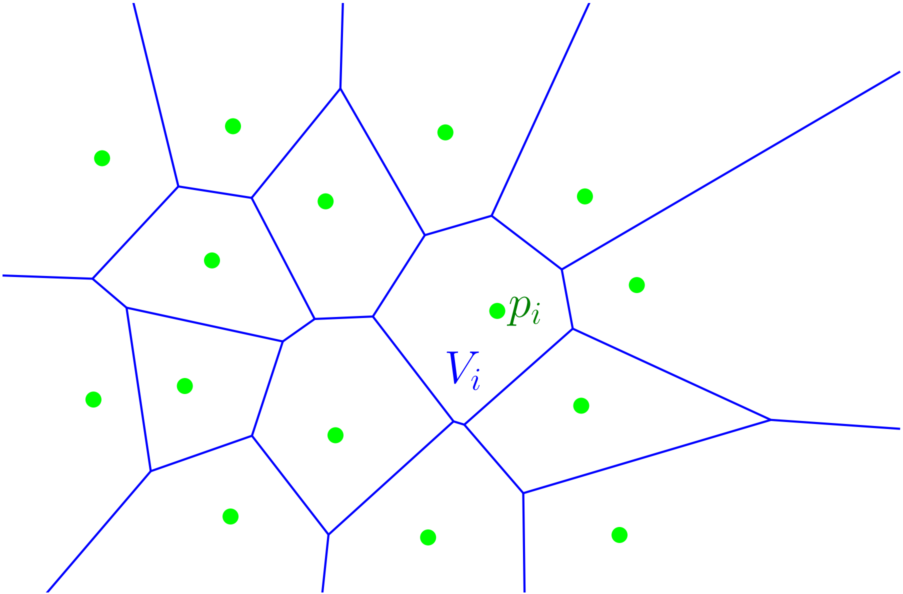
looking for nearest neighbor



looking for nearest neighbor

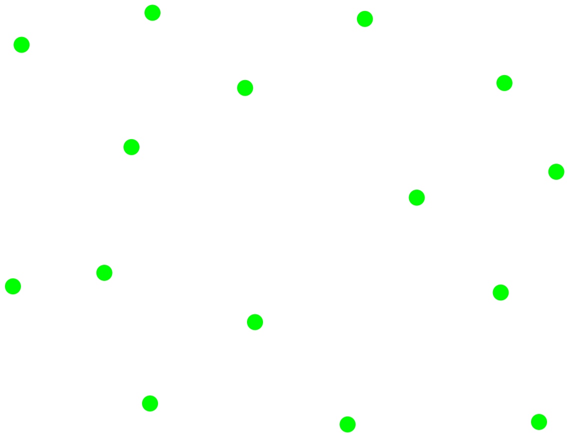


Voronoi diagram of $\{p_1, \cdots, p_n\} \subset \mathbb{R}^d$



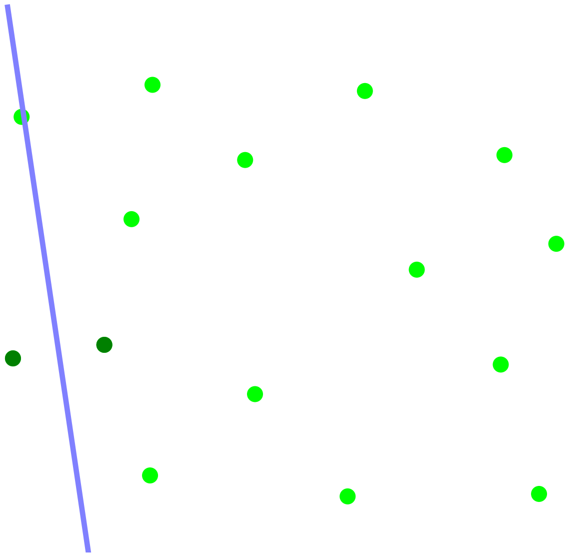
$$V_i := \{q \in \mathbb{R}^d \mid \|q - p_i\| \leq \|q - p_j\| \ \forall j\}$$

Voronoi



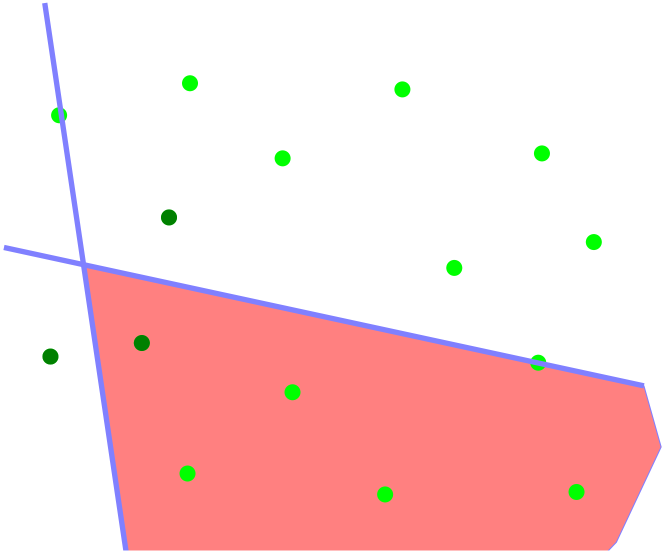
faces of the Voronoi diagram

Voronoi

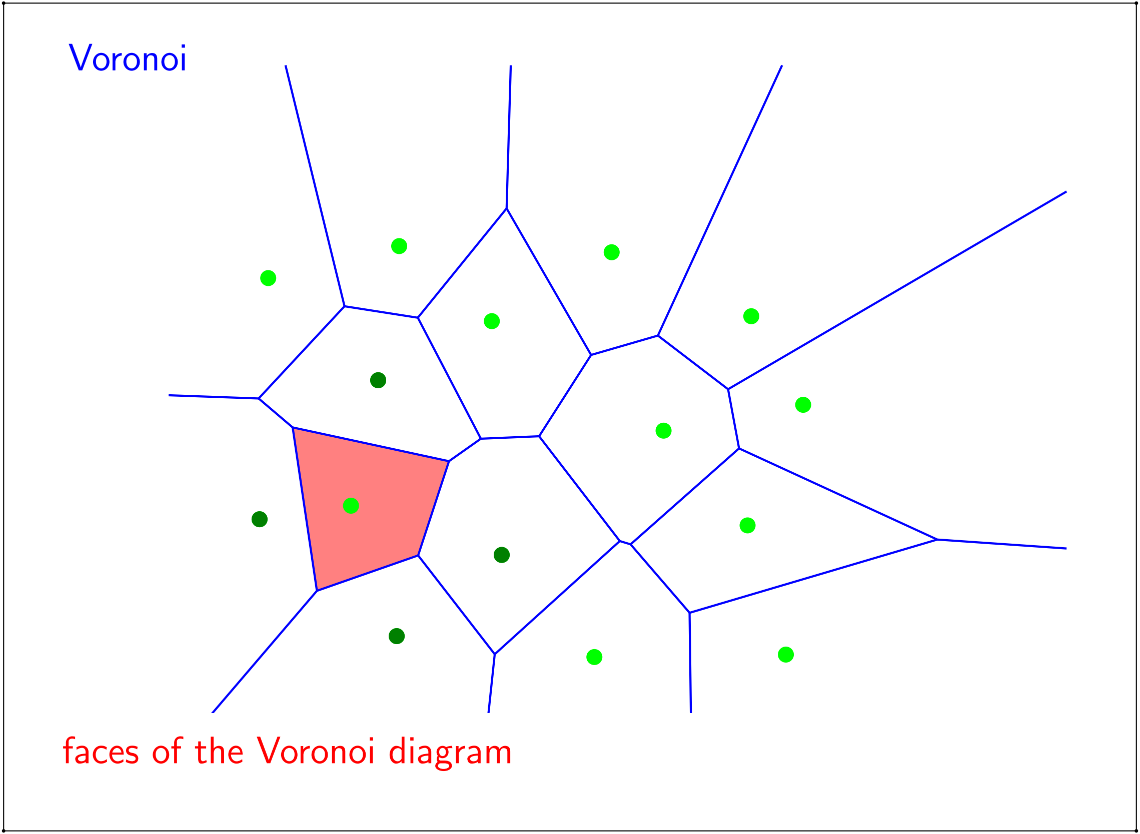


faces of the Voronoi diagram

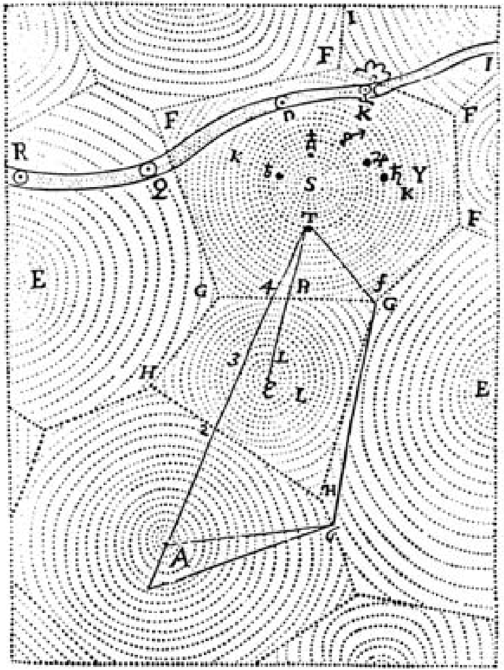
Voronoi



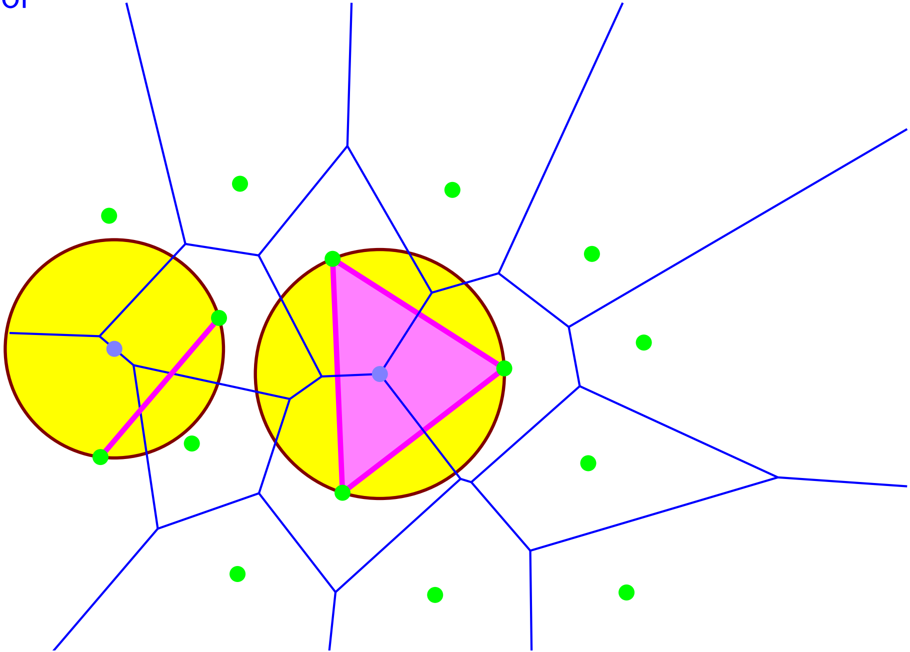
faces of the Voronoi diagram


Voronoi
faces of the Voronoi diagram

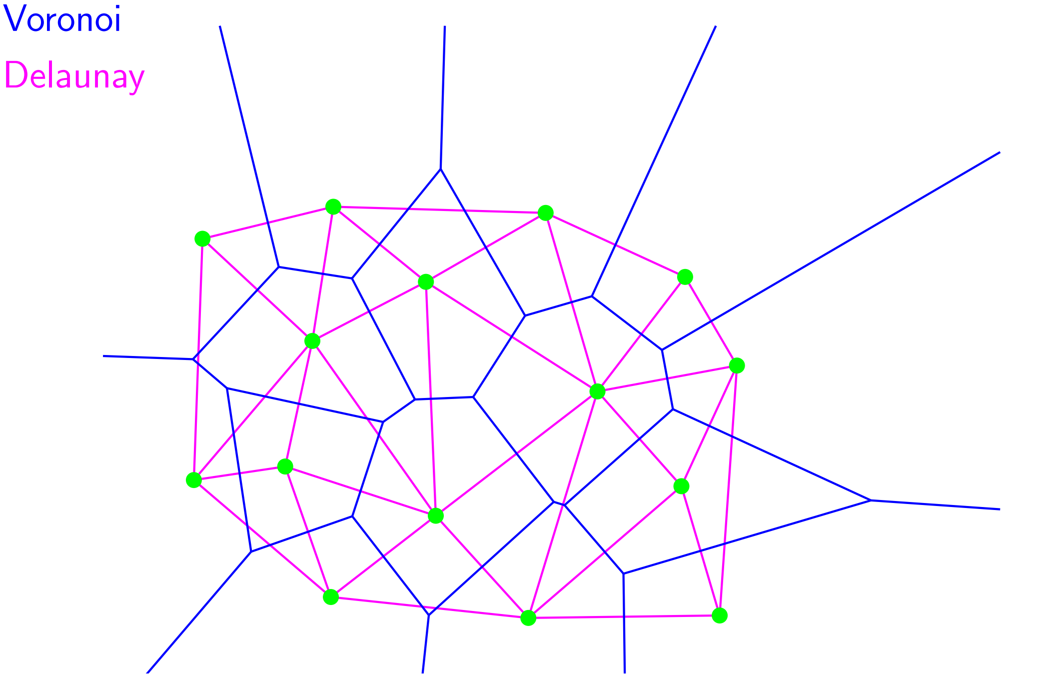
Voronoi is everywhere



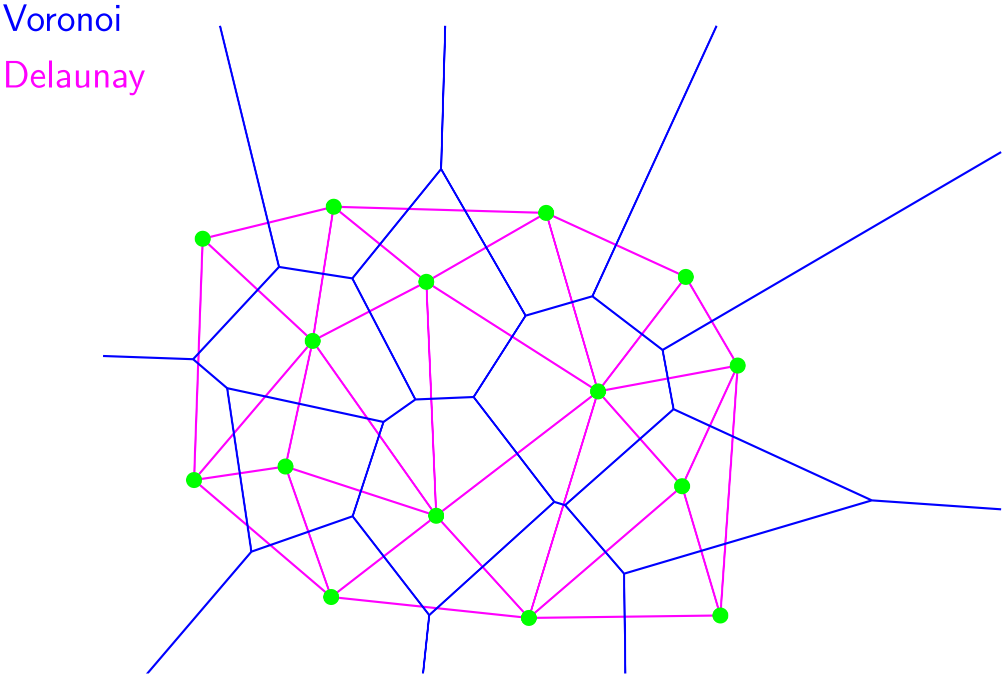
Voronoi



Empty sphere property



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



Voronoi $\leftrightarrow$ geometry Delaunay $\leftrightarrow$ connectivity (nerve)

Geometric simplicial complexes

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

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

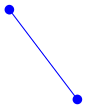
inclusion property (τ *face* of σ):

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

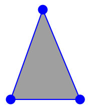
intersection property:

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

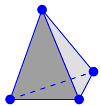
0-simplex
(vertex)



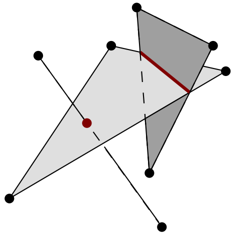
1-simplex
(edge)



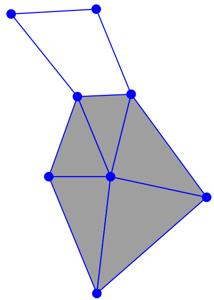
2-simplex
(triangle)



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):

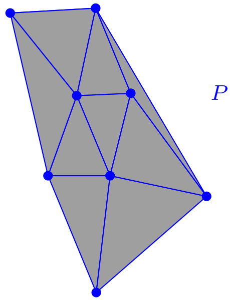
$\sigma \in K$ and $V(\tau) \subseteq V(\sigma) \Longrightarrow \tau \in K$

intersection property:

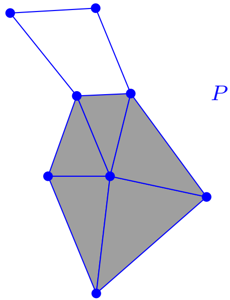
$\sigma_1, \sigma_2 \in K$ and $\sigma_1 \cap \sigma_2 \neq \emptyset \Longrightarrow$
$\sigma_1 \cap \sigma_2 \in K$ and is a face of both

triangulation of P:

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

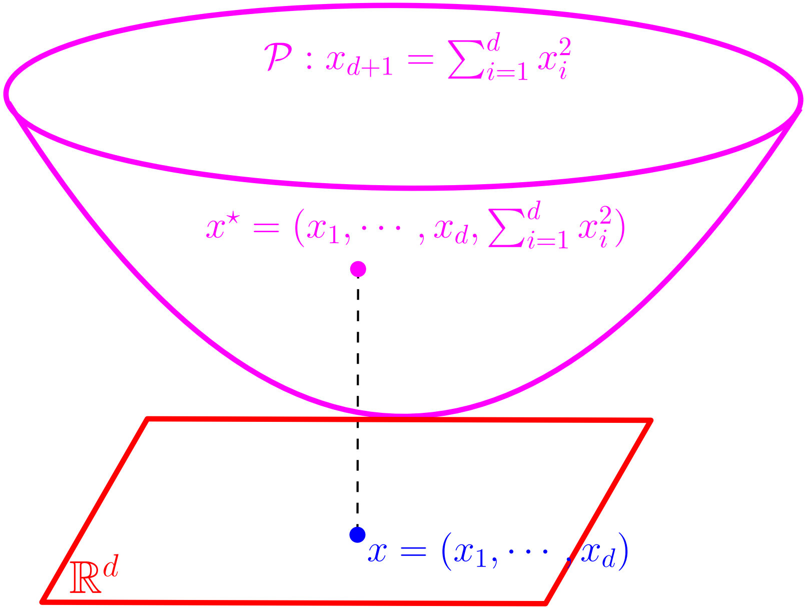


valid triangulation of P

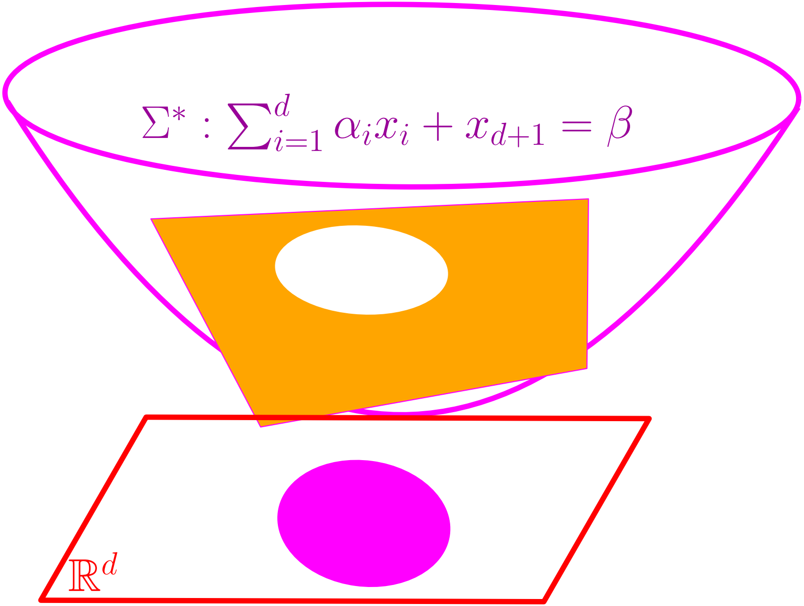


invalid triangulation of P

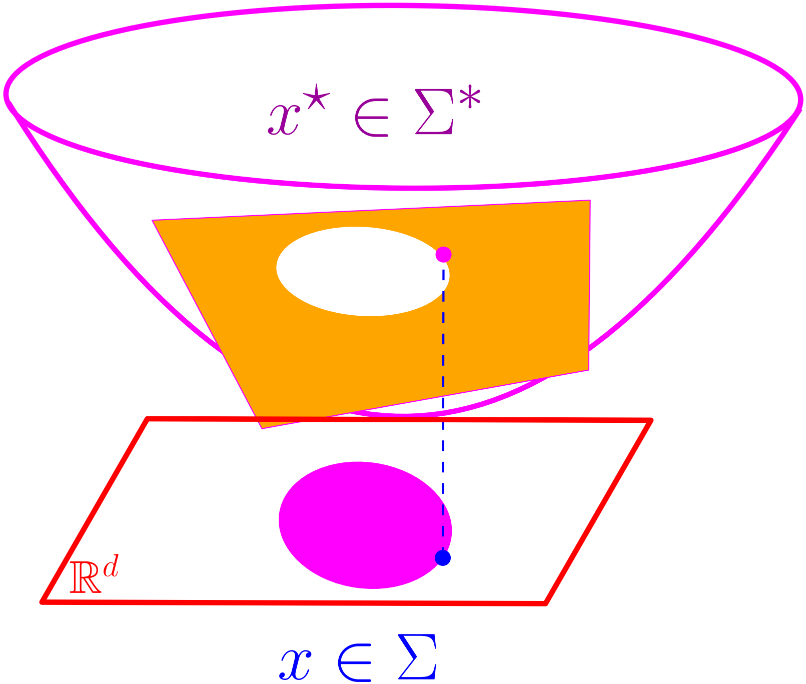
point / sphere lifting



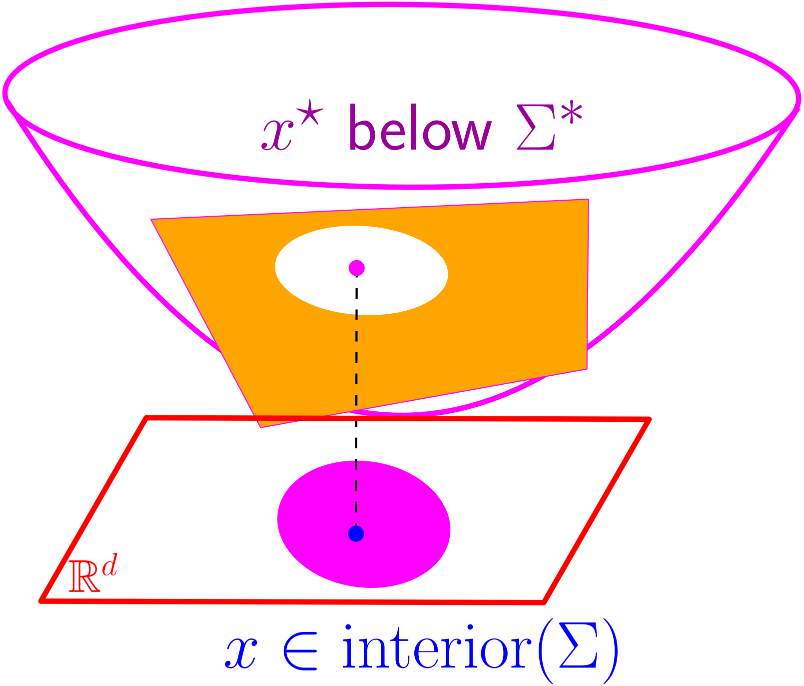
point / sphere lifting



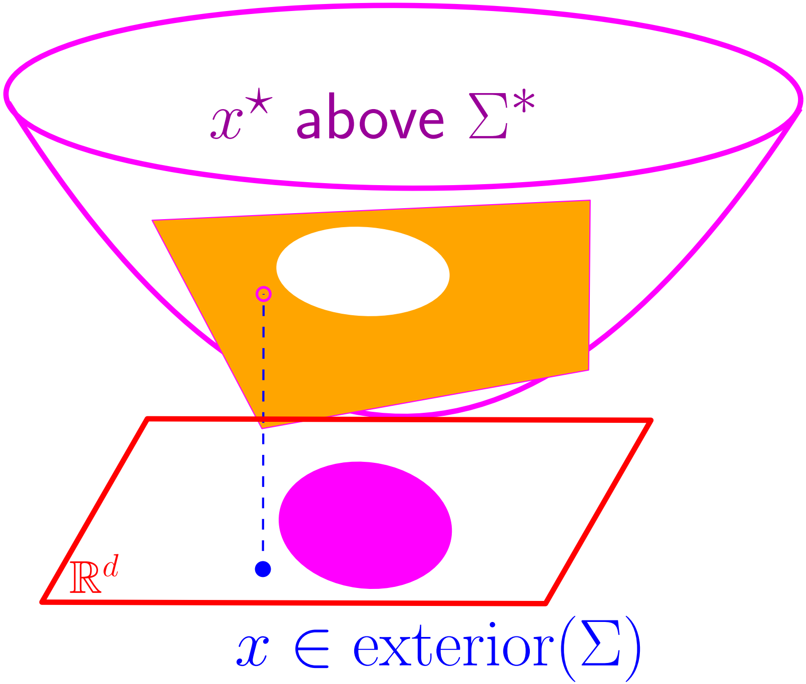
point / sphere lifting



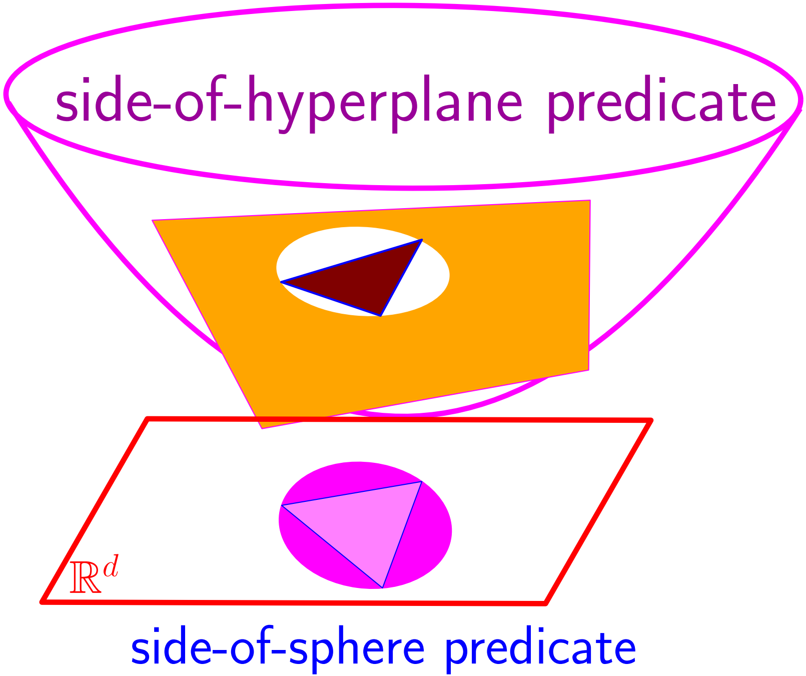
point / sphere lifting



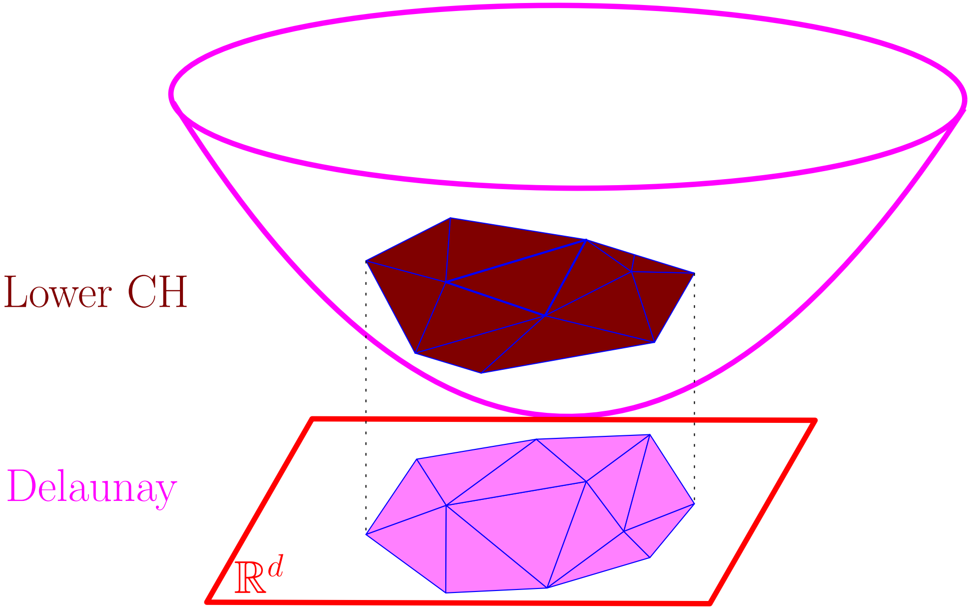
point / sphere lifting



point / sphere lifting



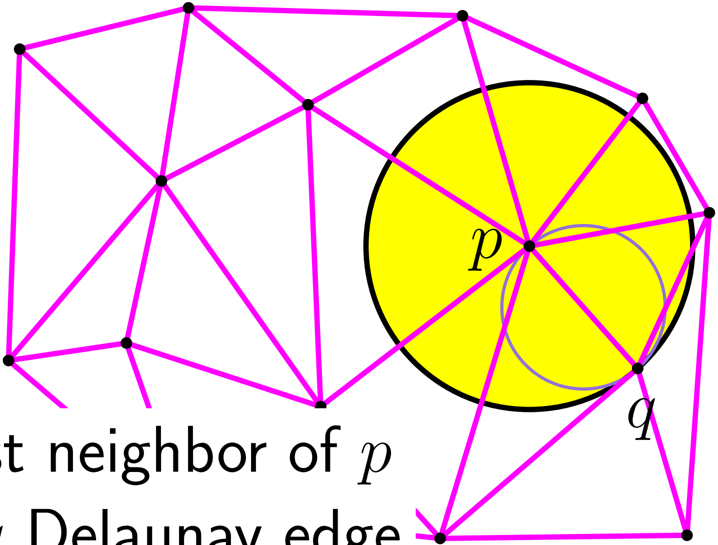
point / sphere lifting



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

Basic properties and applications

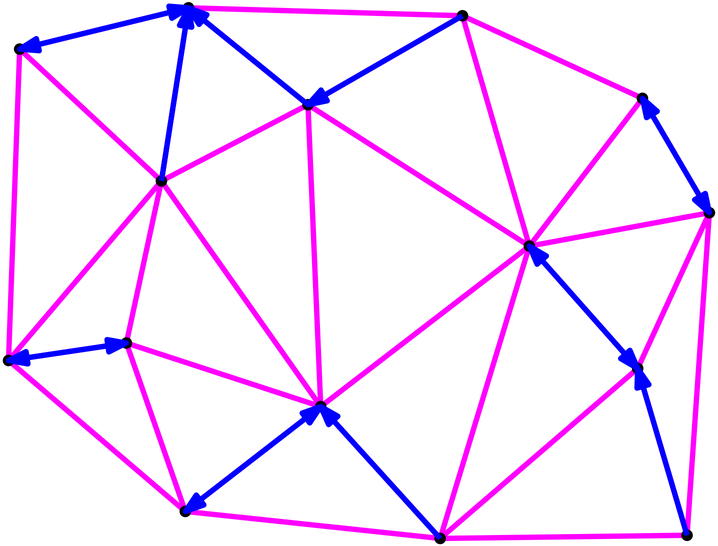
nearest neighbor graph



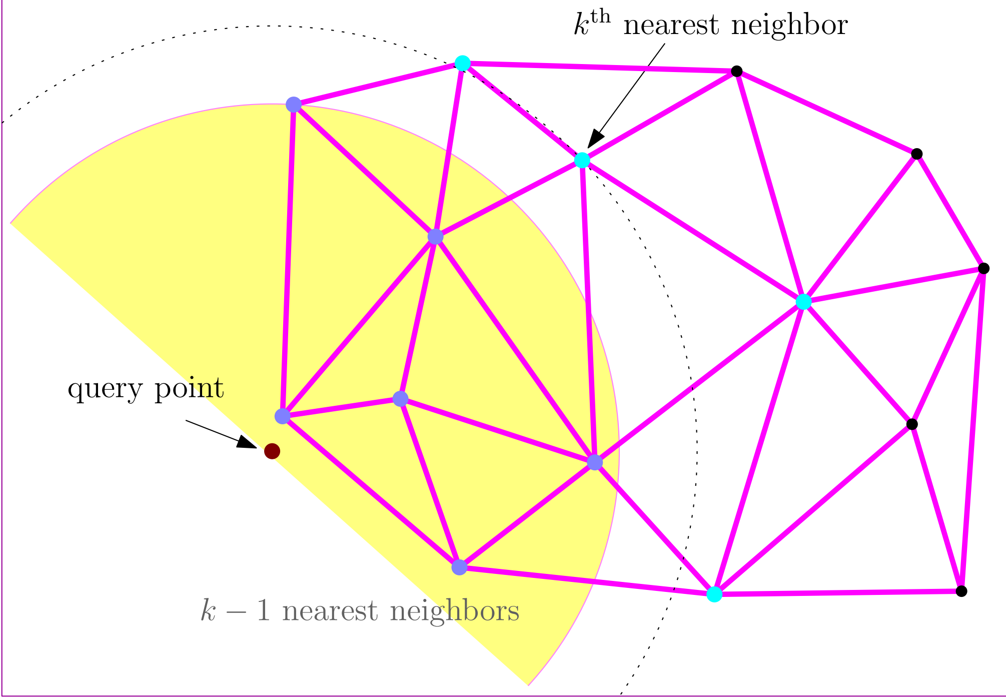
q nearest neighbor of p

$\Rightarrow pq$ Delaunay edge

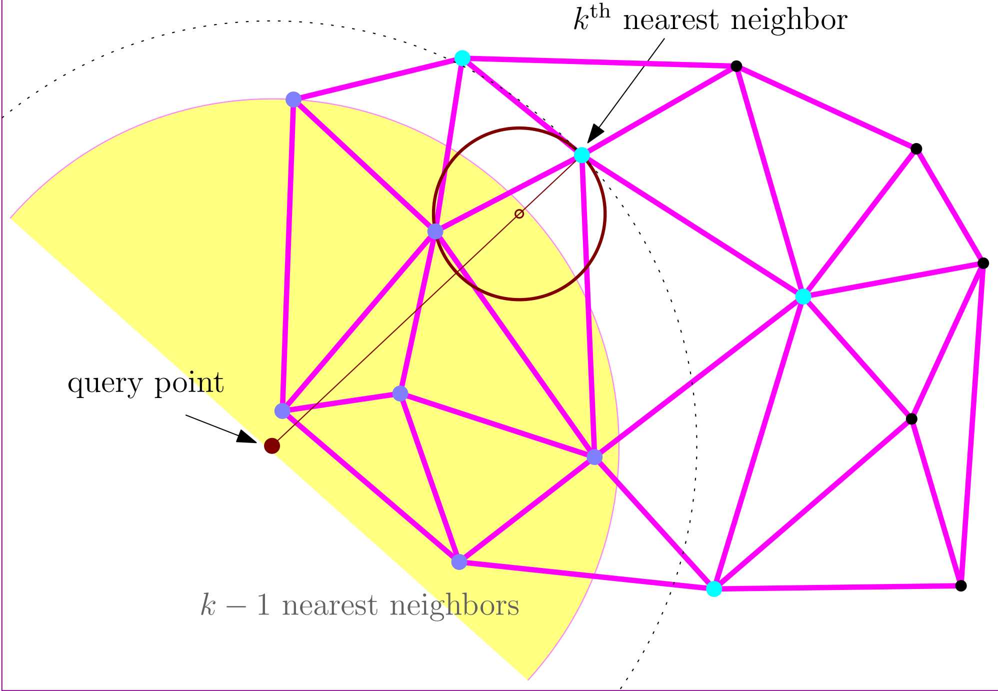
nearest neighbor graph



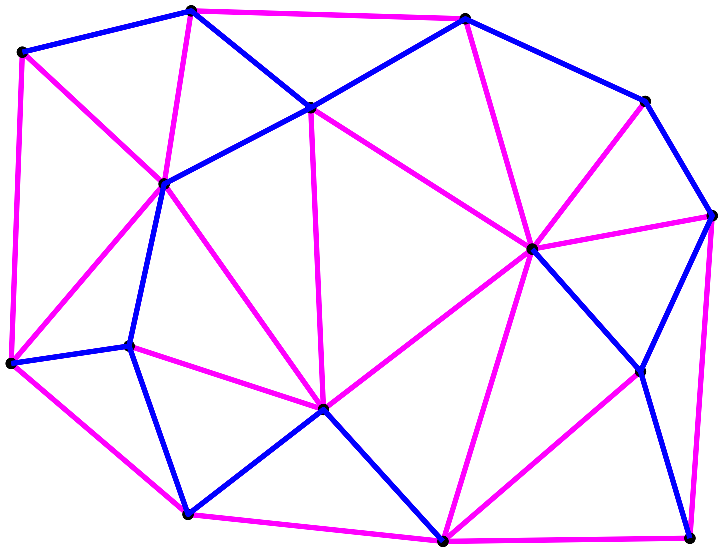
k nearest neighbors



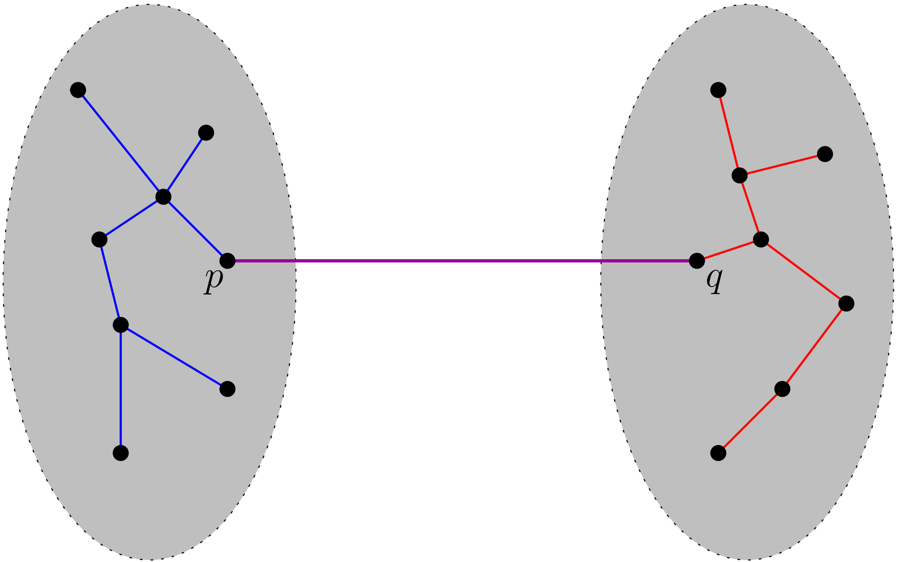
k nearest neighbors



Minimum Spanning Tree

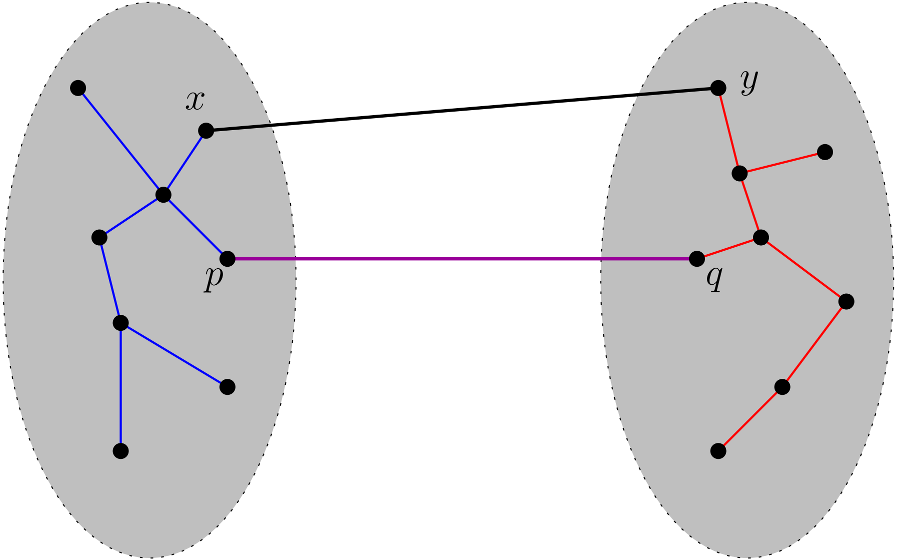


Minimum Spanning Tree

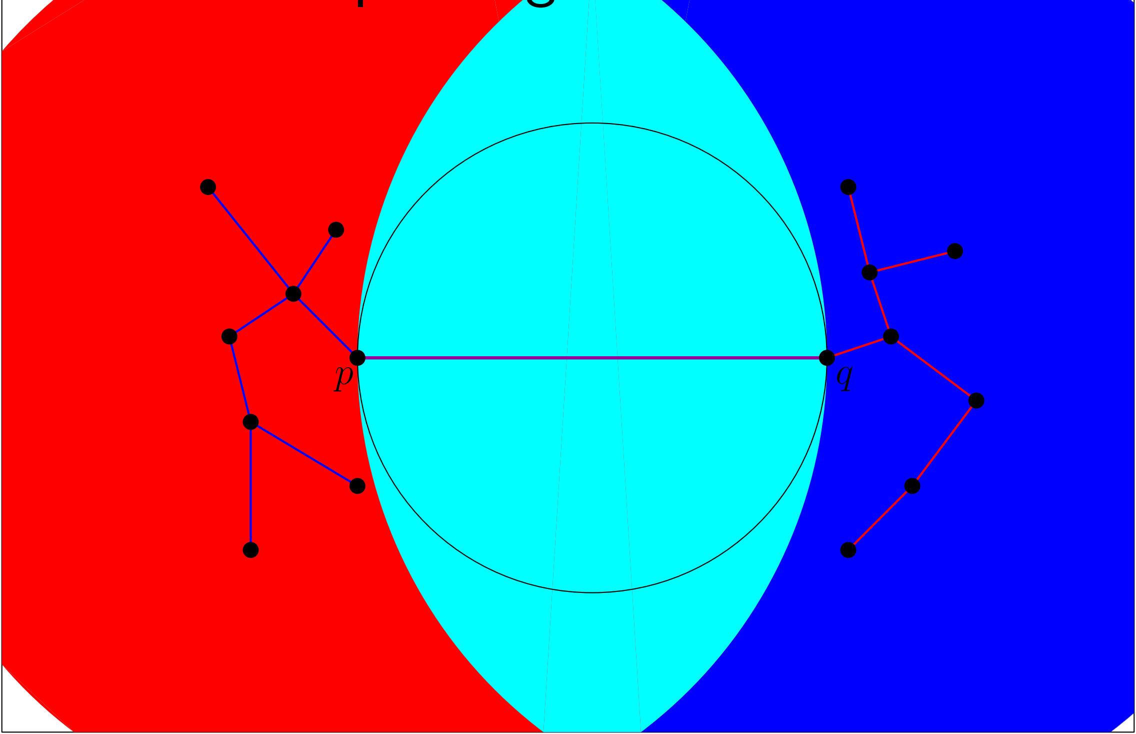


Minimum Spanning Tree

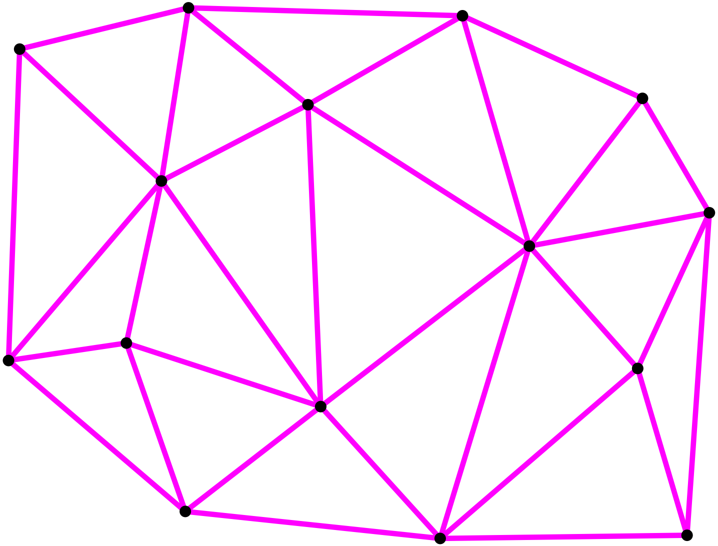
$$\forall [pq] \in A,\ \|p - q\| = \min\{\|x - y\| \mid x \in A_p, y \in A_q\}$$



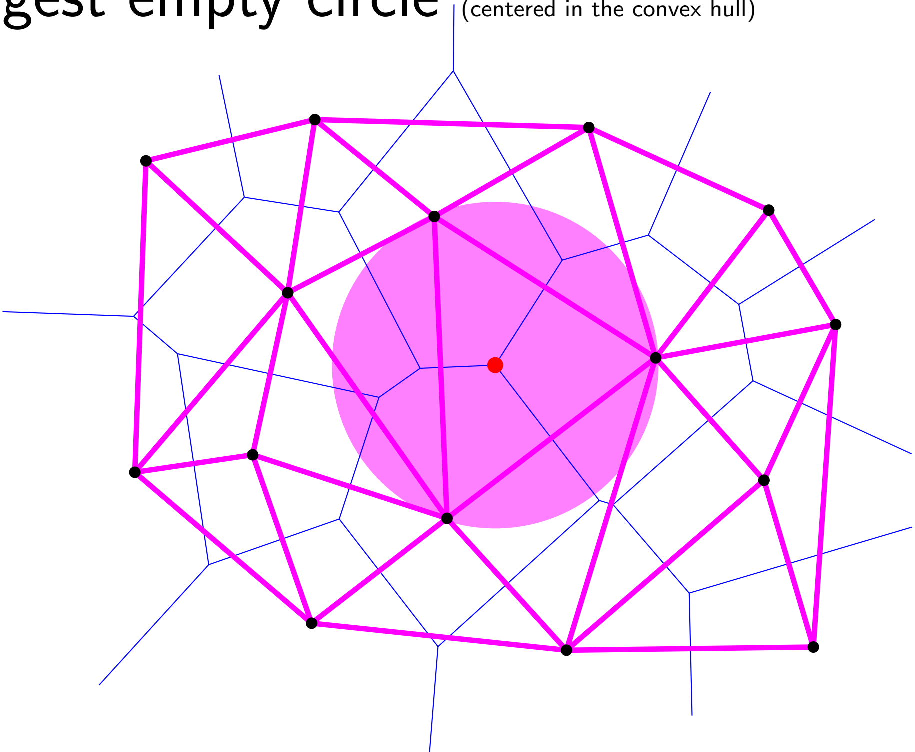
Minimum Spanning Tree



Largest empty circle (centered in the convex hull)

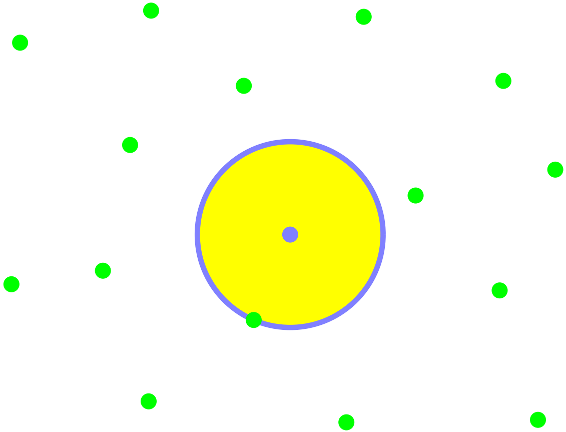


Largest empty circle (centered in the convex hull)



Applications

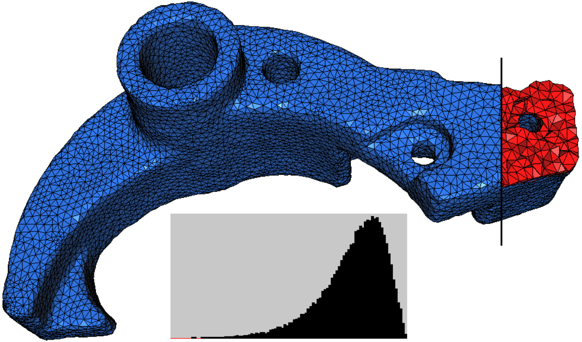
Databases, AI (NN-search)

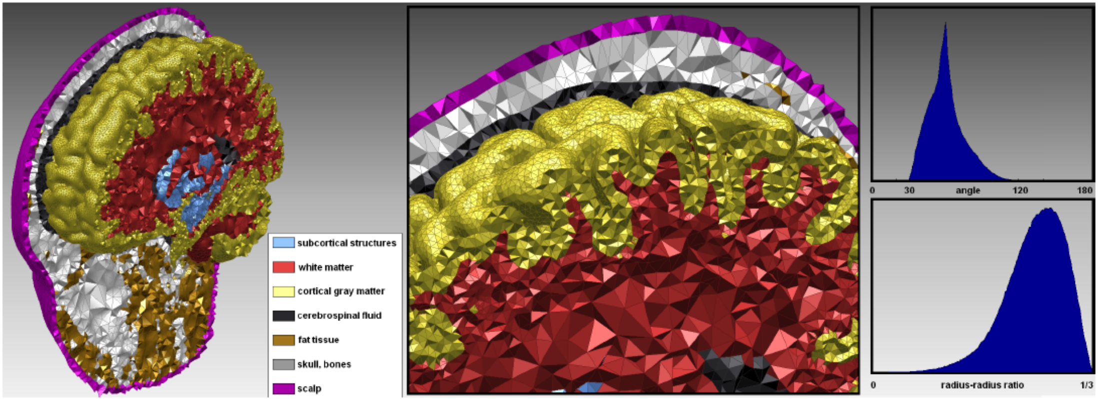


Applications

Databases, AI

Mesh generation



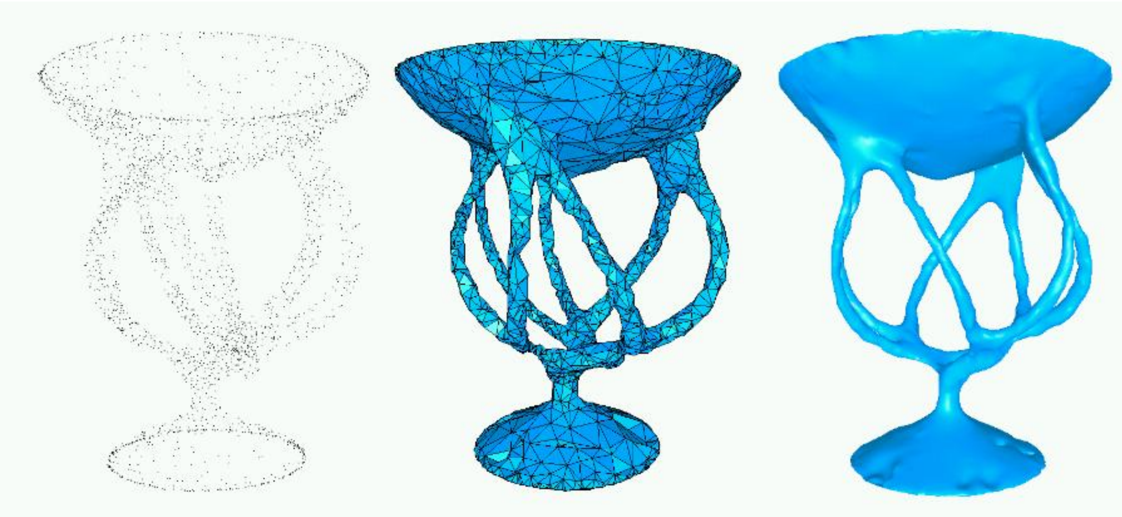


Applications

Databases, AI

Mesh generation

Reconstruction



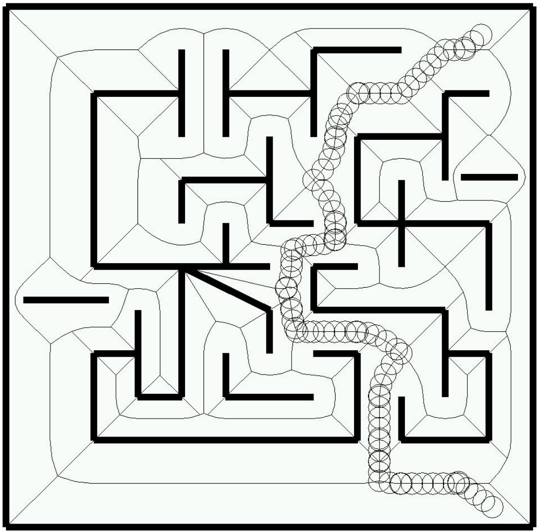
Applications

Databases, AI

Path planning

Mesh generation

Reconstruction



Applications

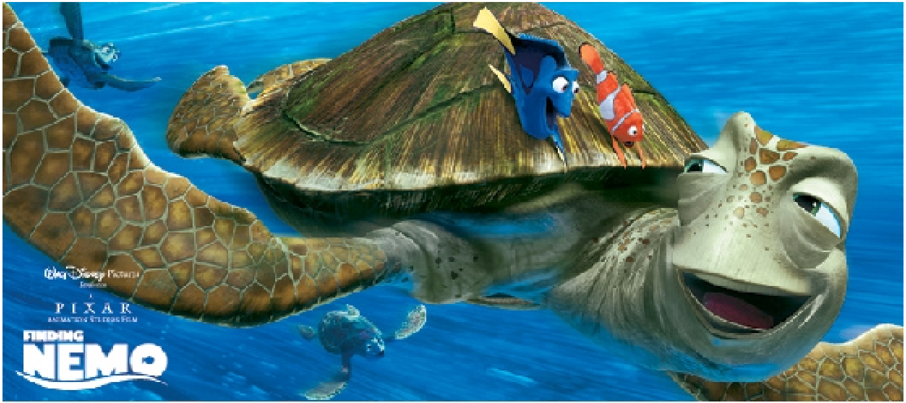
Databases, AI

Mesh generation

Reconstruction

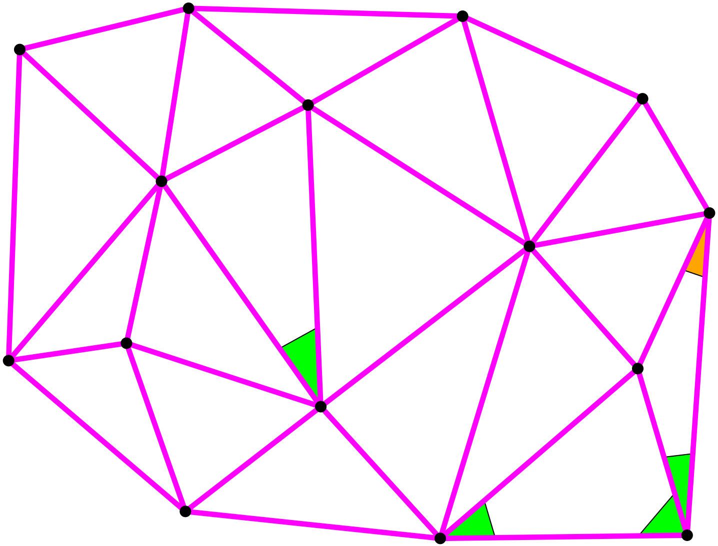
Path planning

and many more
(e.g. texture synthesis)

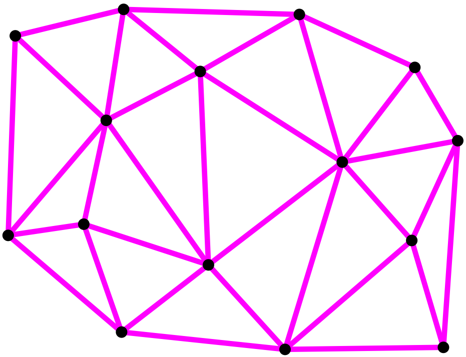


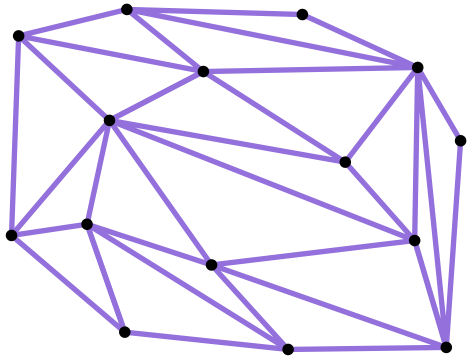
Properties specific to 2D Delaunay

Delaunay maximizes the smallest angle

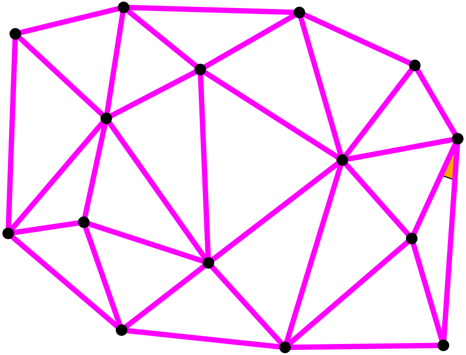


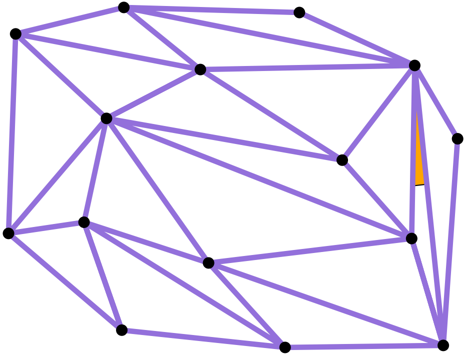
Delaunay maximizes the smallest angle



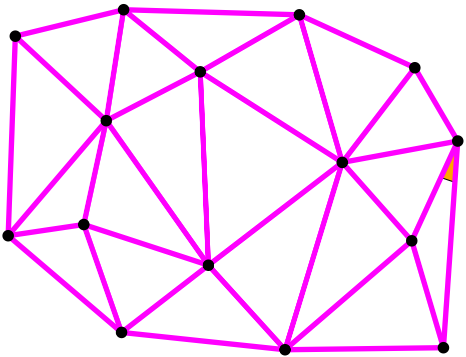


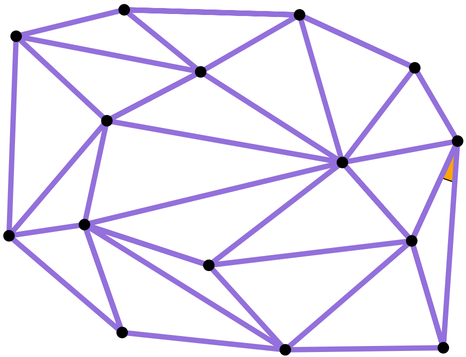
Delaunay maximizes the smallest angle



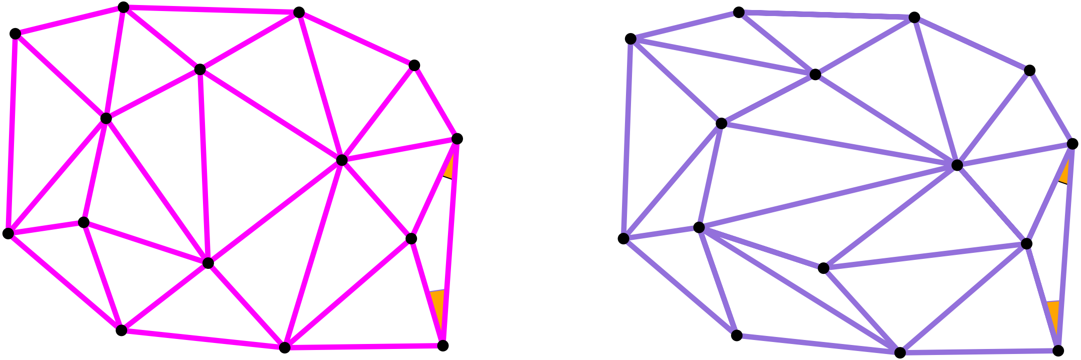


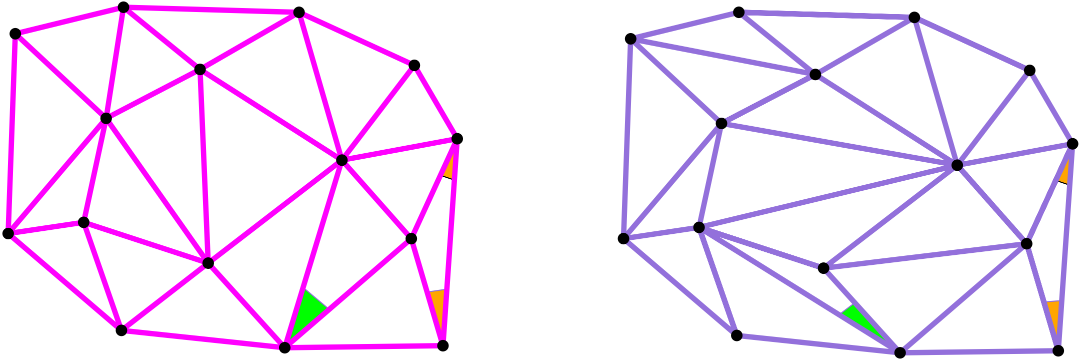
Delaunay maximizes the smallest angle





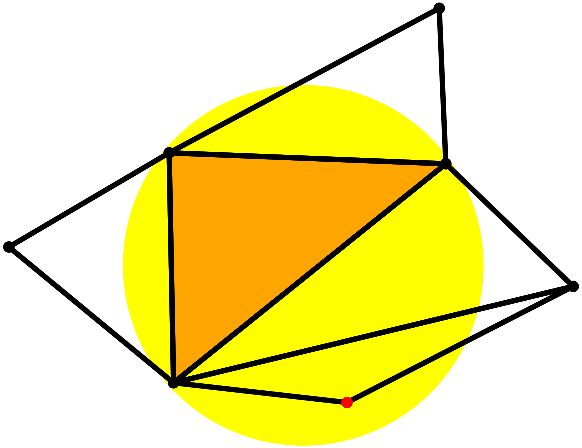
... but the converse is false





Delaunay maximizes the sequence of angles in lexicographic order

Local optimality vs global optimality



highlighted triangle is only locally Delaunay

Theorem

Locally Delaunay everywhere

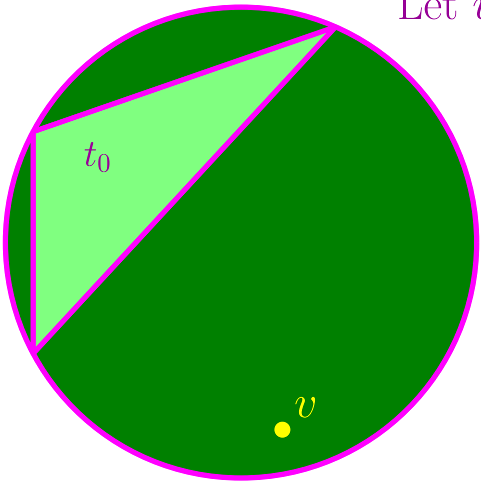
$$\Longleftrightarrow$$

Globally Delaunay

Proof:

Let t_0 be locally Delaunay, but not globally Delaunay

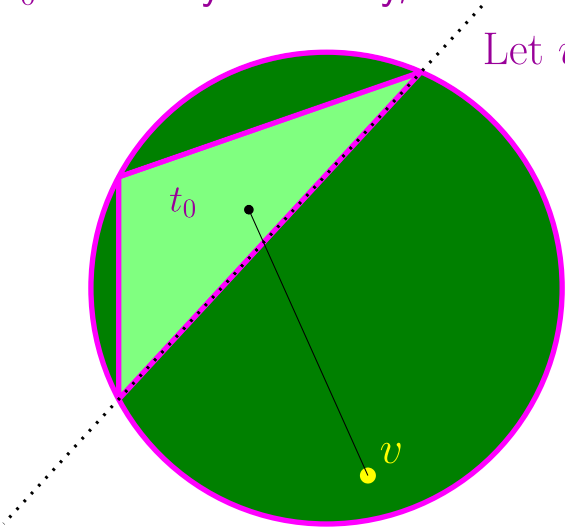
Let $v \in \mathrm{disk}(t)$ $(v \notin t)$



Proof:

Let t_0 be locally Delaunay, but not globally Delaunay

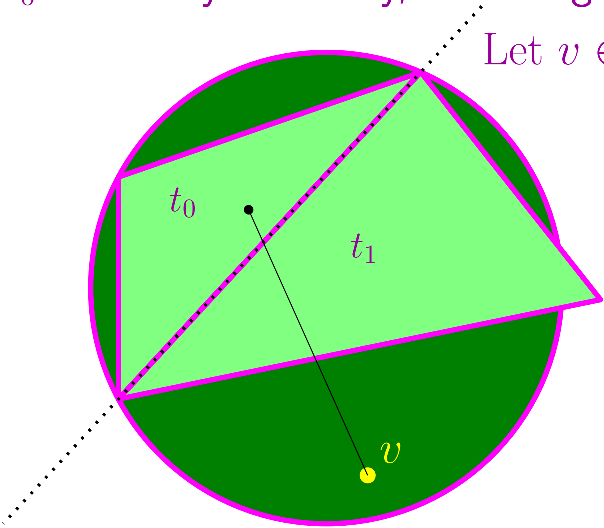
Let $v \in \mathrm{disk}(t)$ $(v \notin t)$



Proof:

Let t_0 be locally Delaunay, but not globally Delaunay

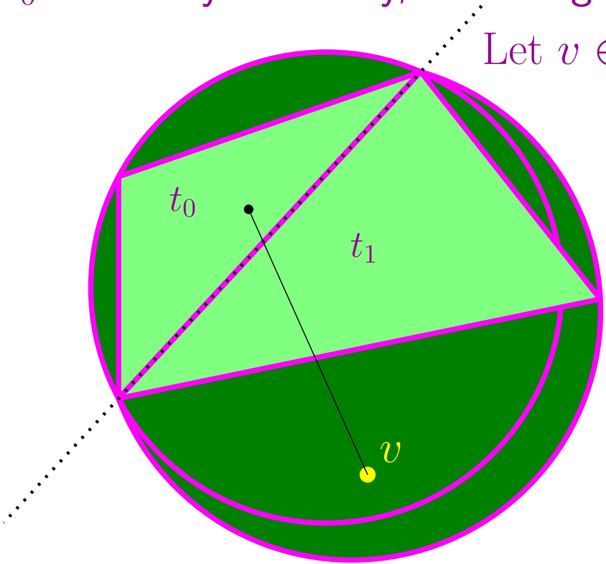
Let $v \in \text{disk}(t)$ $(v \notin t)$



Proof:

Let t_0 be locally Delaunay, but not globally Delaunay

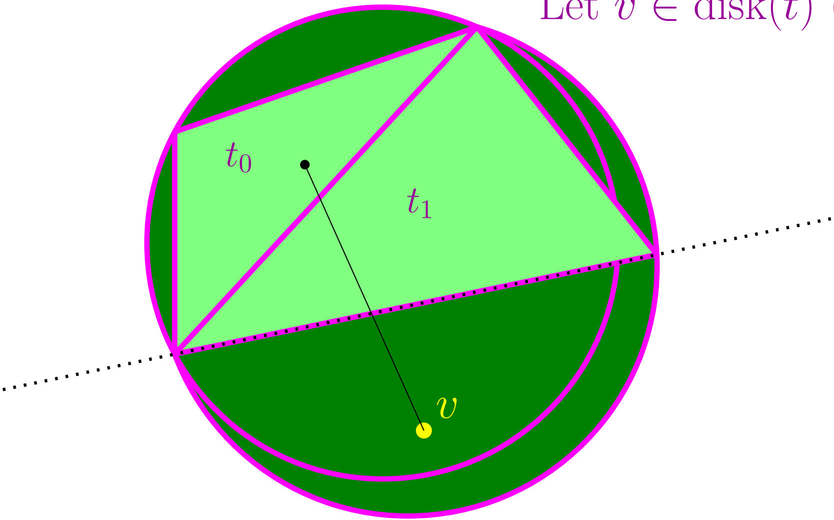
Let $v \in \mathrm{disk}(t)$ $(v \notin t)$



Proof:

Let t_0 be locally Delaunay, but not globally Delaunay

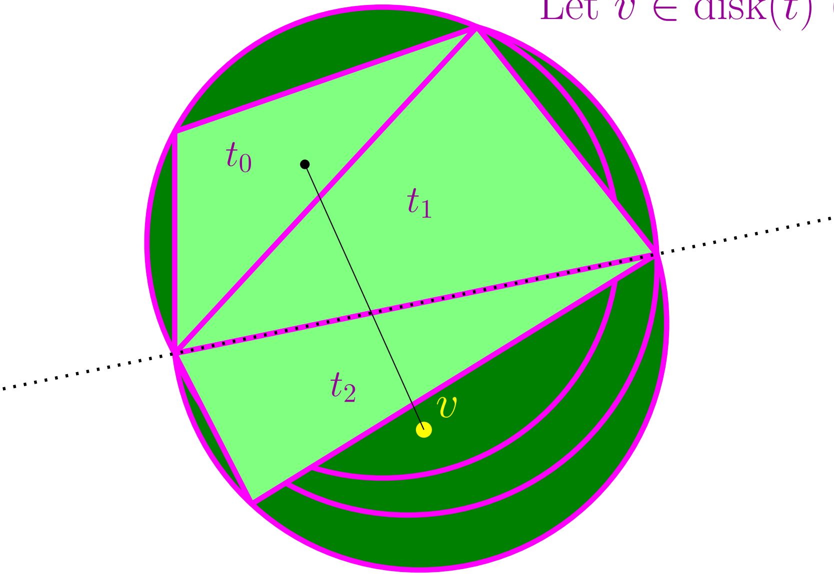
Let $v \in \mathrm{disk}(t)$ $(v \notin t)$



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

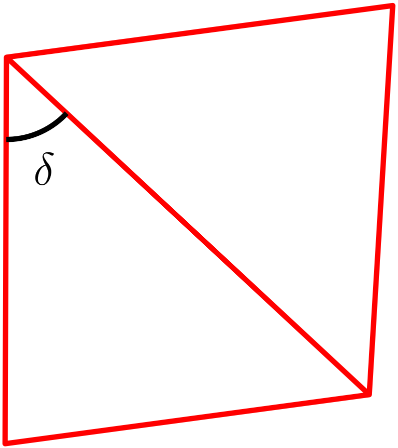
Local optimality and smallest angle

Case of 4 points

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

Local optimality and smallest angle

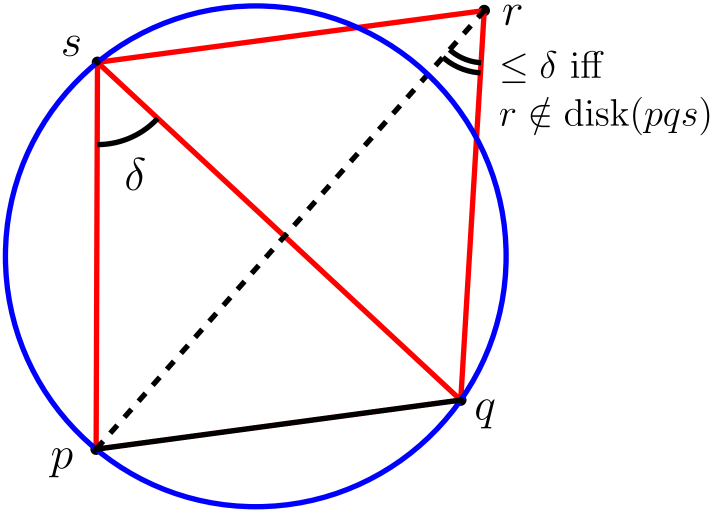
Case of 4 points



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

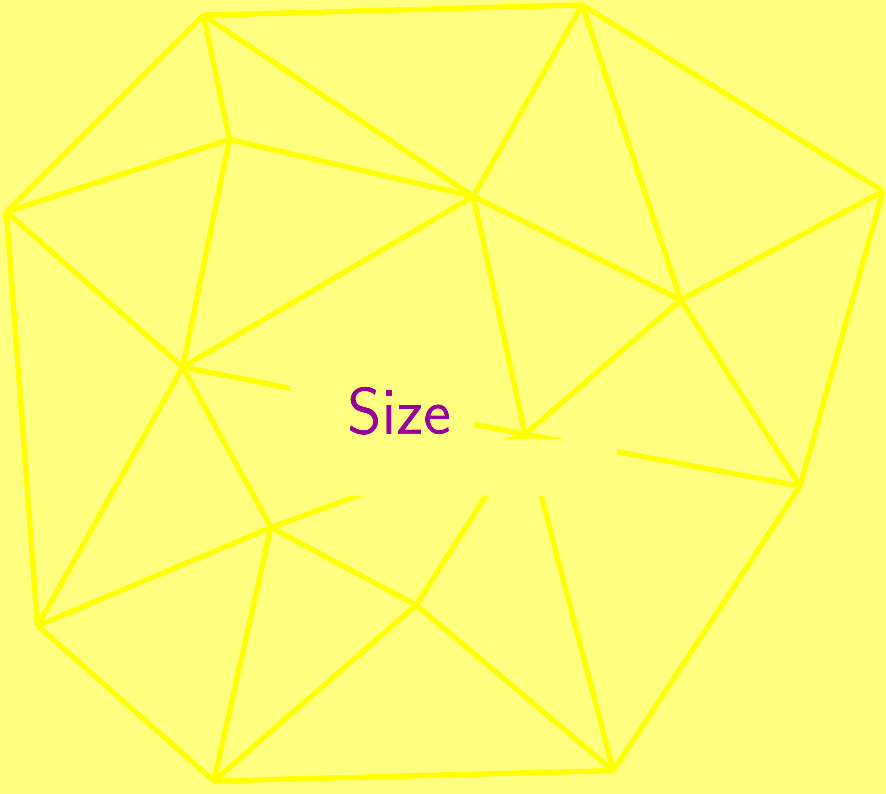
Proof: Let T triangulation

Apply flipping algorithm on T

$\rightarrow$ output is Delaunay

Each flip increases angles within quadrangle

$\rightarrow$ output has larger smallest angle


Size

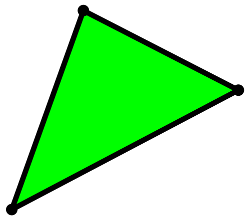
Euler formula

f: number of facets (except ∞)

e: number of edges

v: number of vertices

$$\boxed{f - e + v = 1}$$



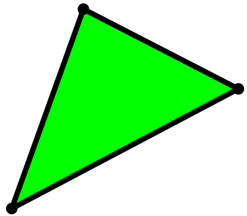
Euler formula

f: number of facets (except ∞)

e: number of edges

v: number of vertices

$$\boxed{f - e + v = 1}$$



$$\boxed{1 - 3 + 3 = 1}$$

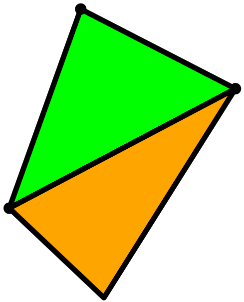
Euler formula

f: number of facets (except ∞)

e: number of edges

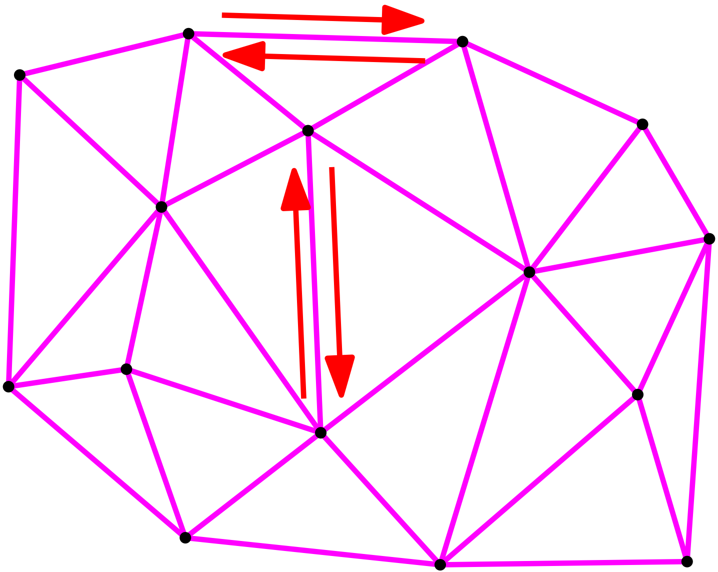
v: number of vertices

$$f - e + v = 1$$



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

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

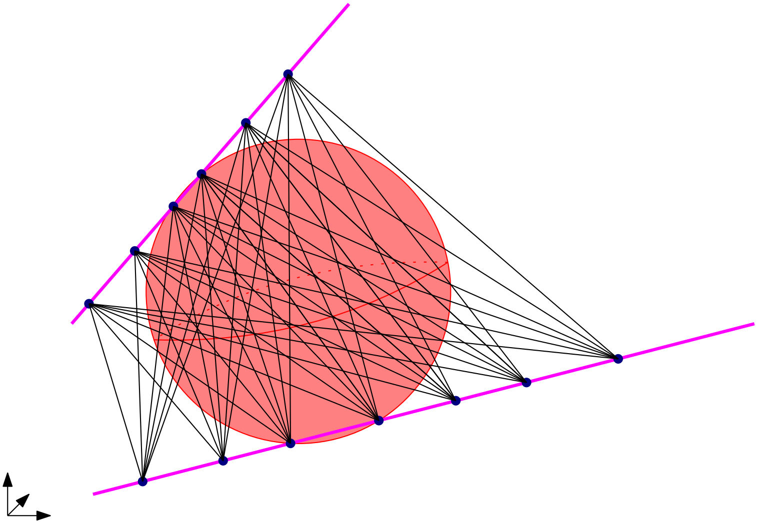
$$2e = 3f + k$$

2D Delaunay has linear size

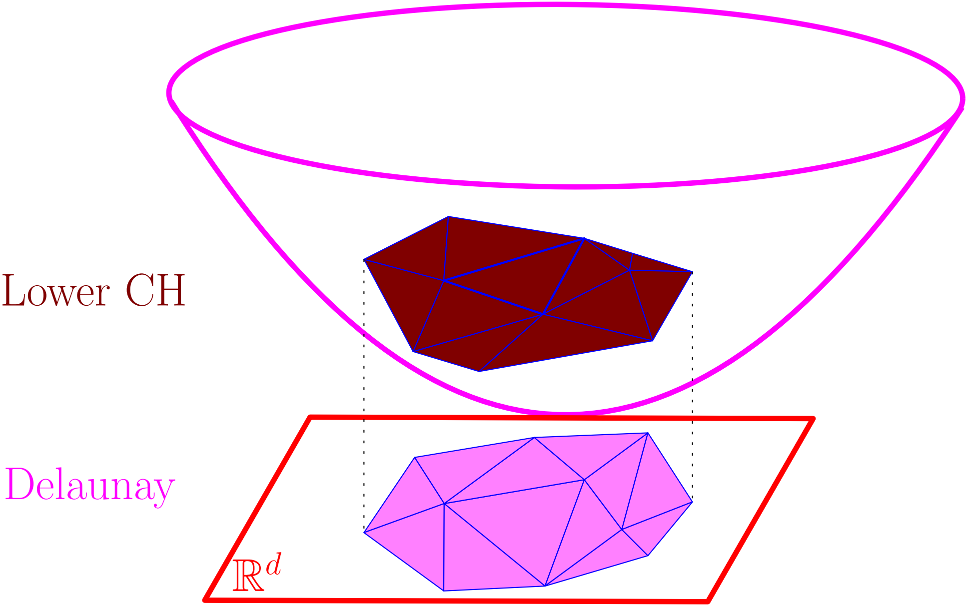
$$f = 2v - 2 - k = O(v)$$

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

3D Delaunay can have quadratic size



point / sphere lifting



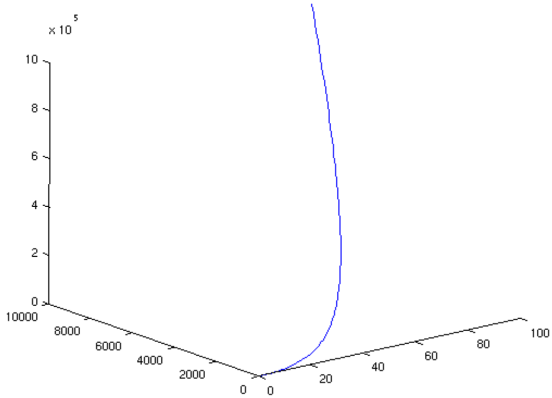
Size of Delaunay in $\mathbb{R}^d$

- 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 $|\mathrm{Del}(P)| \geq |\mathrm{Conv}(P)| = \Omega(|P|^{\lfloor \frac{d}{2} \rfloor}) = \Omega(|P|^{\lceil \frac{d}{2} \rceil})$.

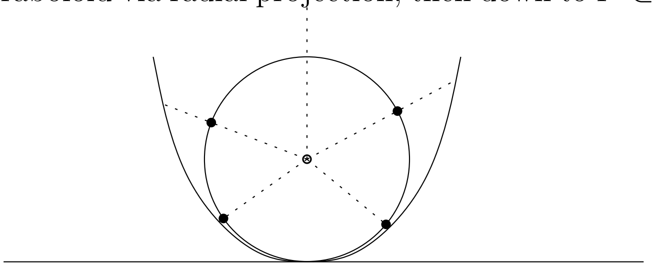


Size of Delaunay in $\mathbb{R}^d$

- 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 $\mathbb{R}^d$

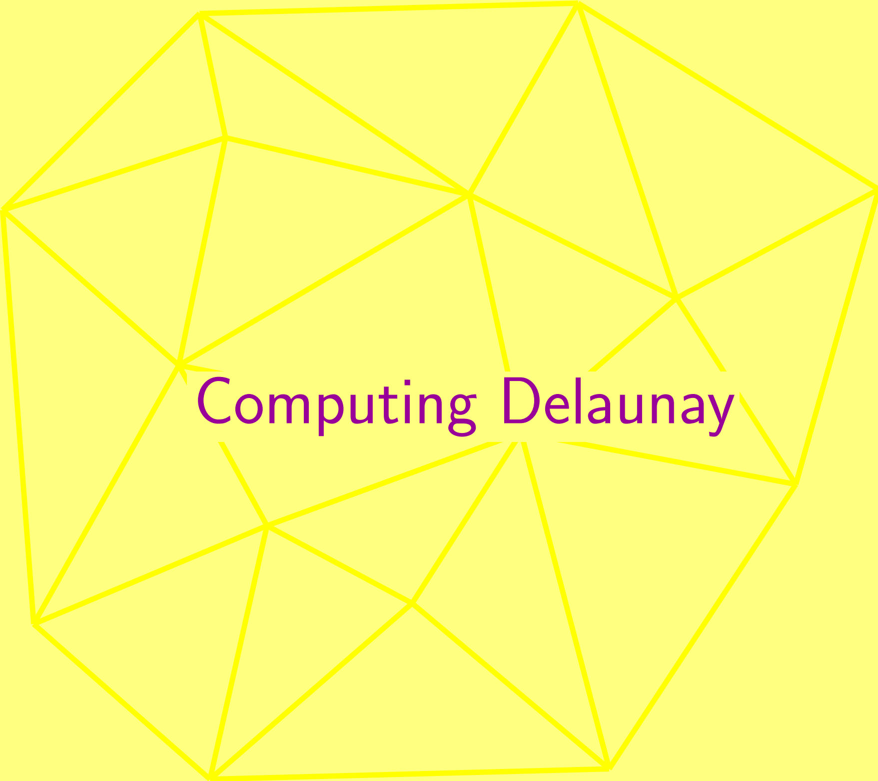
- By point/sphere lifting, $|\mathrm{Del}(P)| = |\mathrm{Conv}(P^*)| = O(|P|^{\lfloor \frac{d+1}{2} \rfloor}) = O(|P|^{\lceil \frac{d}{2} \rceil})$

- 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 $\mathbb{R}^d$: $\Theta(n^{\lceil \frac{d}{2} \rceil})$

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

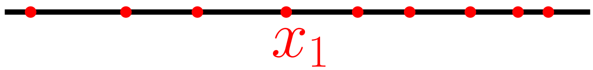
Take an instance of sort

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

Use Delaunay to solve this instance of sort

Lower bound for Delaunay

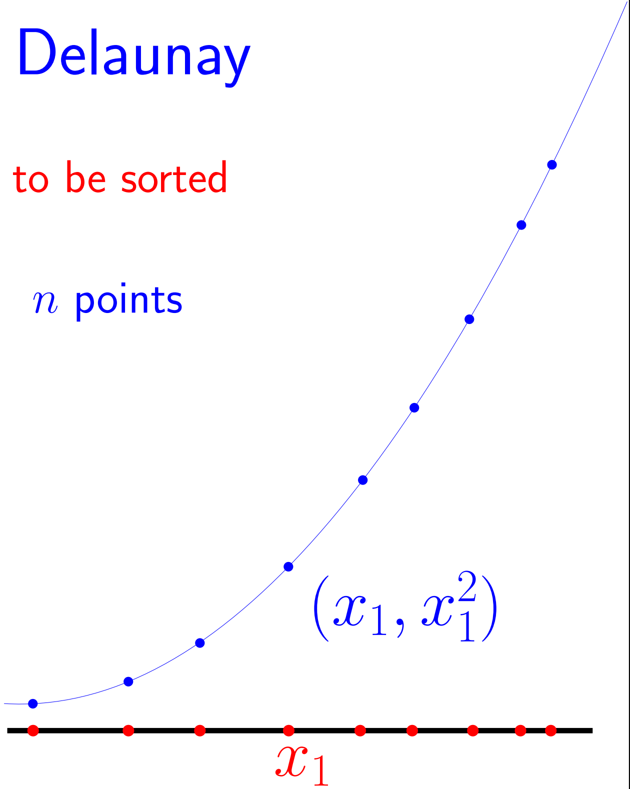
Let $x_1, x_2, \ldots, x_n \in \mathbb{R}$, to be sorted



Lower bound for Delaunay

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

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



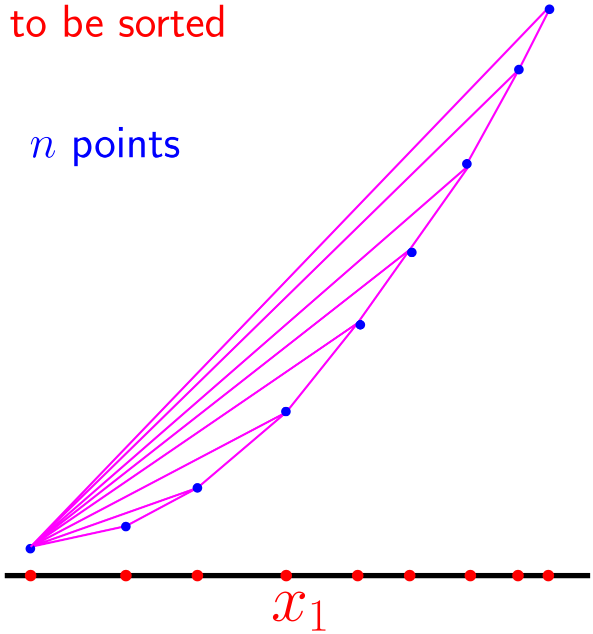
Lower bound for Delaunay

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

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

Delaunay

$\rightarrow$ order in x



Lower bound for Delaunay

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

$\downarrow O(n)$

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

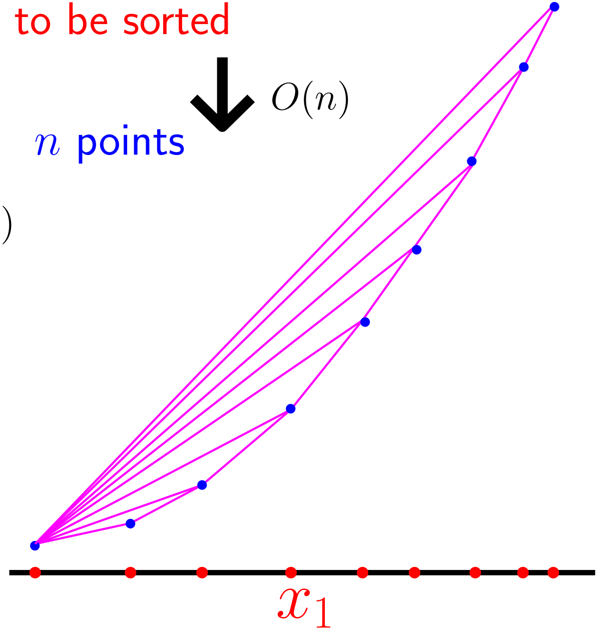
$\downarrow f(n)$

Delaunay

$\downarrow O(n)$

$\rightarrow$ order in x

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



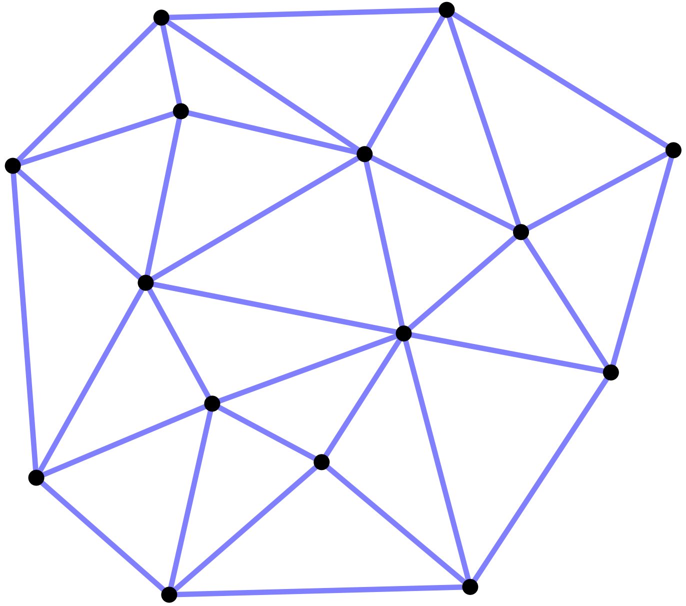
Lower bound for Delaunay

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

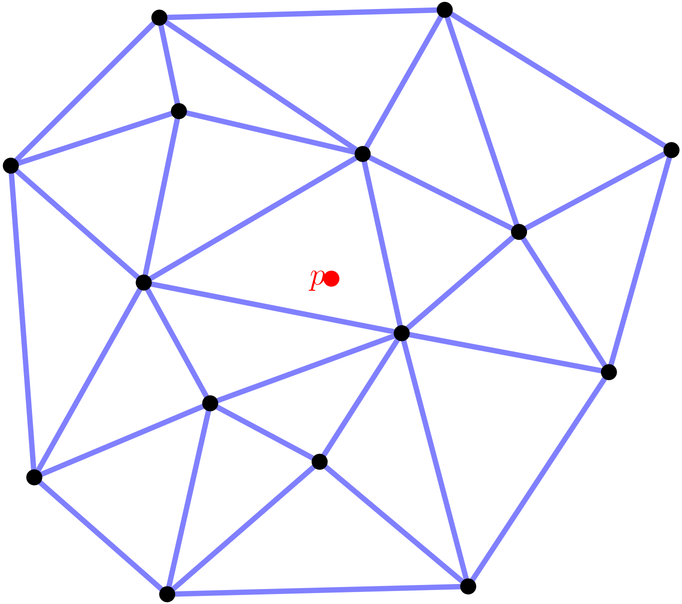
Computing Delaunay

Incremental algorithm

Algorithm overview

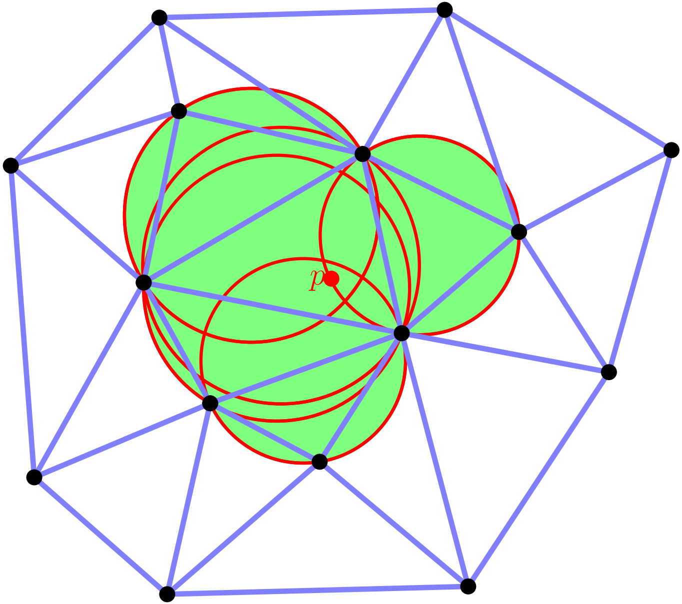


Algorithm overview



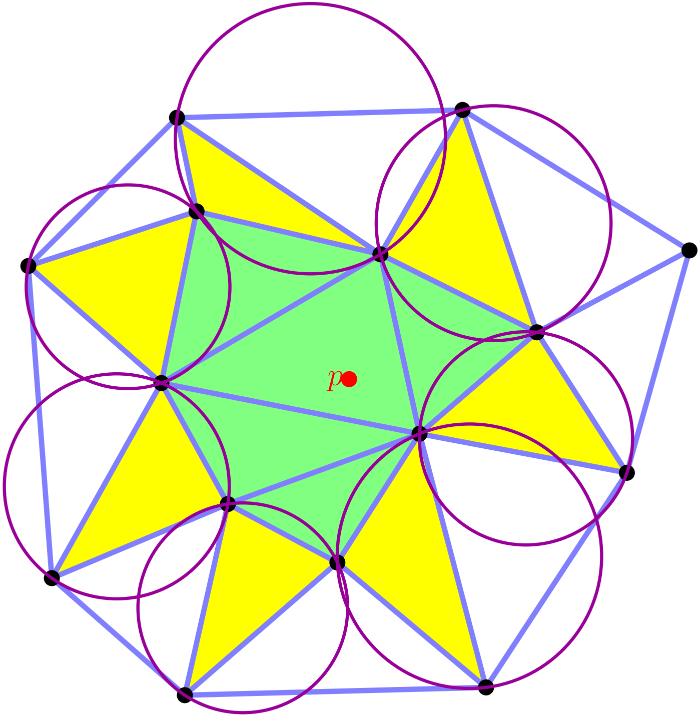
Algorithm overview

- Find triangles in conflict with p



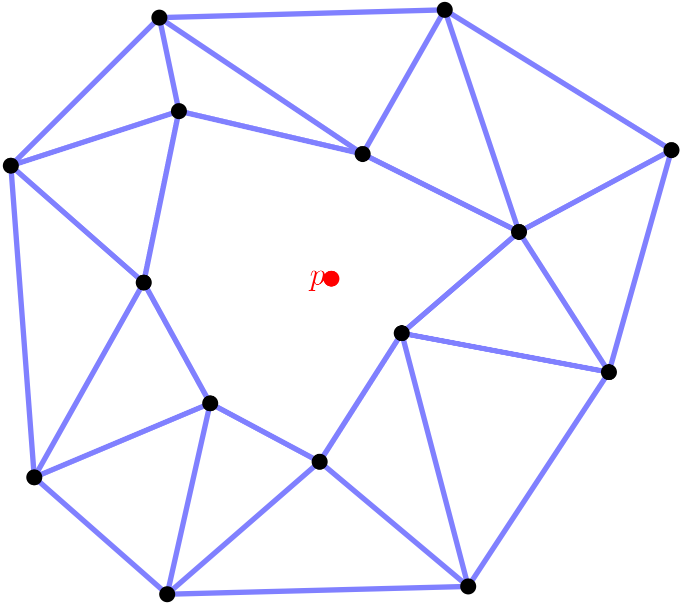
Algorithm overview

- Find triangles in conflict with p



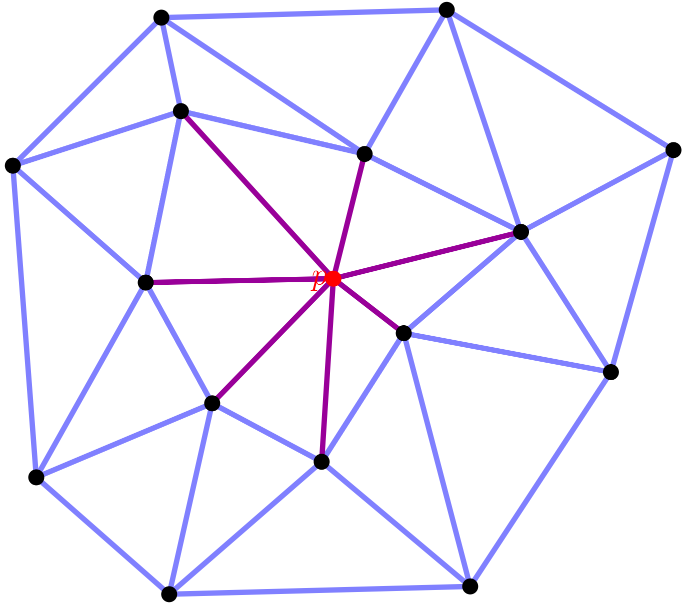
Algorithm overview

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



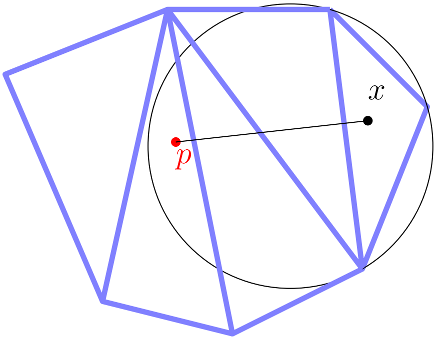
Algorithm overview

- 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)

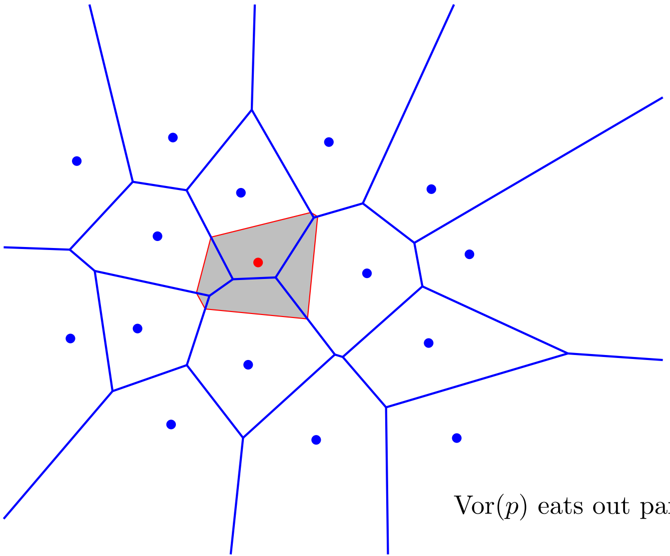
$\rightarrow$ can be computed by a traversal in the dual graph from some $\sigma \ni p$

$\rightarrow$ 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].

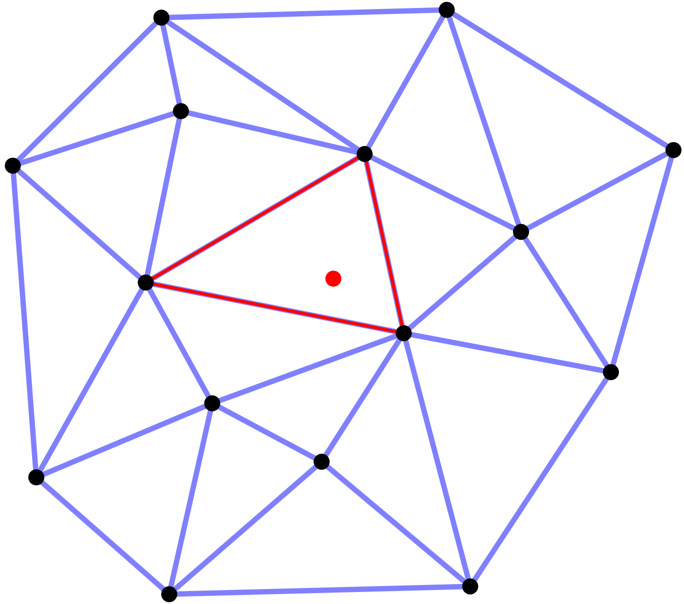
$$\Rightarrow \text{total complexity} = O(n \log n + n^{\lceil \frac{d+1}{2} \rceil})$$

(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)

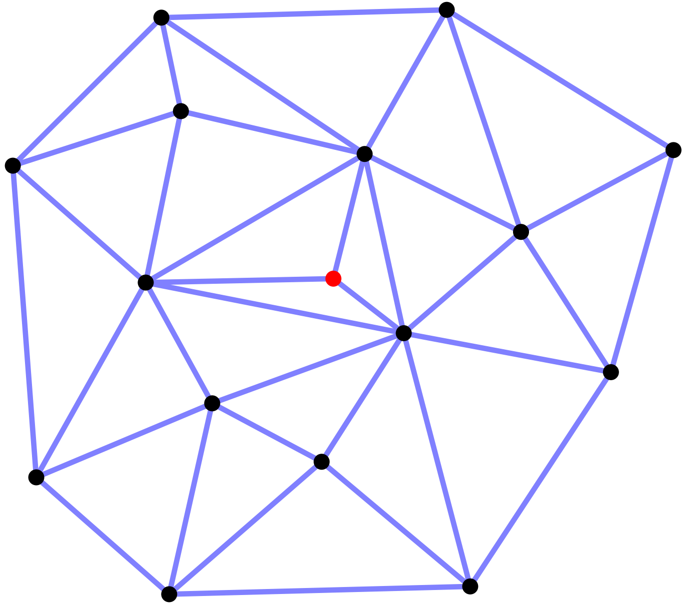
The Guibas/Stolfi variant in 2D

- Locate point in triangulation



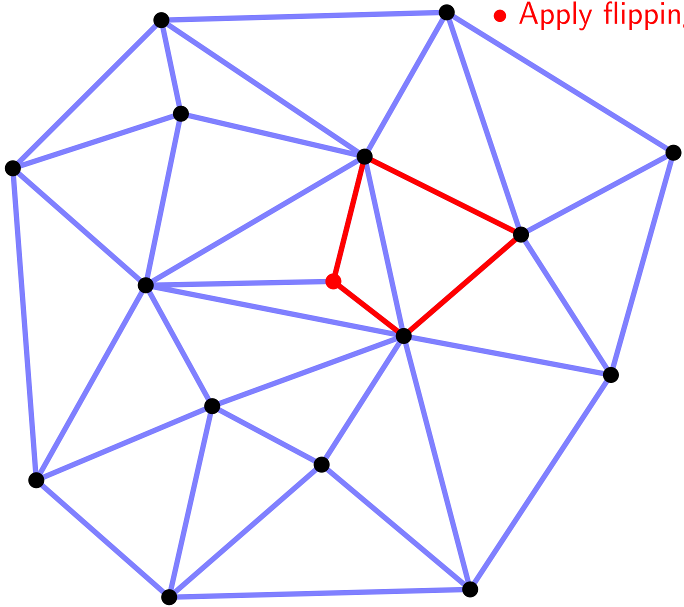
The Guibas/Stolfi variant in 2D

- Locate point in triangulation
- Star triangle



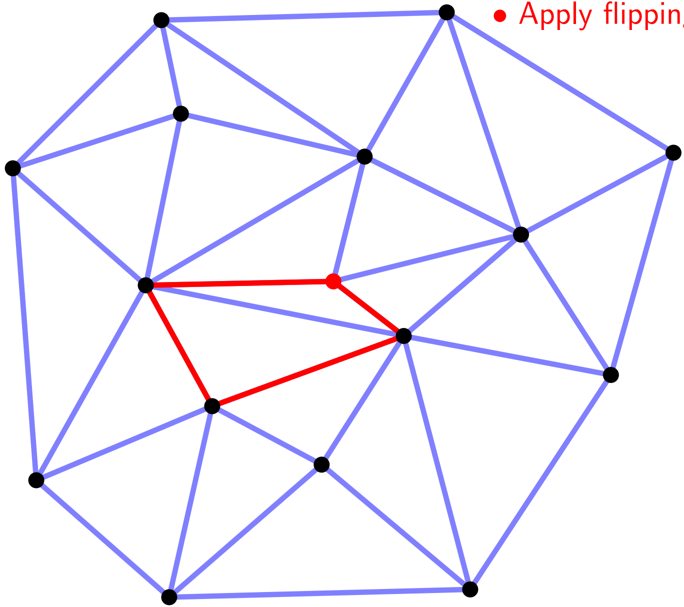
The Guibas/Stolfi variant in 2D

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



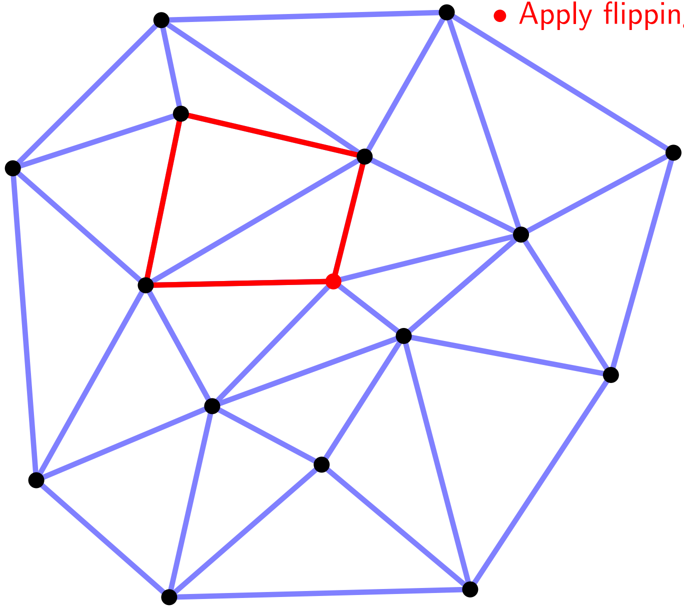
The Guibas/Stolfi variant in 2D

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



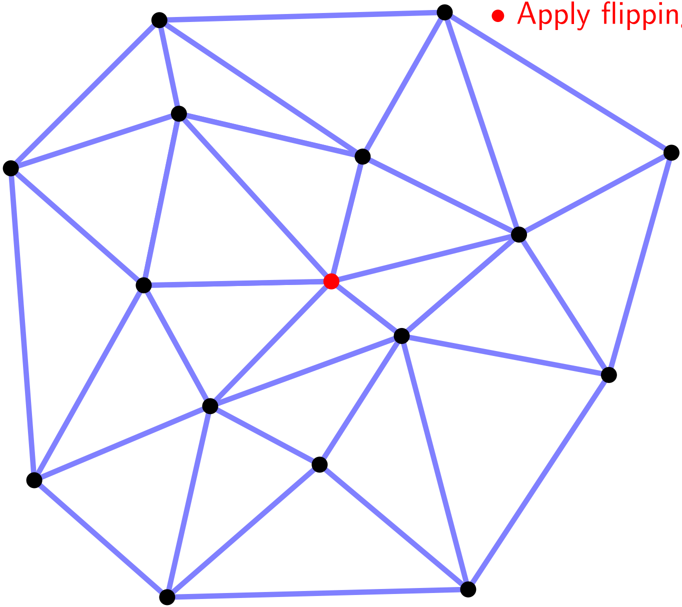
The Guibas/Stolfi variant in 2D

- 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 $O\left({}^{n}/_{2}\right)$

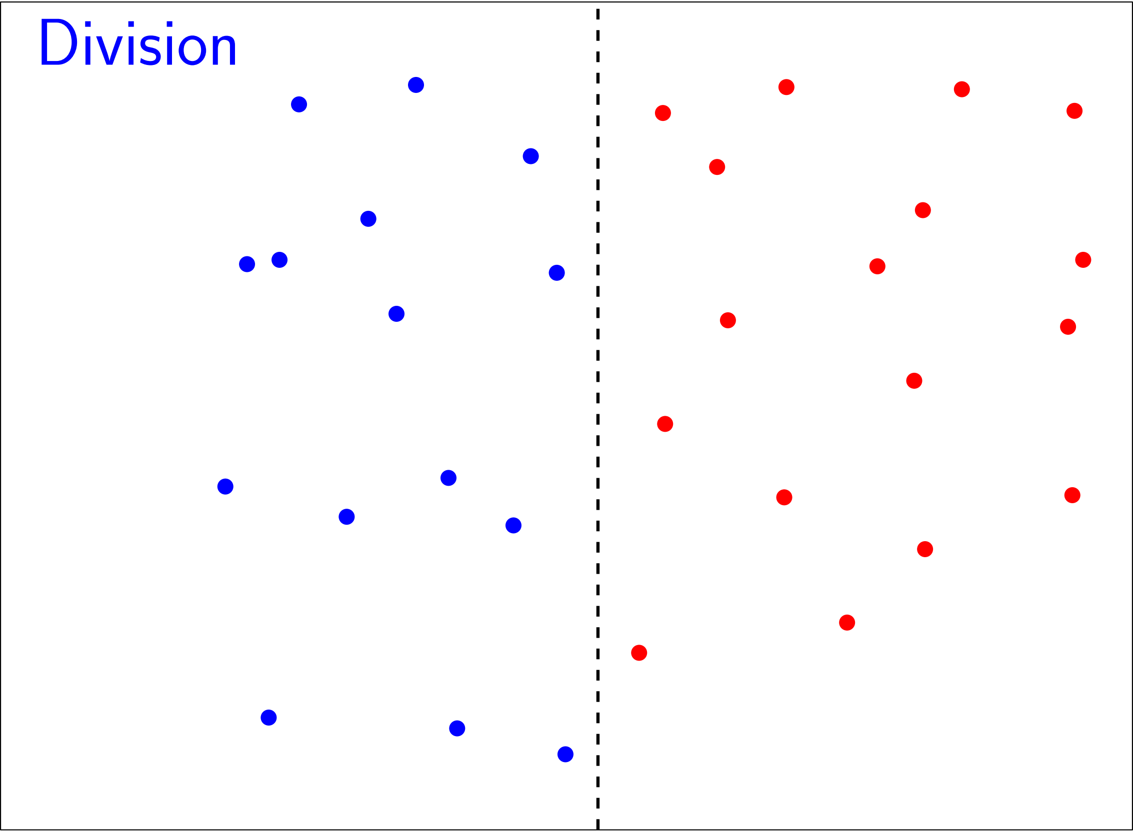
$O(n)$

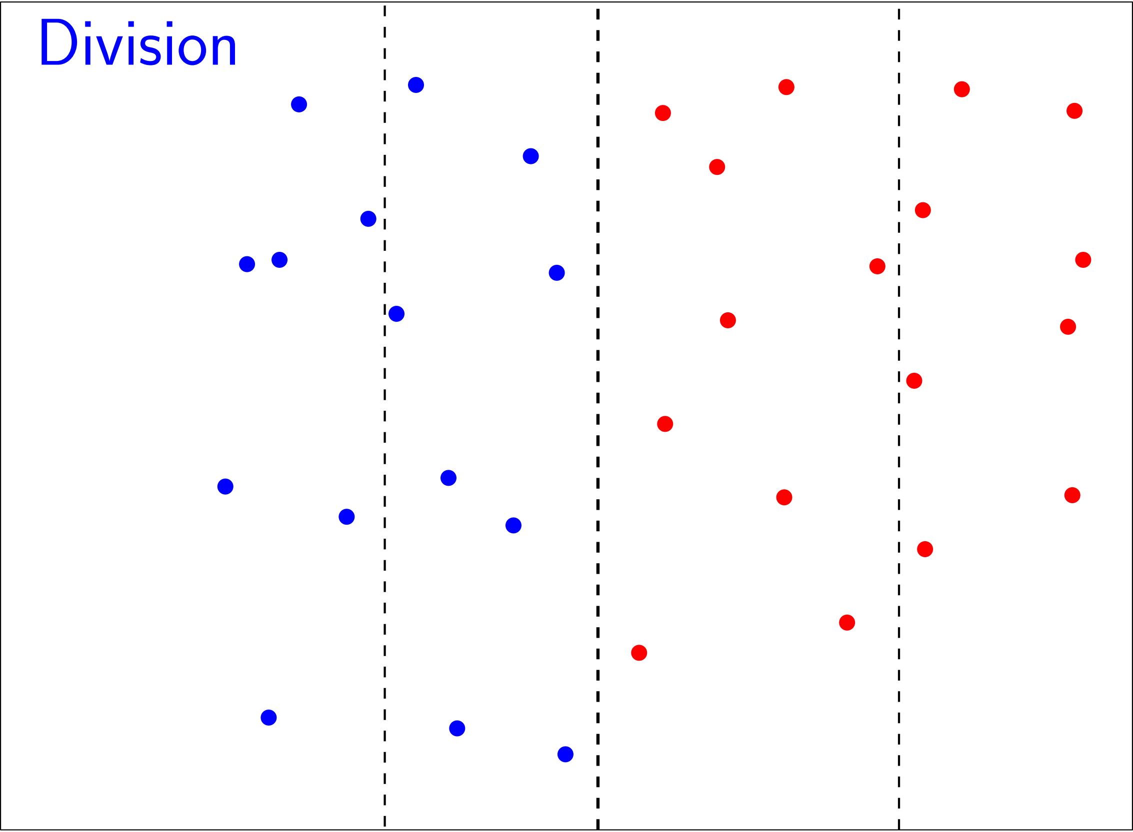
$\rightarrow$ recursive call on sub-problems

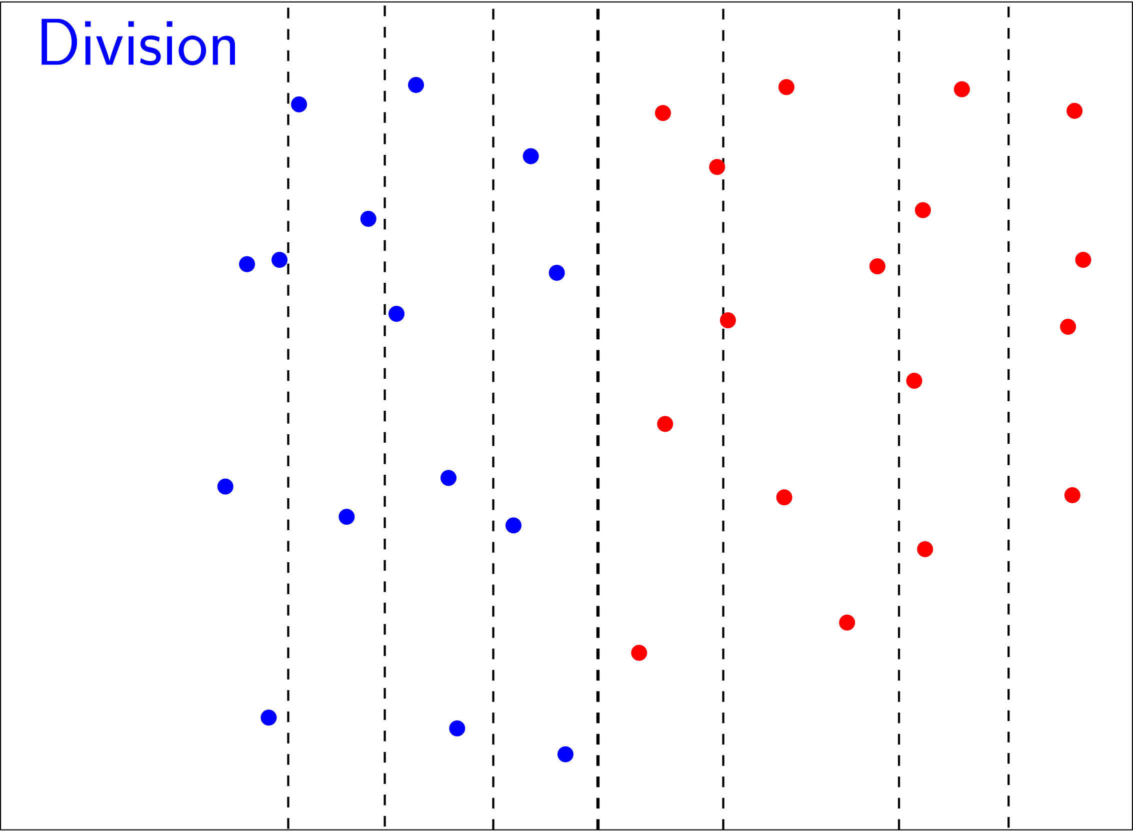
$2\,f\left(\frac{n}{2}\right)$

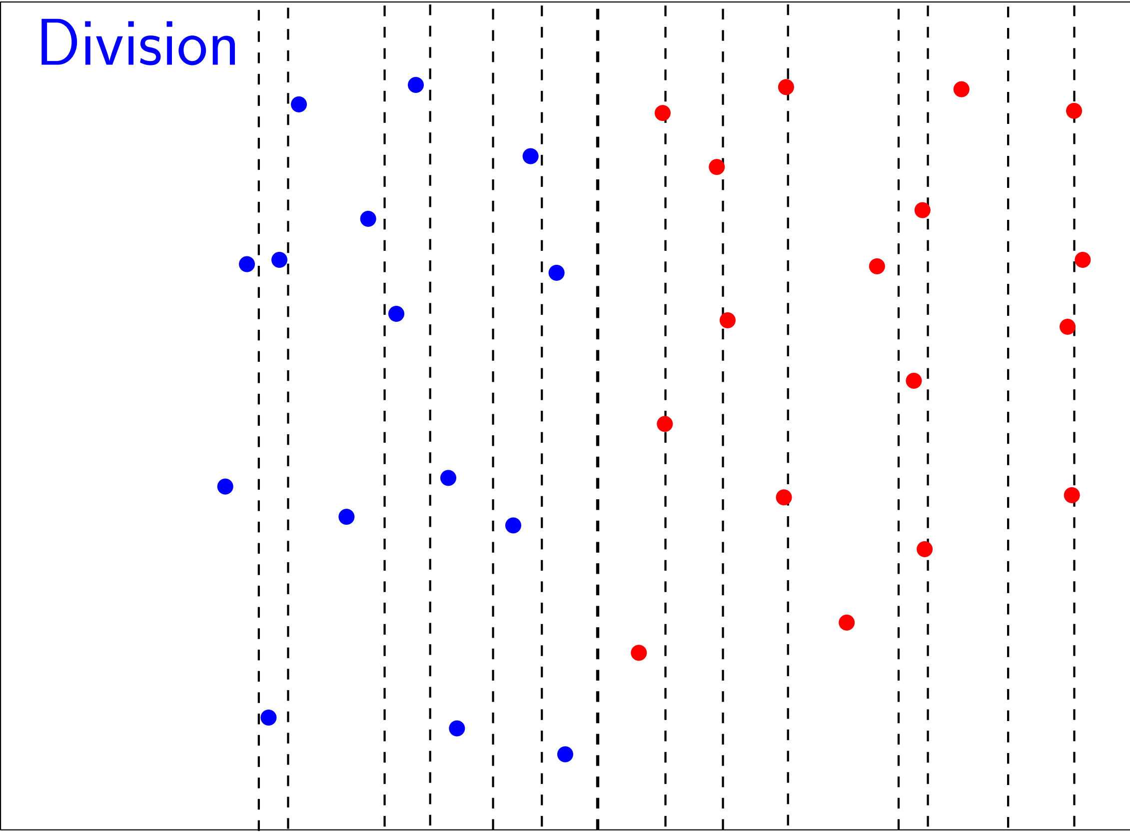
$\rightarrow$ fusion

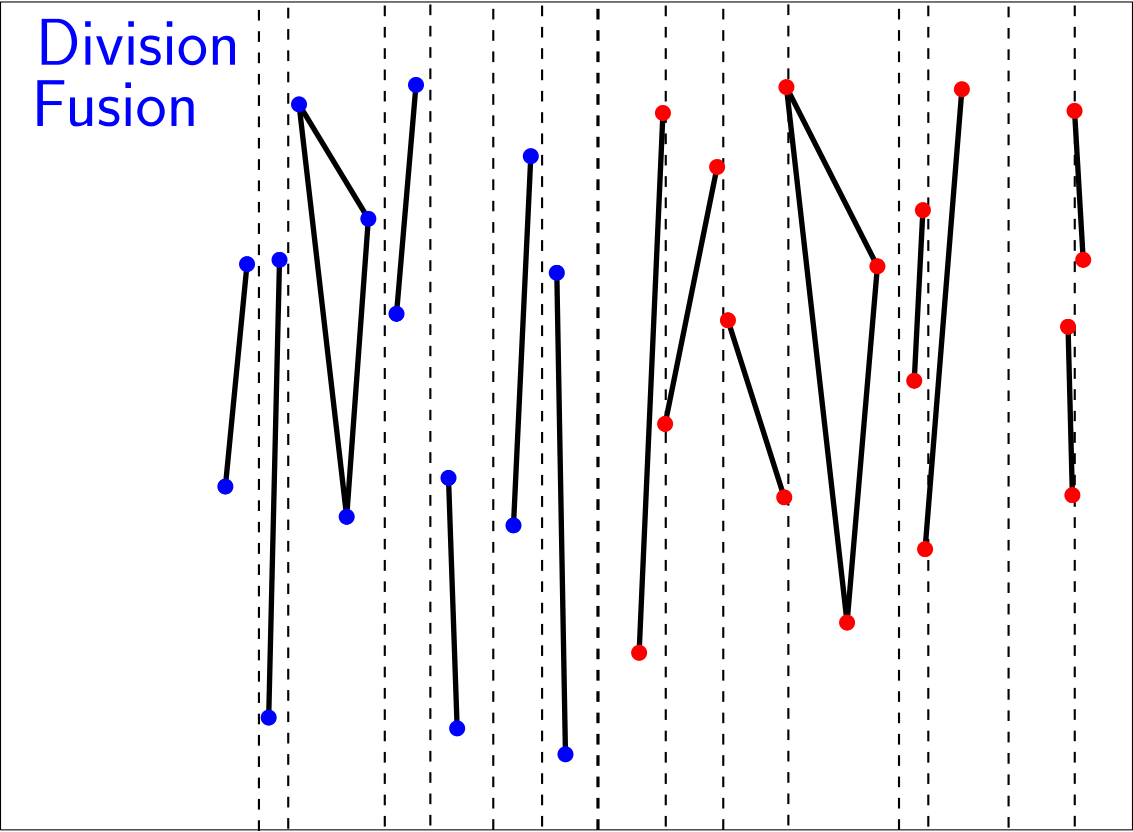
$O(n)$


Division

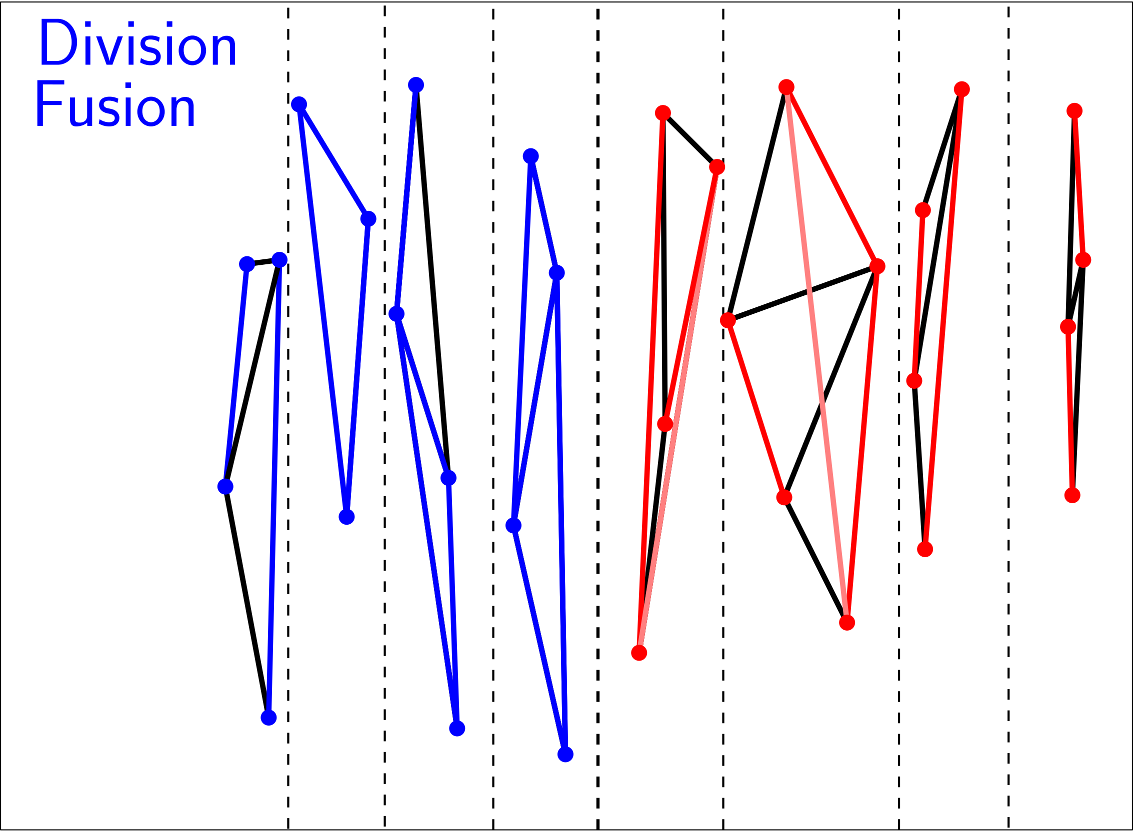

Division


Division

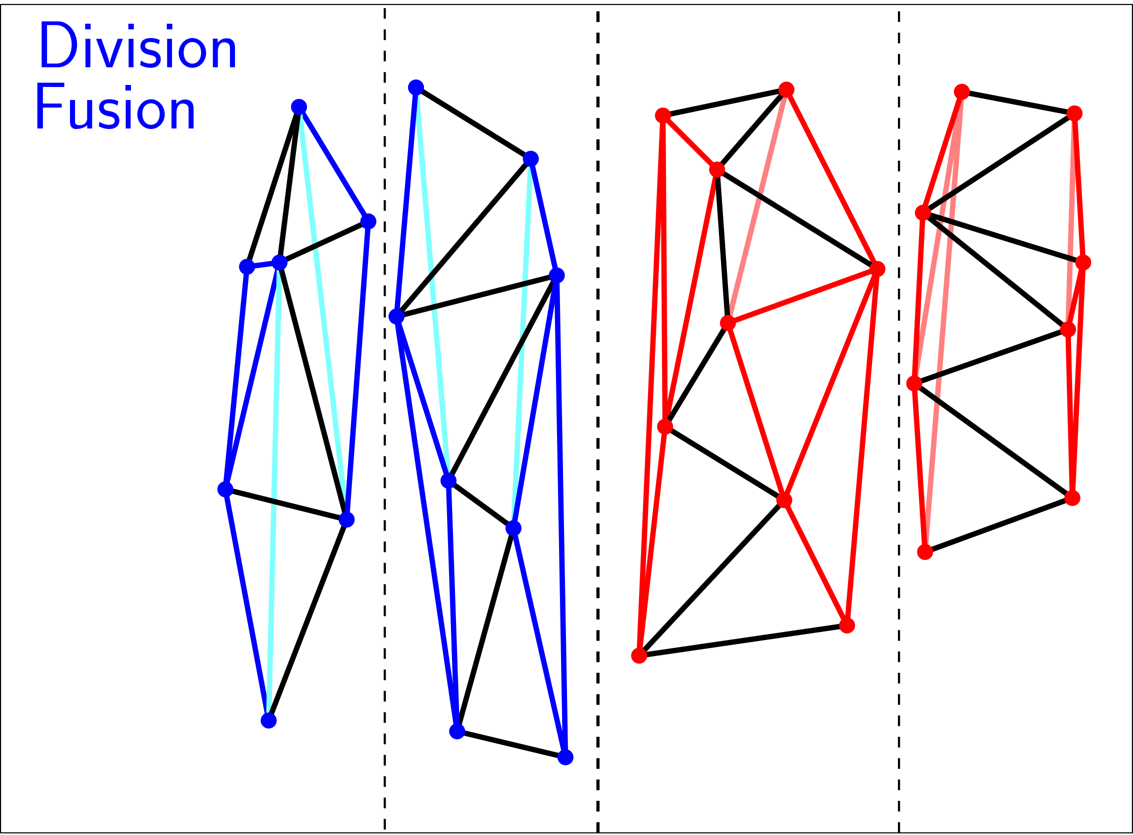

Division

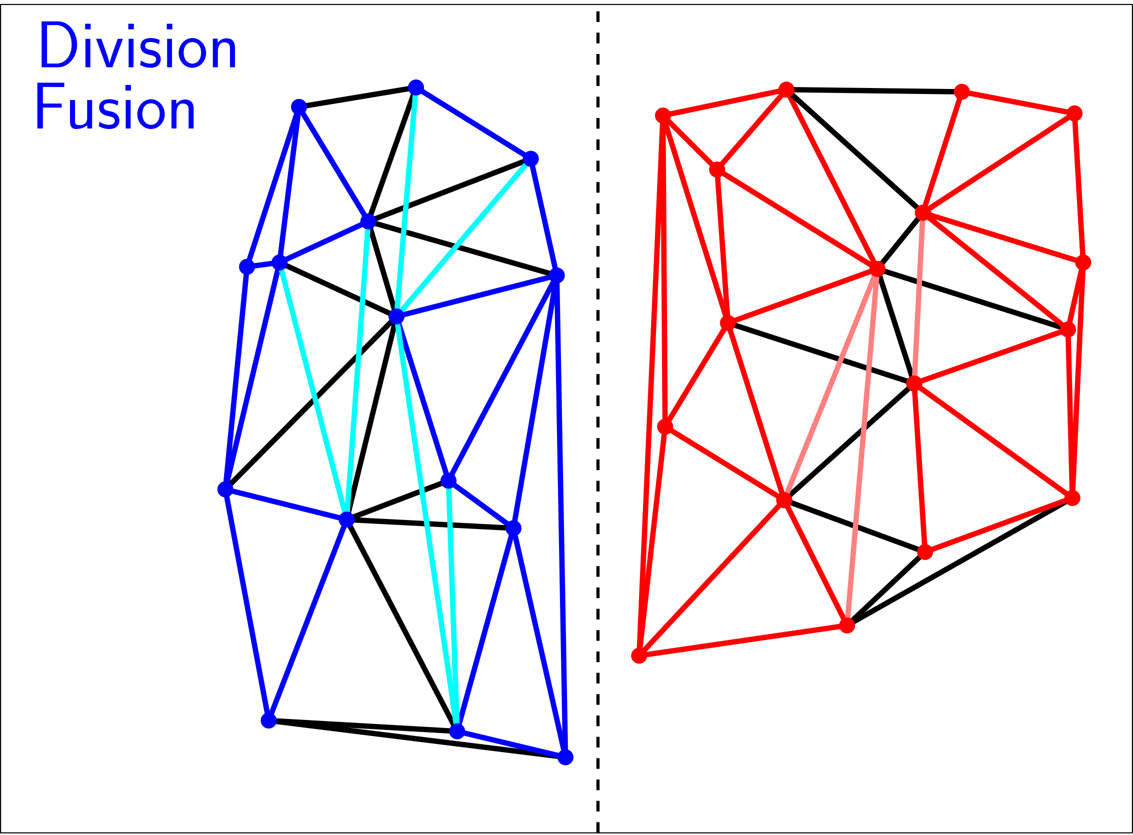


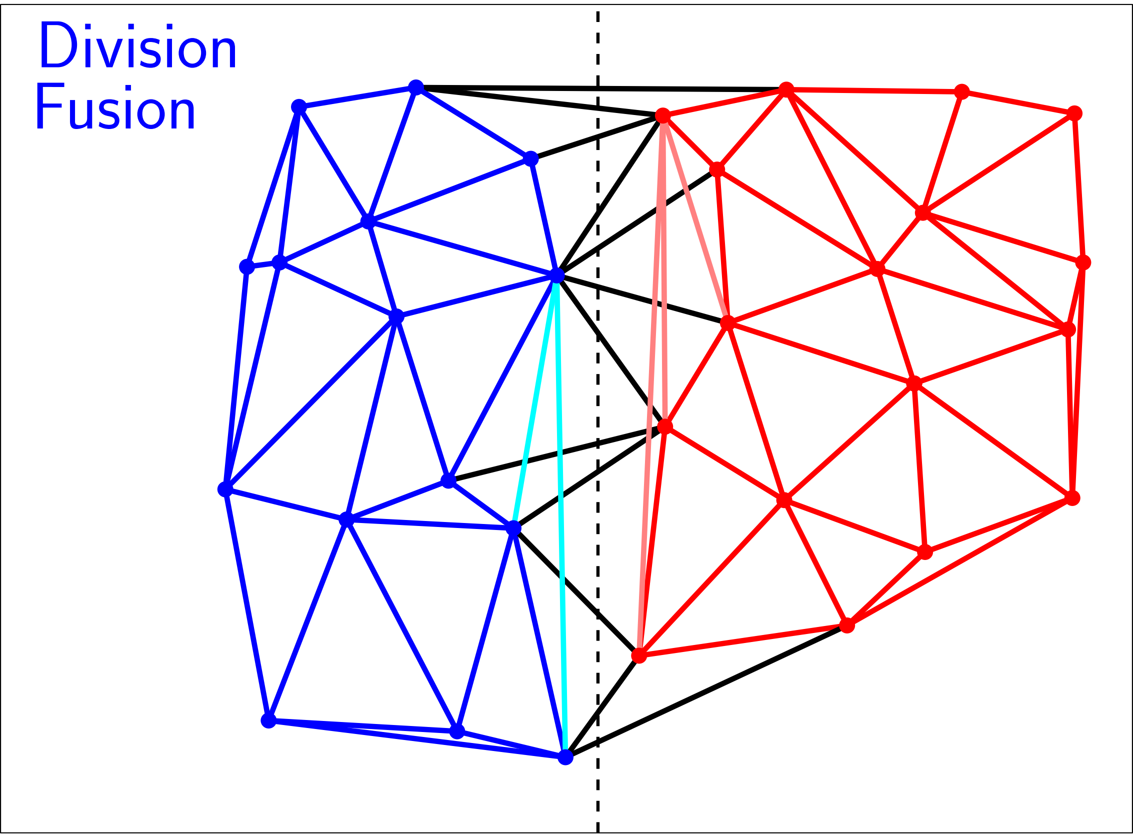
Division
Fusion

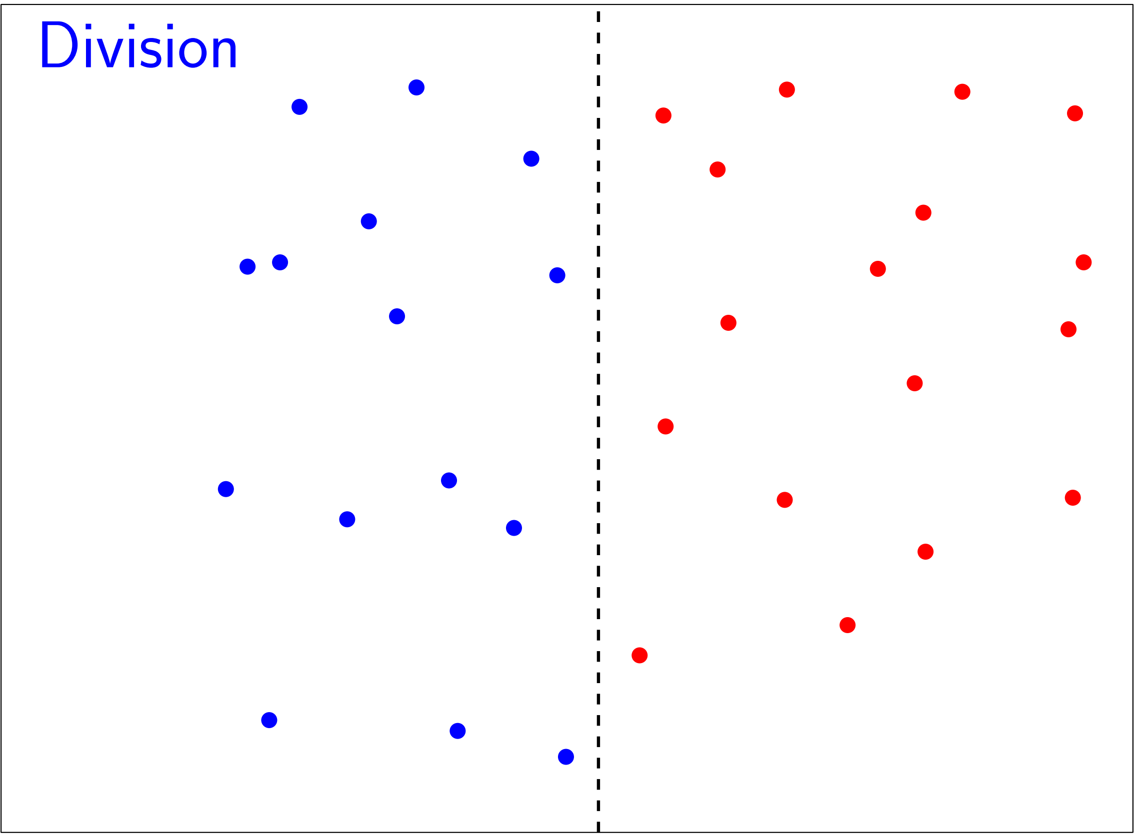


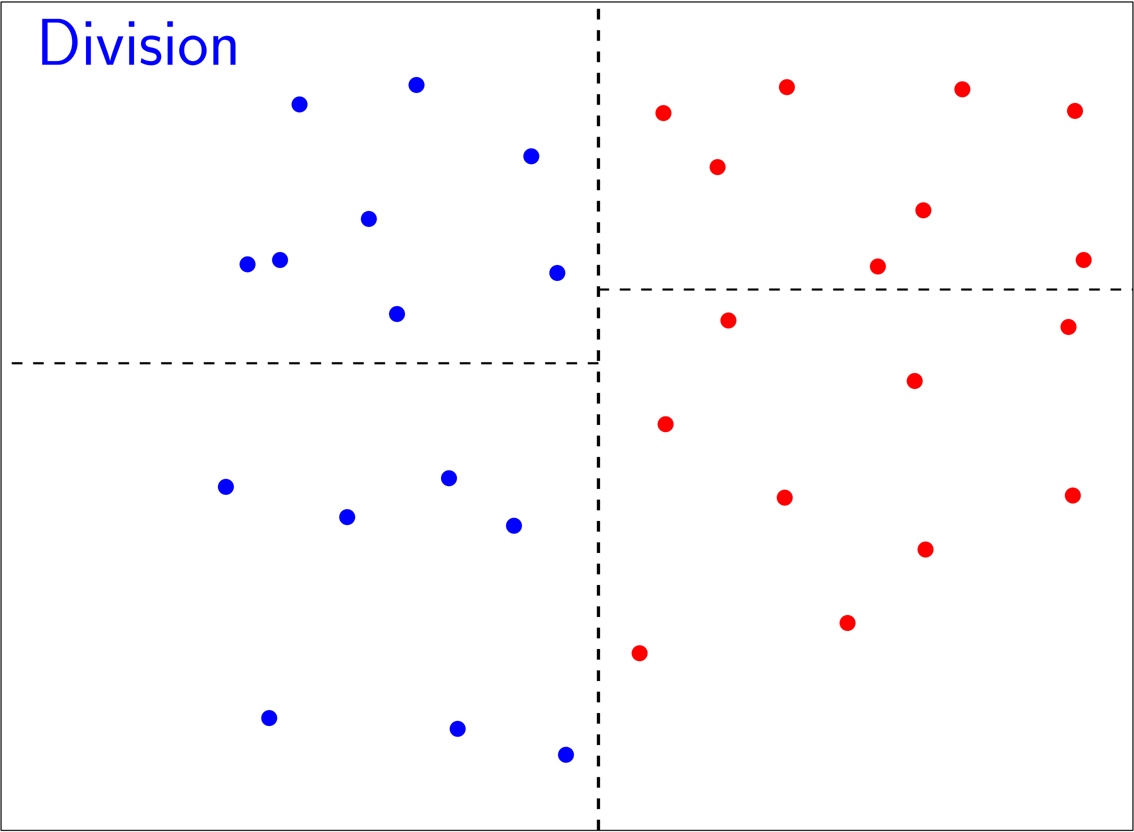
Division
Fusion

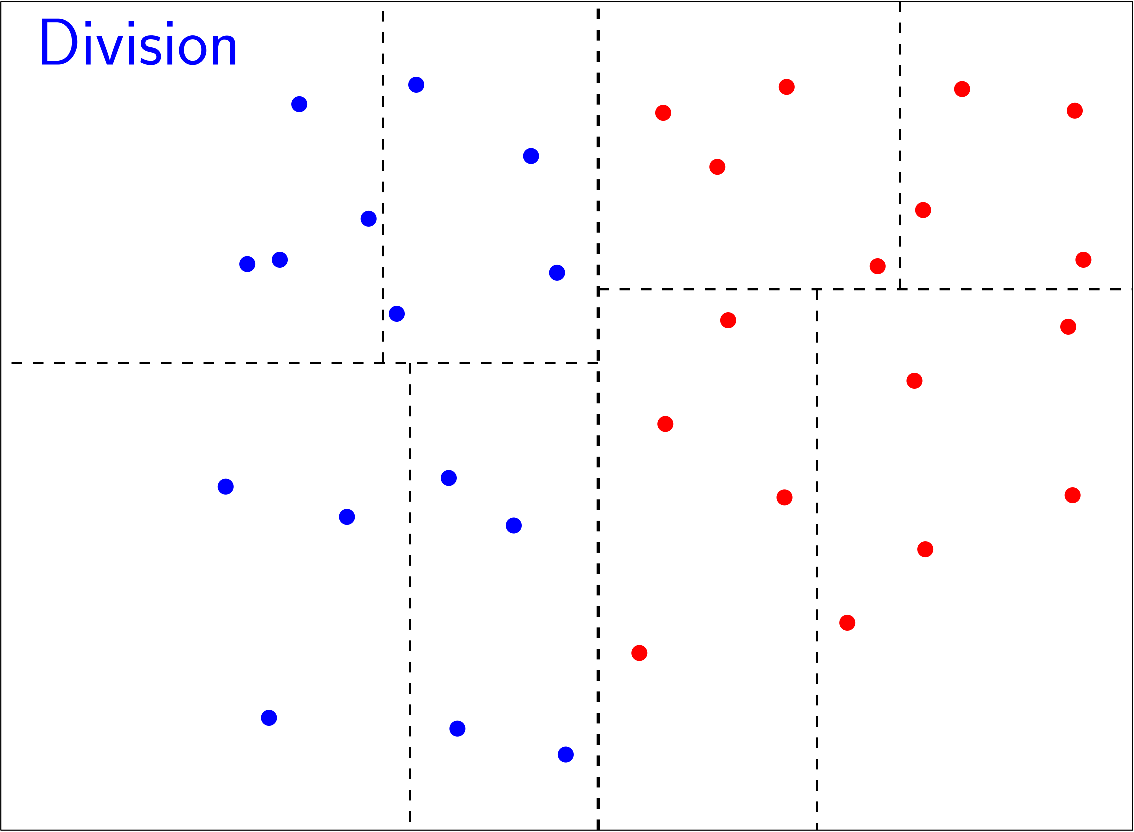

Division
Fusion

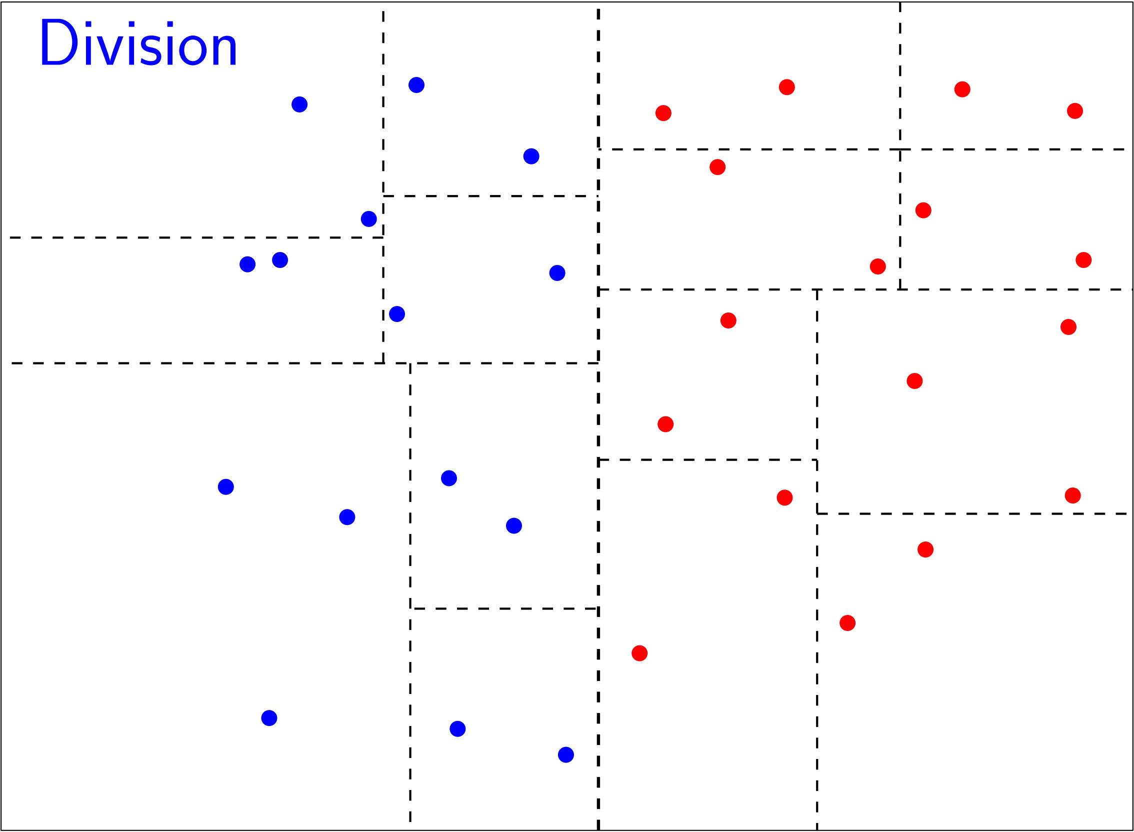

Division
Fusion

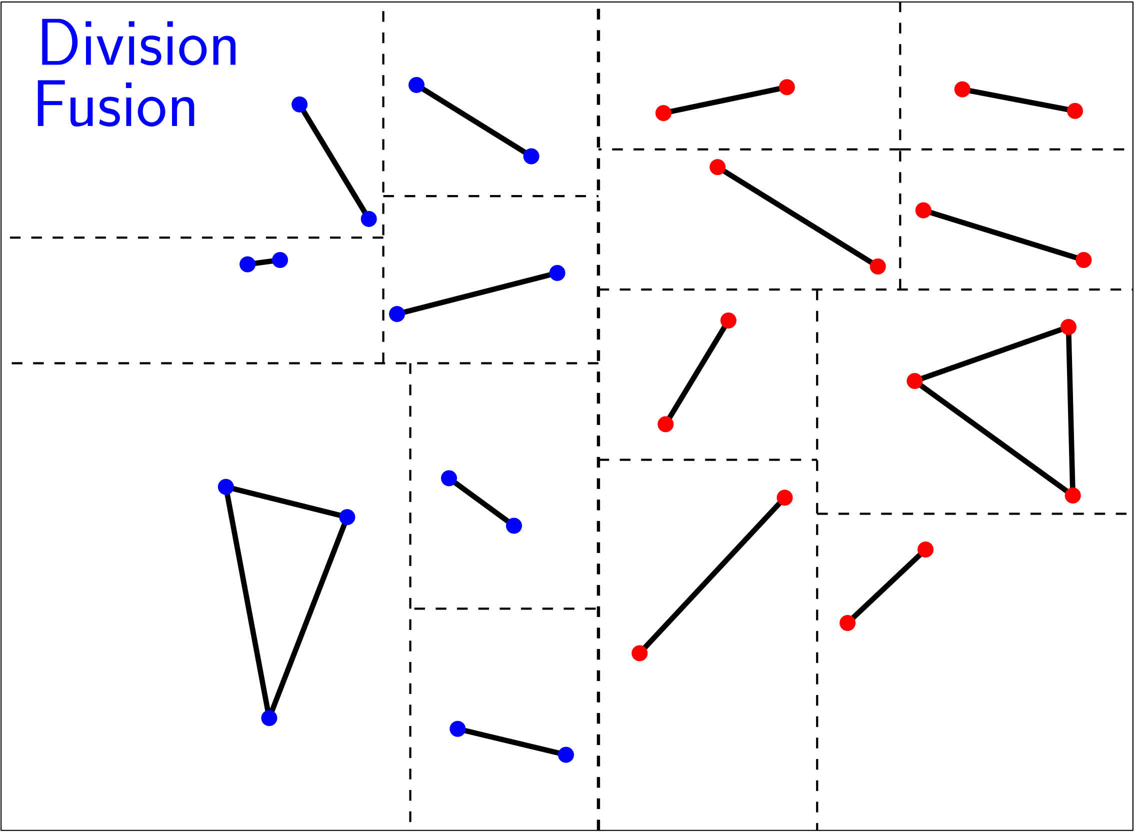

Division
Fusion

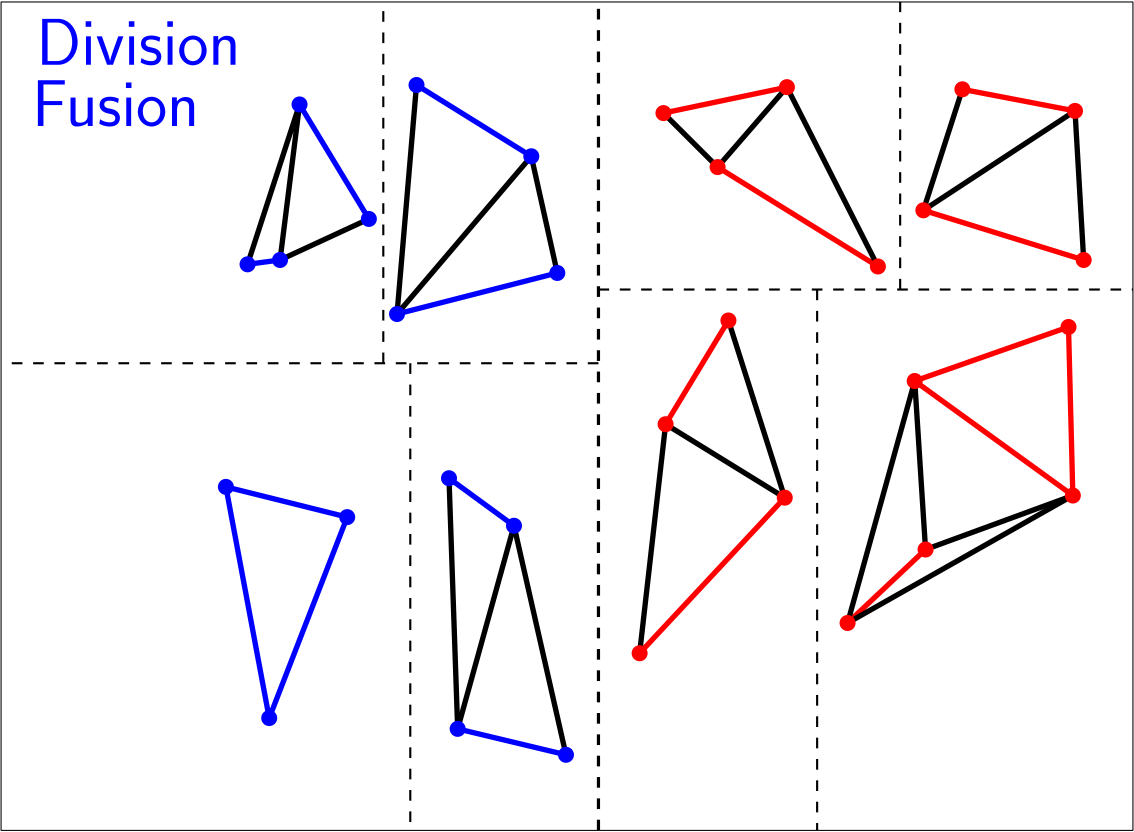

Division

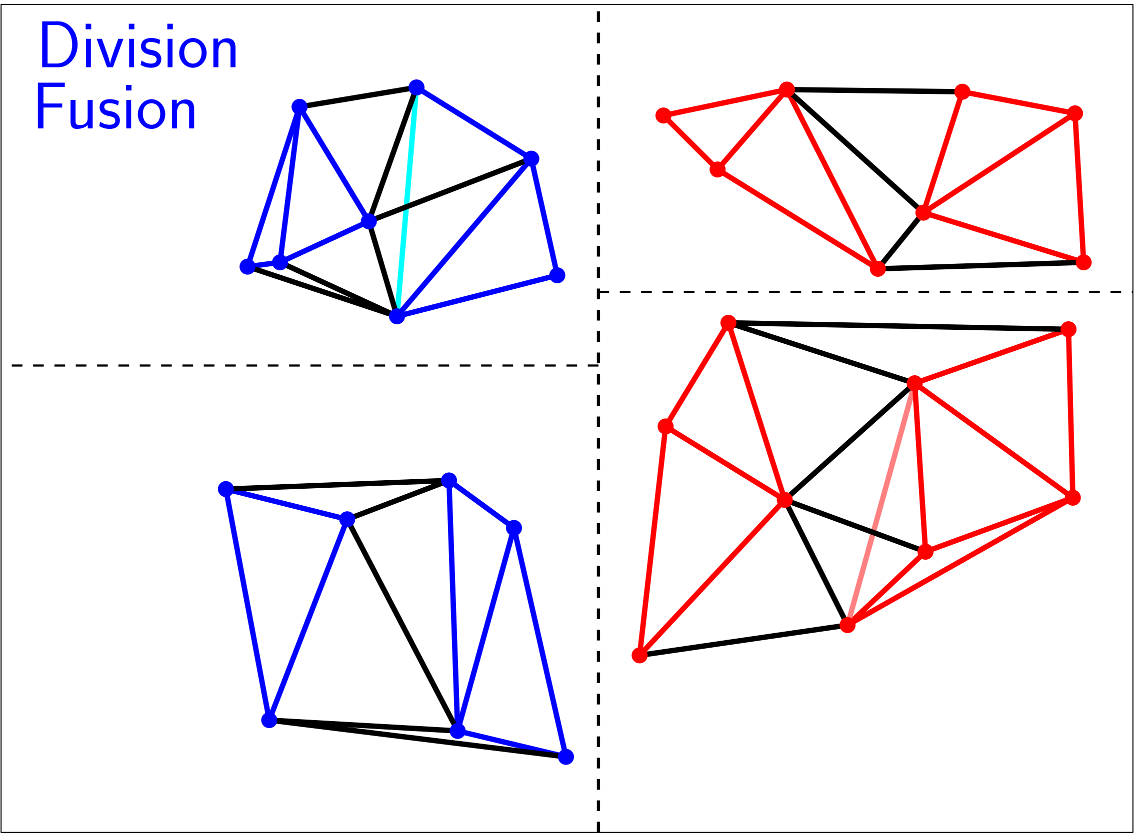

Division

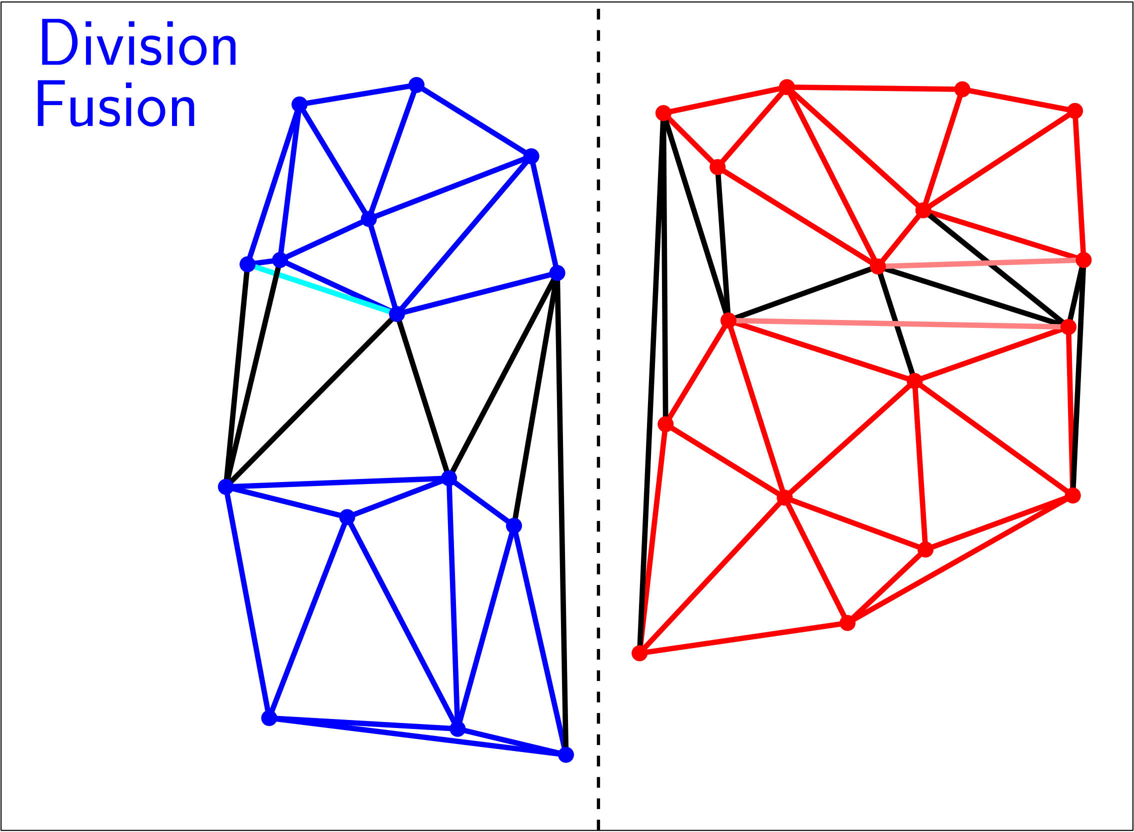

Division

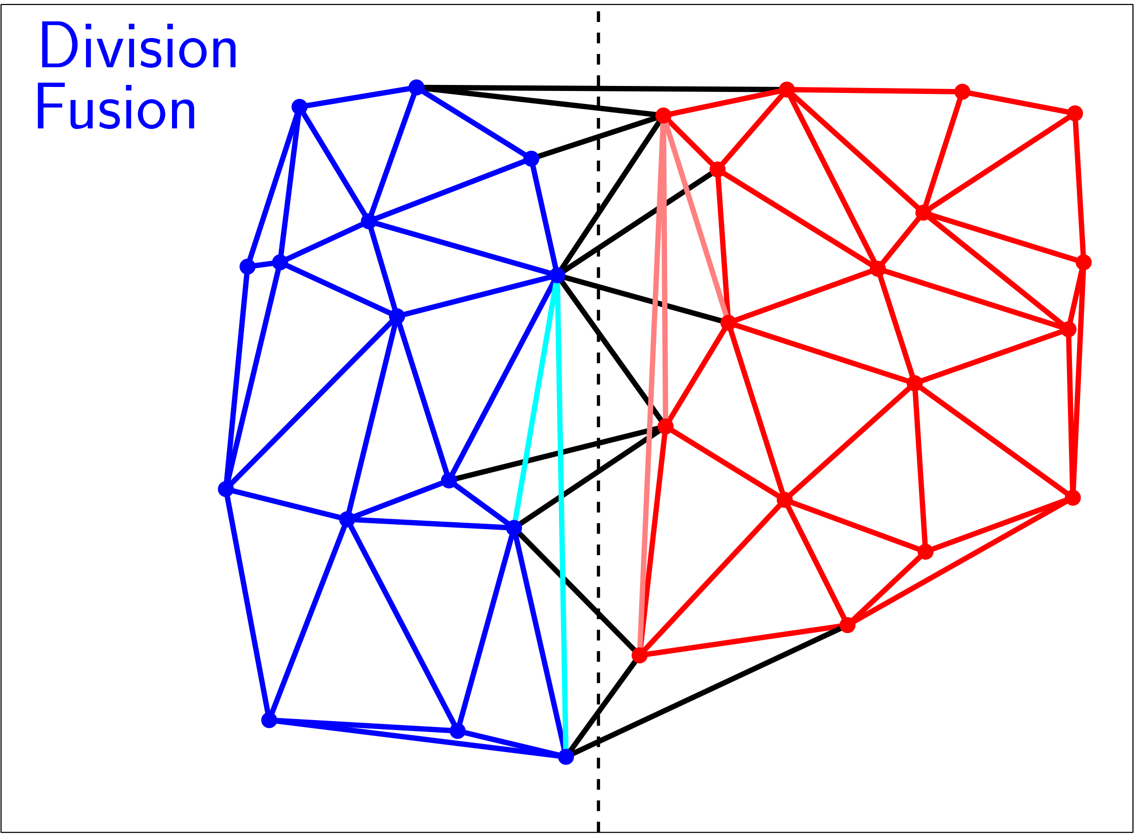

Division

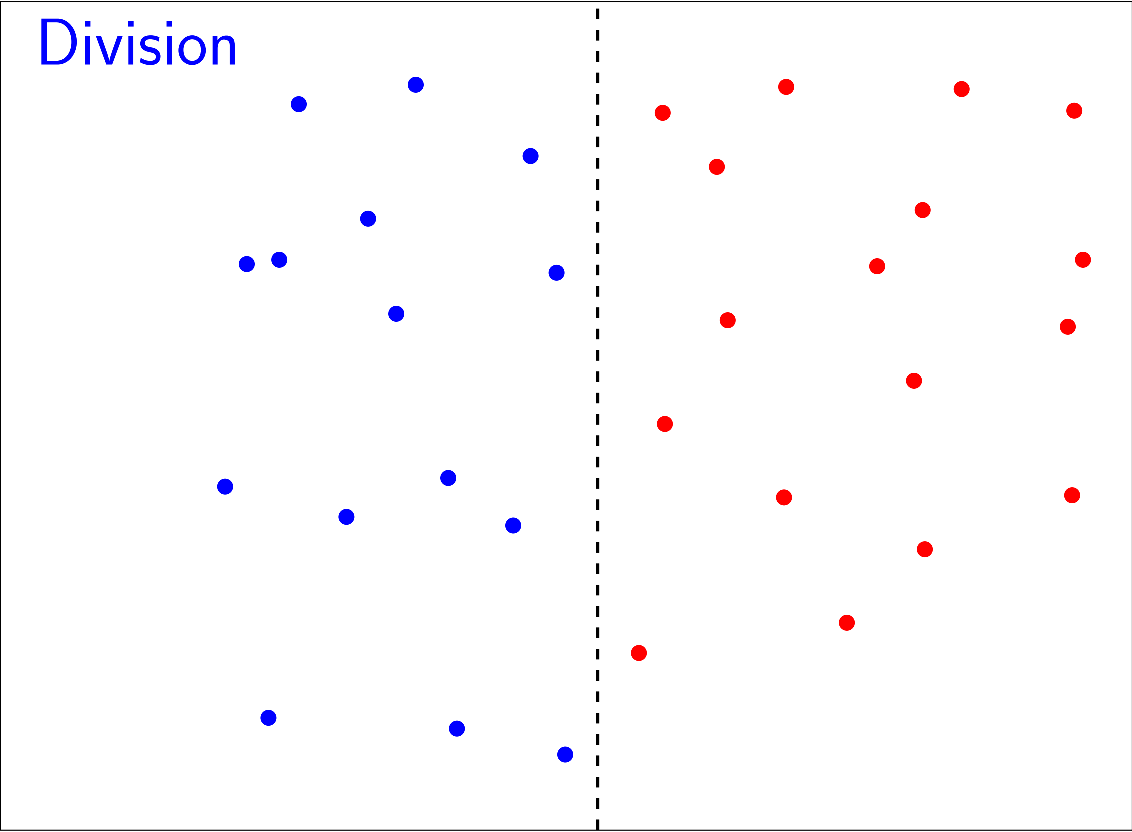

Division
Fusion


Division
Fusion

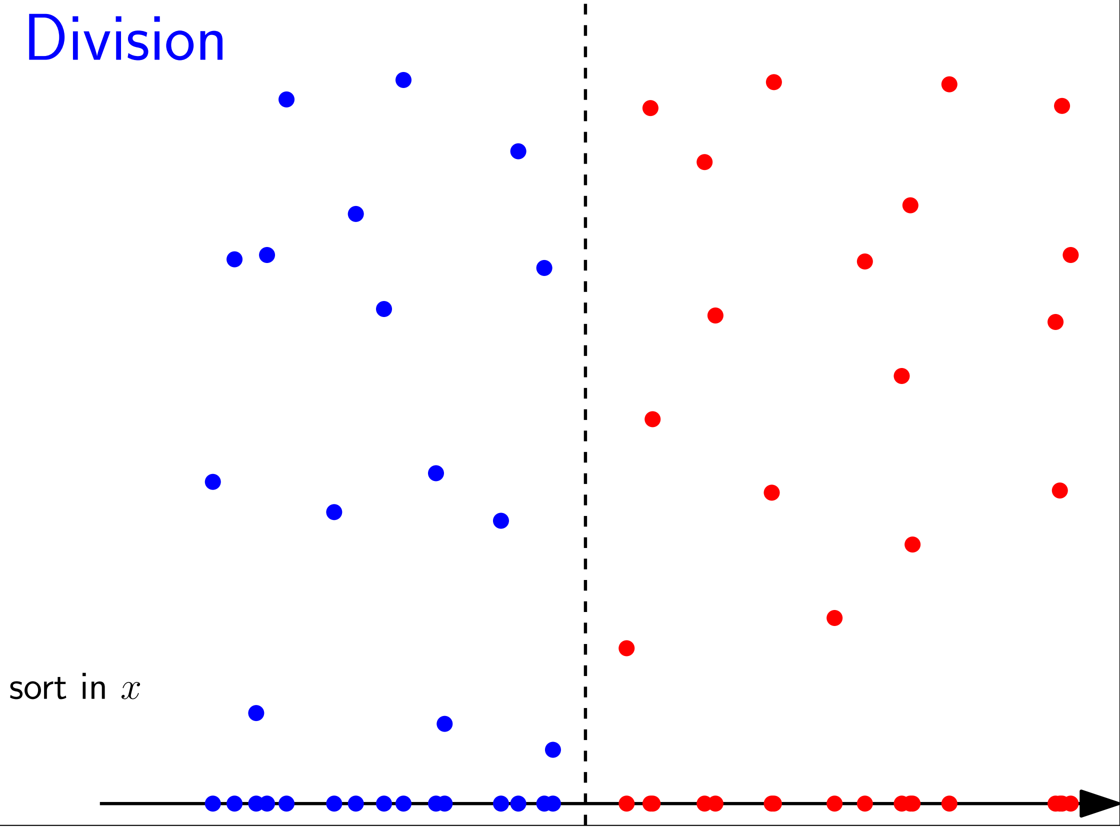

Division
Fusion


Division
Fusion

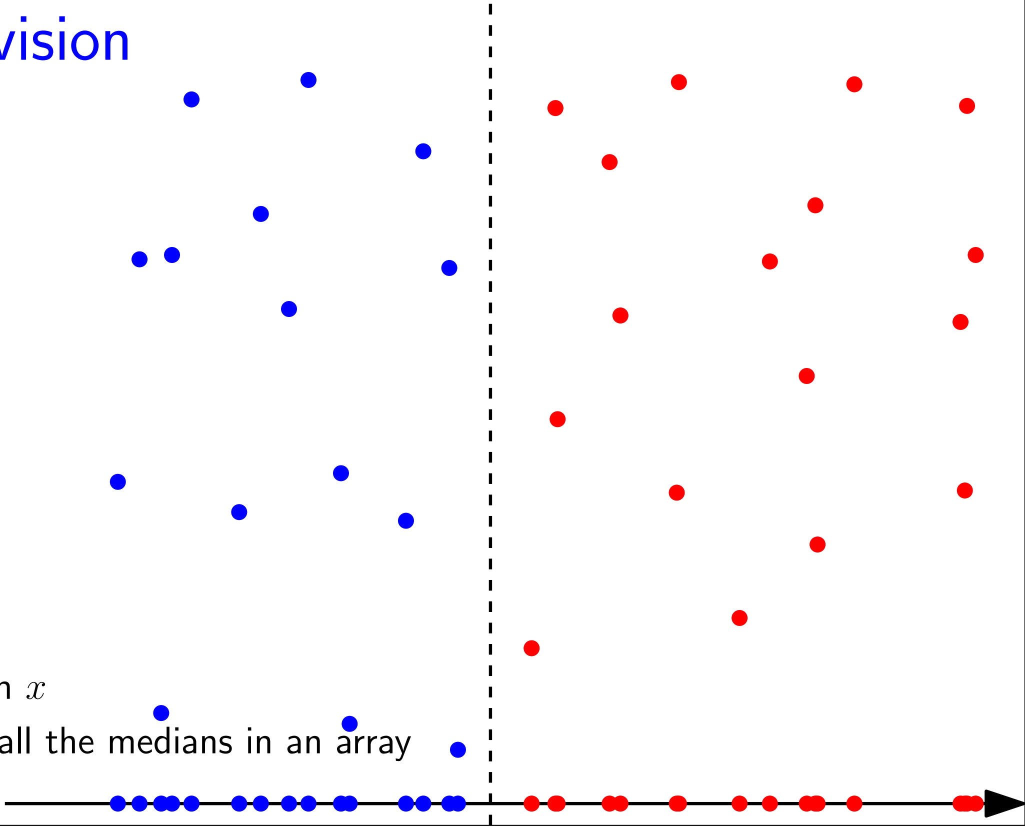

Division
Fusion


Division

Division



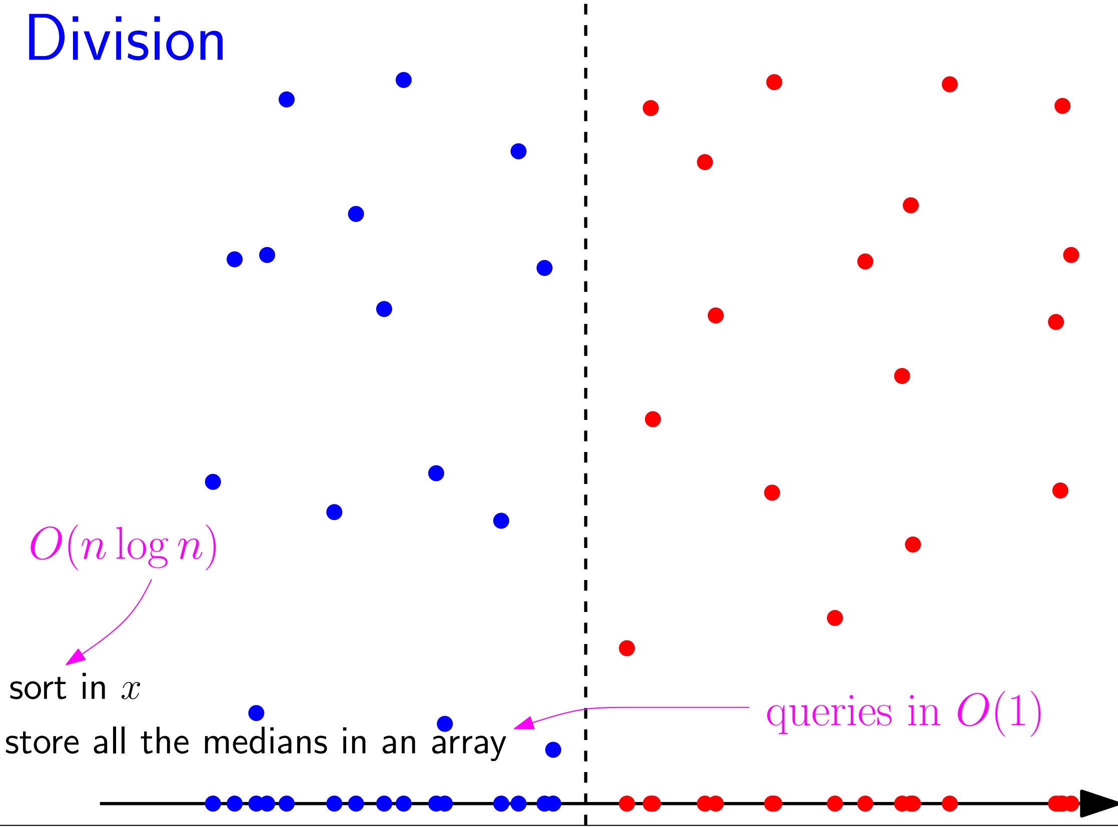
Division



sort in x

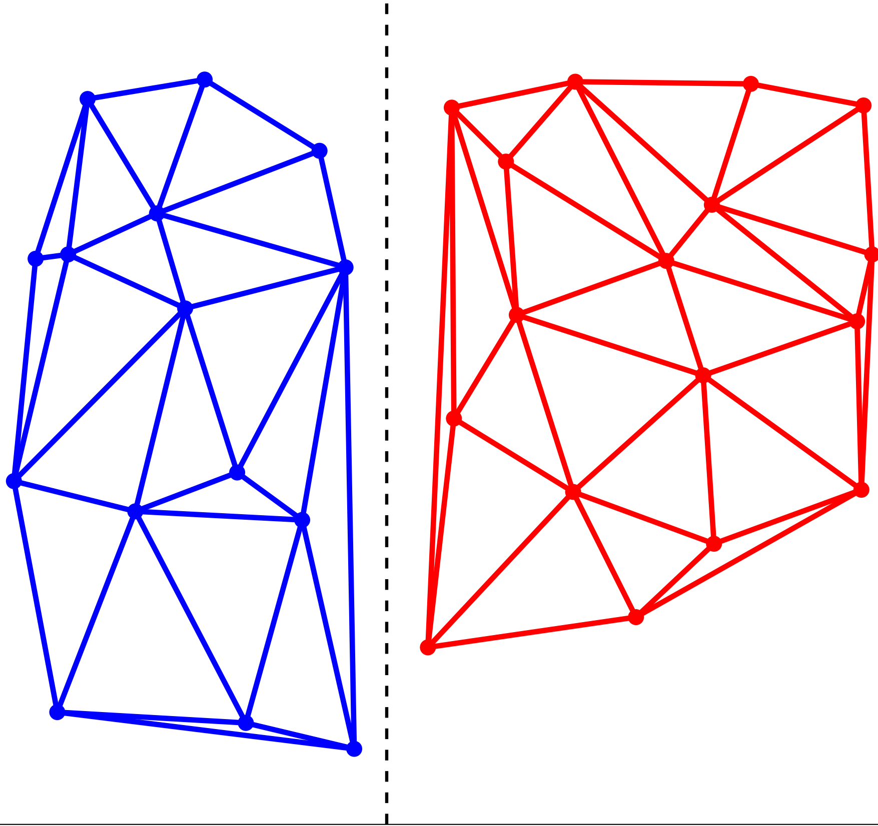
store all the medians in an array

Division

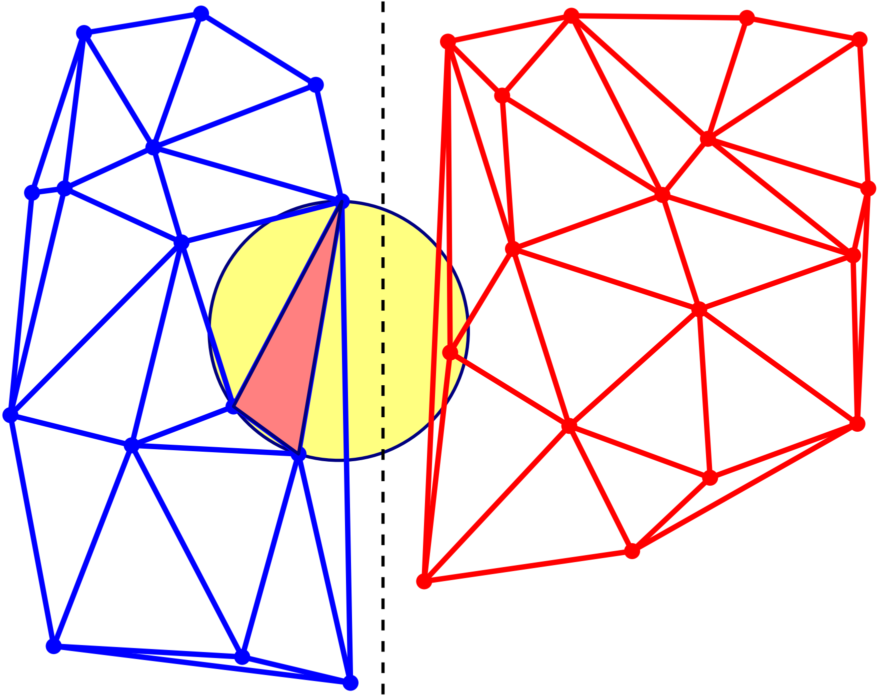


$O(n \log n)$
sort in x
store all the medians in an array
queries in $O(1)$

Fusion

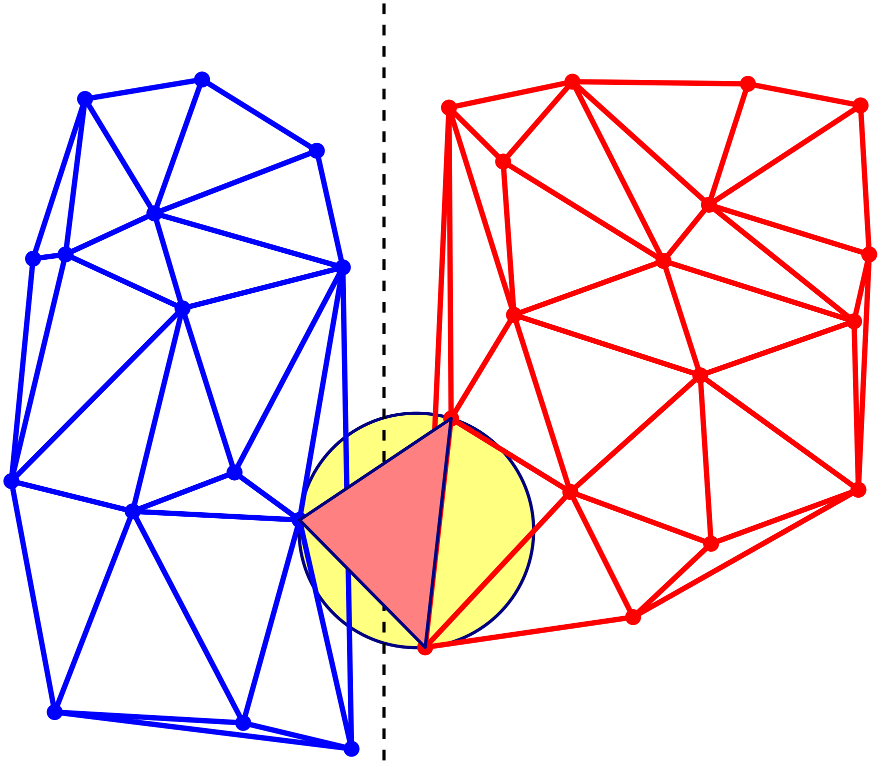


Fusion

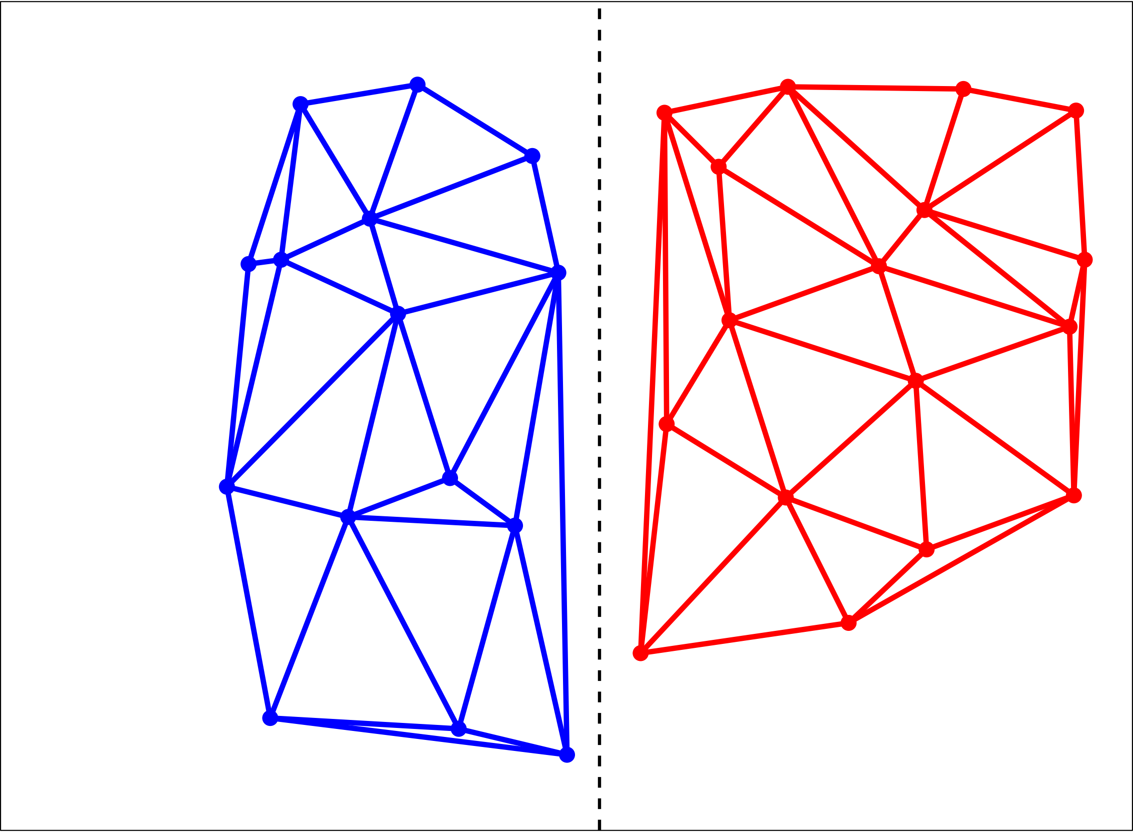


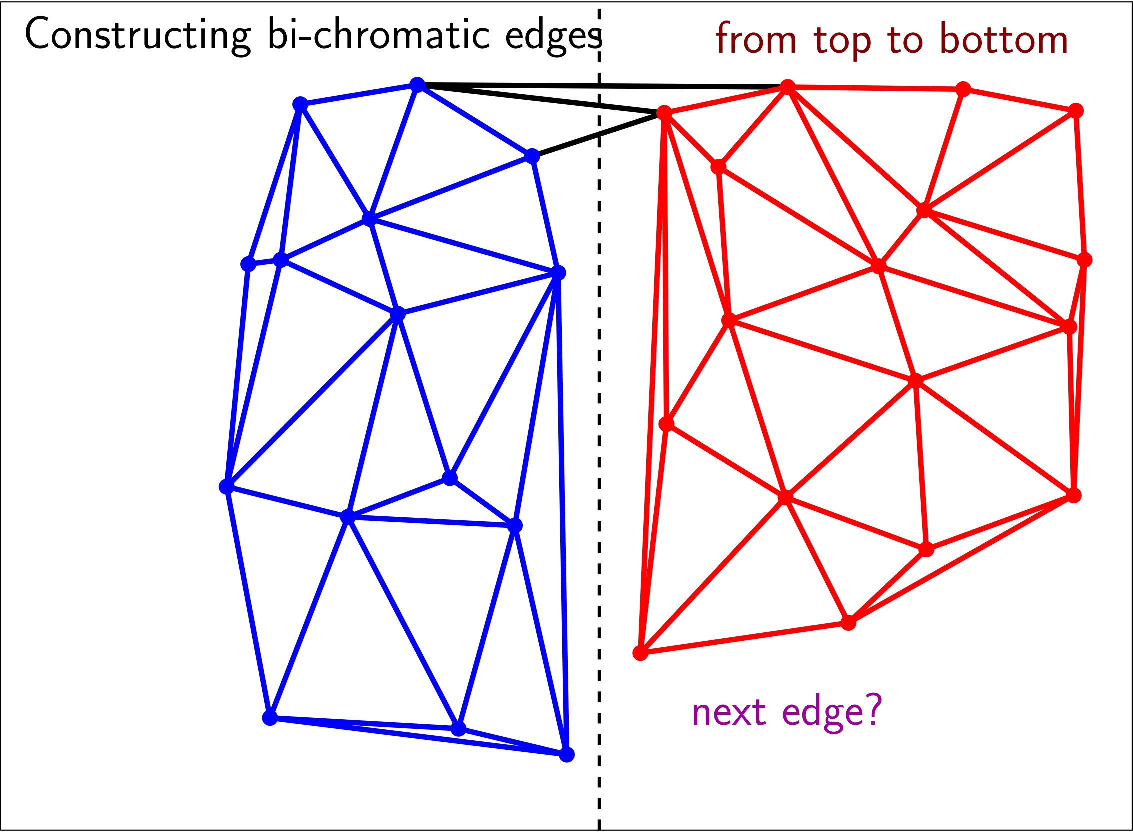
Monochromatic triangles to be deleted

Fusion

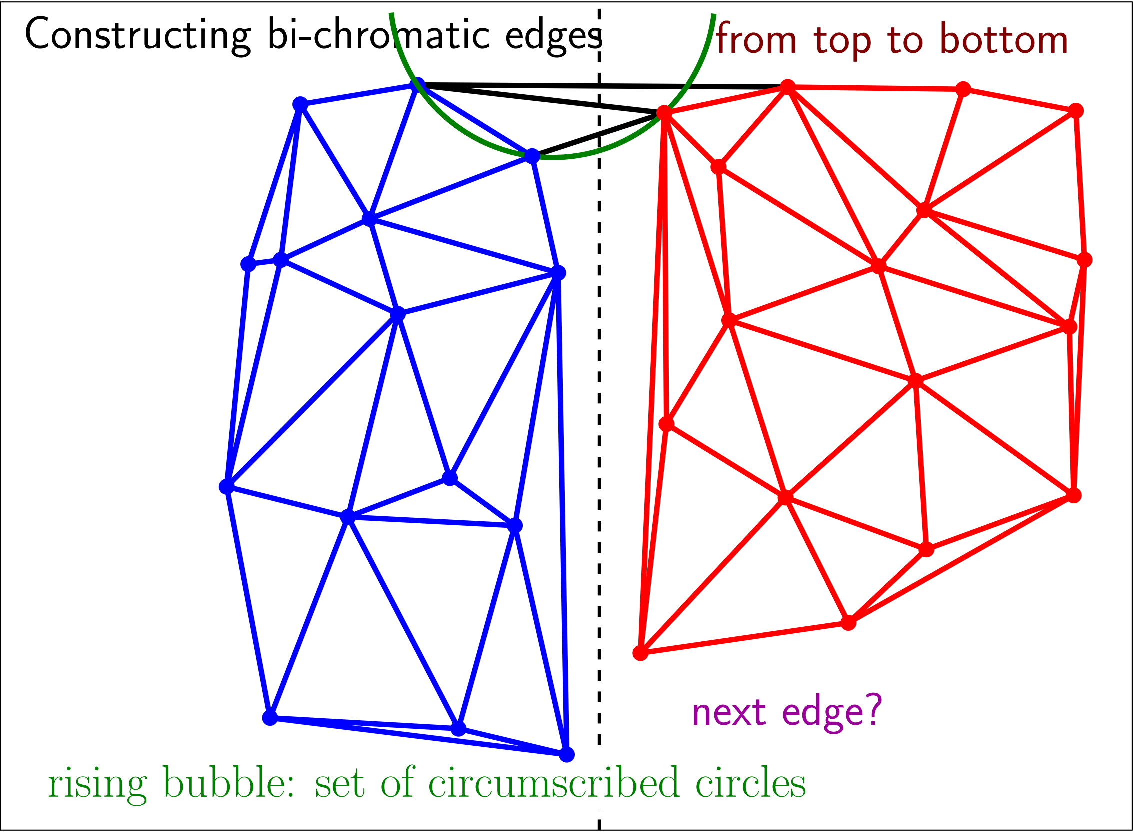


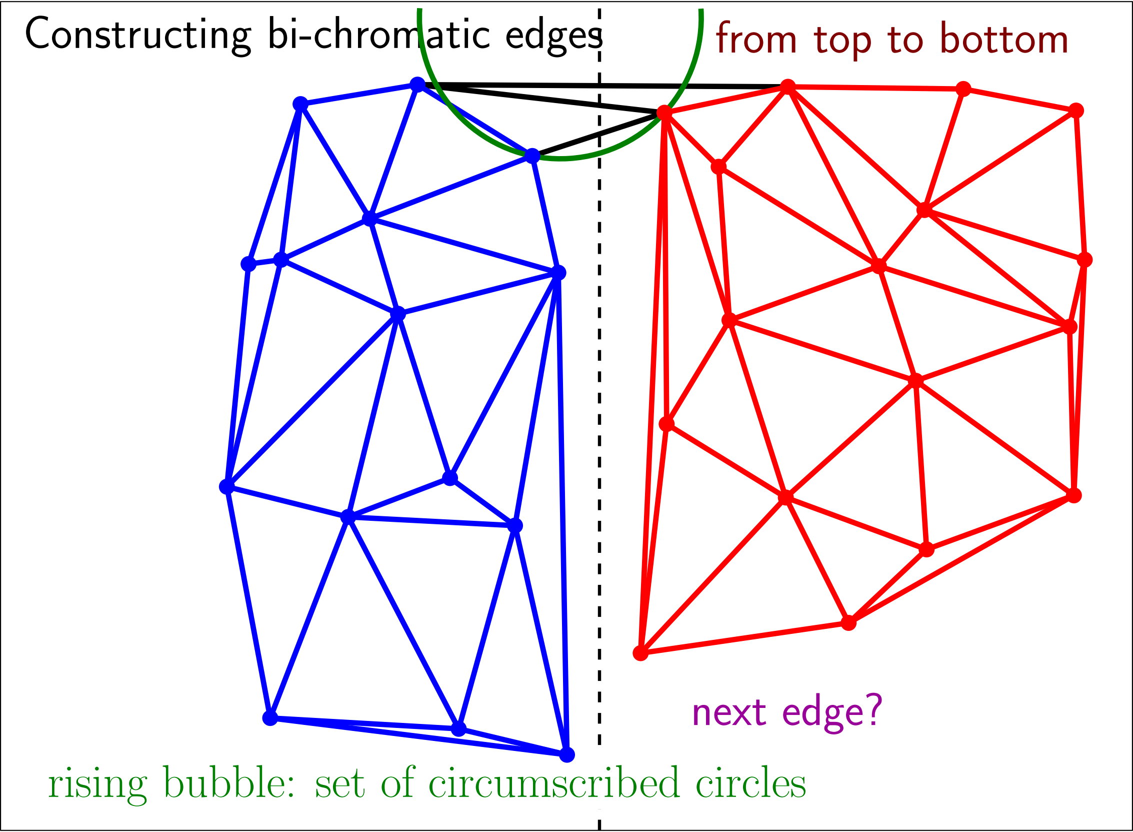
Bi-chromatic triangles to be construced





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

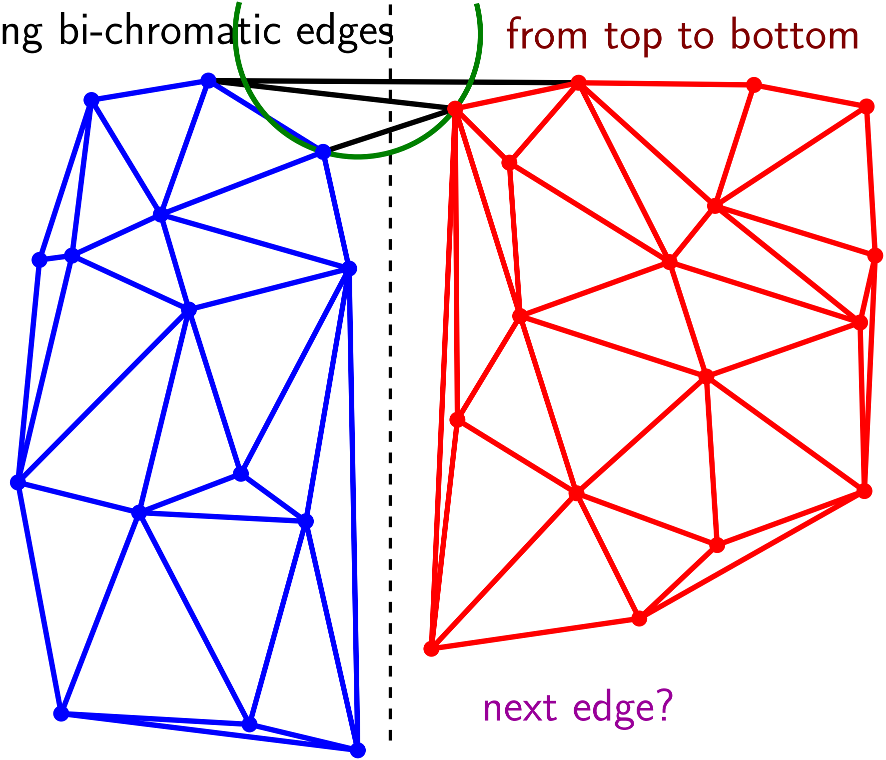

Constructing bi-chromatic edges
from top to bottom
next edge?
rising bubble: set of circumscribed circles



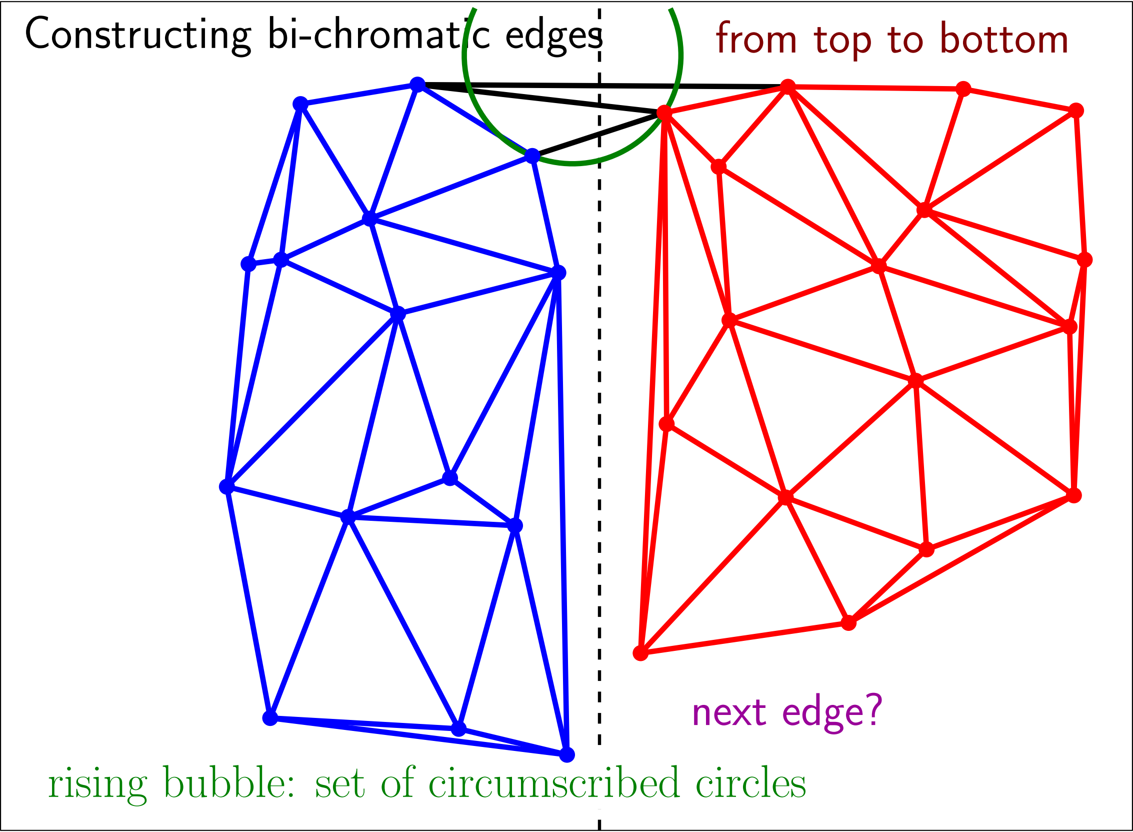
Constructing bi-chromatic edges
from top to bottom
next edge?
rising bubble: set of circumscribed circles

Constructing bi-chromatic edges

from top to bottom

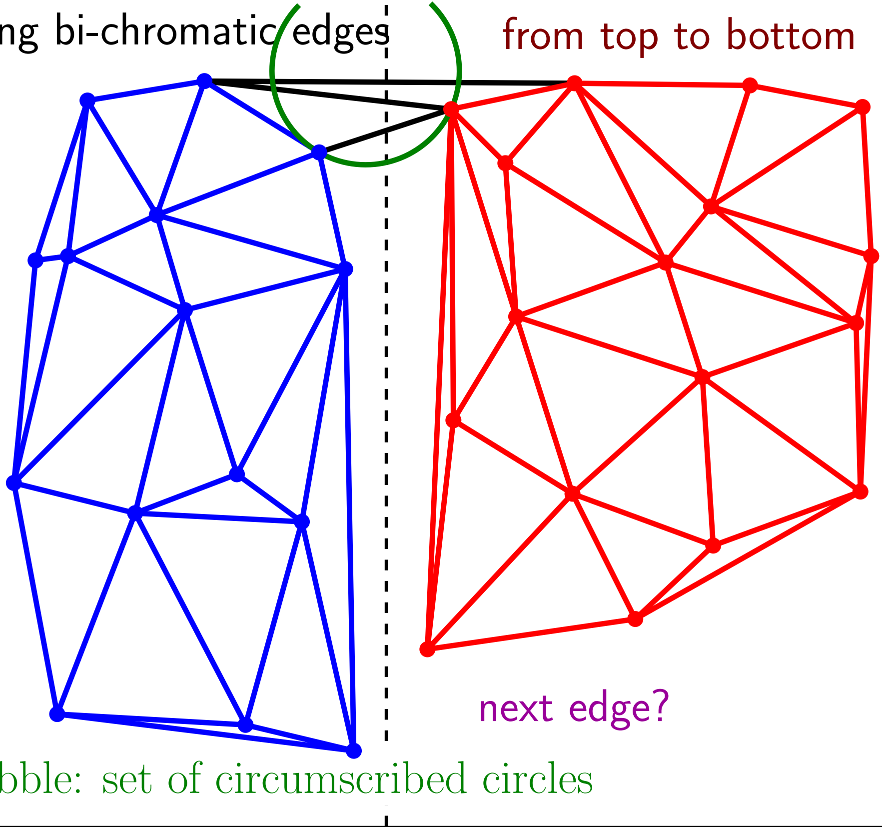


rising bubble: set of circumscribed circles



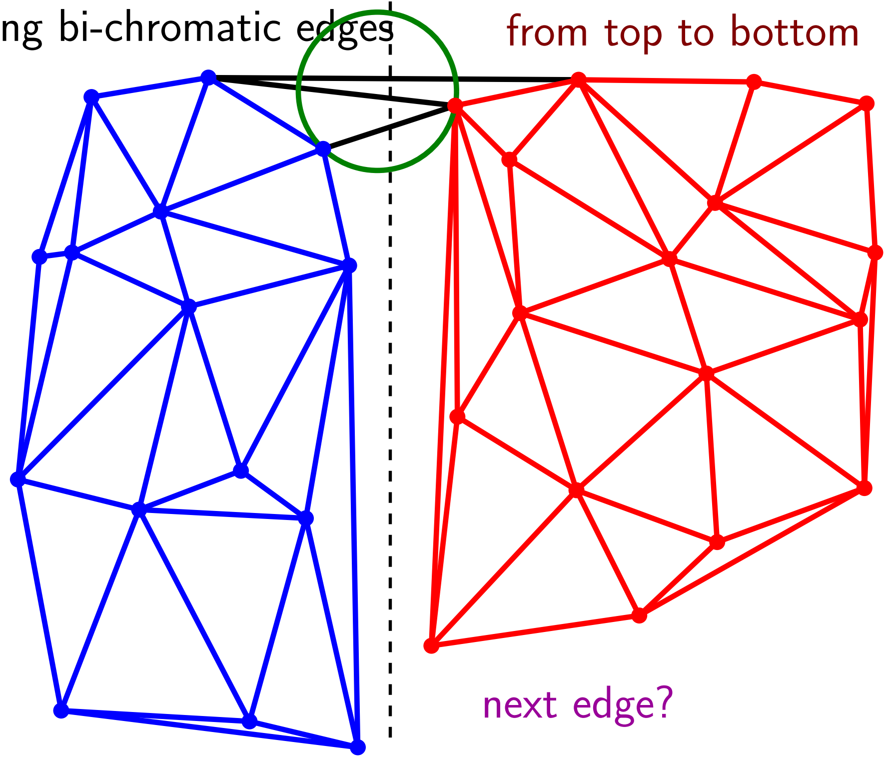
Constructing bi-chromatic edges
from top to bottom
next edge?
rising bubble: set of circumscribed circles

Constructing bi-chromatic edges from top to bottom



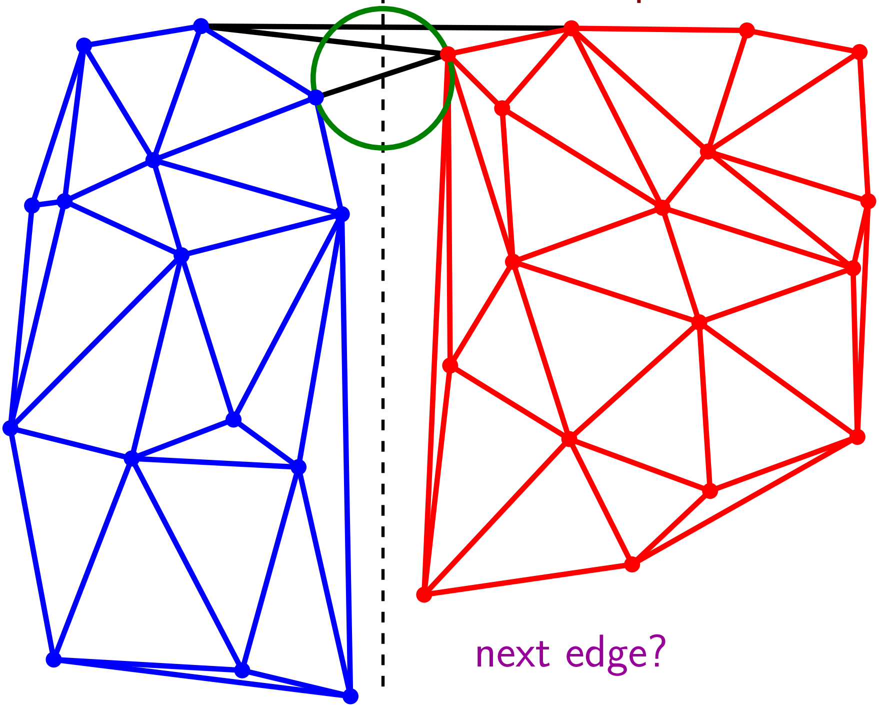
rising bubble: set of circumscribed circles

Constructing bi-chromatic edges



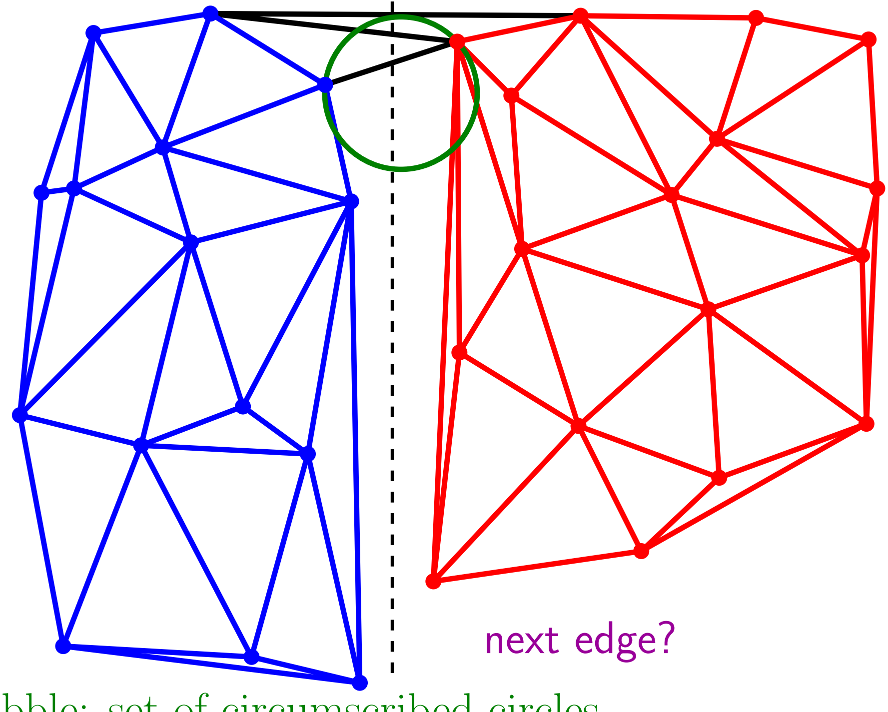
rising bubble: set of circumscribed circles

Constructing bi-chromatic edges from top to bottom



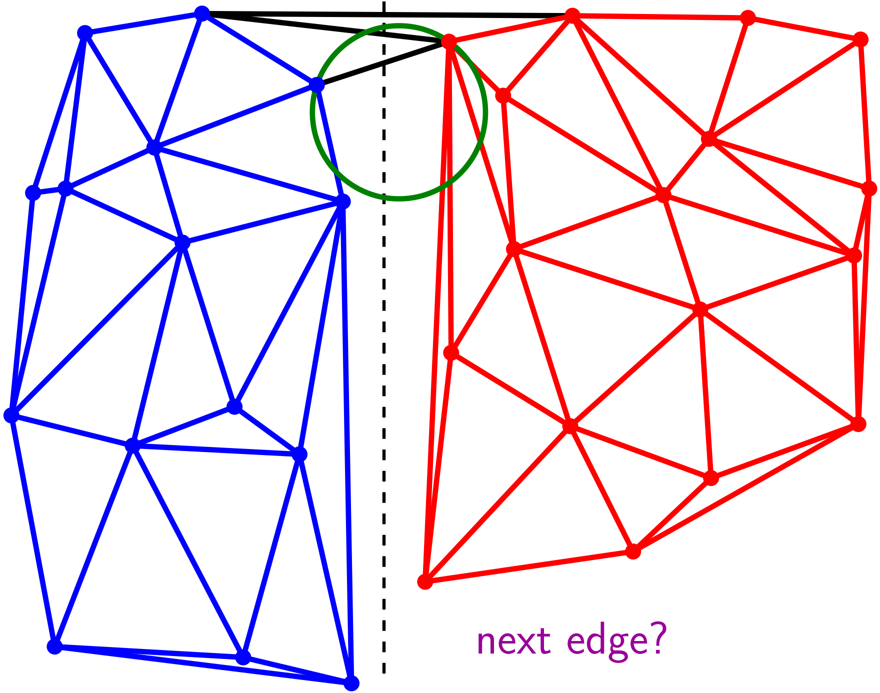
rising bubble: set of circumscribed circles

Constructing bi-chromatic edges from top to bottom



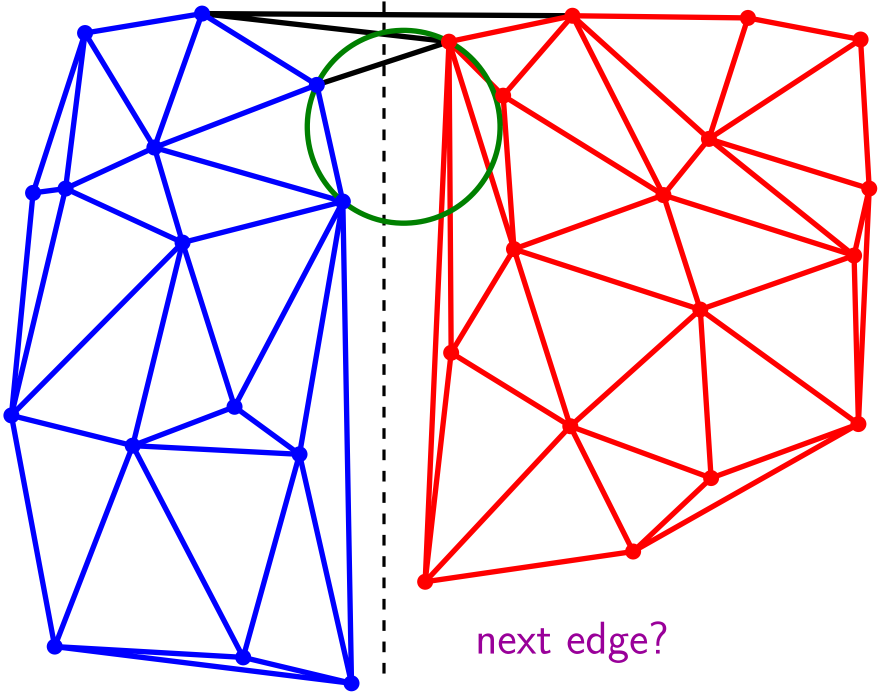
rising bubble: set of circumscribed circles

Constructing bi-chromatic edges from top to bottom



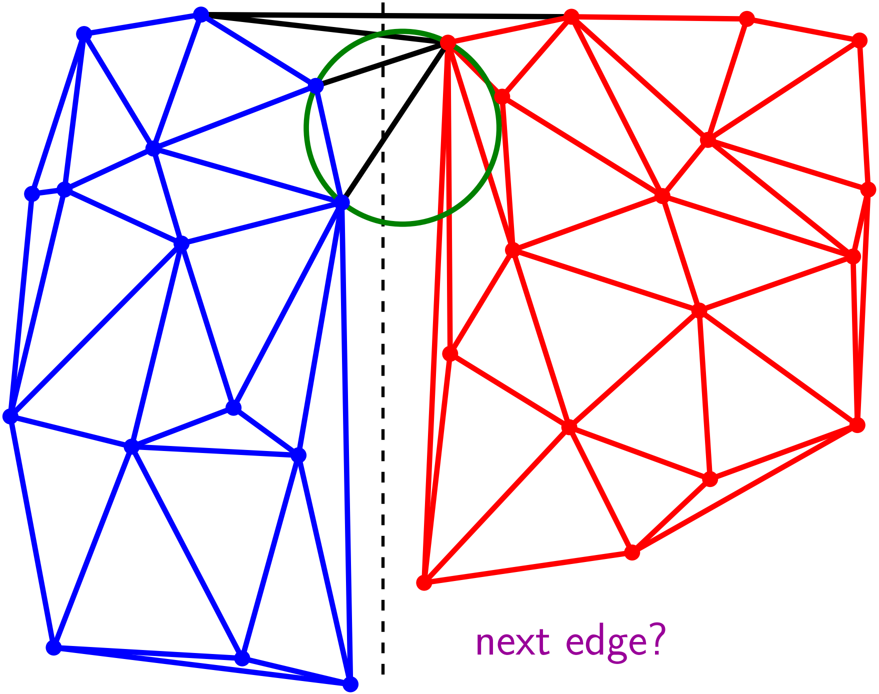
rising bubble: set of circumscribed circles

Constructing bi-chromatic edges



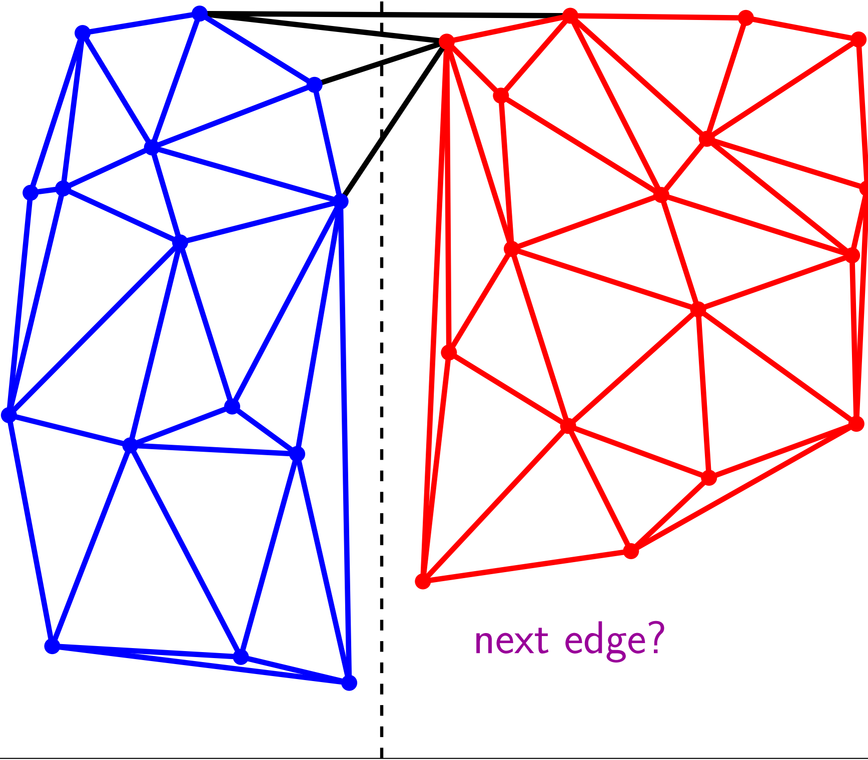
rising bubble: set of circumscribed circles

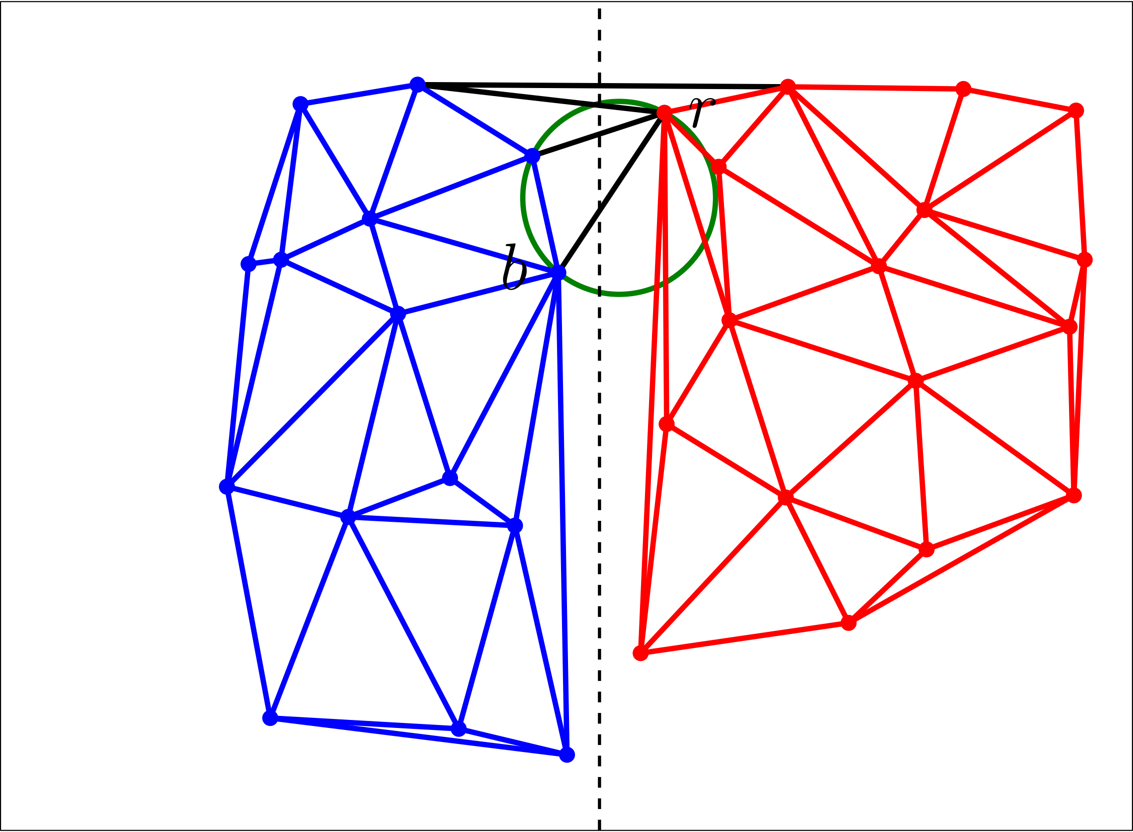
Constructing bi-chromatic edges from top to bottom



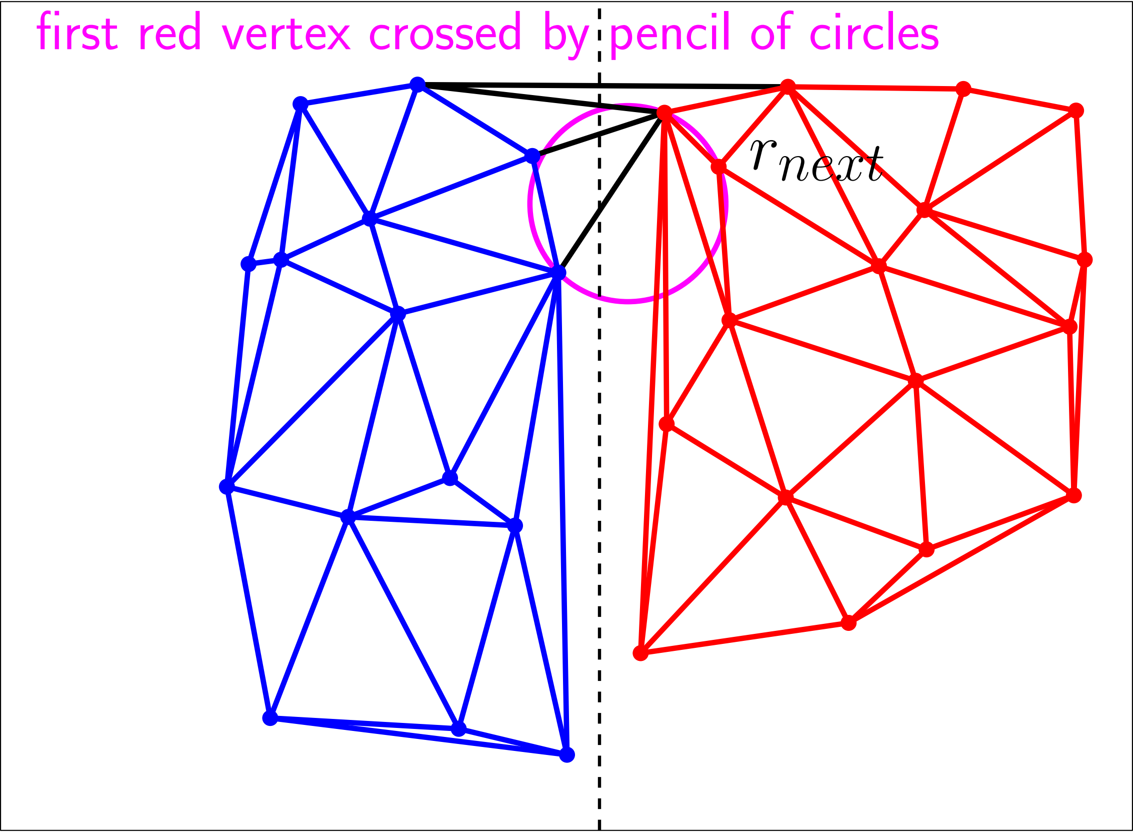
rising bubble: set of circumscribed circles

Constructing bi-chromatic edges from top to bottom

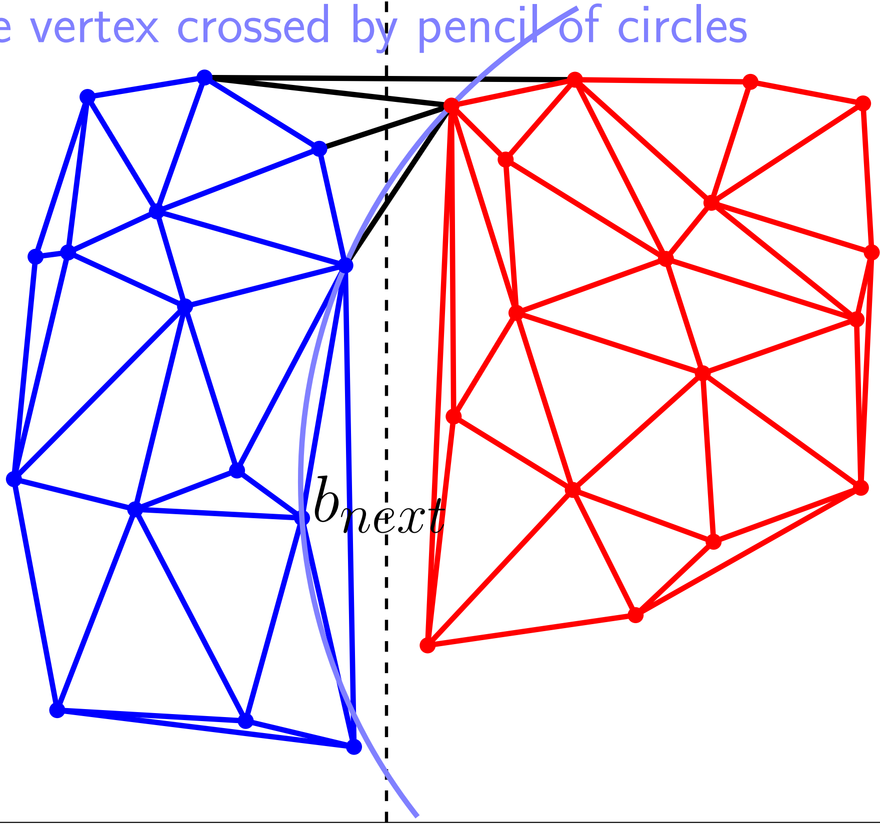




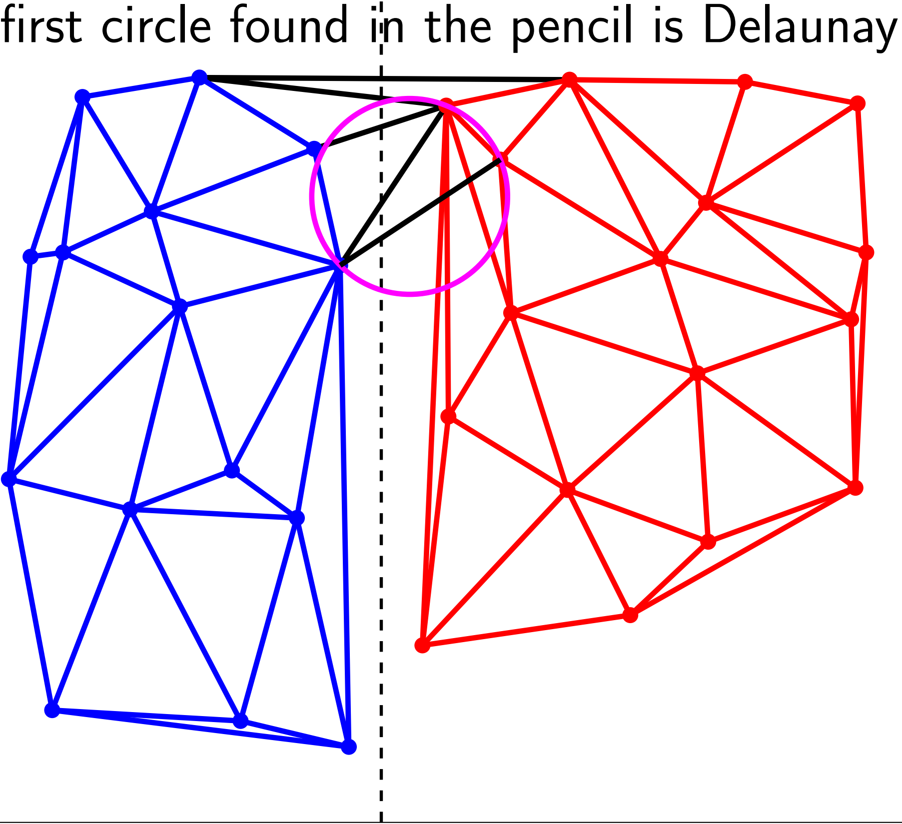
r
b

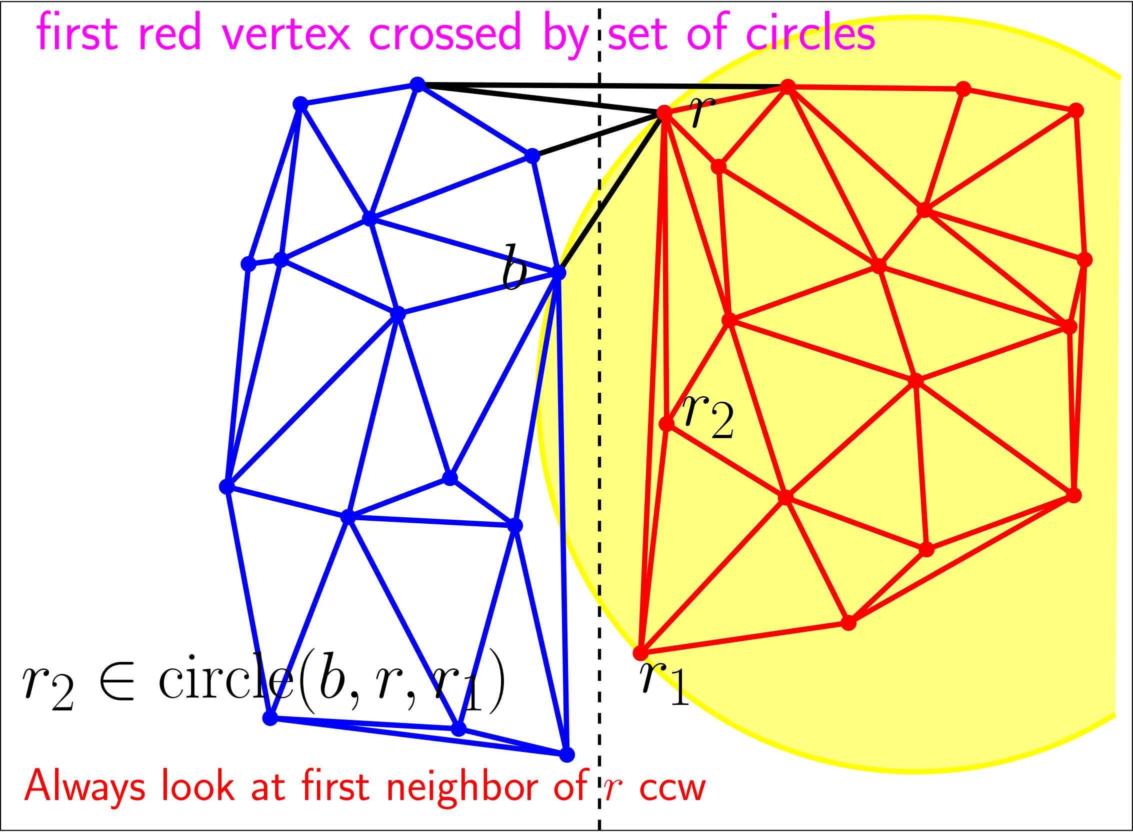

first red vertex crossed by pencil of circles
r_{next}

first blue vertex crossed by pencil of circles



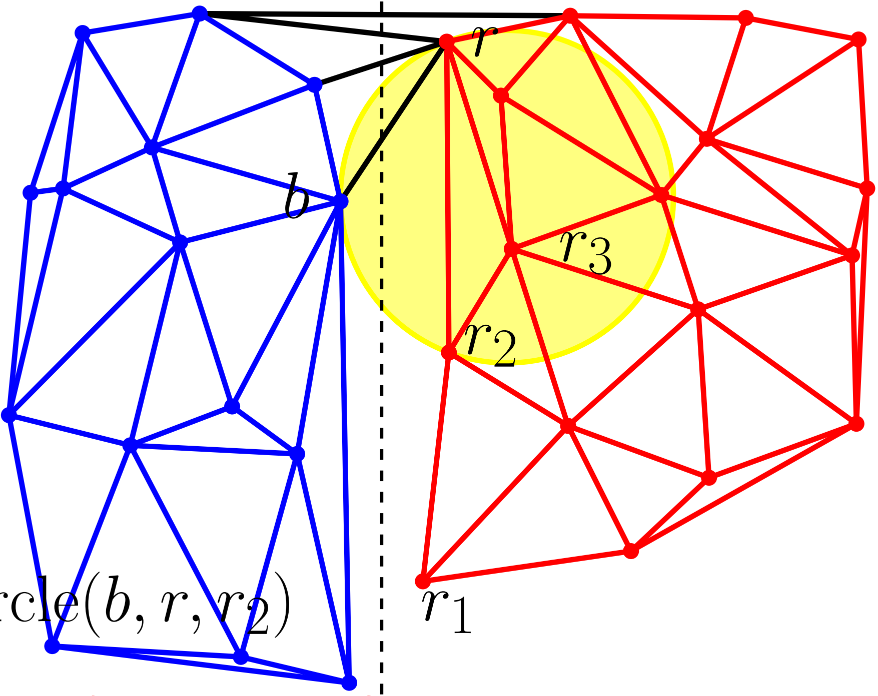
Only the first circle found in the pencil is Delaunay





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

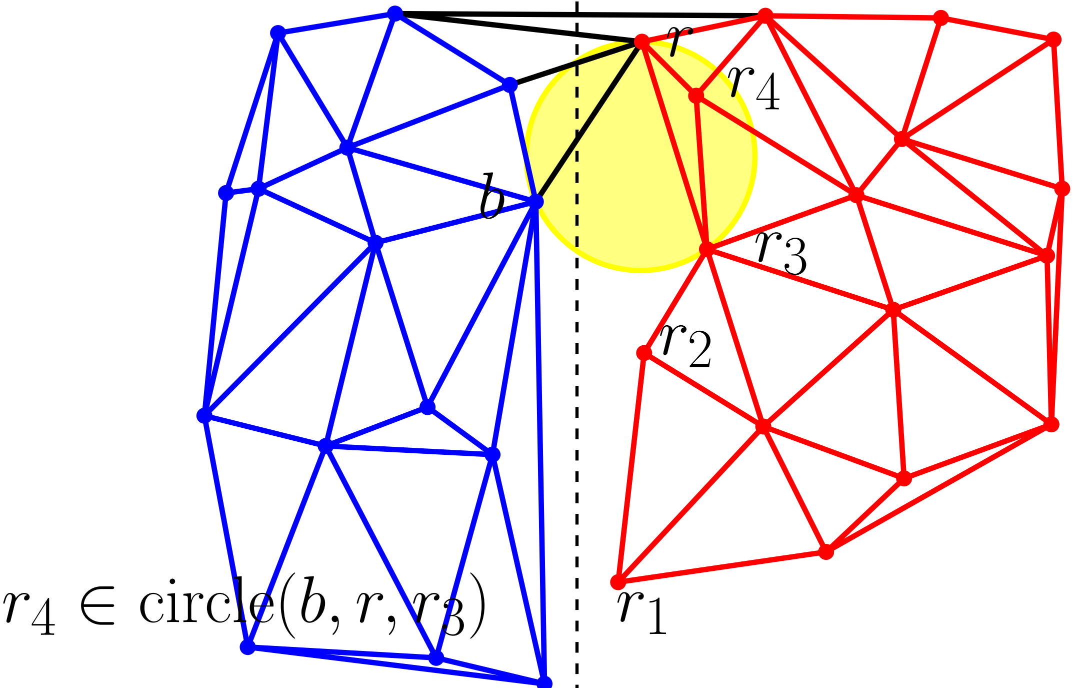
first red vertex crossed by set of circles



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

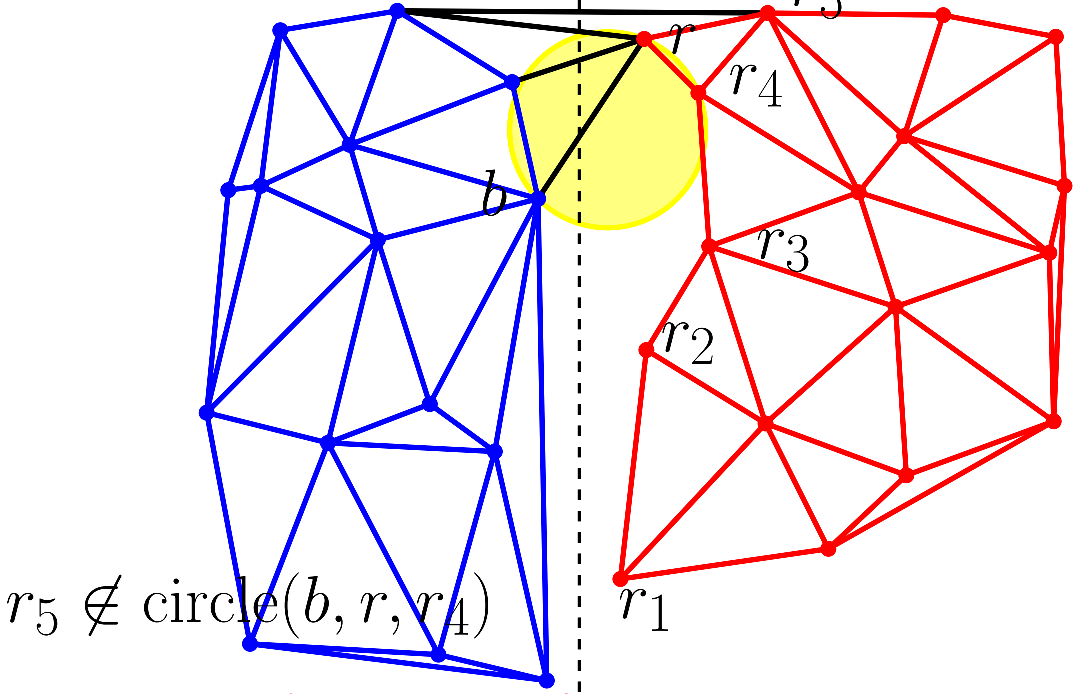
Always look at first neighbor of r ccw

first red vertex crossed by set of circles



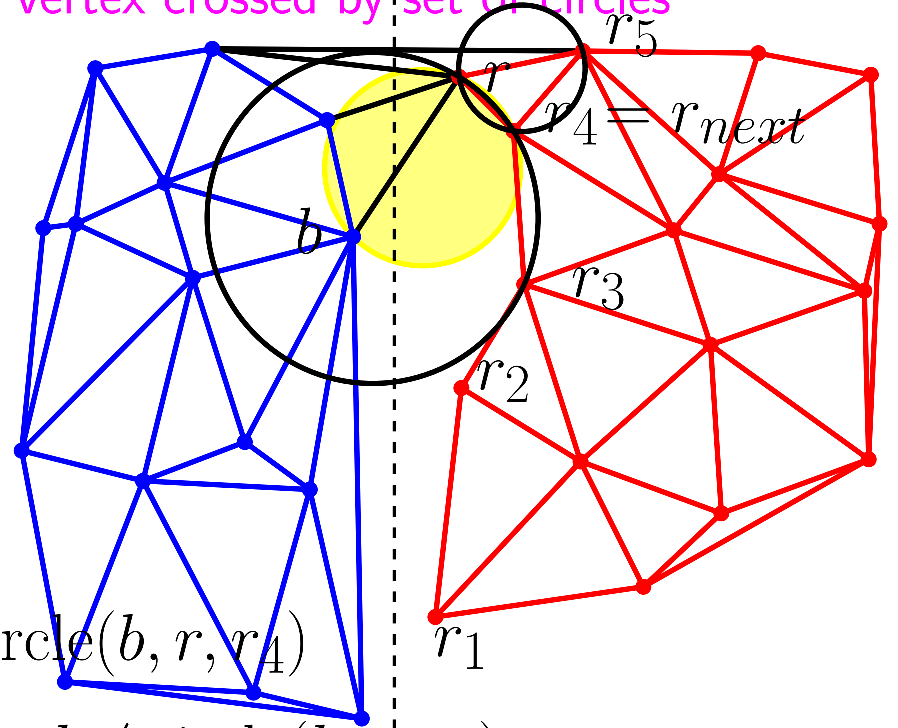
Always look at first neighbor of r ccw

first red vertex crossed by set of circles



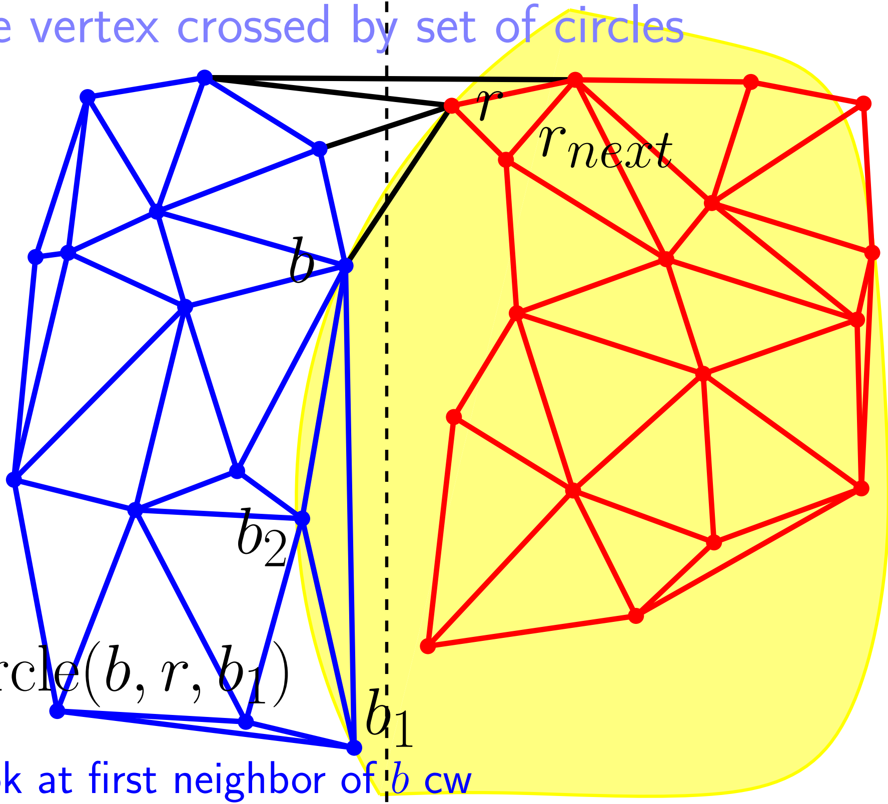
Always look at first neighbor of r ccw

first red vertex crossed by set of circles



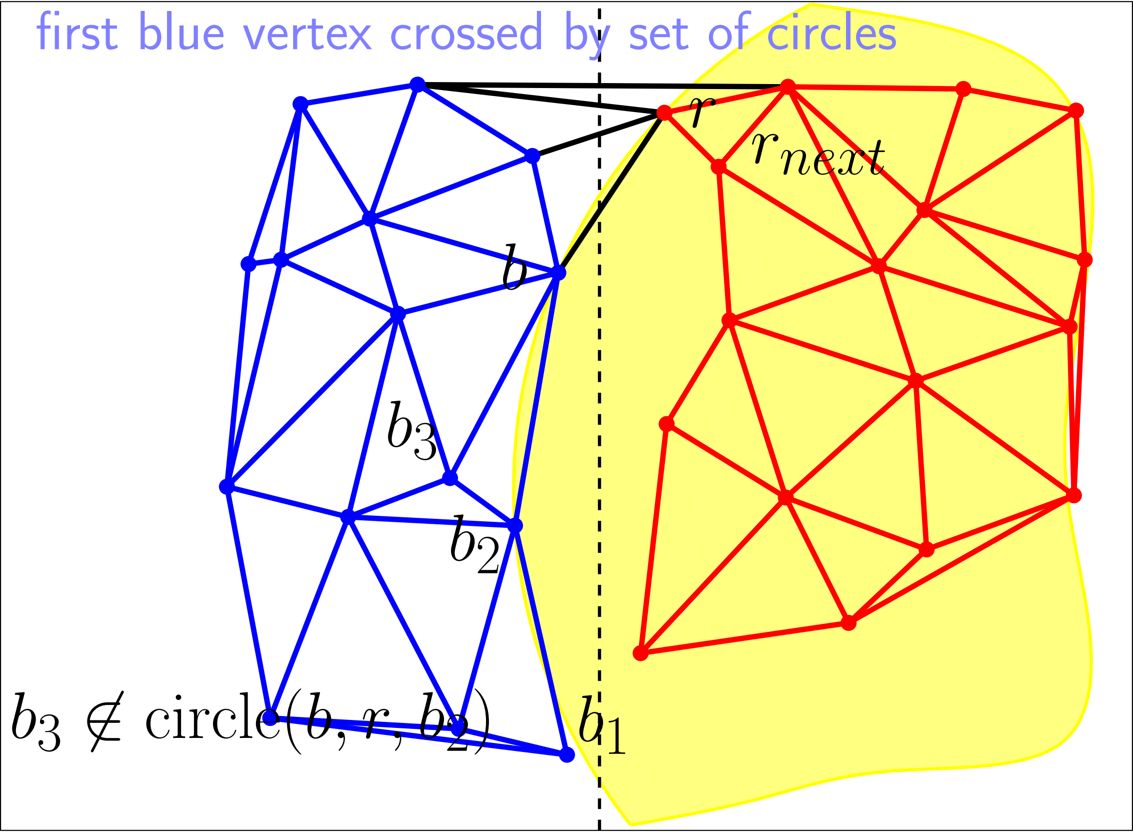
$r_5 \notin \text{circle}(b, r, r_4)$

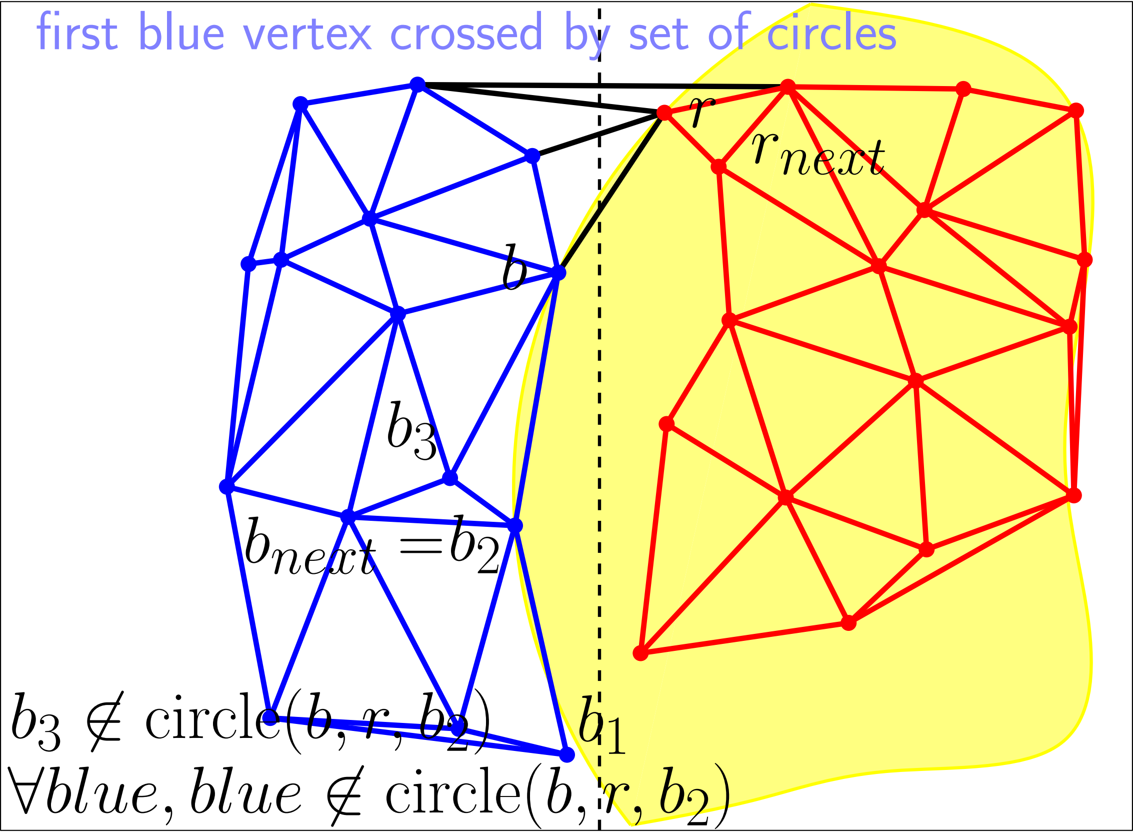
$\forall red, red \notin \text{circle}(b, r, r_4)$ (black disks are Delaunay for red set) first blue vertex crossed by set of circles

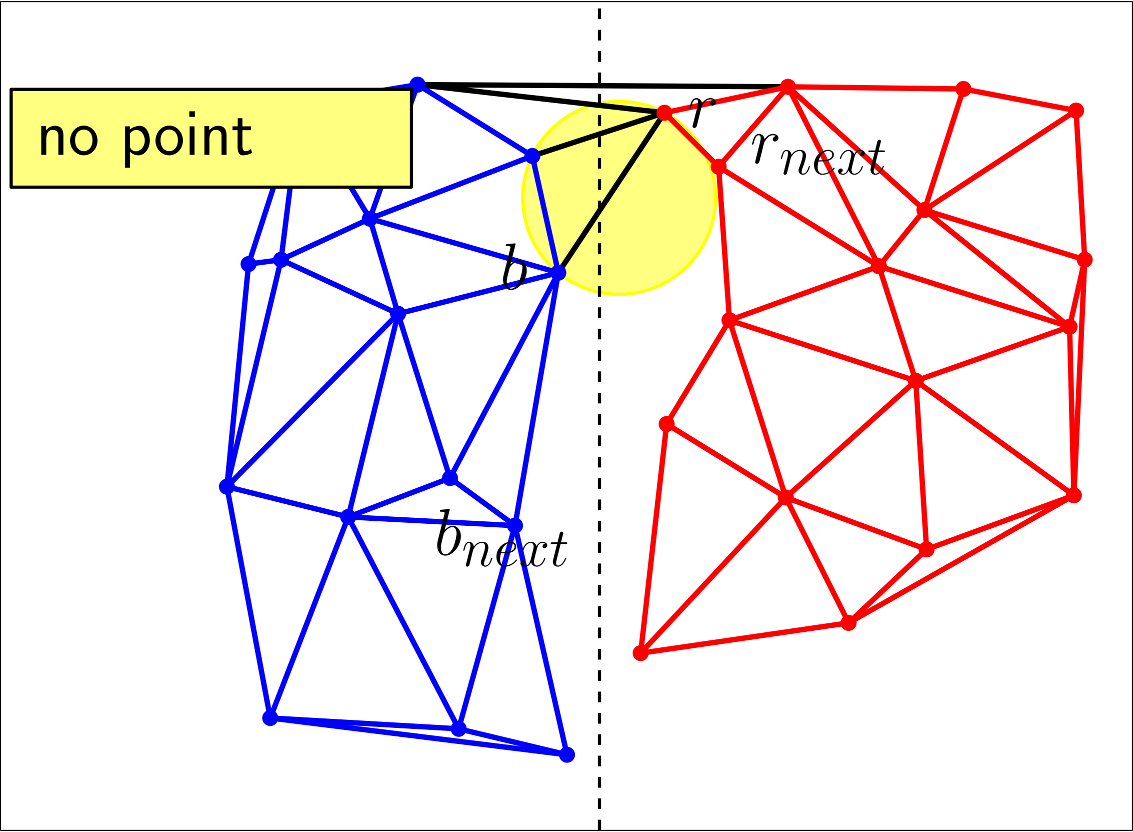


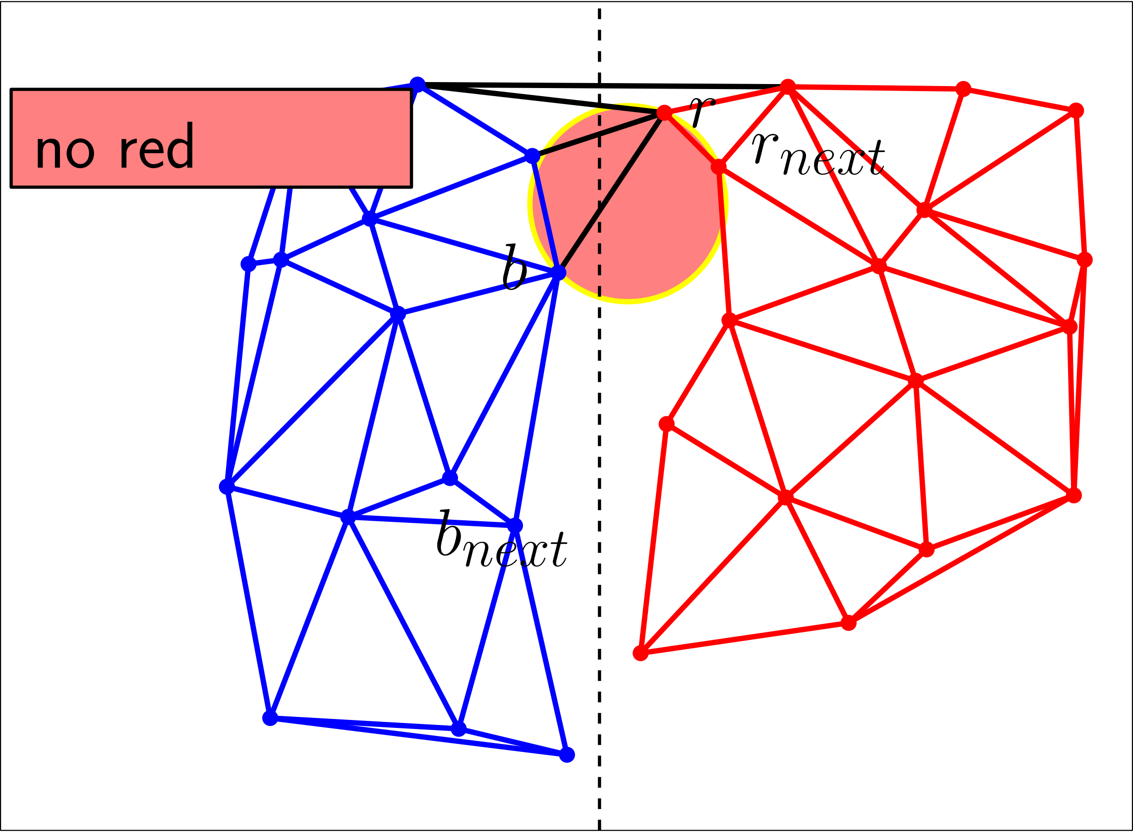
$$b_2 \in \mathrm{circle}(b, r, b_1)$$

Always look at first neighbor of b cw

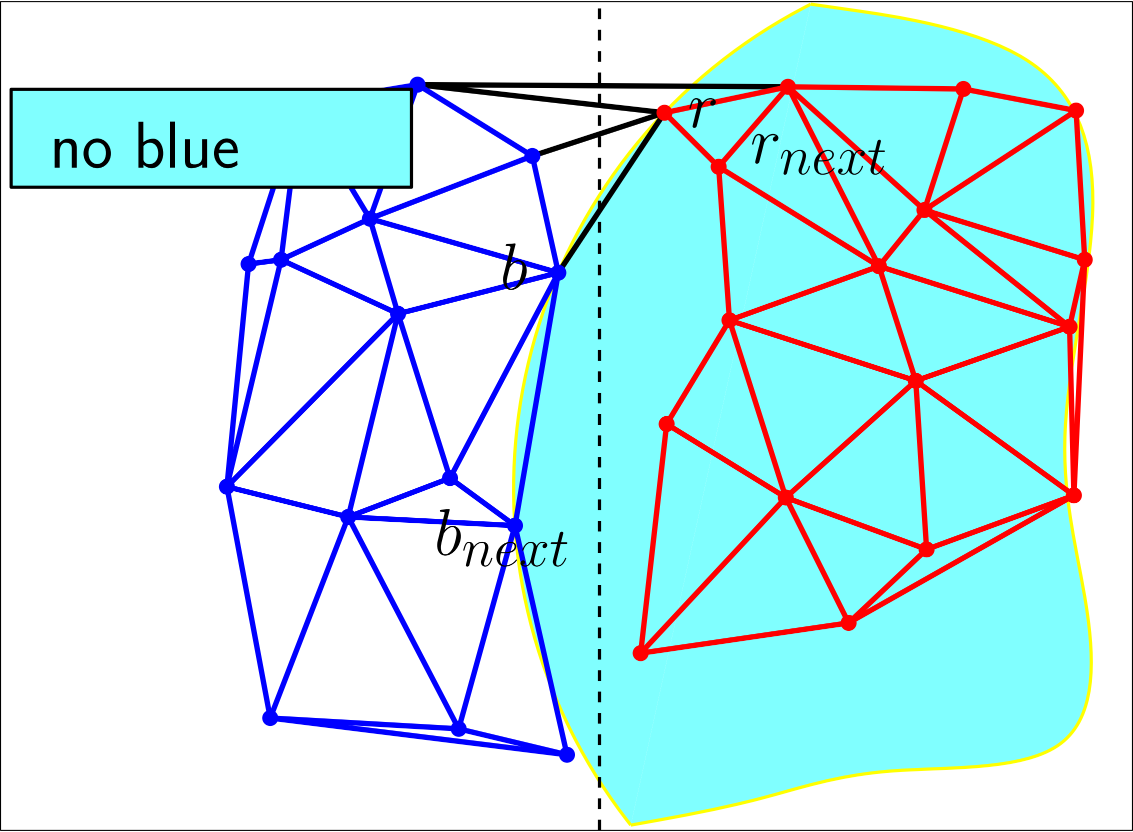

first blue vertex crossed by set of circles
r
r_{next}
b
b_3
b_2
b_1
$b_3 \notin \text{circle}(b, r, b_2)$

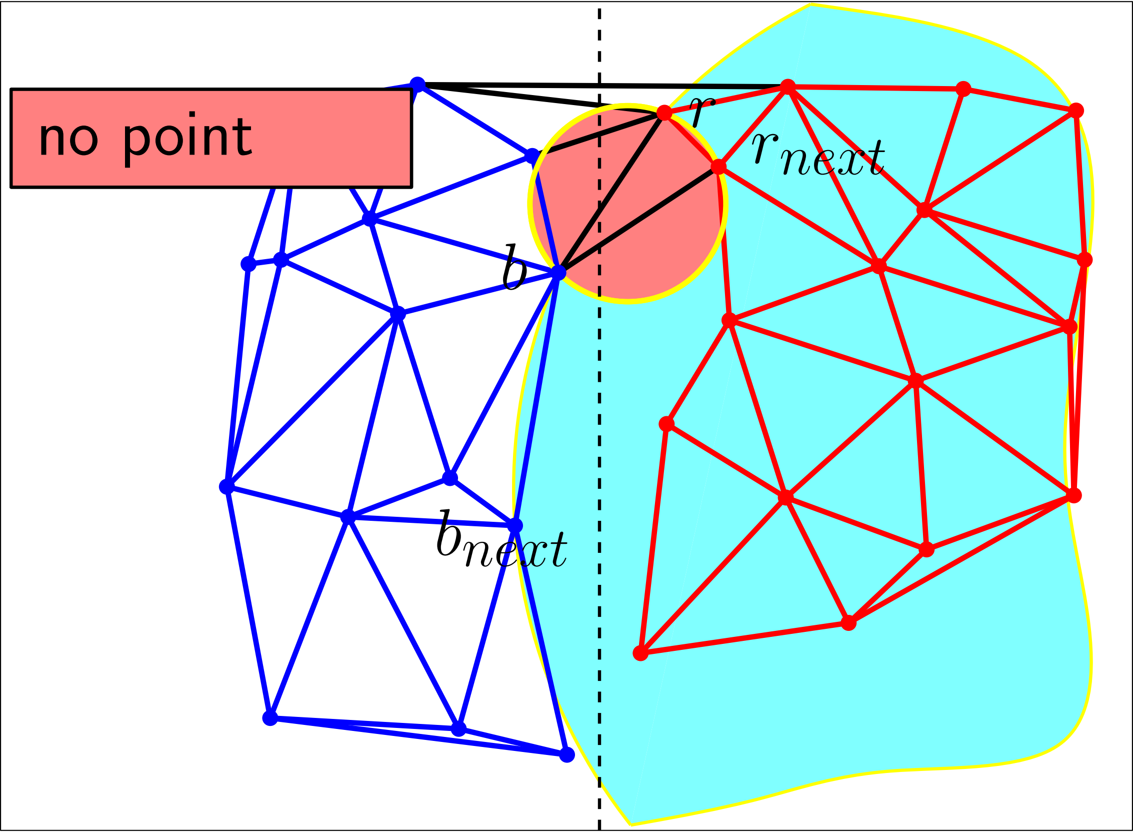

first blue vertex crossed by set of circles
r
r_{next}
b
b_3
$b_{next} = b_2$
b_1
$b_3 \notin \mathrm{circle}(b, r, b_2)$
$\forall blue, blue \notin \mathrm{circle}(b, r, b_2)$


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

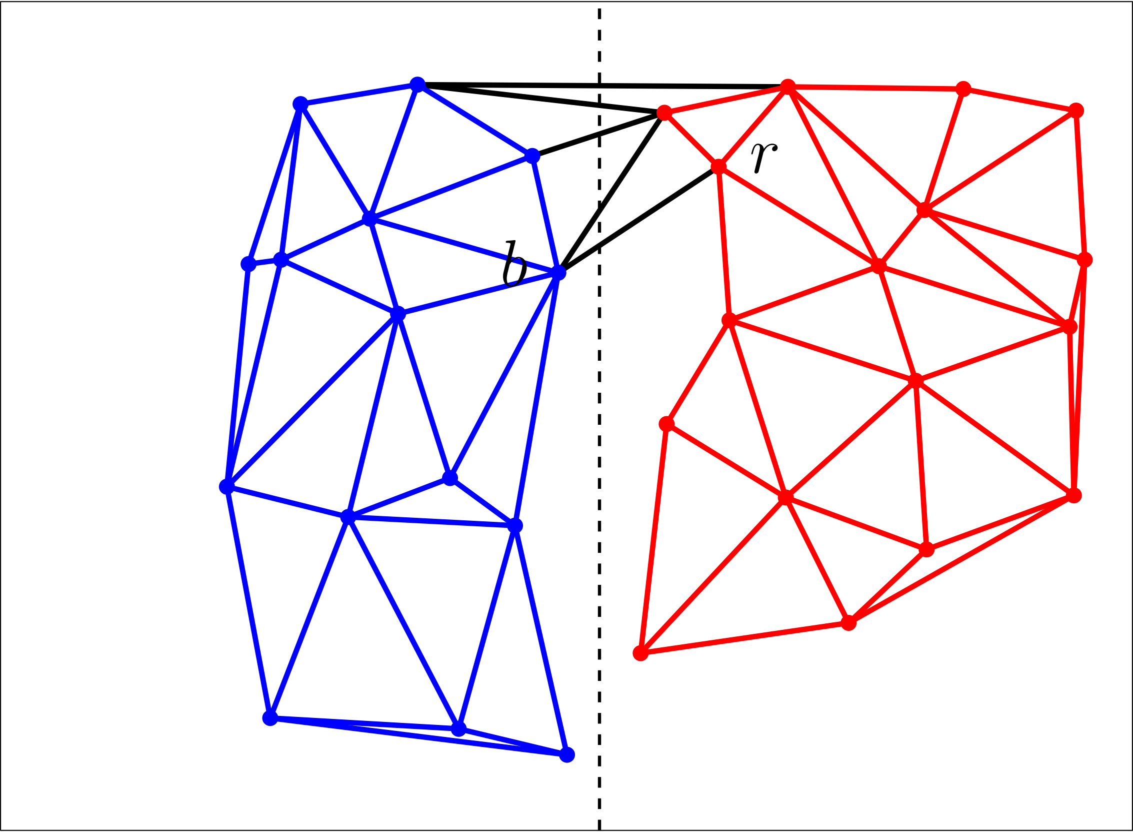


no red
r
r_next
b
b_next

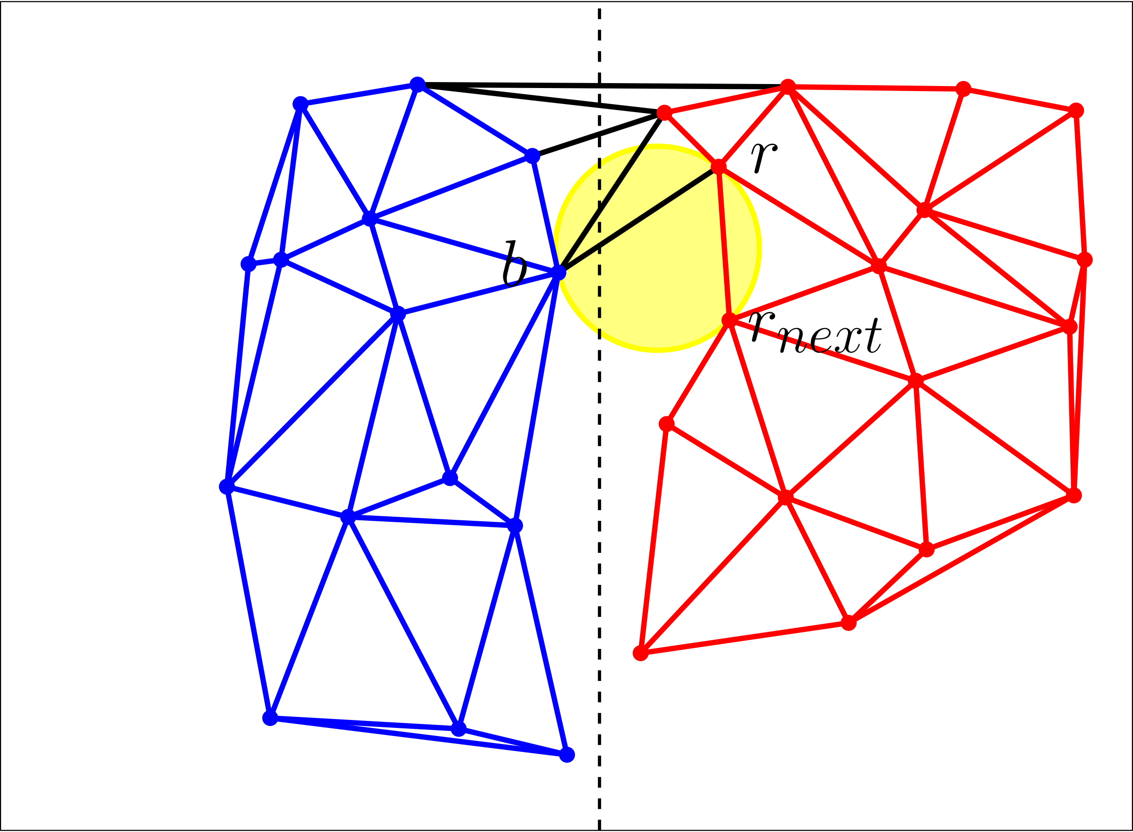

no blue
r
r_{next}
b
b_{next}



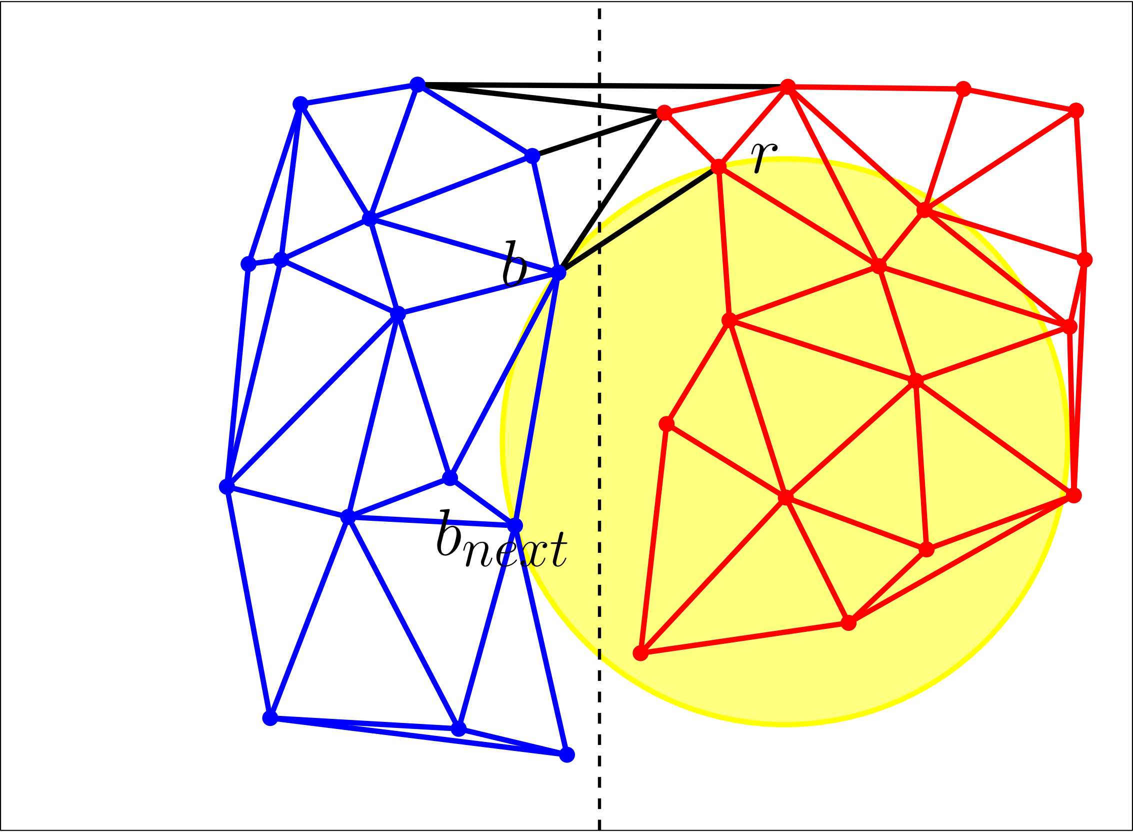
no point
r
r_{next}
b
b_{next}



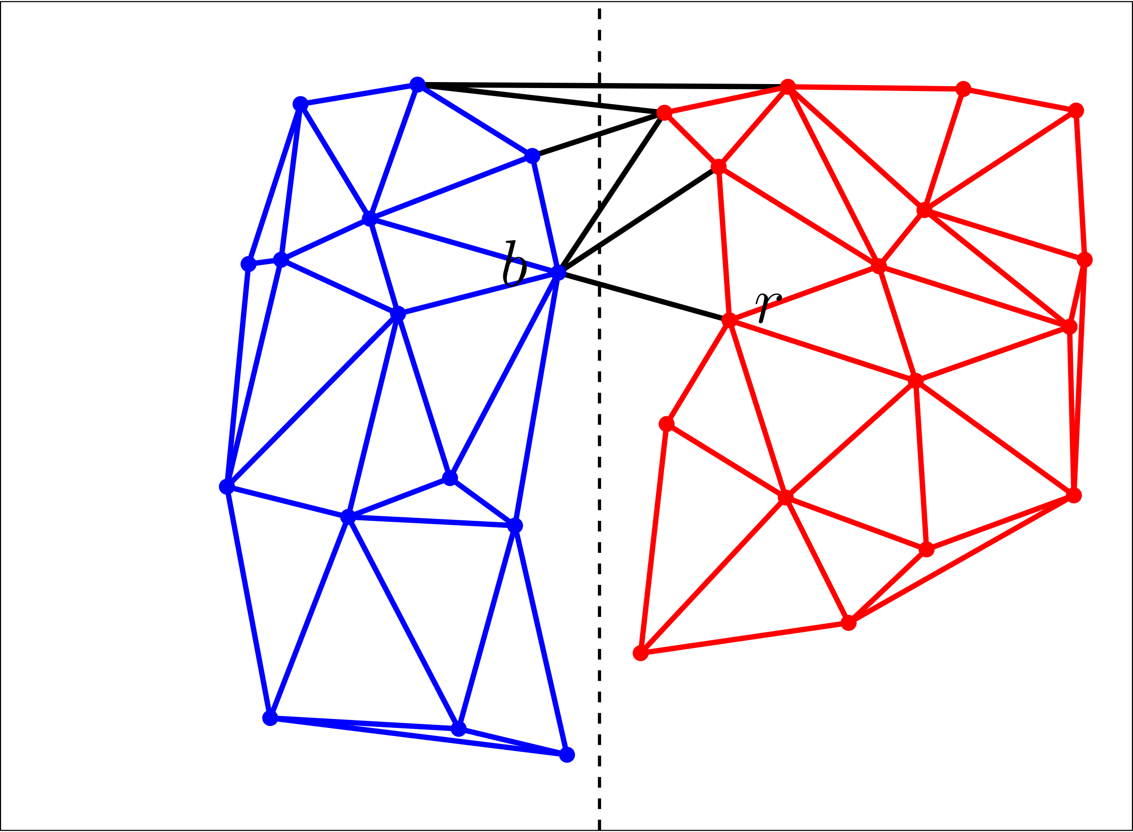
b
r



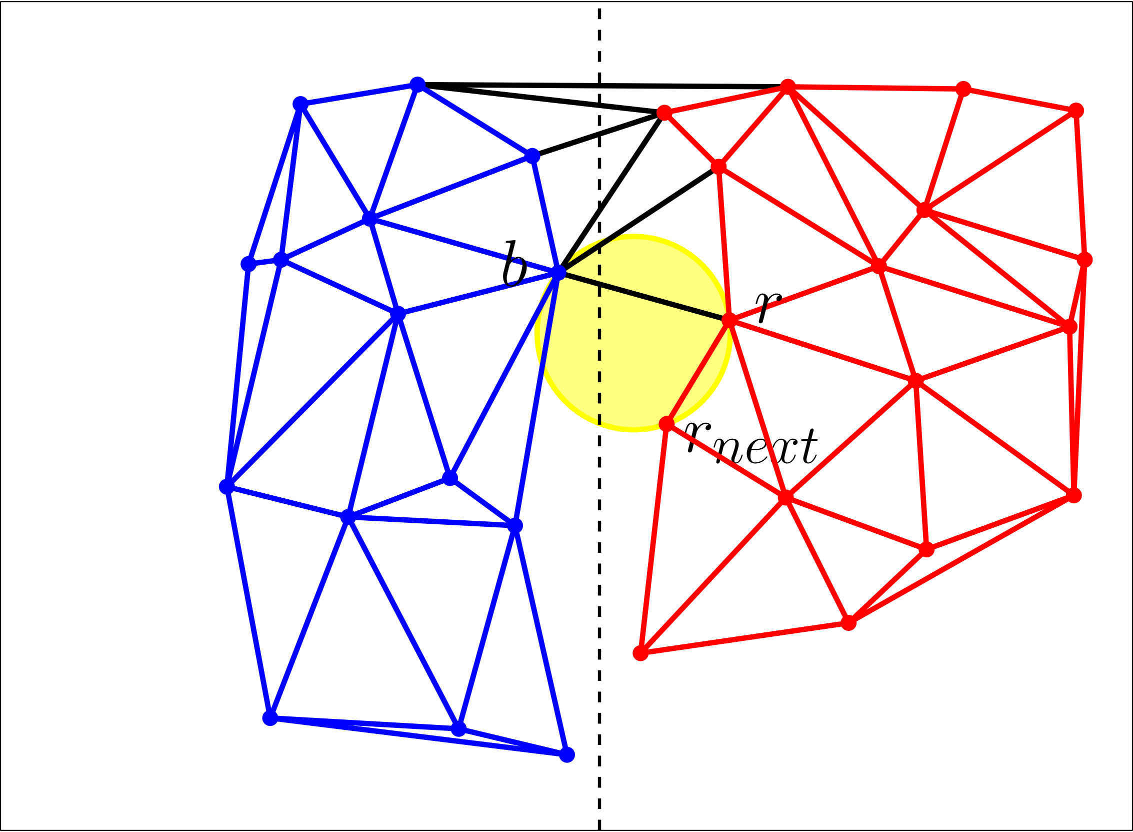
r
b
r_next



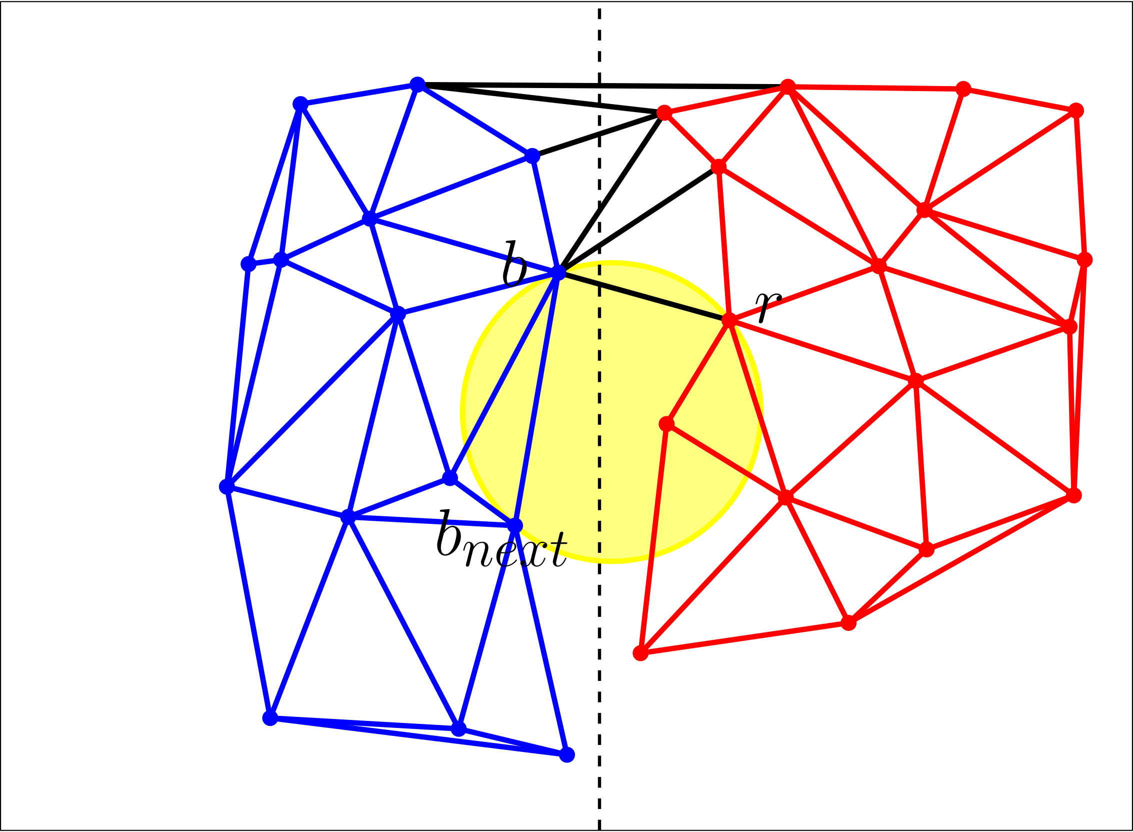
r
b
b_{next}



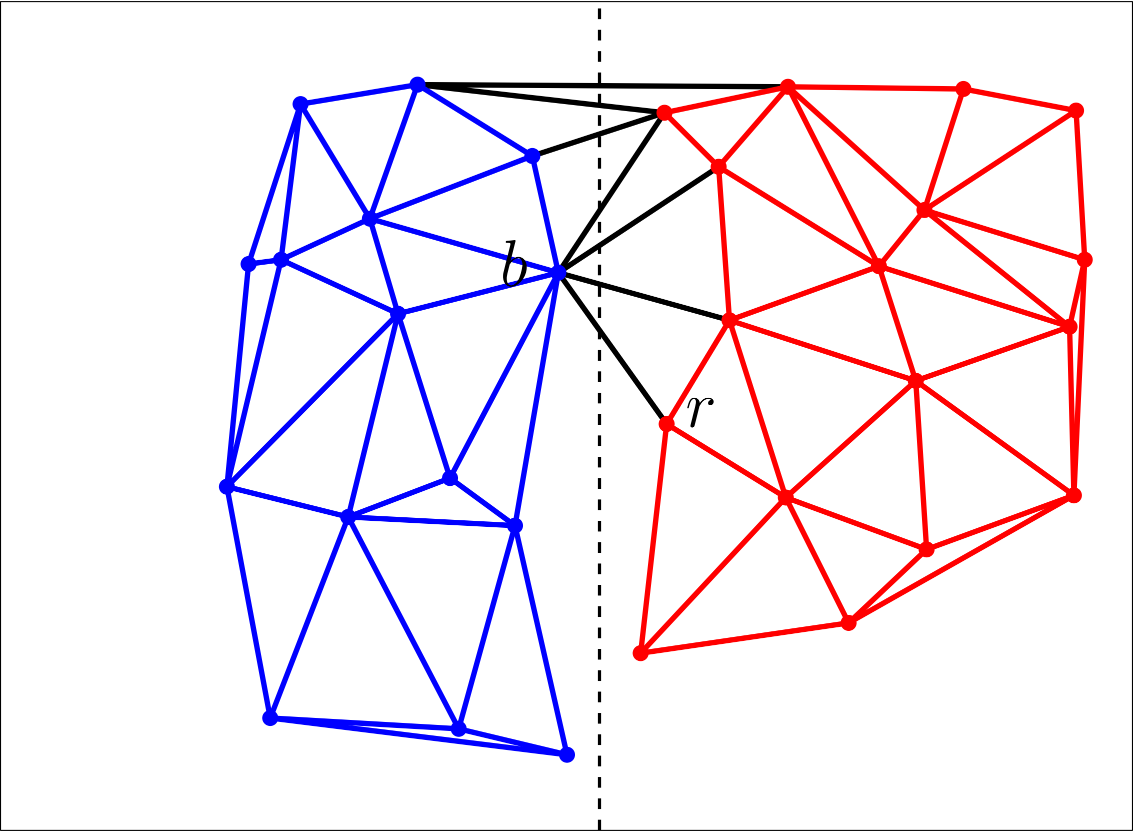
b
r



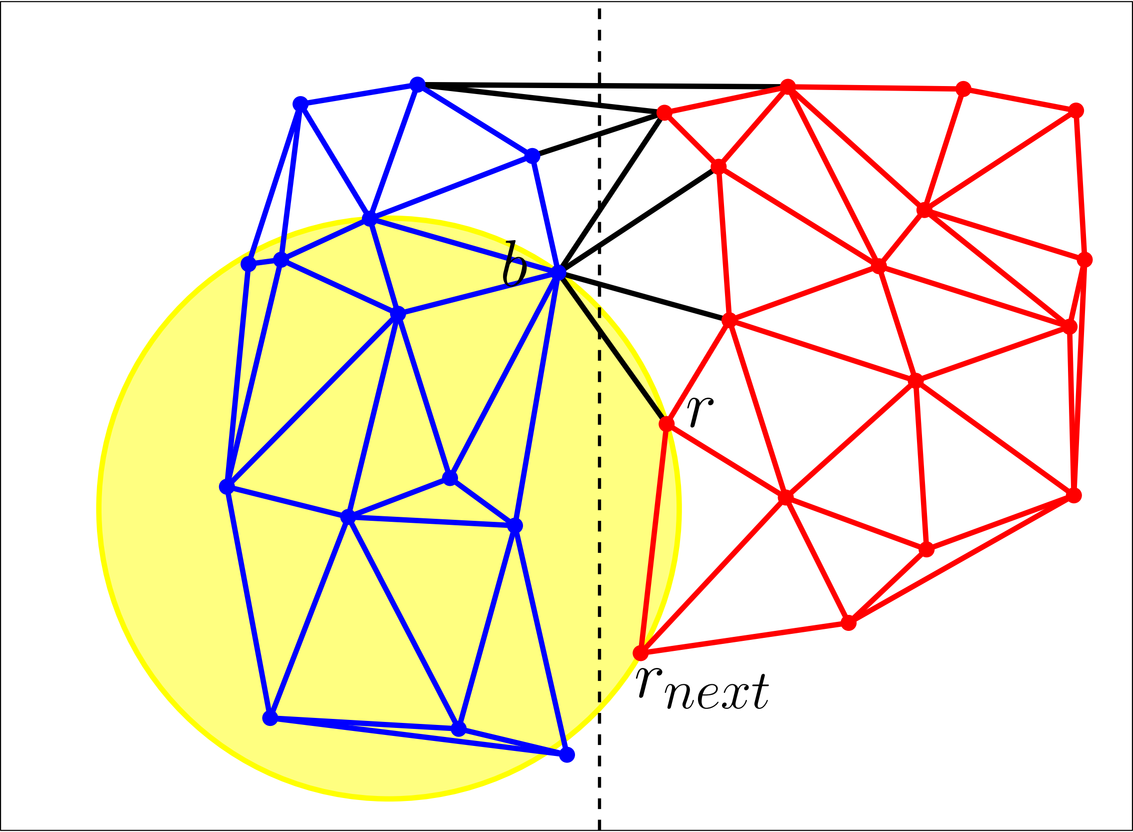
b
r
r_{next}



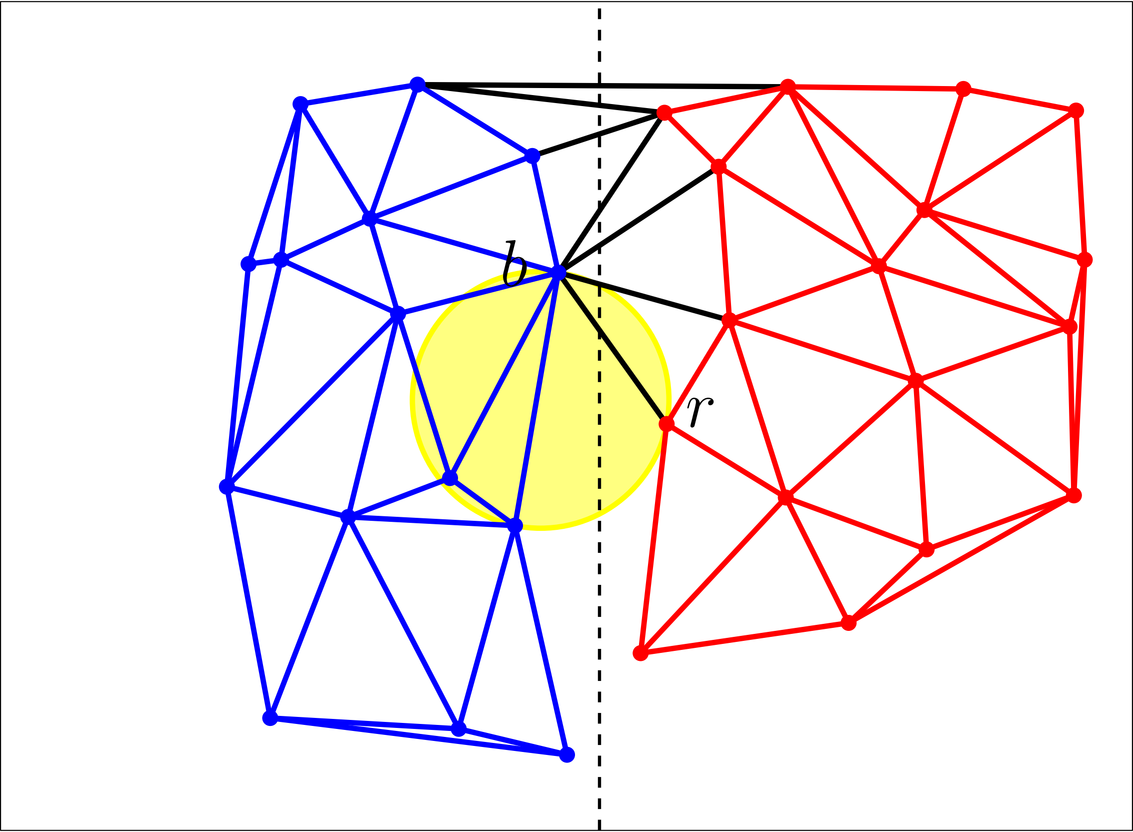
b
r
b_{next}



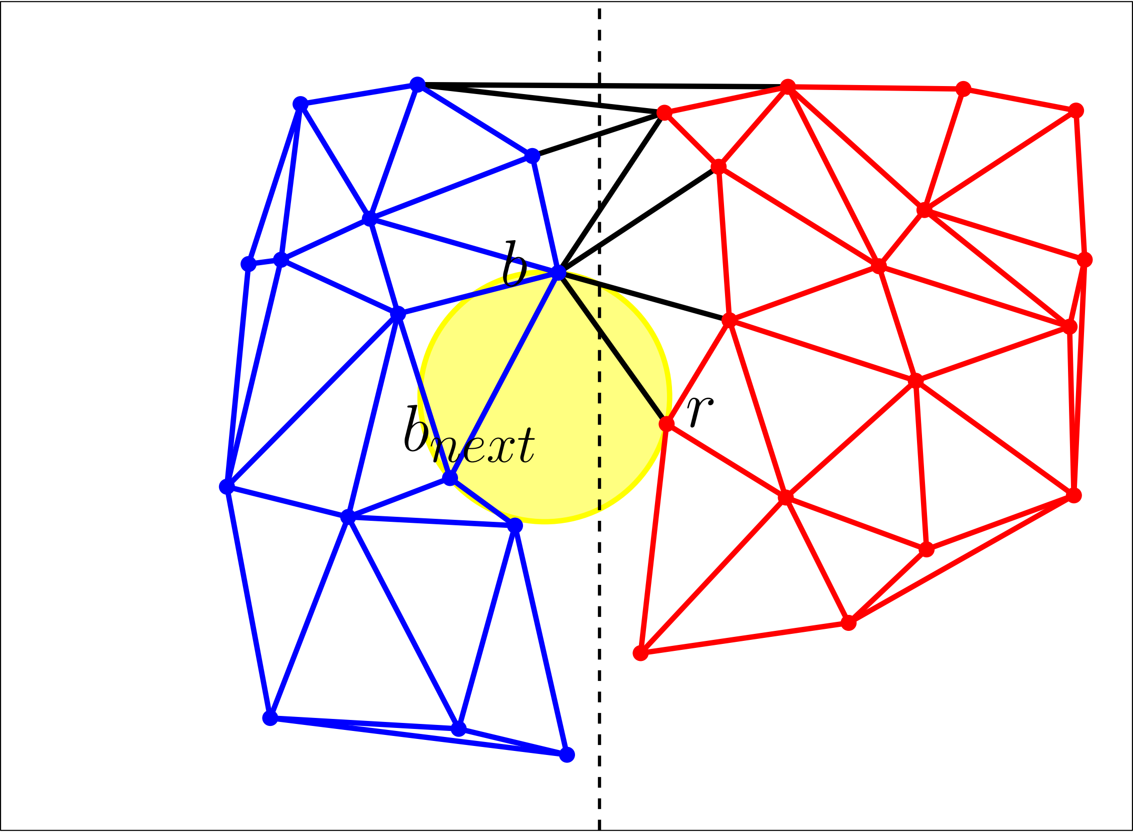
b
r



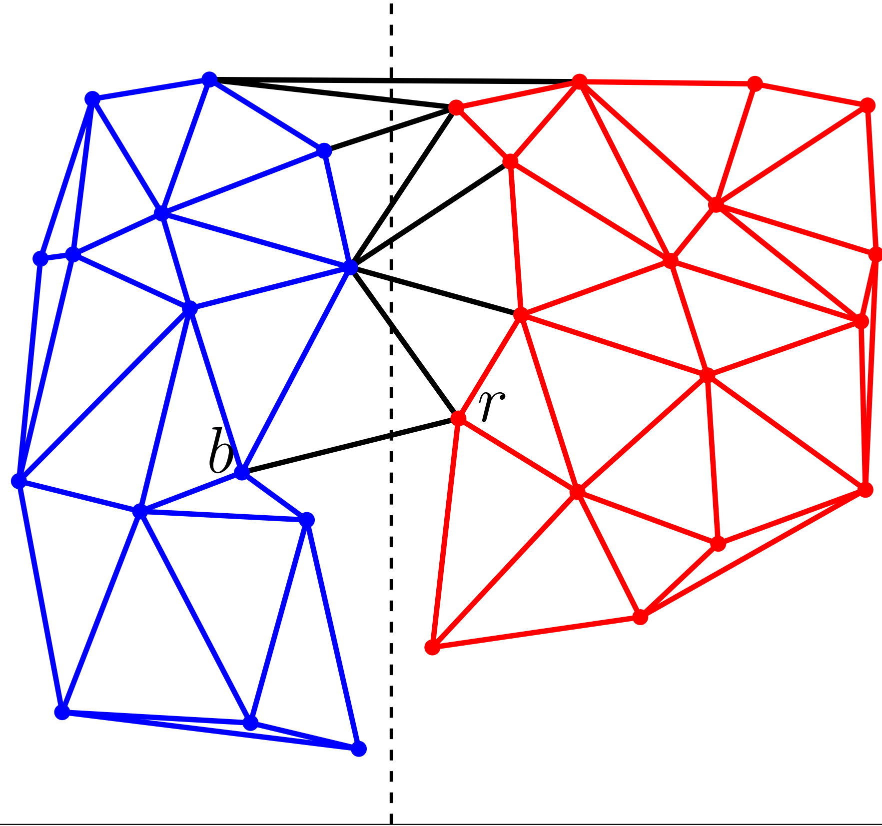
b
r
r_next



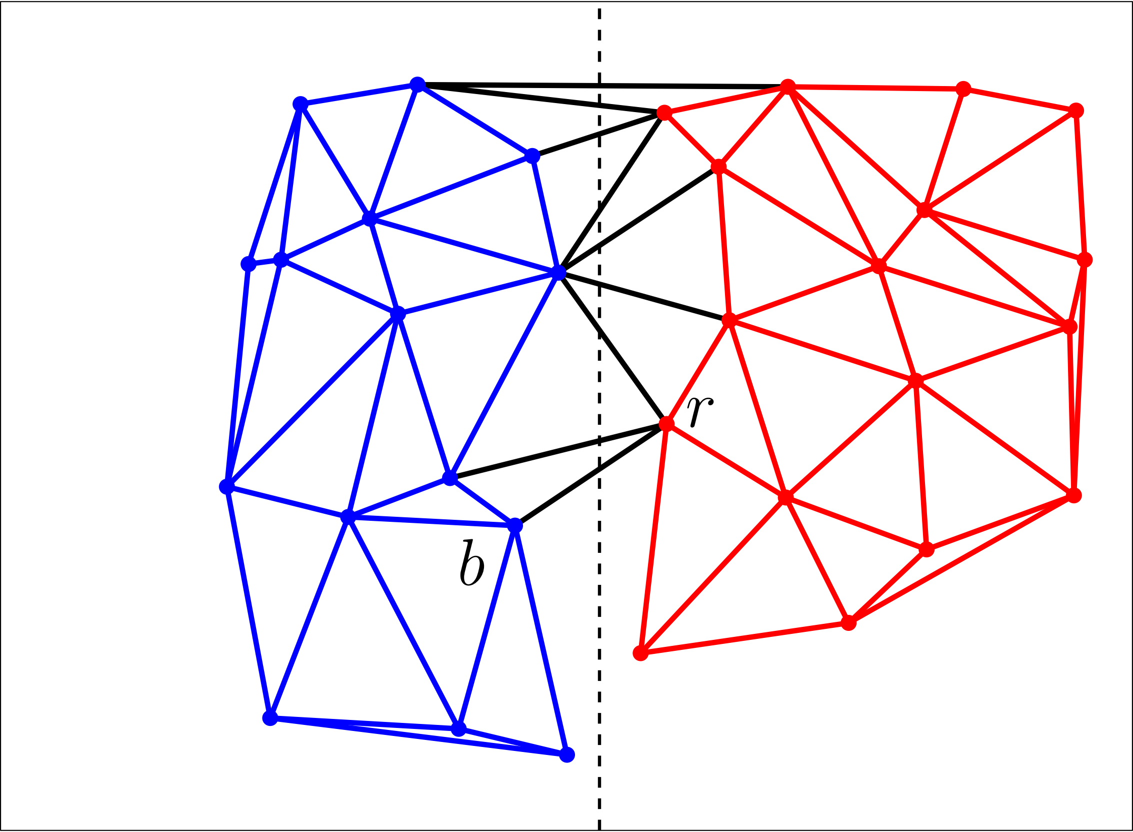
b
r



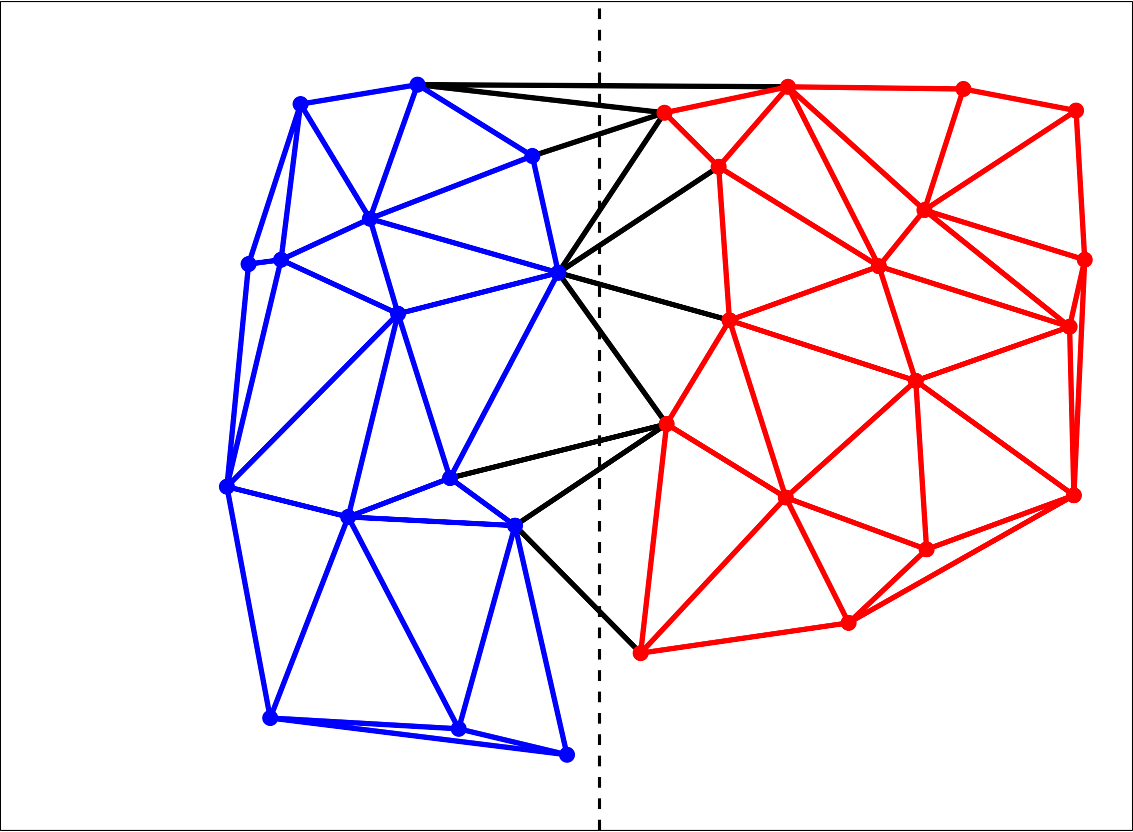
b
b_{next}
r

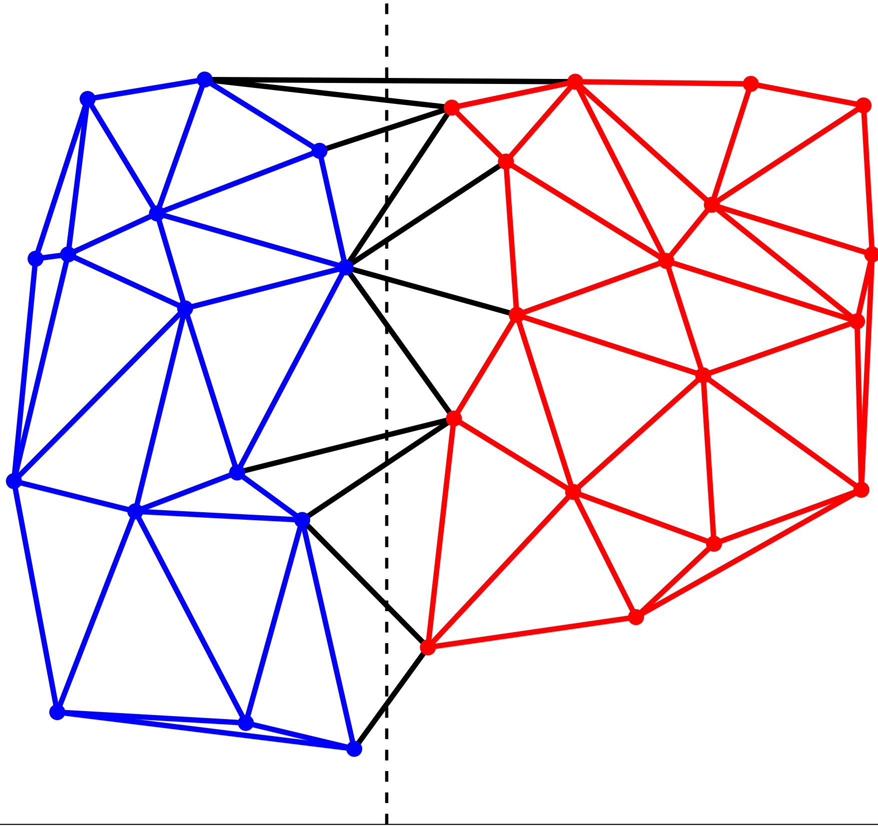


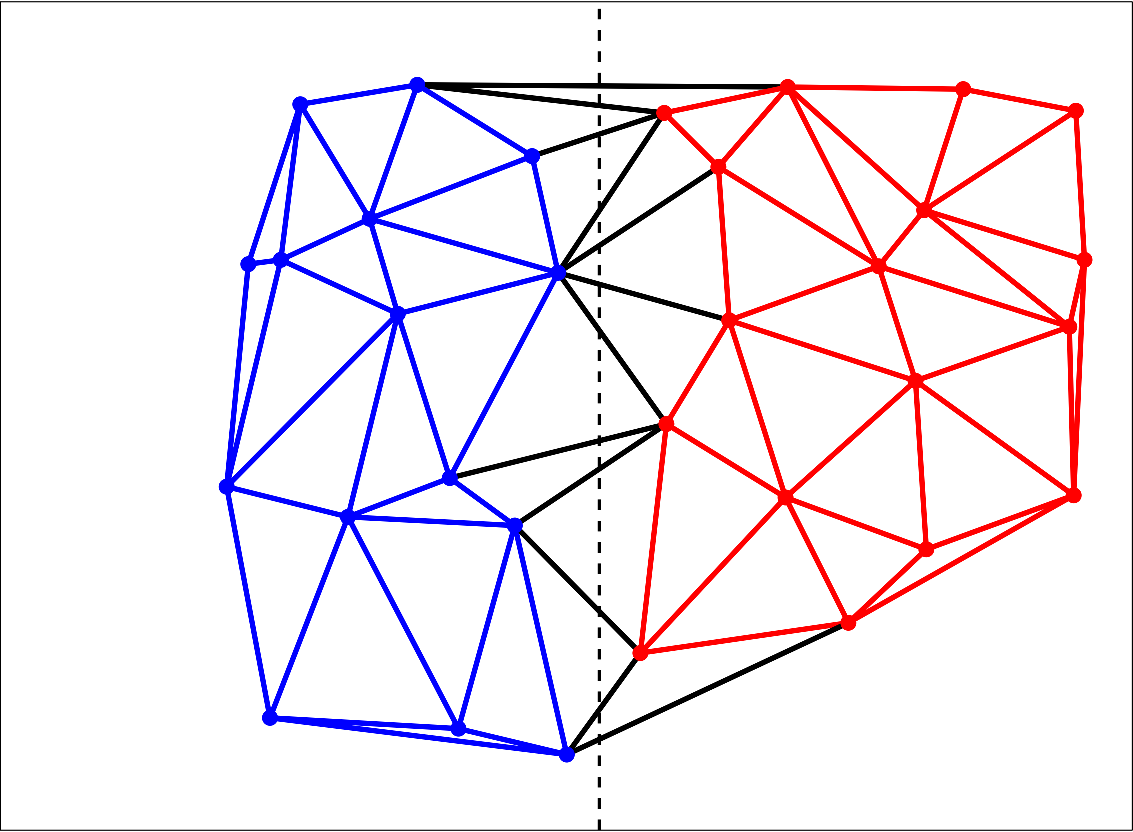
b
r



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

$$\text{Complexity} \leq \quad \sharp\ \text{red edges} \\ +\sharp\ \text{blue edges} \\ +\sharp\ \text{black edges}$$

Complexity of Fusion

$$\text{Complexity} \leq \quad \sharp \text{ red edges} + \sharp \text{ blue edges} + \sharp \text{ black edges}$$

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

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

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

Change

ambient space (for q)

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

Voronoi diagram

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

Change

metrics

Euclidean L_2

L_1, L_∞, L_p

hyperbolic

additive weights

multiplicative weights

Voronoi diagram

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

Change

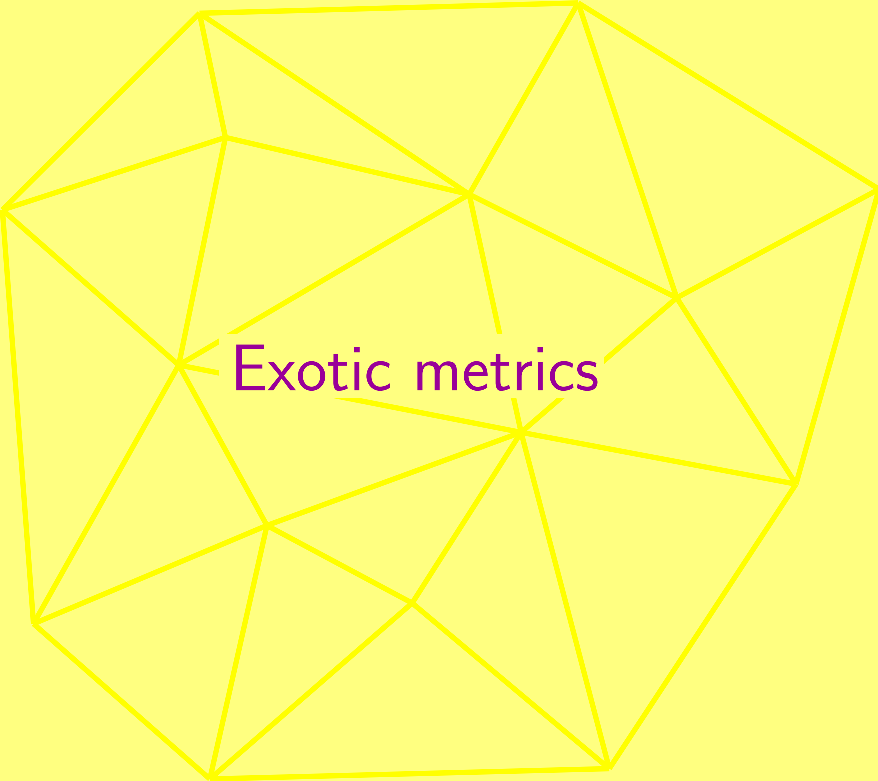
universal set $\supset \mathcal{S}$

points of $\mathbb{R}^d$

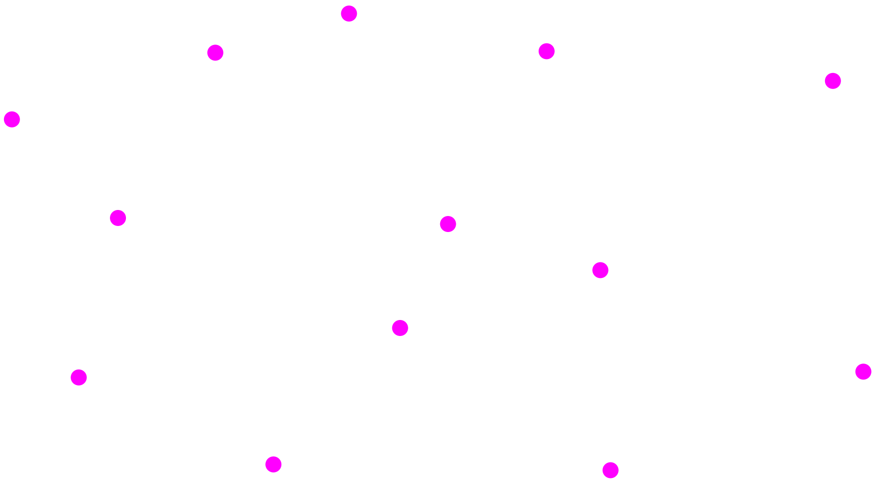
segments of $\mathbb{R}^d$

spheres of $\mathbb{R}^d$

Exotic metrics

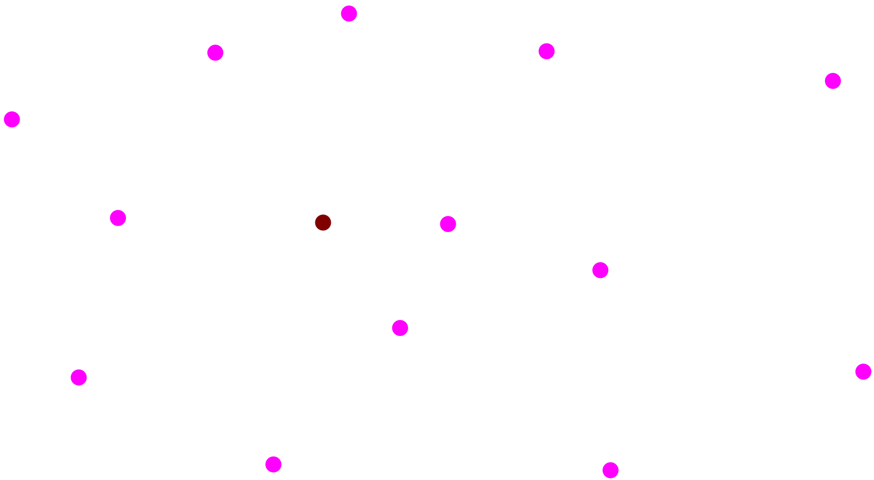


Norm L_∞: $\max(|x|, |y|)$

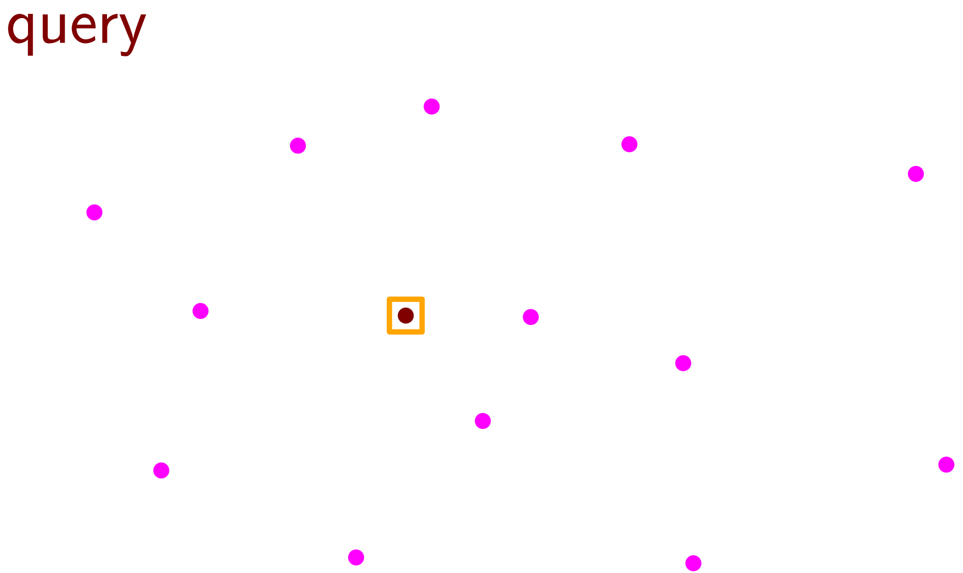


Norm L_∞: $\max(|x|, |y|)$

query

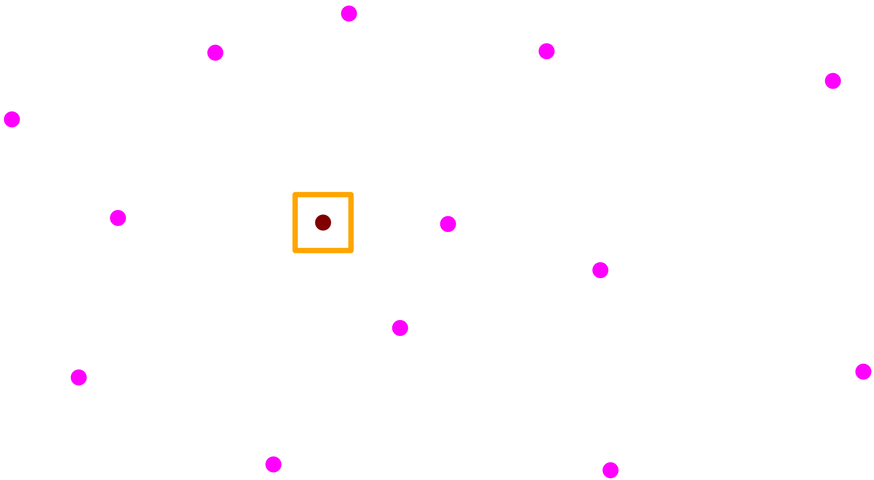


Norm L_∞: $\max(|x|, |y|)$

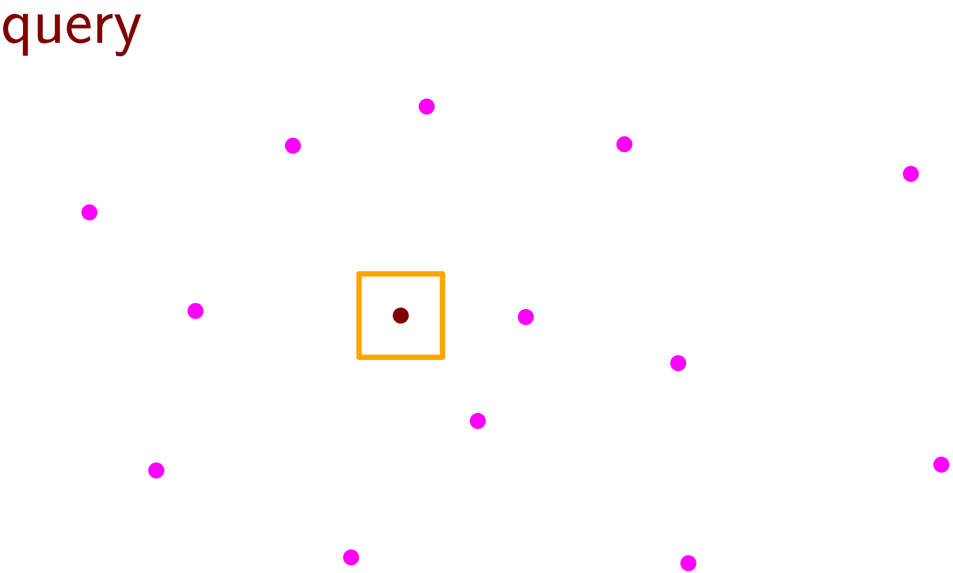


Norm L_∞: $\max(|x|, |y|)$

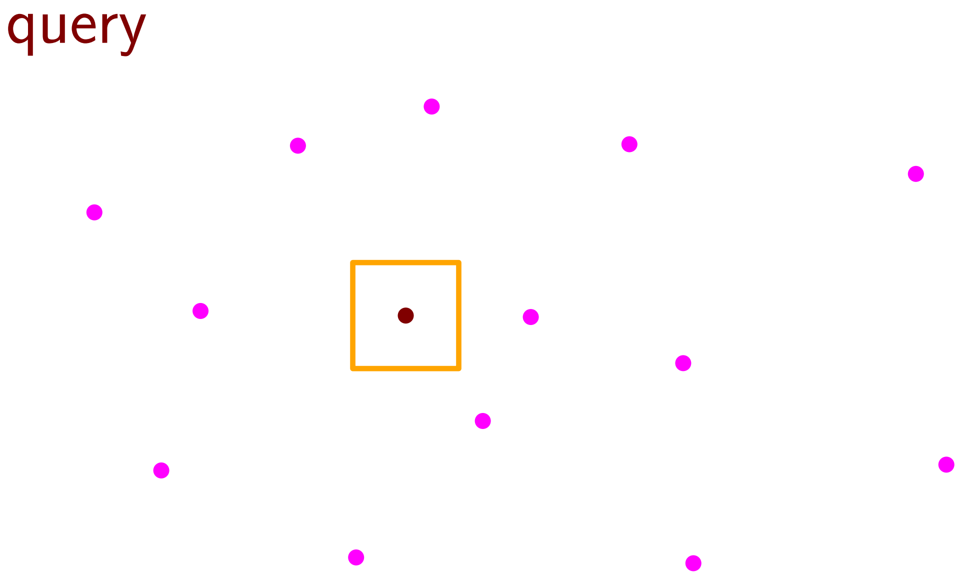
query



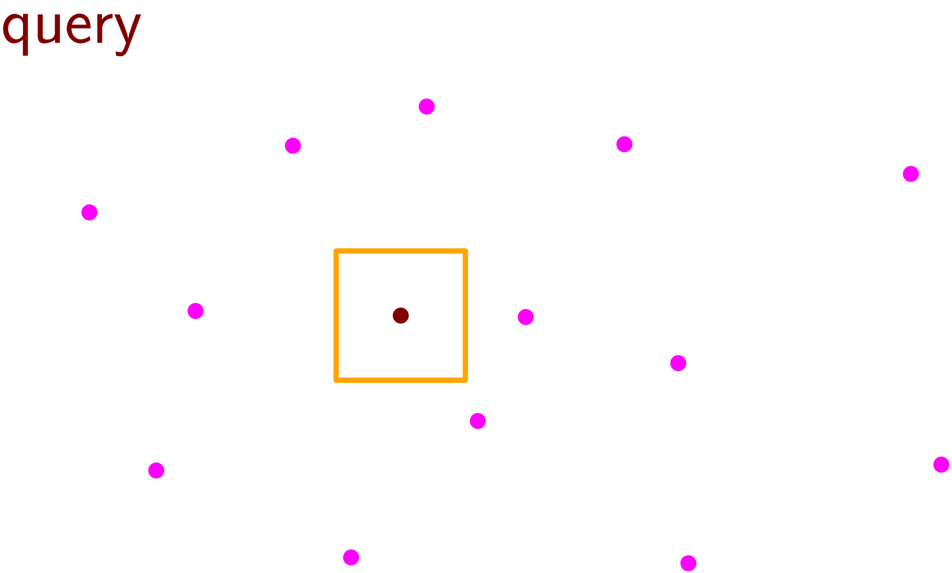
Norm L_∞: max$(|x|, |y|)$



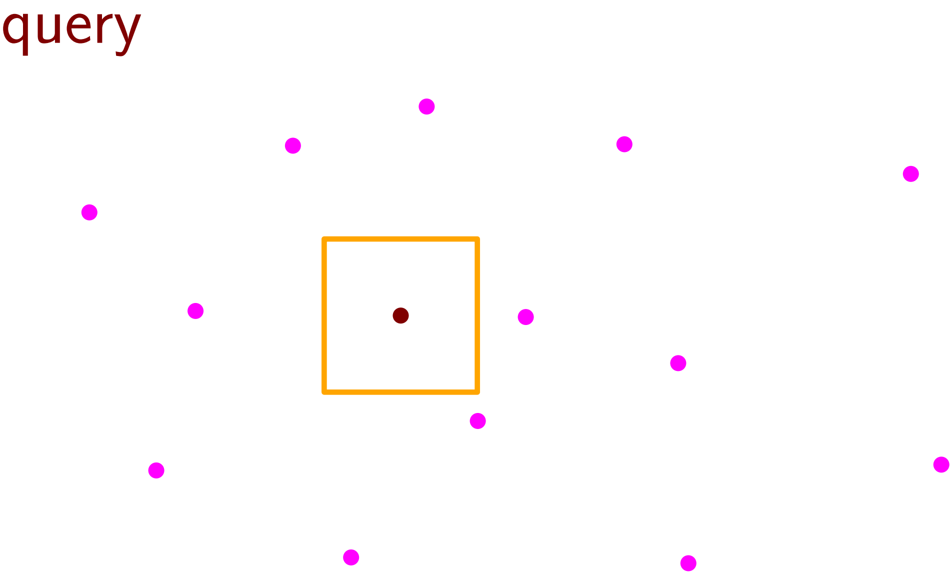
Norm L_∞: $\max(|x|, |y|)$



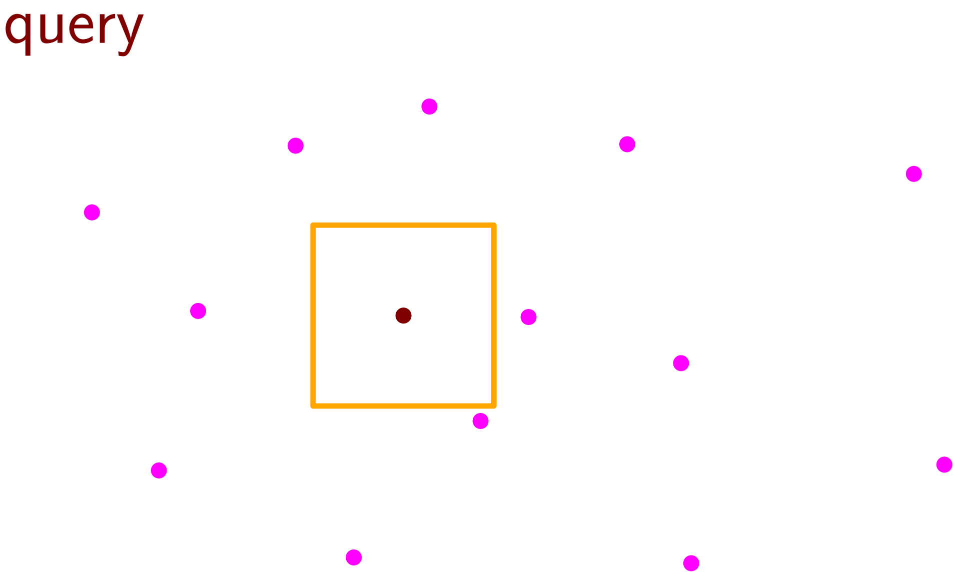
Norm L_∞: max$(|x|, |y|)$



Norm L_∞: $\max(|x|, |y|)$

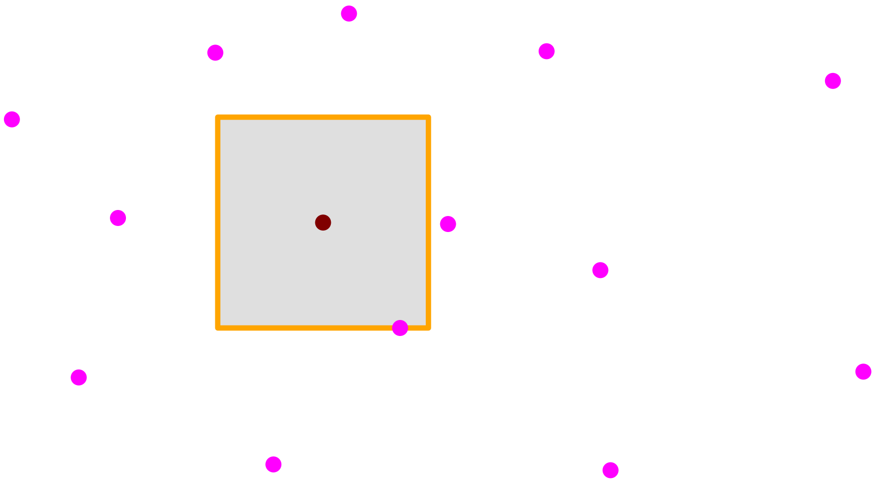


Norm L_∞: max$(|x|, |y|)$

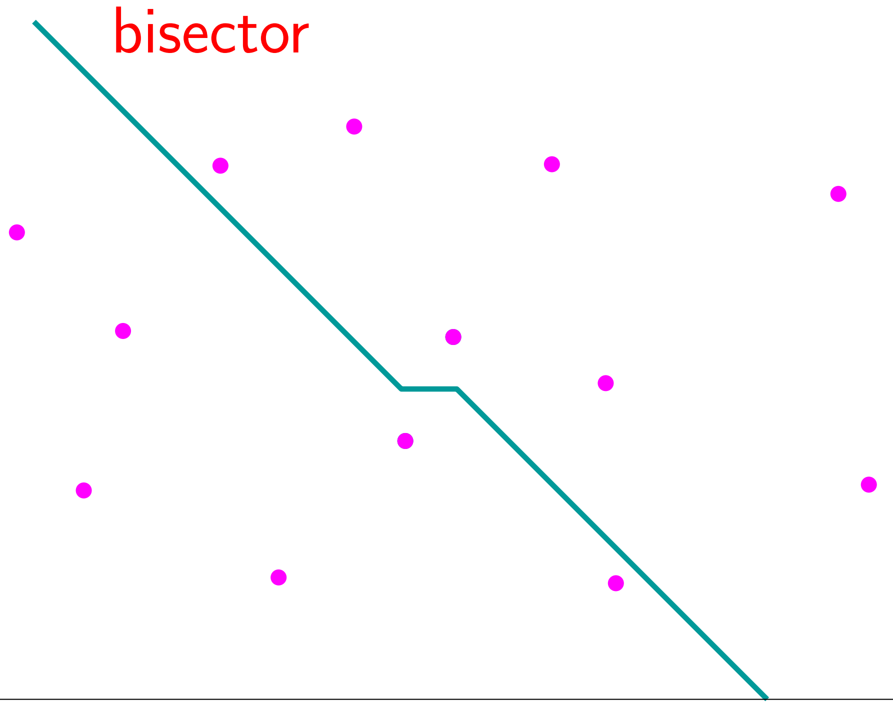


Norm L_∞: $\max(|x|, |y|)$

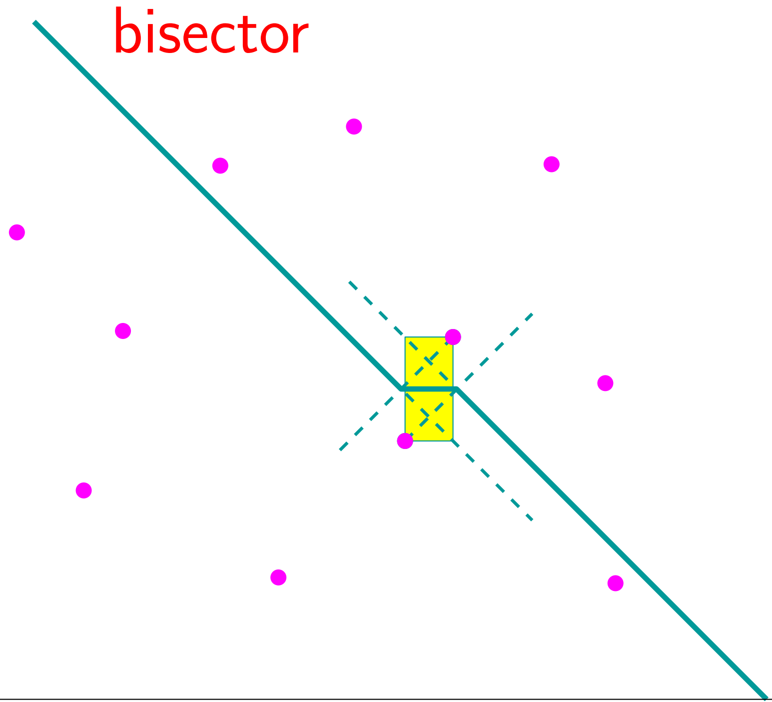
query



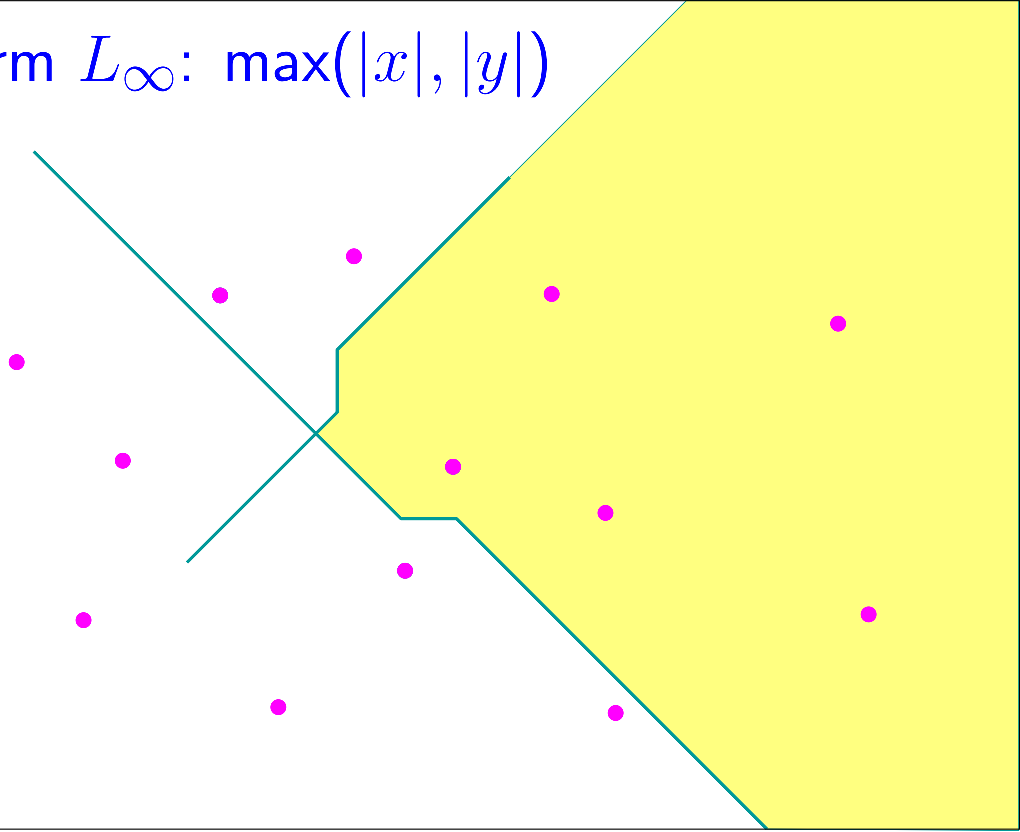
Norm L_∞: $\max(|x|, |y|)$



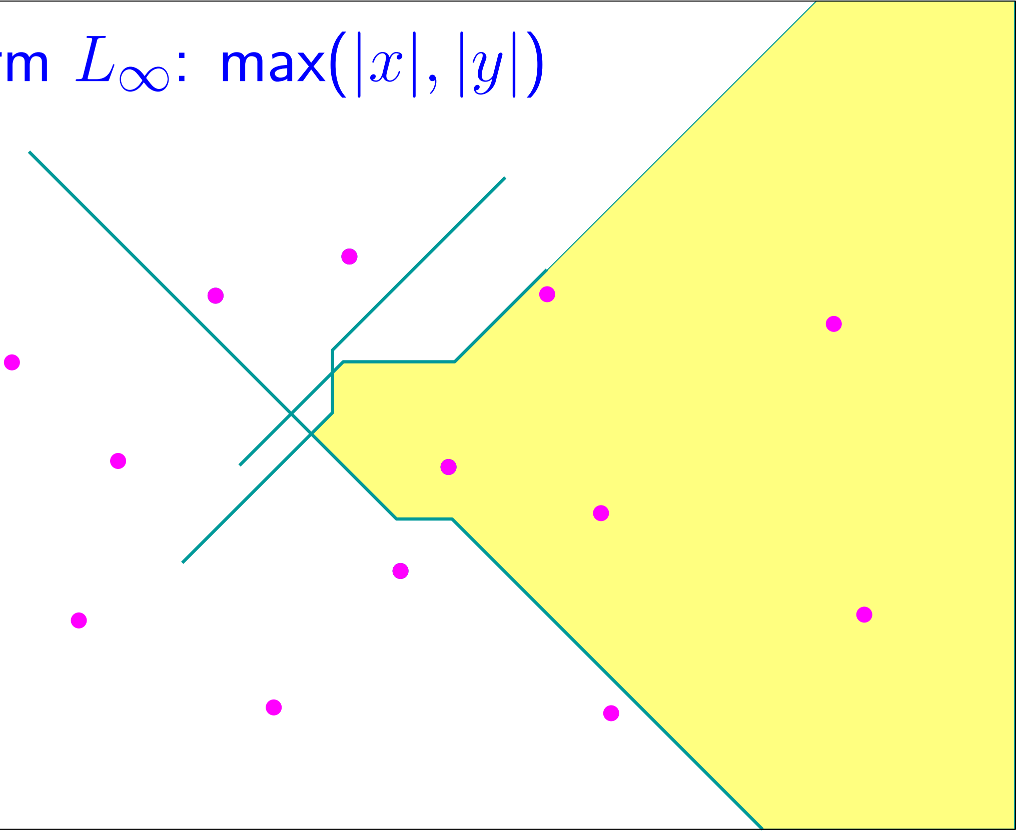
Norm L_∞: max$(|x|, |y|)$



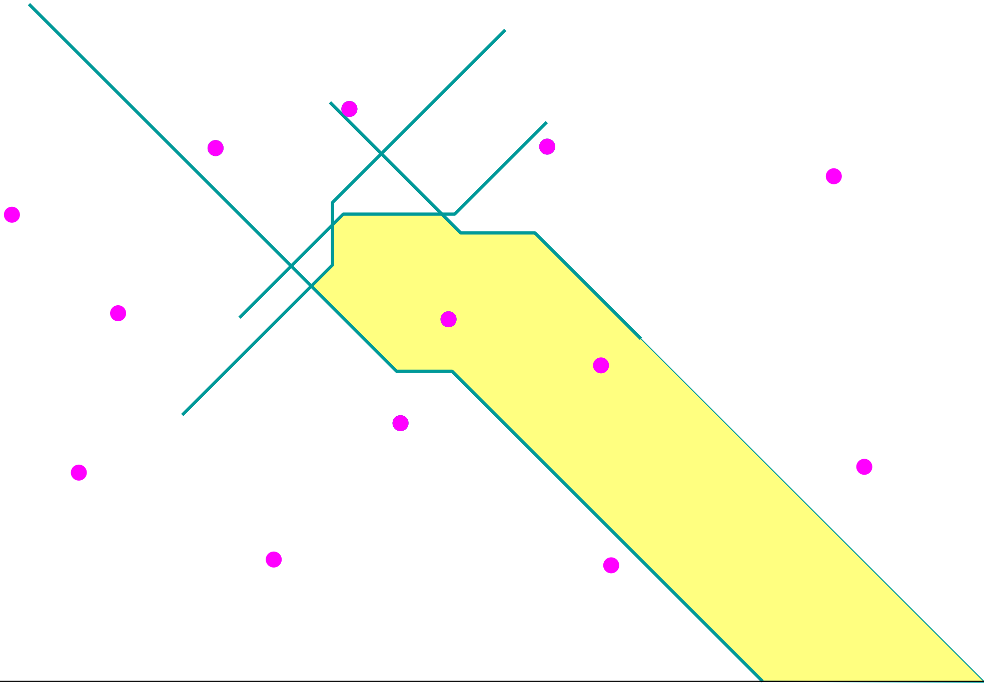
Norm L_∞: max$(|x|, |y|)$



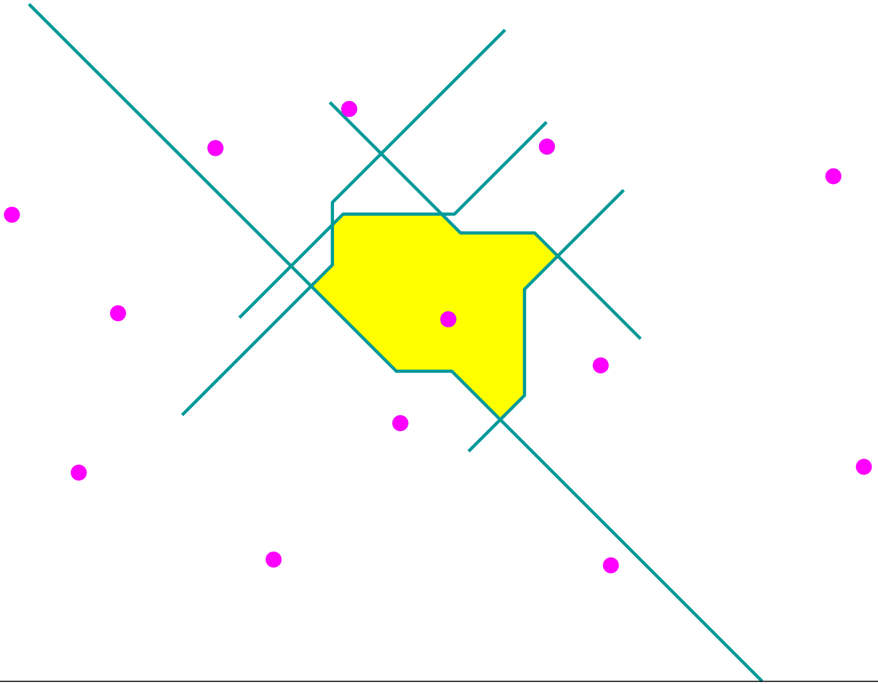
Norm L_∞: max$(|x|, |y|)$

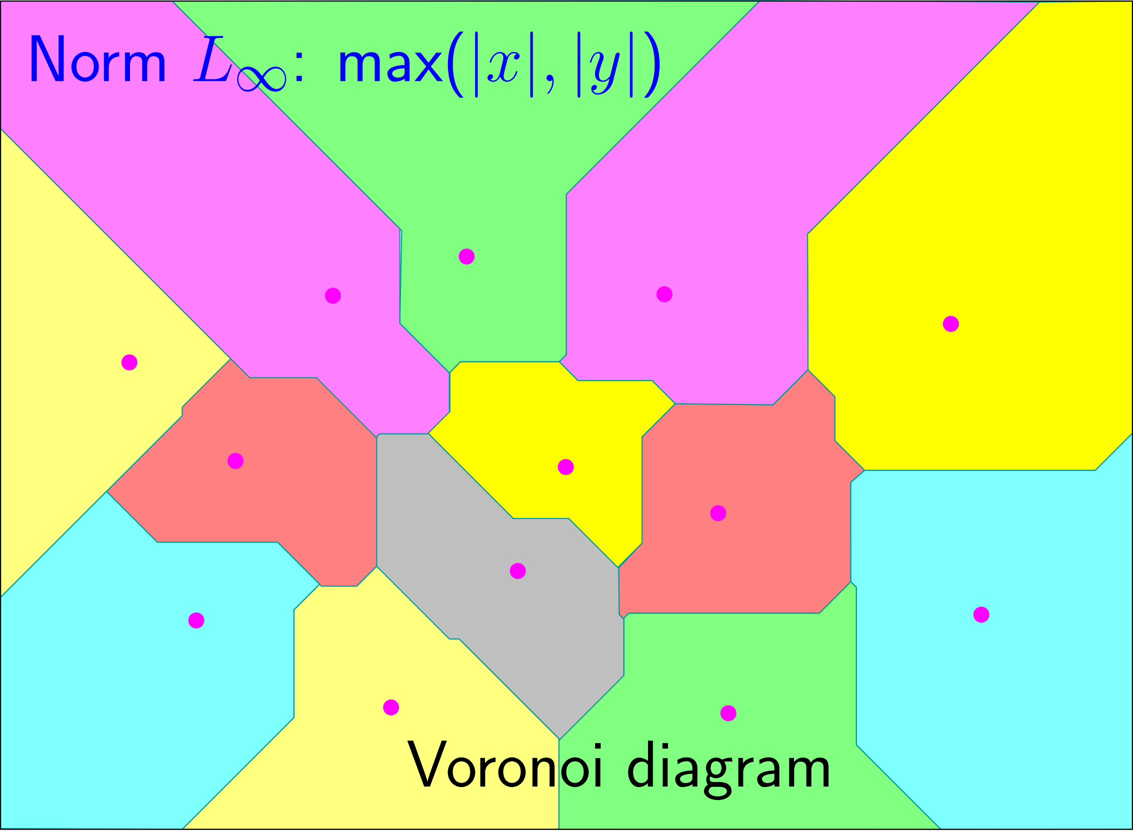


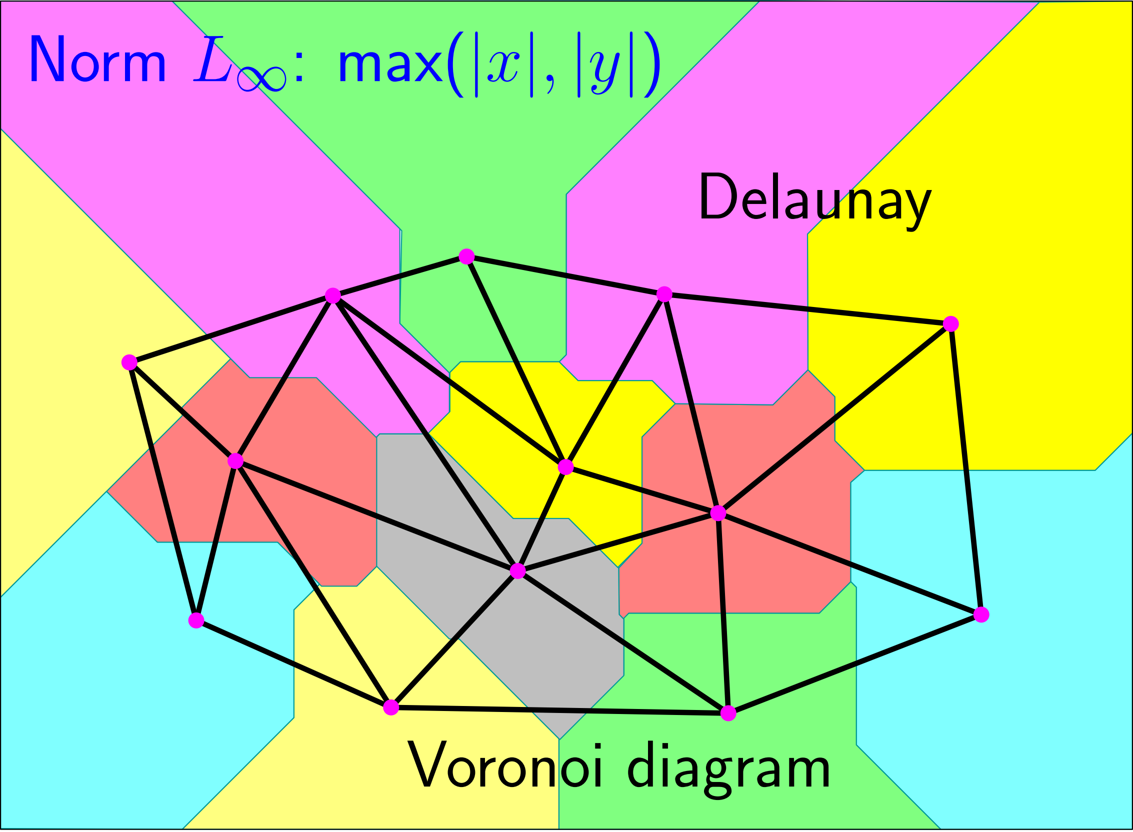
Norm L_∞: $\max(|x|, |y|)$



Norm L_∞: $\max(|x|, |y|)$




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

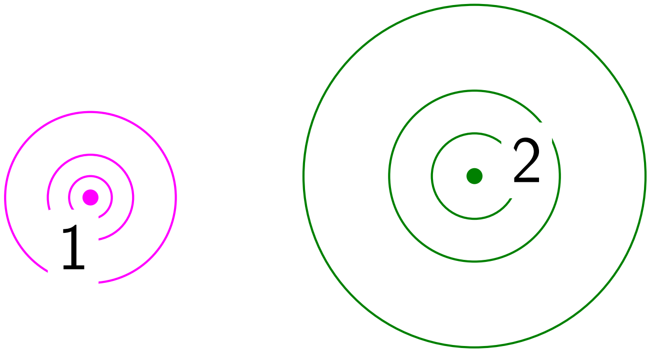

Norm L_∞: $\max(|x|, |y|)$
Delaunay
Voronoi diagram

multiplicatively-weighted points

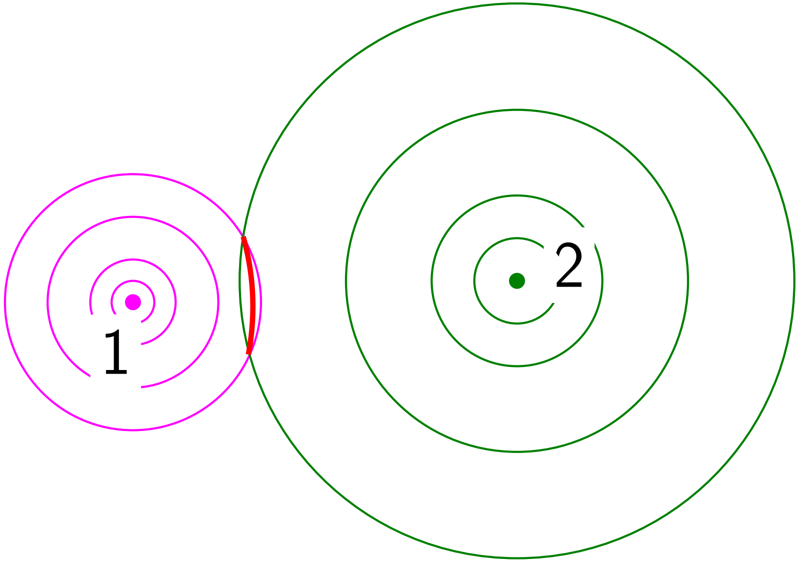
multiplicatively-weighted points

multiplicatively-weighted points

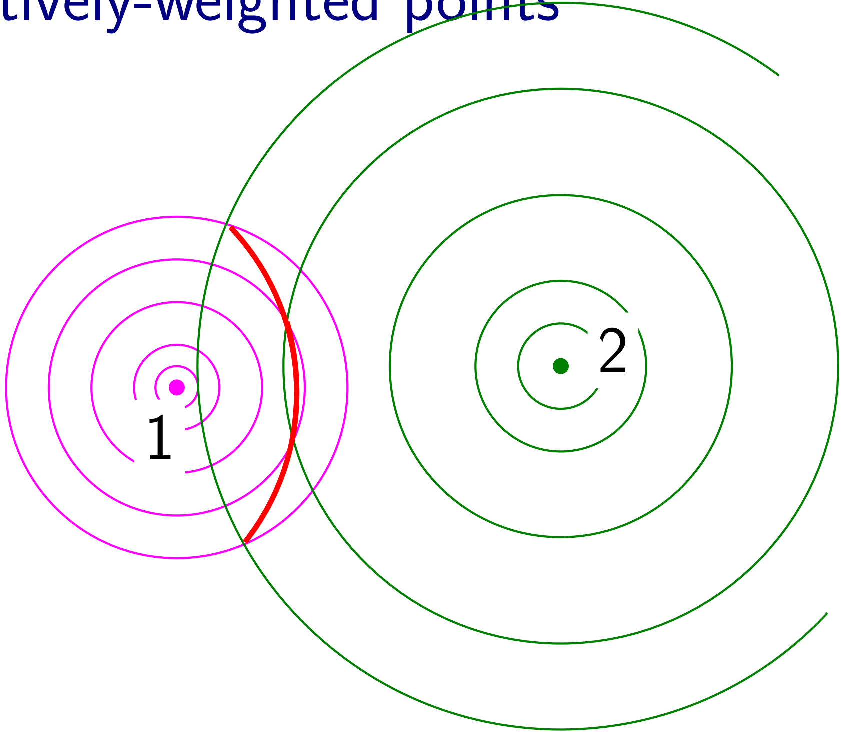
multiplicatively-weighted points



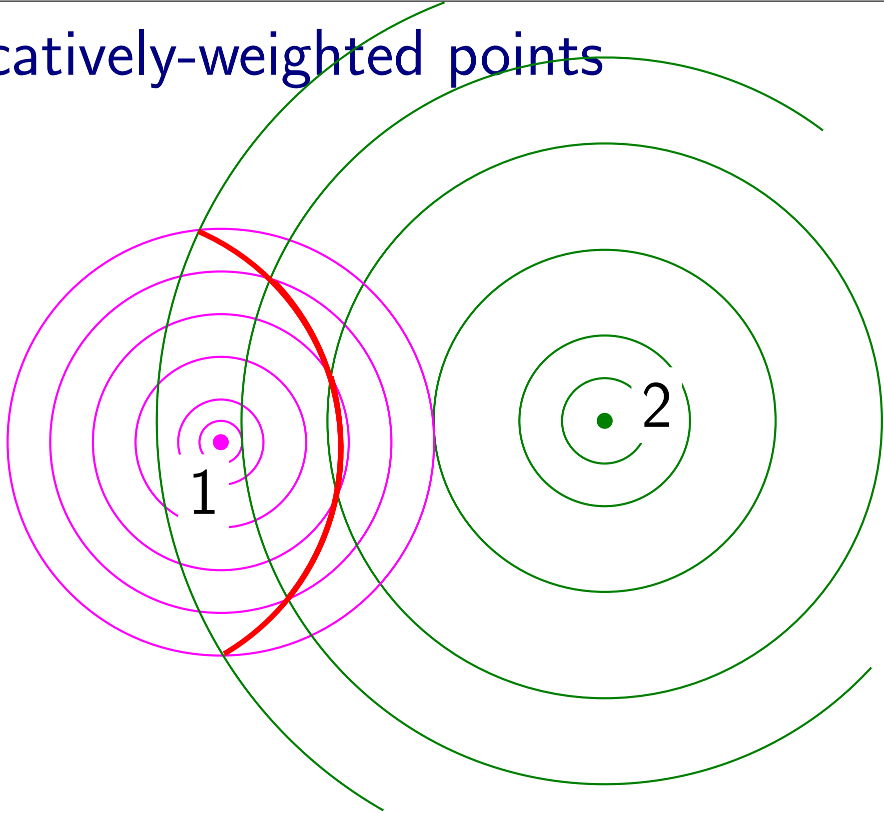
multiplicatively-weighted points



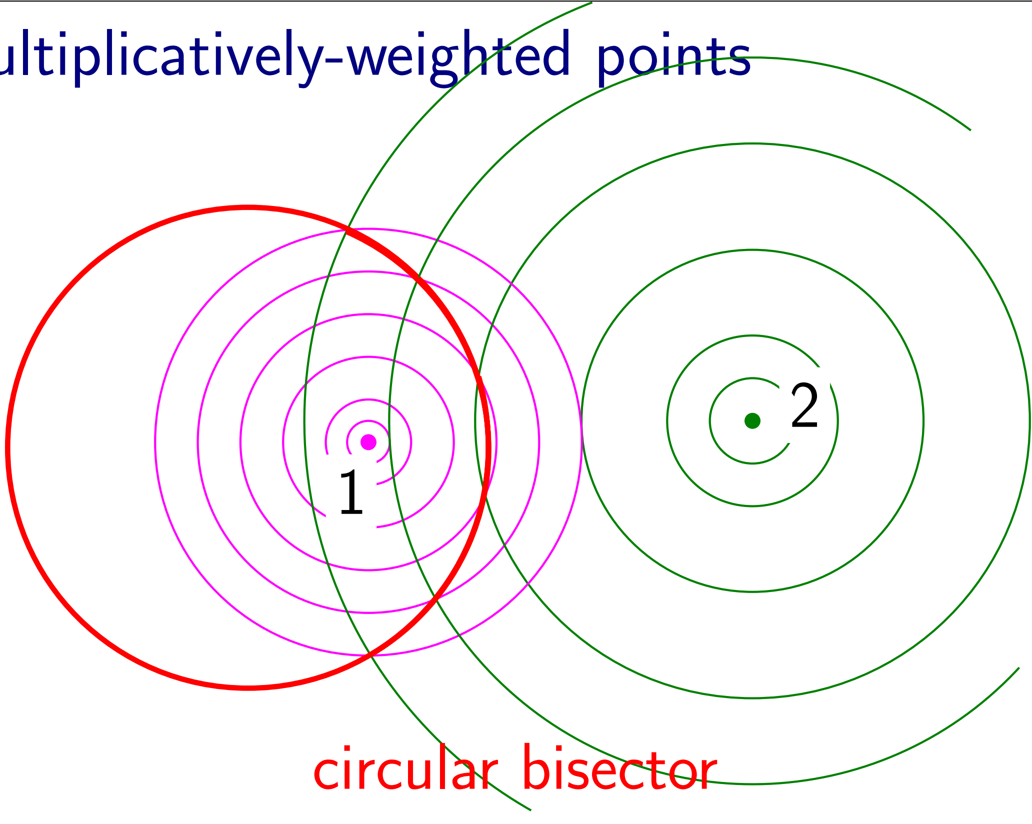
multiplicatively-weighted points



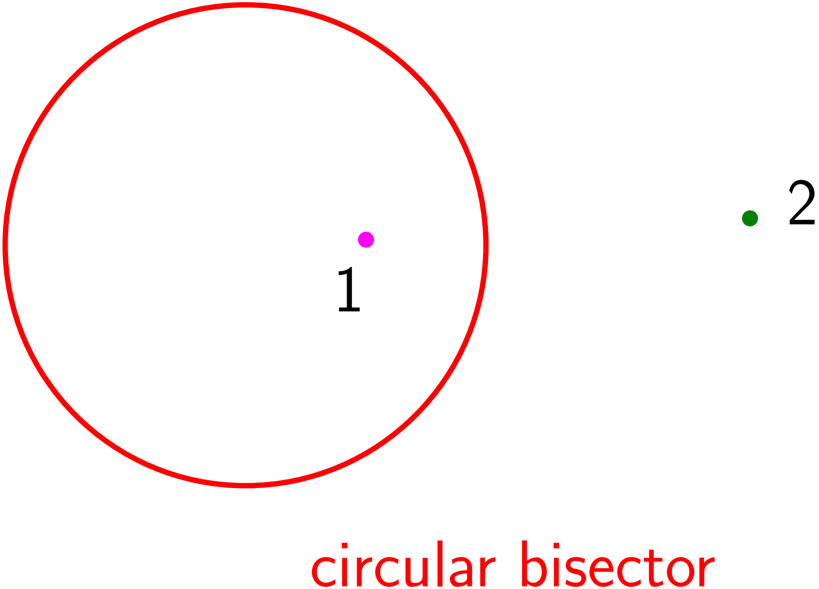
multiplicatively-weighted points



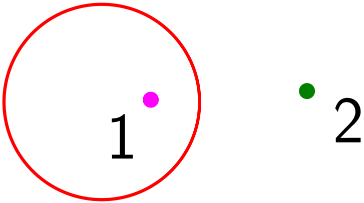
multiplicatively-weighted points



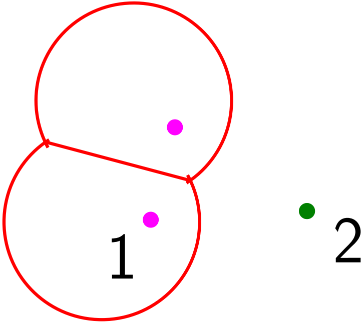
multiplicatively-weighted points



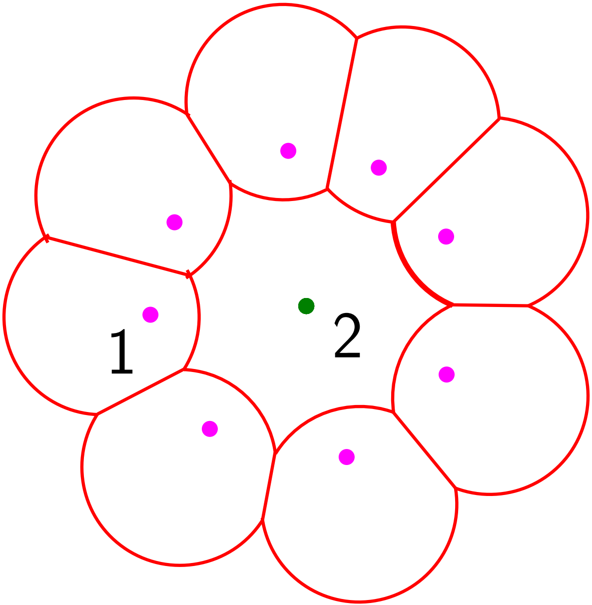
multiplicatively-weighted points



multiplicatively-weighted points



multiplicatively-weighted points

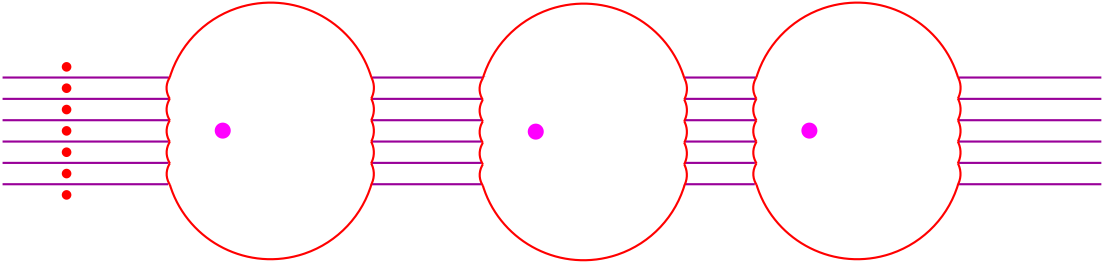


disconnected cell

multiplicatively-weighted points

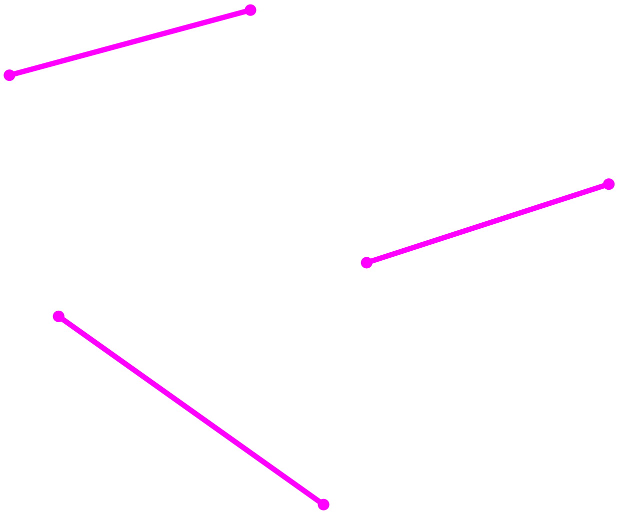
multiplicatively-weighted points

multiplicatively-weighted points

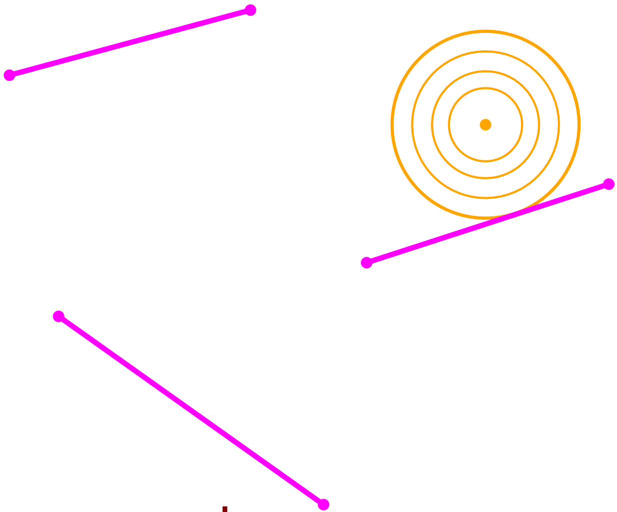


quadratic size

Voronoi diagram of segments

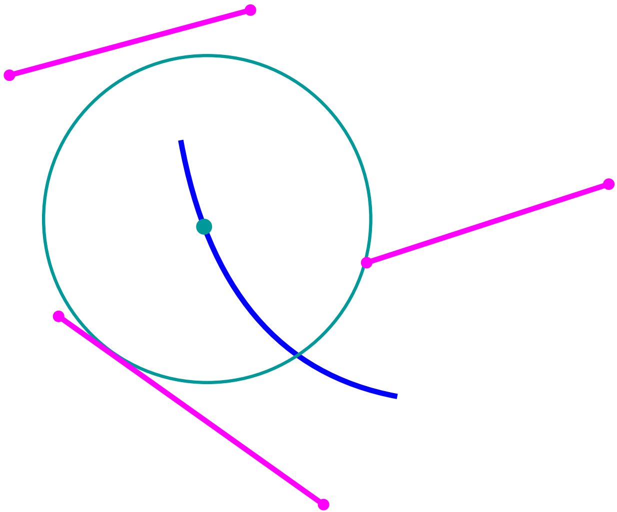


Voronoi diagram of segments



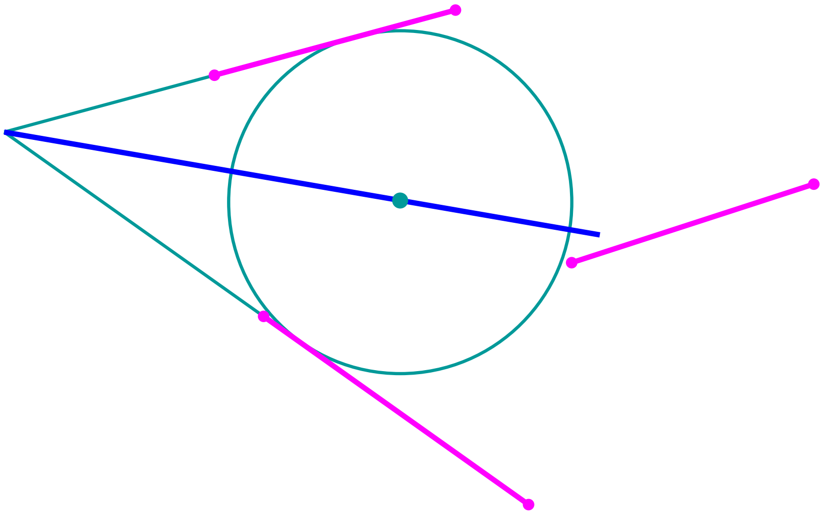
$\boxed{Q}$ nearest segment

Voronoi diagram of segments



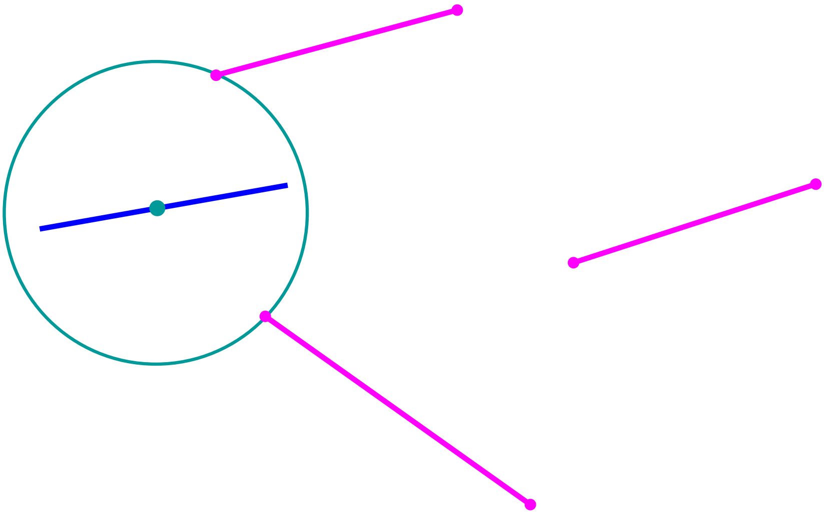
parabolic bisector

Voronoi diagram of segments

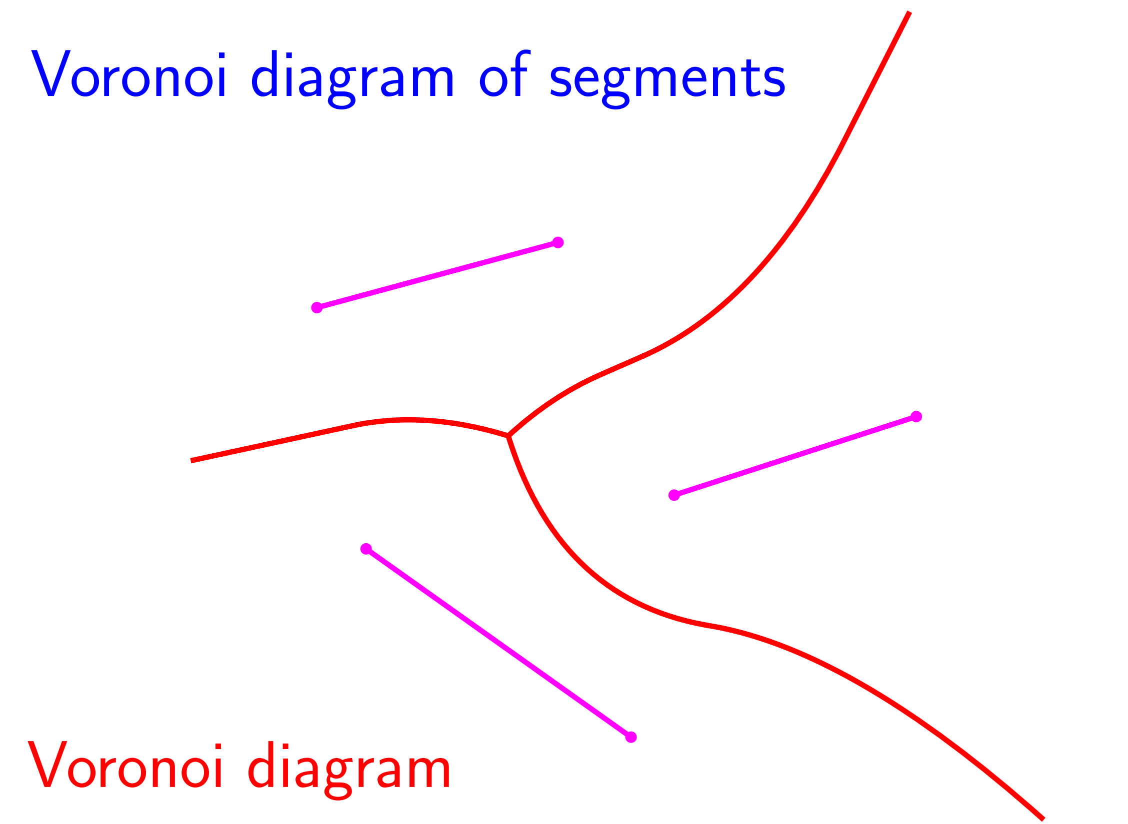


angle bisector

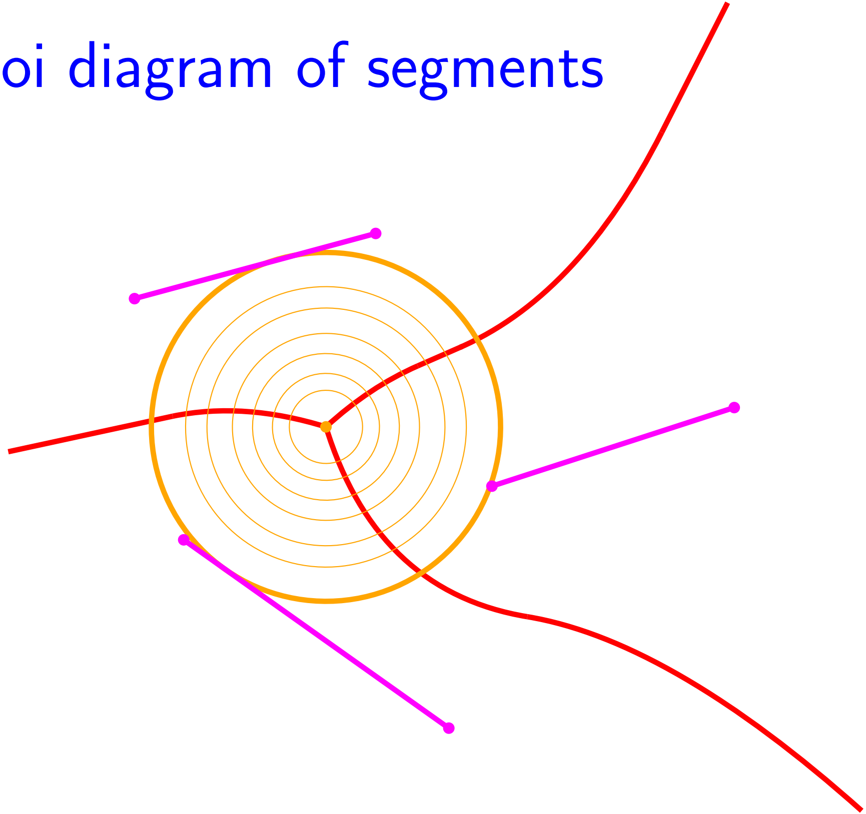
Voronoi diagram of segments



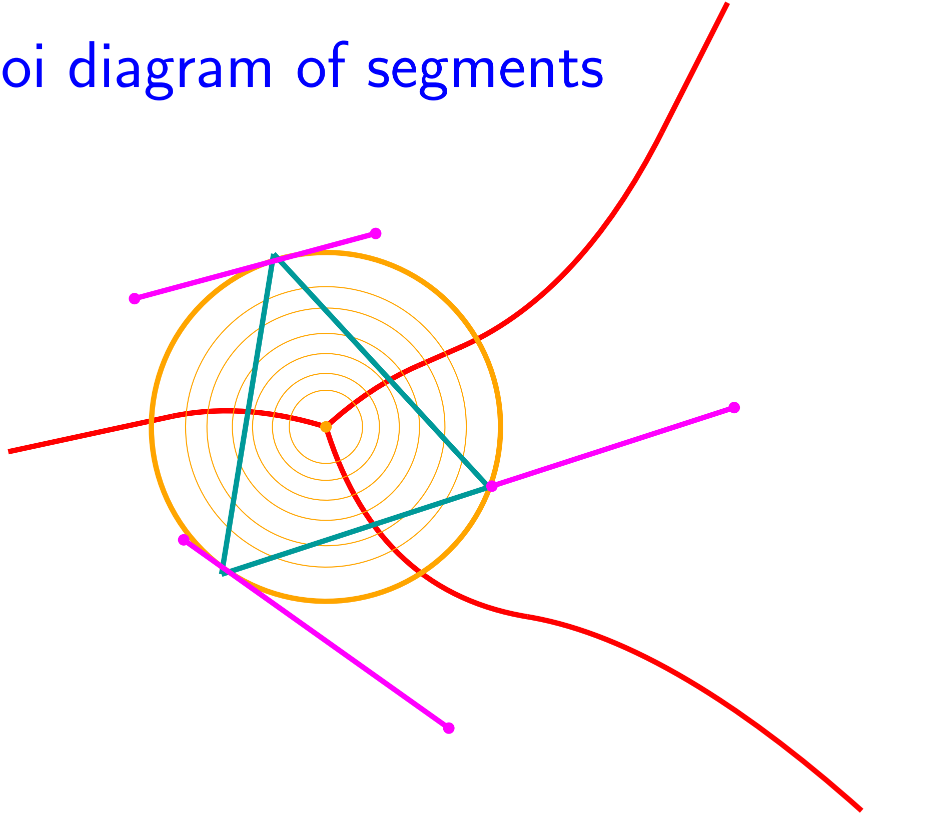
points bisector


Voronoi diagram of segments
Voronoi diagram

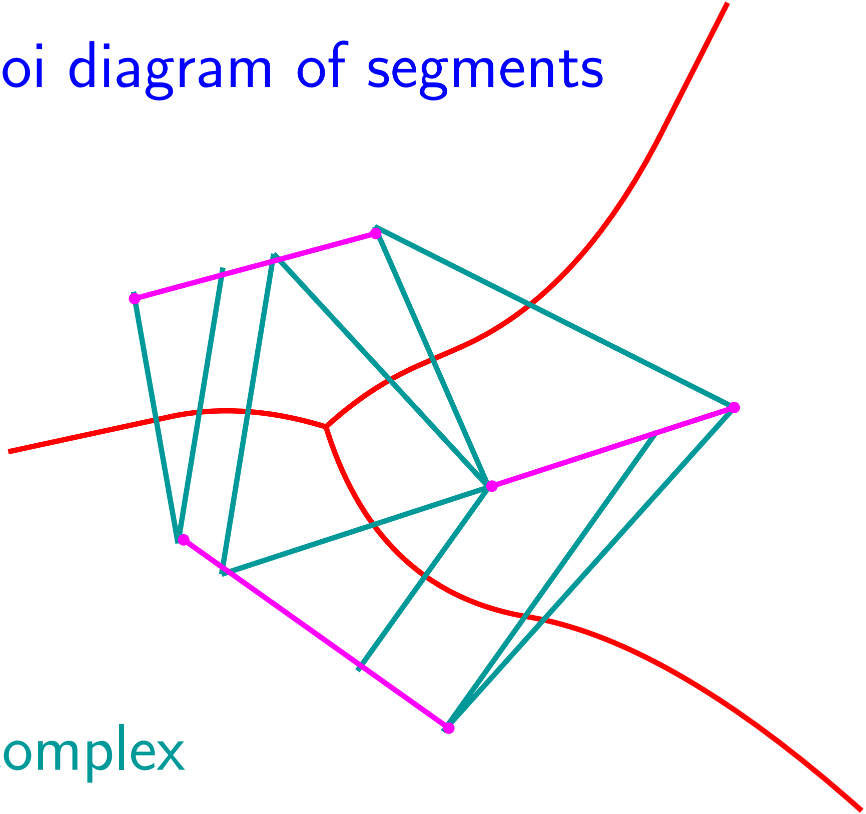
Voronoi diagram of segments



Voronoi diagram of segments



Voronoi diagram of segments



Dual complex