Delaunay triangulations: properties, algorithms, and complexity

Olivier Devillers

Delaunay triangulations: properties, algorithms, and complexity

Extra question: what is the difference

between an algorithm and a program?

Olivier Devillers

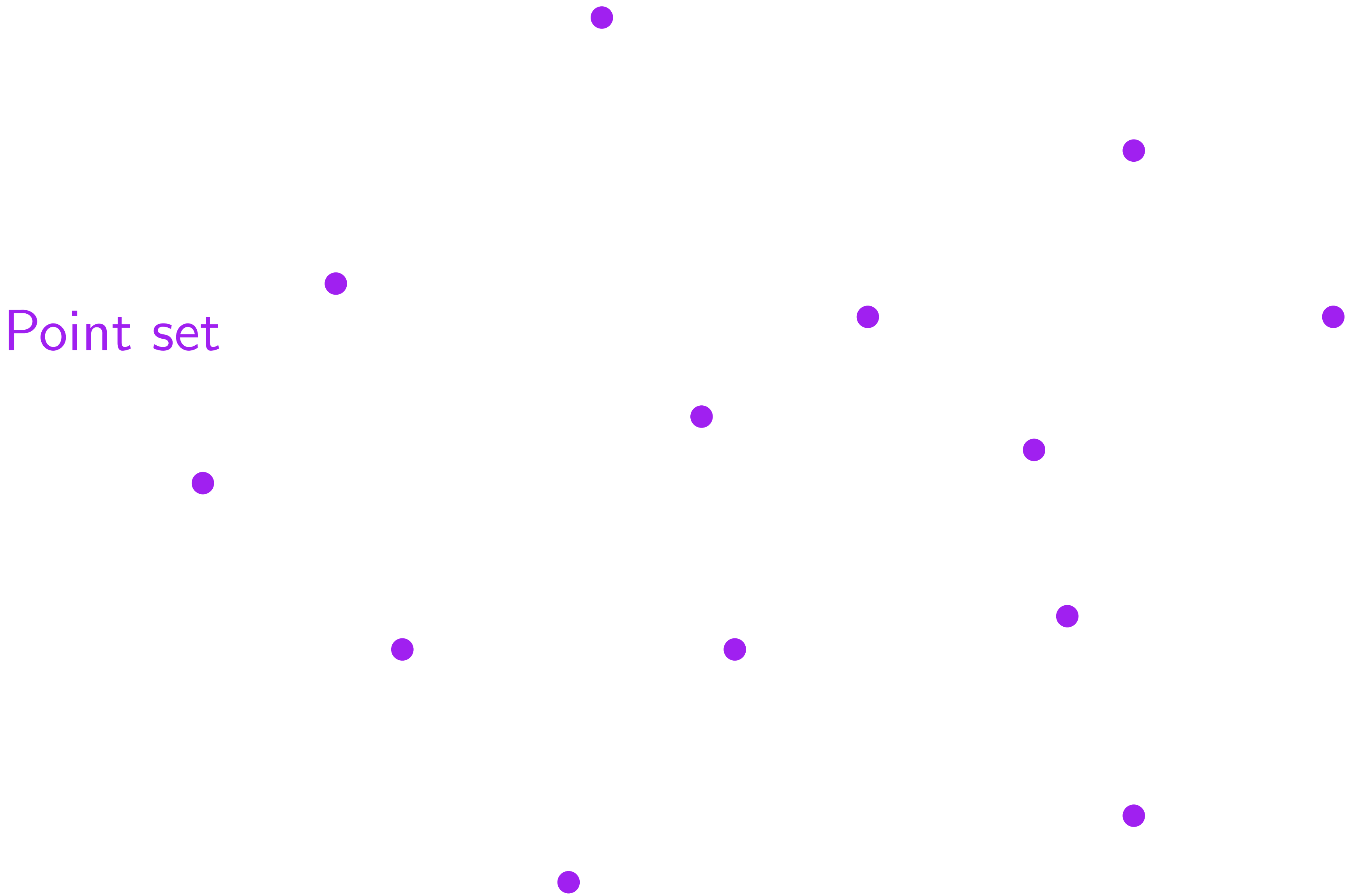
Algorithm

Recipe to go from the input to the output

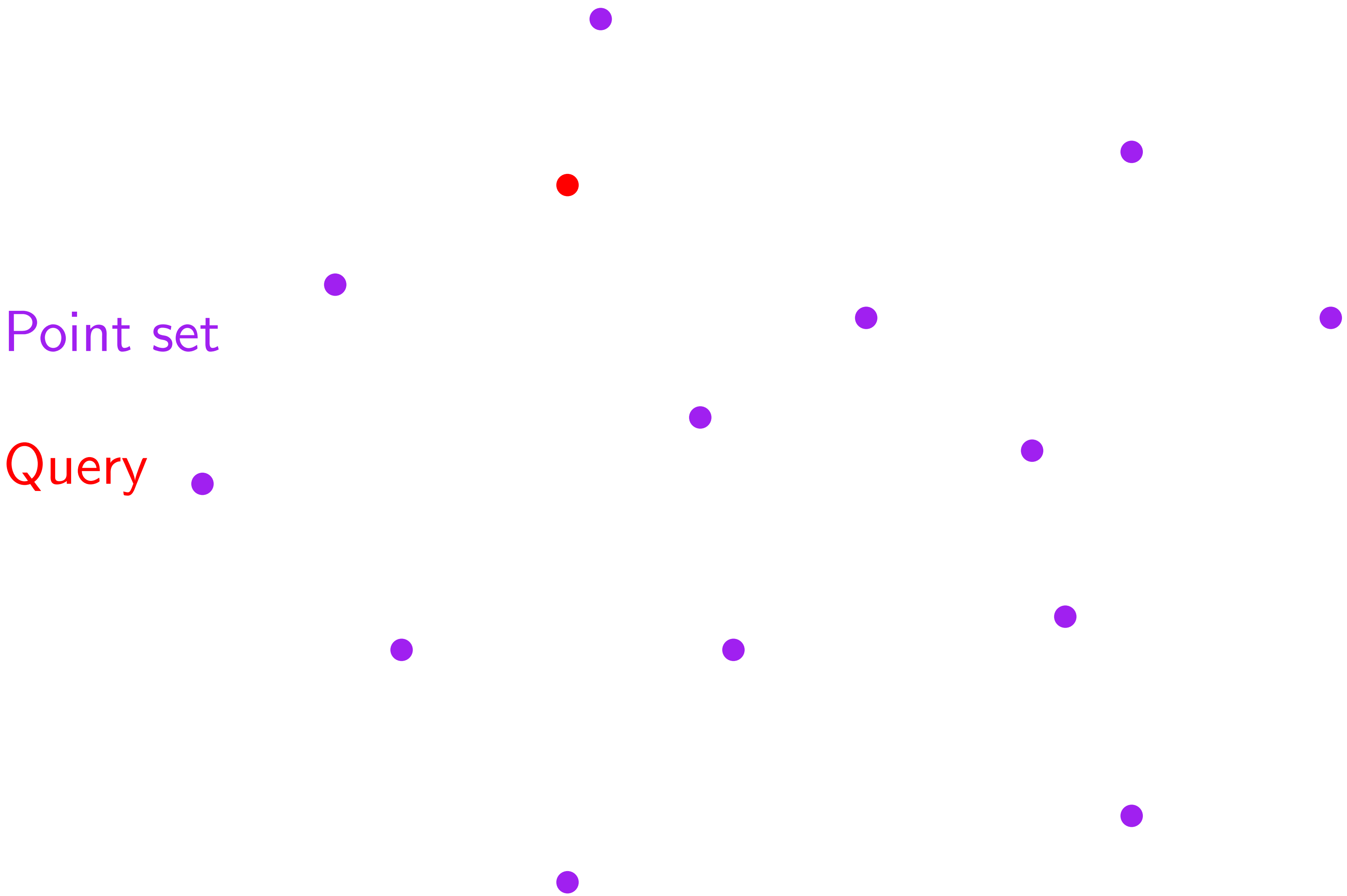
- Formalized description in some language
- May use data structure
- Proof of correctness
- Complexity analysis

Delaunay Triangulation: definition, empty circle property

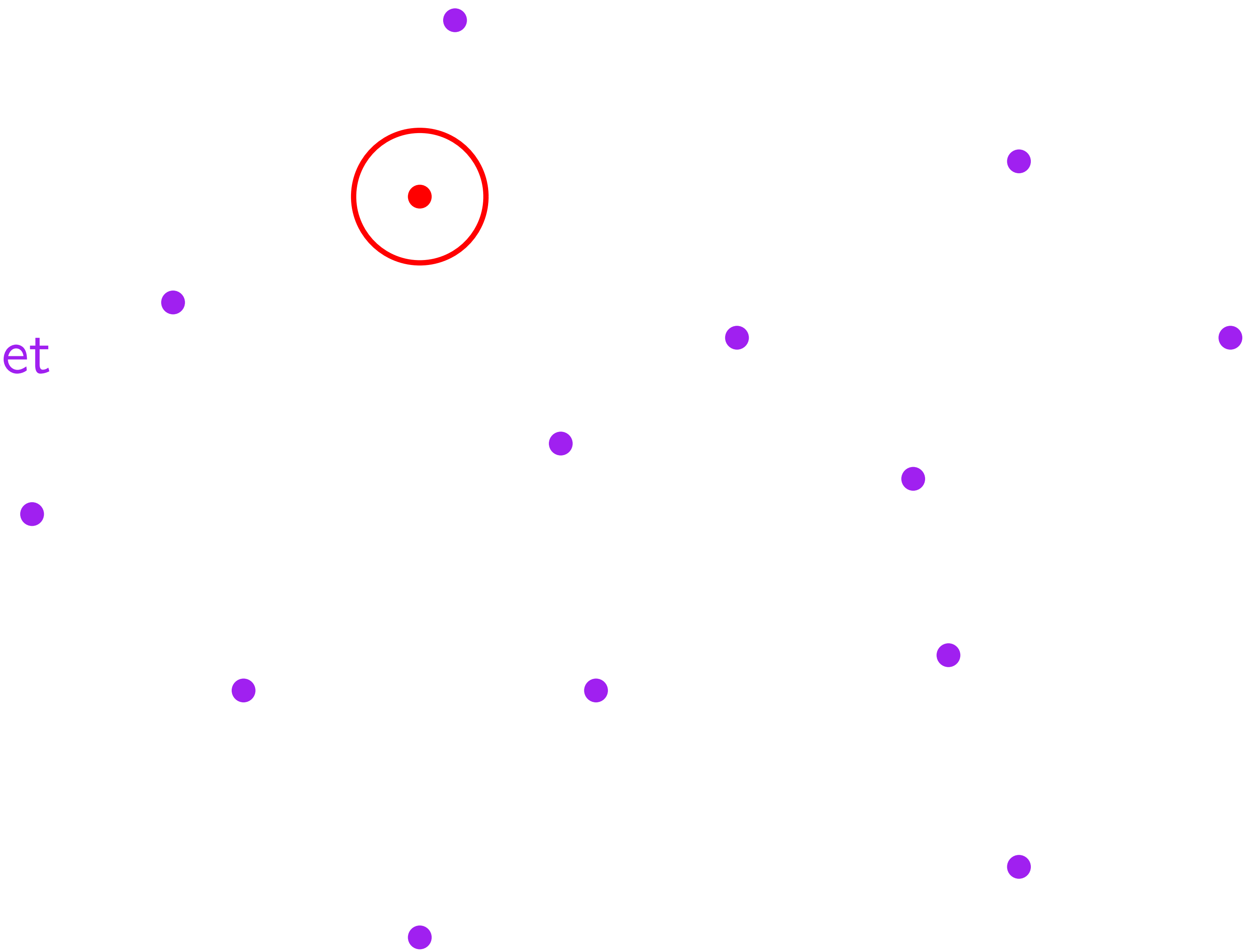
Delaunay Triangulation: definition, empty circle property



Delaunay Triangulation: definition, empty circle property



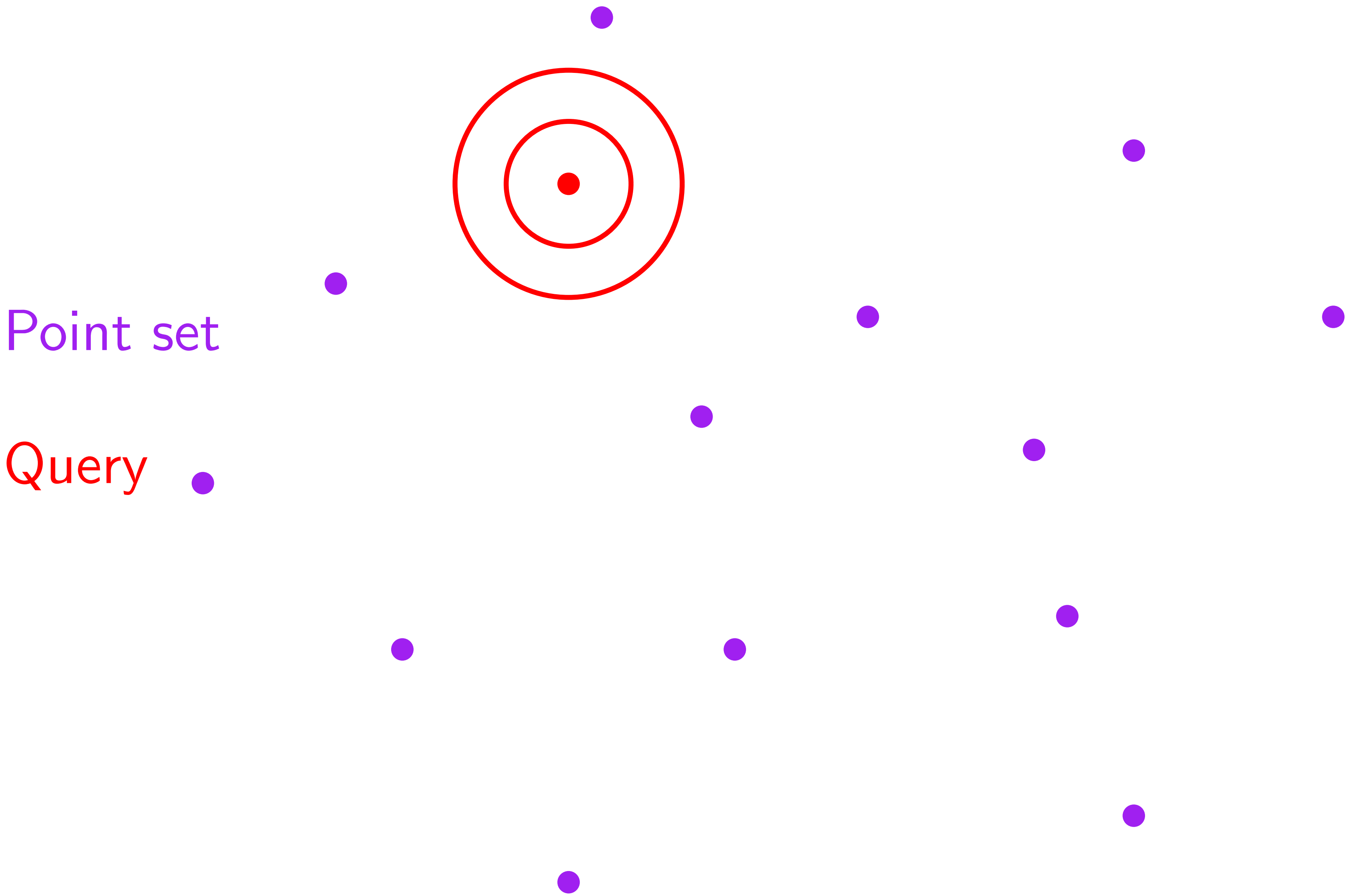
Delaunay Triangulation: definition, empty circle property



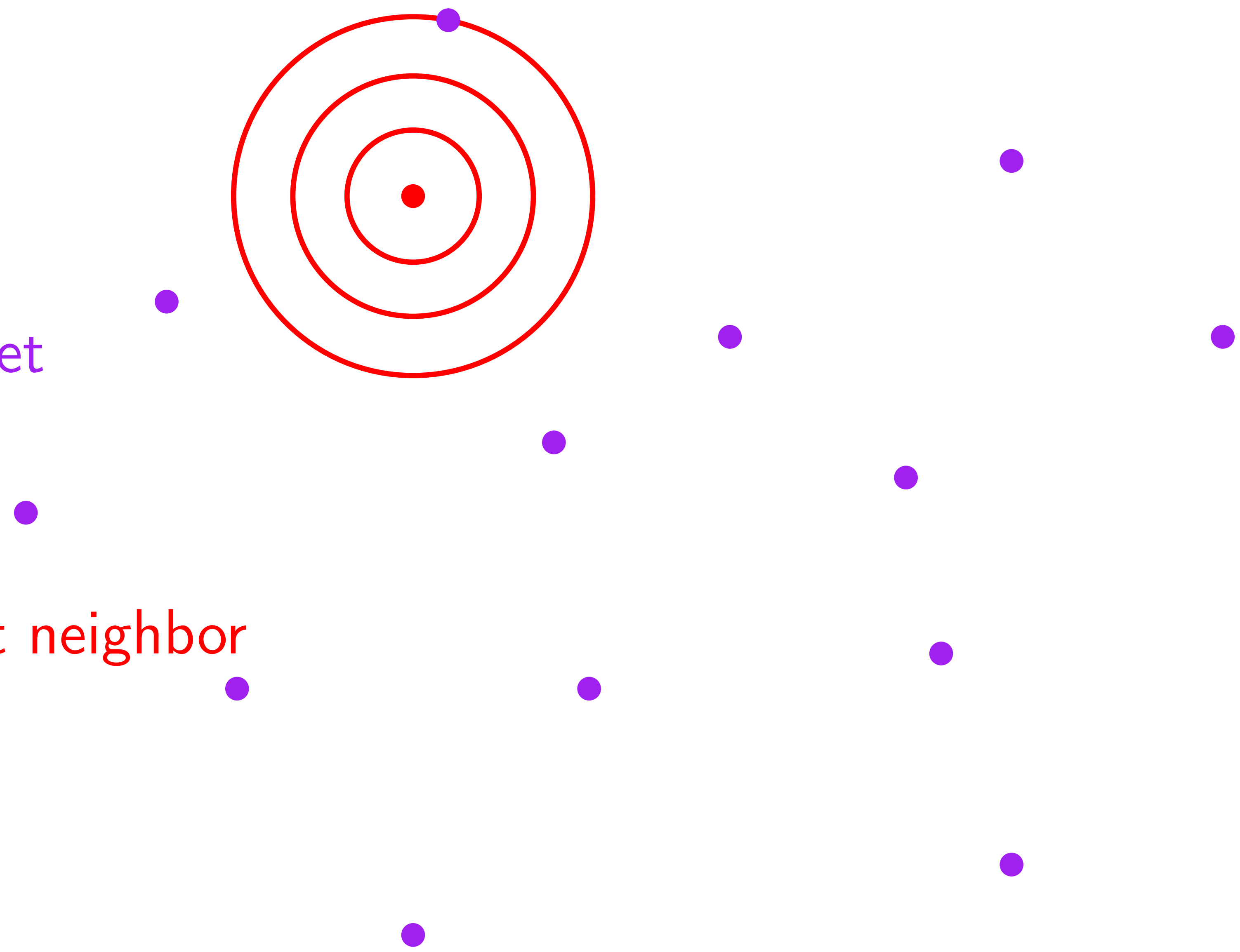
Point set

Query

Delaunay Triangulation: definition, empty circle property



Delaunay Triangulation: definition, empty circle property

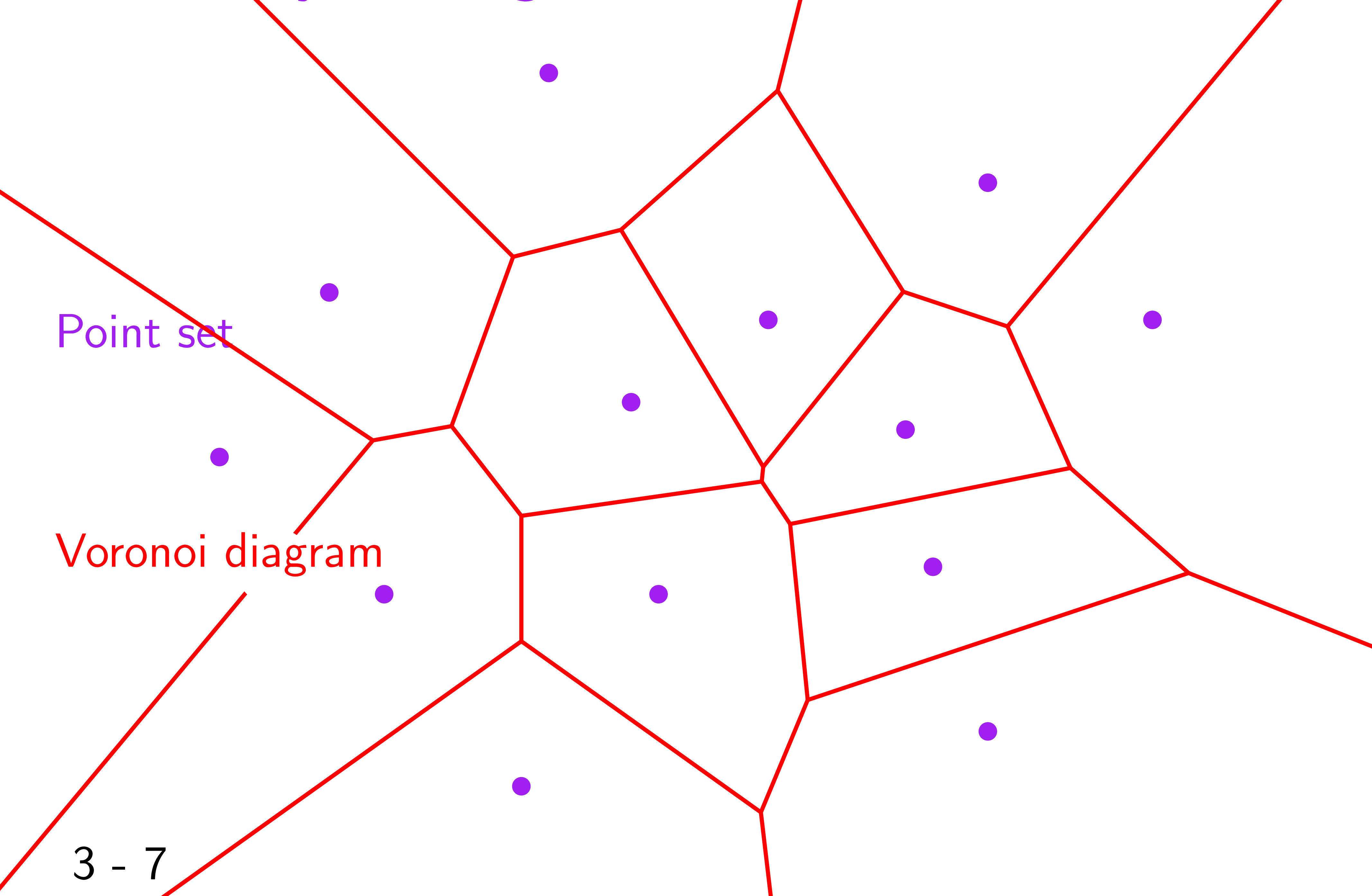


Point set

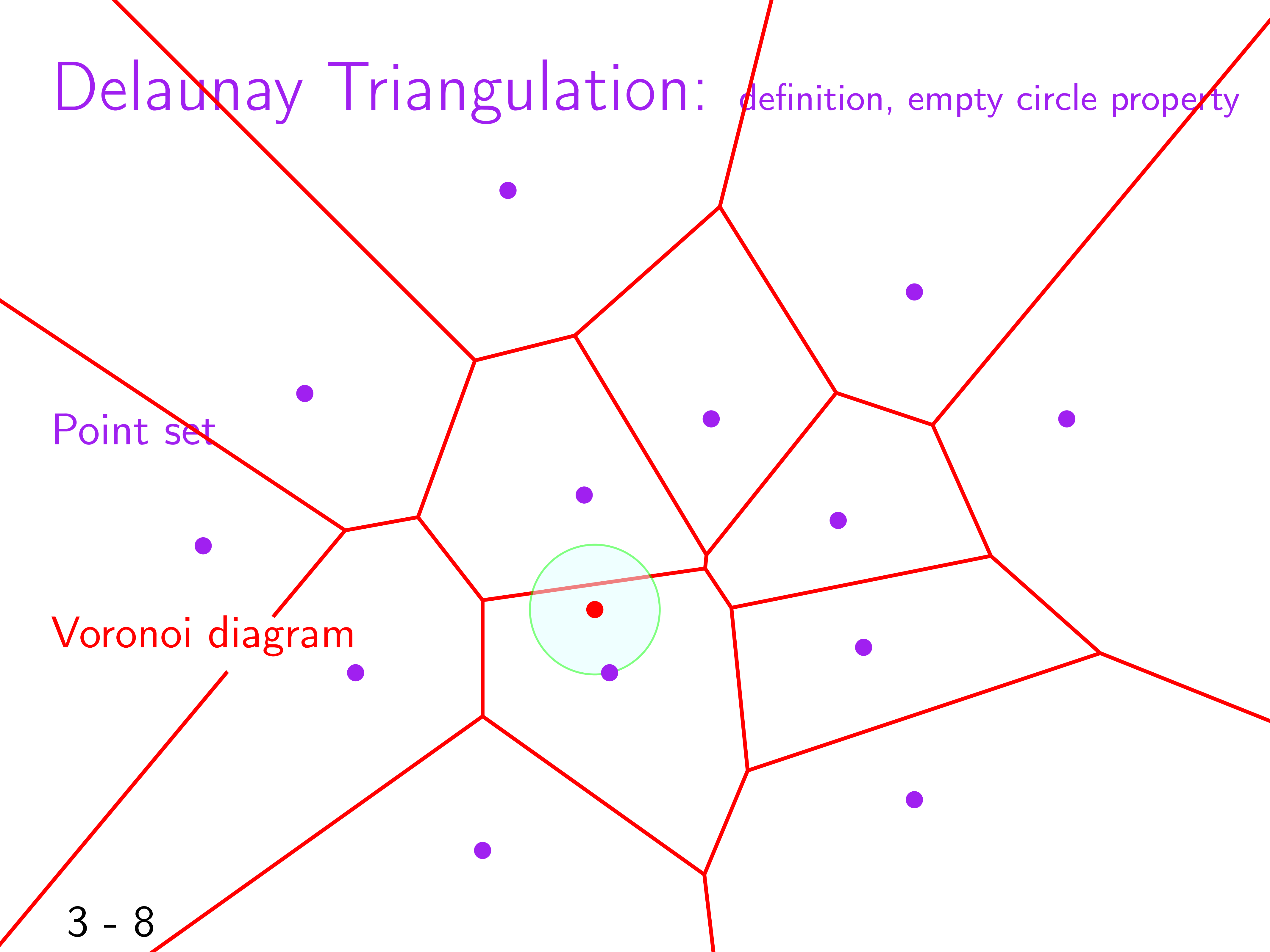
Query

Nearest neighbor

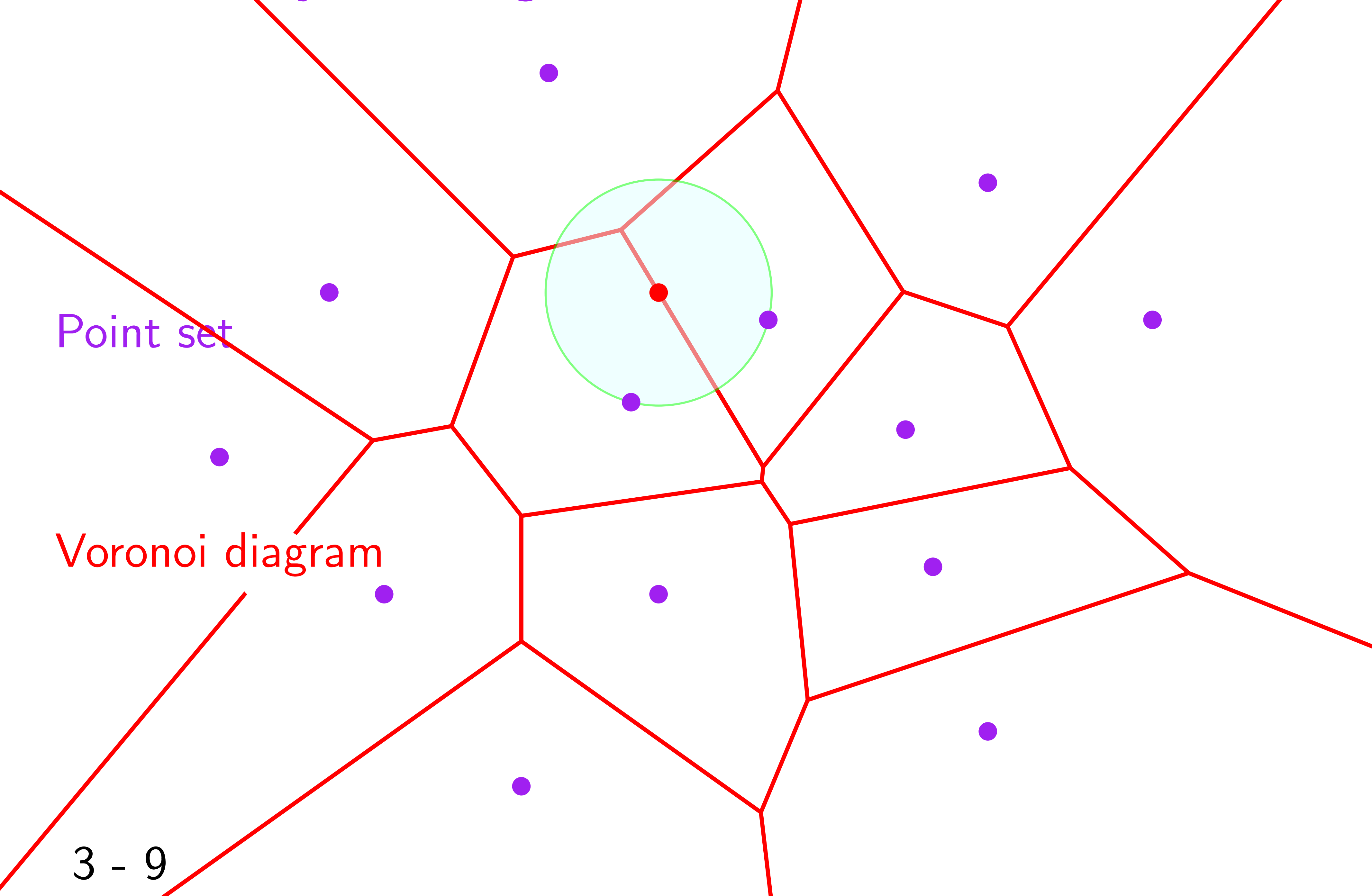
Delaunay Triangulation: definition, empty circle property



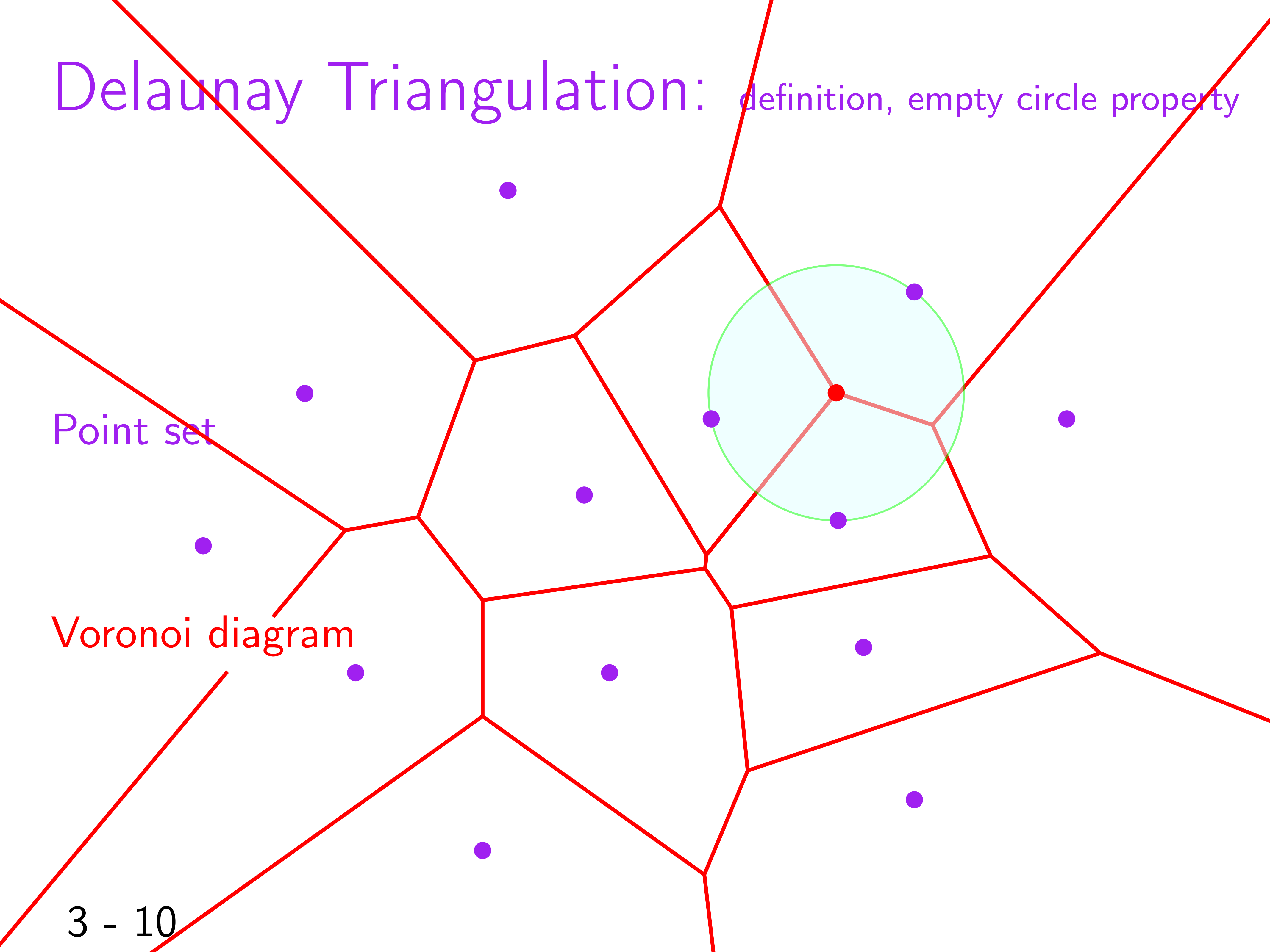
Delaunay Triangulation: definition, empty circle property



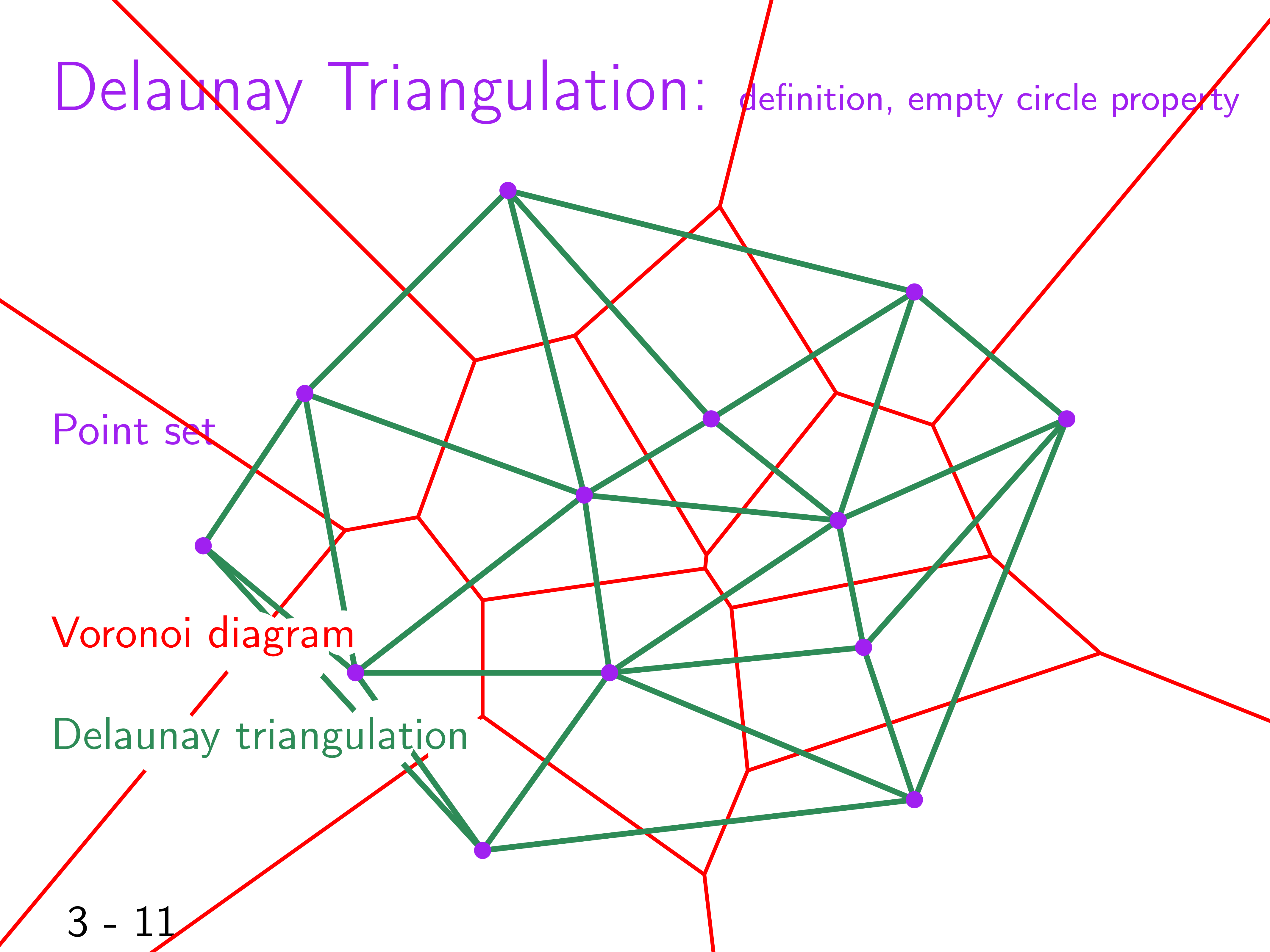
Delaunay Triangulation: definition, empty circle property



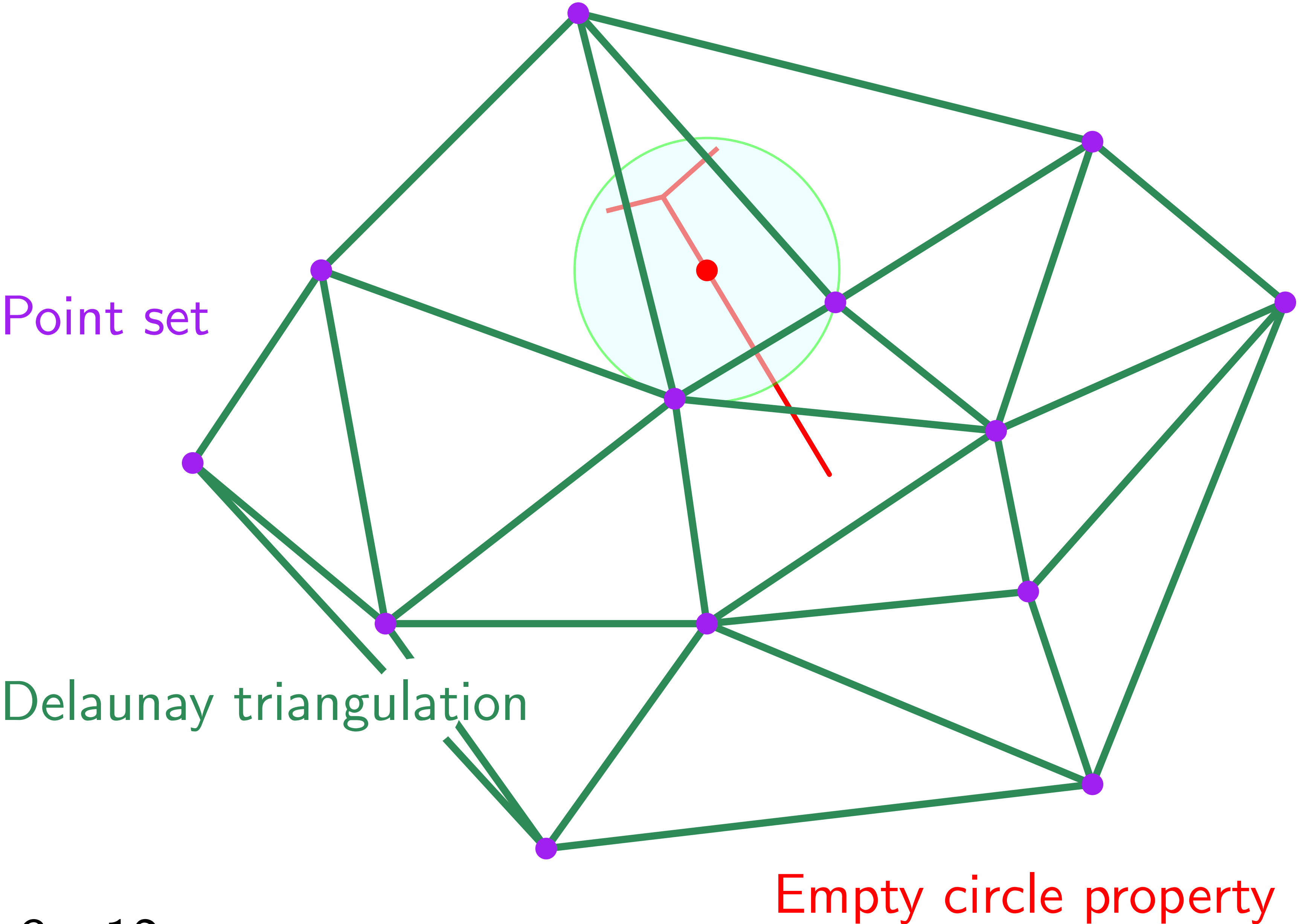
Delaunay Triangulation: definition, empty circle property



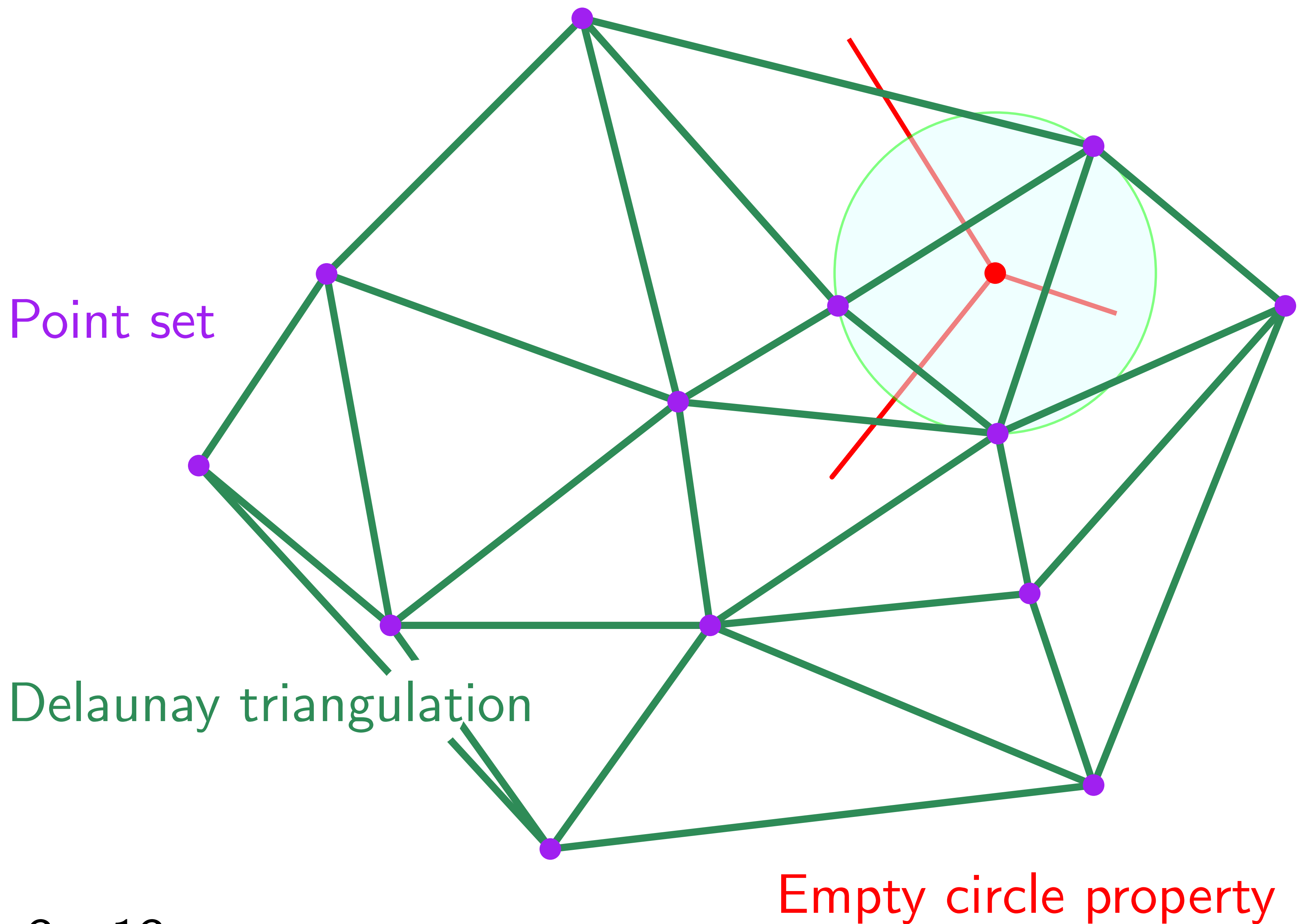
Delaunay Triangulation: definition, empty circle property



Delaunay Triangulation: definition, empty circle property



Delaunay Triangulation: definition, empty circle property



~~Delaunay~~ Triangulation: size

~~Delaunay~~ Triangulation: size

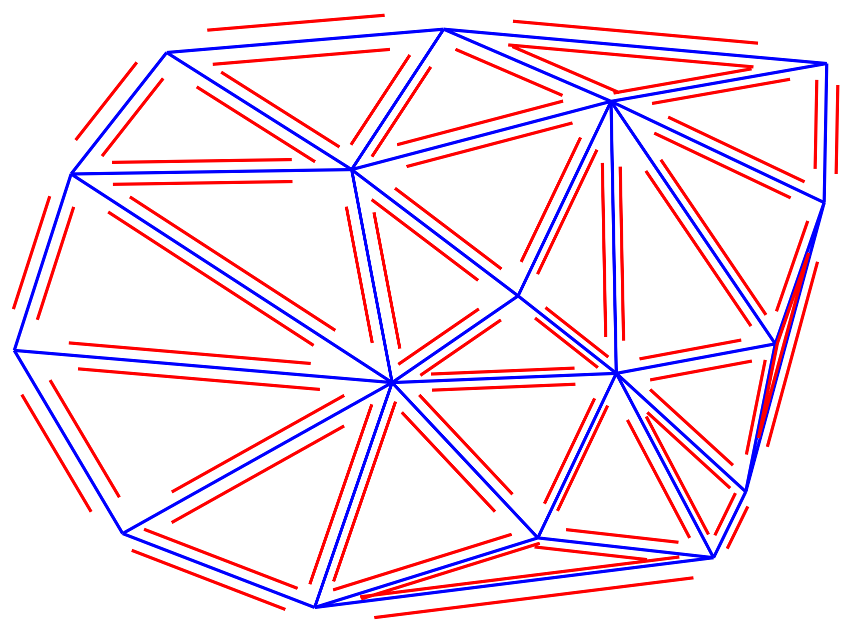
Vertices Edges Faces

Euler relation $n - e + t + 1 = 2$

~~Delaunay~~ Triangulation: size

	Vertices	Edges	Faces
Euler relation	$n -$	e	$+ \ t + 1 \ = 2$

triangular faces $\quad 3t + k = 2e$



~~Delaunay~~ Triangulation: size

	Vertices		Edges		Faces		
Euler relation	n	$-$	e	$+$	$t+1$		$=2$

triangular faces $3t + k = 2e$

$$t = 2n - k - 2 < 2n$$

$$e = 3n - k - 3 < 3n$$

~~Delaunay~~ Triangulation: size

	Vertices	Edges		Faces		
Euler relation	n $-$	e	$+$	$t+1$		$=2$
triangular faces	$3t+k=2e$					

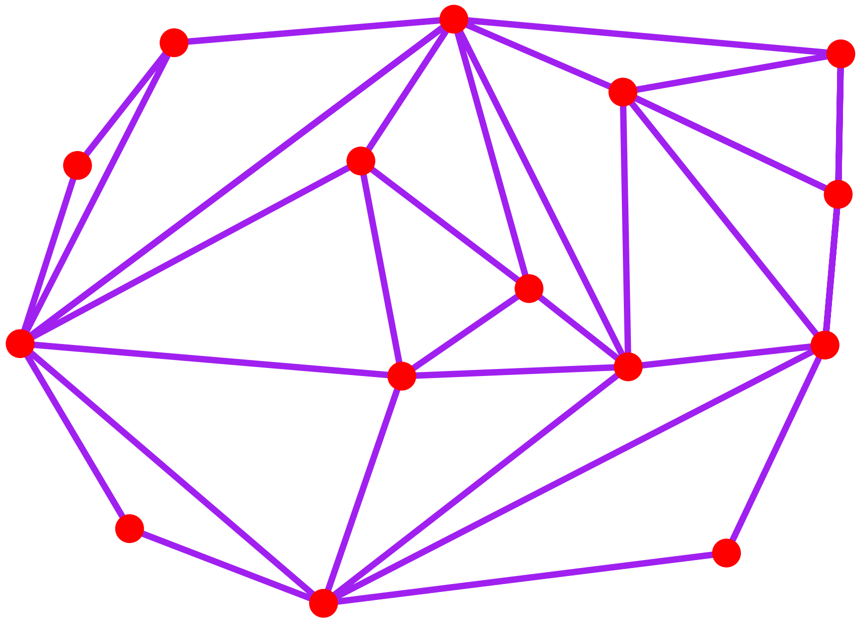
$$\sum_{p\in S} d^\circ(p) = 2e = 6n - 2k - 6$$

$$\mathbb{E}(d^\circ(p)) = \frac{1}{n}\sum_{p\in S} d^\circ(p) < 6$$

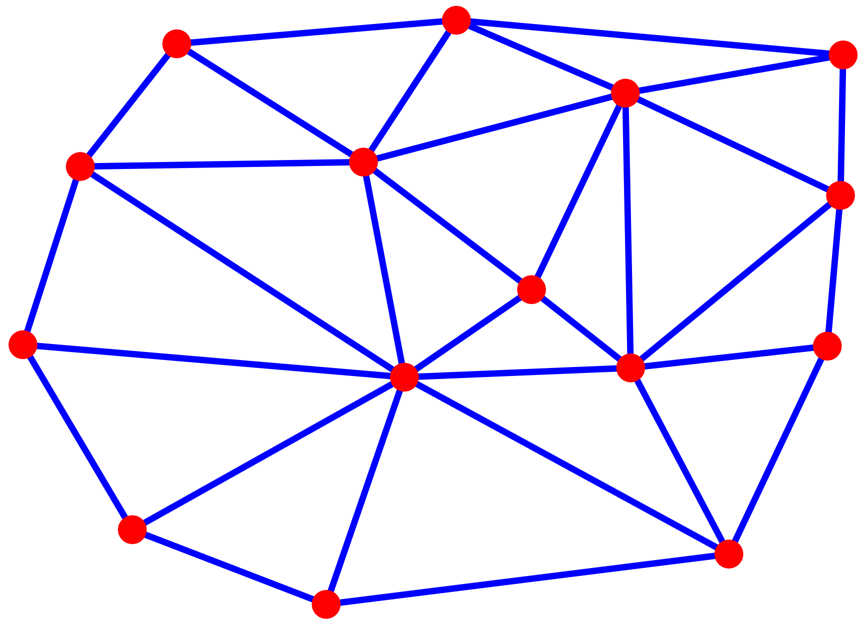
average on the choice of point p in set of points S

Delaunay Triangulation: max-min angle

Delaunay Triangulation: max-min angle

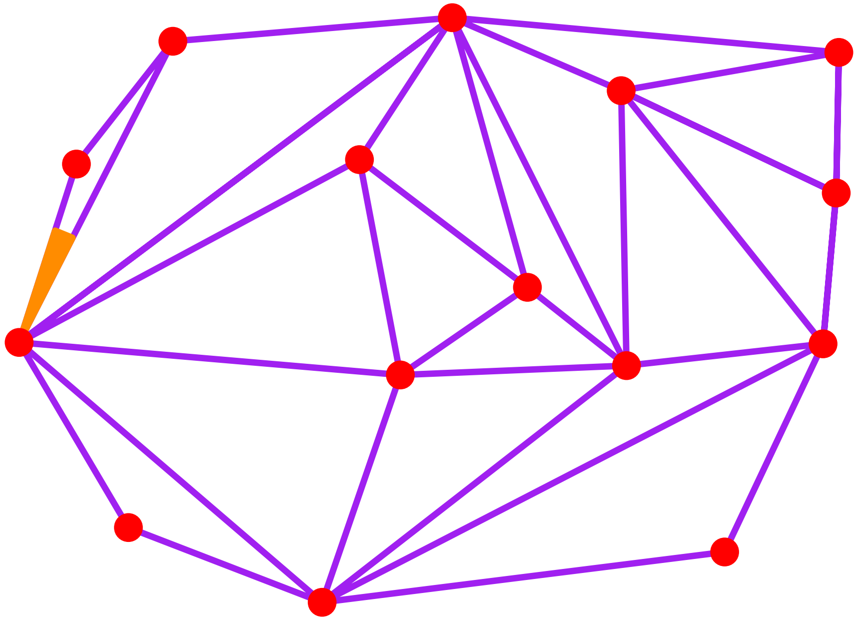


Triangulation

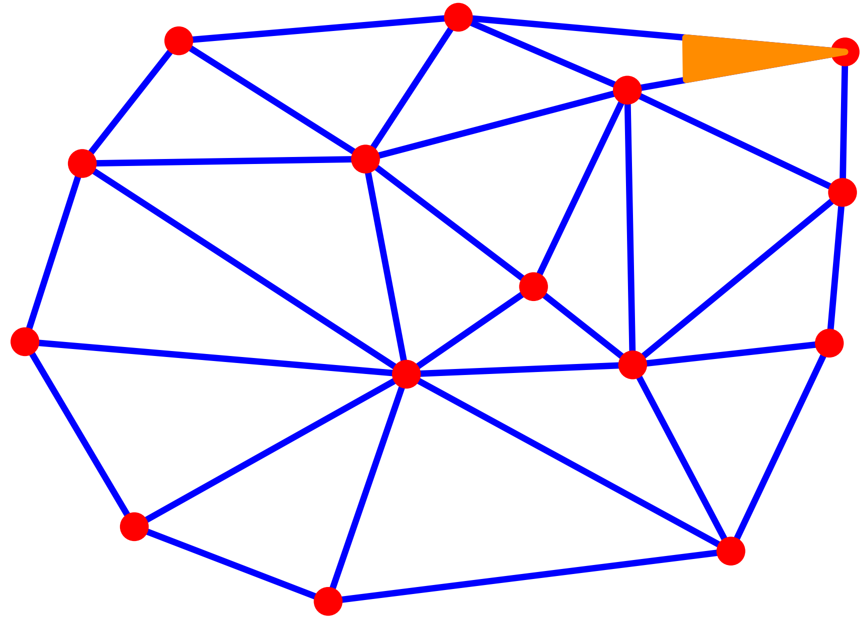


Delaunay

Delaunay Triangulation: max-min angle



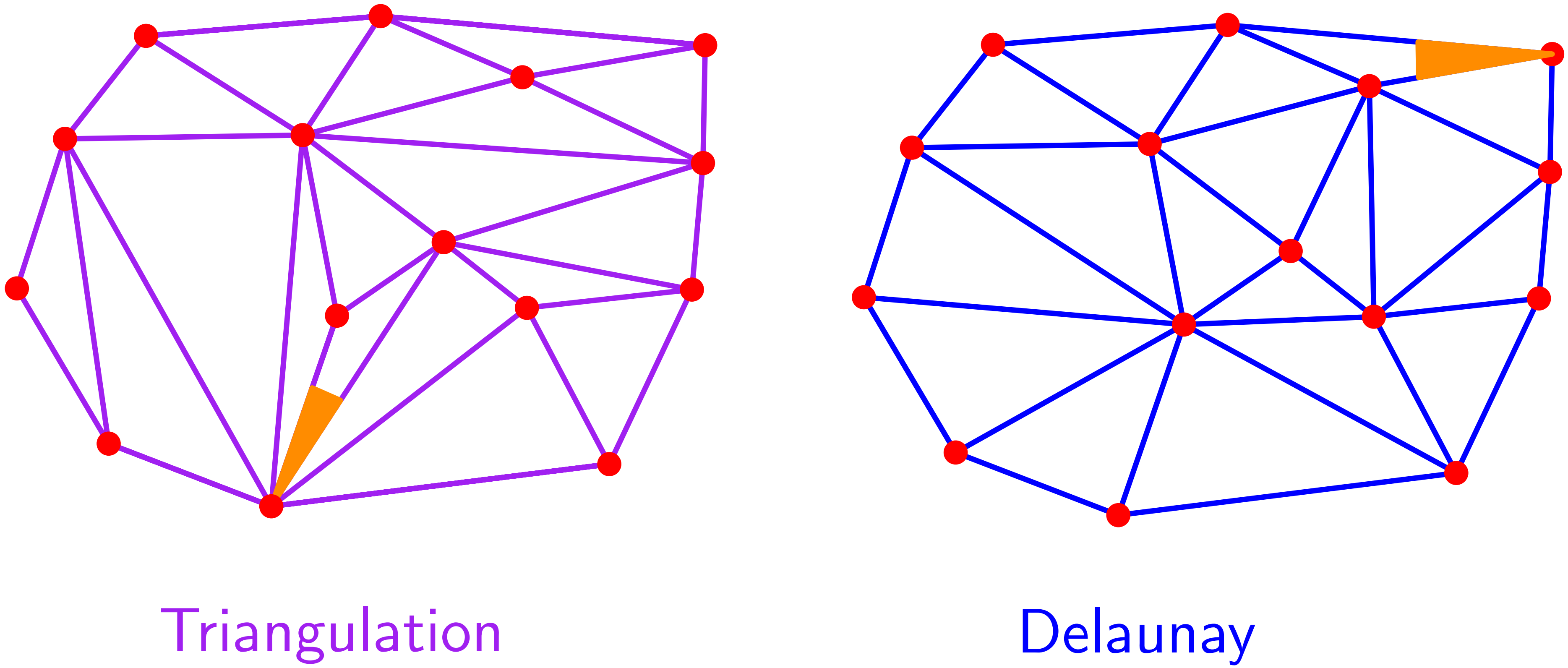
Triangulation



Delaunay

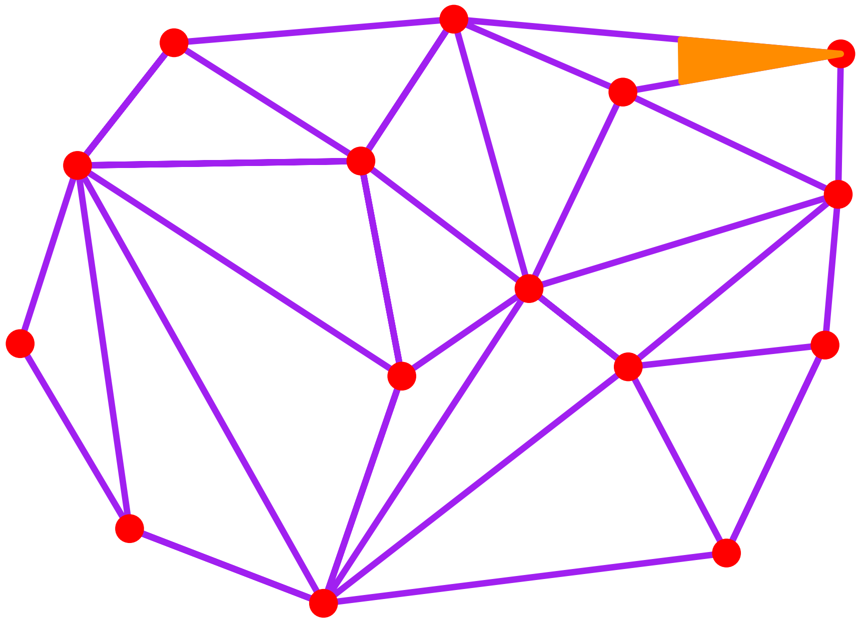
smallest angle

Delaunay Triangulation: max-min angle

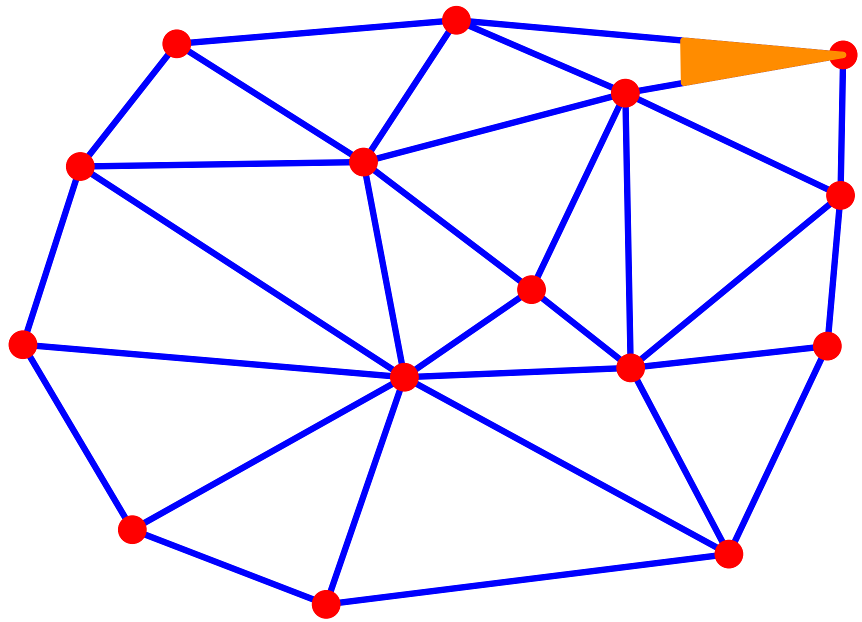


smallest angle

Delaunay Triangulation: max-min angle



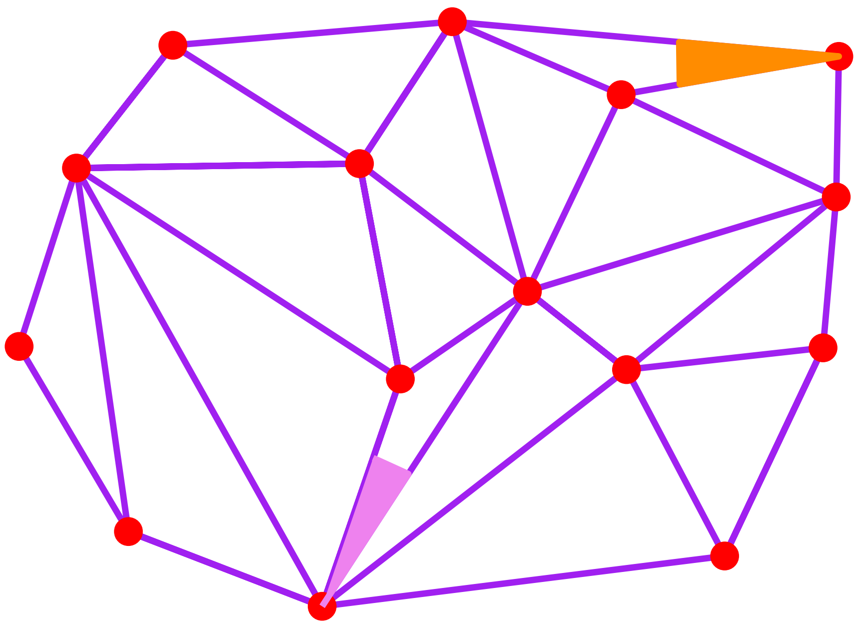
Triangulation



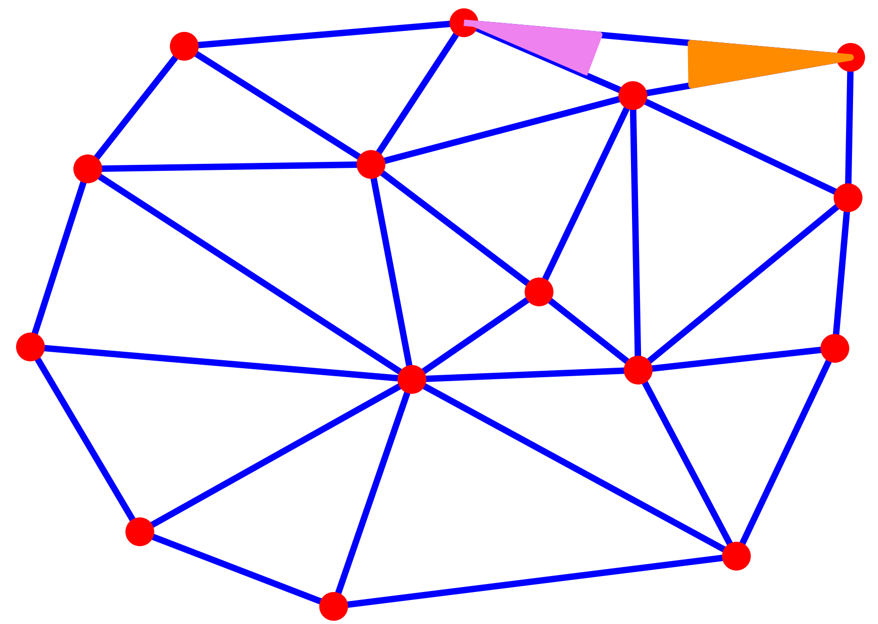
Delaunay

smallest angle

Delaunay Triangulation: max-min angle



Triangulation



Delaunay

smallest angle

second smallest angle

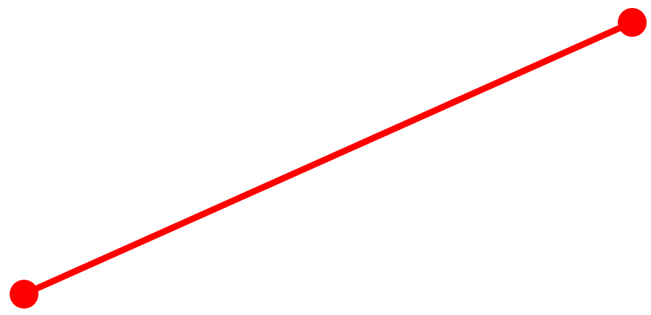
Delaunay Triangulation: max-min angle

Proof

Delaunay Triangulation: max-min angle

Definition

Delaunay edge

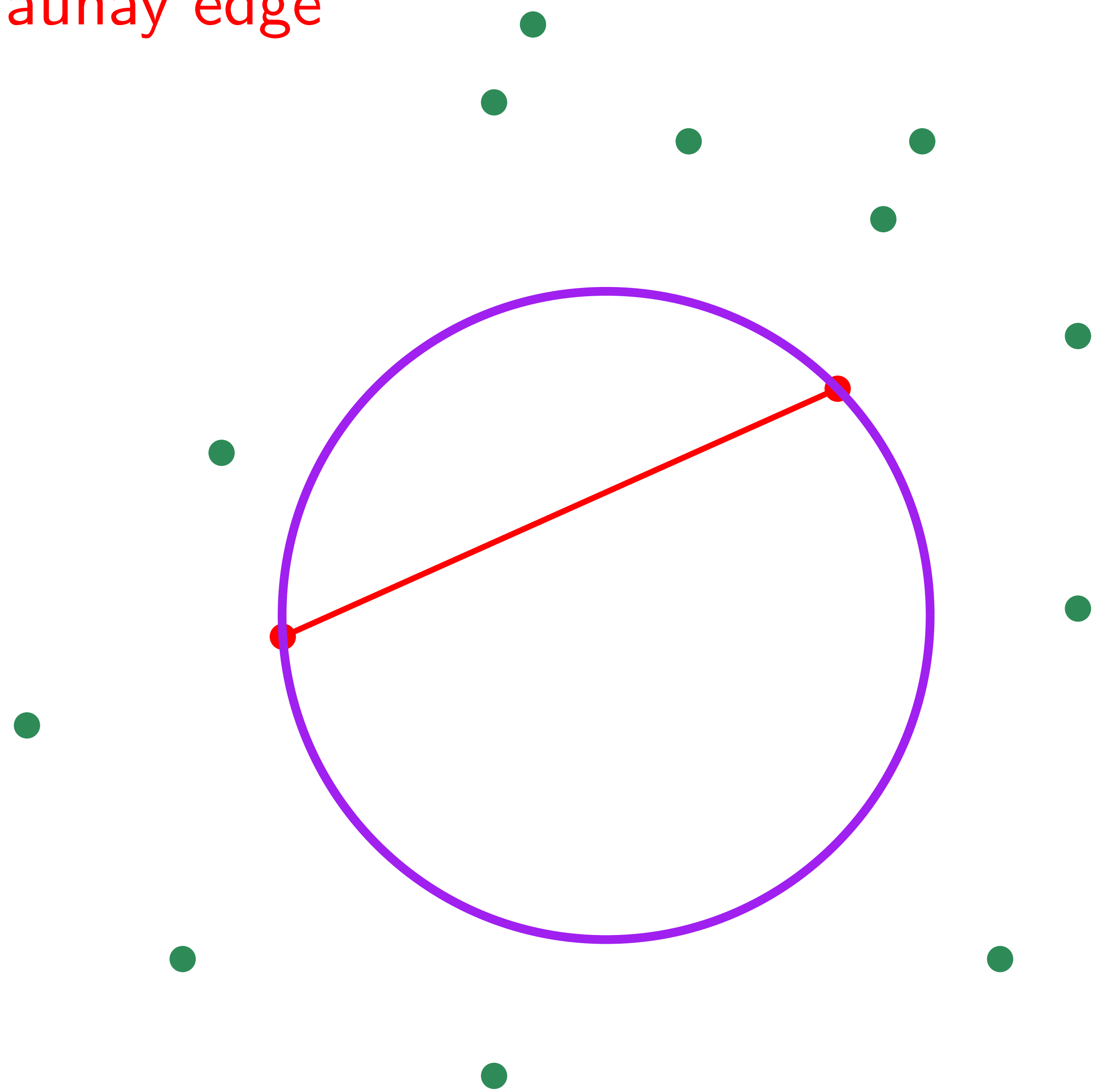


Delaunay Triangulation: max-min angle

Definition

Delaunay edge

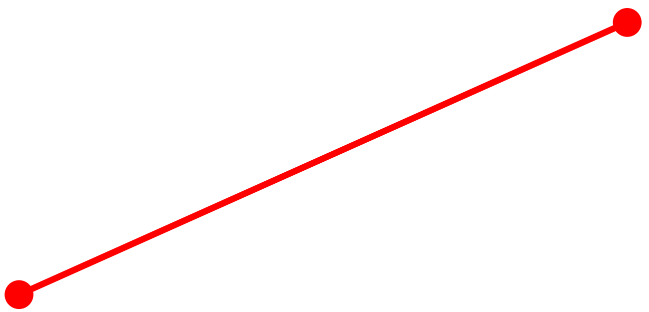
∃ empty circle



Delaunay Triangulation: max-min angle

Definition

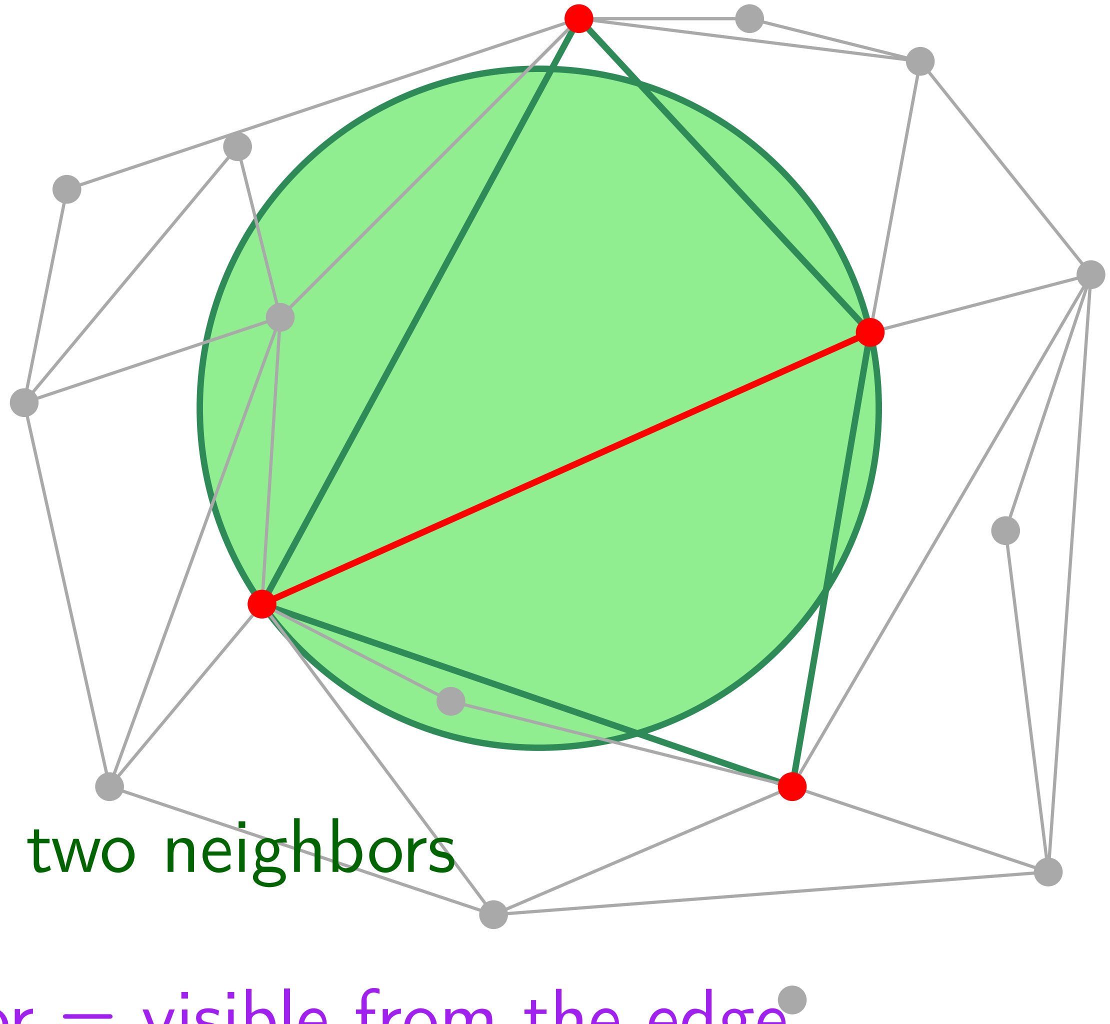
locally Delaunay edge w.r.t. a triangulation



Delaunay Triangulation: max-min angle

Definition

locally Delaunay edge w.r.t. a triangulation



∃ circle

not enclosing the two neighbors

neighbor = visible from the edge

Delaunay Triangulation: max-min angle

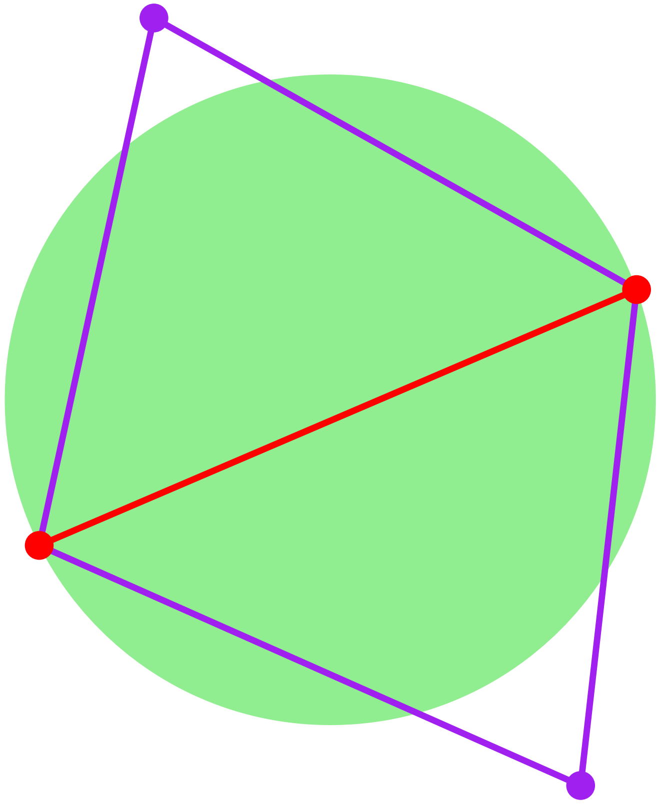
Lemma $(\forall$ edge: locally Delaunay$) \Longleftrightarrow$ Delaunay

Delaunay Triangulation: max-min angle

Lemma (∀ edge: locally Delaunay) $\Longleftrightarrow$ Delaunay

Proof:

choose an edge

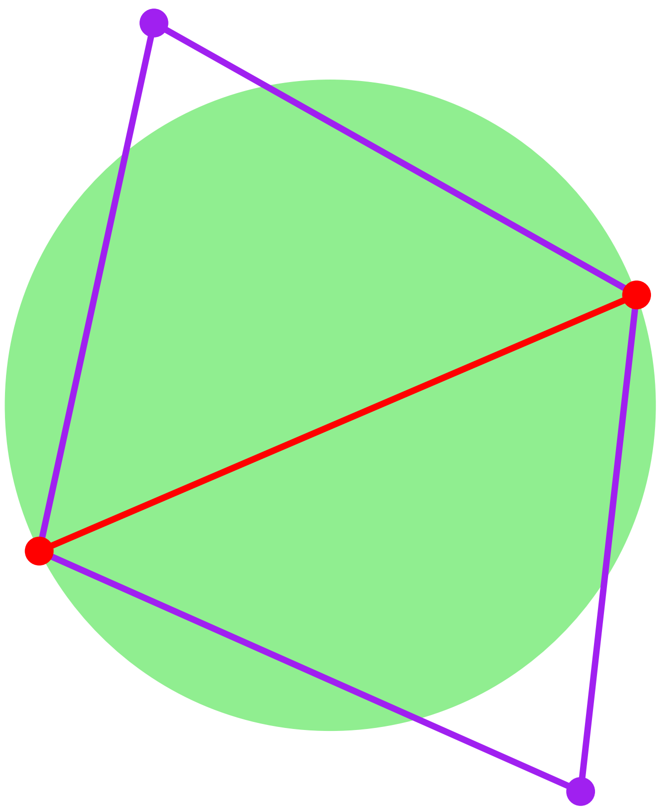


Delaunay Triangulation: max-min angle

Lemma (∀ edge: locally Delaunay) $\iff$ Delaunay

Proof:

- choose an edge
- edges of the quadrilateral
  are locally Delaunay

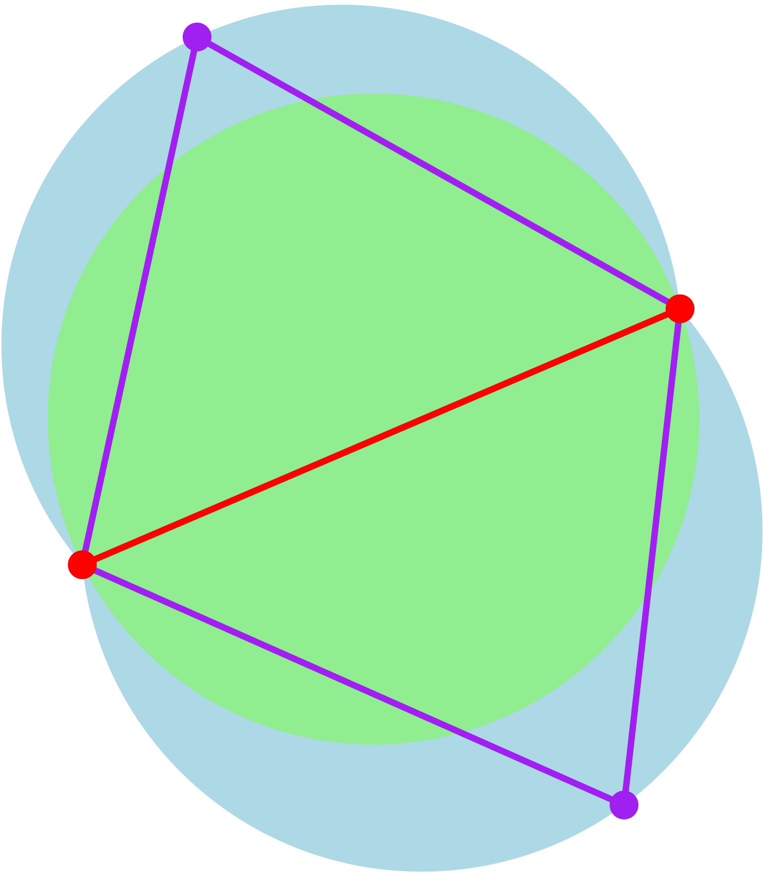


Delaunay Triangulation: max-min angle

Lemma (∀ edge: locally Delaunay) $\iff$ Delaunay

Proof:

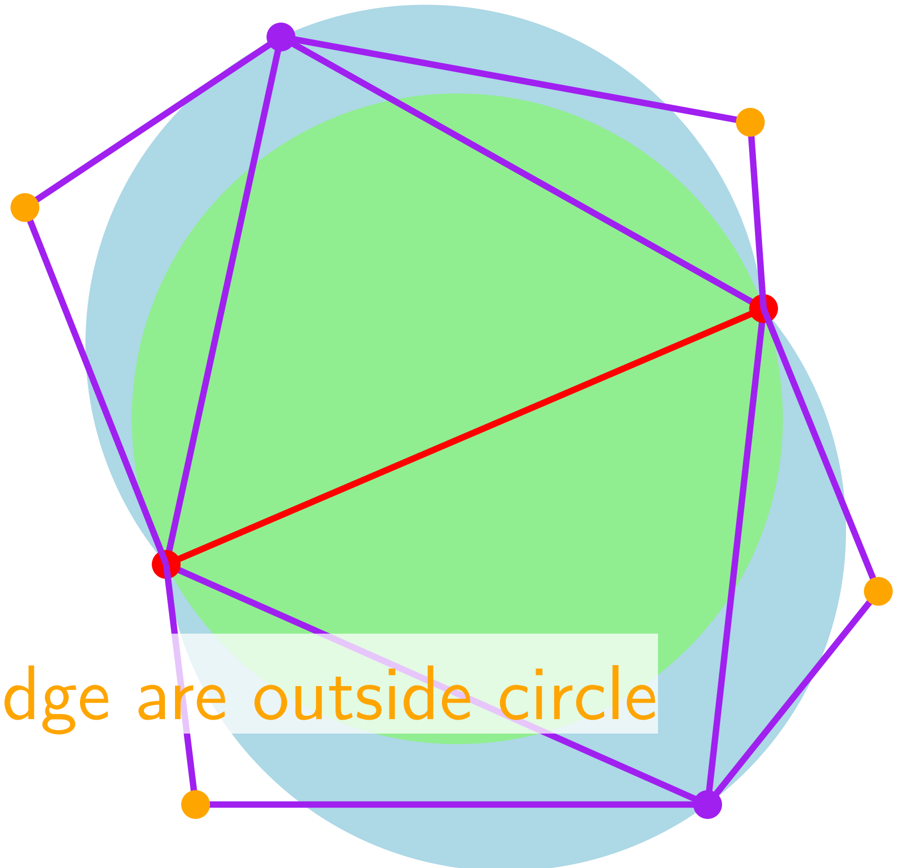
- choose an edge
- edges of the quadrilateral are locally Delaunay



Delaunay Triangulation: max-min angle

Lemma (∀ edge: locally Delaunay) $\iff$ Delaunay

Proof:

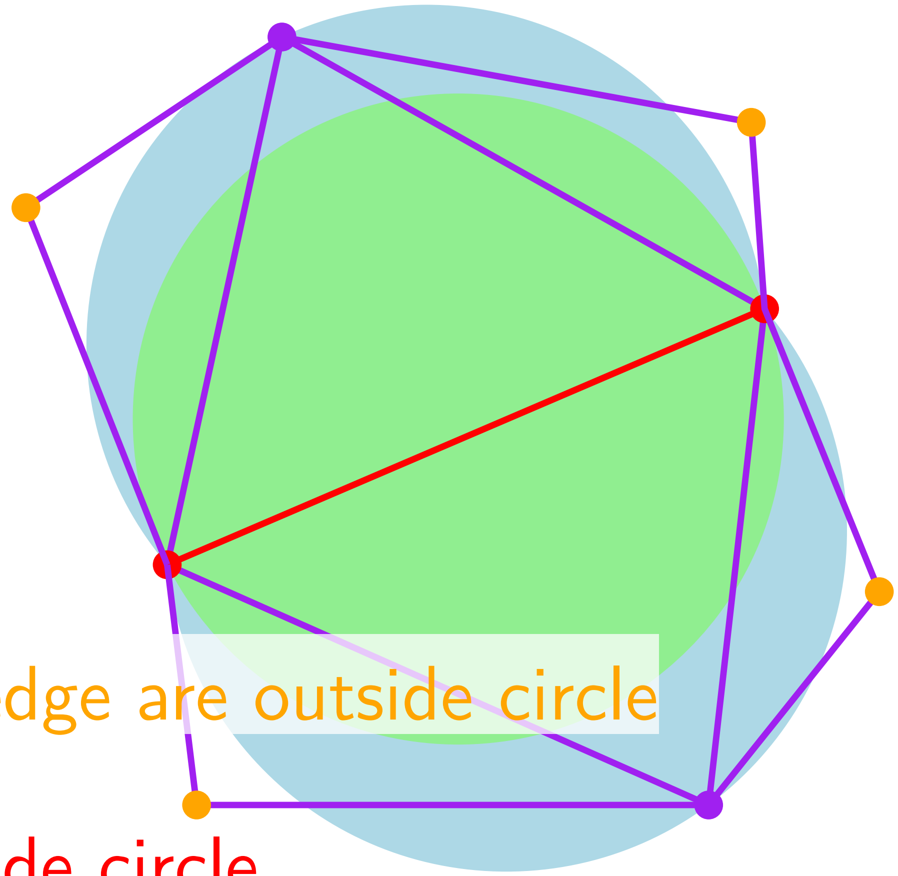


- choose an edge
- edges of the quadrilateral are locally Delaunay
- Vertices visible through one edge are outside circle

Delaunay Triangulation: max-min angle

Lemma (∀ edge: locally Delaunay) $\Longleftrightarrow$ Delaunay

Proof:



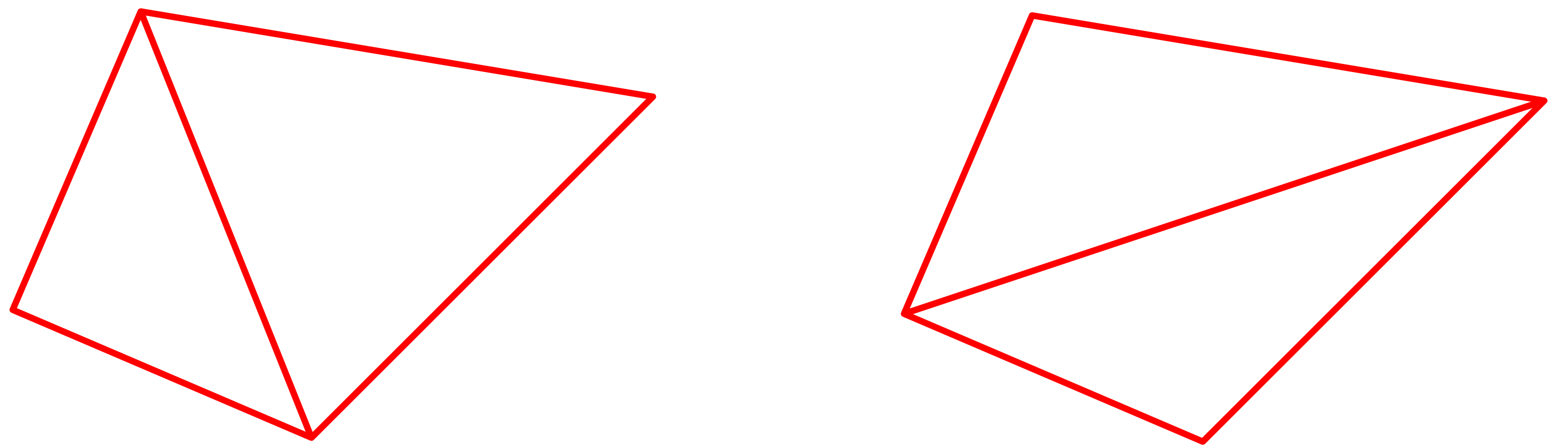
- choose an edge
- edges of the quadrilateral are locally Delaunay
- Vertices visible through one edge are outside circle
- Induction $\rightarrow$ all vertices outside circle

Delaunay Triangulation: max-min angle

Lemma   For four points in convex position

Delaunay $\Longleftrightarrow$ maximize the smallest angle

Two possible triangulation

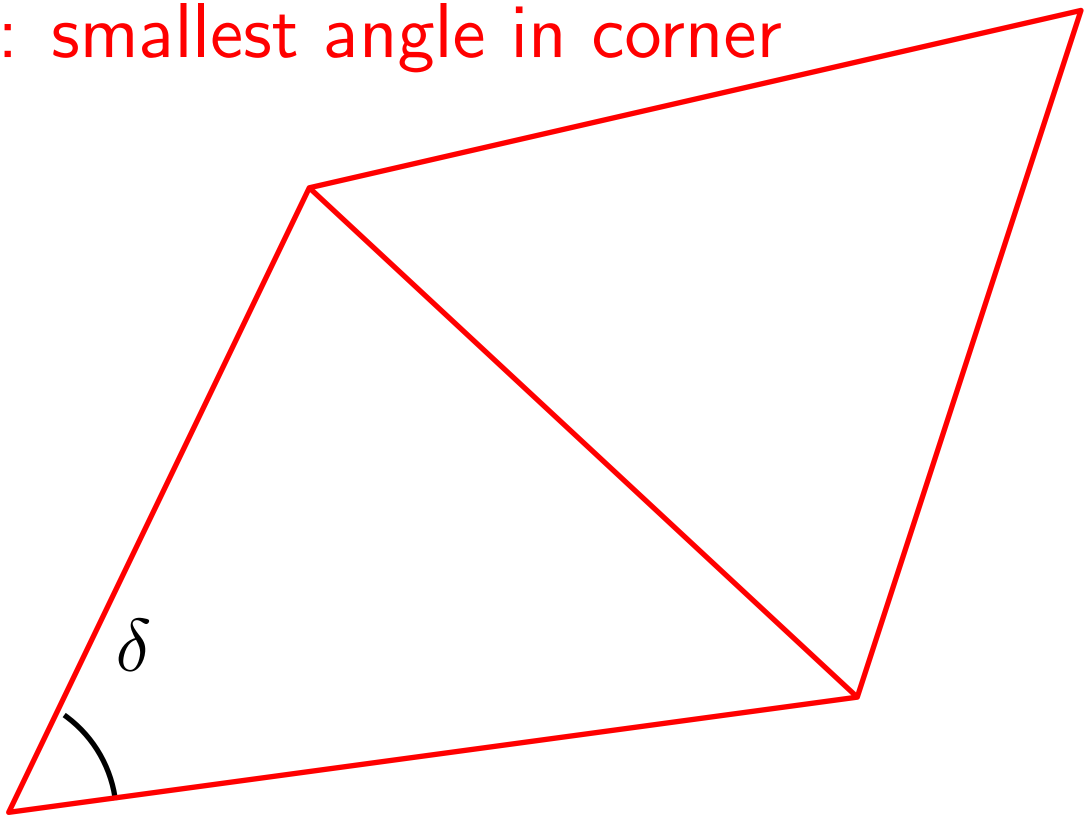


Delaunay Triangulation: max-min angle

Lemma For four points in convex position

Delaunay $\Longleftrightarrow$ maximize the smallest angle

Case 1: smallest angle in corner

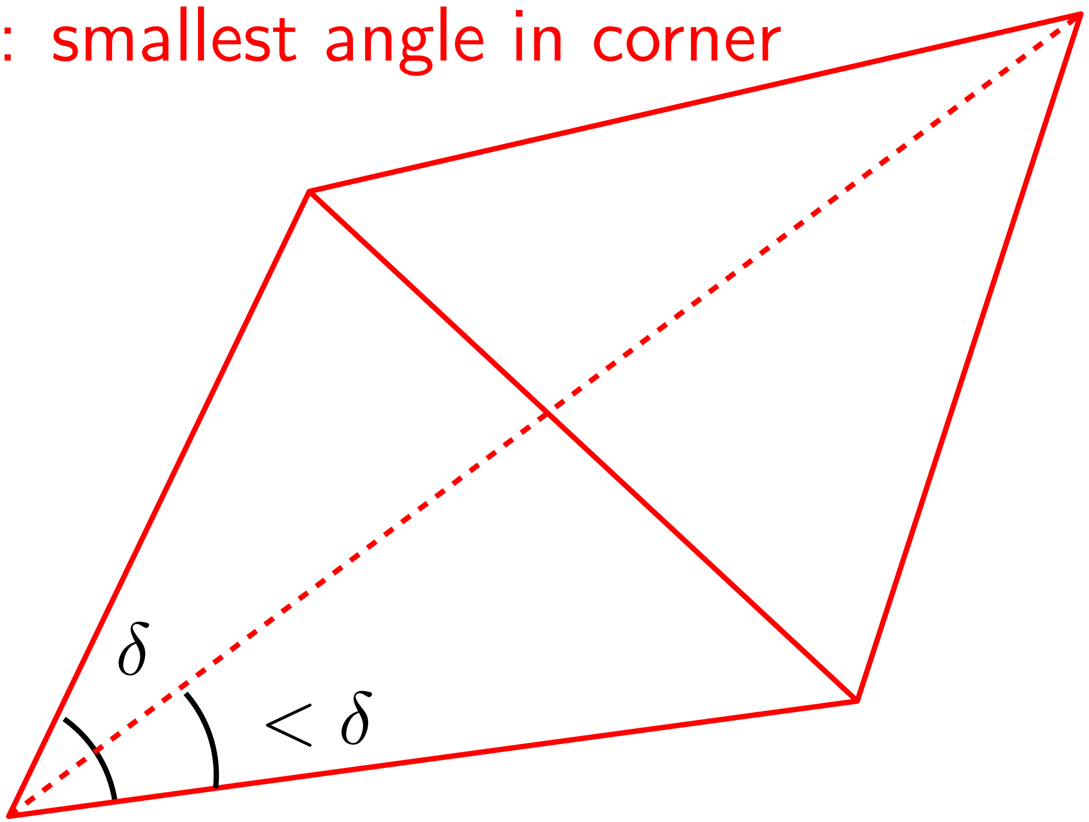


Delaunay Triangulation: max-min angle

Lemma For four points in convex position

Delaunay $\Longleftrightarrow$ maximize the smallest angle

Case 1: smallest angle in corner



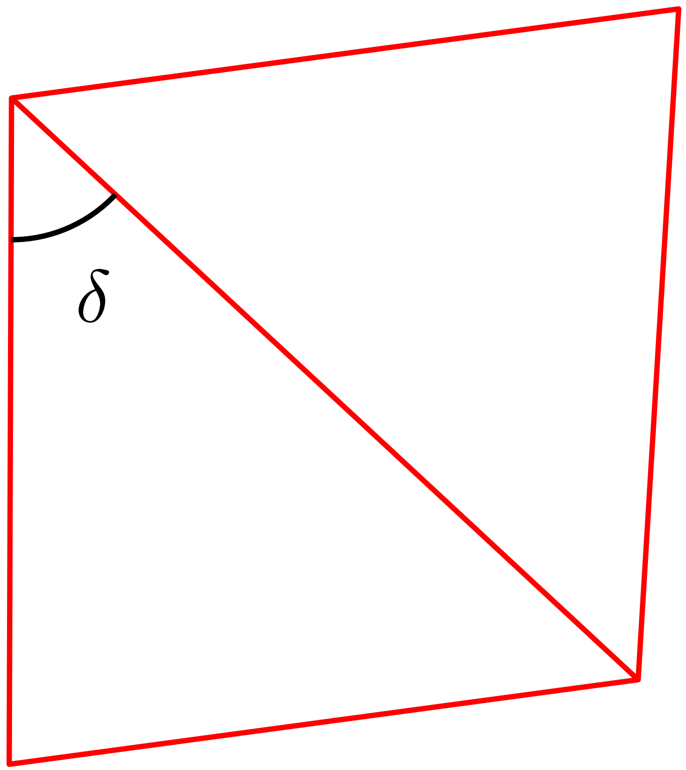
$\exists$ a smaller angle $\in$ other triangulation

Delaunay Triangulation: max-min angle

Lemma For four points in convex position

Delaunay $\Longleftrightarrow$ maximize the smallest angle

Case 2: smallest angle along diagonal

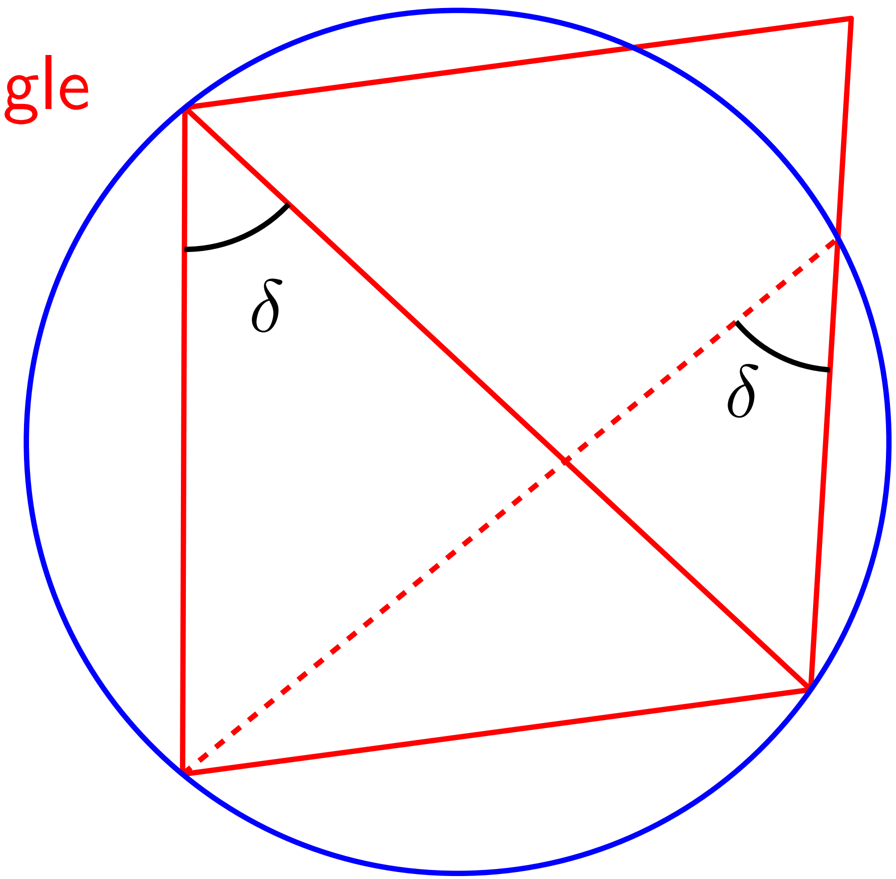


Delaunay Triangulation: max-min angle

Lemma For four points in convex position

Delaunay $\Longleftrightarrow$ maximize the smallest angle

Case 2: smallest angle along diagonal

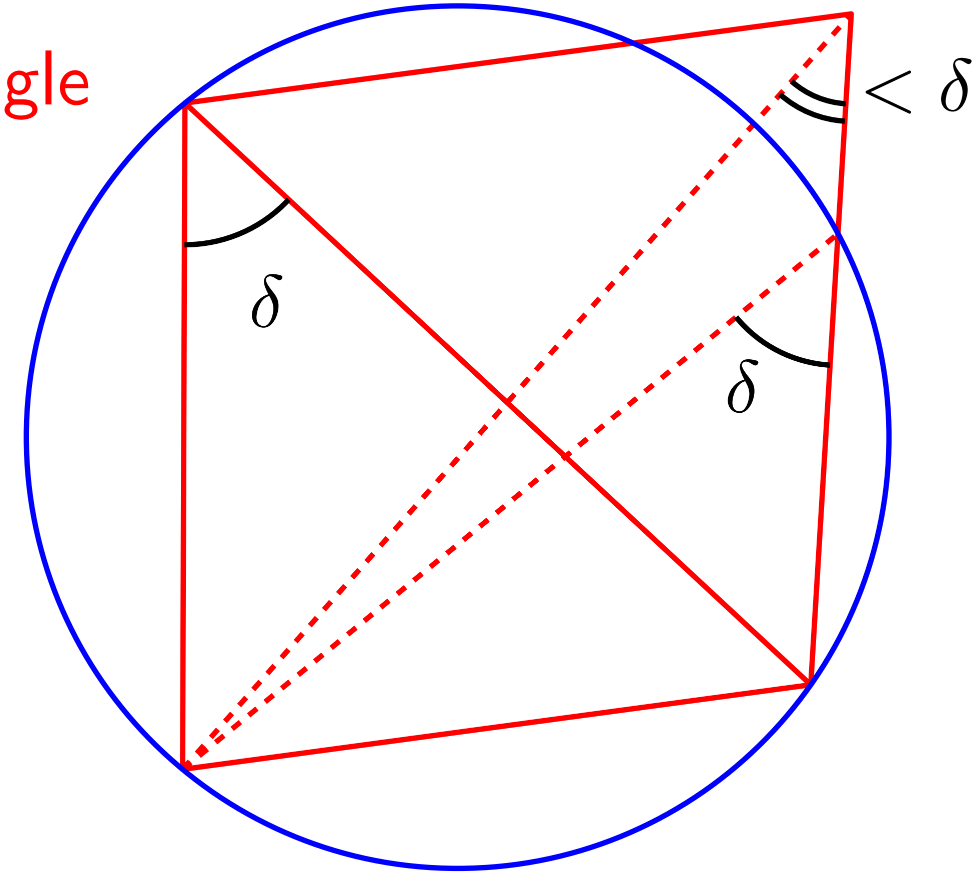


Delaunay Triangulation: max-min angle

Lemma For four points in convex position

Delaunay $\iff$ maximize the smallest angle

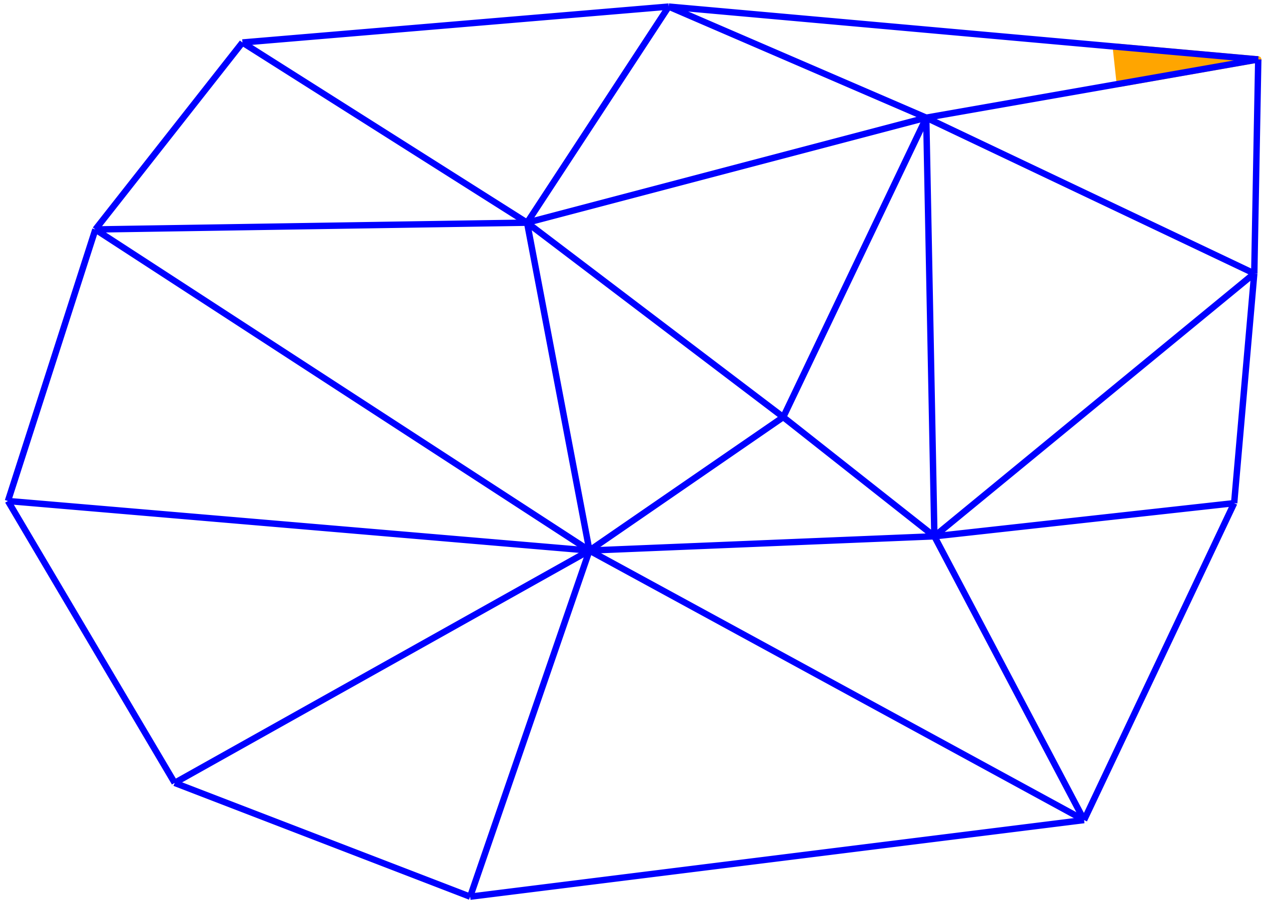
Case 2: smallest angle along diagonal



$\exists$ a smaller angle $\in$ other triangulation

Delaunay Triangulation: max-min angle

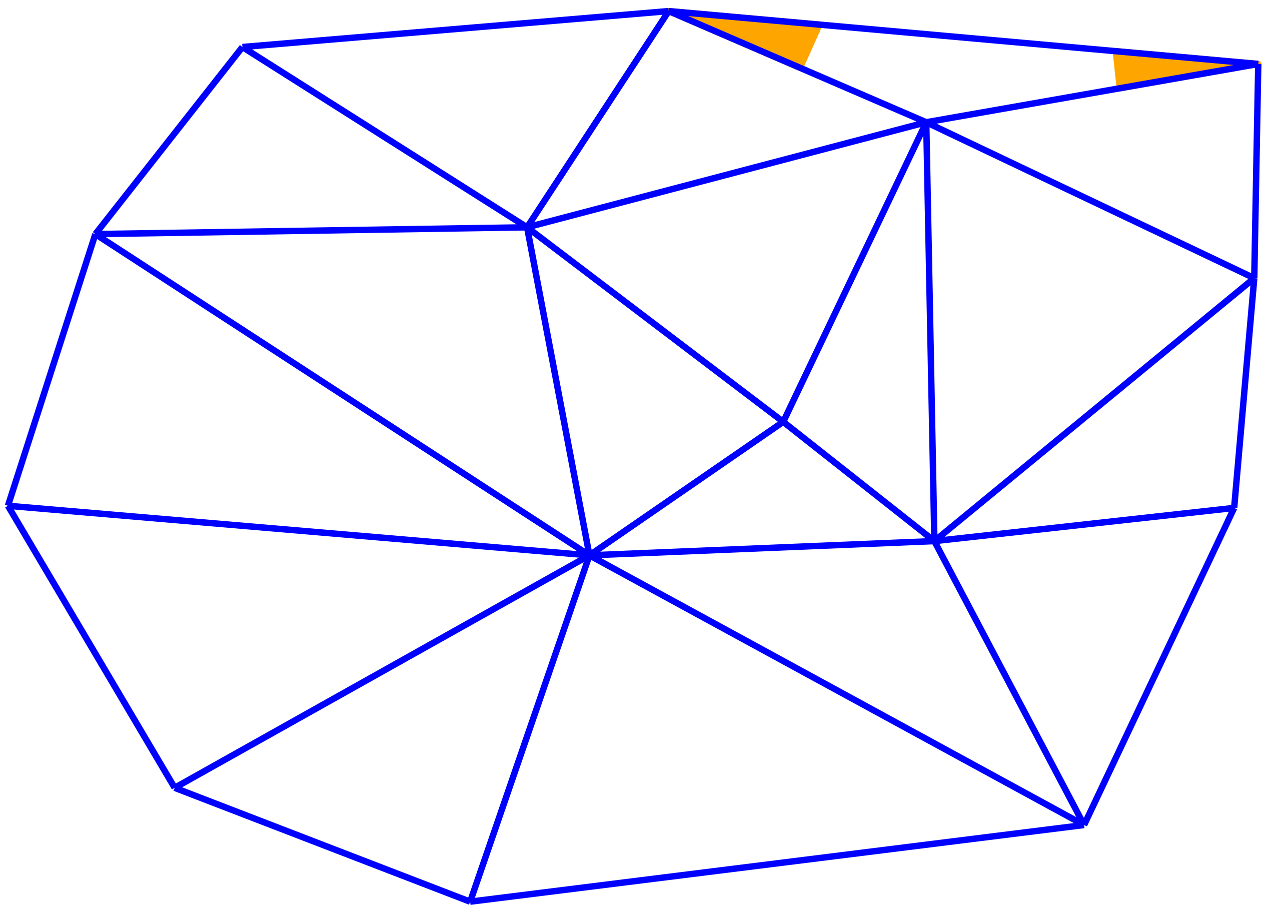
Map: Triangulations $\longrightarrow \mathbb{R}^{6n-3k-4}$ smallest angle α_1



Delaunay Triangulation: max-min angle

Map: Triangulations $\longrightarrow \mathbb{R}^{6n-3k-4}$

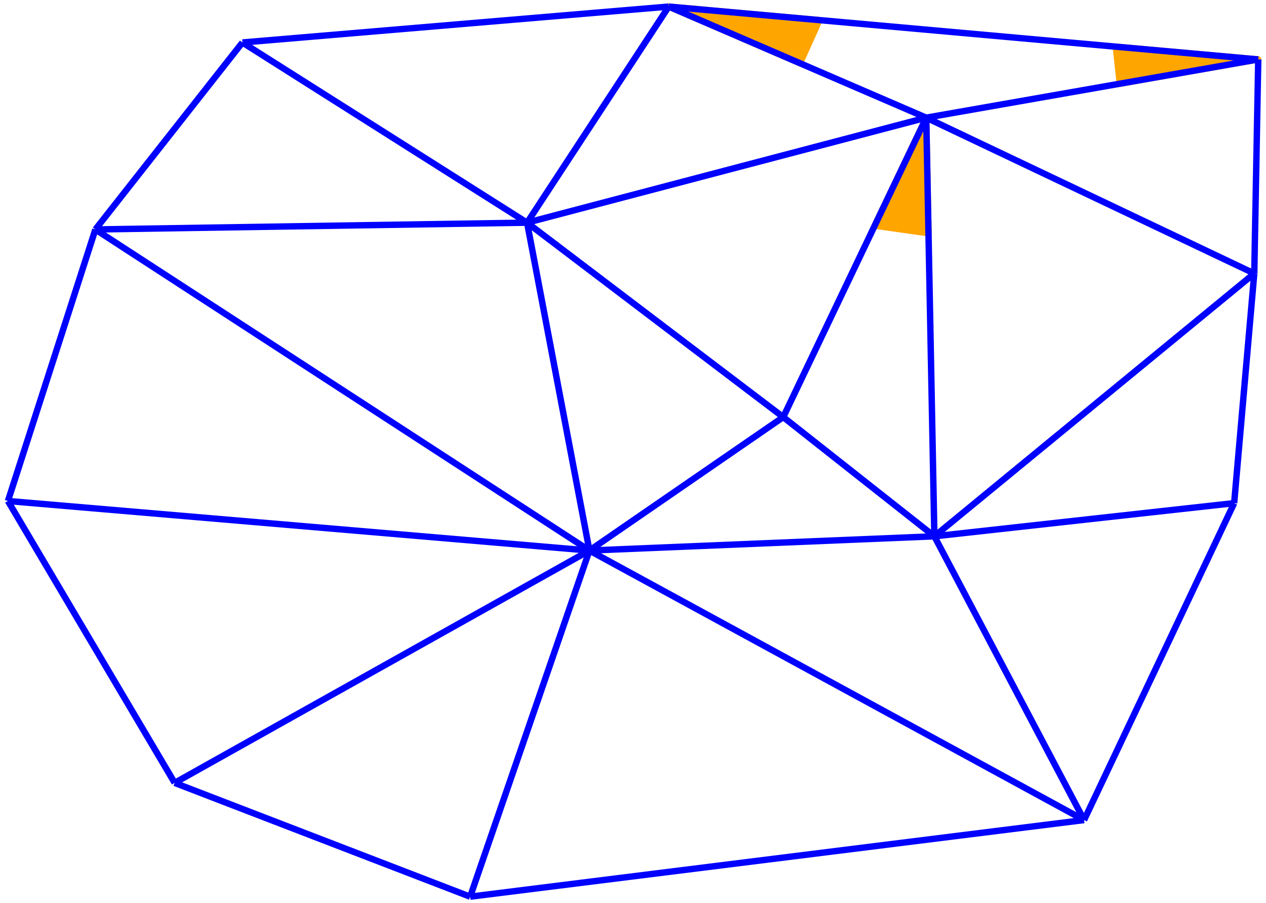
smallest angle α_1
second smallest angle α_2



Delaunay Triangulation: max-min angle

Map: Triangulations $\longrightarrow \mathbb{R}^{6n-3k-4}$

smallest angle α_1
second smallest angle α_2
third smallest angle α_3

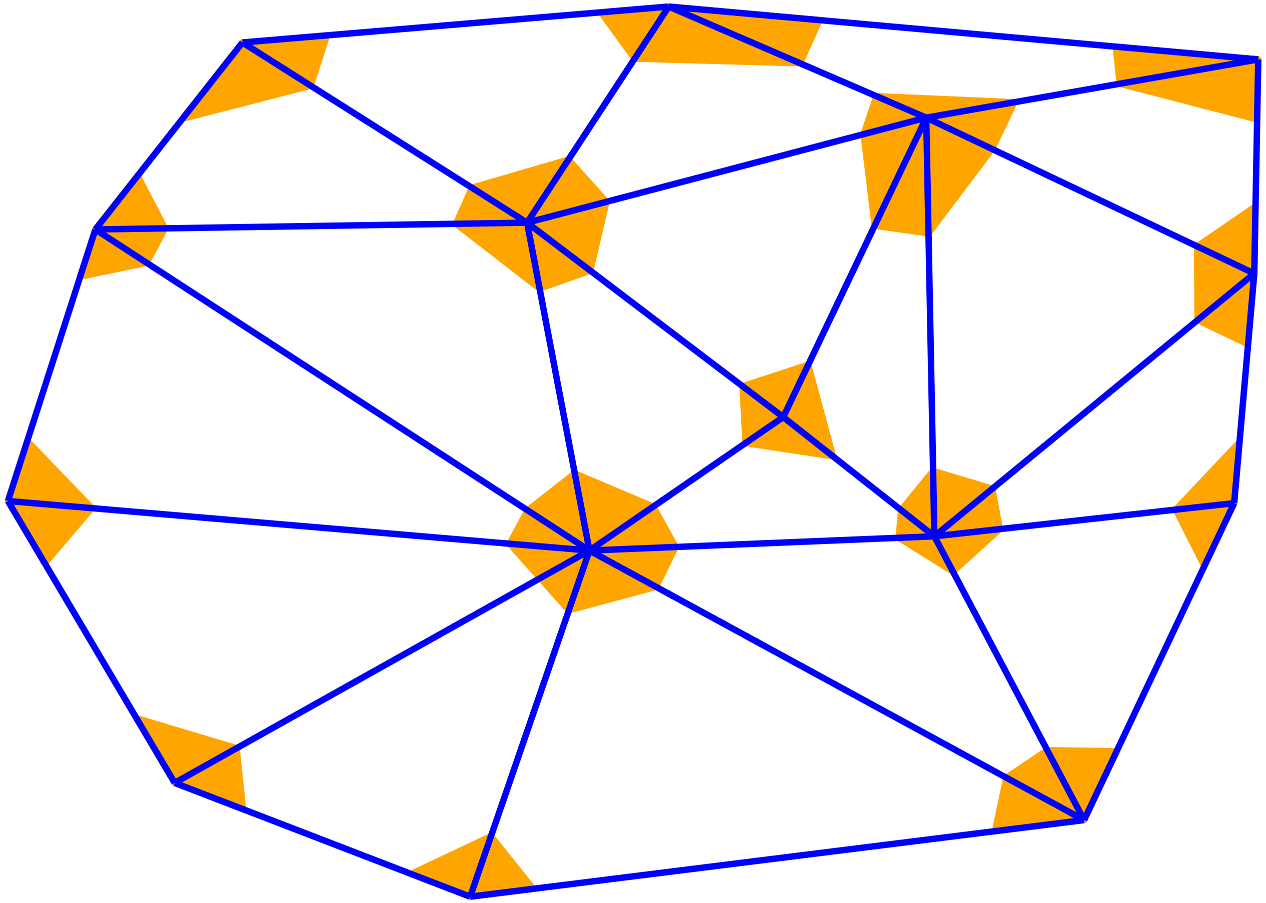


Delaunay Triangulation: max-min angle

Map: Triangulations $\longrightarrow \mathbb{R}^{6n-3k-4}$

$(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{6n-3k-4})$

smallest angle α_1
second smallest angle α_2
third smallest angle α_3

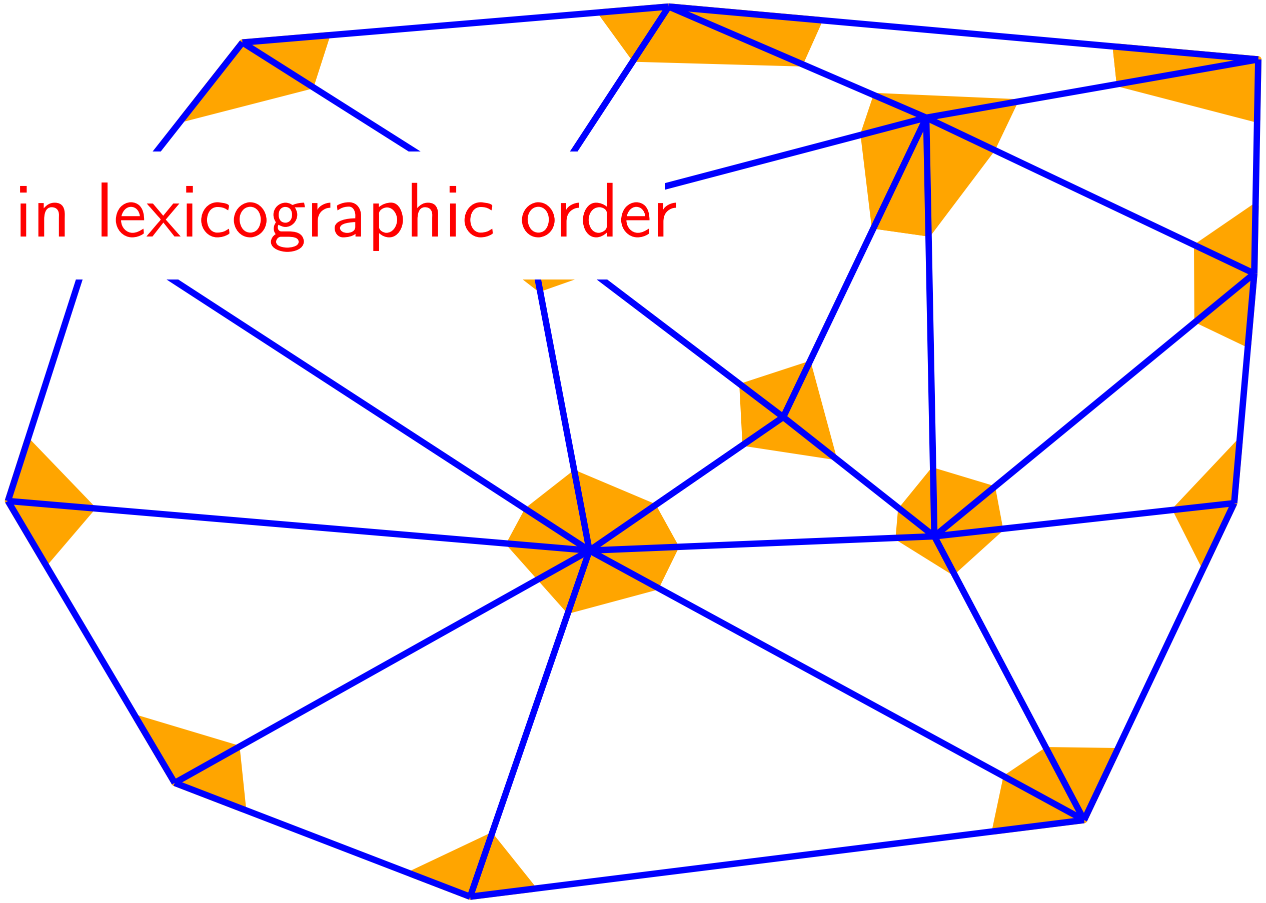


Delaunay Triangulation: max-min angle

Map: Triangulations $\longrightarrow \mathbb{R}^{6n-3k-4}$

$(\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{6n-3k-4})$

smallest angle α_1
second smallest angle α_2
third smallest angle α_3



sort triangulations in lexicographic order

Delaunay Triangulation: max-min angle

Theorem:

Delaunay maximizes minimum angles (in lexicographic order)

Delaunay Triangulation: max-min angle

Theorem:

Delaunay maximizes minimum angles (in lexicographic order)

Proof:

Let T be the triangulation maximizing angles

Delaunay Triangulation: max-min angle

Theorem:

Delaunay maximizes minimum angles (in lexicographic order)

Proof:

Let T be the triangulation maximizing angles

$\Longrightarrow \forall$ convex quadrilateral (from 2 triangles $\in T$)

the diagonal maximizes smallest angle (in quad)

Delaunay Triangulation: max-min angle

Theorem:

Delaunay maximizes minimum angles (in lexicographic order)

Proof:

Let T be the triangulation maximizing angles

$\Longrightarrow \forall$ convex quadrilateral (from 2 triangles $\in T$)

the diagonal maximizes smallest angle (in quad)

$\Longrightarrow \forall$ edge, it is locally Delaunay

Delaunay Triangulation: max-min angle

Theorem:

Delaunay maximizes minimum angles (in lexicographic order)

Proof:

Let T be the triangulation maximizing angles

$\Longrightarrow \forall$ convex quadrilateral (from 2 triangles $\in T$)

the diagonal maximizes smallest angle (in quad)

$\Longrightarrow \forall$ edge, it is locally Delaunay

$\Longrightarrow T =$ Delaunay

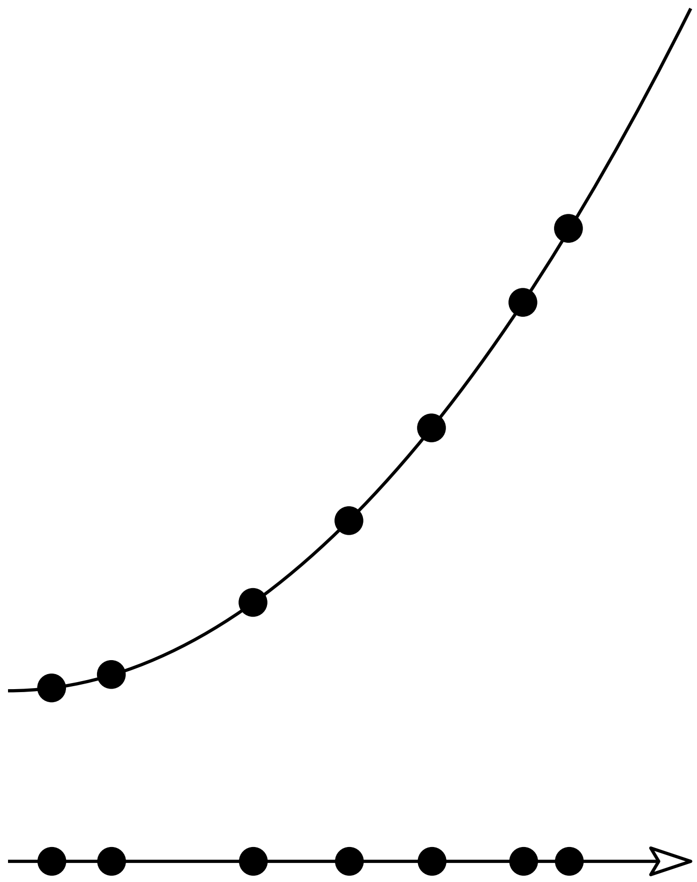
Delaunay triangulation

A stupid algorithm for sorting numbers

Delaunay triangulation

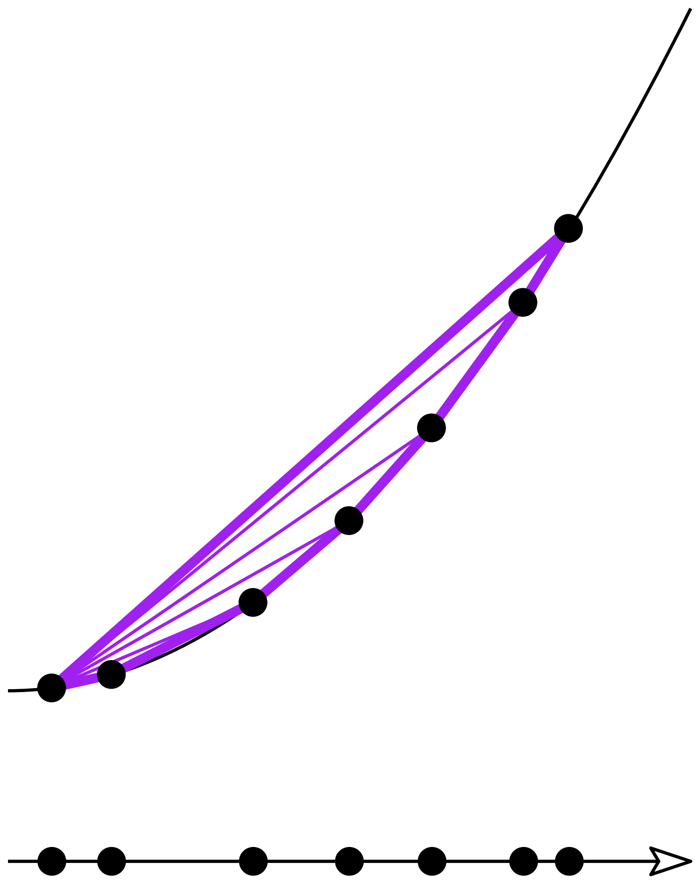
A stupid algorithm for sorting numbers

project on parabola



Delaunay triangulation

A stupid algorithm for sorting numbers

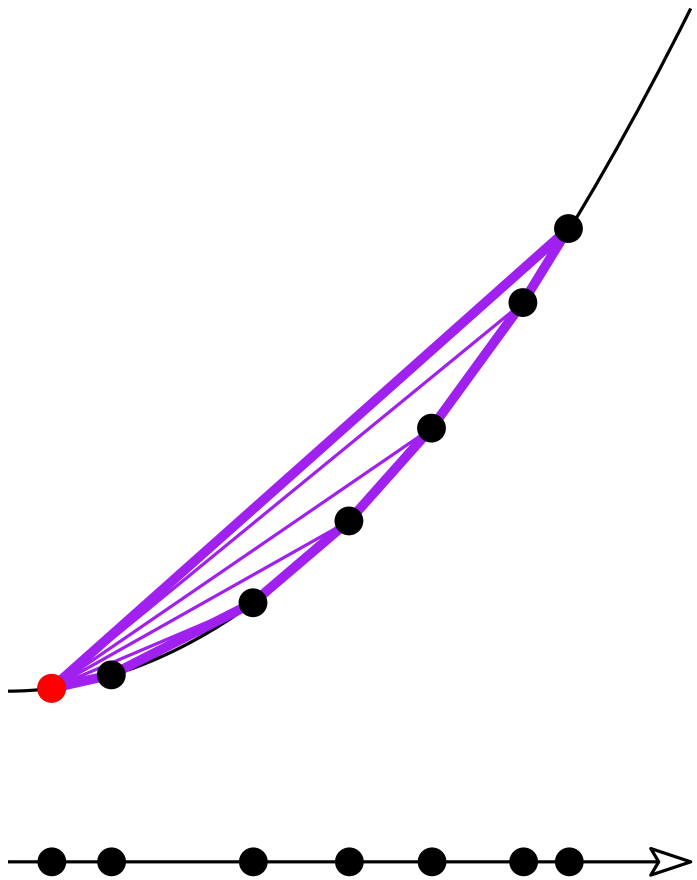


project on parabola

compute Delaunay triang.

Delaunay triangulation

A stupid algorithm for sorting numbers



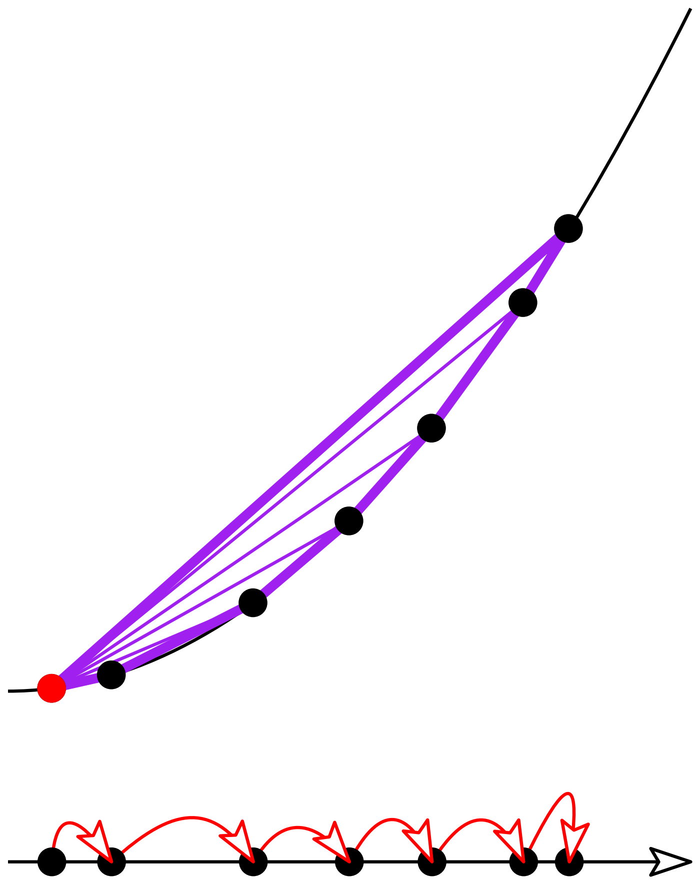
project on parabola

compute Delaunay triang.

find lowest point

Delaunay triangulation

A stupid algorithm for sorting numbers



project on parabola

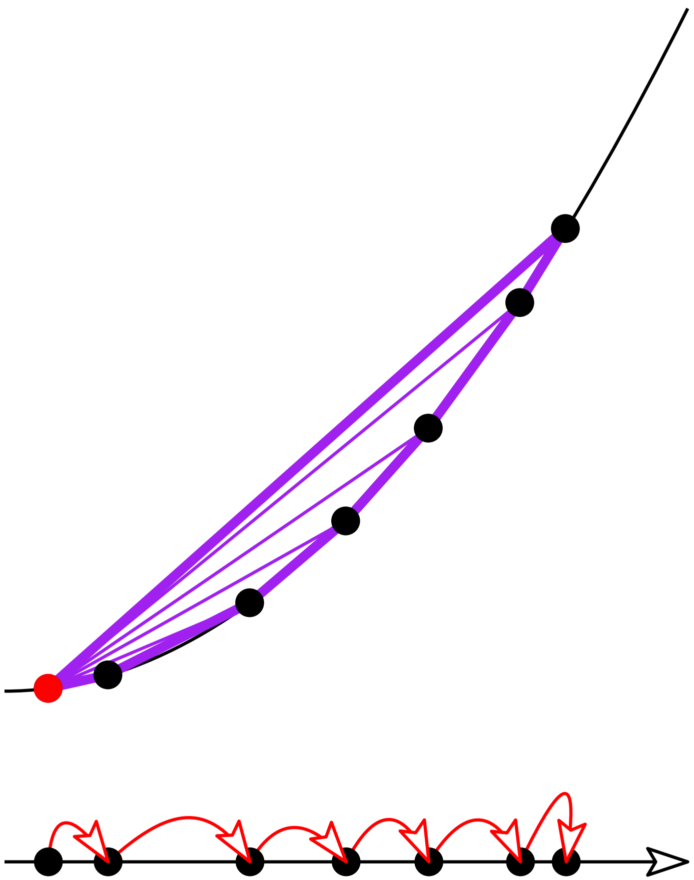
compute Delaunay triang.

find lowest point

enumerate x coordinates
in ccw CH order

Delaunay triangulation

A stupid algorithm for sorting numbers



$O(n)$ project on parabola

$f(n)$ compute Delaunay triang.

$O(n)$ find lowest point

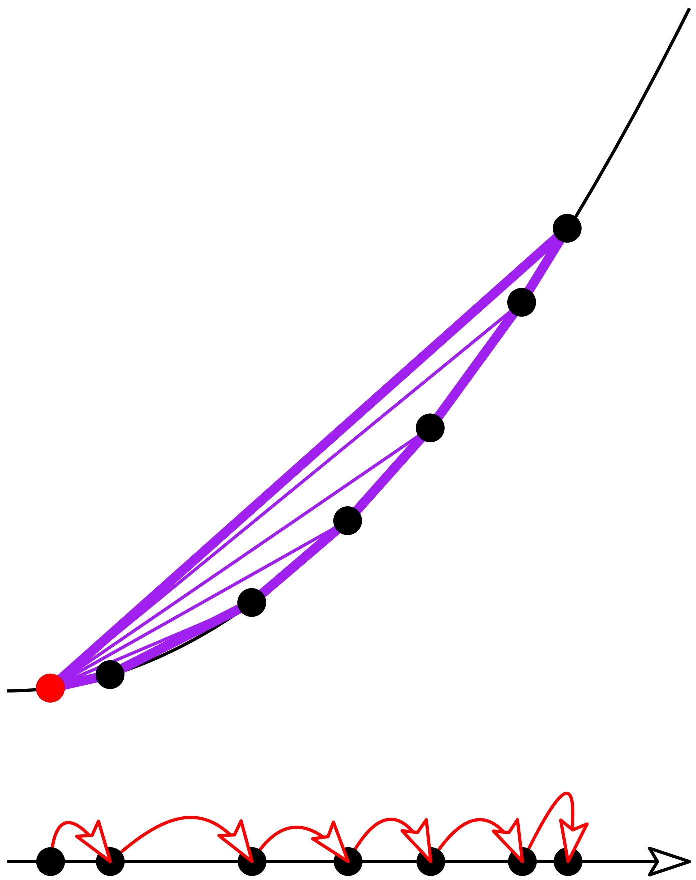
$O(n)$ enumerate x coordinates in ccw CH order

Lower bound on sorting

$\implies f(n) + O(n) \geq \Omega(n \log n)$

Delaunay triangulation

A stupid algorithm for sorting numbers



$O(n)$ project on parabola

$f(n)$ compute Delaunay triang.

$O(n)$ find lowest point

$O(n)$ enumerate x coordinates in ccw CH order

Lower bound on sorting

$\Longrightarrow f(n) + O(n) \geq \Omega(n \log n)$

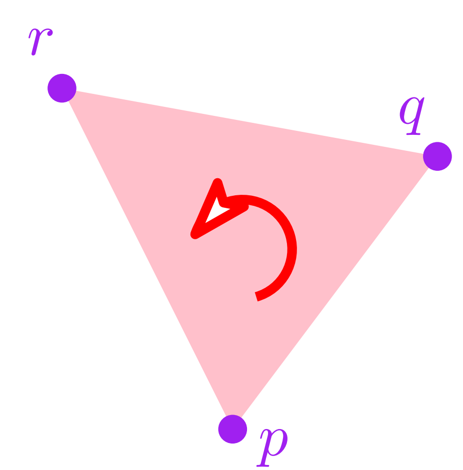
Delaunay Triangulation: predicates

Delaunay Triangulation: predicates

Orientation predicate

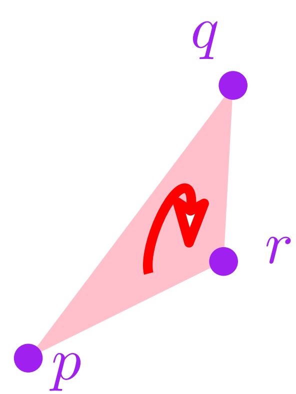
pqr + ?

$$\begin{vmatrix} x_q - x_p & x_r - x_p \\ y_q - y_p & y_r - y_p \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 \\ x_p & x_q & x_r \\ y_p & y_q & y_r \end{vmatrix} > 0$$



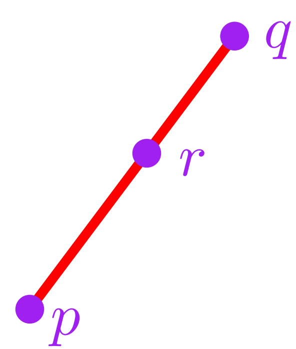
pqr - ?

$$\begin{vmatrix} 1 & 1 & 1 \\ x_p & x_q & x_r \\ y_p & y_q & y_r \end{vmatrix} < 0$$



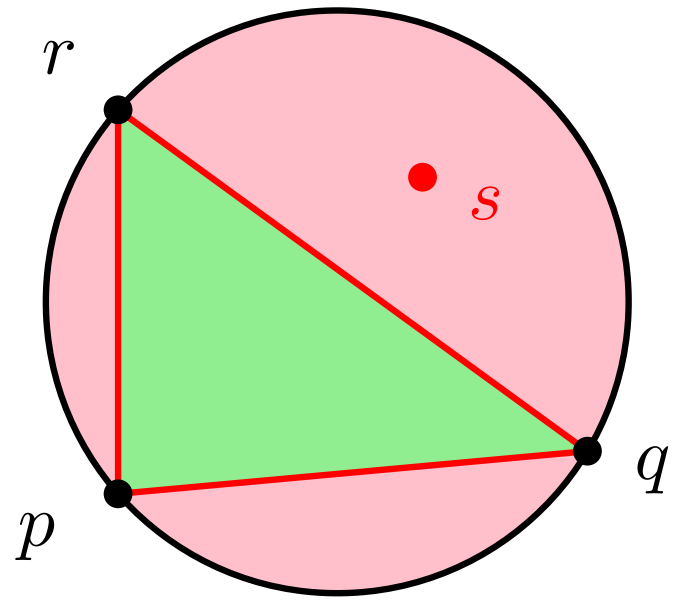
pqr 0 ?

$$\begin{vmatrix} 1 & 1 & 1 \\ x_p & x_q & x_r \\ y_p & y_q & y_r \end{vmatrix} = 0$$



Delaunay Triangulation: predicates

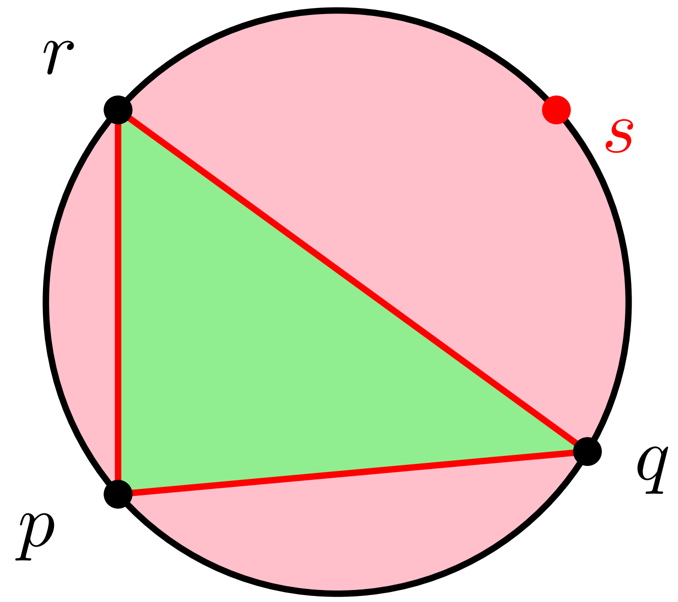
Incircle predicate



pqr ccw triangle

query s inside circumcircle

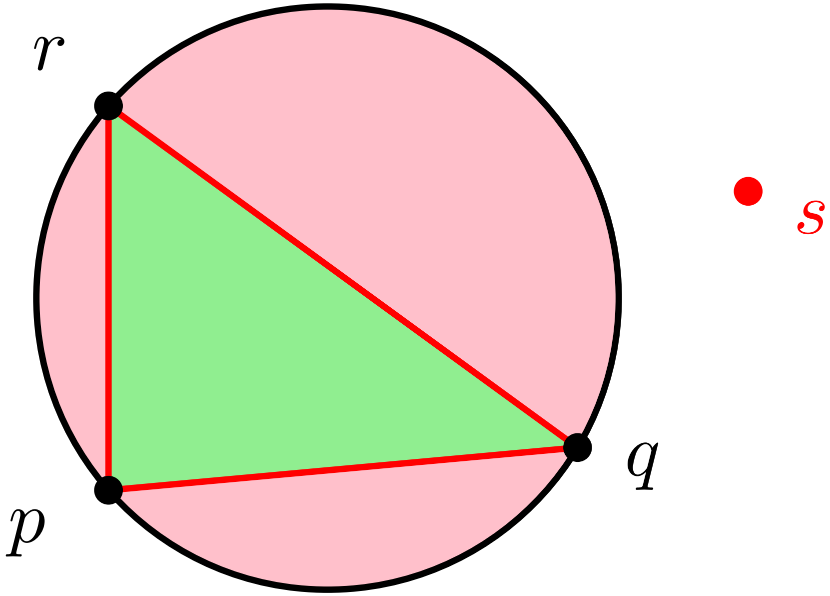
Delaunay Triangulation: predicates



pqr ccw triangle

query s cocircular

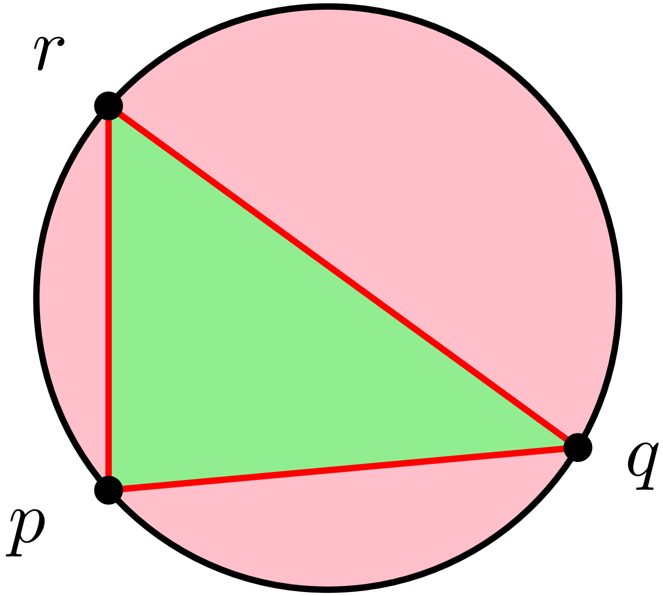
Delaunay Triangulation: predicates



pqr ccw triangle

query s outside circumcircle

Delaunay Triangulation: predicates

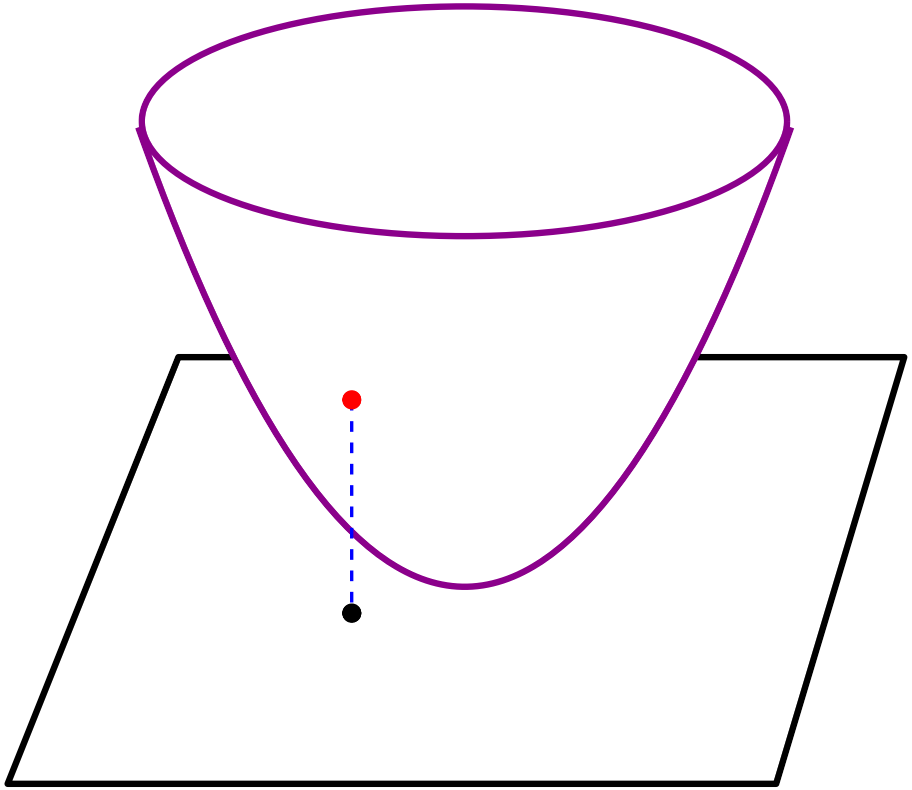


pqr ccw triangle

query s

Delaunay Triangulation: predicates

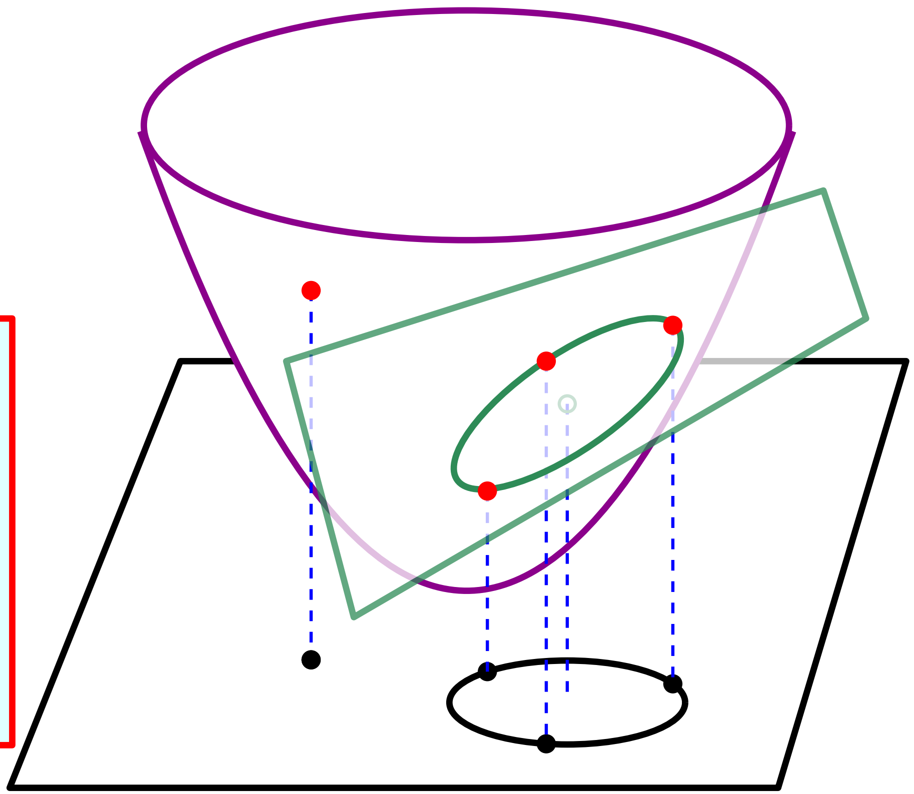
Space of circles



$$p = (x, y) \rightsquigarrow p^\star = (x, y, x^2 + y^2)$$

Delaunay Triangulation: predicates

Space of circles



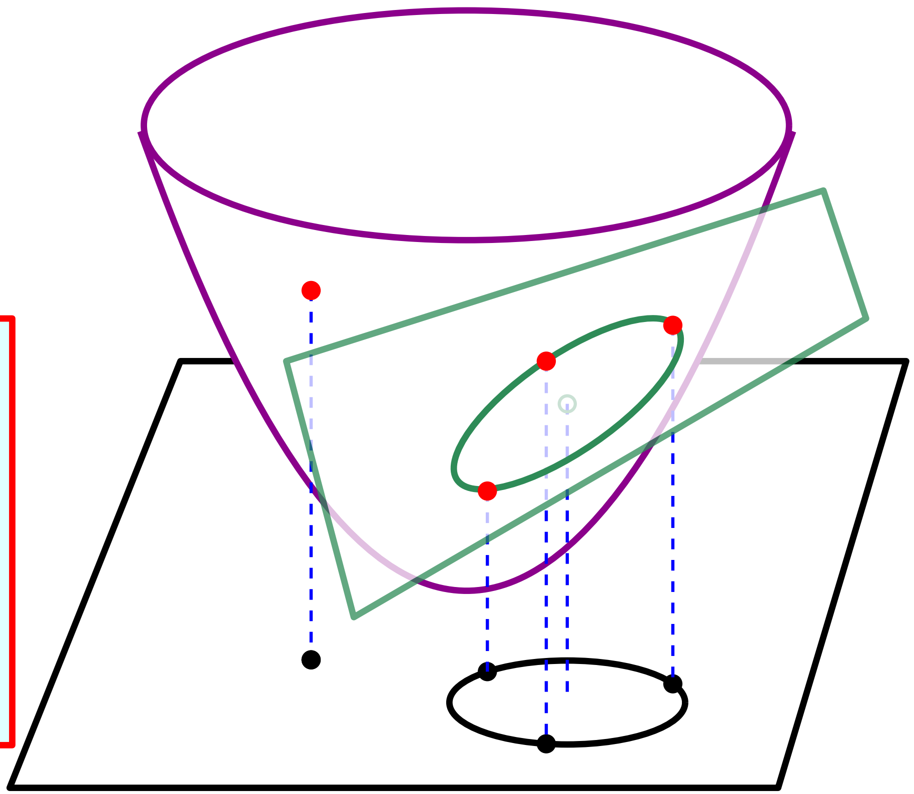
s inside/outside of
circle through pqr
$\rightsquigarrow$ plane through $p^\star q^\star r^\star$
above/below $s^\star$

incircle predicate

$\rightsquigarrow$ 3D orientation predicate

Delaunay Triangulation: predicates

Space of circles



s inside/outside of circle through pqr

$\rightsquigarrow$ plane through $p^\star q^\star r^\star$ above/below $s^\star$

incircle predicate

$\rightsquigarrow$ 3D orientation predicate

$$\text{sign}\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_p & x_q & x_r & x_s \\ y_p & y_q & y_r & y_s \\ x_p^2+y_p^2 & x_q^2+y_q^2 & x_r^2+y_r^2 & x_s^2+y_s^2 \end{vmatrix}$$

Delaunay Triangulation: data structure

Data structure for (Delaunay) triangulation ?

Representing incidences

Representing hull boundary

Representing user's data put colors in triangles

...

Delaunay Triangulation: very naive algorithm

Delaunay Triangulation: very naive algorithm

For each triple of points (p, q, r)

    If pqr ccw

        Delaunay = true;

        For each point s

            If s in circle pqr

                Delaunay = false;

        Output pqr

Delaunay Triangulation: very naive algorithm

For each triple of points (p, q, r)

If pqr ccw

Delaunay = true; Correctness: easy

For each point s

If s in circle pqr

Delaunay = false;

Output pqr

Delaunay Triangulation: very naive algorithm

For each triple of points (p, q, r)

If pqr ccw

Delaunay = true;

For each point s

If s in circle pqr

Delaunay = false;

Output pqr

Correctness: easy

Complexity: $O(n^4)$

Delaunay Triangulation: very naive algorithm

For each triple of points (p, q, r)

If pqr ccw

Delaunay = true;

For each point s

If s in circle pqr

Delaunay = false;

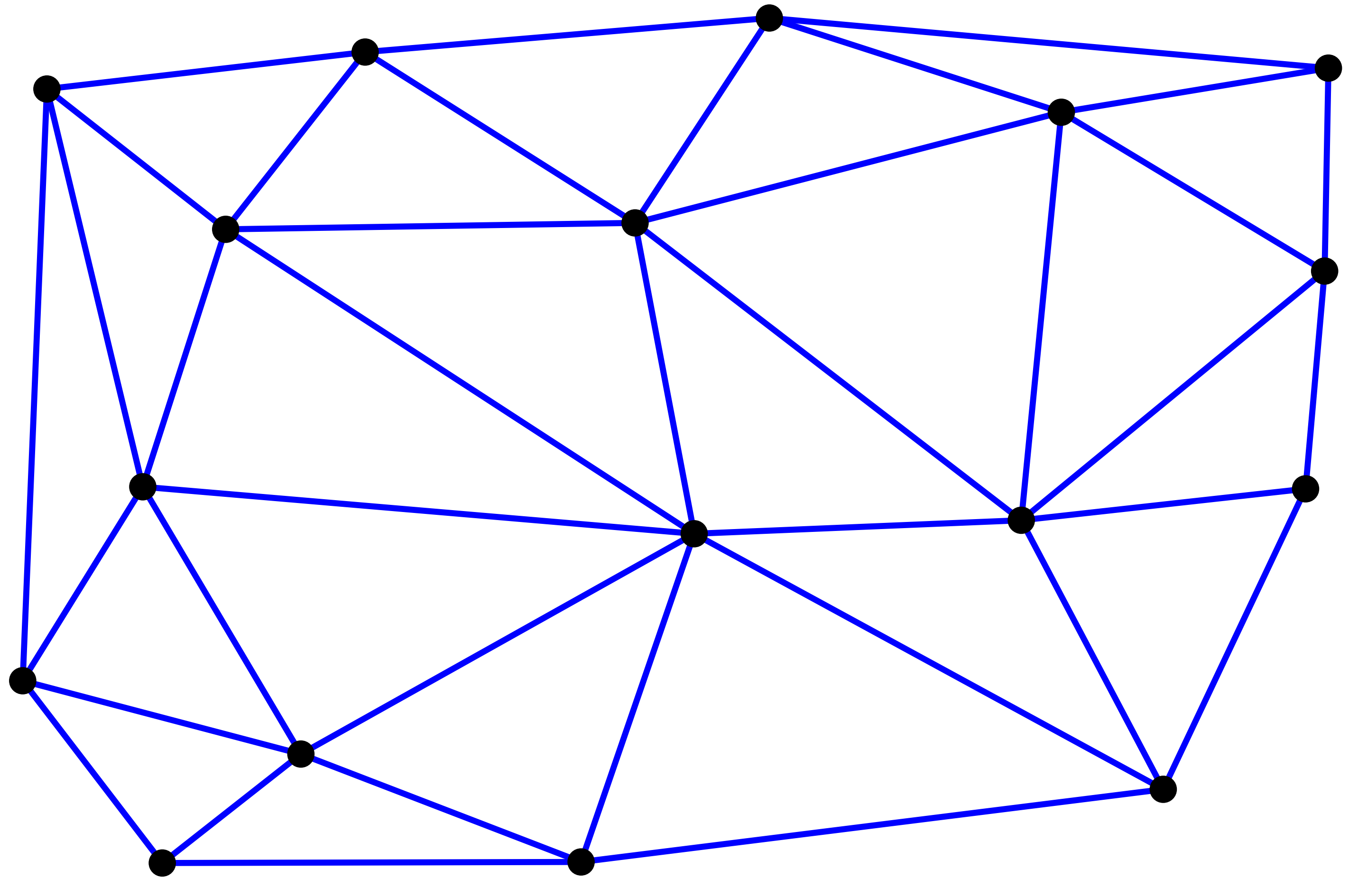
Output pqr

Correctness: easy

Complexity: $O(n^4)$

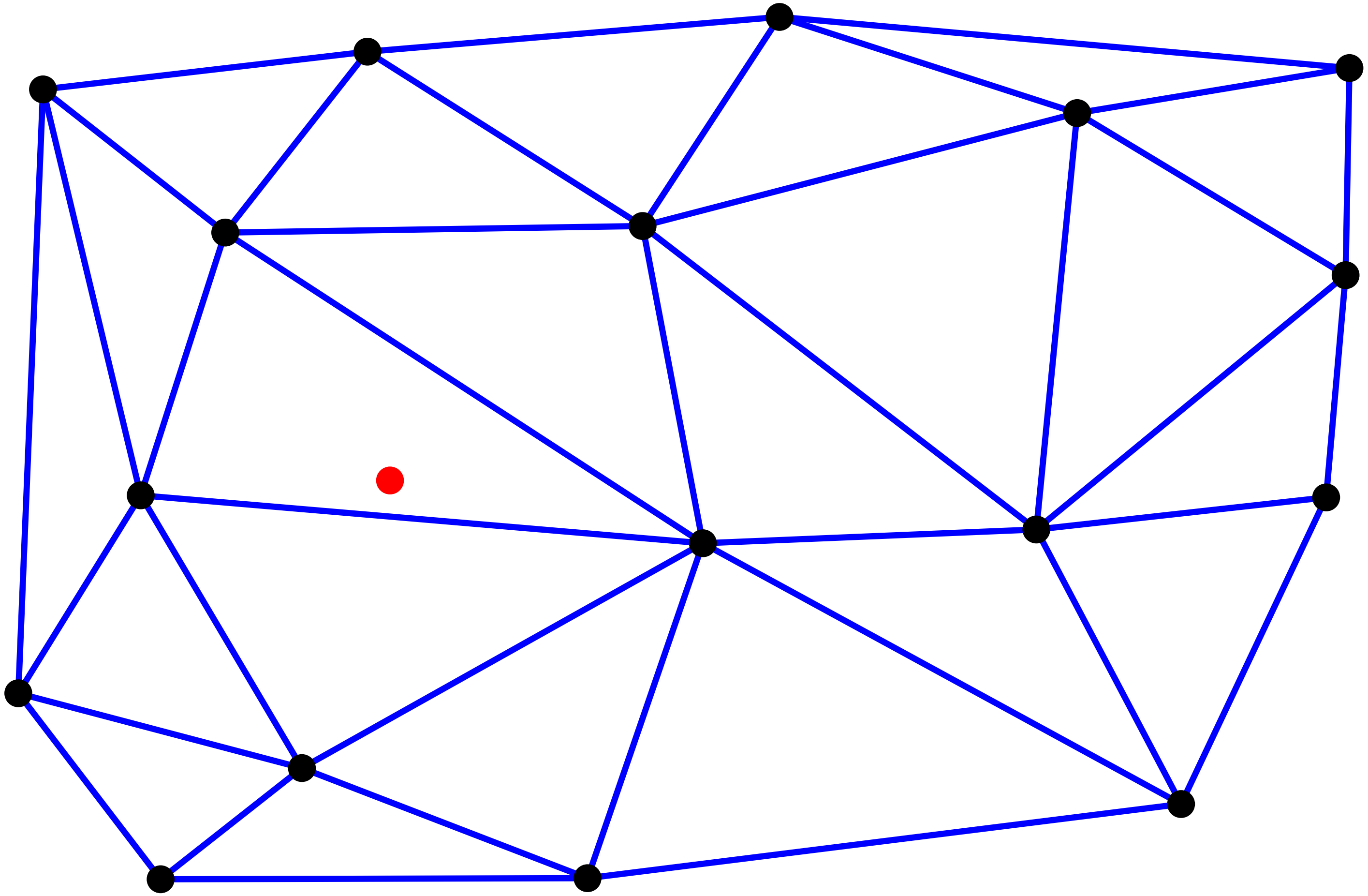
does not compute incidences

Delaunay Triangulation: incremental algorithm



Delaunay Triangulation: incremental algorithm

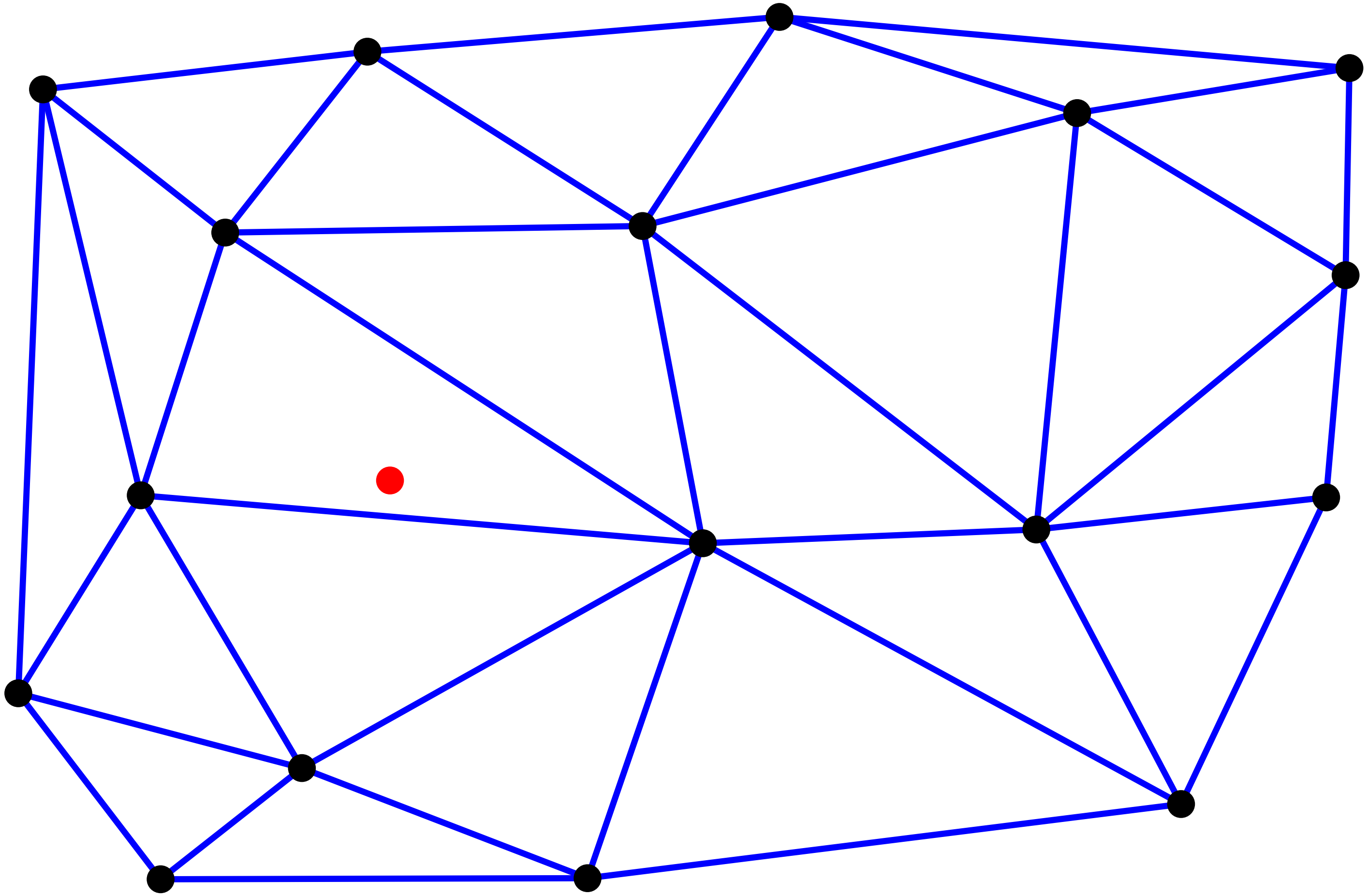
New point



Delaunay Triangulation: incremental algorithm

New point

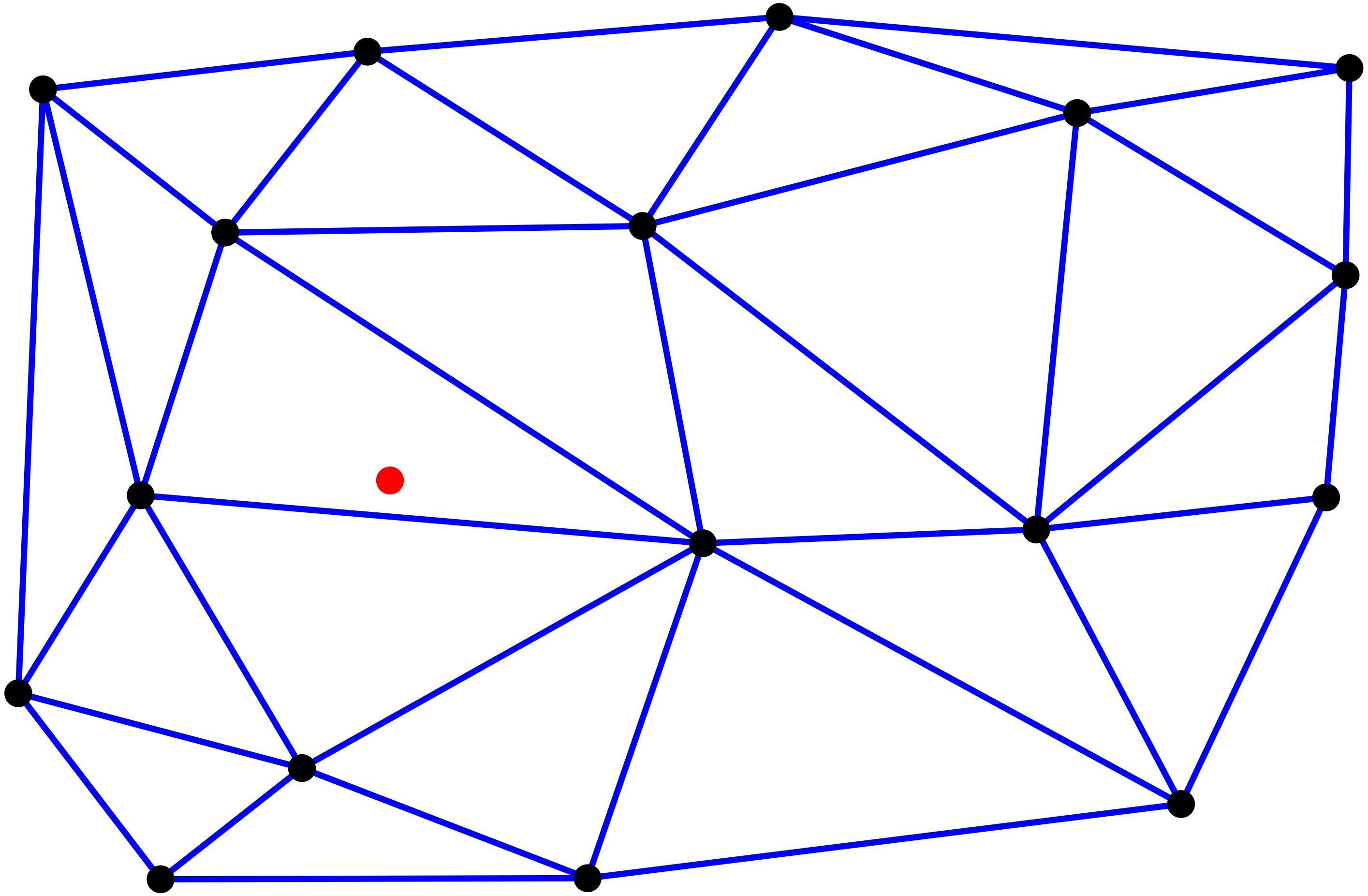
Locate



Delaunay Triangulation: incremental algorithm

New point

Locate

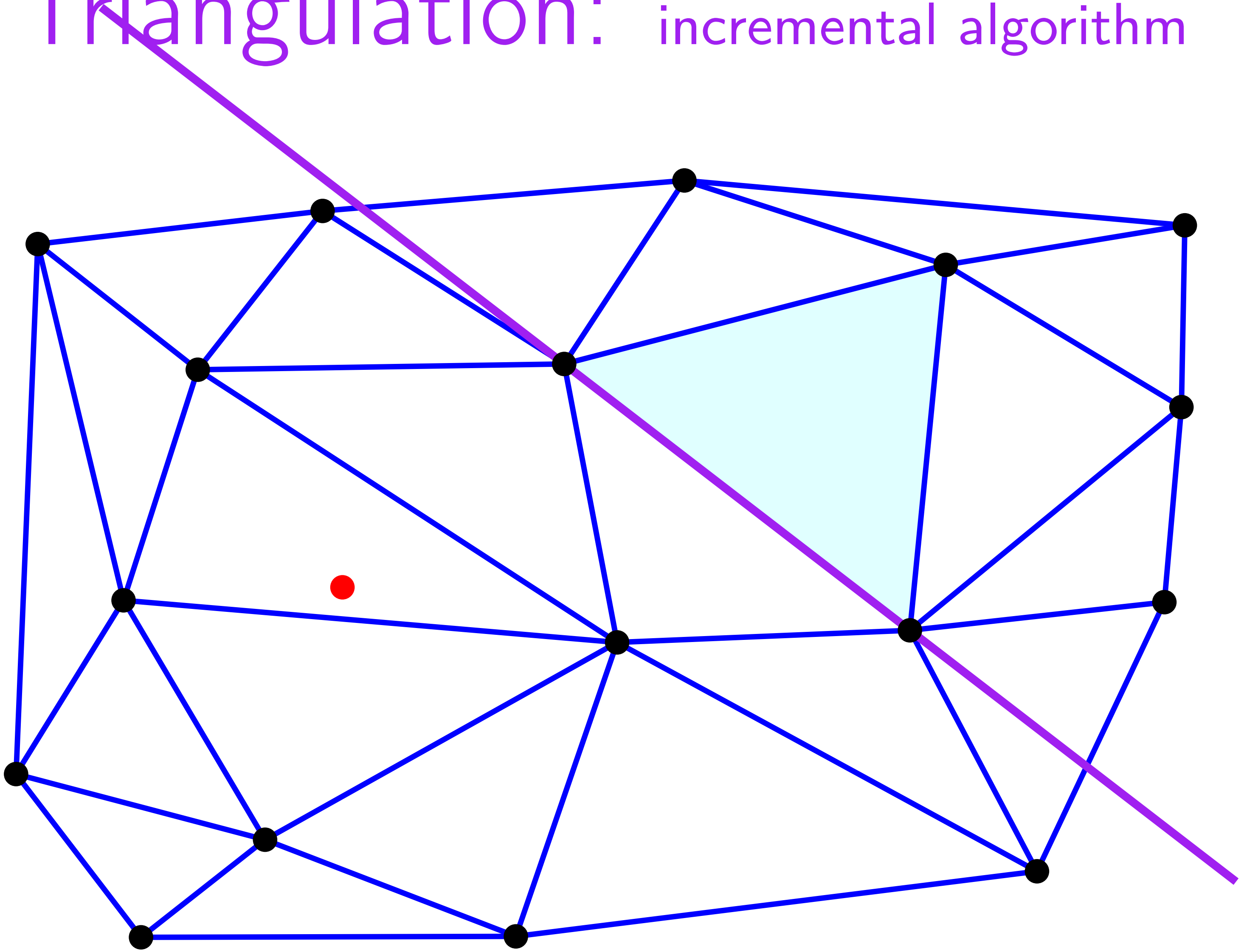


e.g.: visibility walk

Delaunay Triangulation: incremental algorithm

New point

Locate

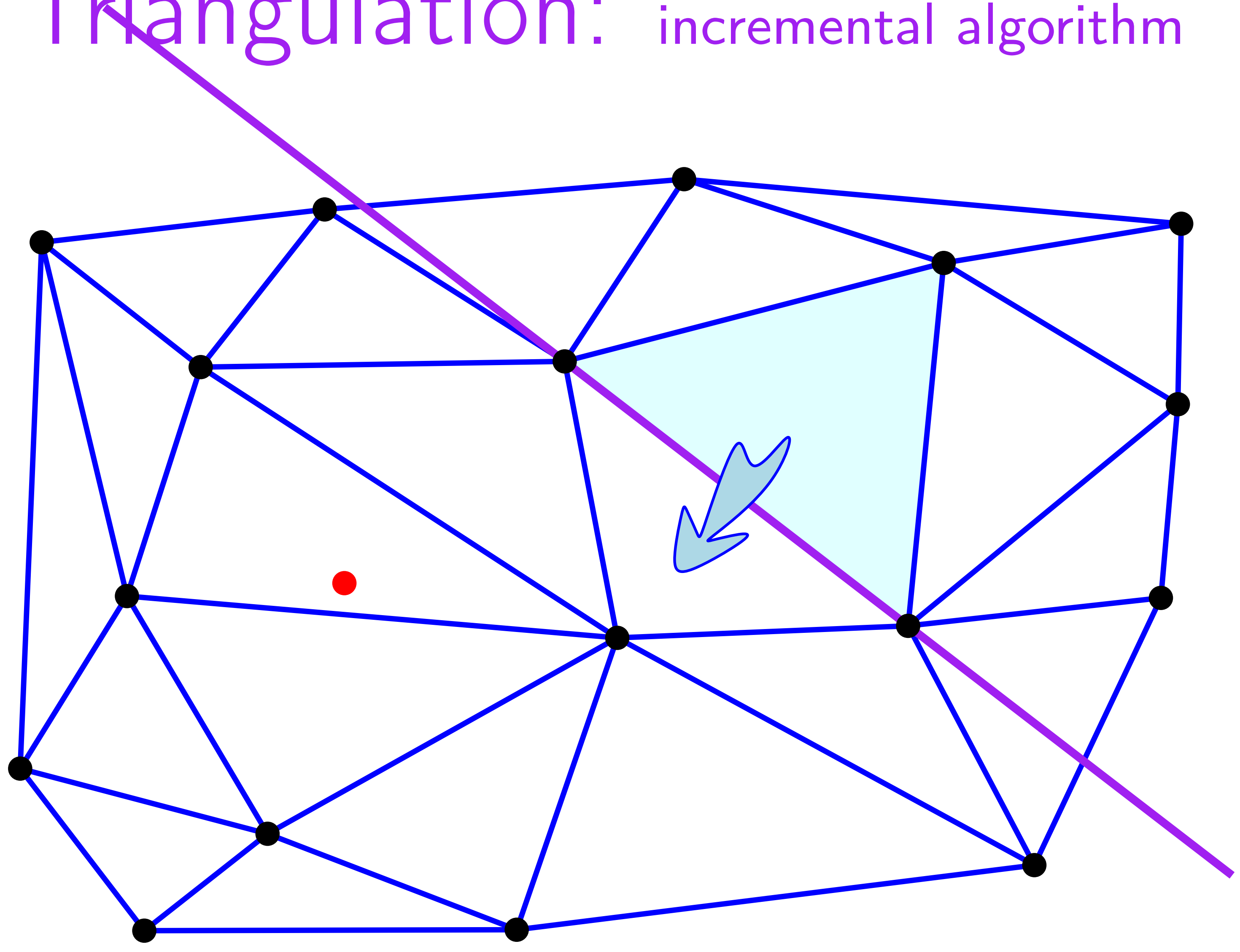


e.g.: visibility walk

Delaunay Triangulation: incremental algorithm

New point

Locate

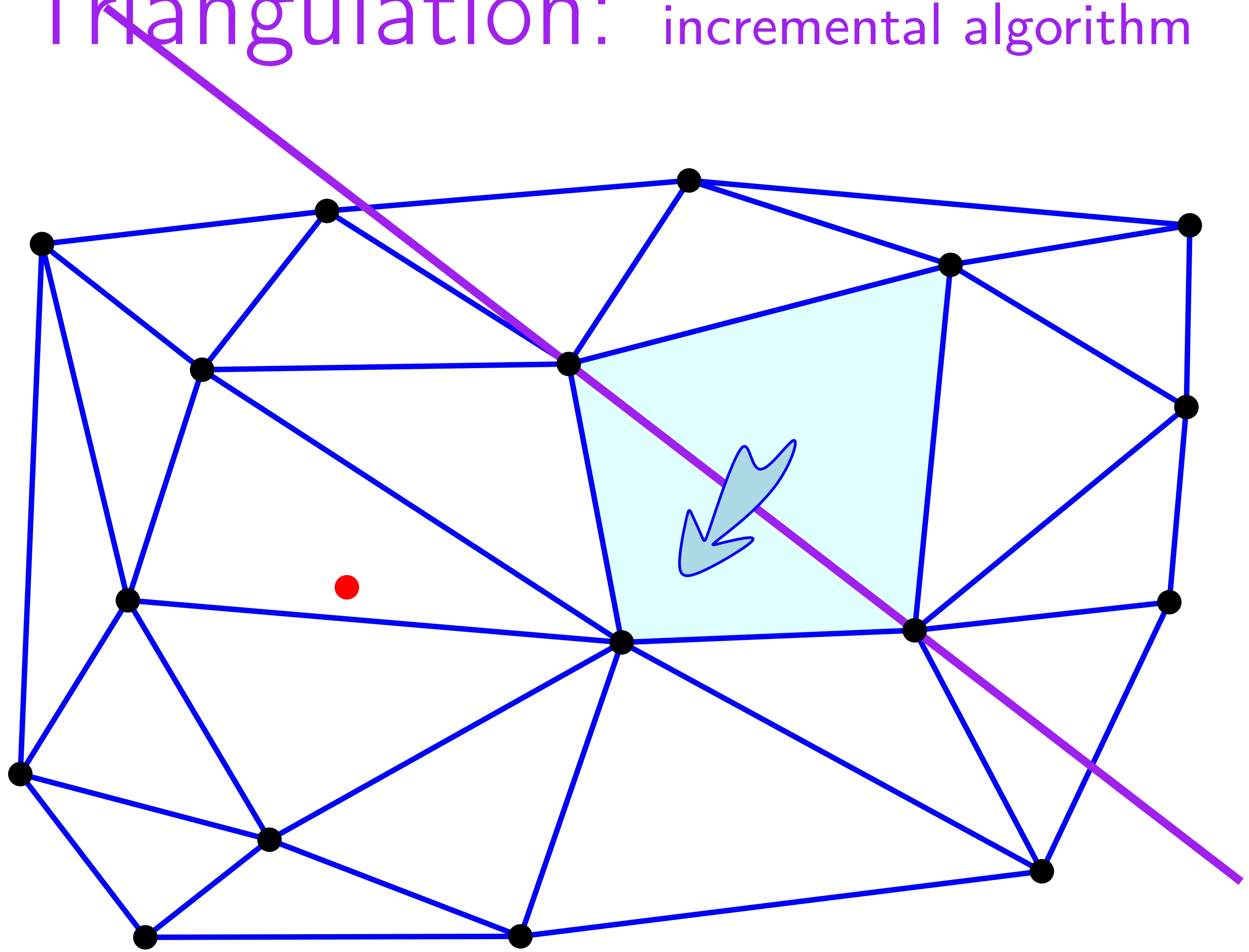


e.g.: visibility walk

Delaunay Triangulation: incremental algorithm

New point

Locate

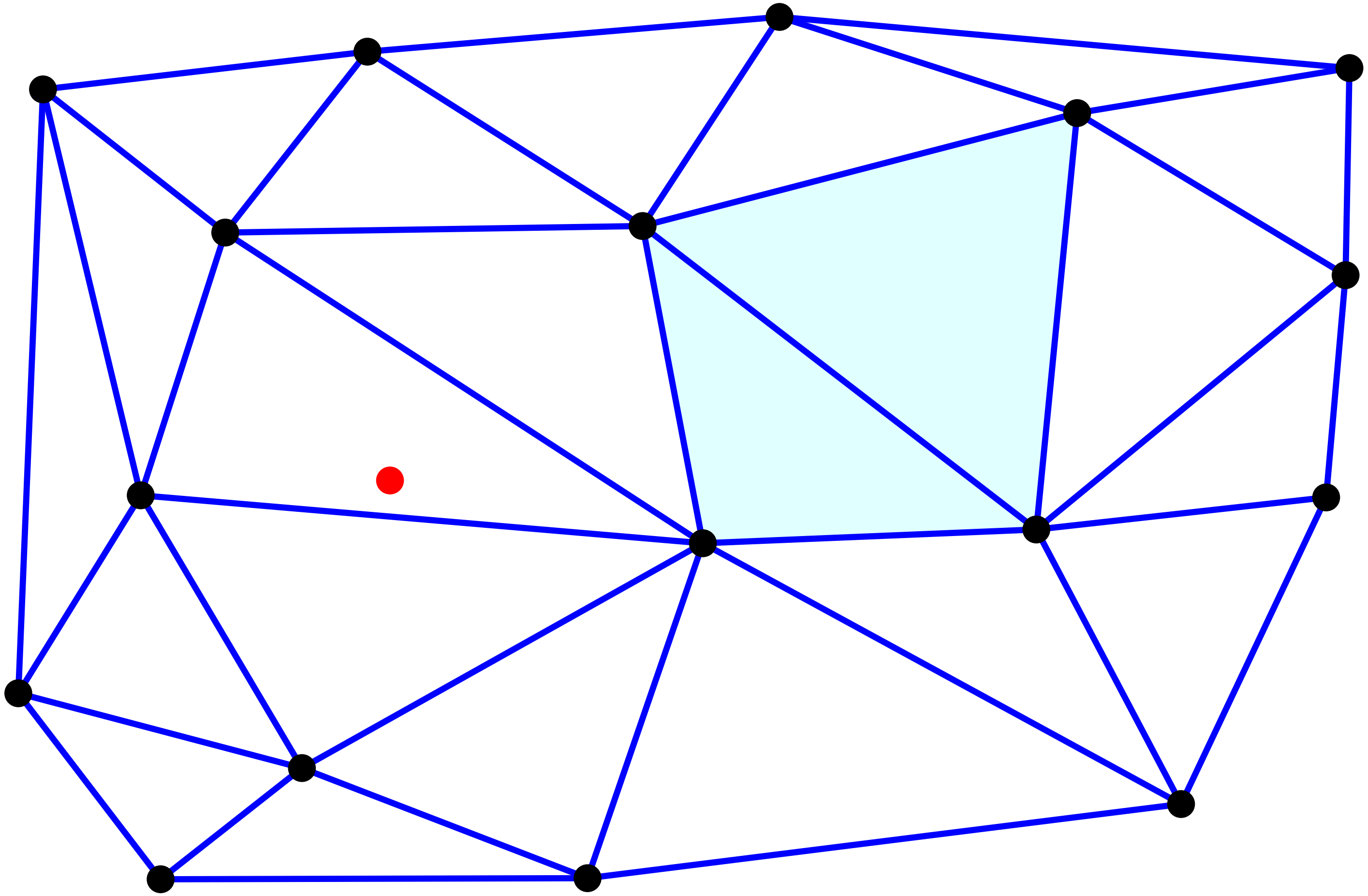


e.g.: visibility walk

Delaunay Triangulation: incremental algorithm

New point

Locate

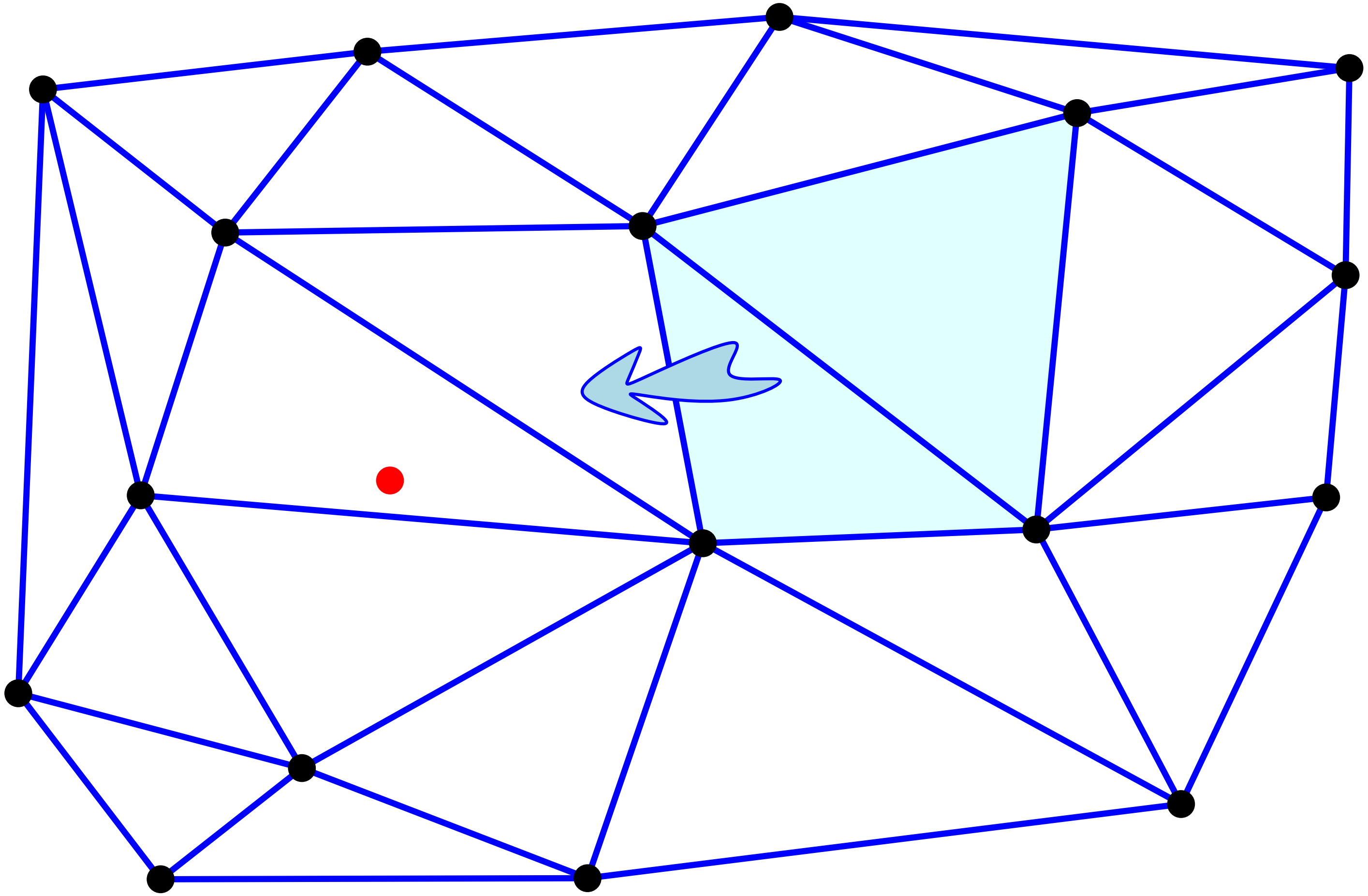


e.g.: visibility walk

Delaunay Triangulation: incremental algorithm

New point

Locate

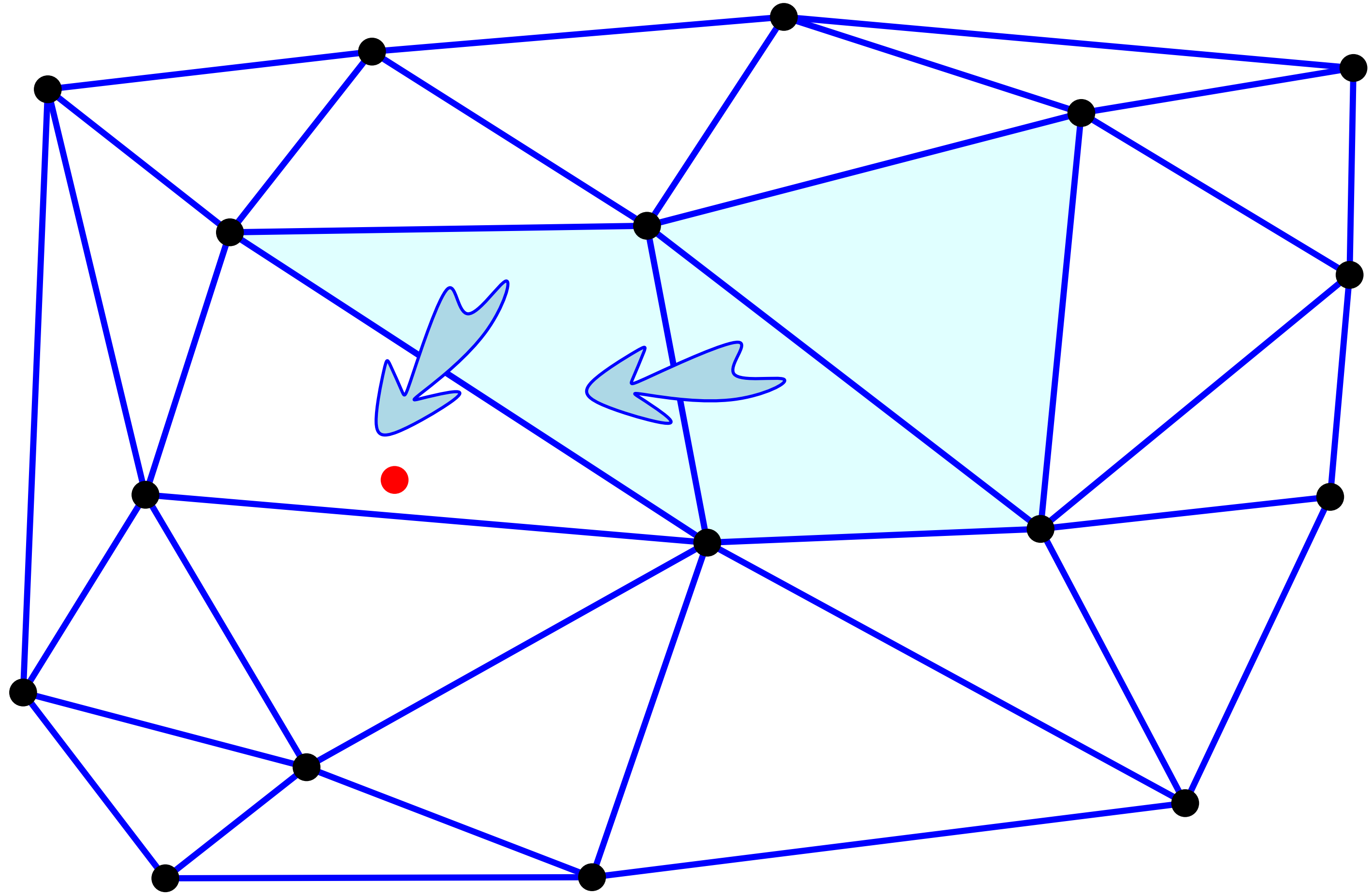


e.g.: visibility walk

Delaunay Triangulation: incremental algorithm

New point

Locate

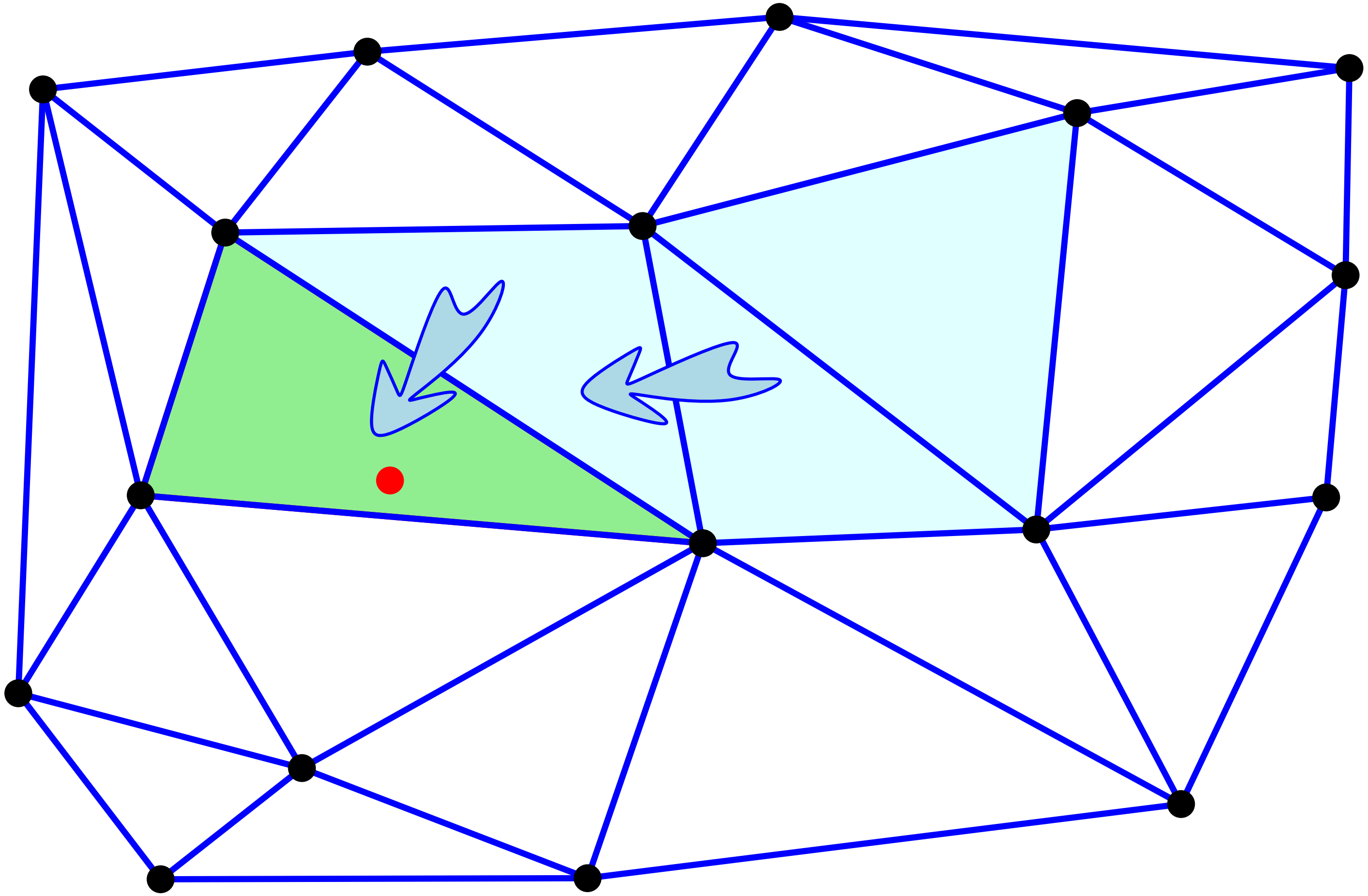


e.g.: visibility walk

Delaunay Triangulation: incremental algorithm

New point

Locate

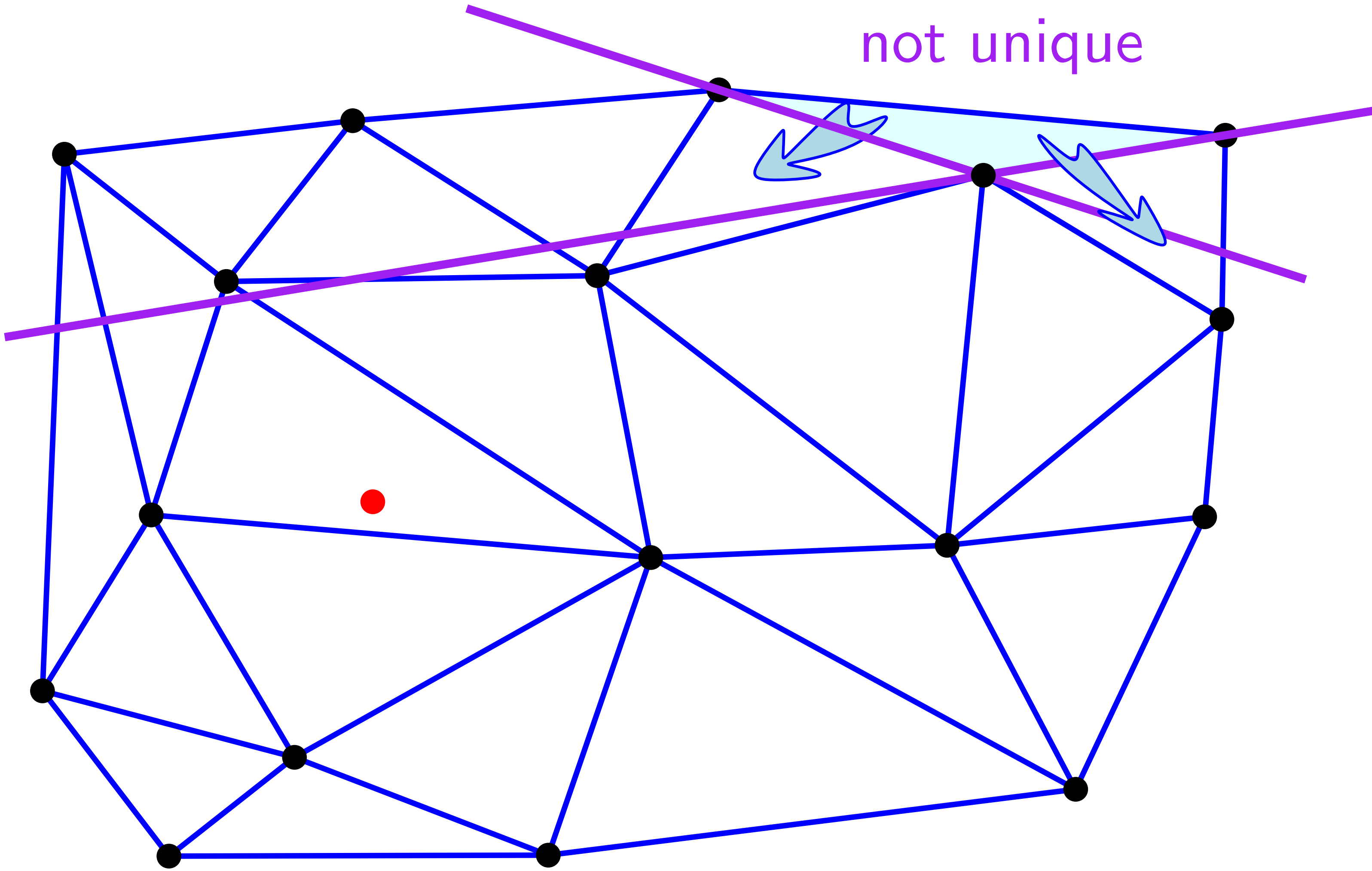


e.g.: visibility walk

Delaunay Triangulation: incremental algorithm

New point

Locate

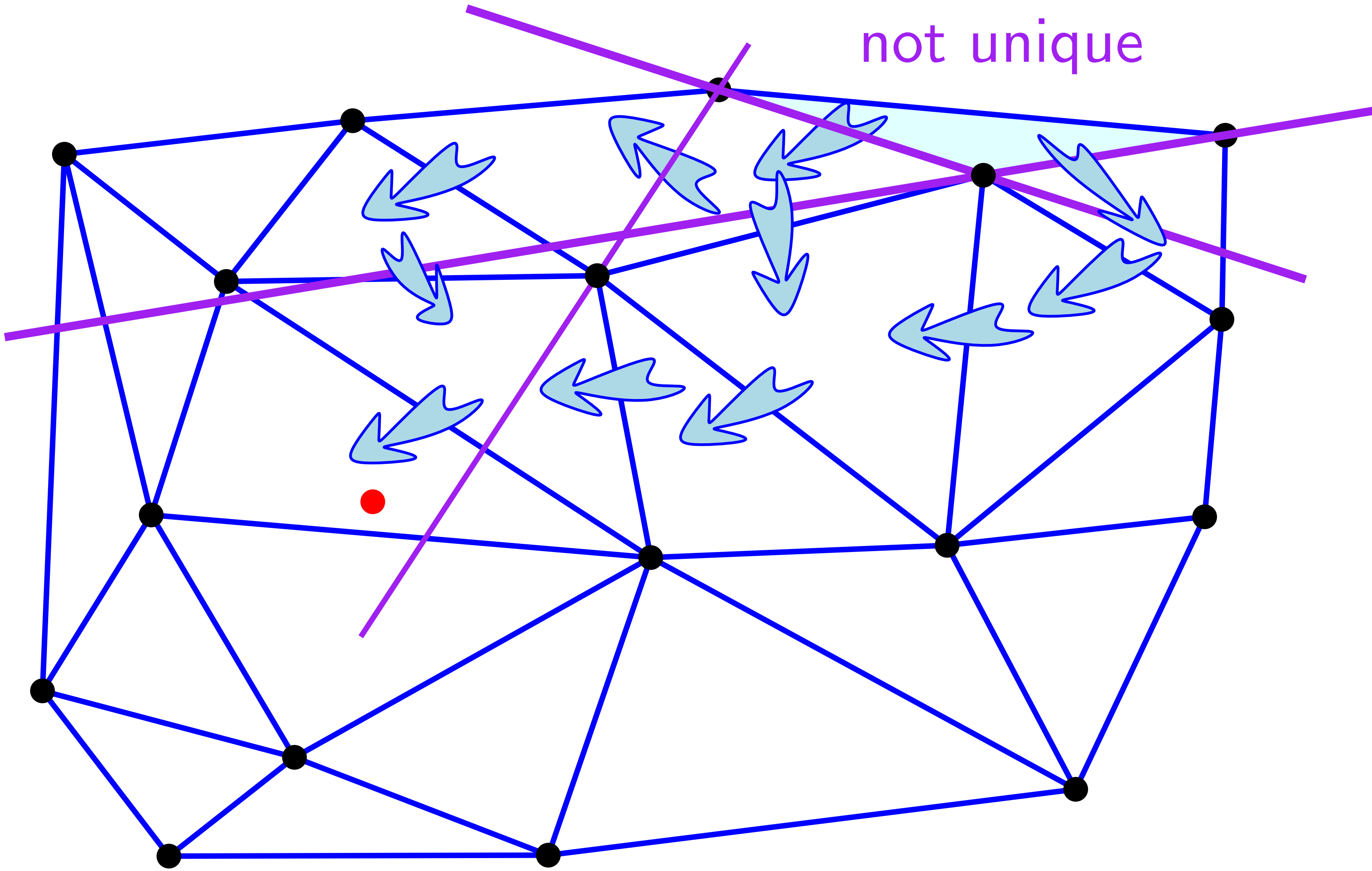


e.g.: visibility walk

Delaunay Triangulation: incremental algorithm

New point

Locate



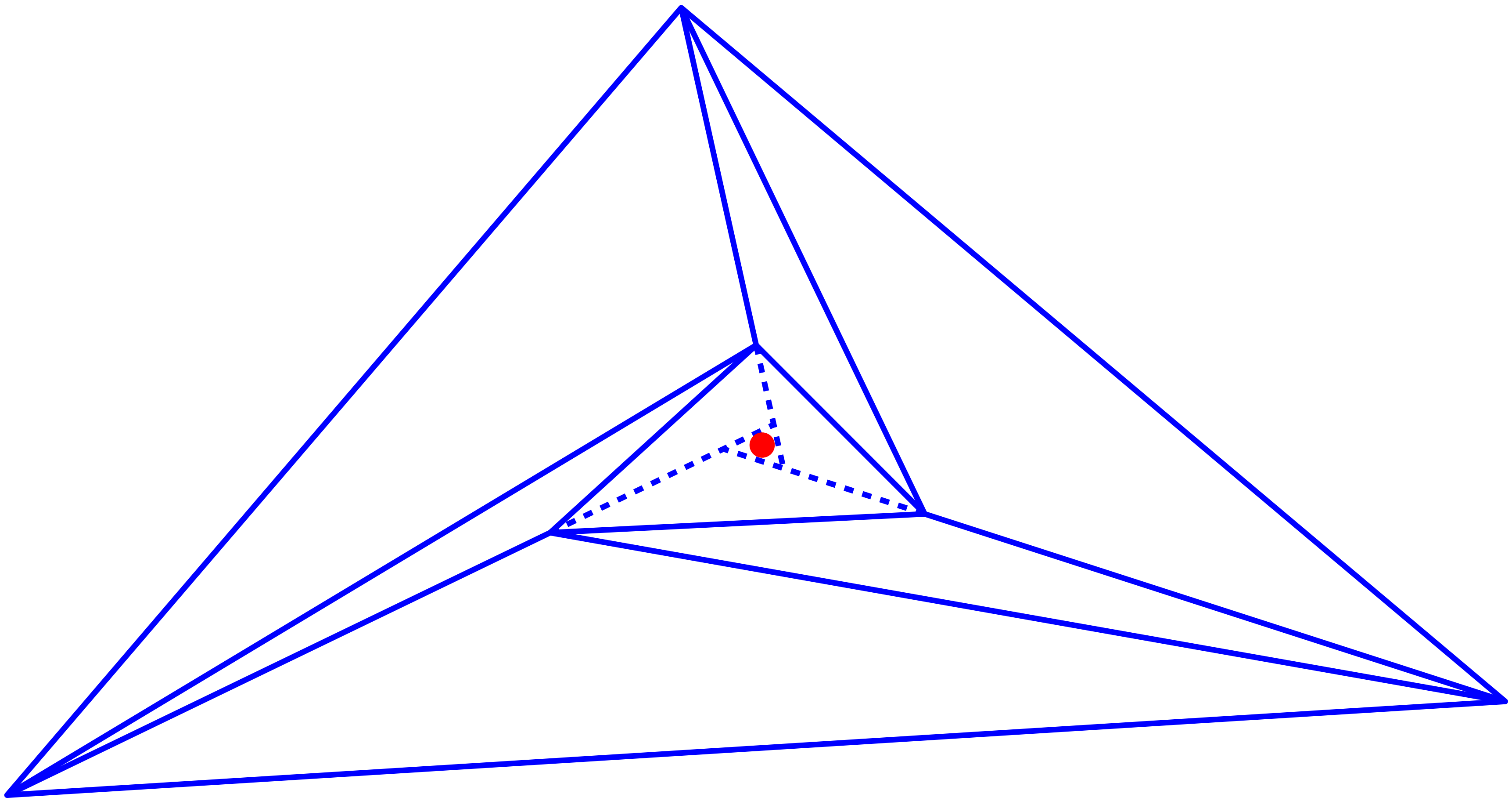
e.g.: visibility walk

Delaunay Triangulation: incremental algorithm

Visibility walk terminates ?

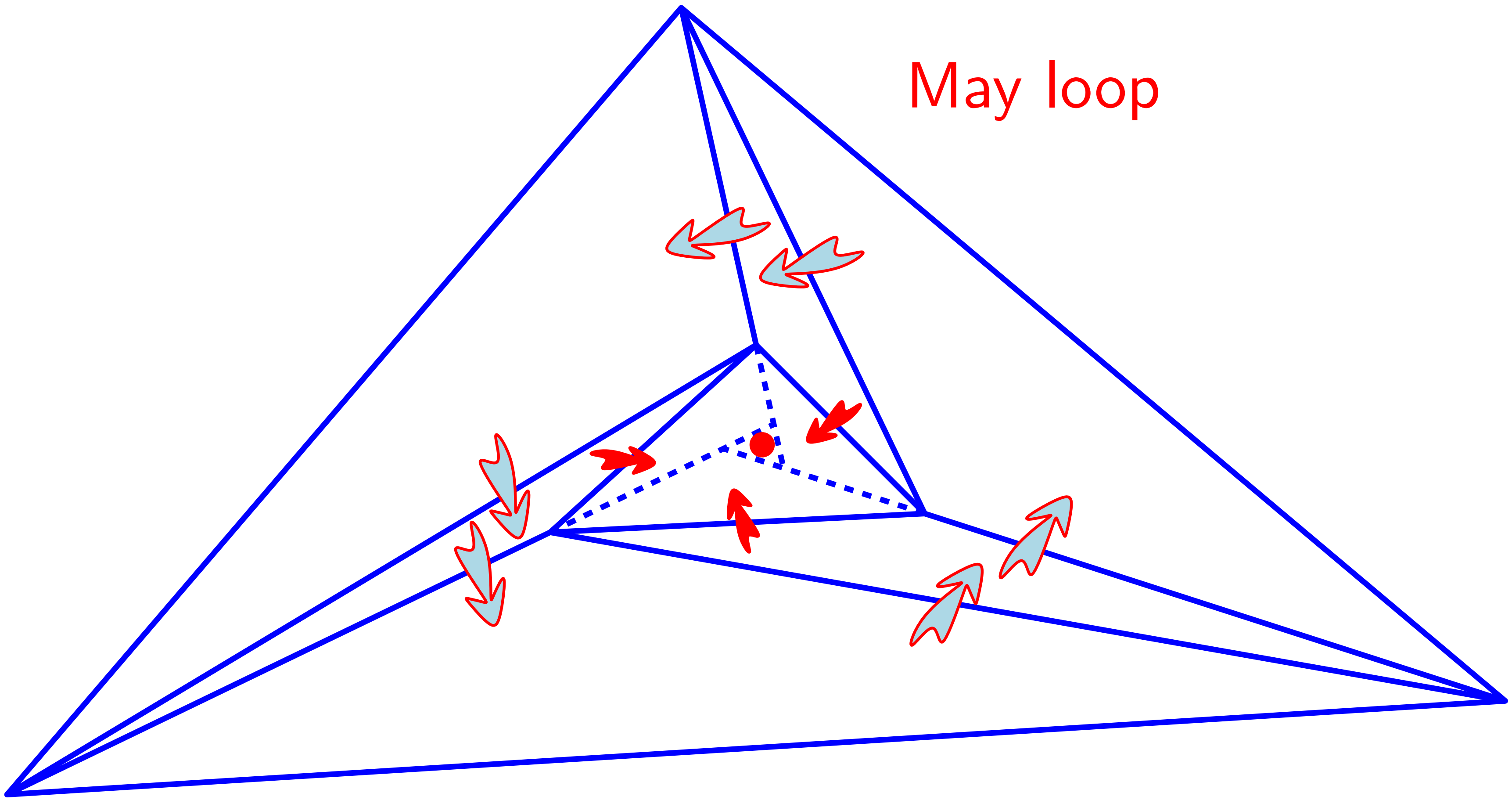
Delaunay Triangulation: incremental algorithm

Visibility walk terminates ?



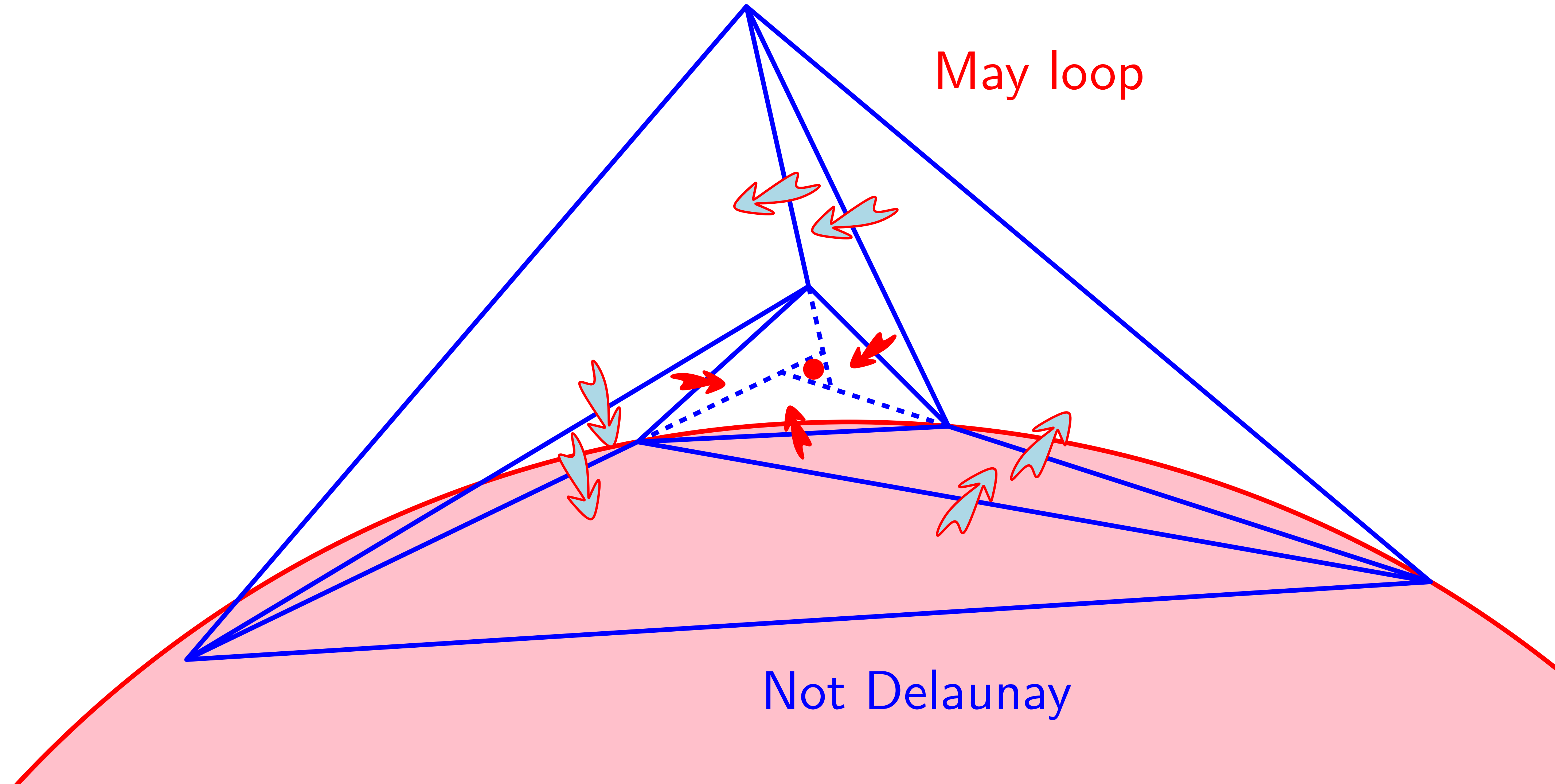
Delaunay Triangulation: incremental algorithm

Visibility walk terminates ?



Delaunay Triangulation: incremental algorithm

Visibility walk terminates ?



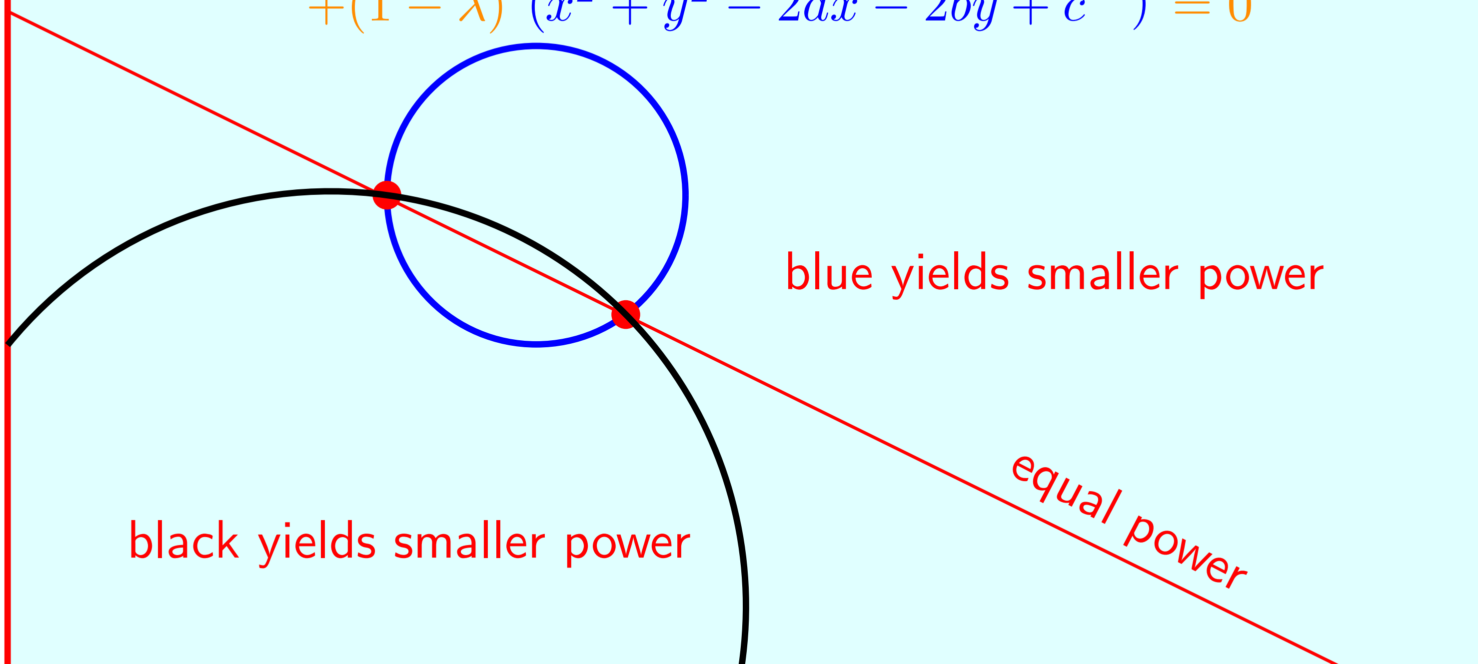
Delaunay Triangulation: incremental algorithm

Visibility walk terminates ?

Delaunay Triangulation: pencils of circles

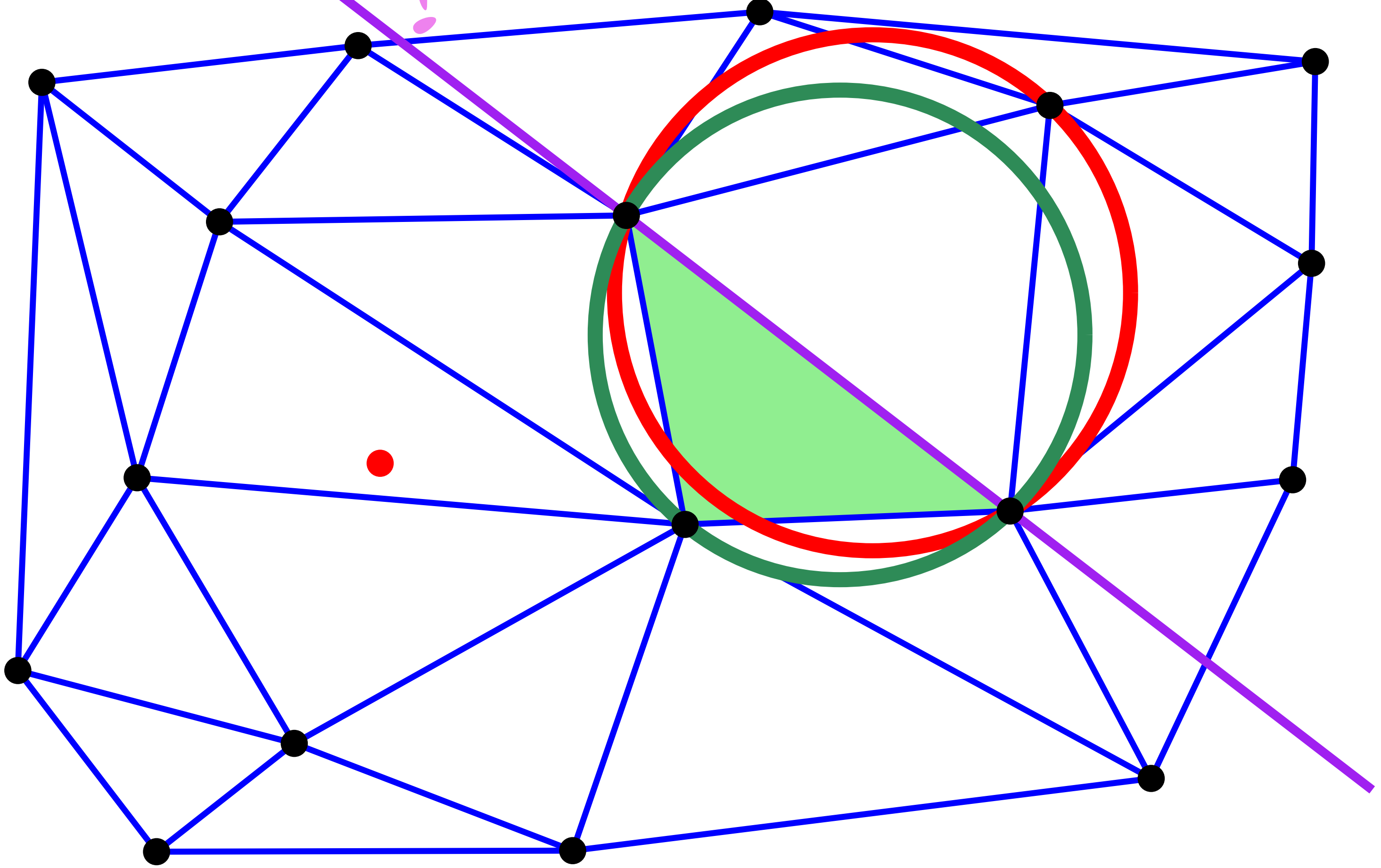
Power of a point w.r.t a circle

$$\lambda \quad (x^2 + y^2 - 2a'x - 2b'y + c')$$
$$+(1-\lambda) \; (x^2 + y^2 - 2ax - 2by + c \quad) = 0$$



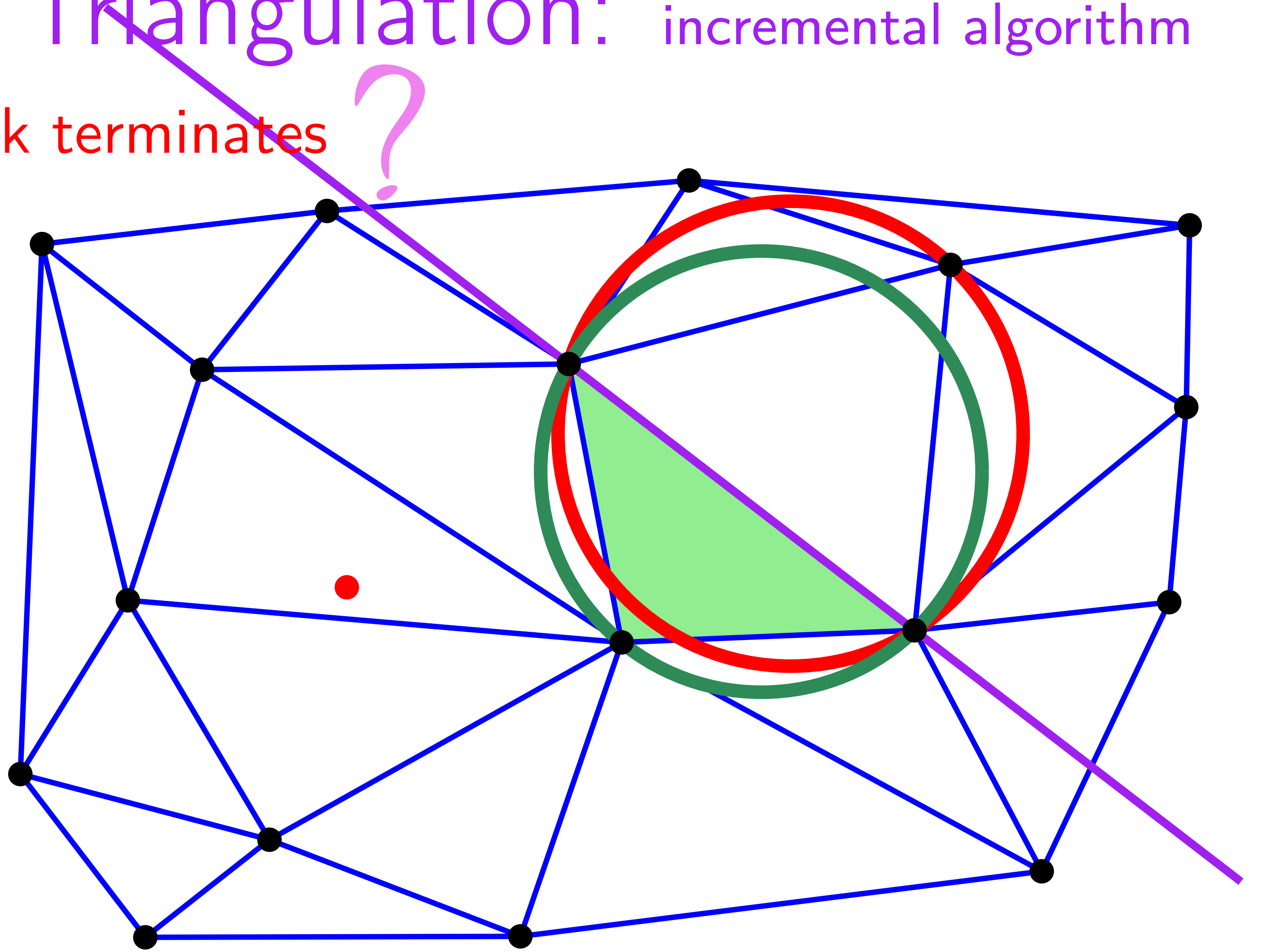
Delaunay Triangulation: incremental algorithm

Visibility walk terminates ?



Delaunay Triangulation: incremental algorithm

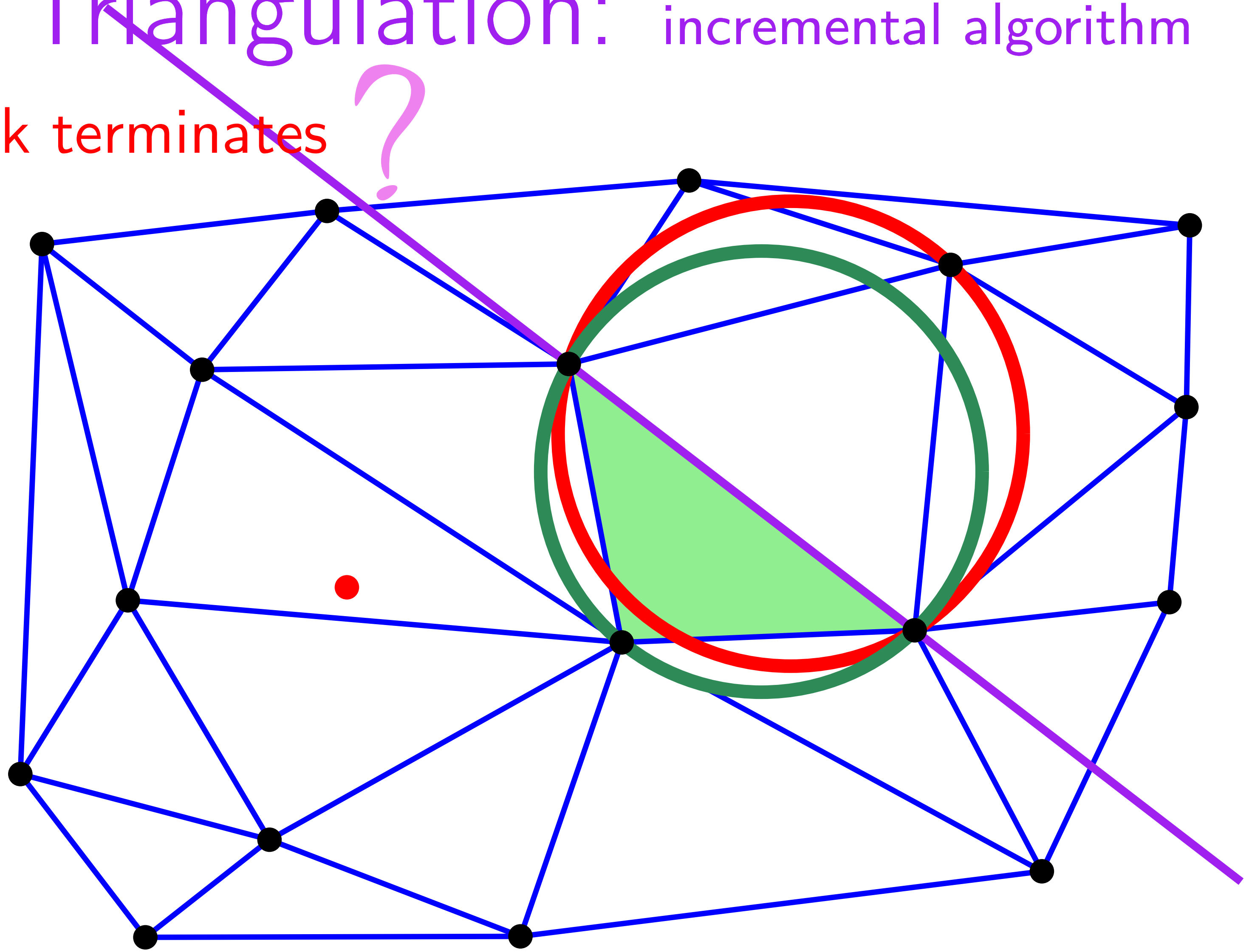
Visibility walk terminates ?



Green power < Red power

Delaunay Triangulation: incremental algorithm

Visibility walk terminates ?

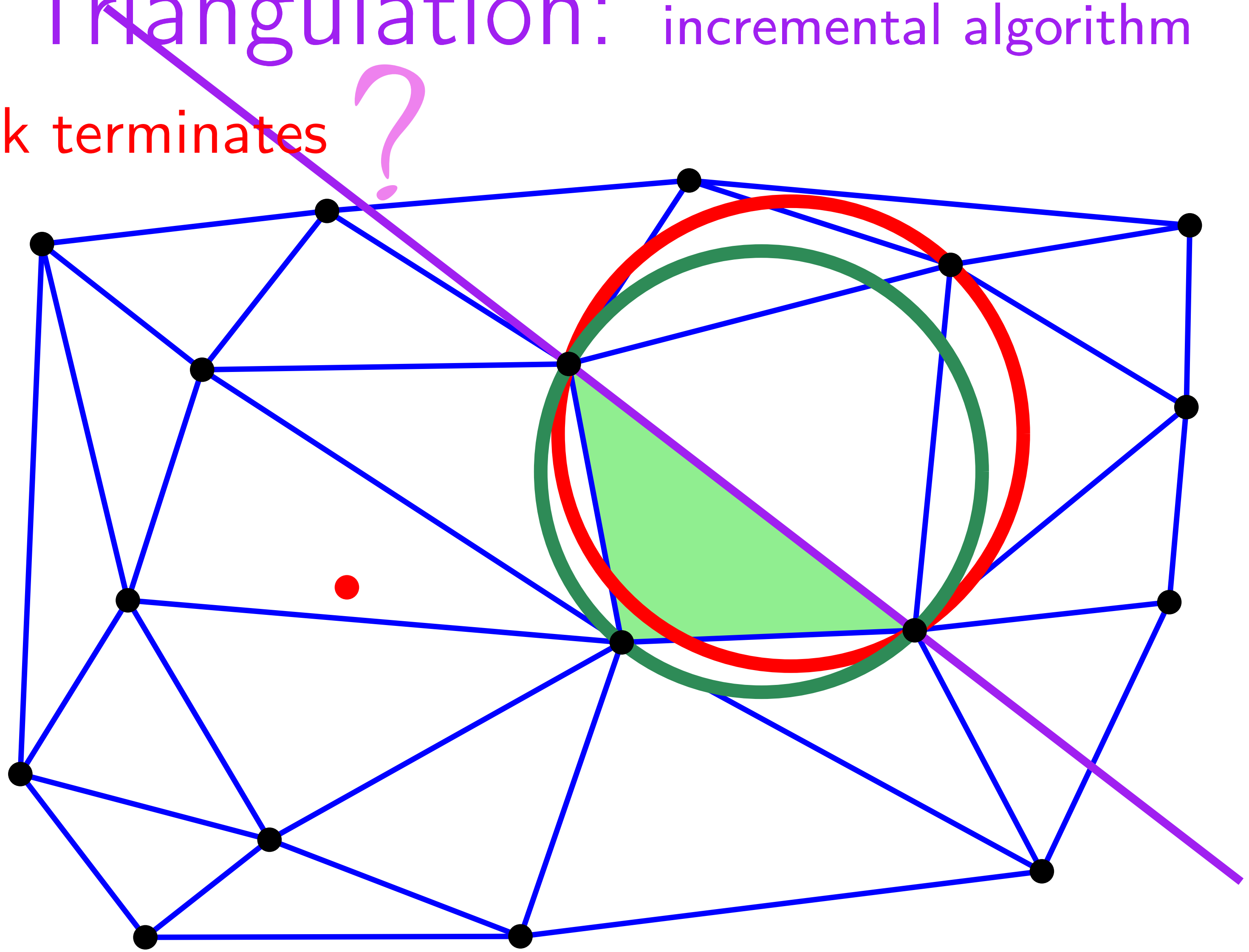


Green power < Red power

Power decreases

Delaunay Triangulation: incremental algorithm

Visibility walk terminates ?



Green power < Red power

Power decreases

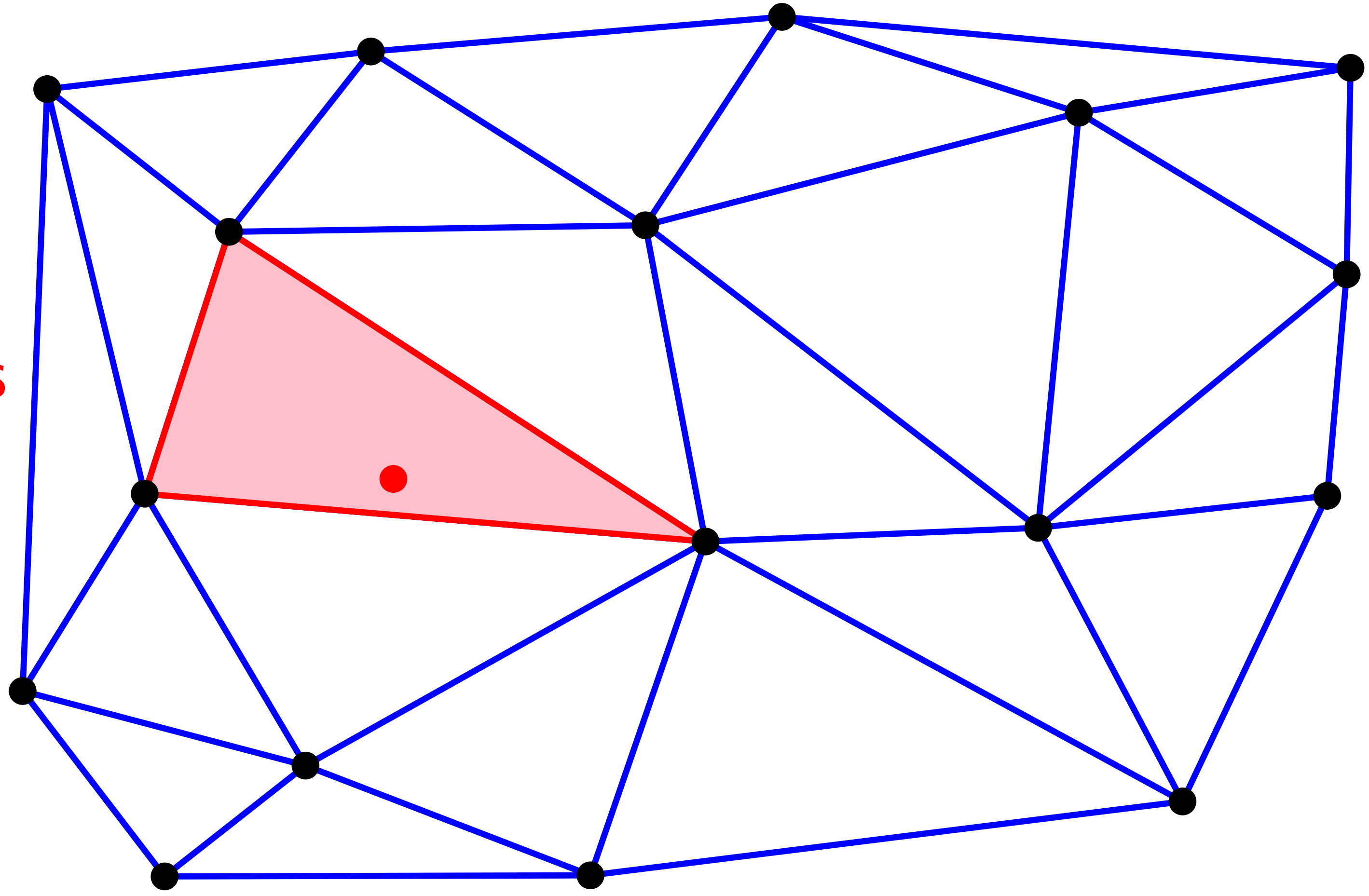
Visibility walk terminates

Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts

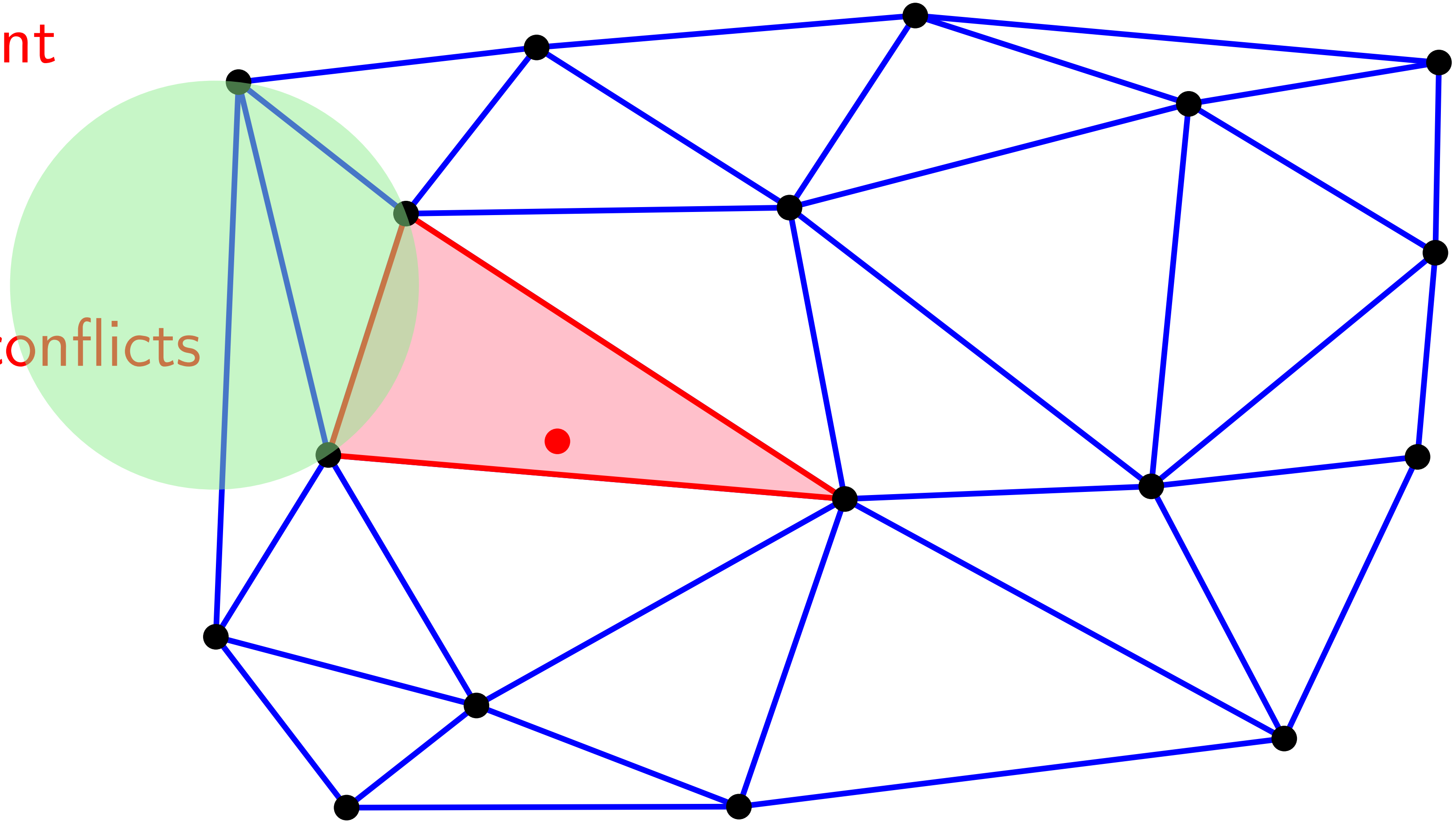


Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts

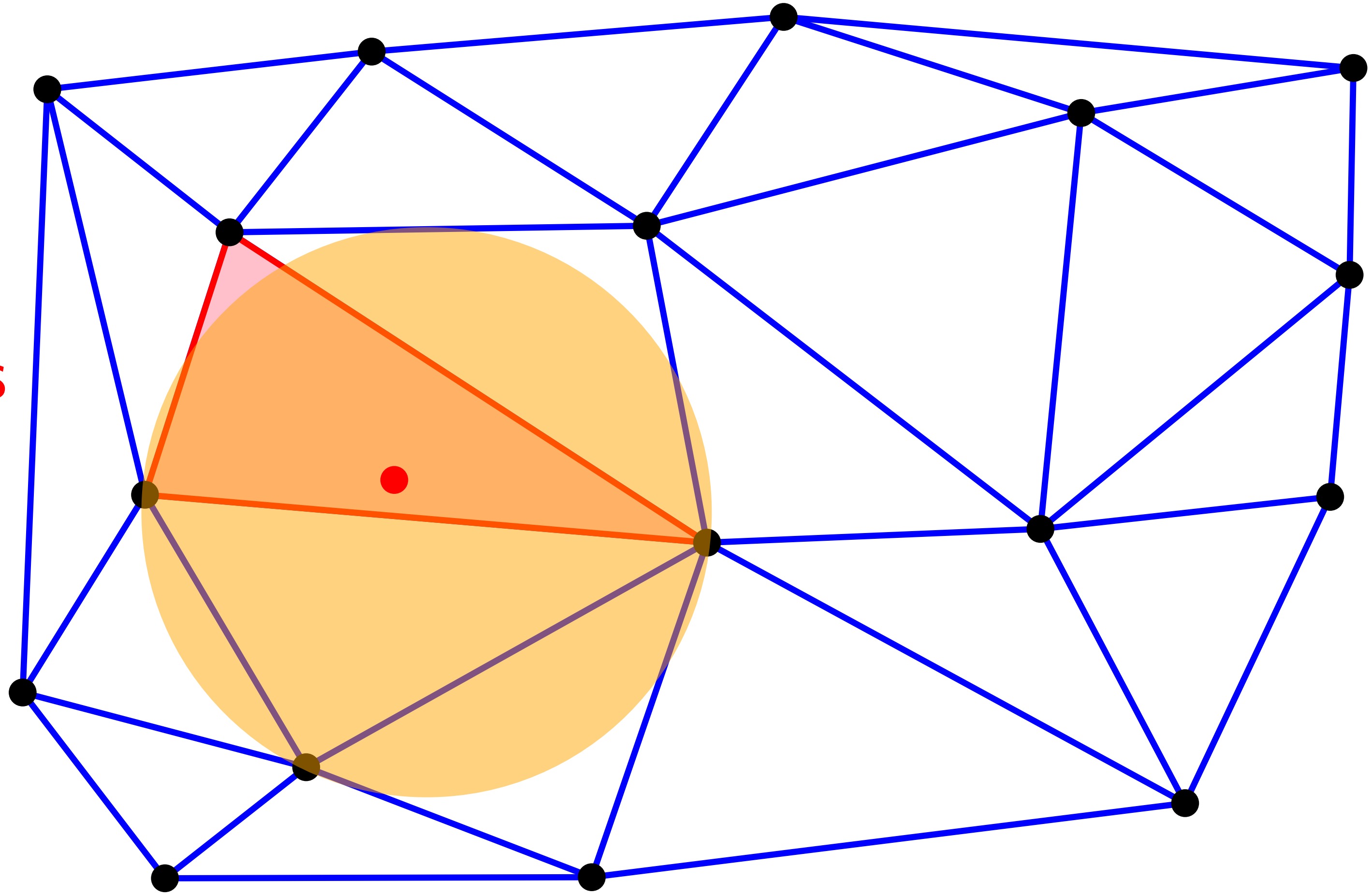


Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts

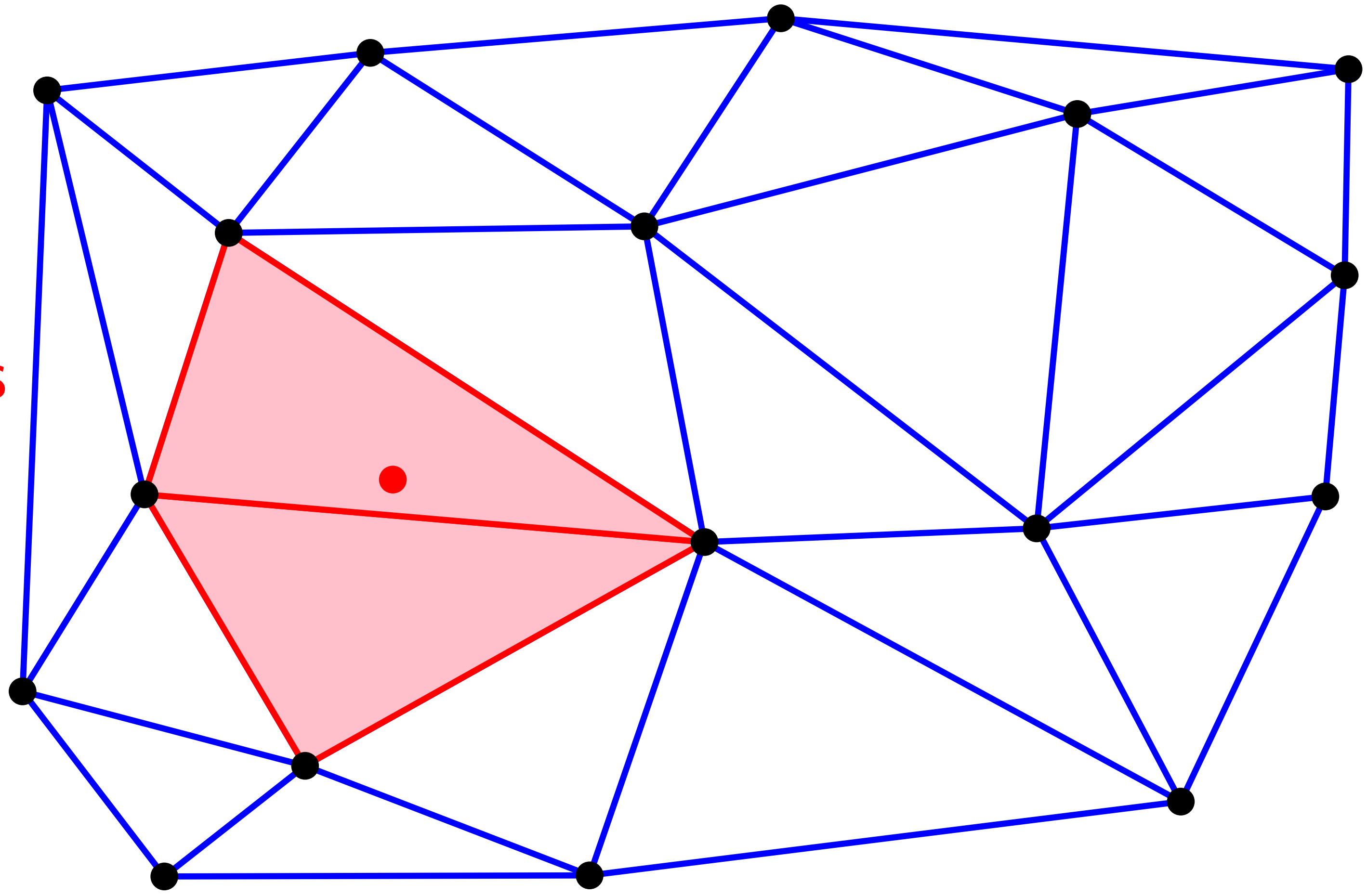


Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts

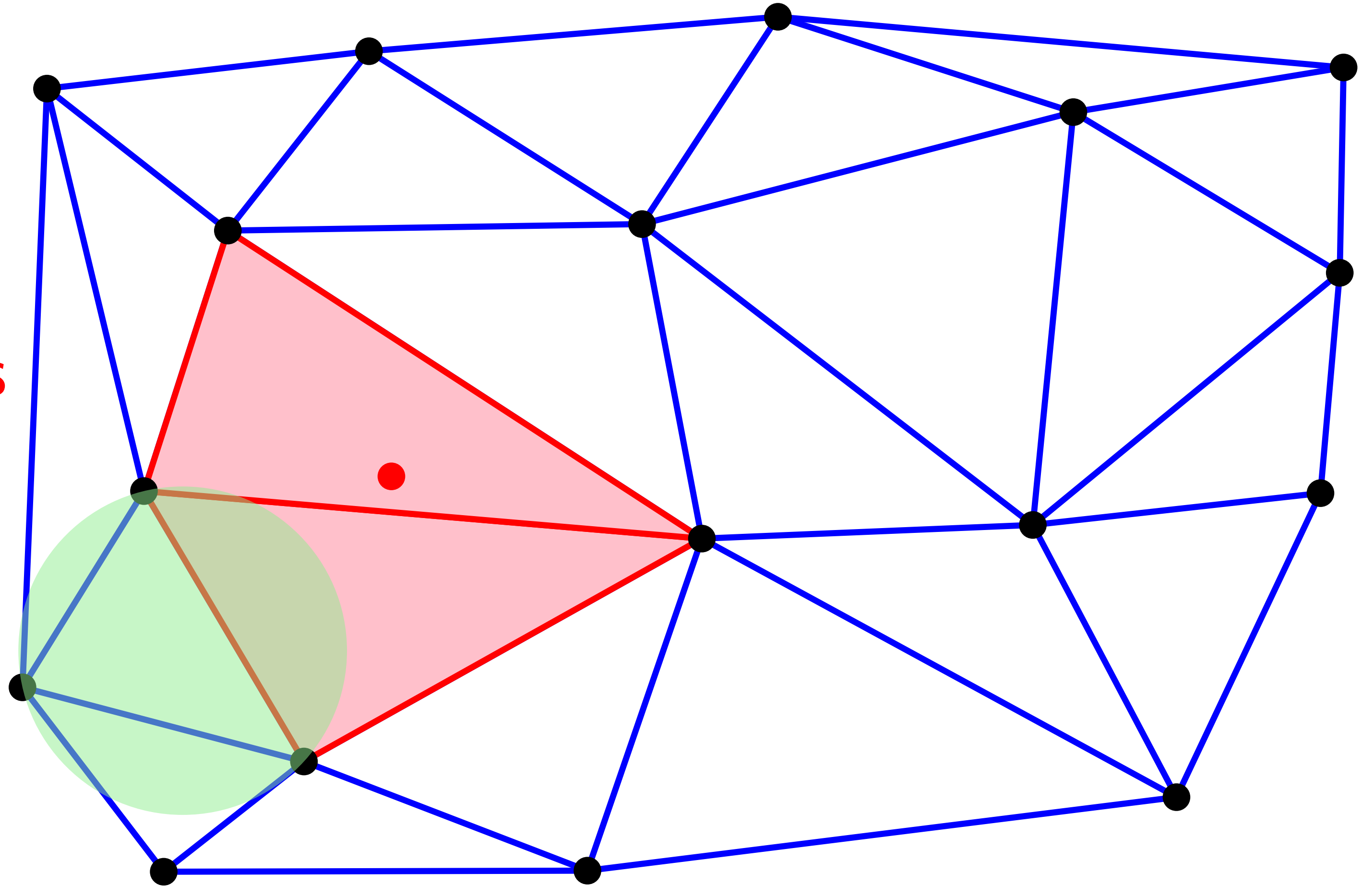


Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts

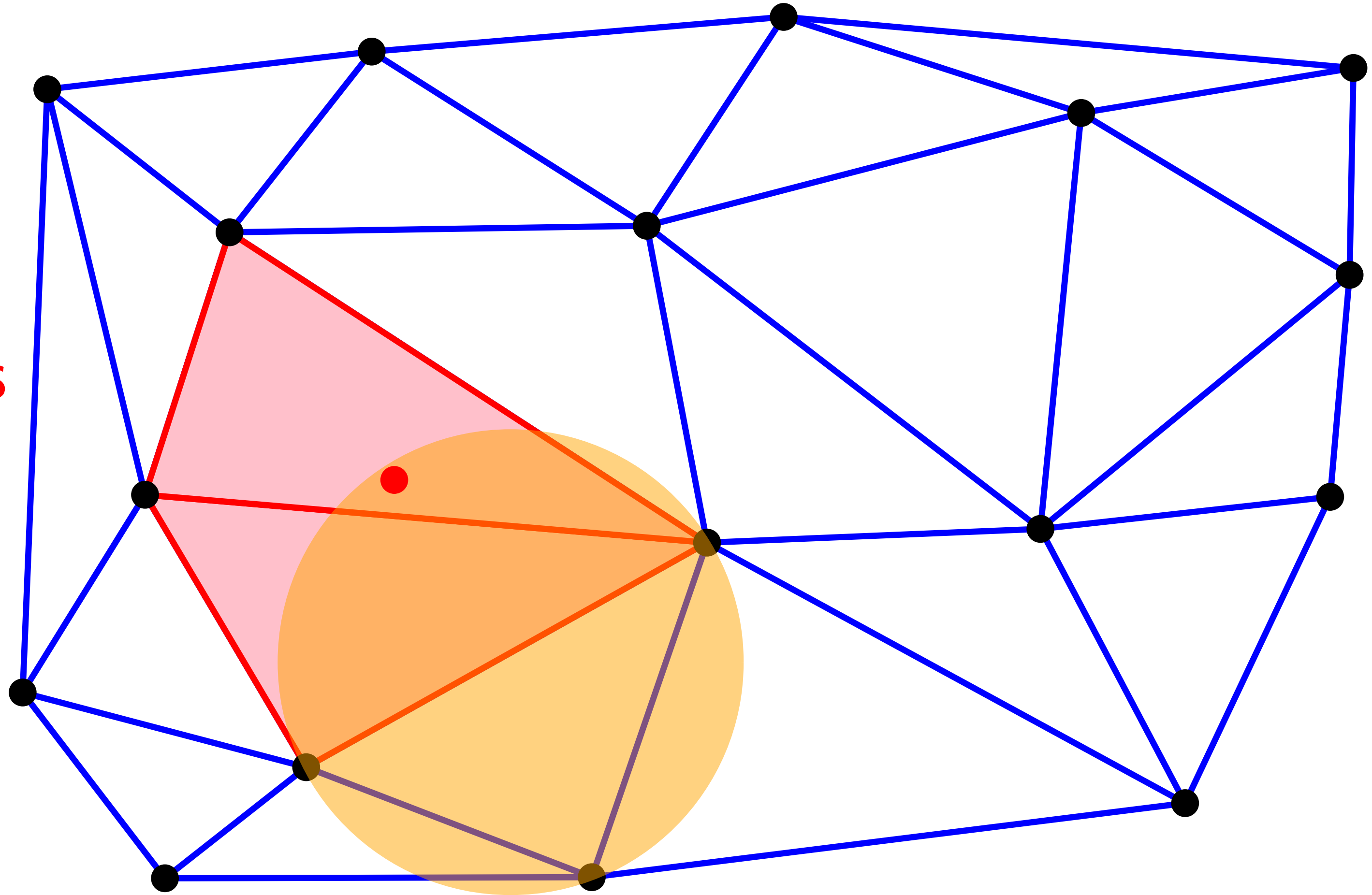


Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts

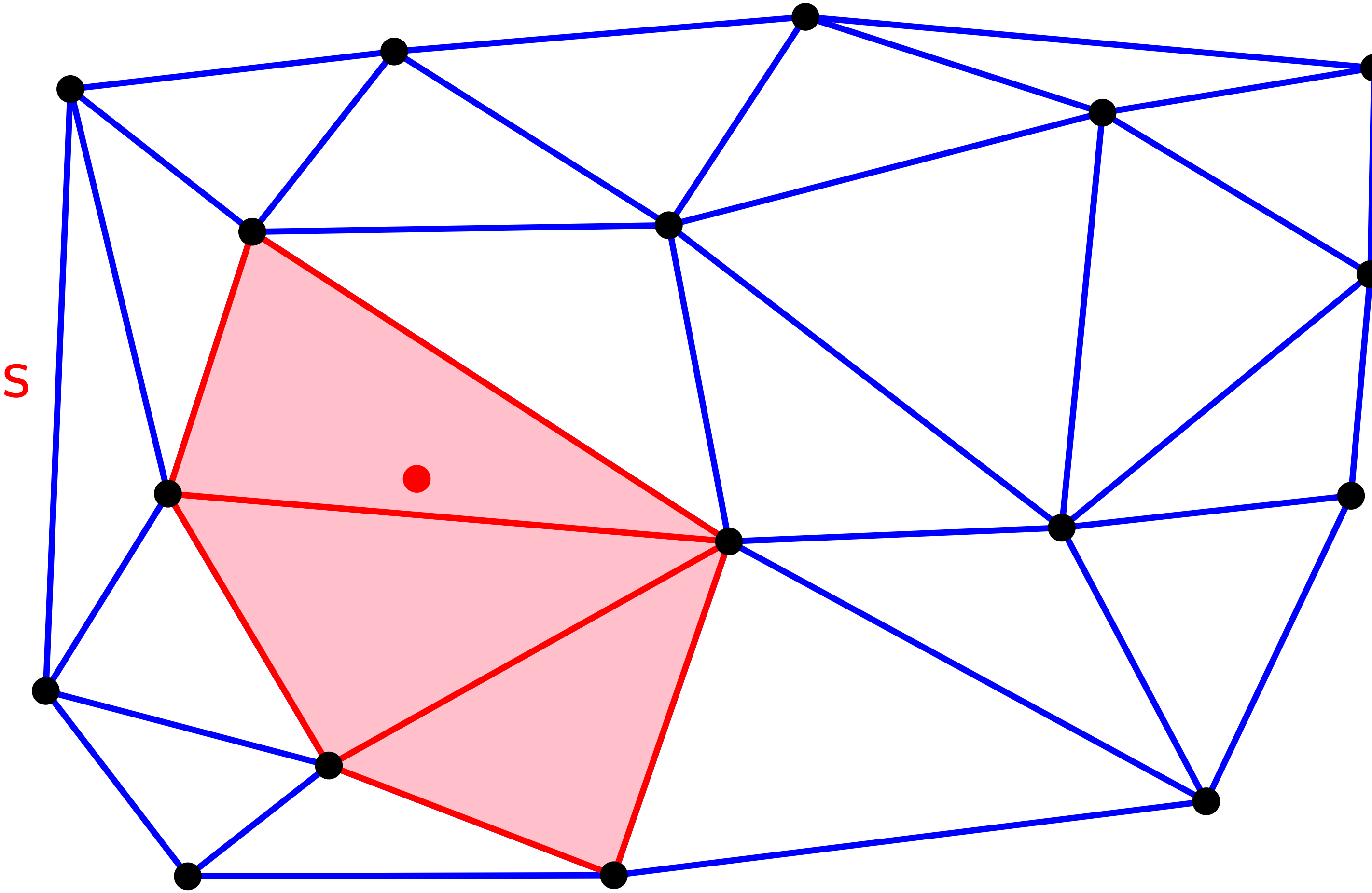


Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts

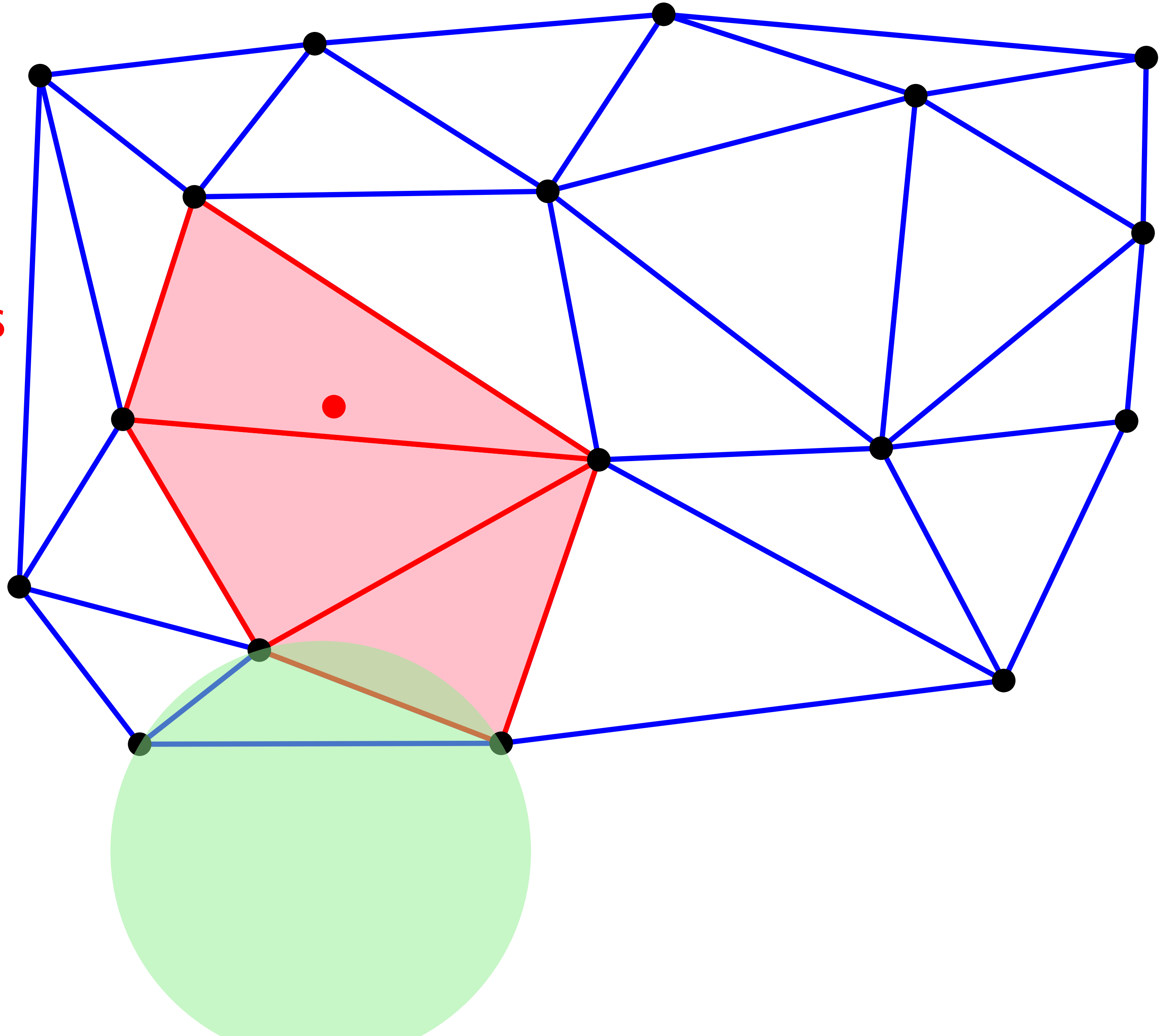


Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts

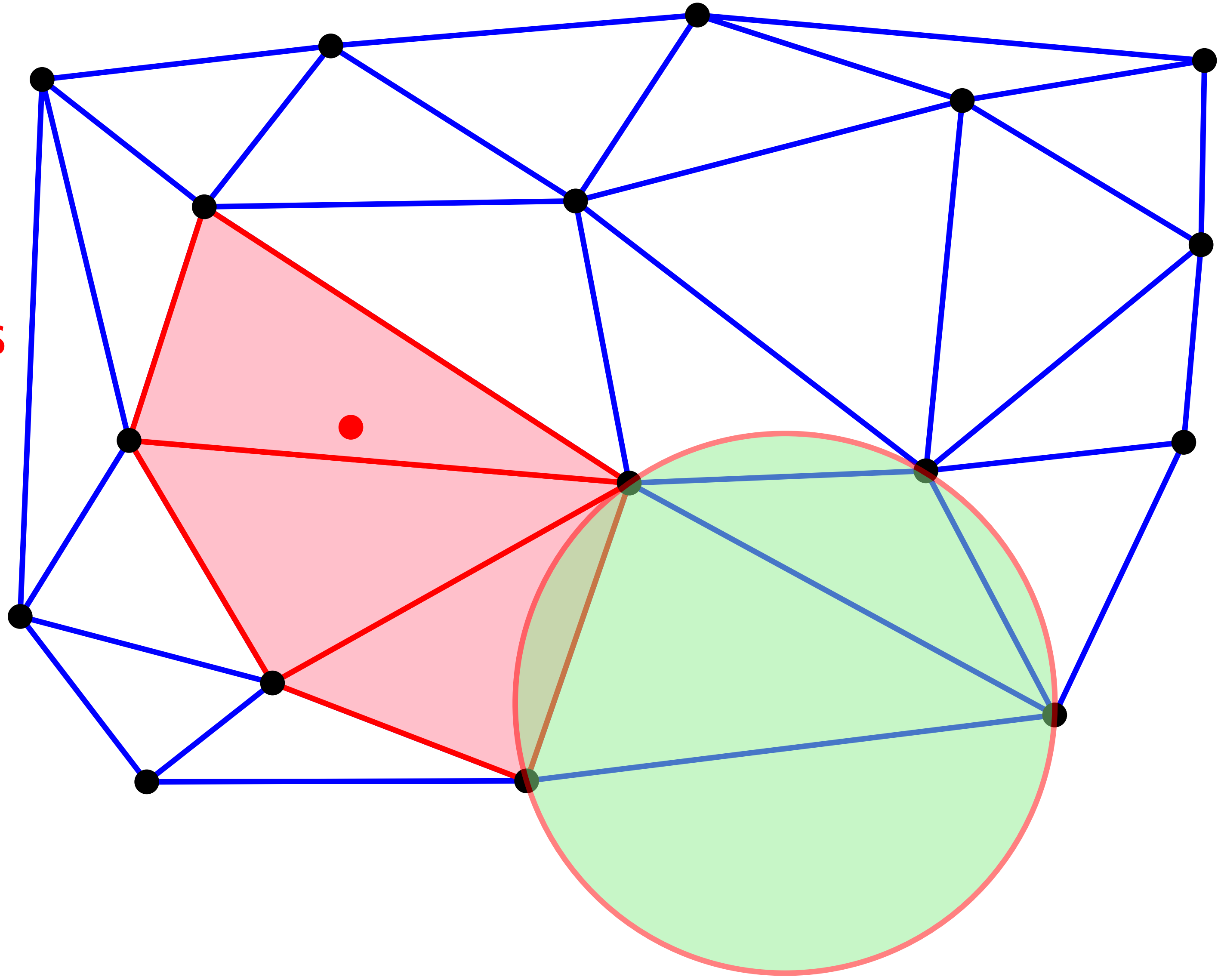


Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts

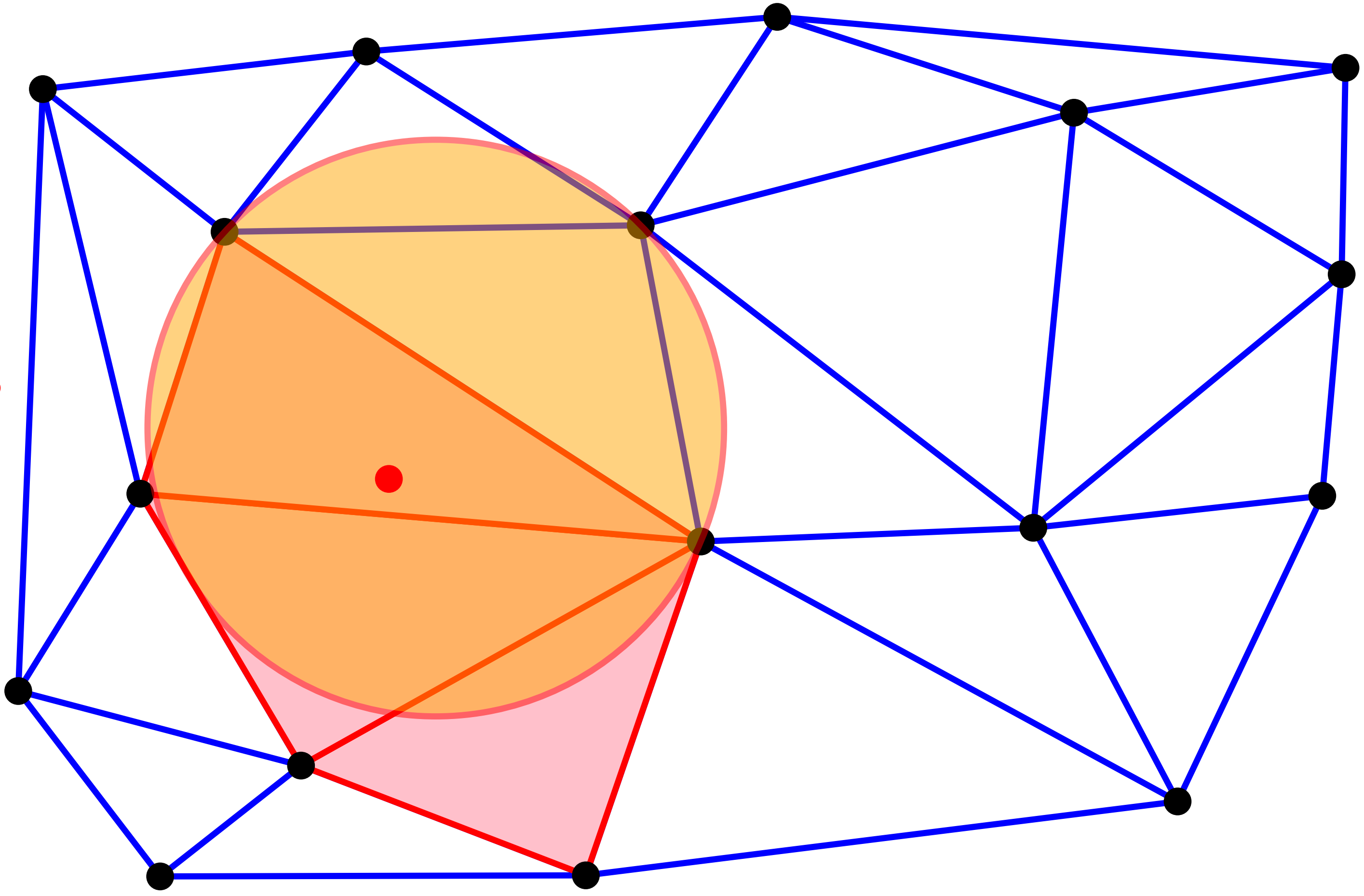


Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts

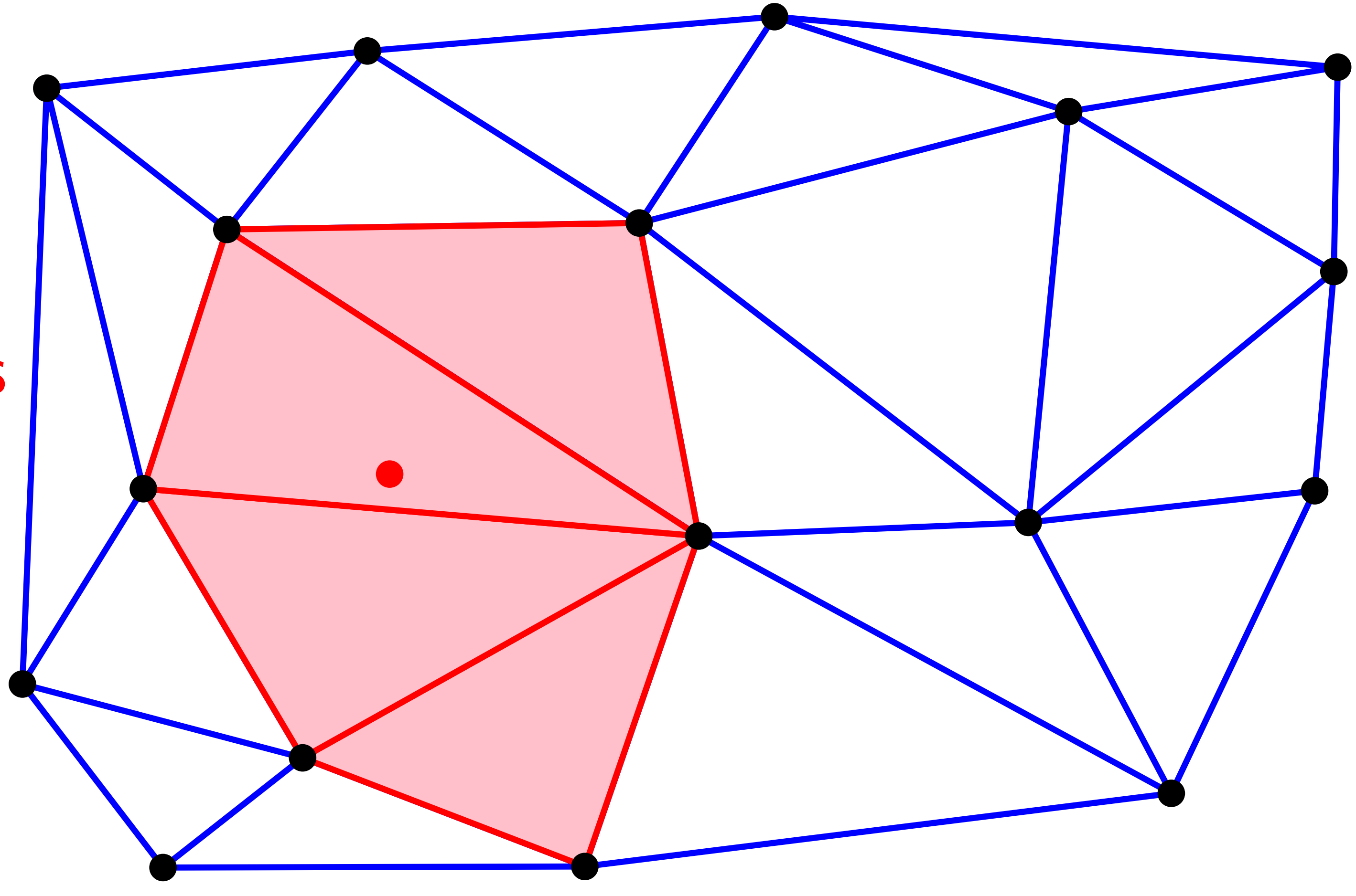


Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts

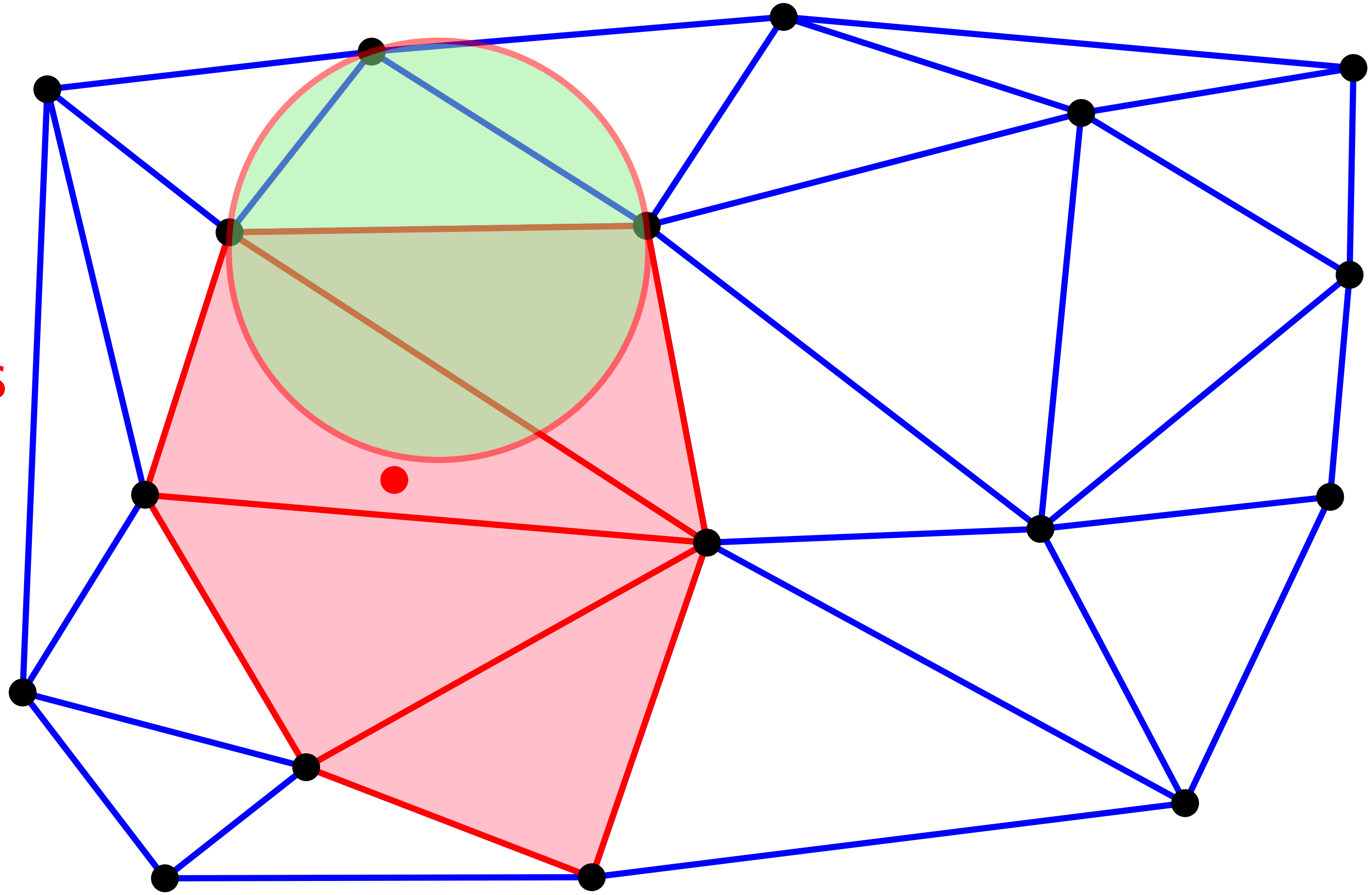


Delaunay Triangulation: incremental algorithm

New point

Locate

Search conflicts

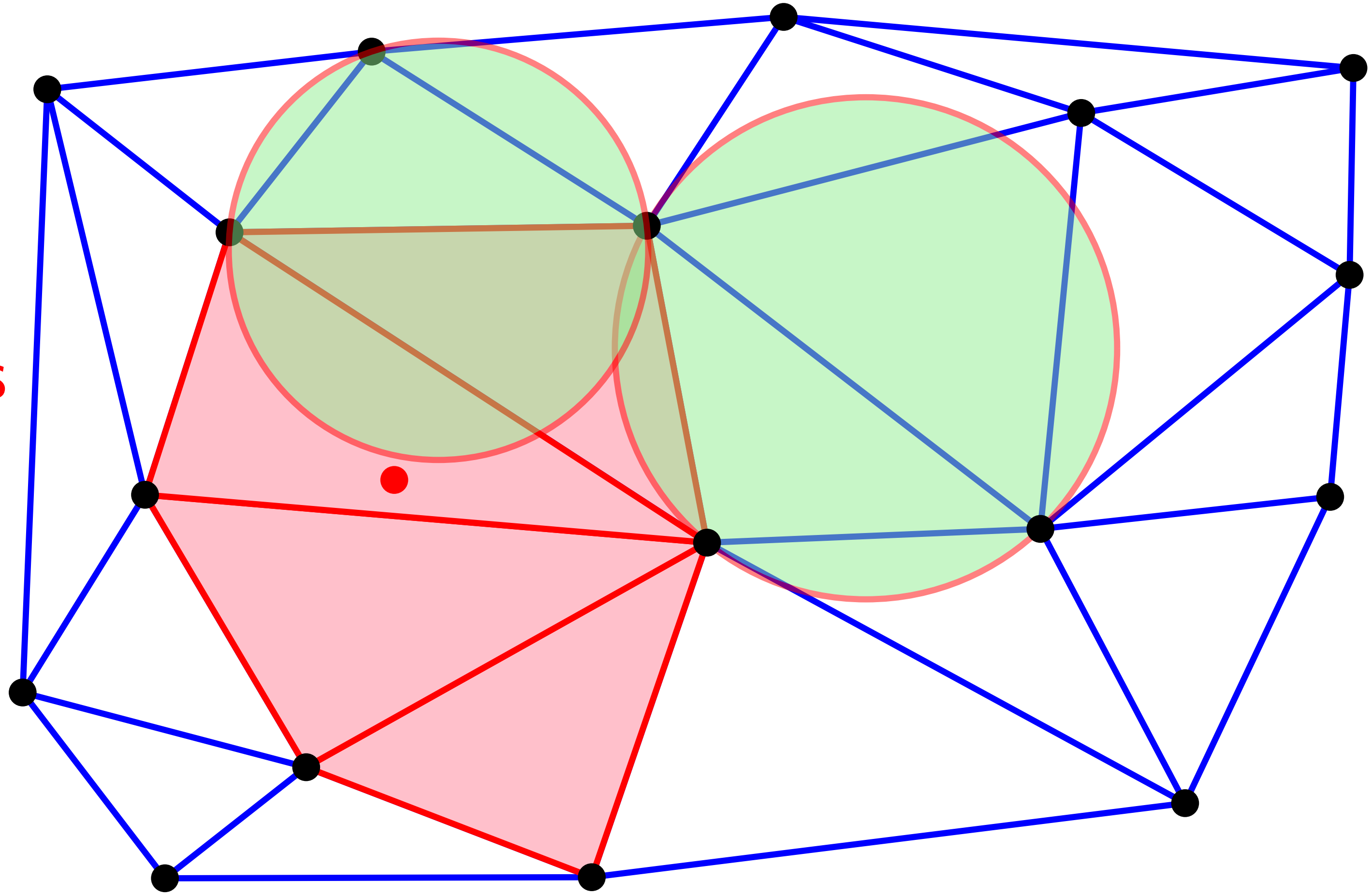


Delaunay Triangulation: incremental algorithm

New point

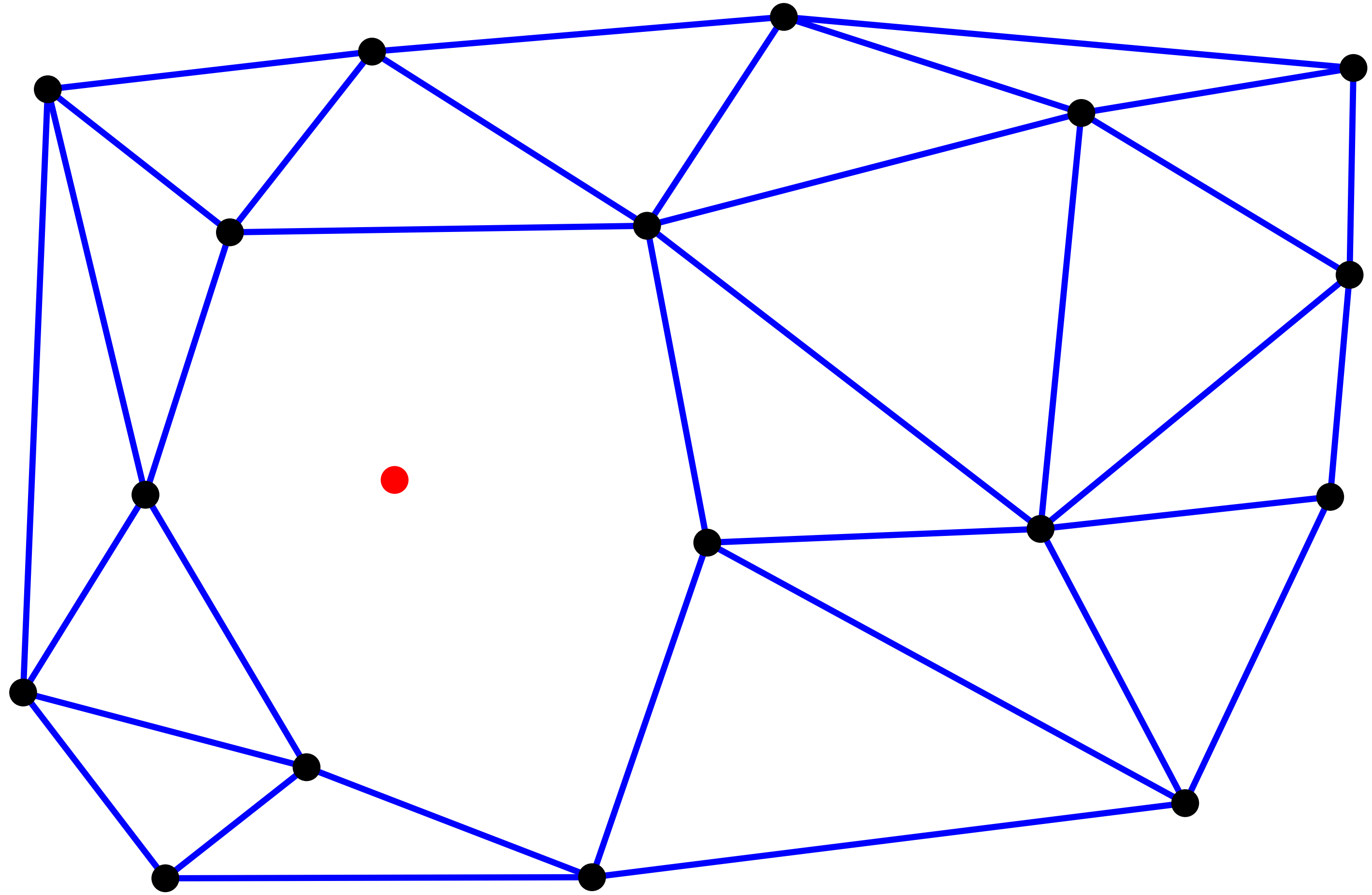
Locate

Search conflicts



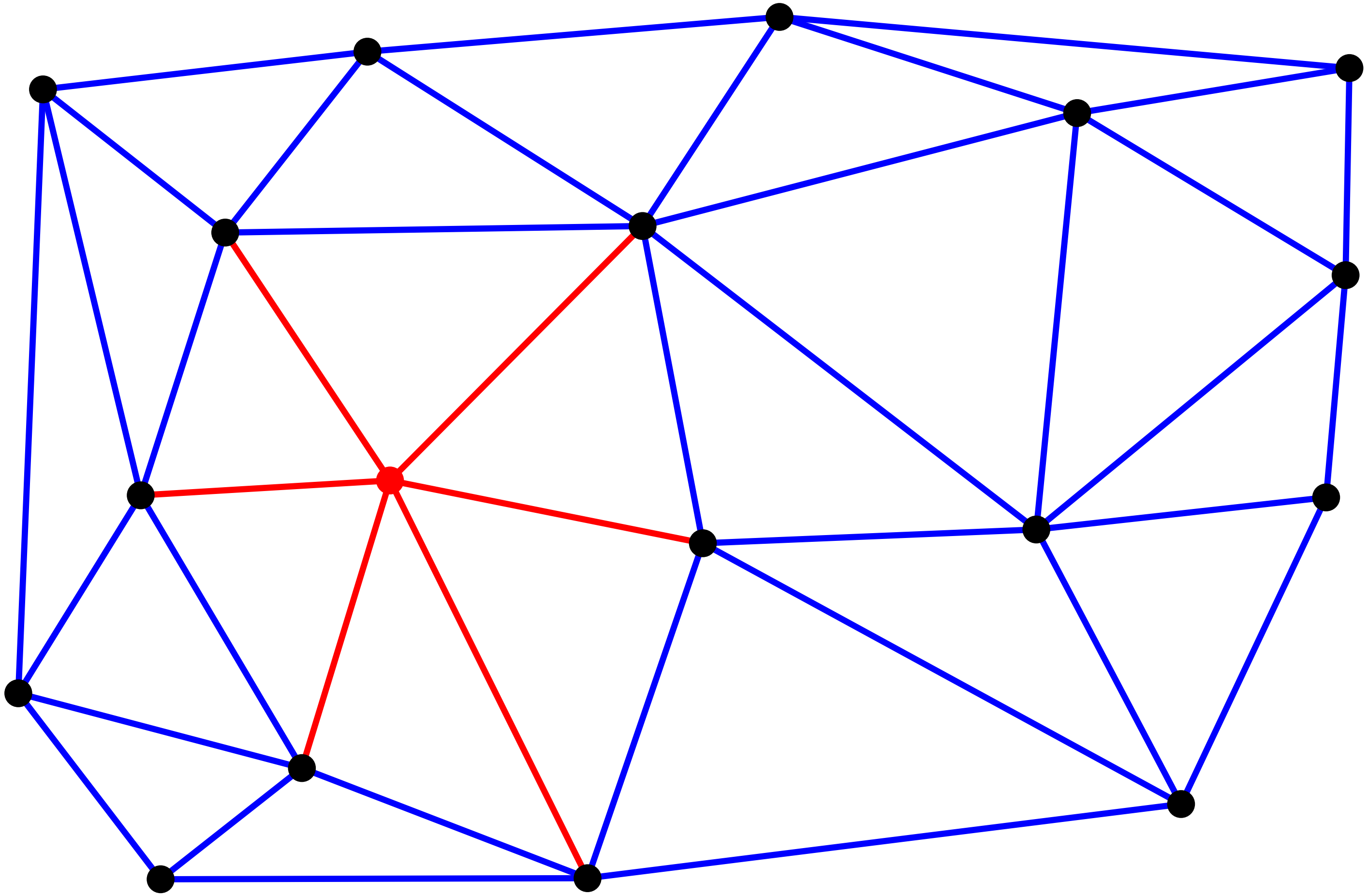
Delaunay Triangulation: incremental algorithm

New point



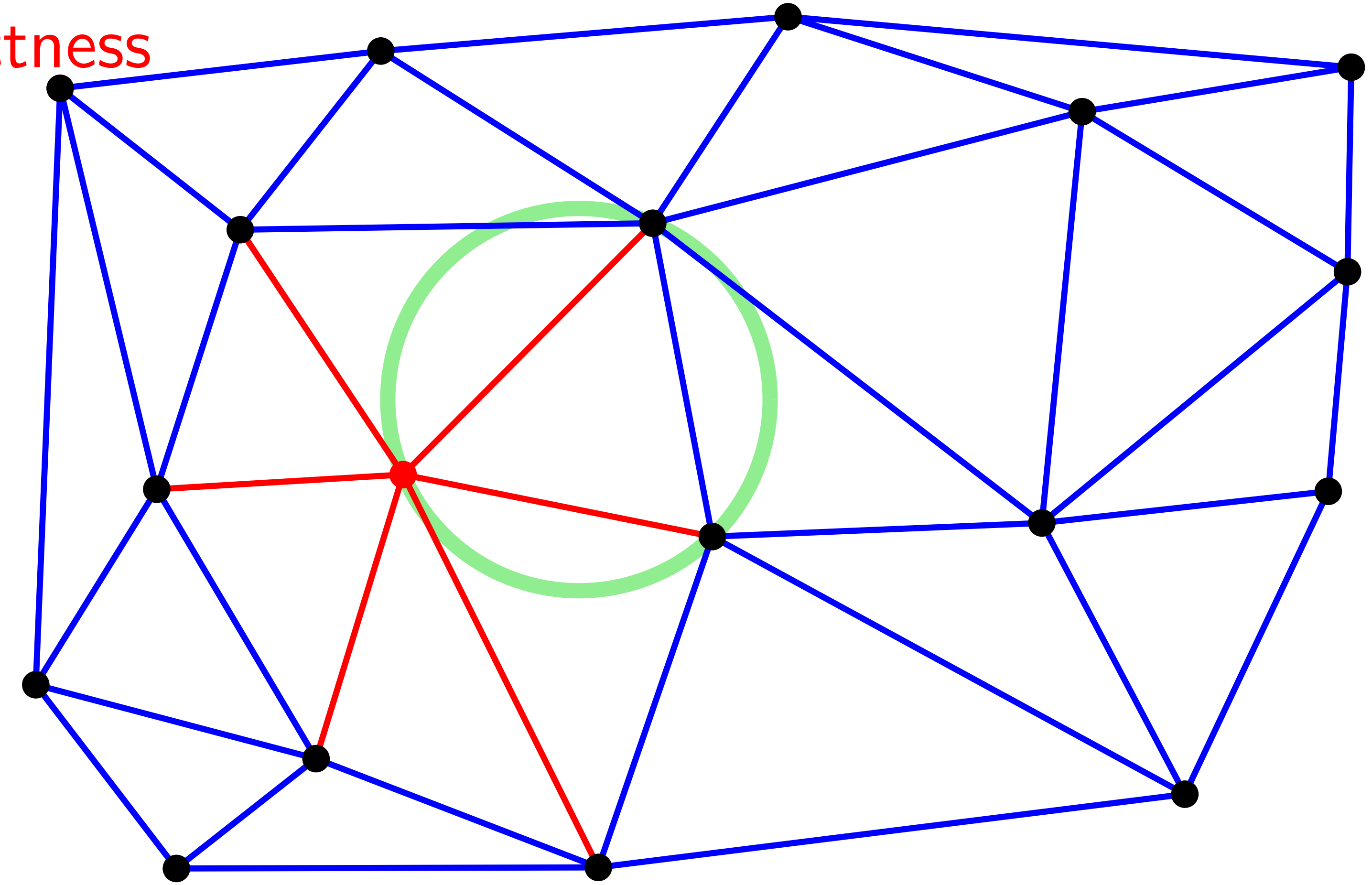
Delaunay Triangulation: incremental algorithm

New point



Delaunay Triangulation: incremental algorithm

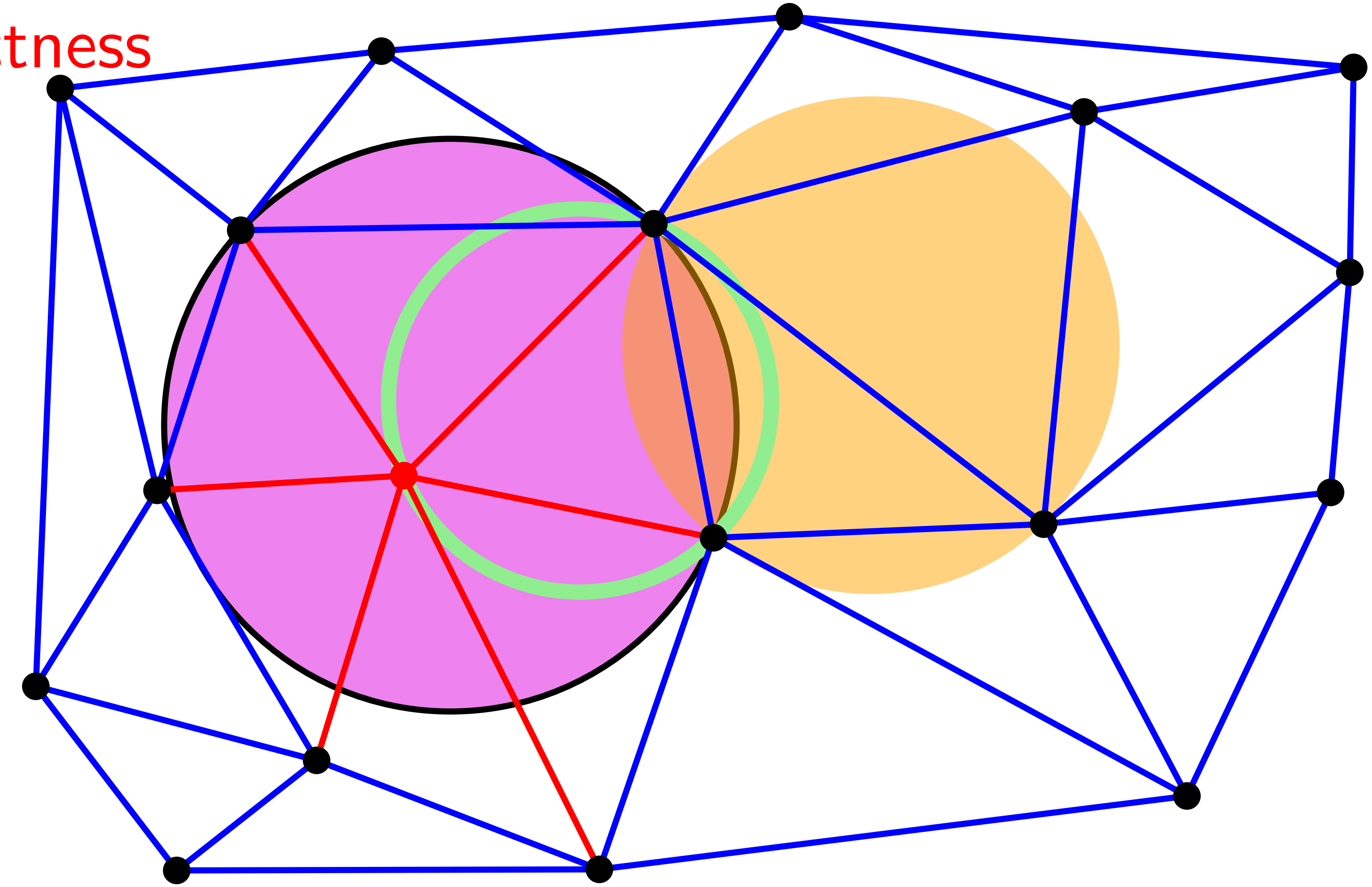
Proof of correctness

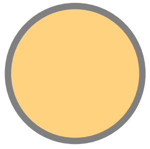
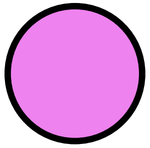


circle of new triangle

Delaunay Triangulation: incremental algorithm

Proof of correctness



circle of new triangle $\subset$  $\cup$ 

Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts

Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts $\sharp$ triangles in conflict

$\sharp$ triangles neighboring triangles in conflict

Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts $\sharp$ triangles in conflict

$\sharp$ triangles neighboring triangles in conflict

degree of new point in new triangulation

$< n$

Delaunay Triangulation: incremental algorithm

Complexity

Locate

Walk may visit all triangles

$< 2n$

Search conflicts

degree of new point in new triangulation

$< n$

Delaunay Triangulation: incremental algorithm

Complexity

Locate $O(n)$ per insertion

Search conflicts

Delaunay Triangulation: incremental algorithm

Complexity

Locate $O(n)$ per insertion

Search conflicts

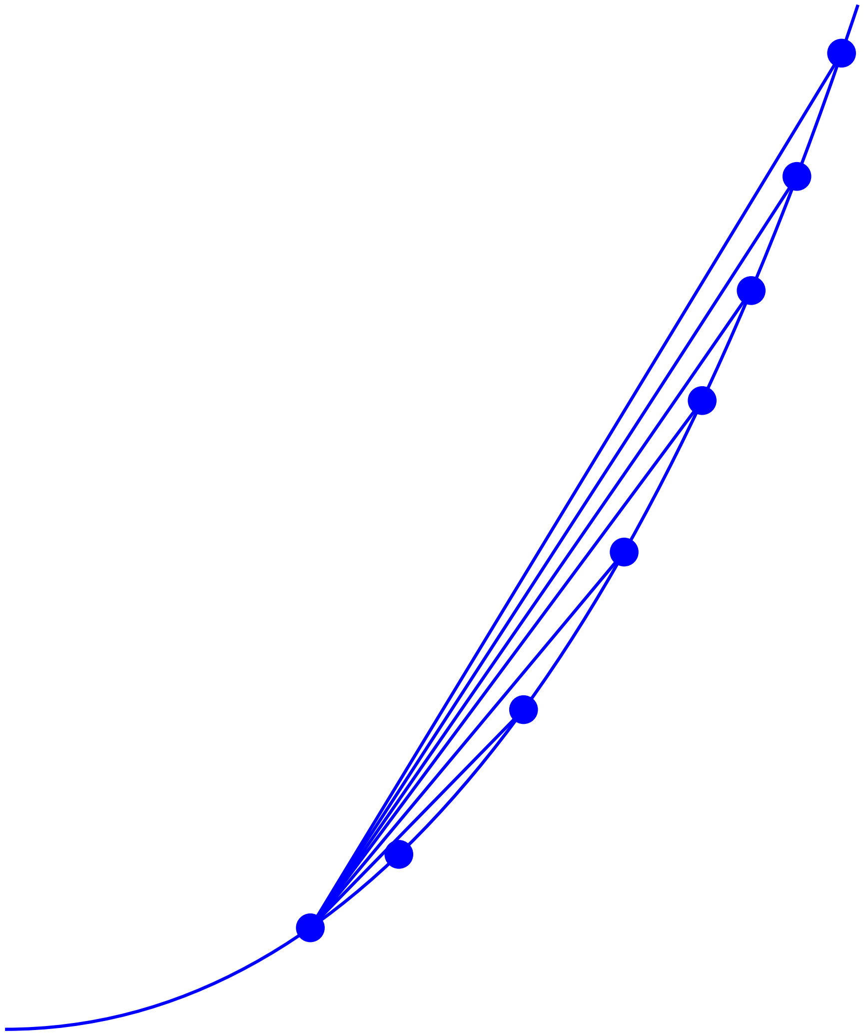
$O(n^2)$ for the whole construction

Delaunay Triangulation: incremental algorithm

Complexity

Locate

Search conflicts



Delaunay Triangulation: incremental algorithm

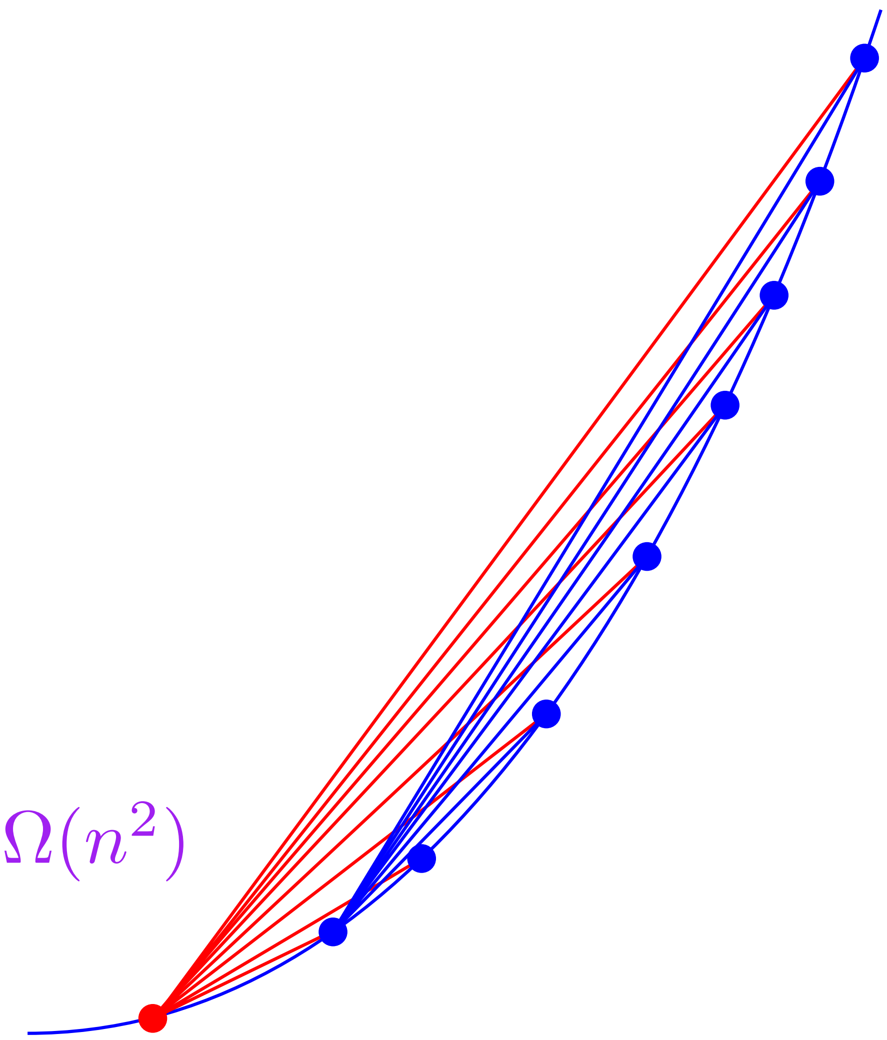
Complexity

Locate

Search conflicts

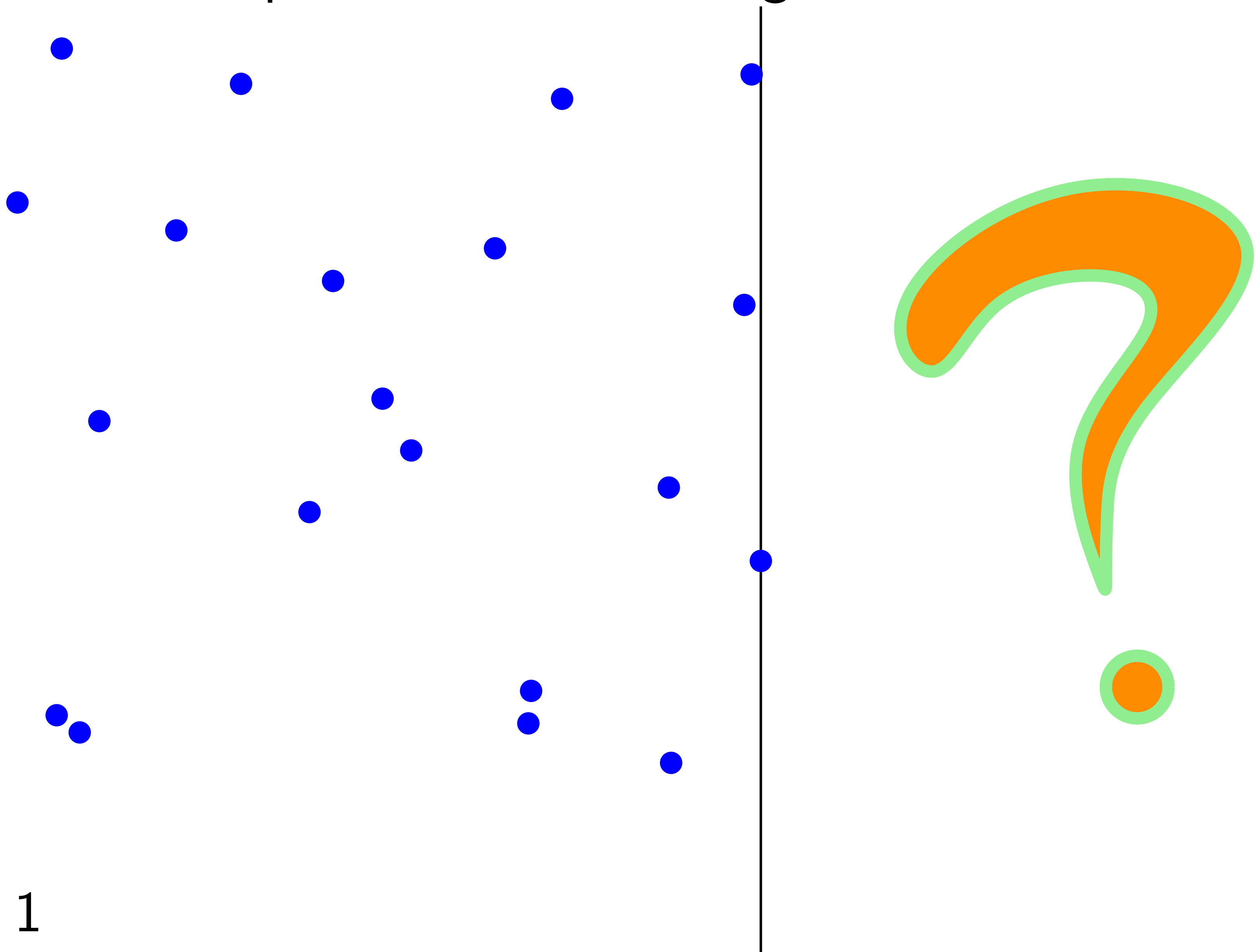
Insertion: $\Omega(n)$

Whole construction: $\Omega(n^2)$



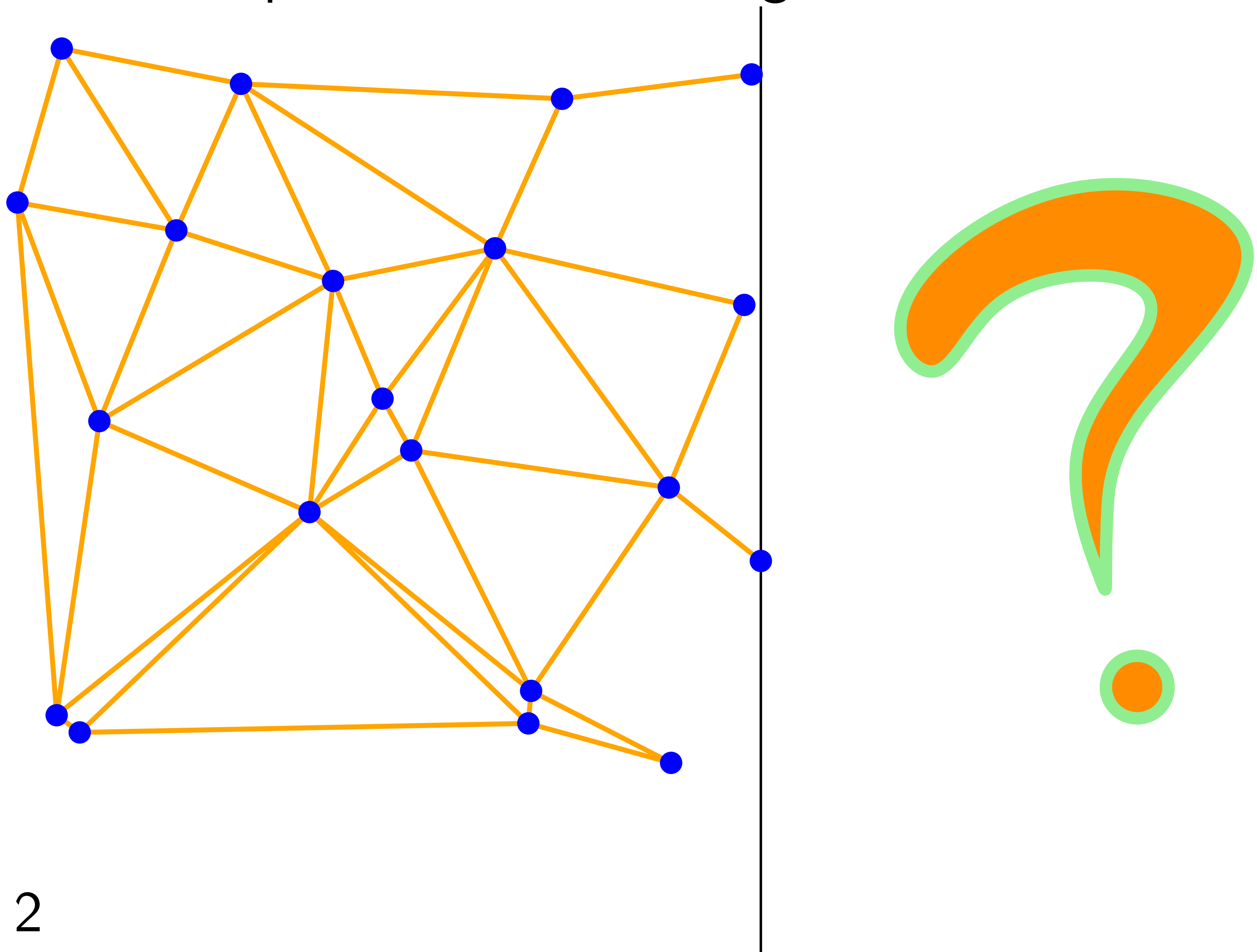
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right



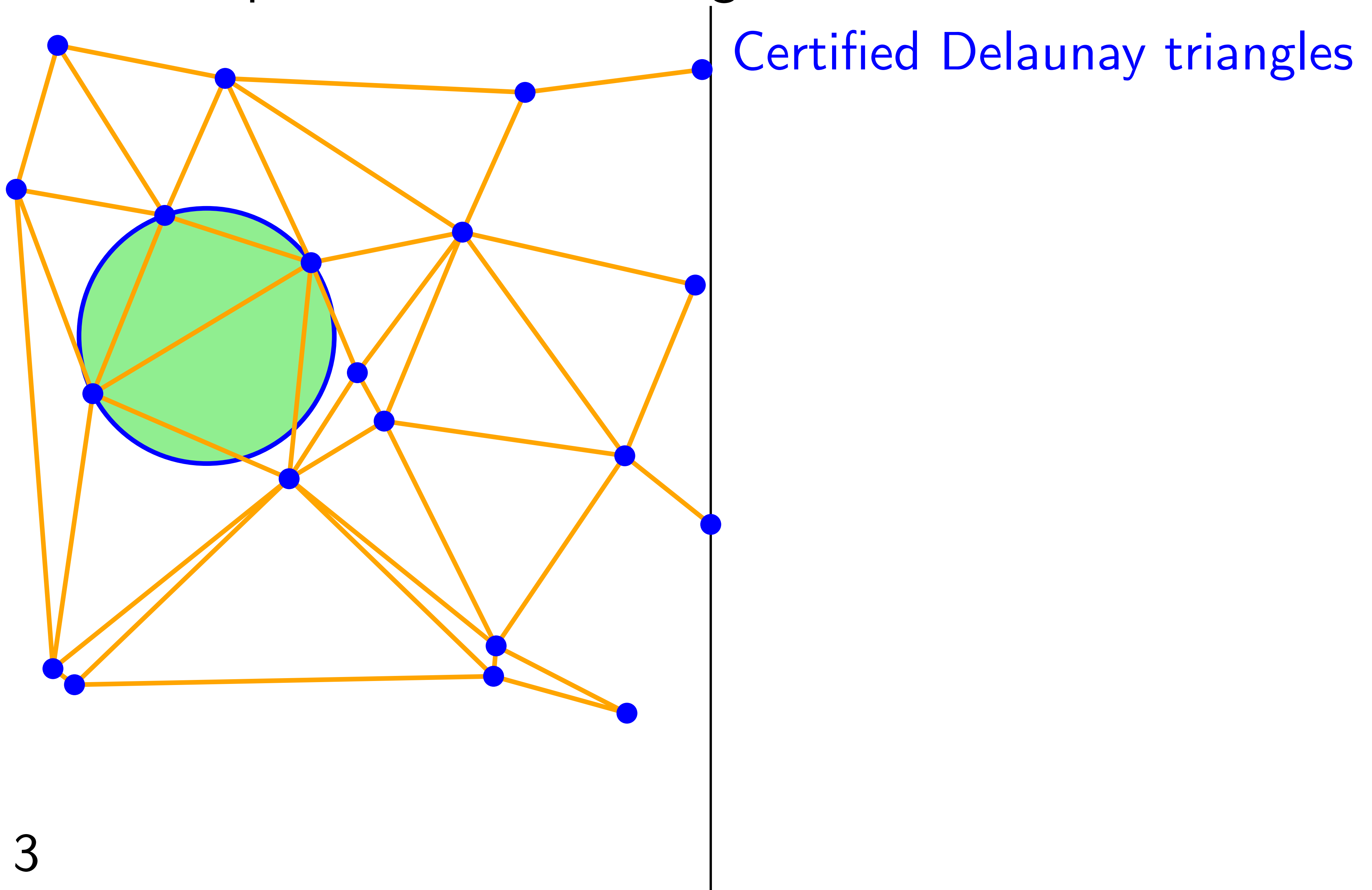
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right



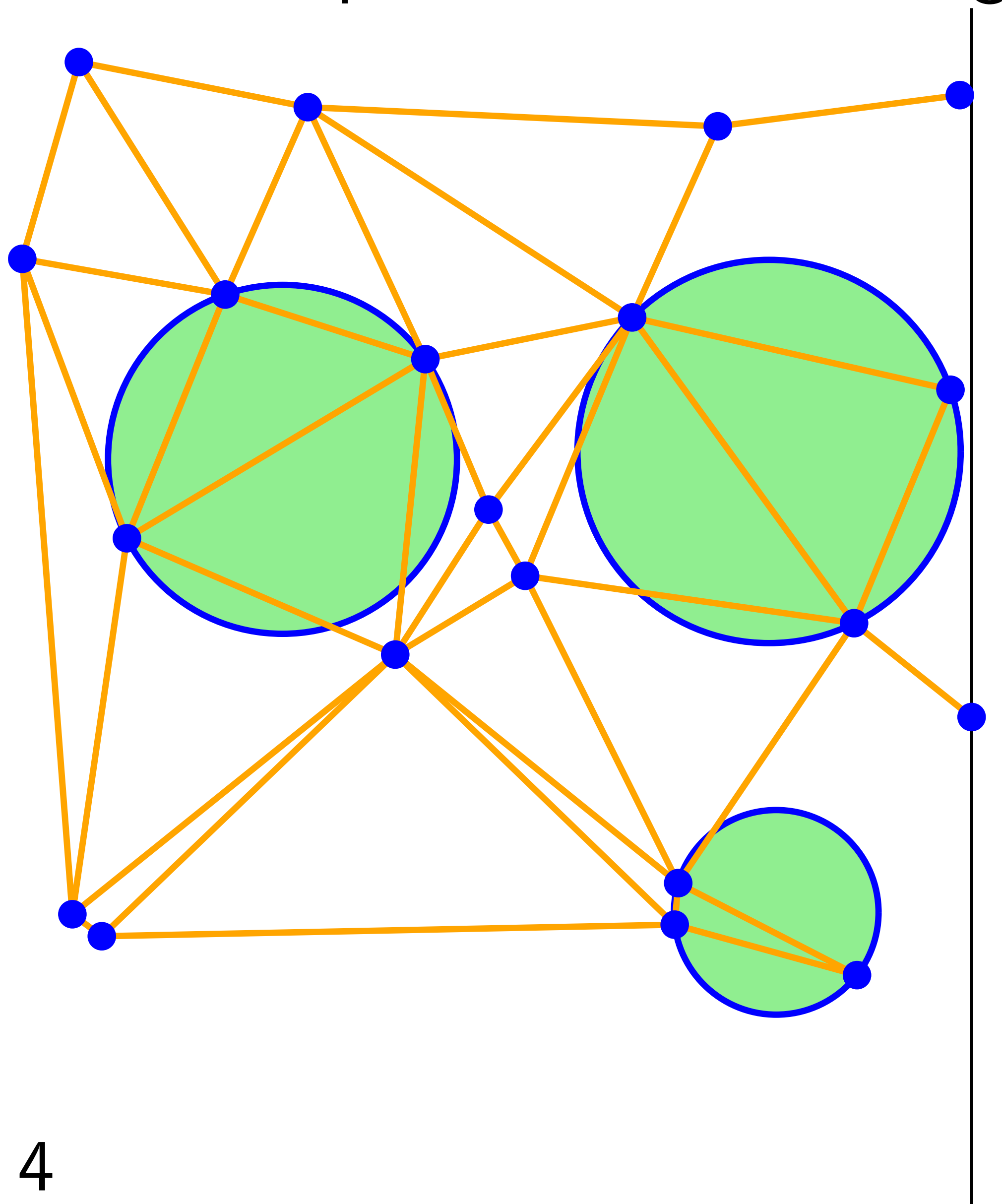
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right



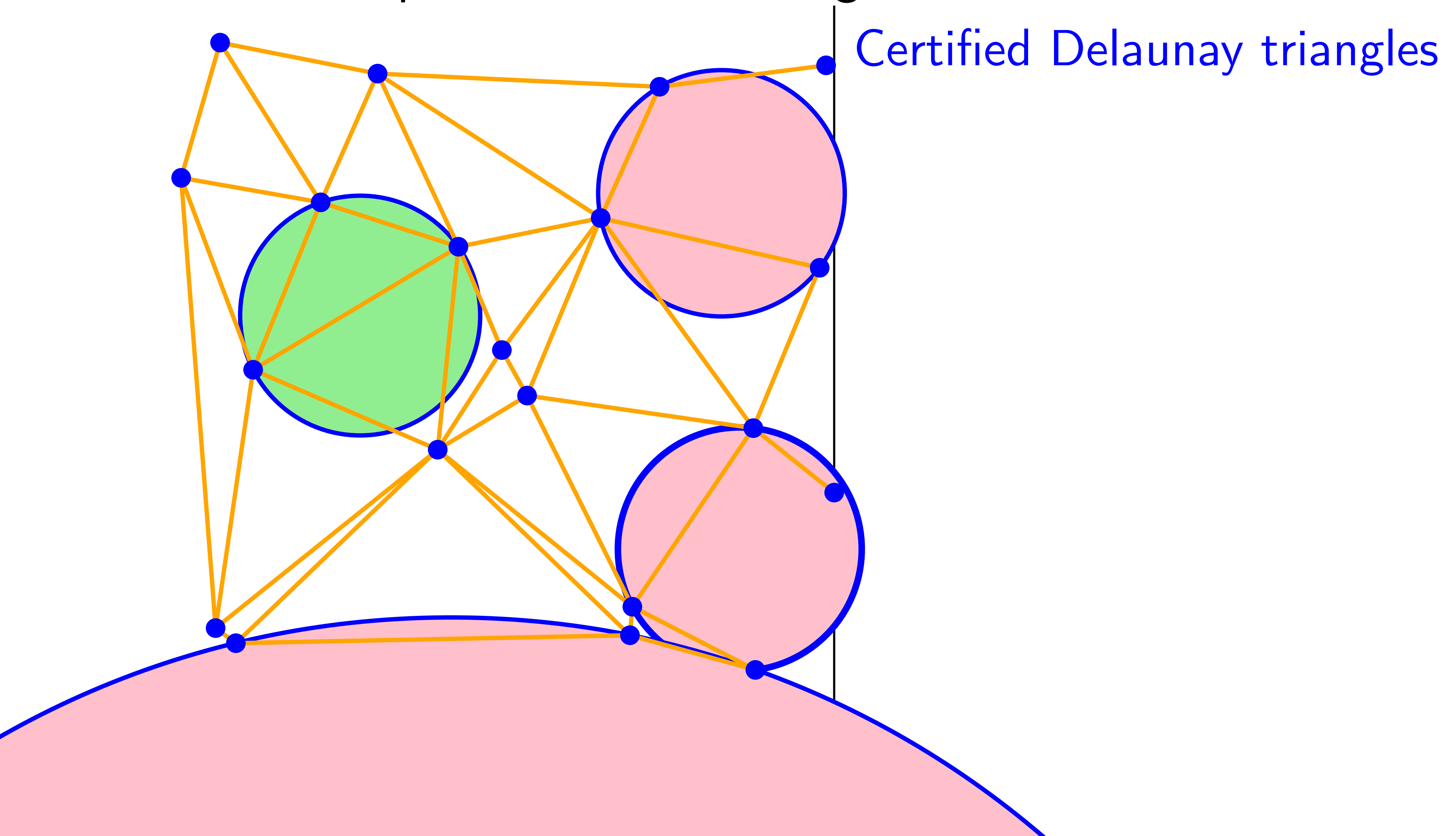
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right



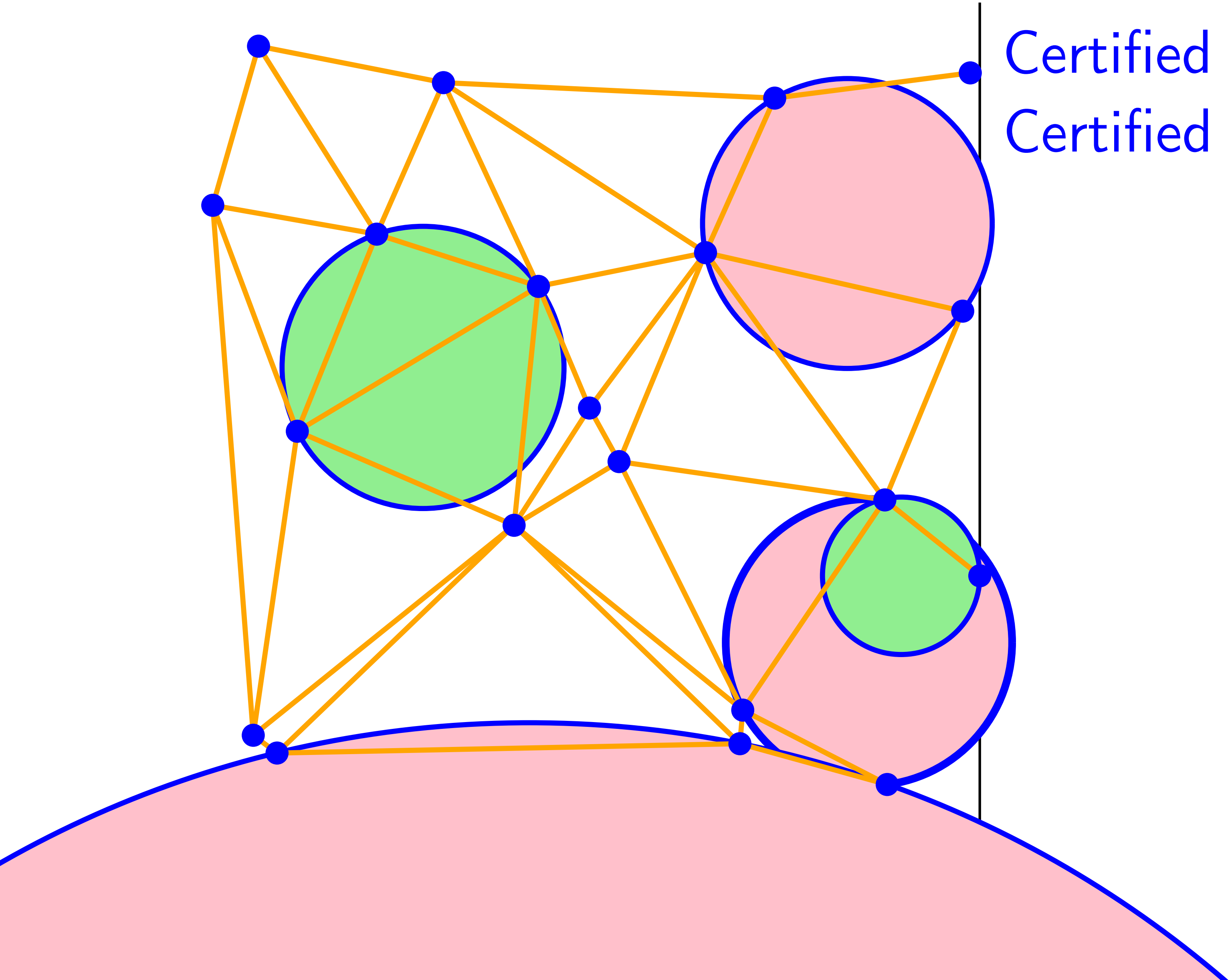
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right



Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right

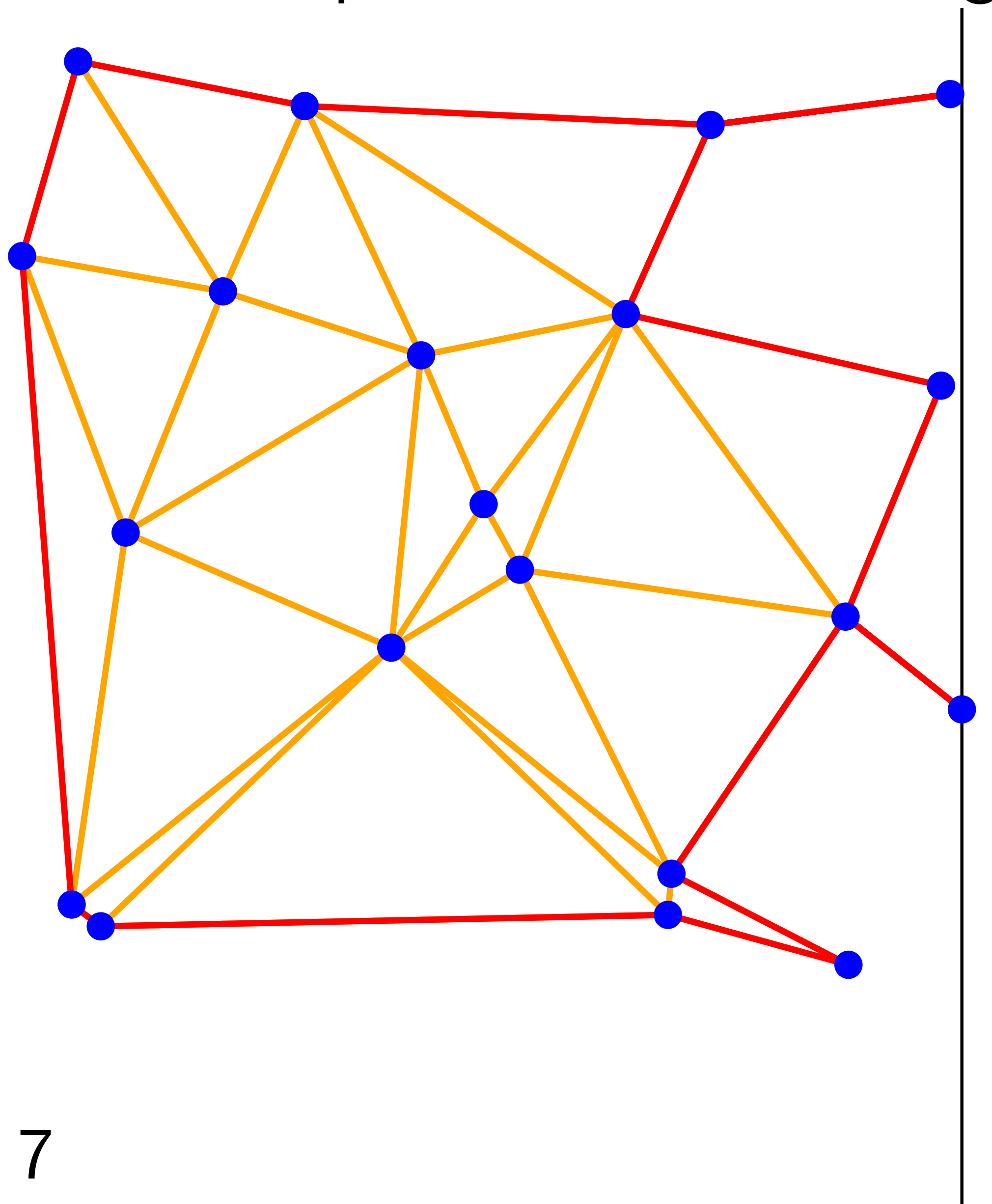


Certified Delaunay triangles

Certified Delaunay edges

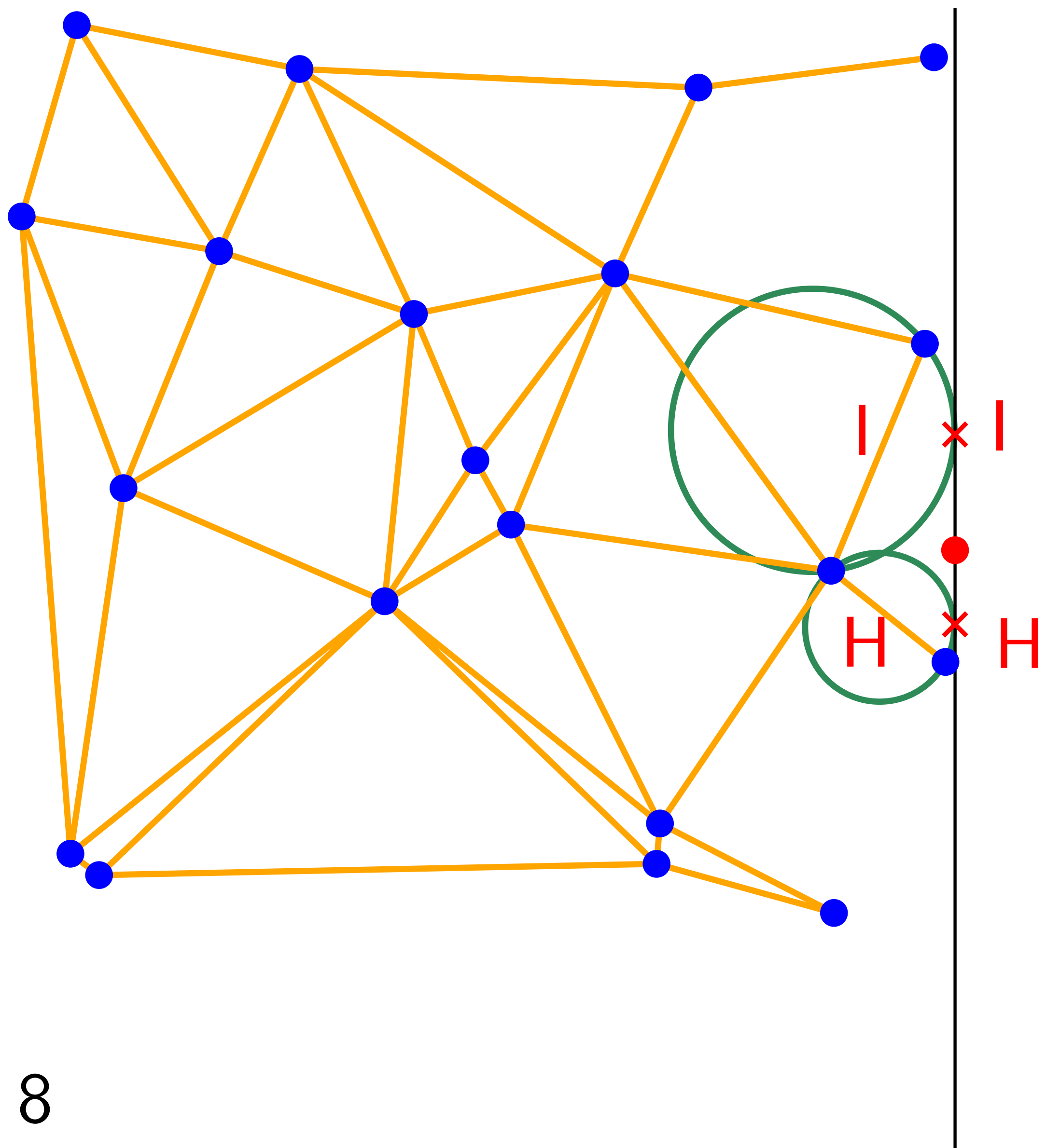
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right



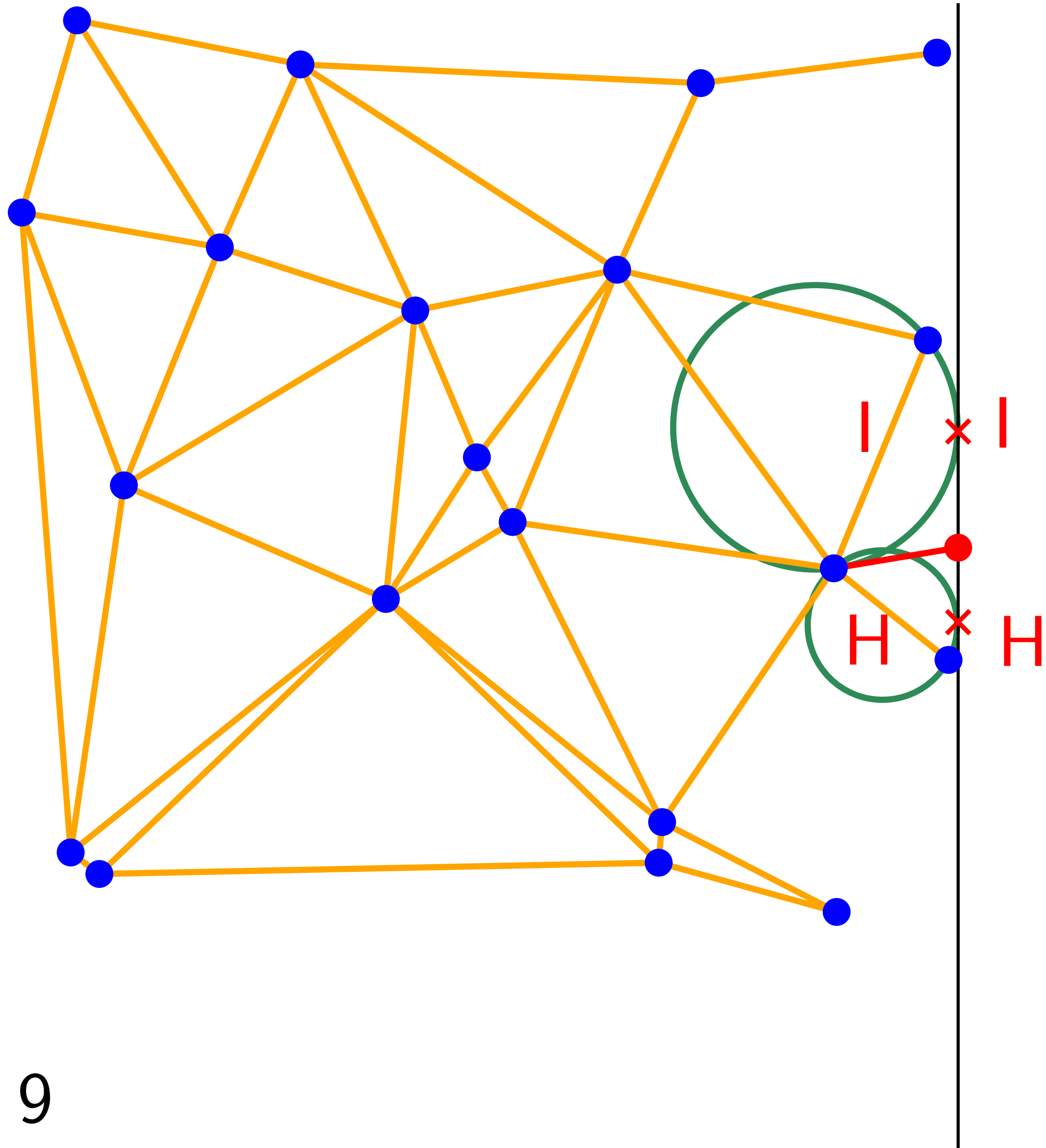
Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right



Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right



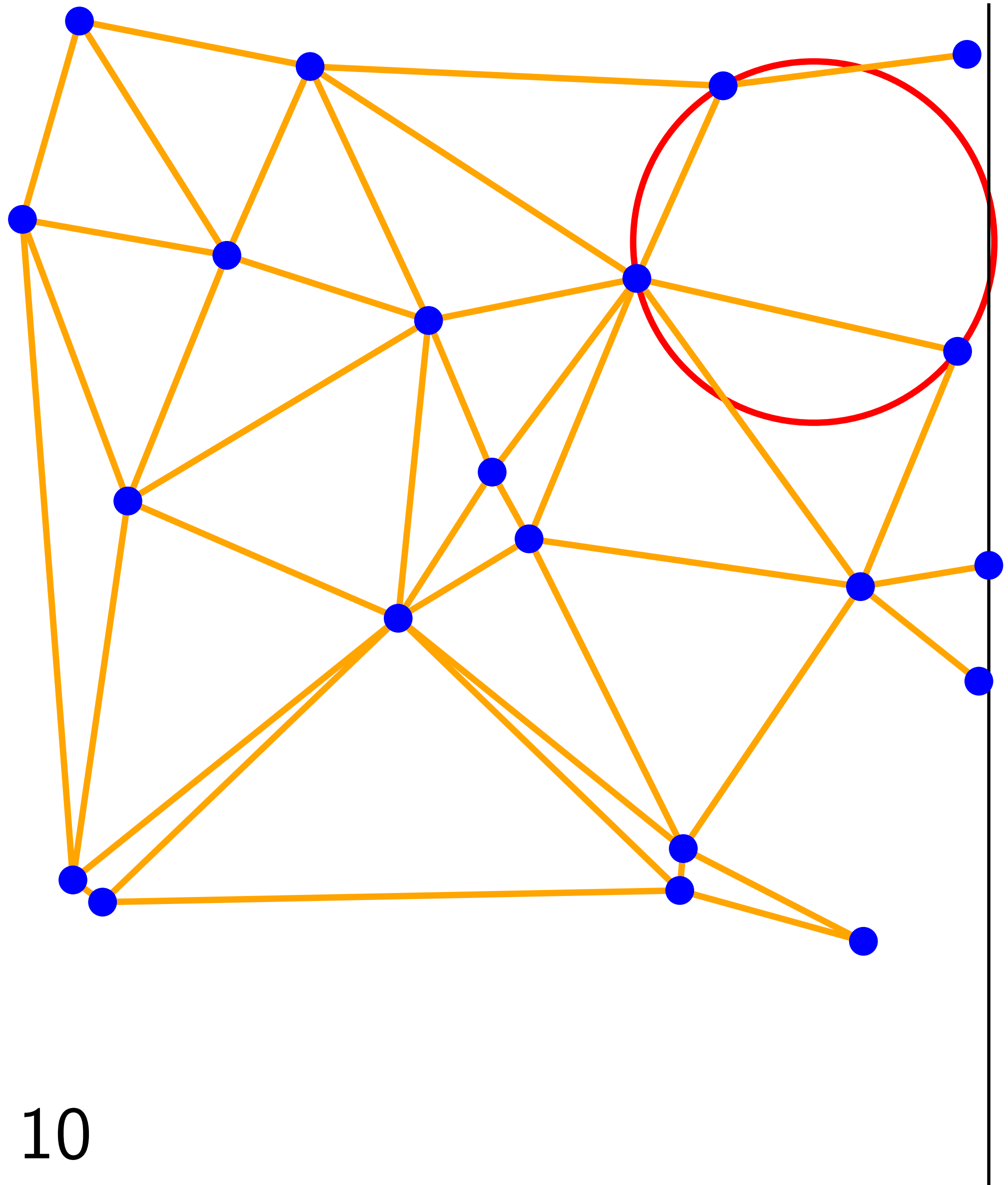
New point

Locate vertically

Create edge

Delaunay Triangulation: sweep-line algorithm

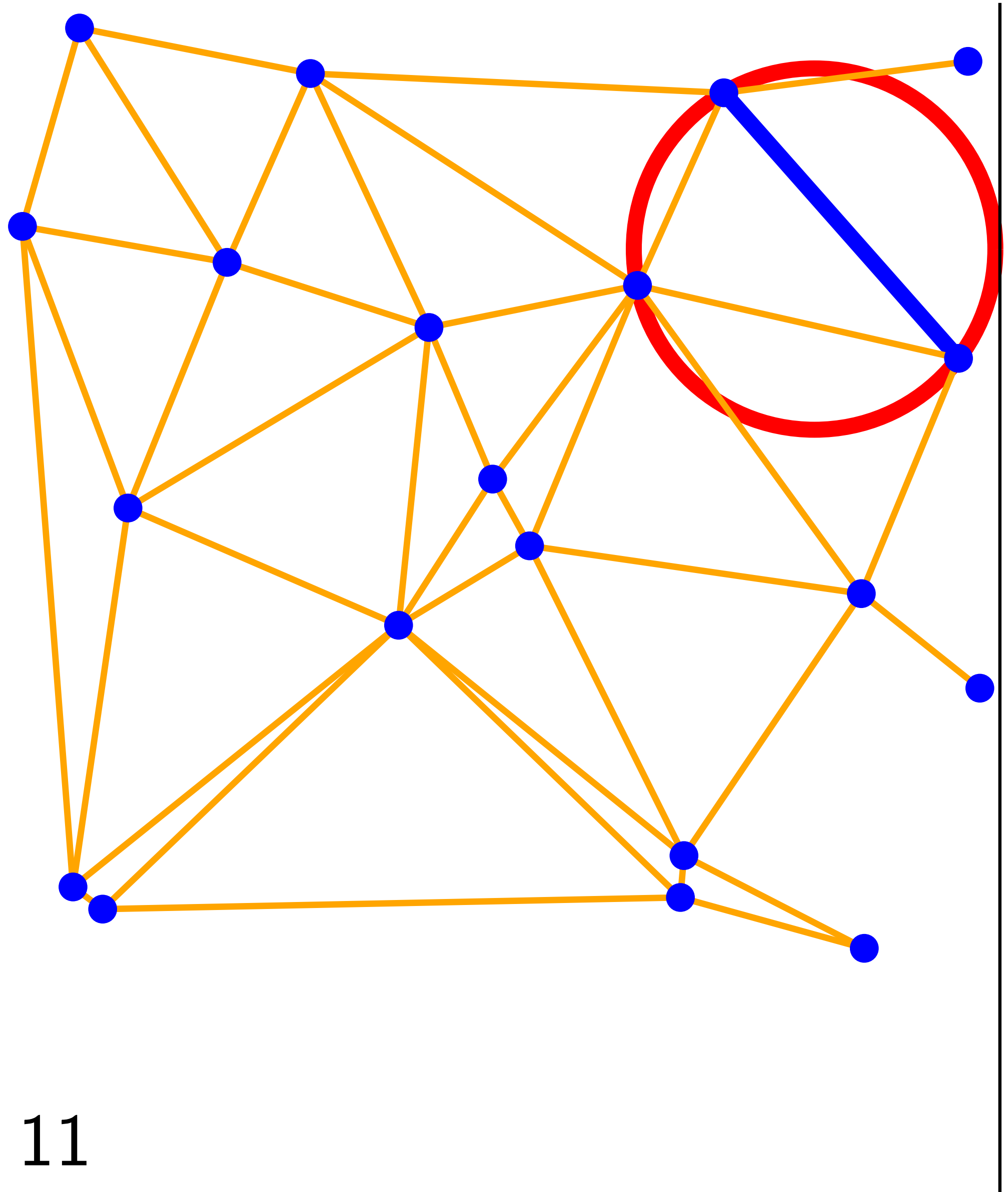
Discover the points from left to right



Closing a triangle ?

Delaunay Triangulation: sweep-line algorithm

Discover the points from left to right



Next circle event

Close triangle

Delaunay Triangulation: sweep-line algorithm

Complexity Number	Circle events processed	Point events
Triangulation		
List of events (x sorted)		
List of boundary edges (ccw sorted)		

Delaunay Triangulation: sweep-line algorithm

	Circle events	Point events
Complexity	processed	
Number	$2n$	n
Triangulation	create 2 triangles per event	create one edge per event
List of events (x sorted)	≤ 3 deletions ≤ 2 insertions per event	≤ 2 deletions ≤ 2 insertions per event
List of boundary edges (ccw sorted)	replace 2 edges by 1 per event	locate, then insert 2 edges per event

Delaunay Triangulation: sweep-line algorithm

Complexity / Number	Circle events processed $2n$	Point events n
Triangulation — $O(1)$ per operation	create 2 triangles per event	create one edge per event
List of events (x sorted) — $O(\log n)$ per operation	≤ 3 deletions ≤ 2 insertions per event	≤ 2 deletions ≤ 2 insertions per event
List of boundary edges (ccw sorted) — $O(\log n)$ per operation	replace 2 edges by 1 per event	locate, then insert 2 edges per event

Delaunay Triangulation: sweep-line algorithm

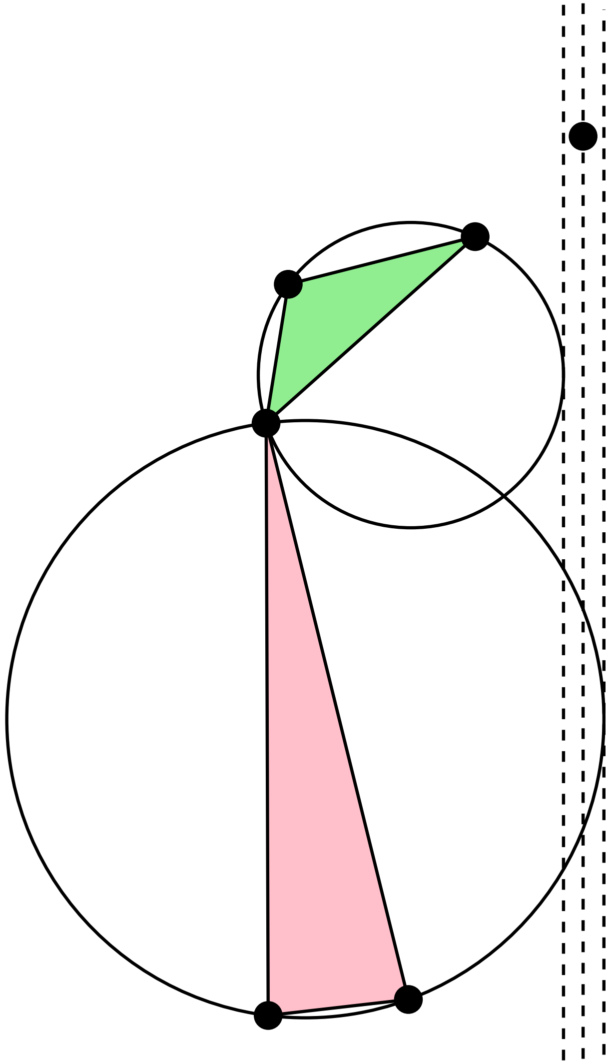
Complexity Number	Circle events processed $2n$	Point events n
Triangulation $O(1)$ per operation	create 2 triangles	create one edge
List of events (x sorted) $O(\log n)$ per operation	per event	per event
List of boundary edges (ccw sorted) $O(\log n)$ per operation	replace 2 edges by 1 per event	locate, then insert 2 edges per event

$$O(n \log n)$$

Delaunay Triangulation: predicates

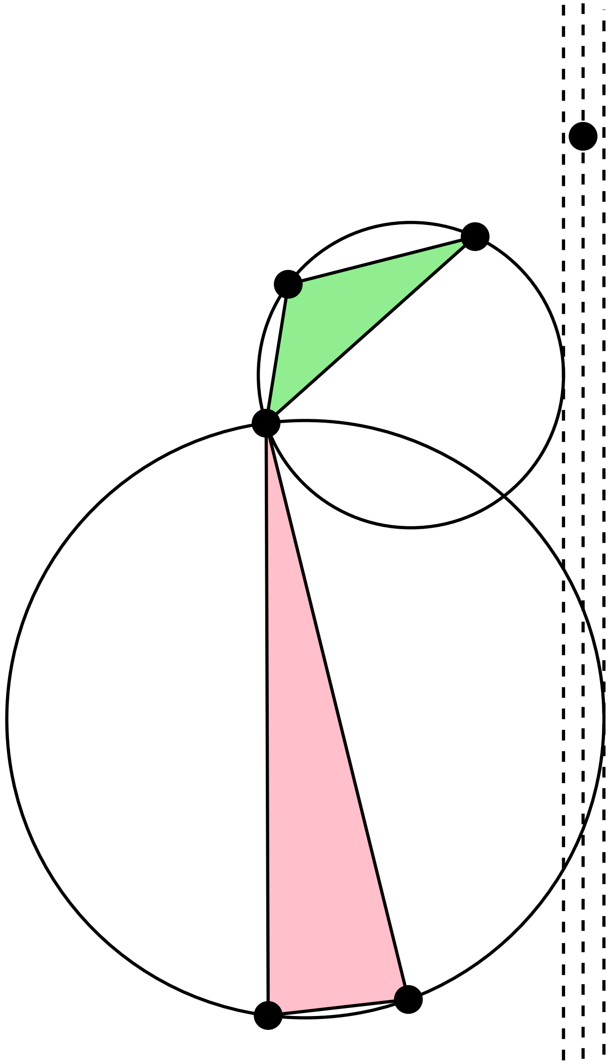
Delaunay Triangulation: predicates

x comparisons

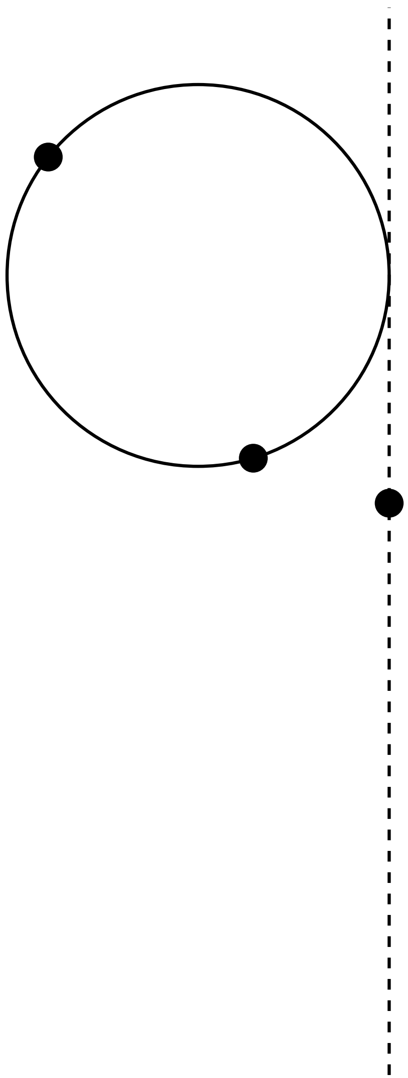


Delaunay Triangulation: predicates

x comparisons

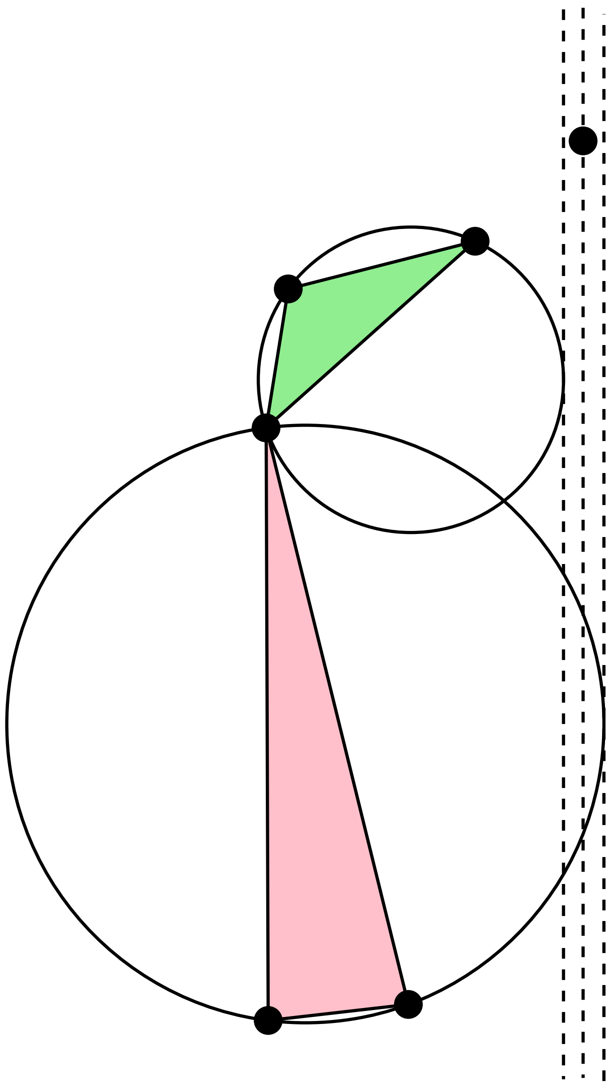


y comparisons

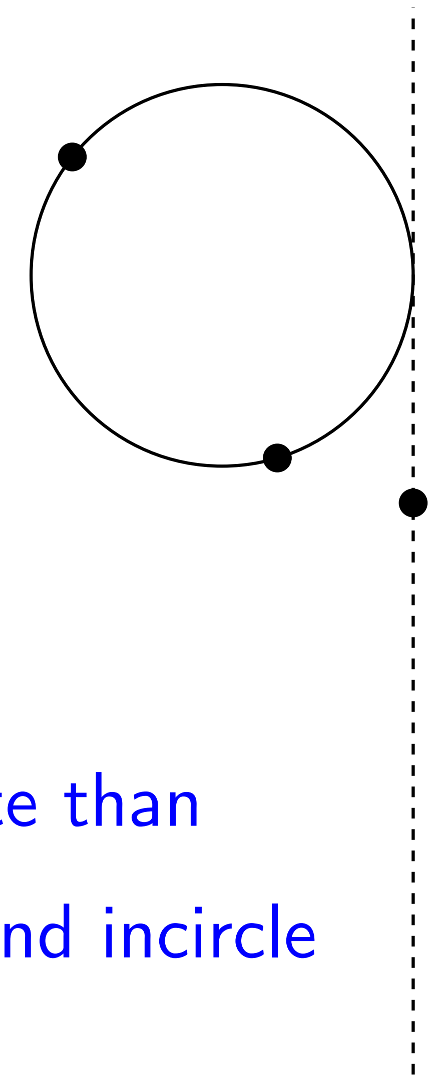


Delaunay Triangulation: predicates

x comparisons



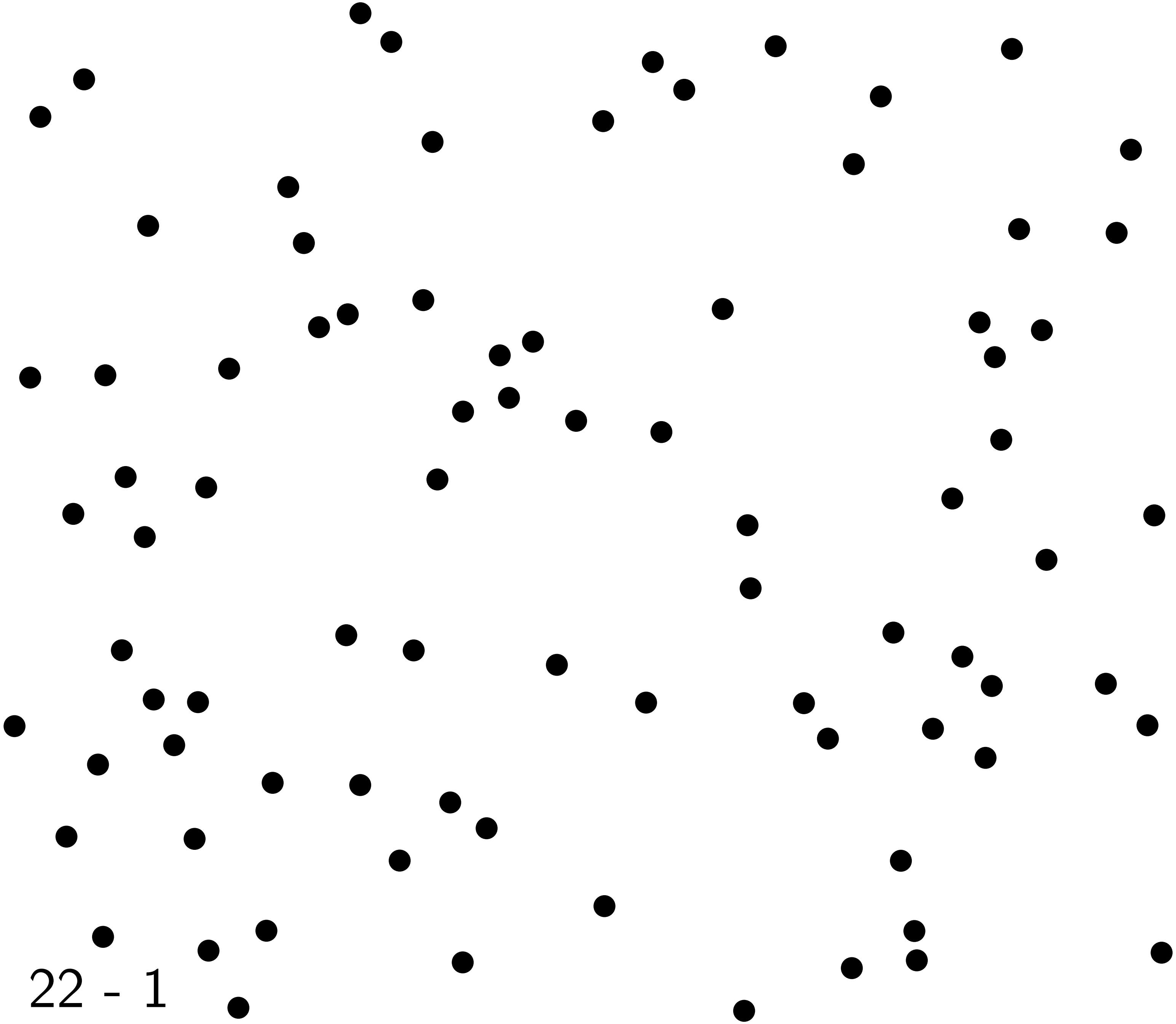
y comparisons



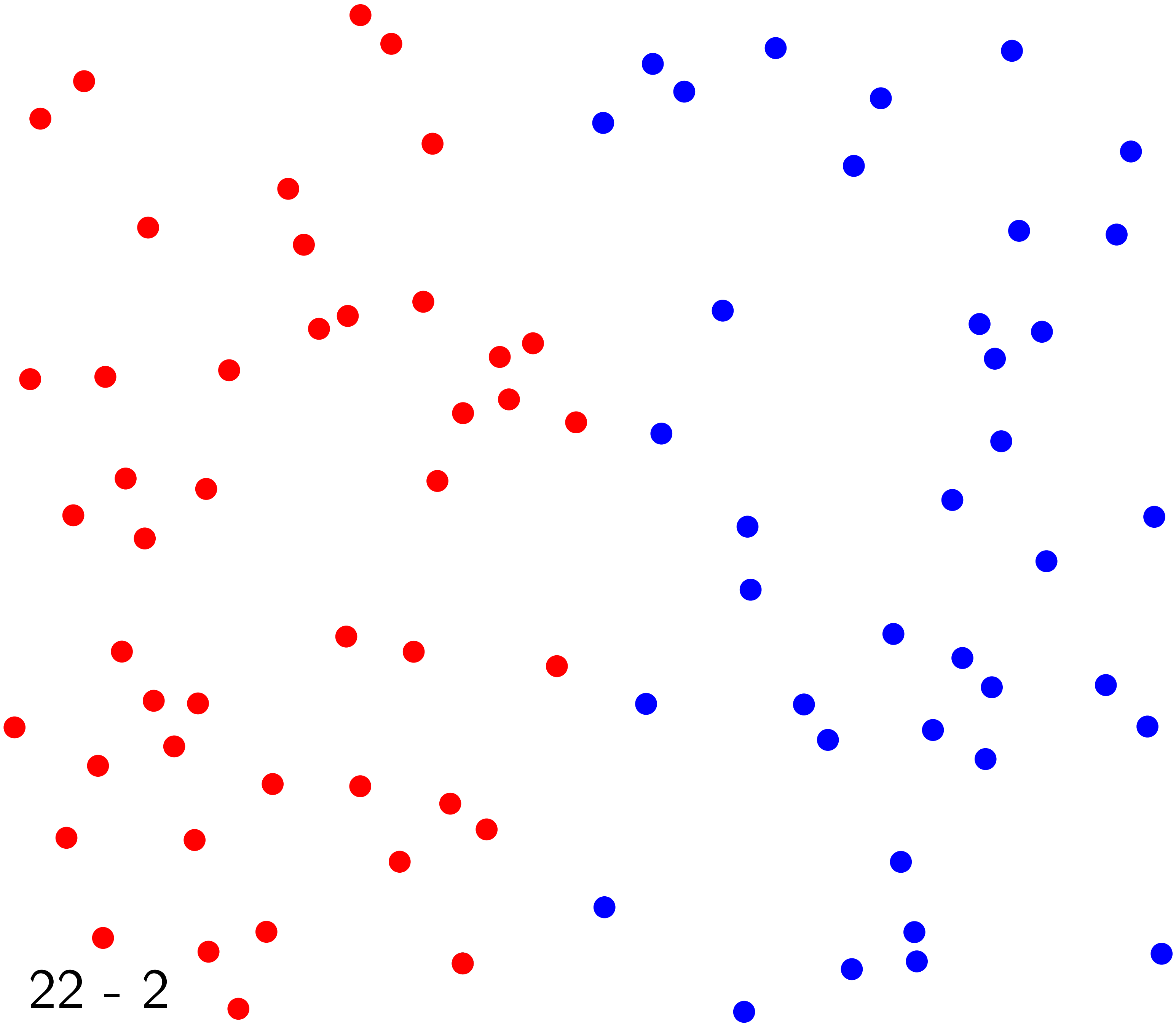
more intricate than

orientation and incircle

Delaunay Triangulation: divide & conquer (sketch)

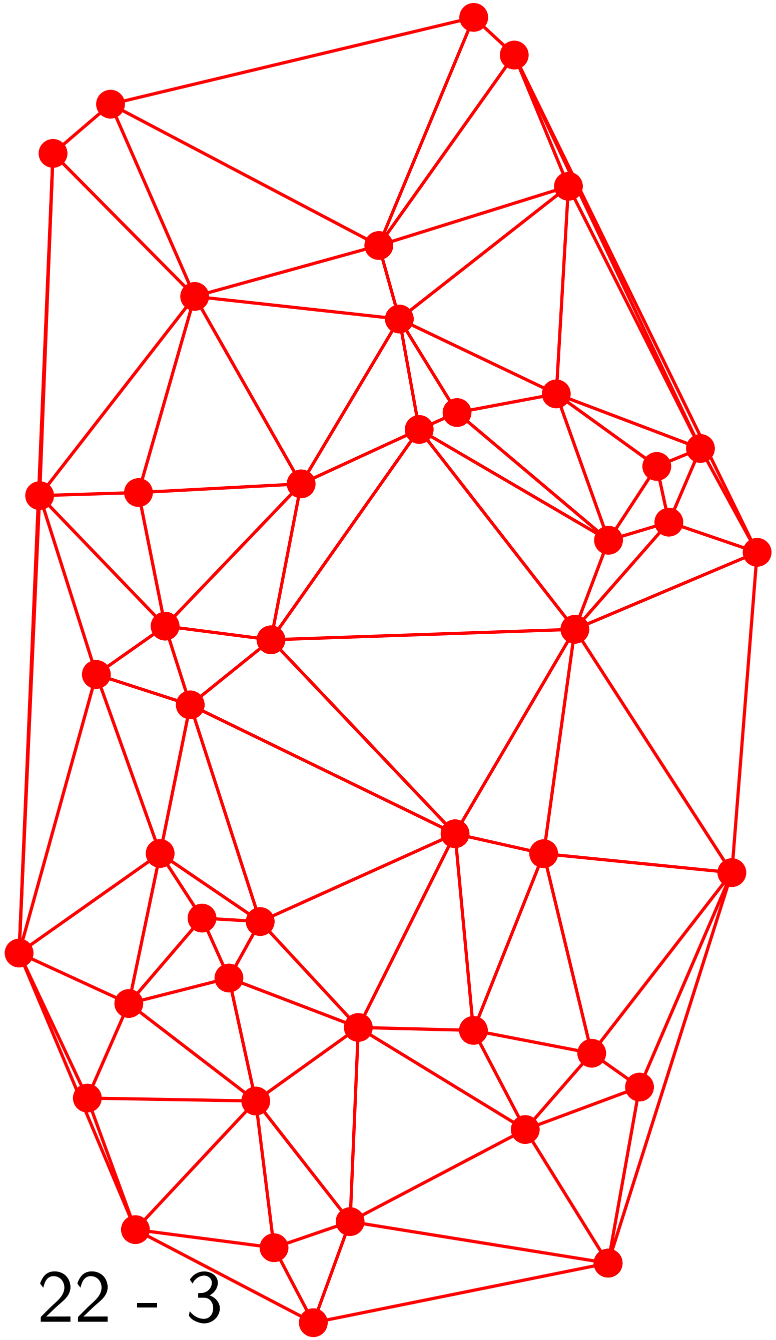


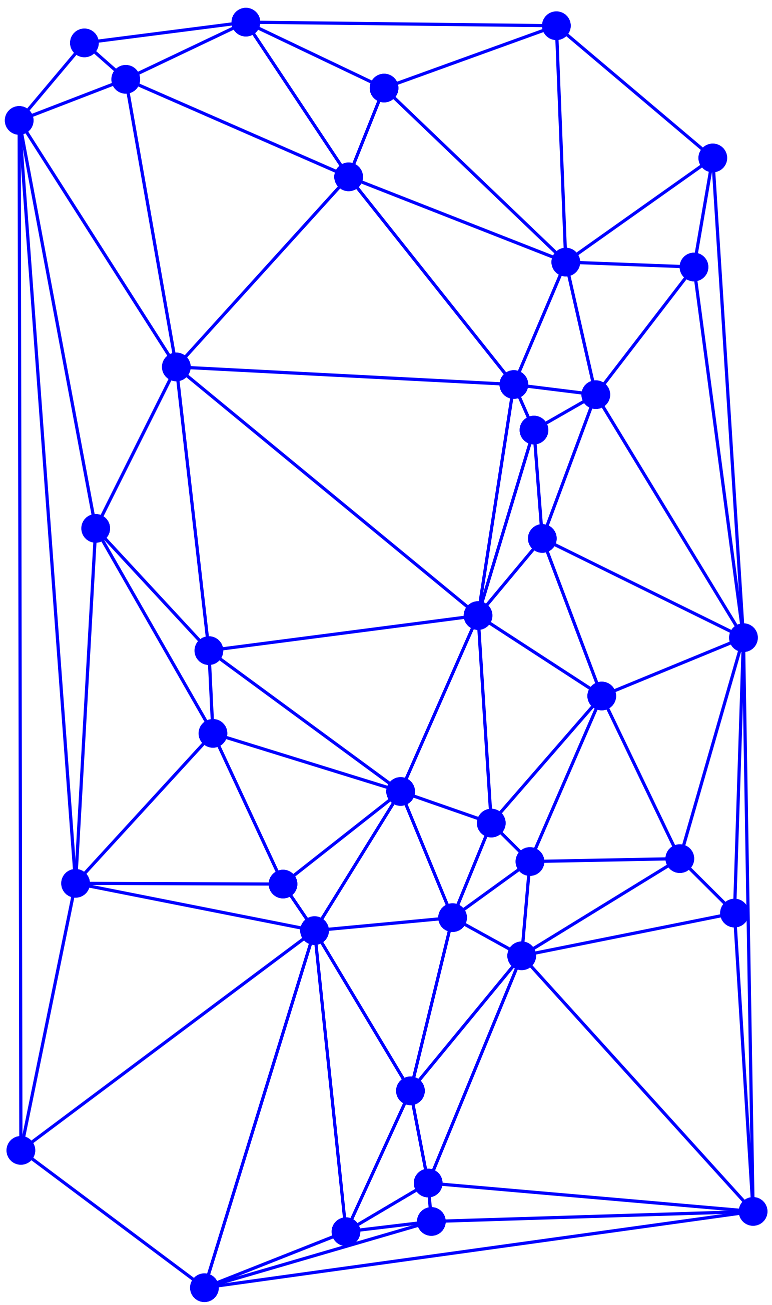
Delaunay Triangulation: divide & conquer (sketch)



Divide

Delaunay Triangulation: divide & conquer (sketch)

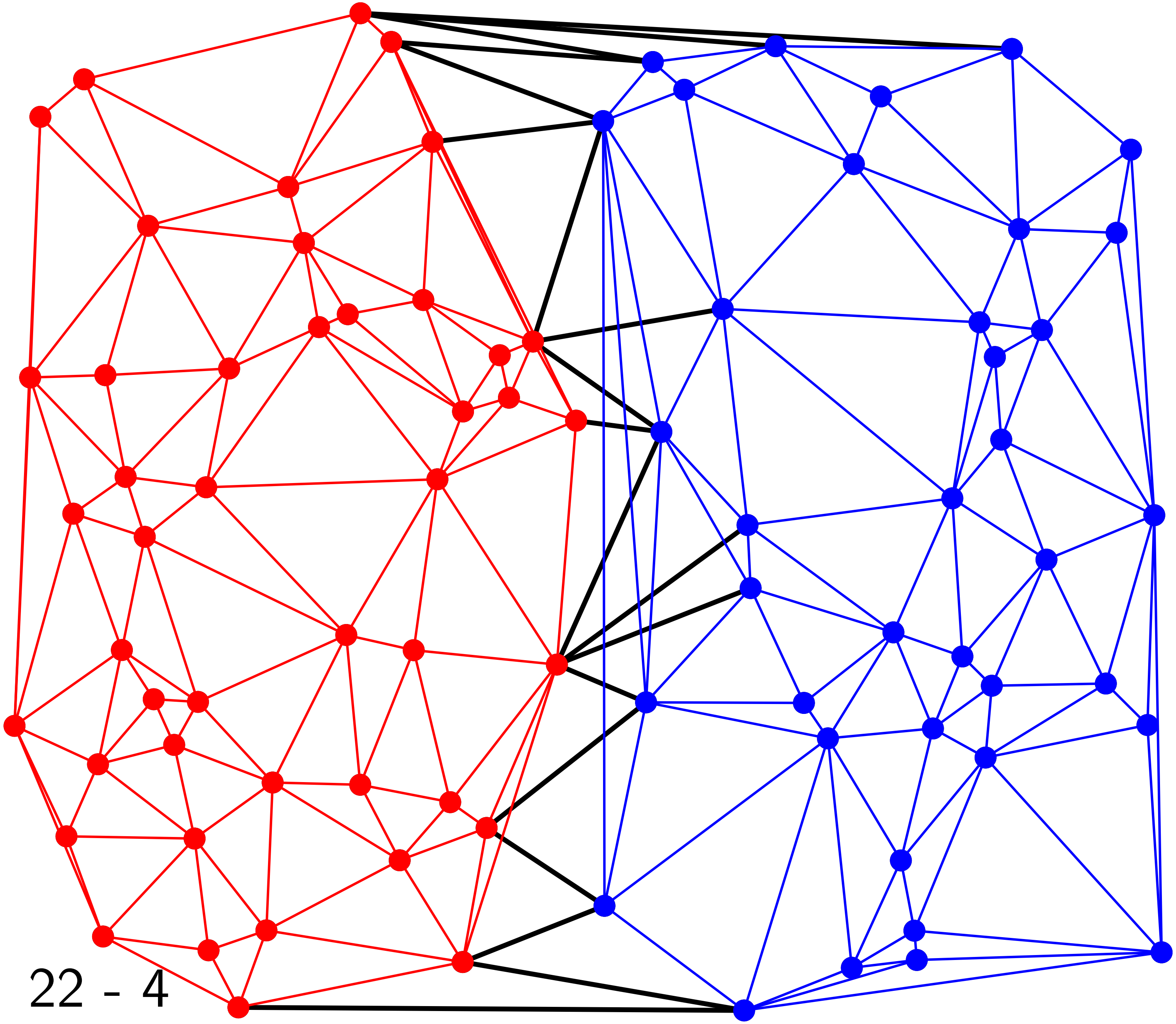




Divide

Recurse

Delaunay Triangulation: divide & conquer (sketch)

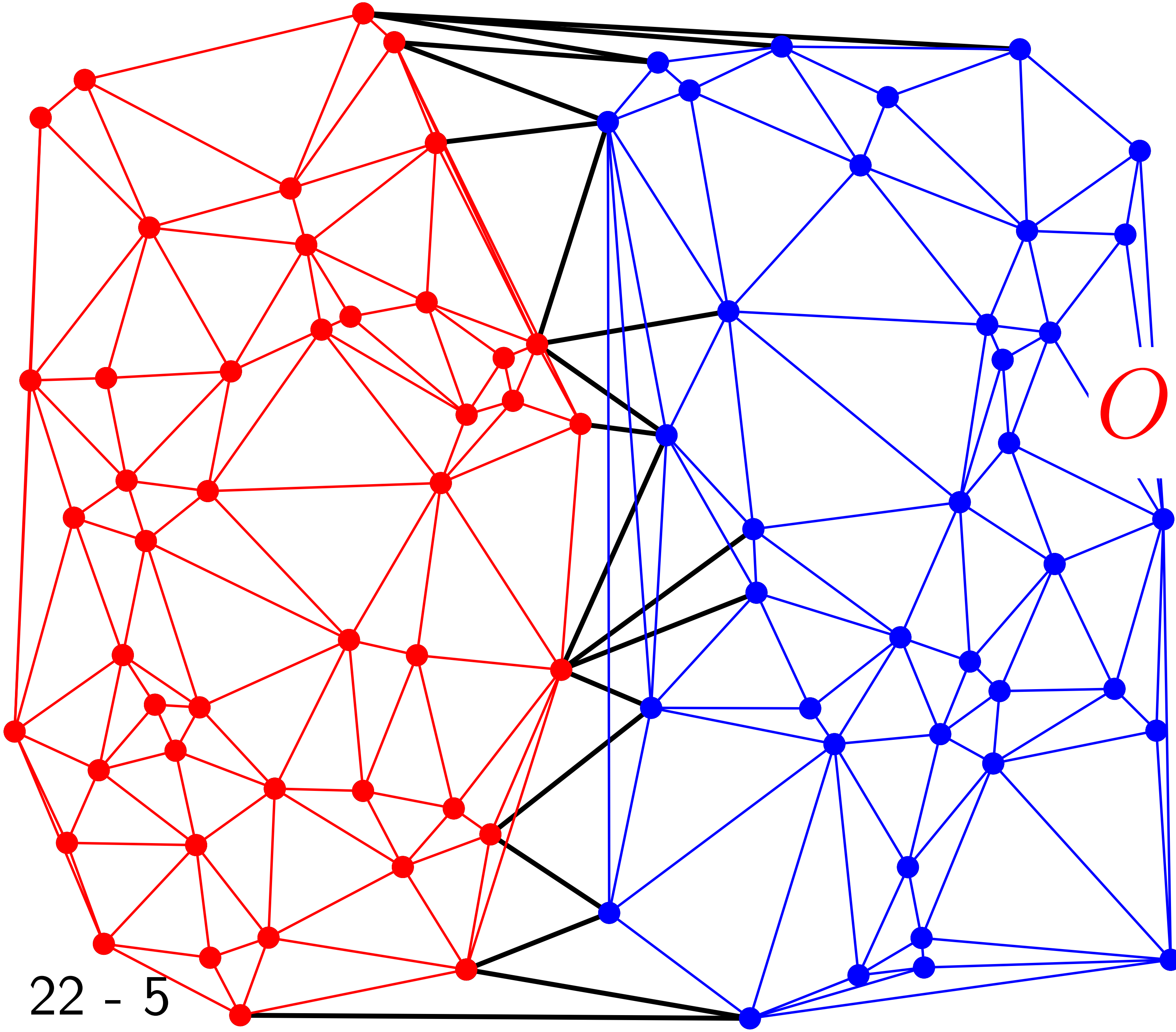


Divide

Recurse

Conquer

Delaunay Triangulation: divide & conquer (sketch)



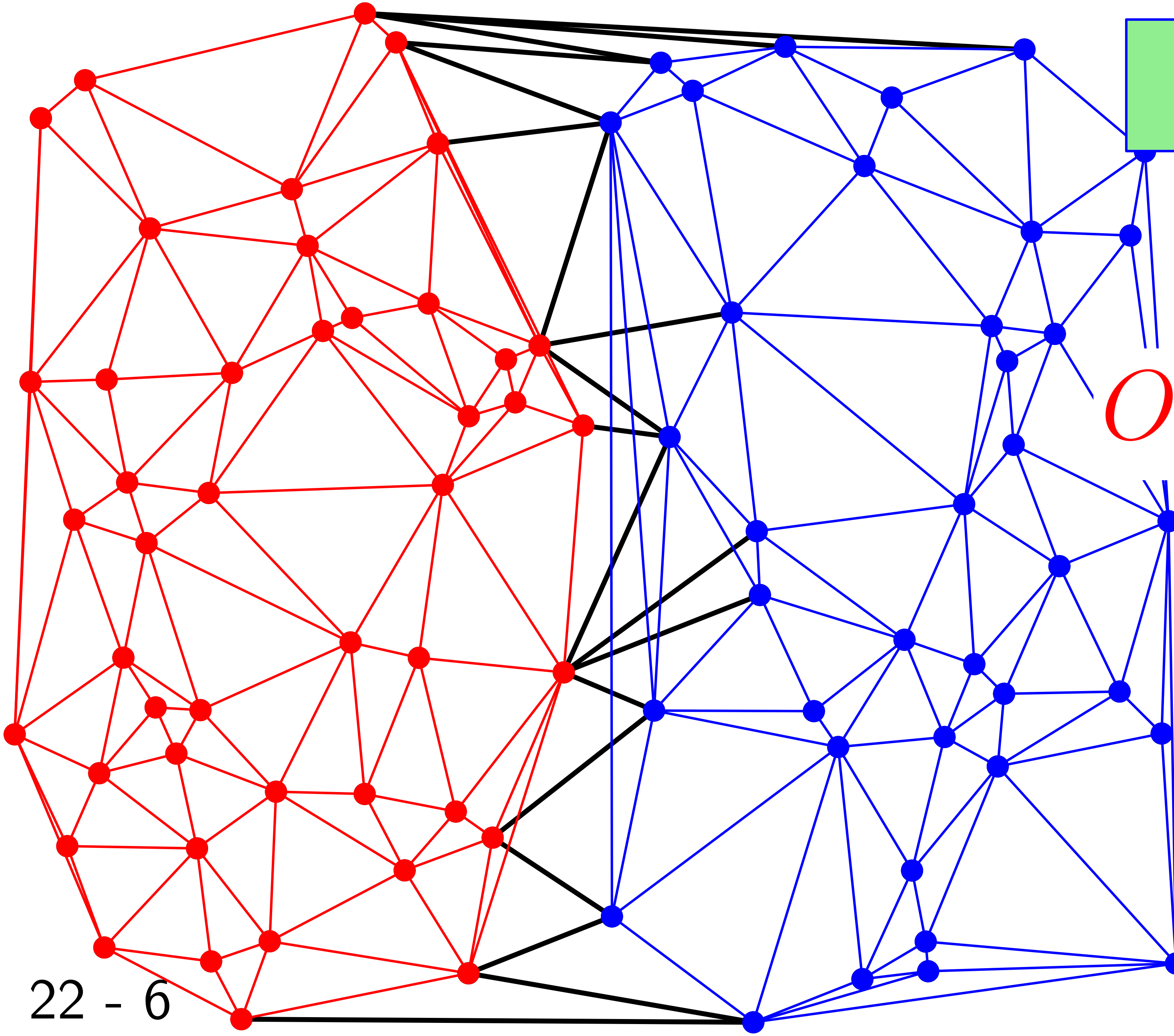
Divide

Recurse

Conquer

$O(n \log n)$

Delaunay Triangulation: divide & conquer (sketch)



Divide

Recurse

Conquer

$O(n \log n)$

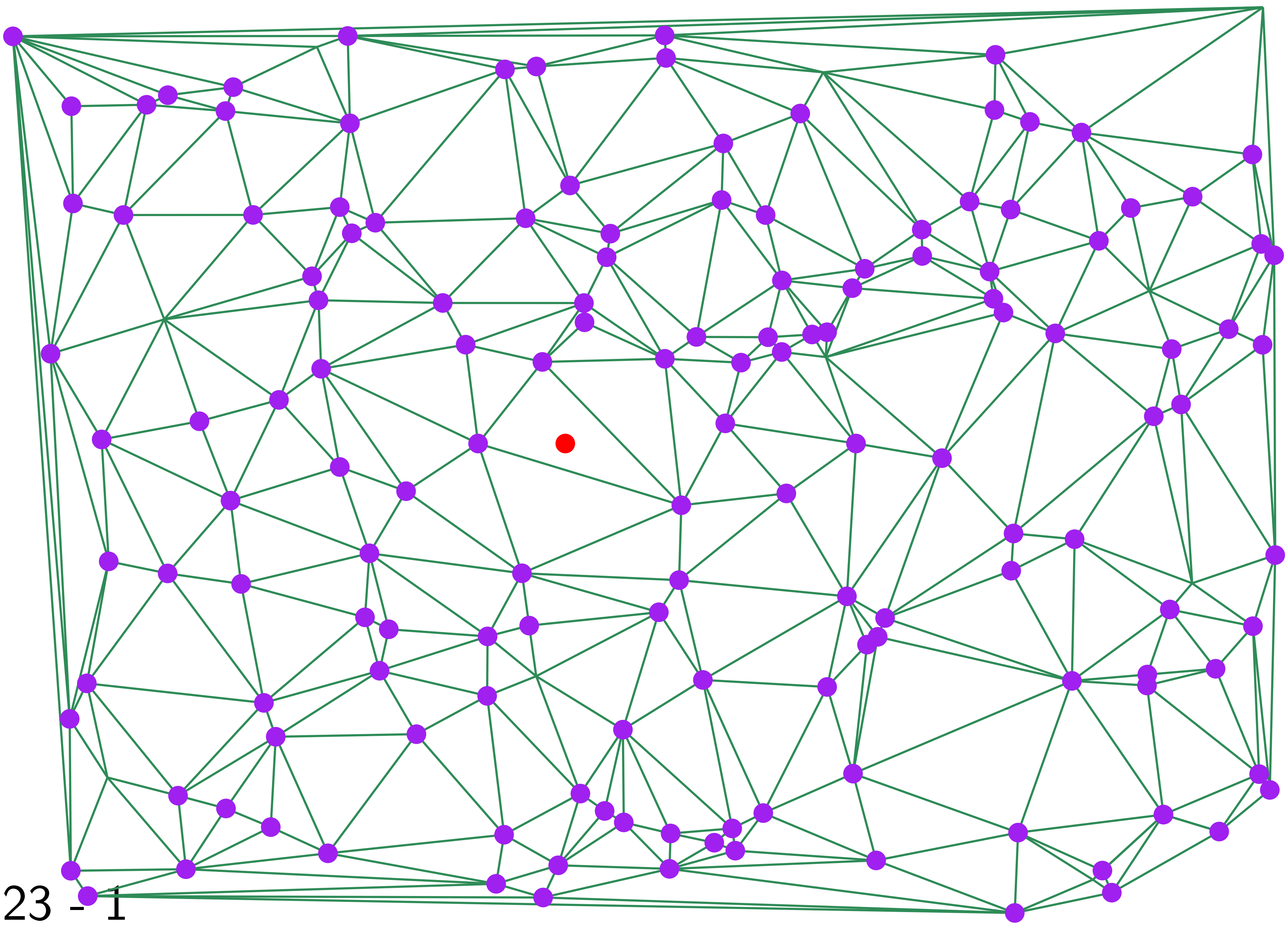
balanced

linear time

easier conquer

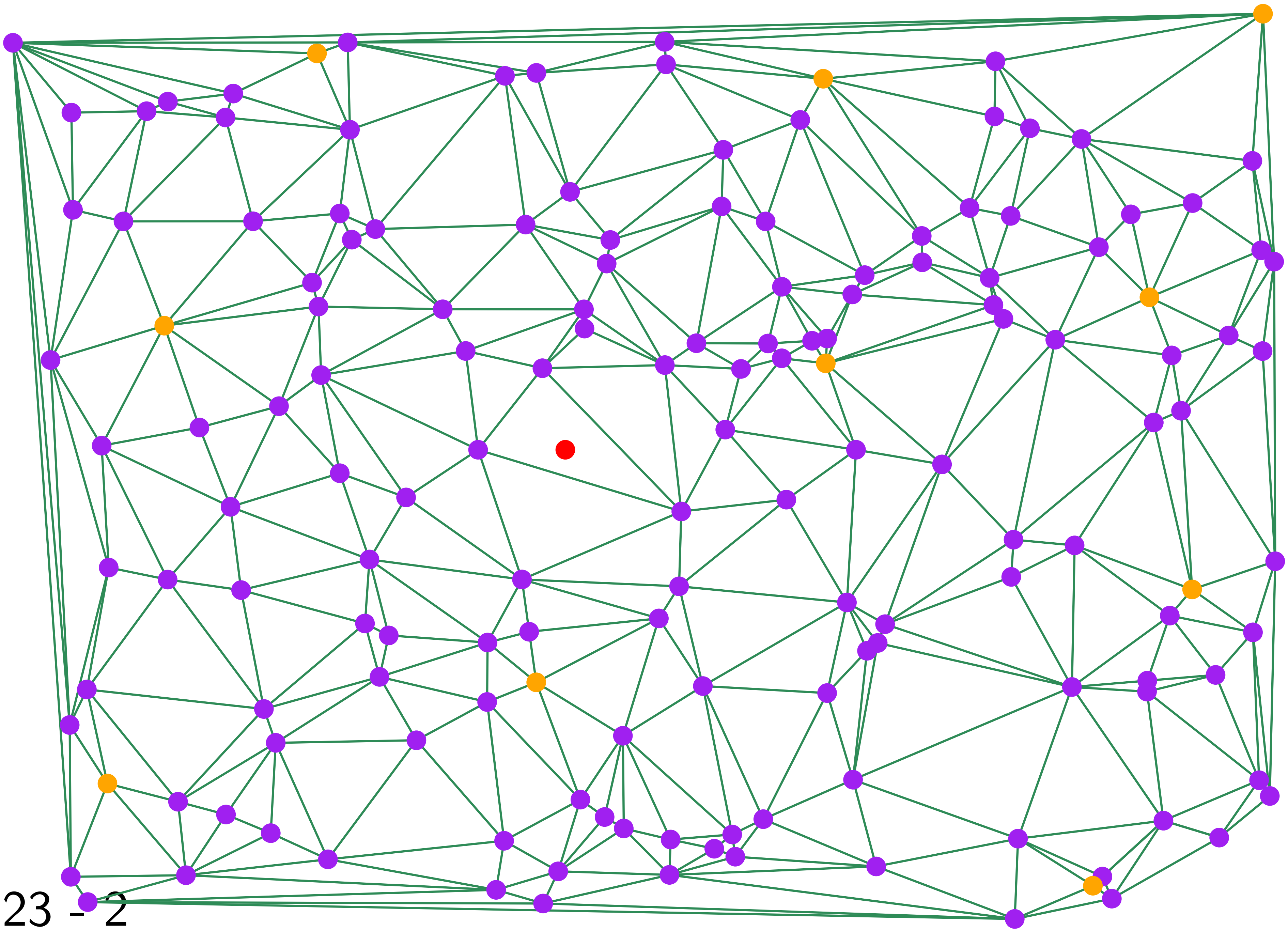
Jump and walk

Use randomness hypotheses



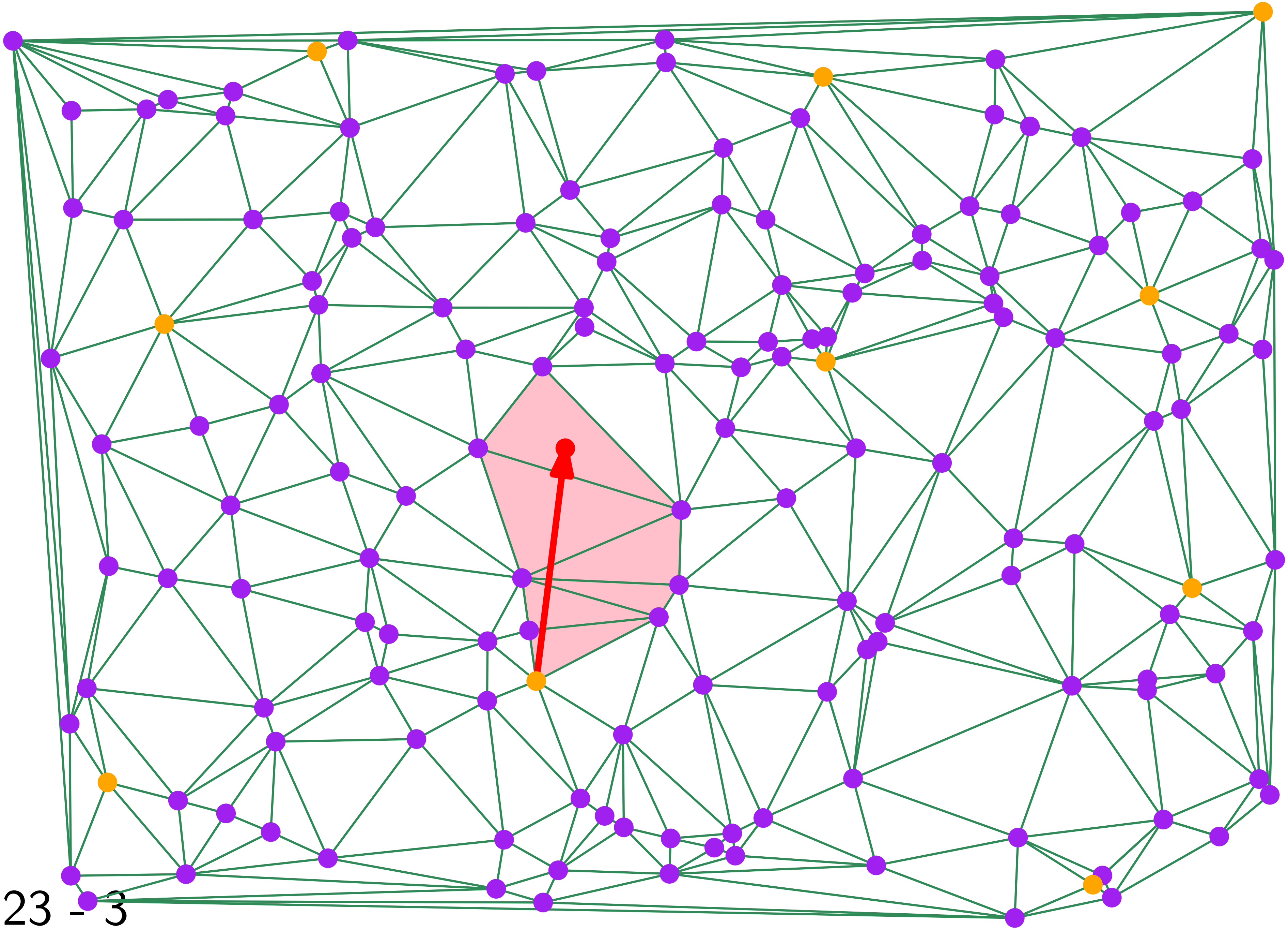
Jump and walk

Use randomness hypotheses



Jump and walk

Use randomness hypotheses



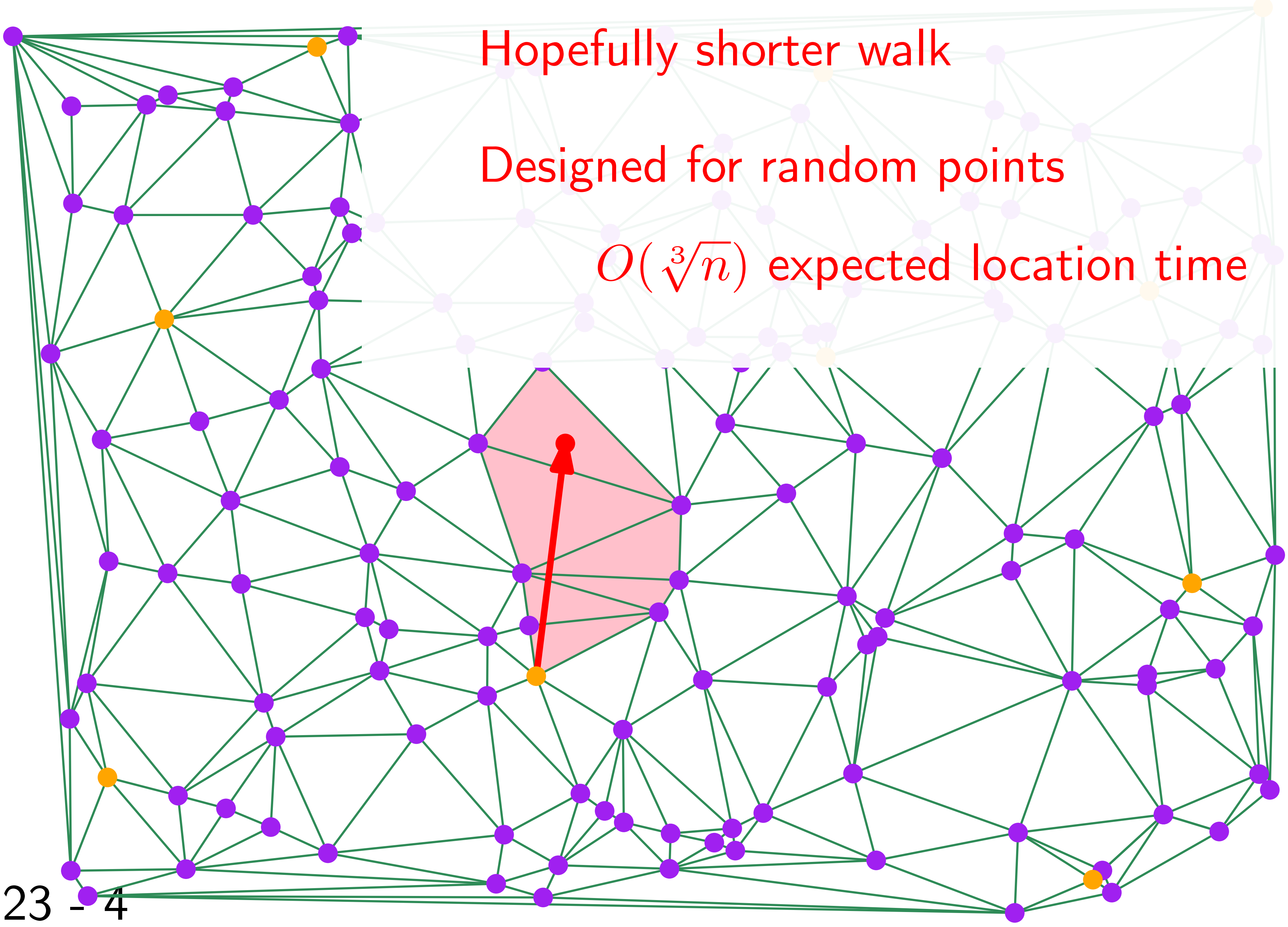
Jump and walk

Use randomness hypotheses

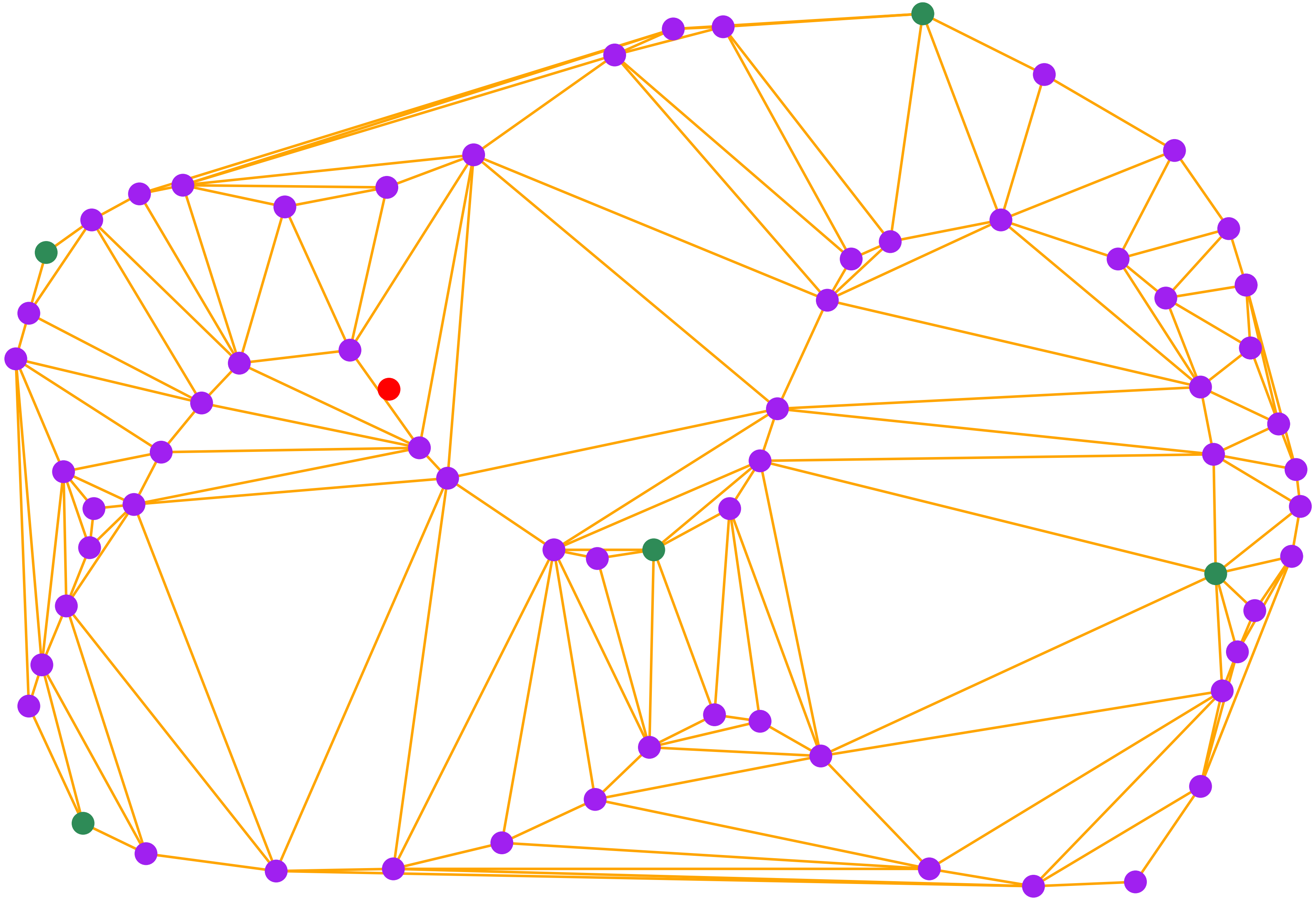
Hopefully shorter walk

Designed for random points

$O(\sqrt[3]{n})$ expected location time

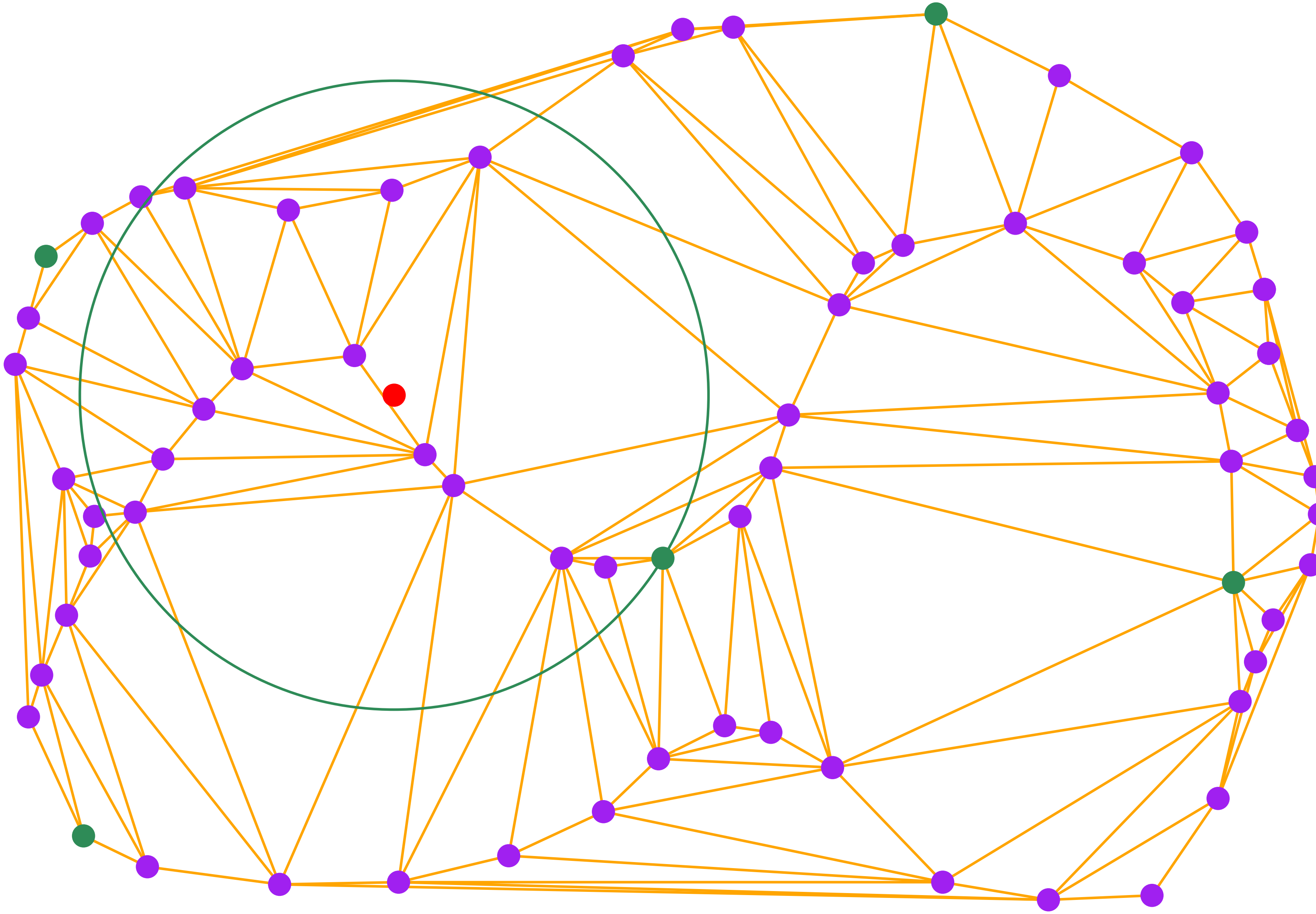


Jump and walk (no distribution hypothesis) Randomized



Jump and walk (no distribution hypothesis) Randomized

$\mathbb{E}\,[\sharp$ of in $] = \frac{n}{k}$

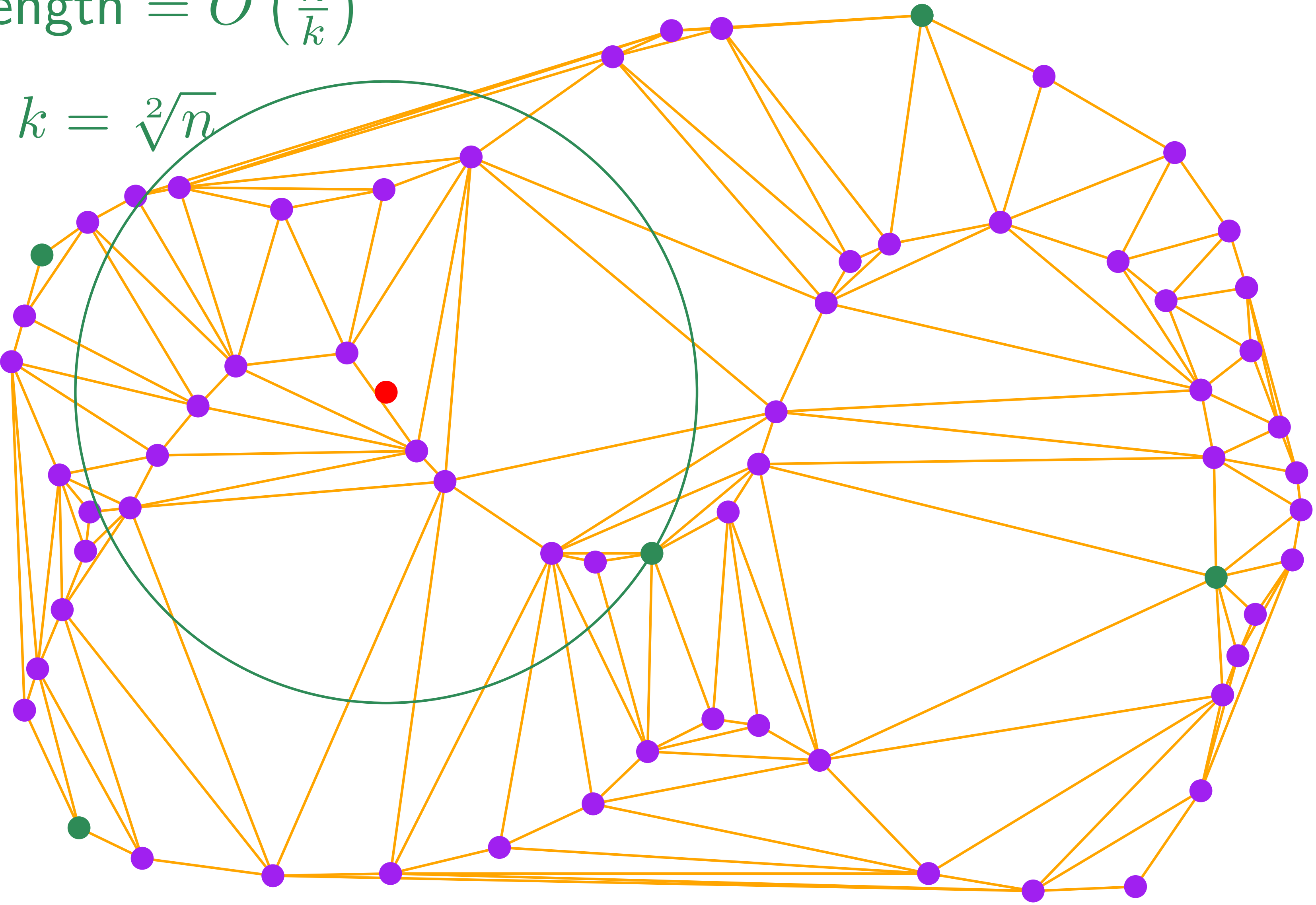


Jump and walk (no distribution hypothesis) Randomized

$\mathbb{E}\left[\sharp \text{ of } \bullet \text{ in } \bigcirc\right] = \frac{n}{k}$

Walk length $= O\left(\frac{n}{k}\right)$

choose $k = \sqrt[2]{n}$



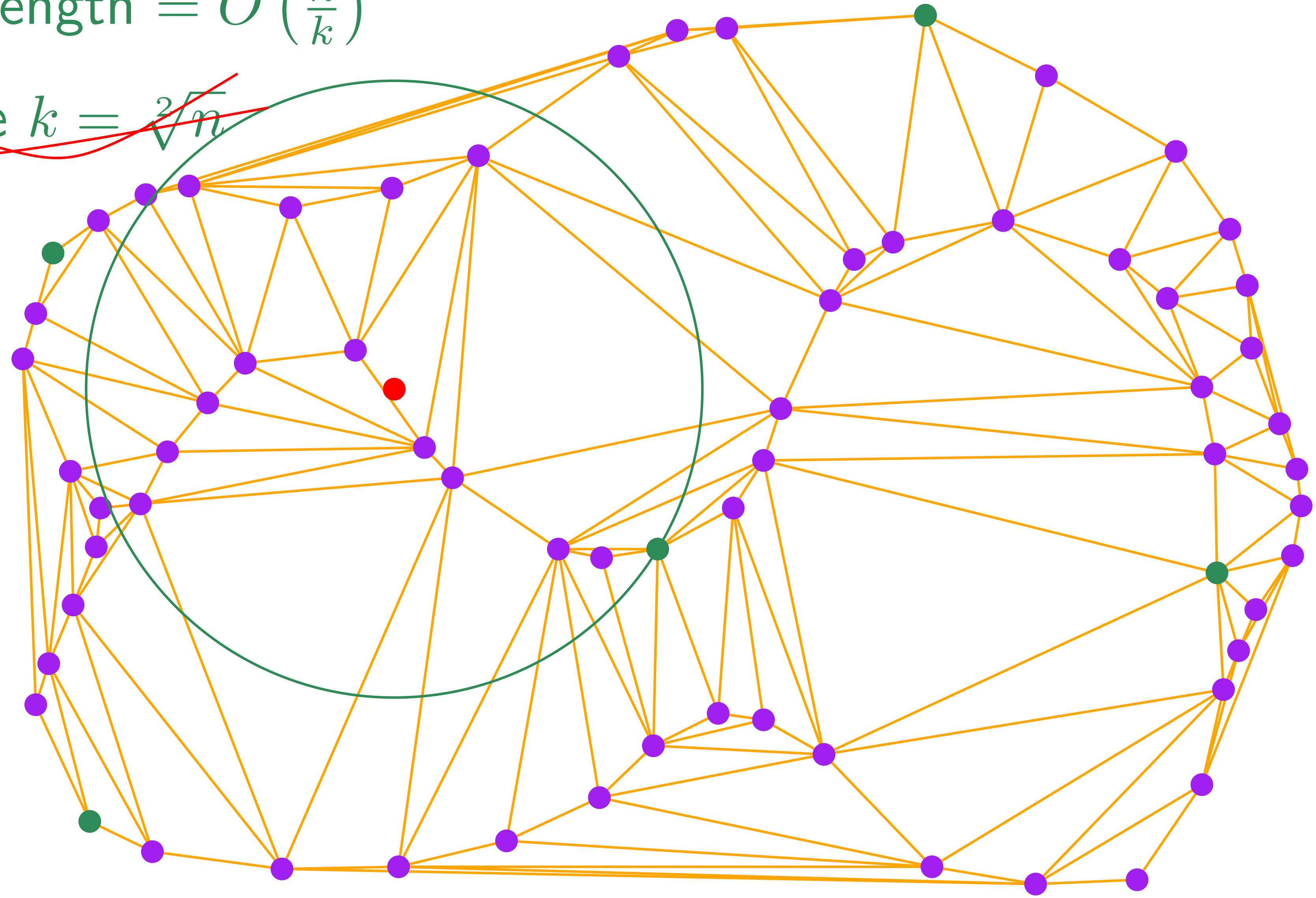
Jump and walk (no distribution hypothesis)

Randomized **Delaunay hierarchy**

$\mathbb{E}[\sharp$ of • in ⊙ $] = \frac{n}{k}$

Walk length $= O\left(\frac{n}{k}\right)$

~~choose $k = \sqrt[2]{n}$~~



Jump and walk (no distribution hypothesis)

Randomized

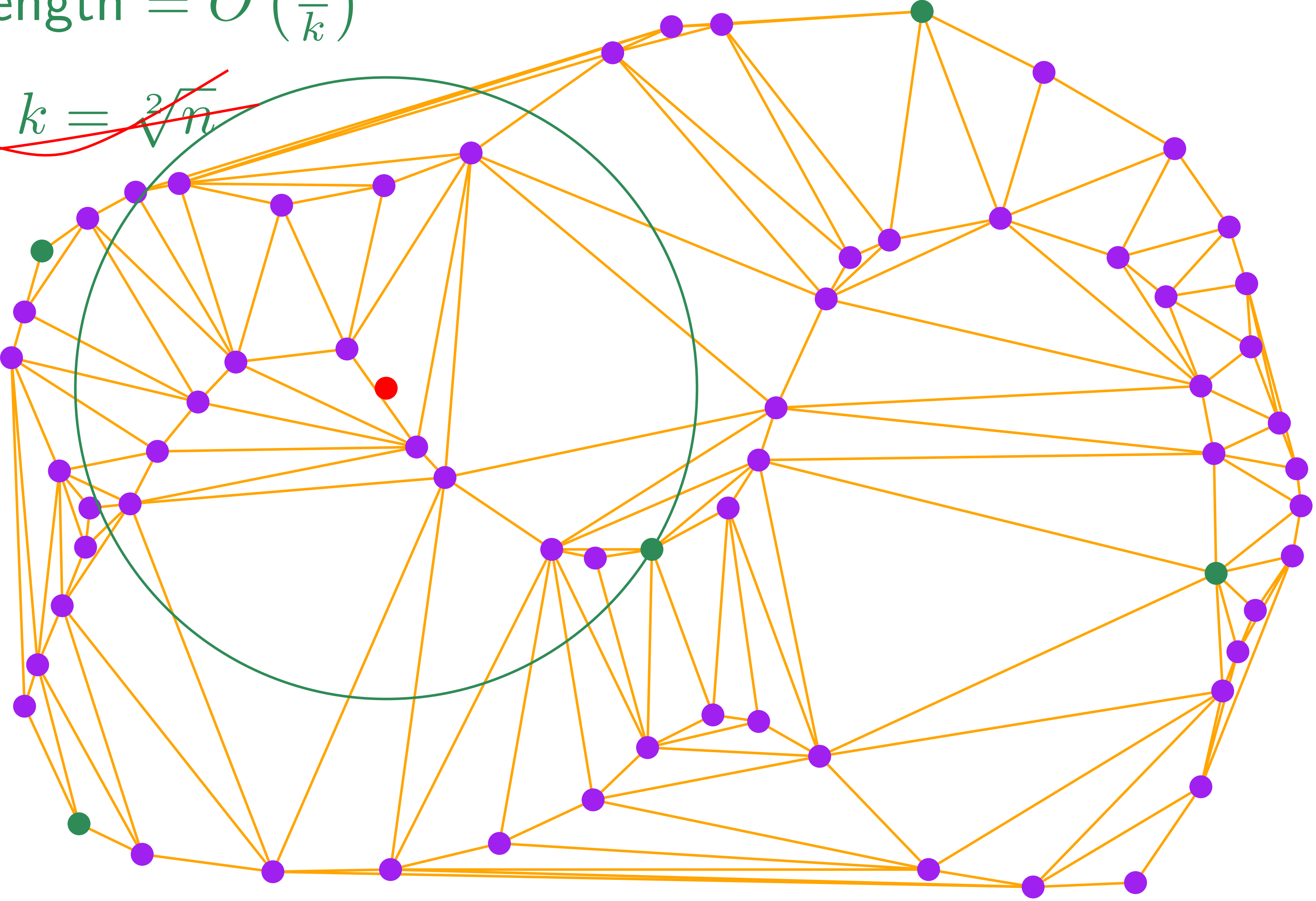
Delaunay hierarchy

$\mathbb{E}\left[\sharp \text{ of } \bullet \text{ in } \bigcirc \right] = \frac{n}{k}$

$\frac{n}{k_1}$

Walk length $= O\left(\frac{n}{k}\right)$

choose $k = \sqrt[2]{n}$



Jump and walk (no distribution hypothesis)

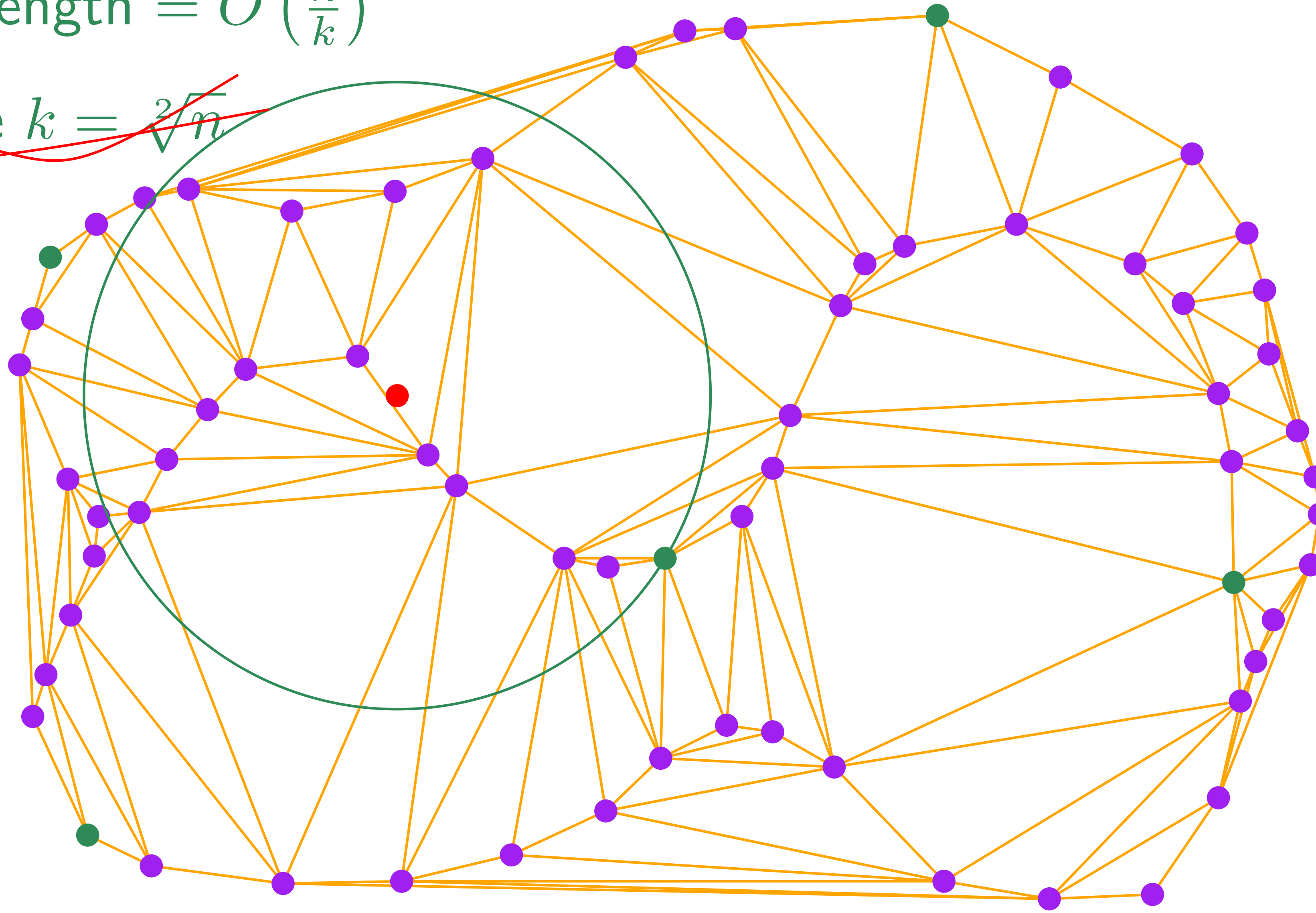
Randomized
Delaunay hierarchy

$\mathbb{E}\left[\sharp \text{ of } \bullet \text{ in } \odot\right] = \frac{n}{k}$

$\frac{n}{k_1} + \frac{k_1}{k_2}$

Walk length $= O\left(\frac{n}{k}\right)$

~~choose $k = \sqrt[2]{n}$~~



Jump and walk (no distribution hypothesis)

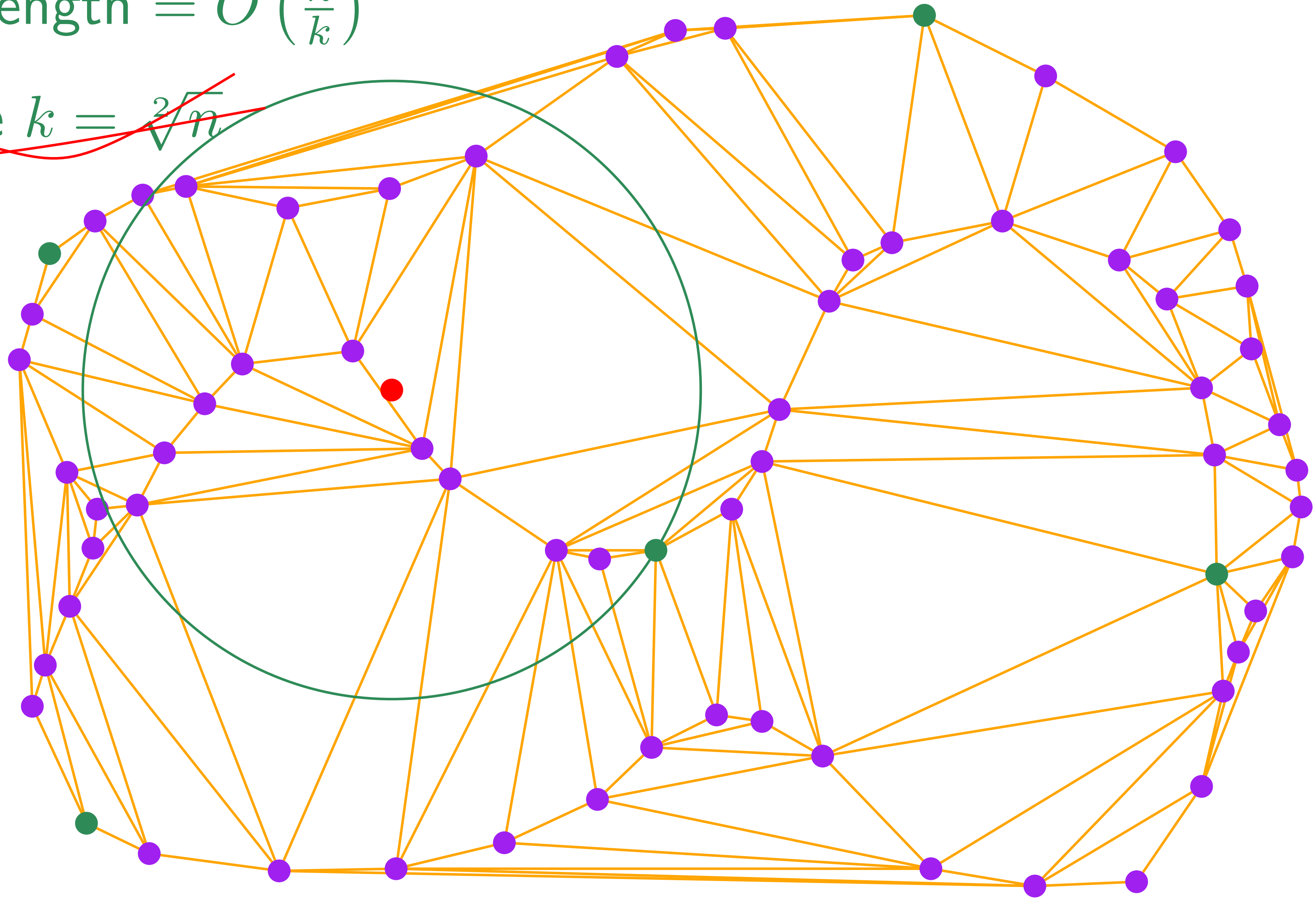
Randomized Delaunay hierarchy

$\mathbb{E}\left[\sharp \text{ of } \bullet \text{ in } \odot\right] = \frac{n}{k}$

$\frac{n}{k_1} + \frac{k_1}{k_2} + \frac{k_2}{k_3} + \ldots$

Walk length $= O\left(\frac{n}{k}\right)$

~~choose $k = \sqrt[2]{n}$~~



Jump and walk (no distribution hypothesis) Randomized Delaunay hierarchy

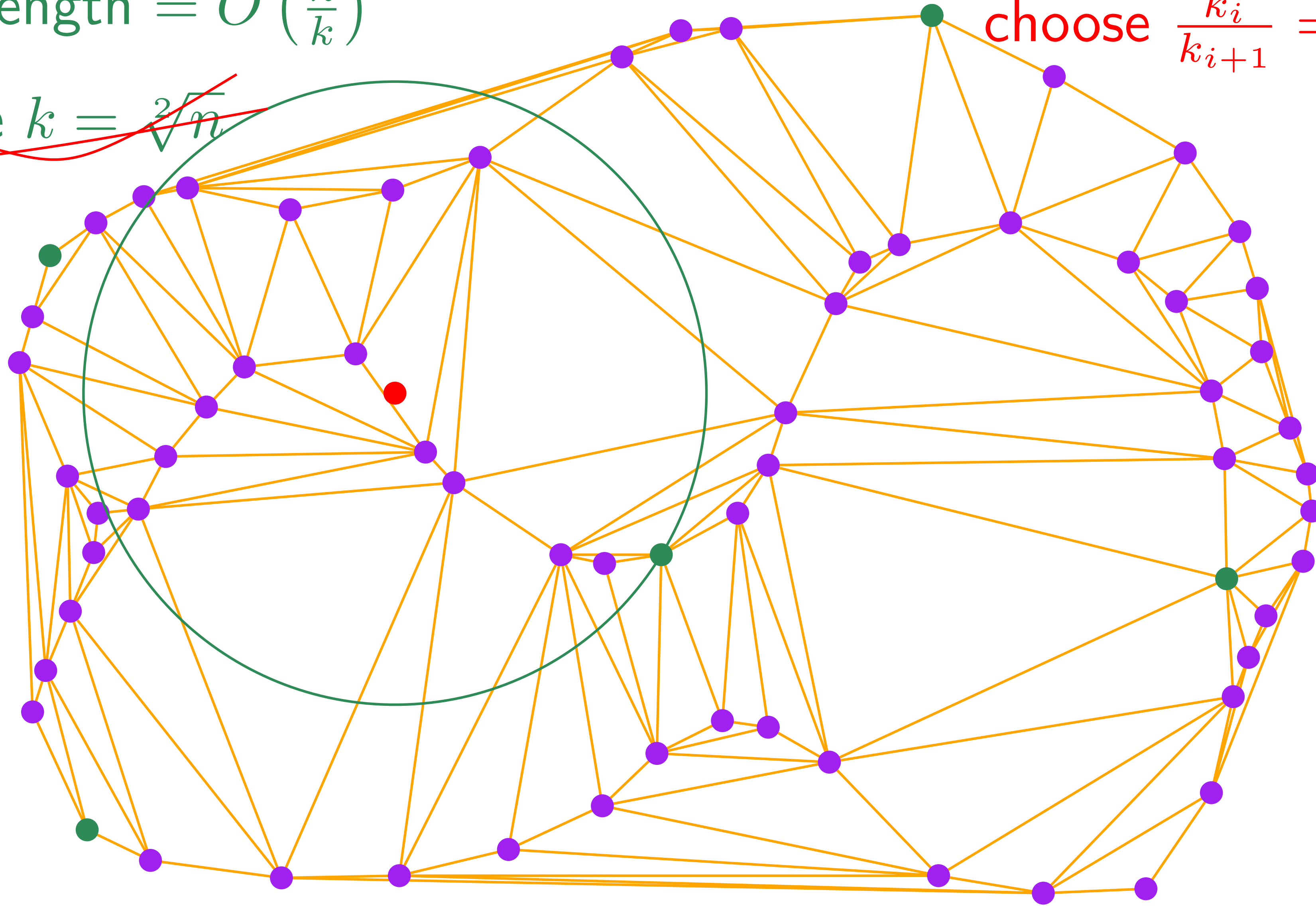
$\mathbb{E}\left[\sharp \text{ of } \bullet \text{ in } \odot\right] = \frac{n}{k}$

$\frac{n}{k_1} + \frac{k_1}{k_2} + \frac{k_2}{k_3} + \cdots$

Walk length $= O\left(\frac{n}{k}\right)$

choose $\frac{k_i}{k_{i+1}} = \alpha$

~~choose $k = \sqrt[2]{n}$~~



Jump and walk (no distribution hypothesis) Randomized

$\mathbb{E}\left[\sharp \text{ of } \bullet \text{ in } \odot\right] = \frac{n}{k}$

Delaunay hierarchy

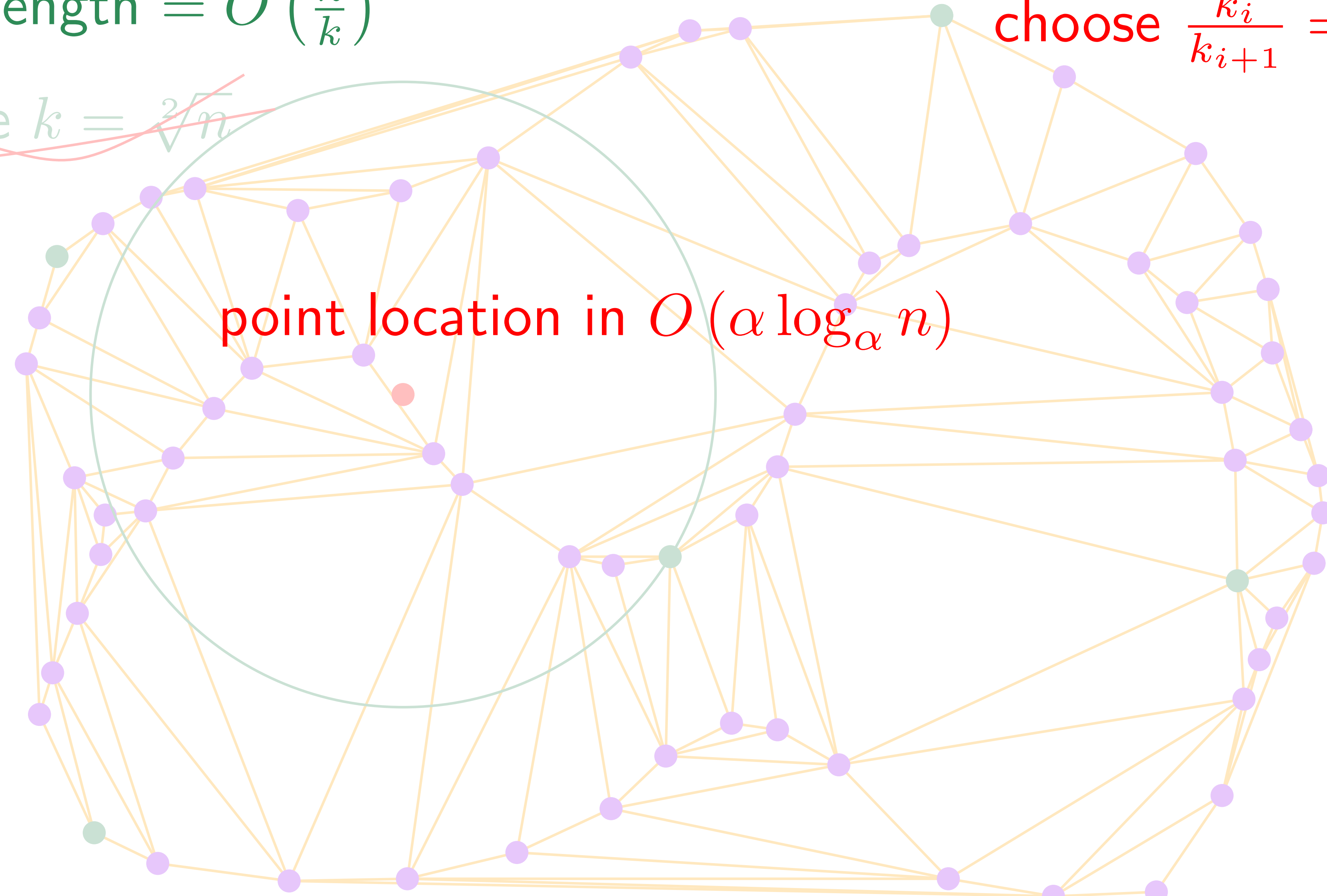
$\frac{n}{k_1} + \frac{k_1}{k_2} + \frac{k_2}{k_3} + \dots$

Walk length $= O\left(\frac{n}{k}\right)$

choose $\frac{k_i}{k_{i+1}} = \alpha$

choose $k = \sqrt[2]{n}$

point location in $O\left(\alpha \log_\alpha n\right)$



Jump and walk (no distribution hypothesis) Randomized

Delaunay hierarchy

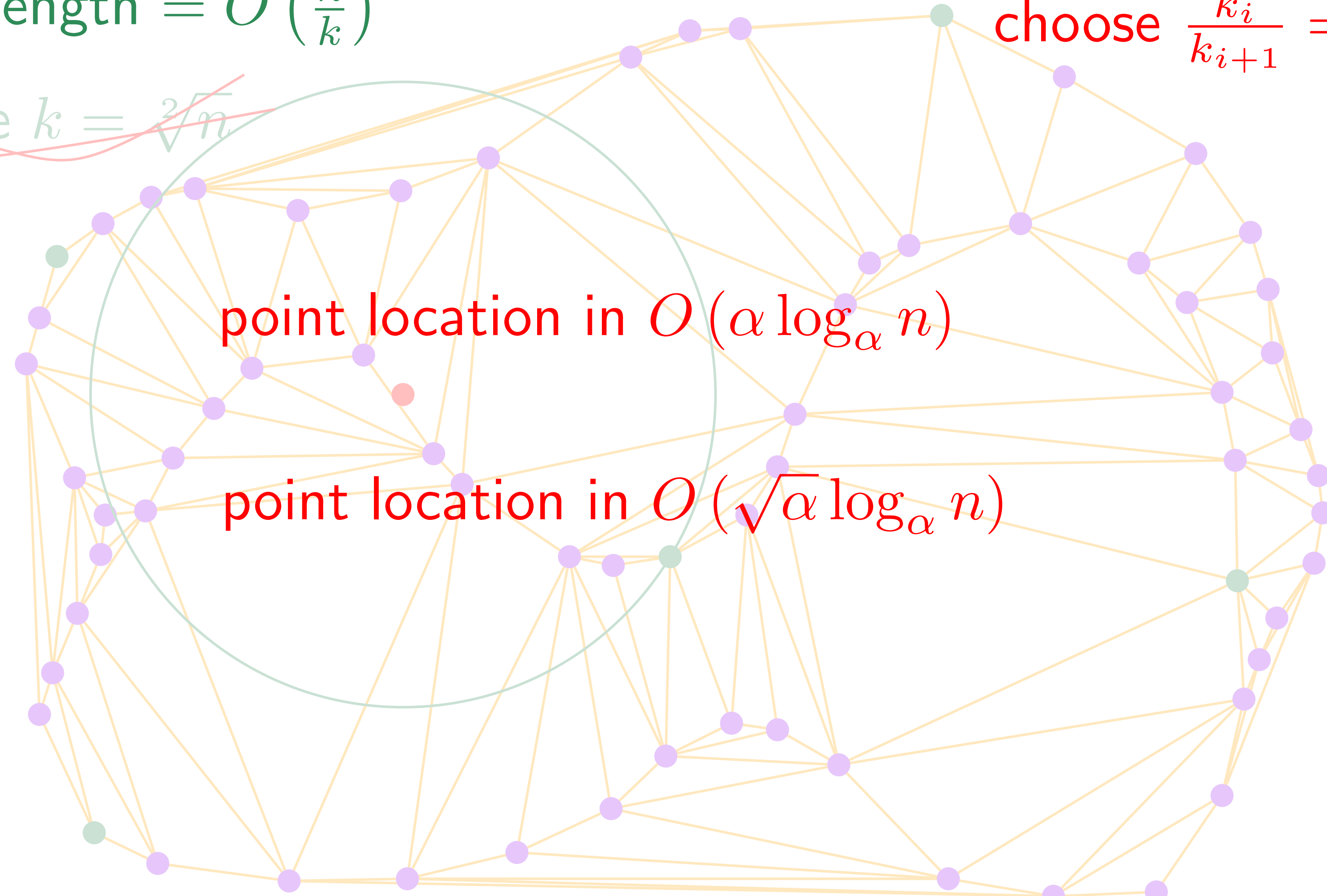
$\mathbb{E}\left[\sharp \text{ of } \bullet \text{ in } \odot\right] = \frac{n}{k}$

$\frac{n}{k_1} + \frac{k_1}{k_2} + \frac{k_2}{k_3} + \ldots$

Walk length $= O\left(\frac{n}{k}\right)$

choose $\frac{k_i}{k_{i+1}} = \alpha$

choose $k = \sqrt[2]{n}$

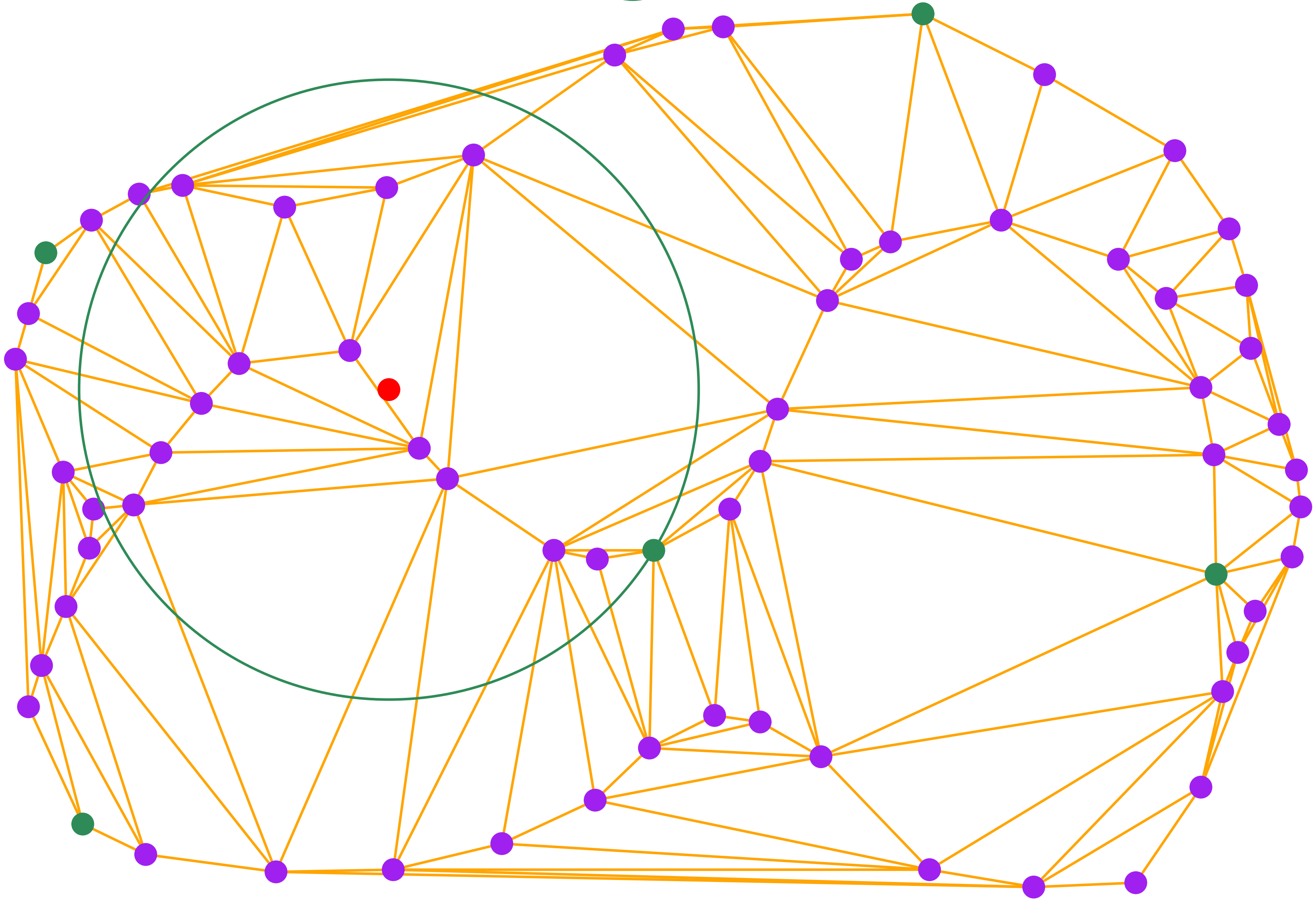


point location in $O\left(\alpha \log_\alpha n\right)$

point location in $O\left(\sqrt{\alpha} \log_\alpha n\right)$

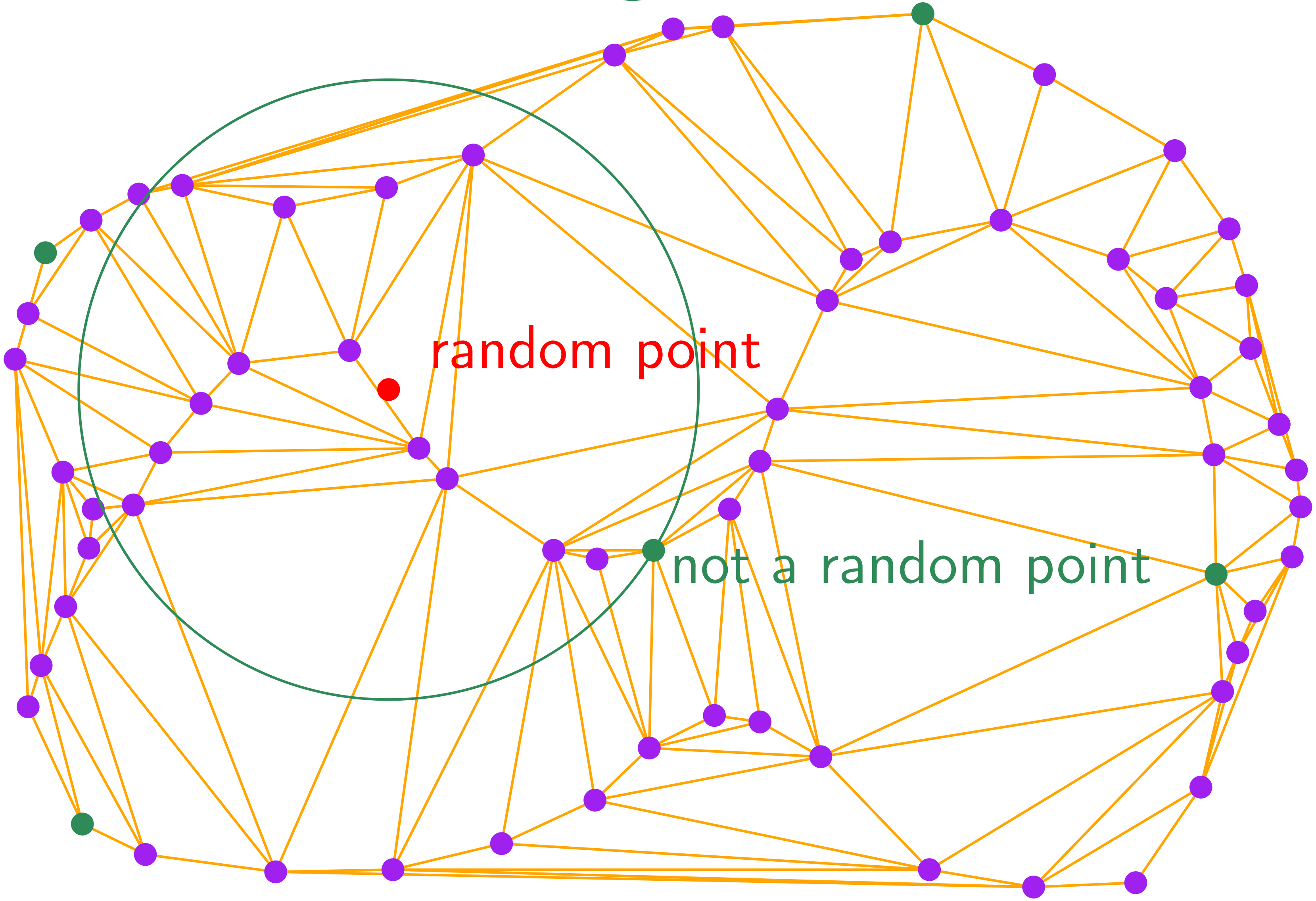
Technical detail

Walk length $= O\left(\sharp \text{ of } \bullet \text{ in } \bigcirc \right) = O\left(\frac{n}{k}\right)$



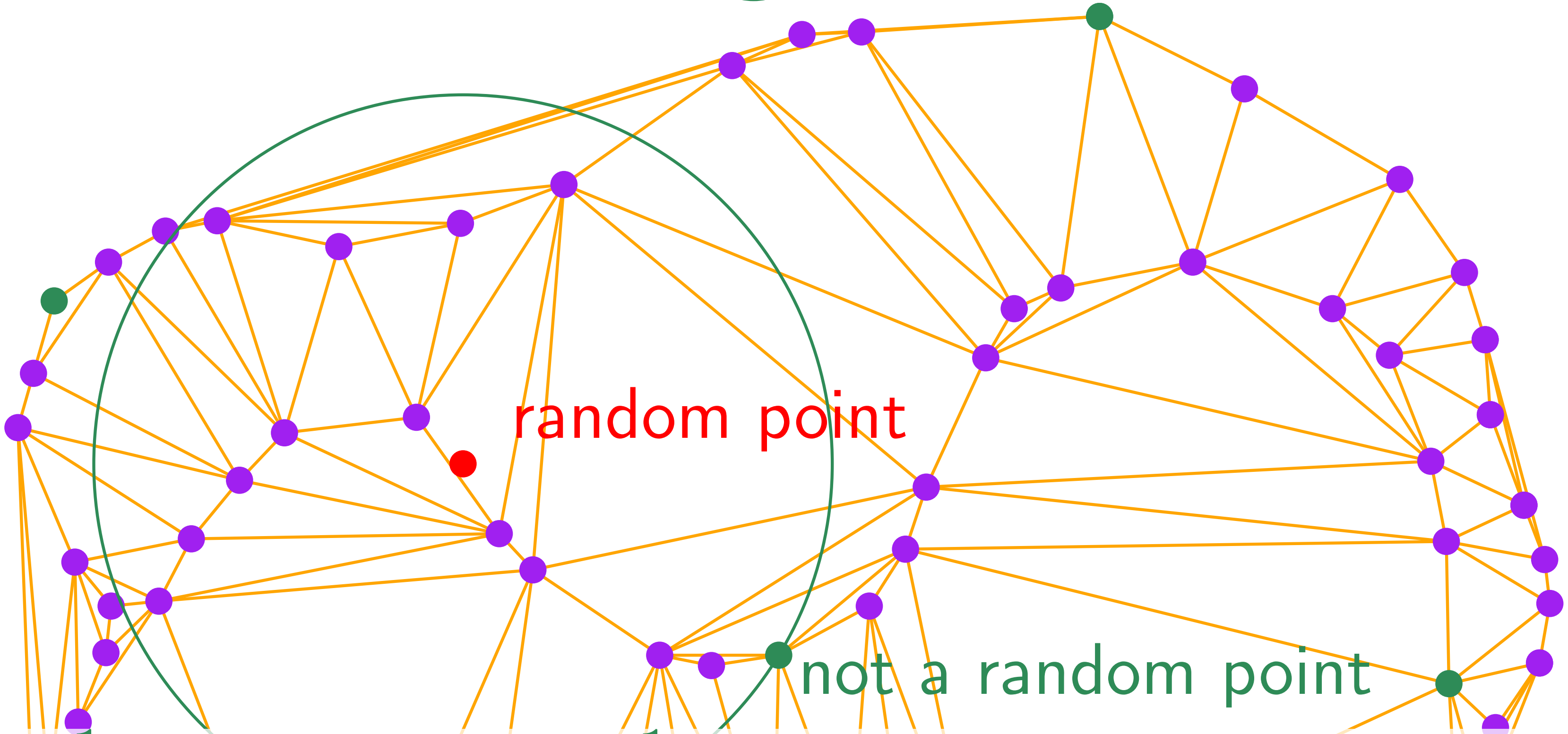
Technical detail

Walk length $= O\left(\sharp \text{ of } \bullet \text{ in } \bigcirc \right) = O\left(\frac{n}{k}\right)$



Technical detail

Walk length $= O\left(\sharp \text{ of } \bullet \text{ in } \odot \right) = O\left(\frac{n}{k}\right)$

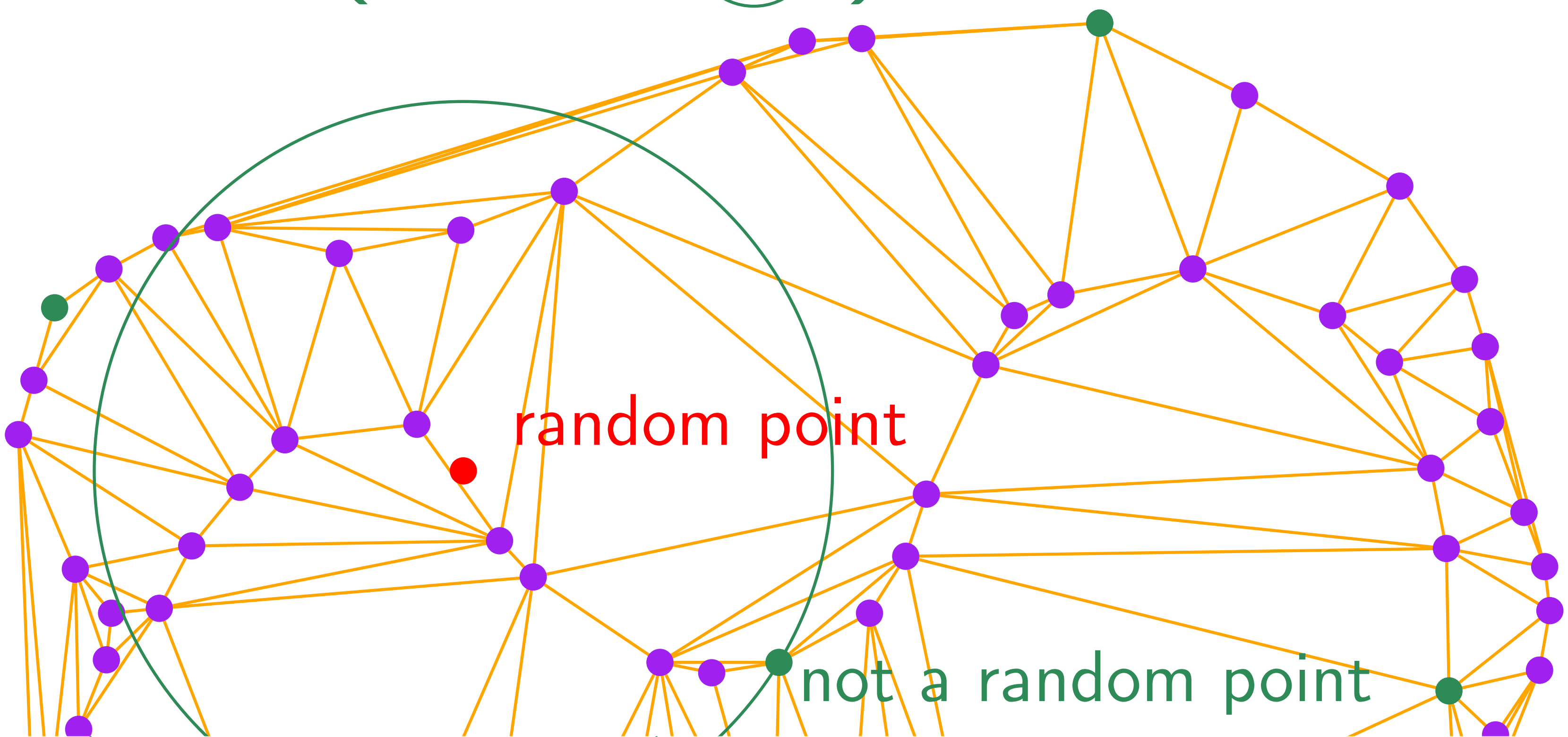


$$\mathbb{E}\left[d^\circ \bullet\right] = \frac{1}{n}\sum_{q} d^\circ NN(q) = \frac{1}{n}\sum_{q}\sum_{v=NN(q)} d^\circ v$$

$$= \frac{1}{n}\sum_{v}\sum_{q;v=NN(q)} d^\circ v \leq \frac{1}{n}\sum_{v} 6 d^\circ v \leq 36$$

Technical detail

Walk length $= O\left(\sharp \text{ of } \bullet \text{ in } \odot\right) = O\left(\frac{n}{k}\right)$, where $\sharp$ of $\bullet$ in $\odot$ $= \sum d^\circ$



$$\mathbb{E}\left[d^\circ \bullet\right] = \frac{1}{n}\sum_{q} d^\circ NN(q) = \frac{1}{n}\sum_{q}\sum_{v=NN(q)} d^\circ v$$

$$= \frac{1}{n}\sum_{v}\sum_{q;v=NN(q)} d^\circ v \leq \frac{1}{n}\sum_{v} 6d^\circ v \leq 36$$

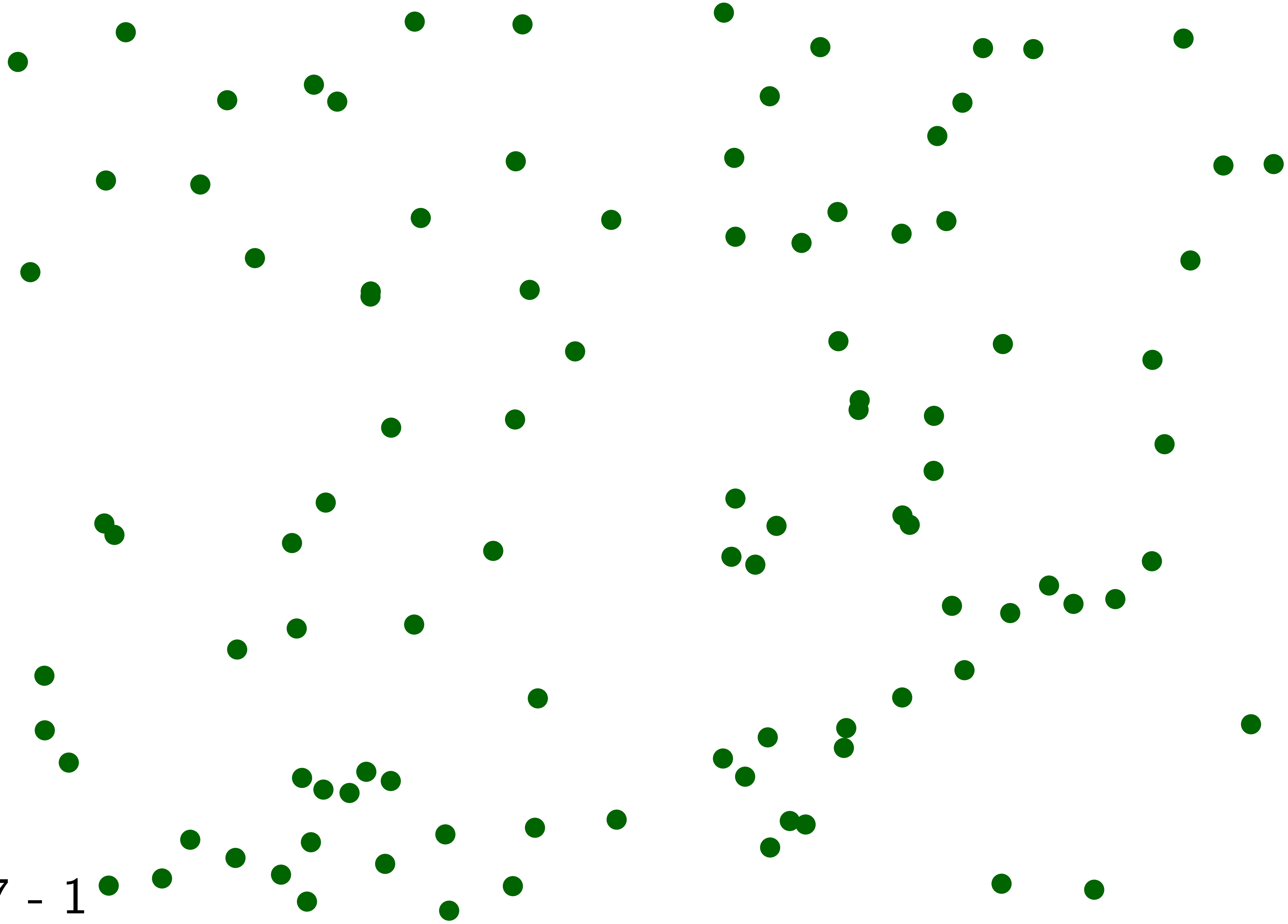
Randomization

Drawbacks of random order

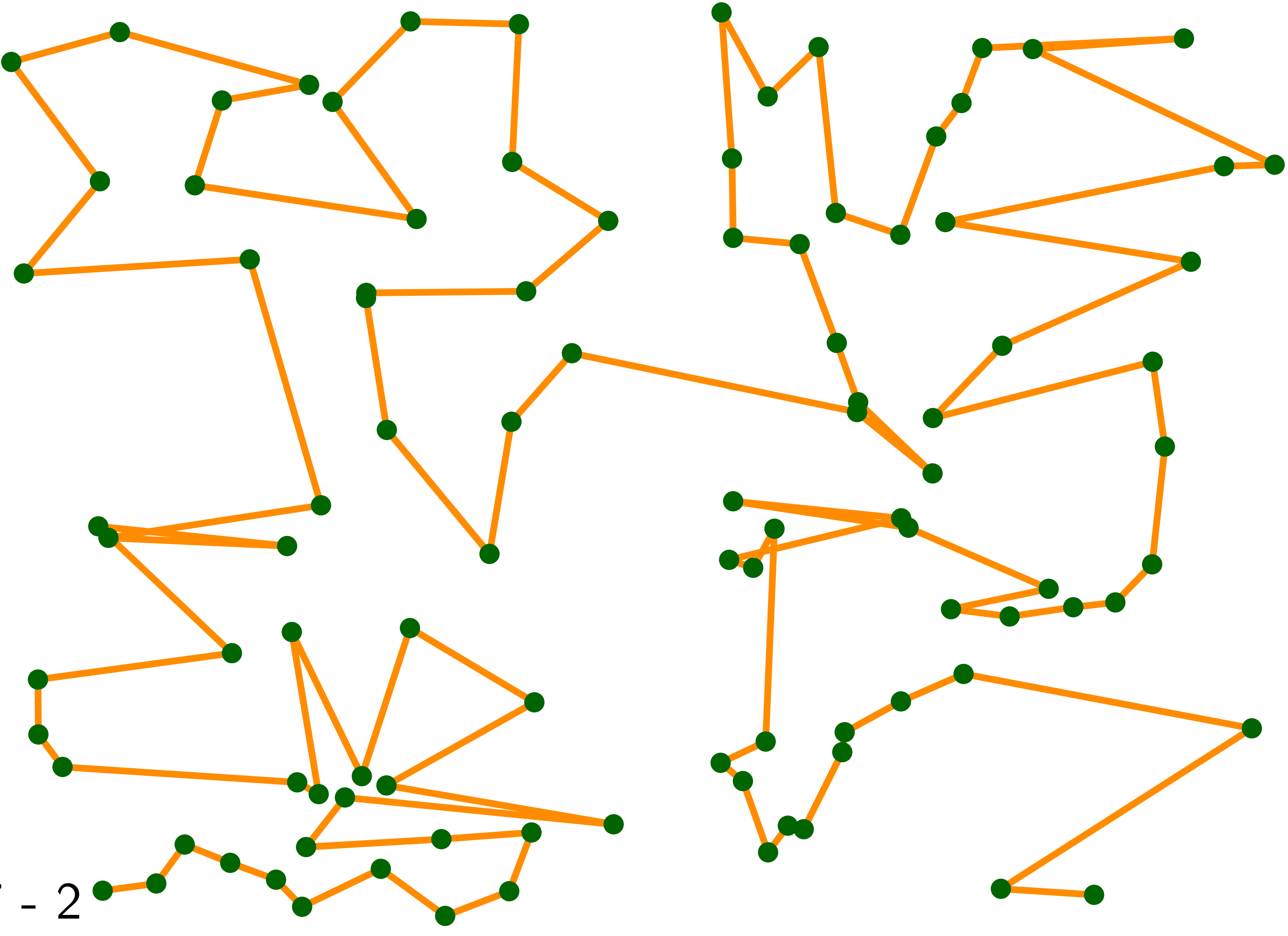
non locality of memory access

data structure for point location

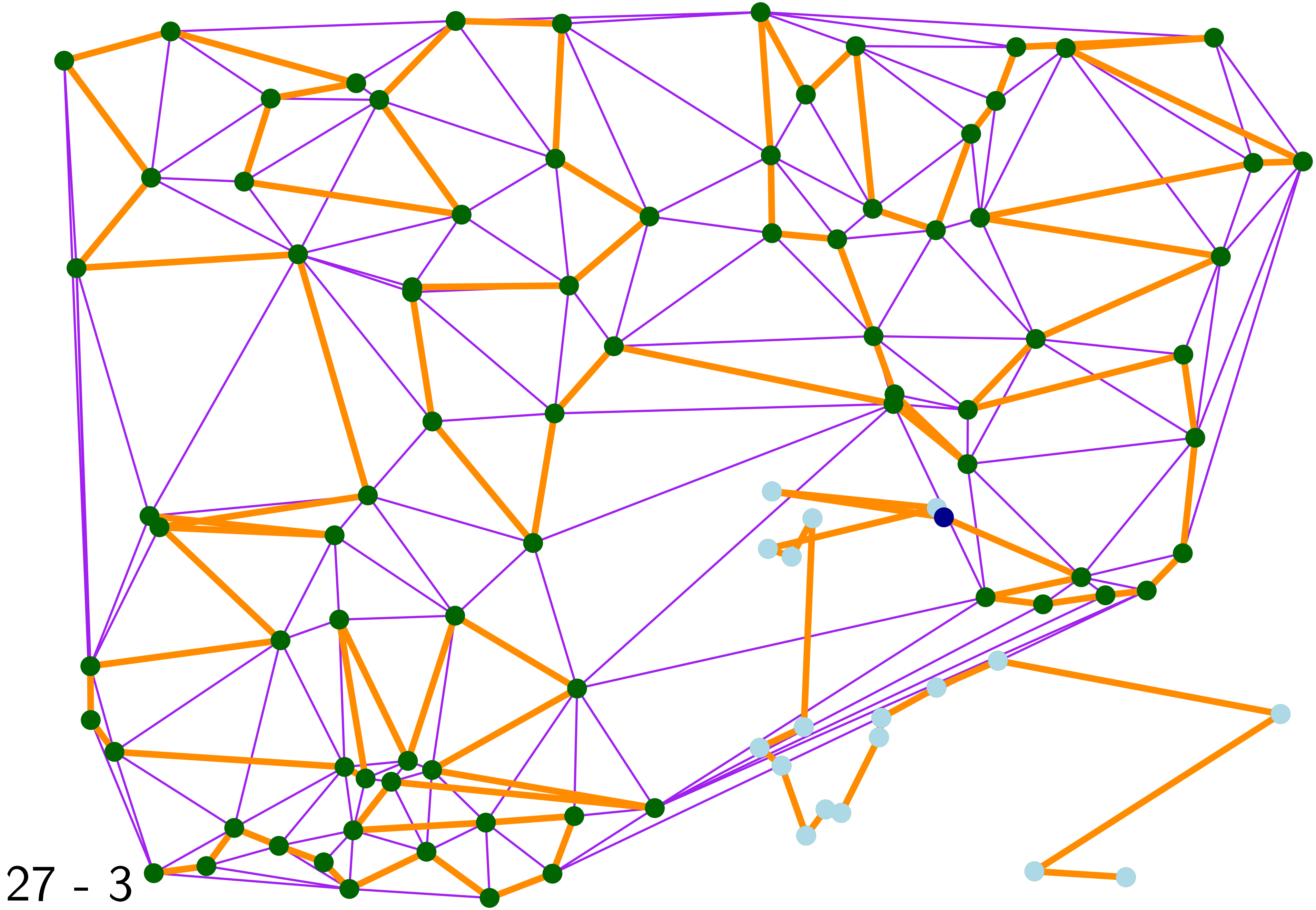
⟶ Hilbert sort

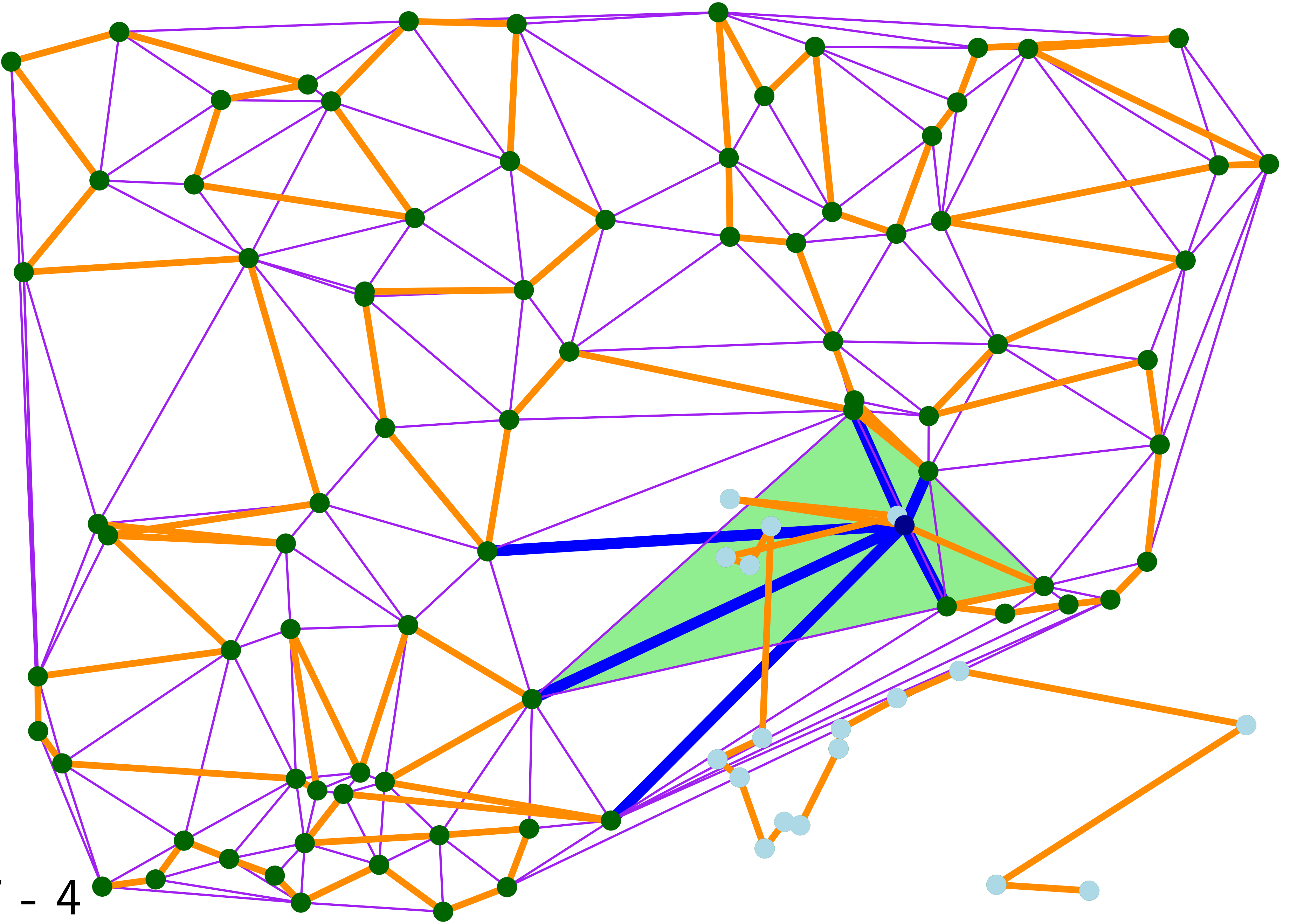


27 - 1



27 - 2


27 - 3



27 - 4

Drawbacks of random order

non locality of memory access

data structure for point location

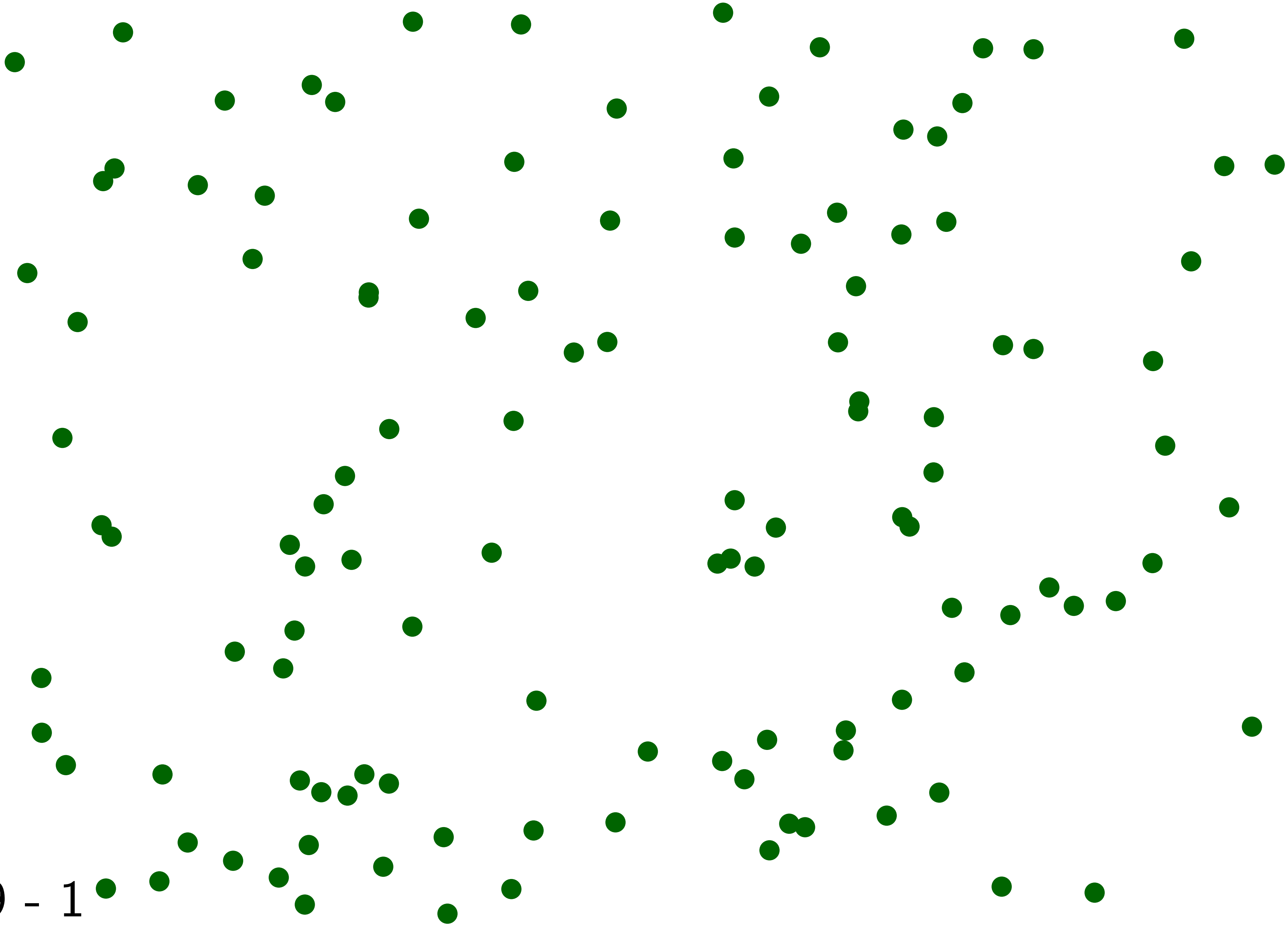
$\longrightarrow$ Hilbert sort

Walk should be fast

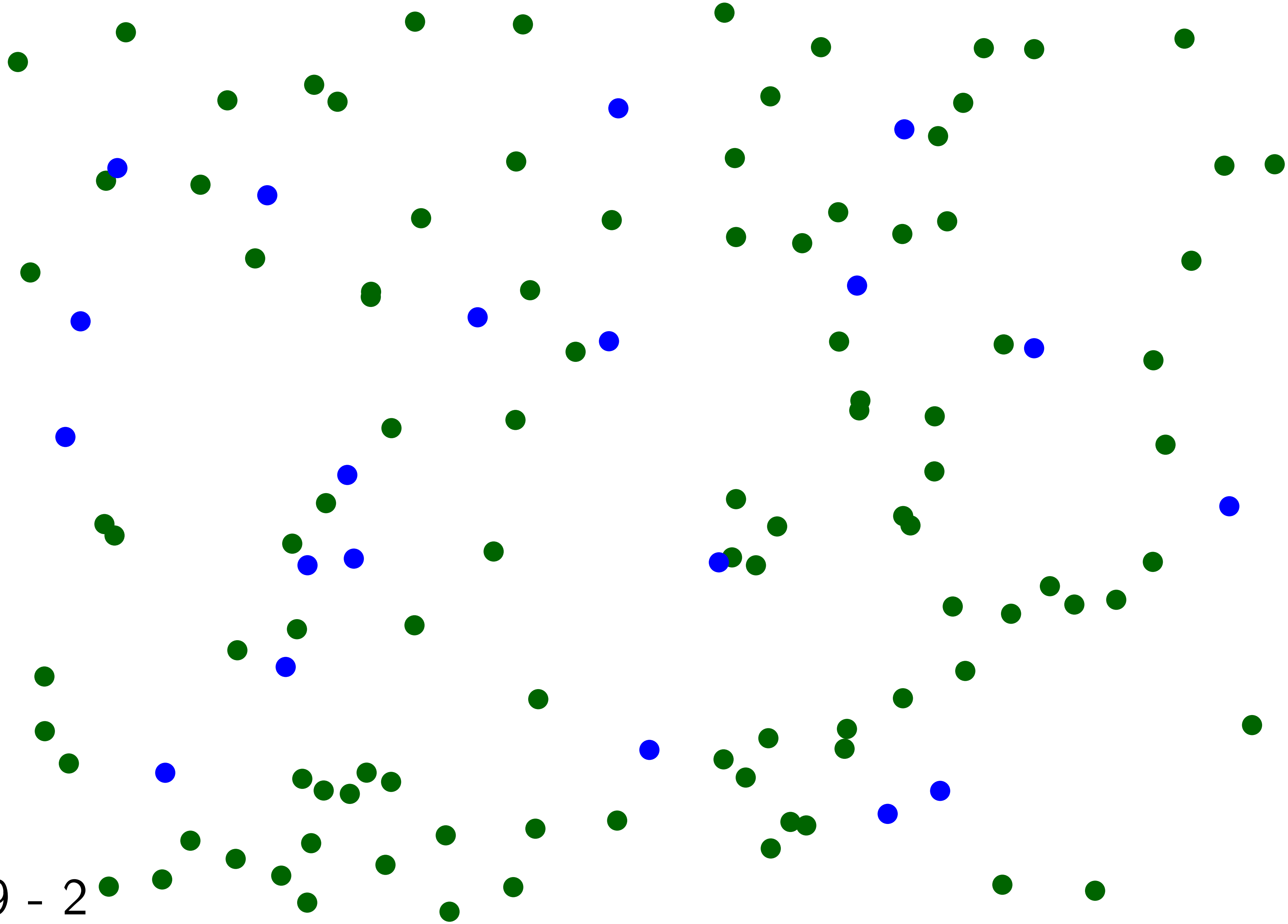
Last point is not at all a random point

$\longrightarrow$ no control of degree of last point

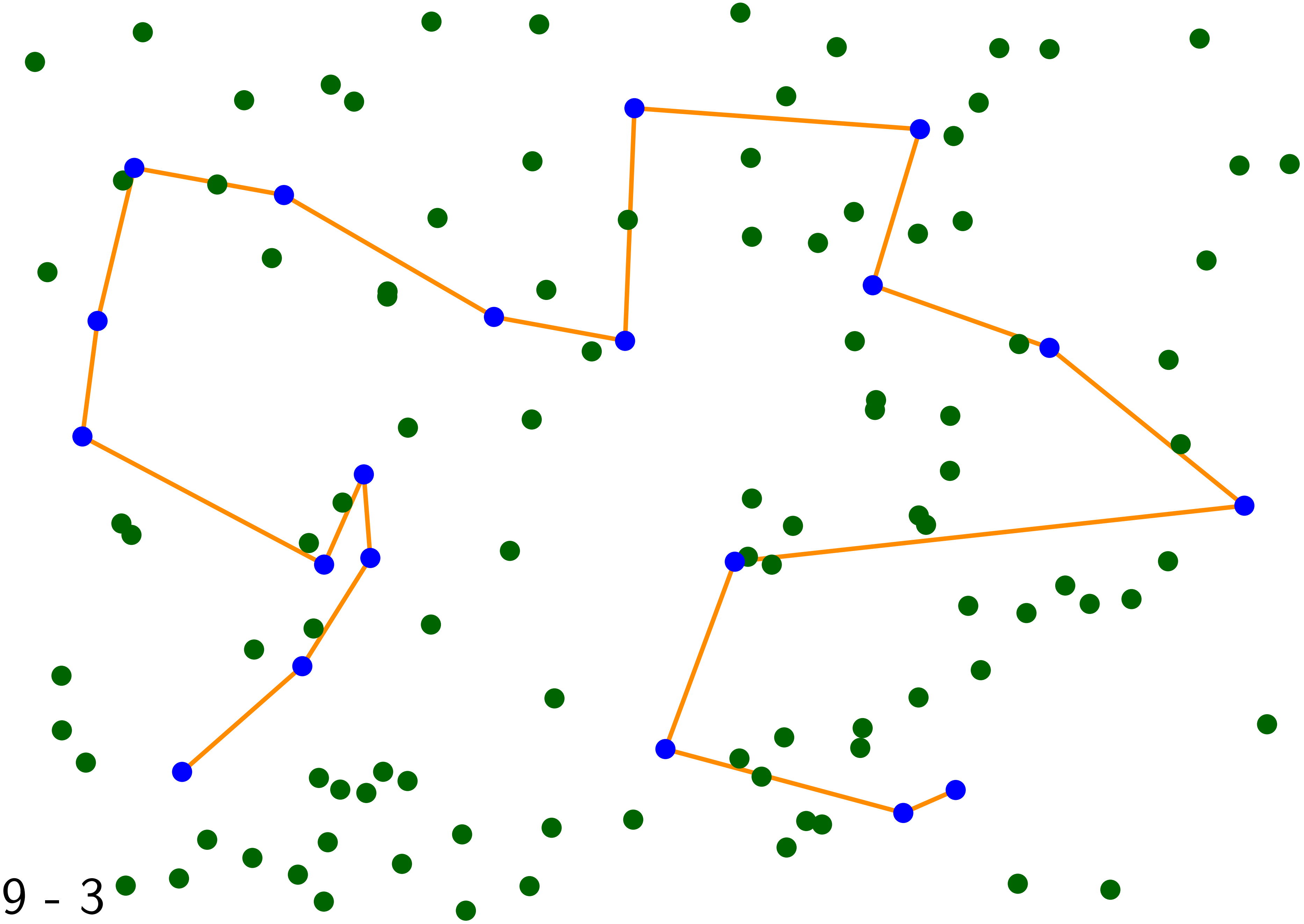
Biased Random Insertion Order (BRIO)



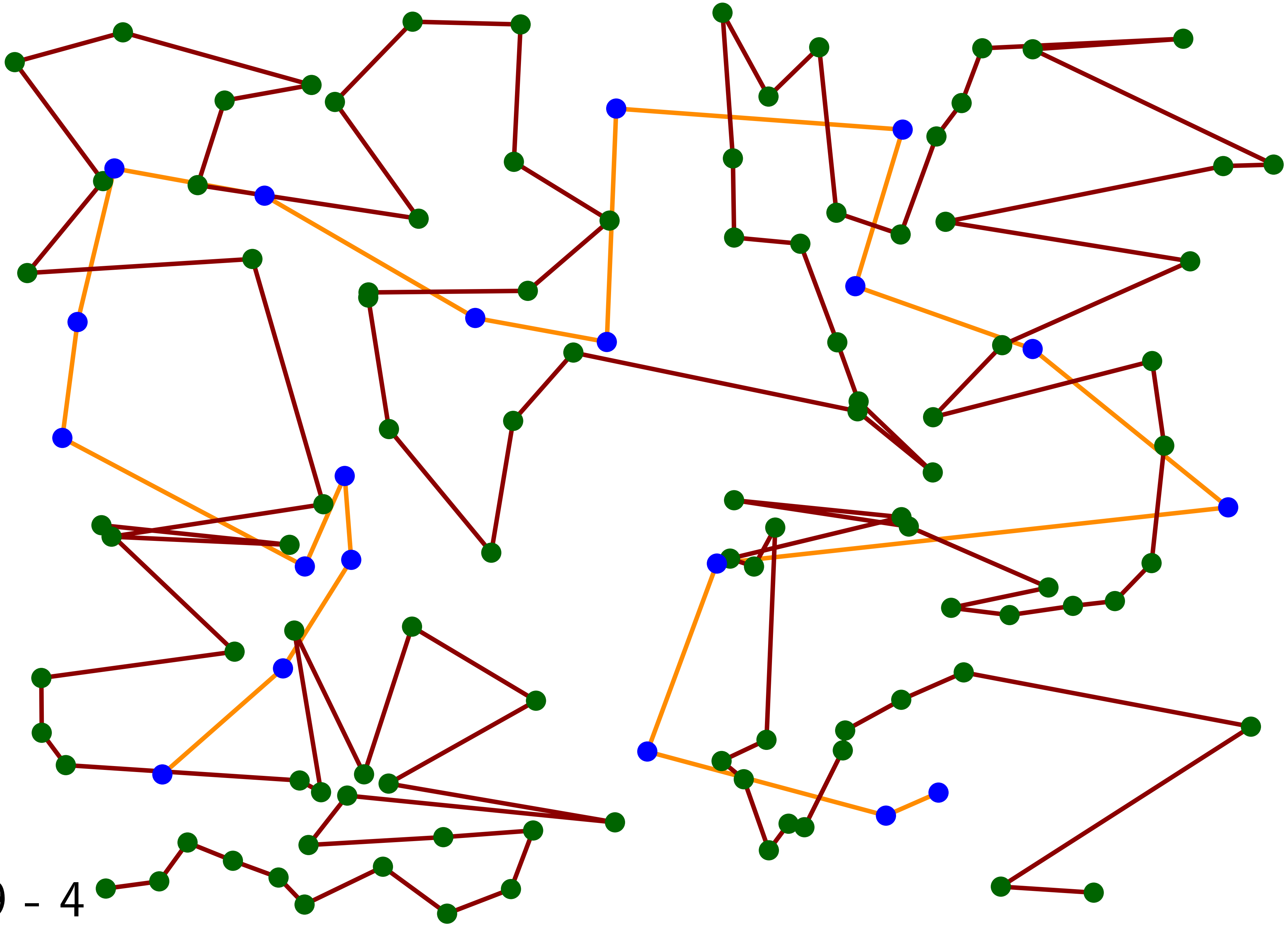
Biased Random Insertion Order (BRIO)



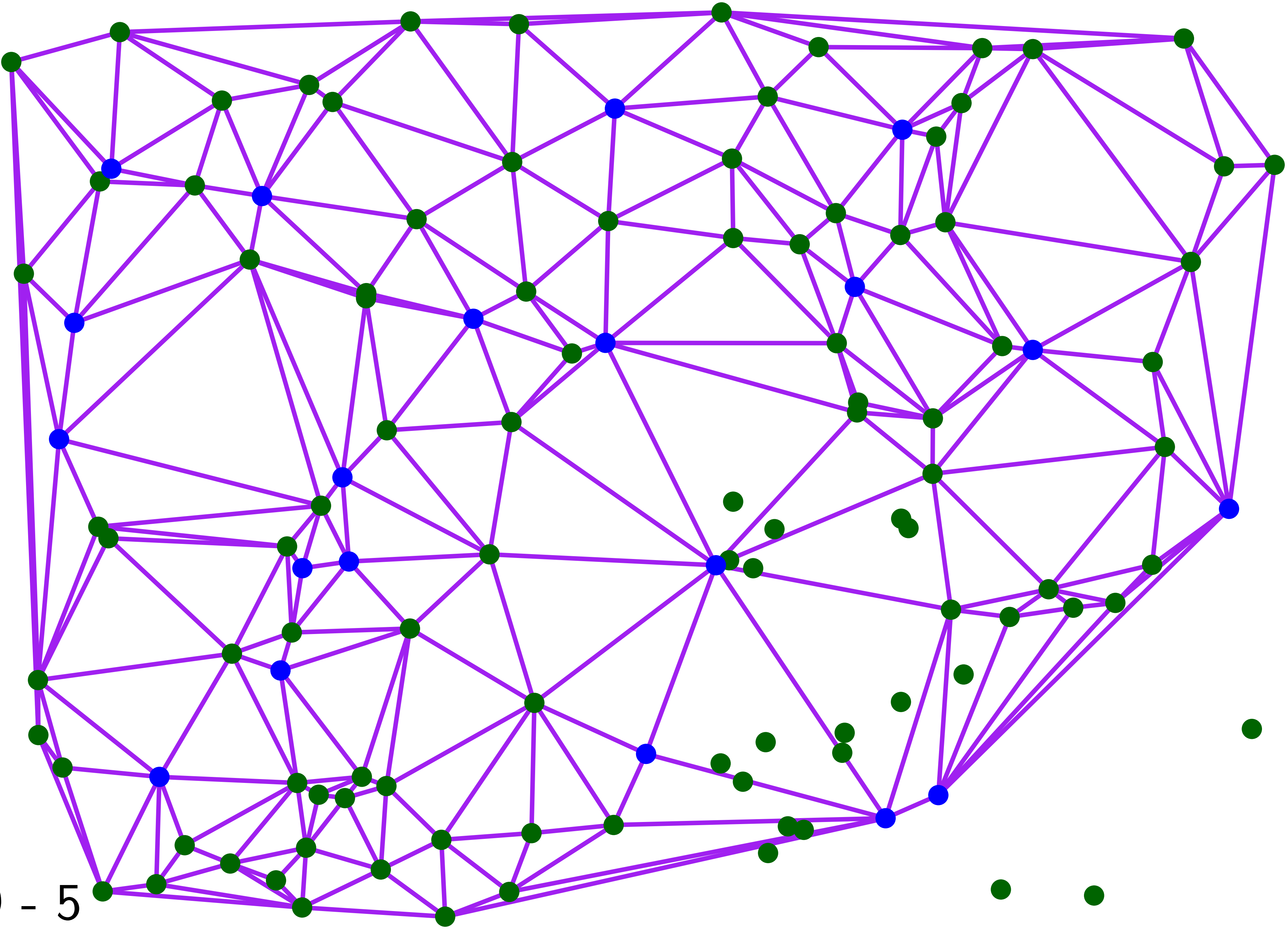
Biased Random Insertion Order (BRIO)



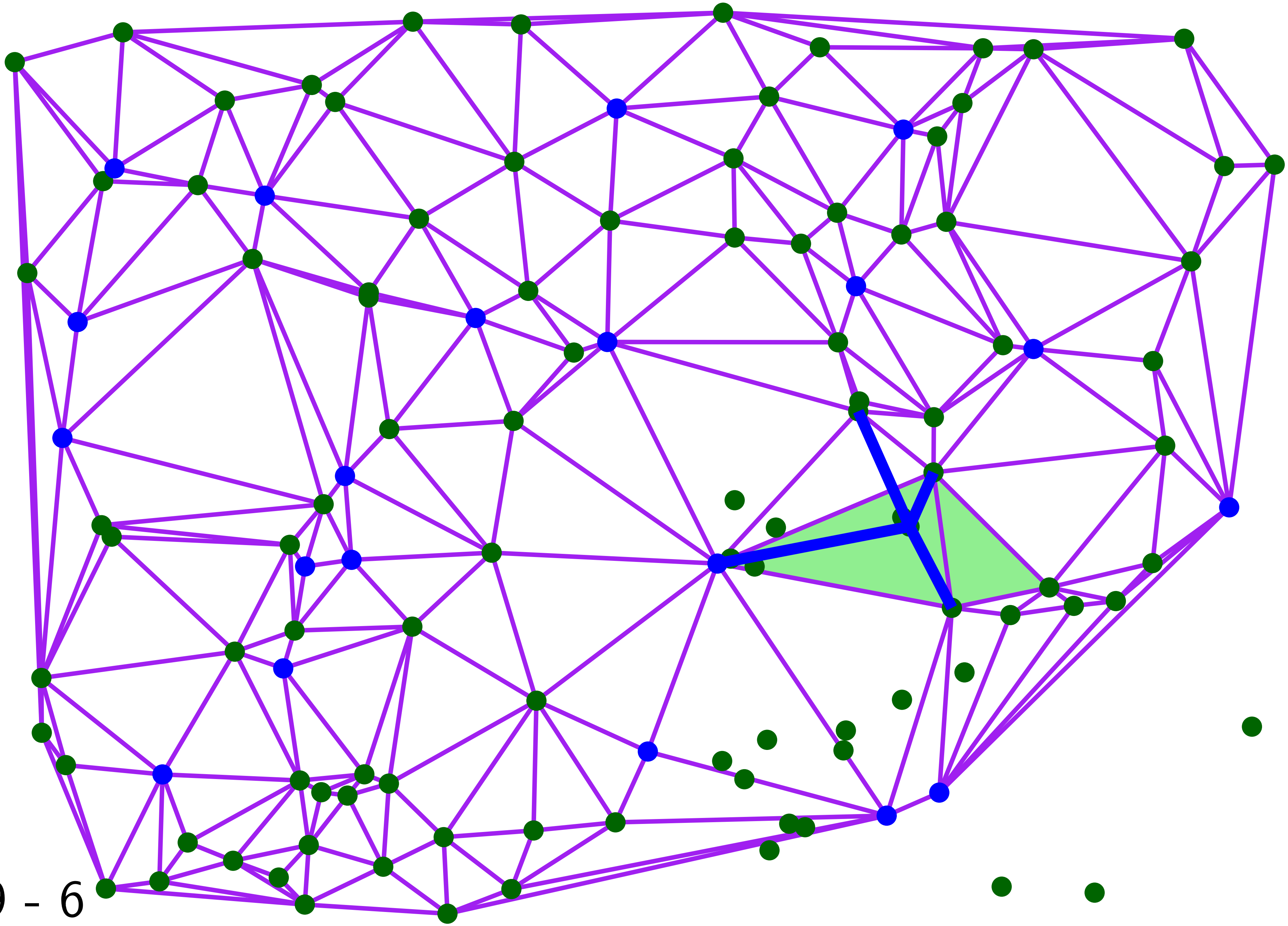
Biased Random Insertion Order (BRIO)



Biased Random Insertion Order (BRIO)



Biased Random Insertion Order (BRIO)



Algorithm

Recipe to go from the input to the output

- Formalized description in some language
- May use data structure
- Proof of correctness
- Complexity analysis

Algorithm Program

Recipe to go from the input to the output

Implementation of an algorithm

Formalized description in some language

Translation in a programming language (C++)

May use data structure

May use software library

Proof of correctness

Debugging

Complexity analysis

Running time

Algorithm may be difficult to transform into Program

Recipe to go from the input to the output

Implementation of an algorithm

Formalized description in some language

Translation in a programming language (C++)

May use data structure

May use software library

Proof of correctness

Debugging

Complexity analysis

Running time

too complicated

too complicated

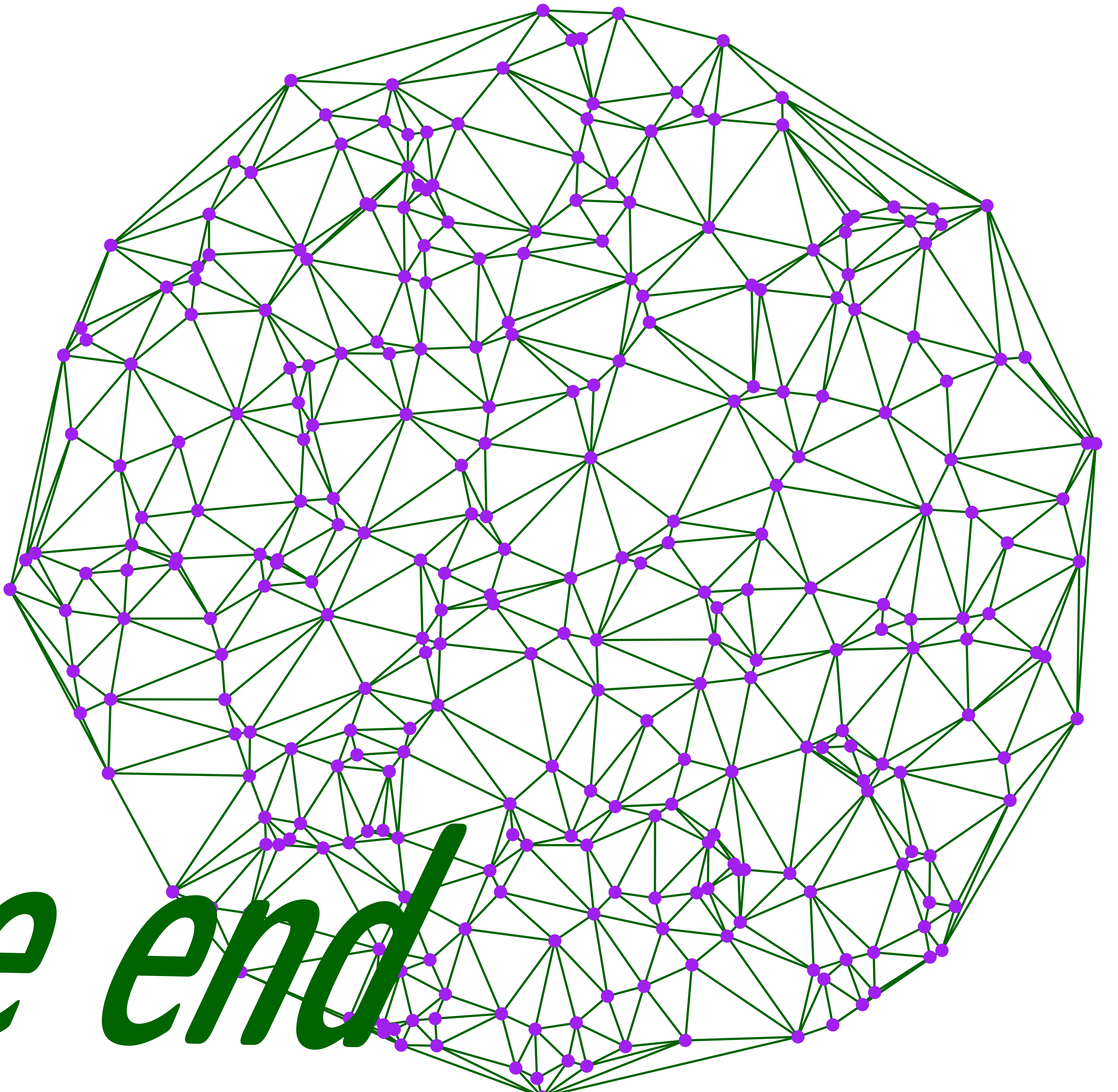
inexistant

Computation model

big O notation

actual computer

(cache, prediction...)



The end