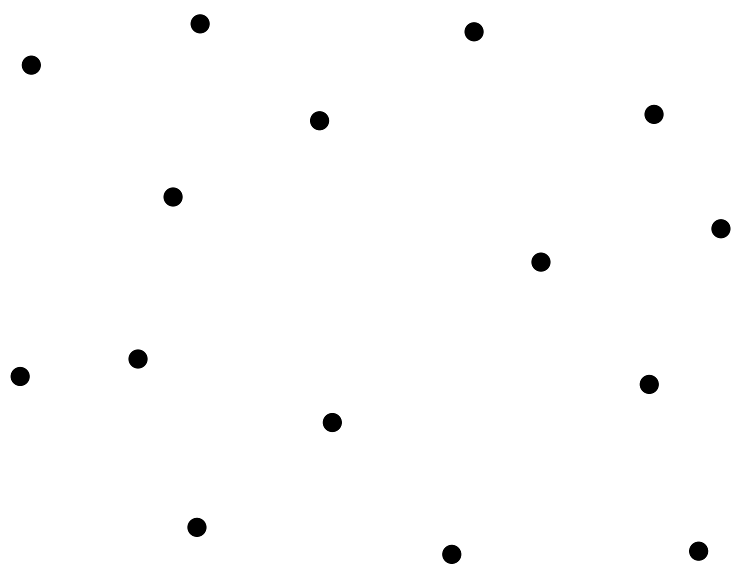
Delaunay triangulation

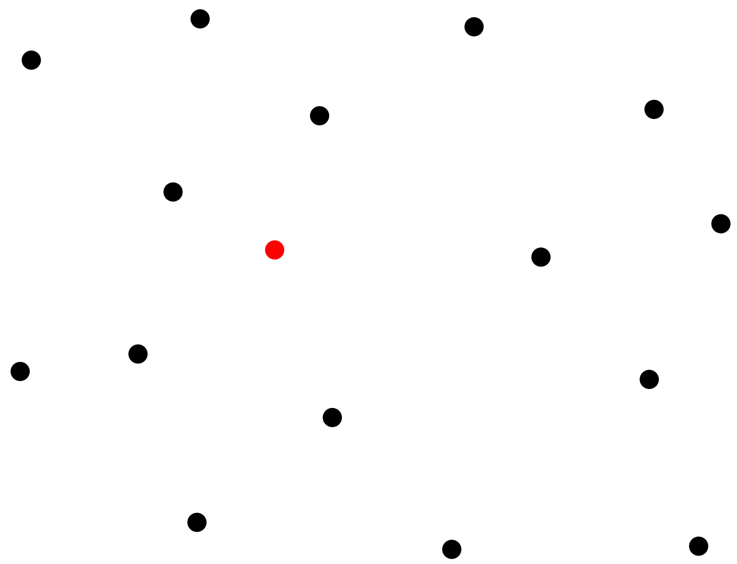
from theory to practice

INRIA Sophia Antipolis

Olivier Devillers

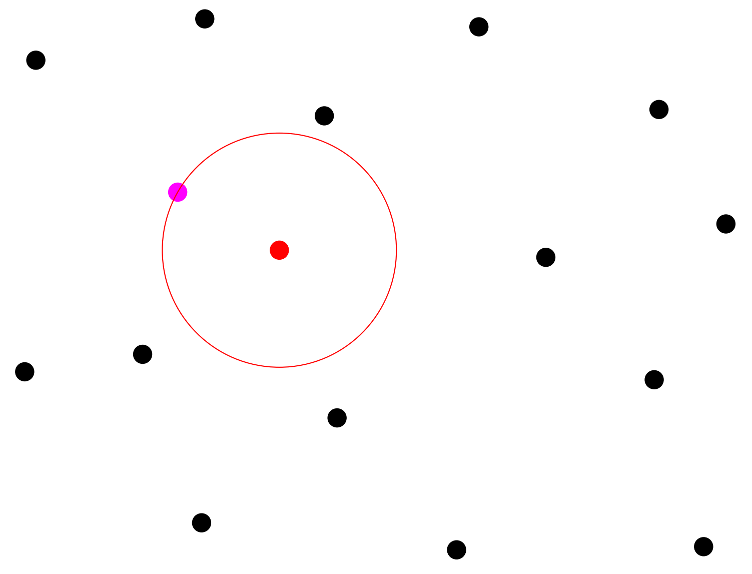
Voronoi Diagram

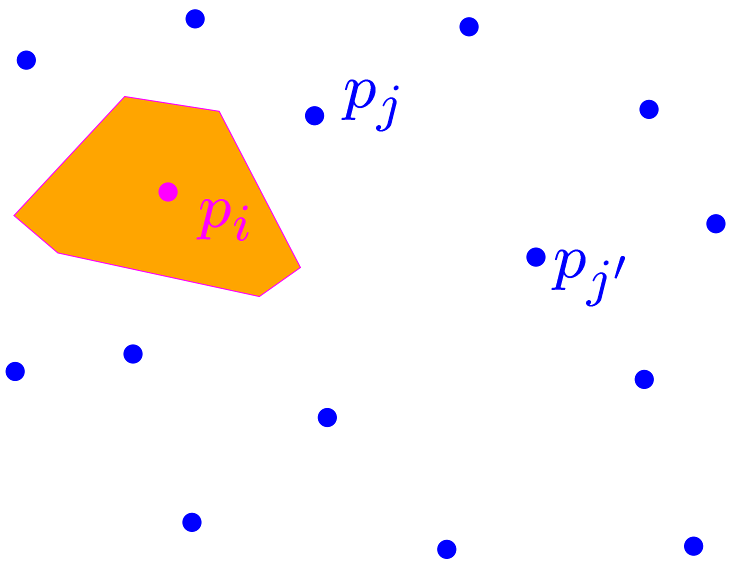




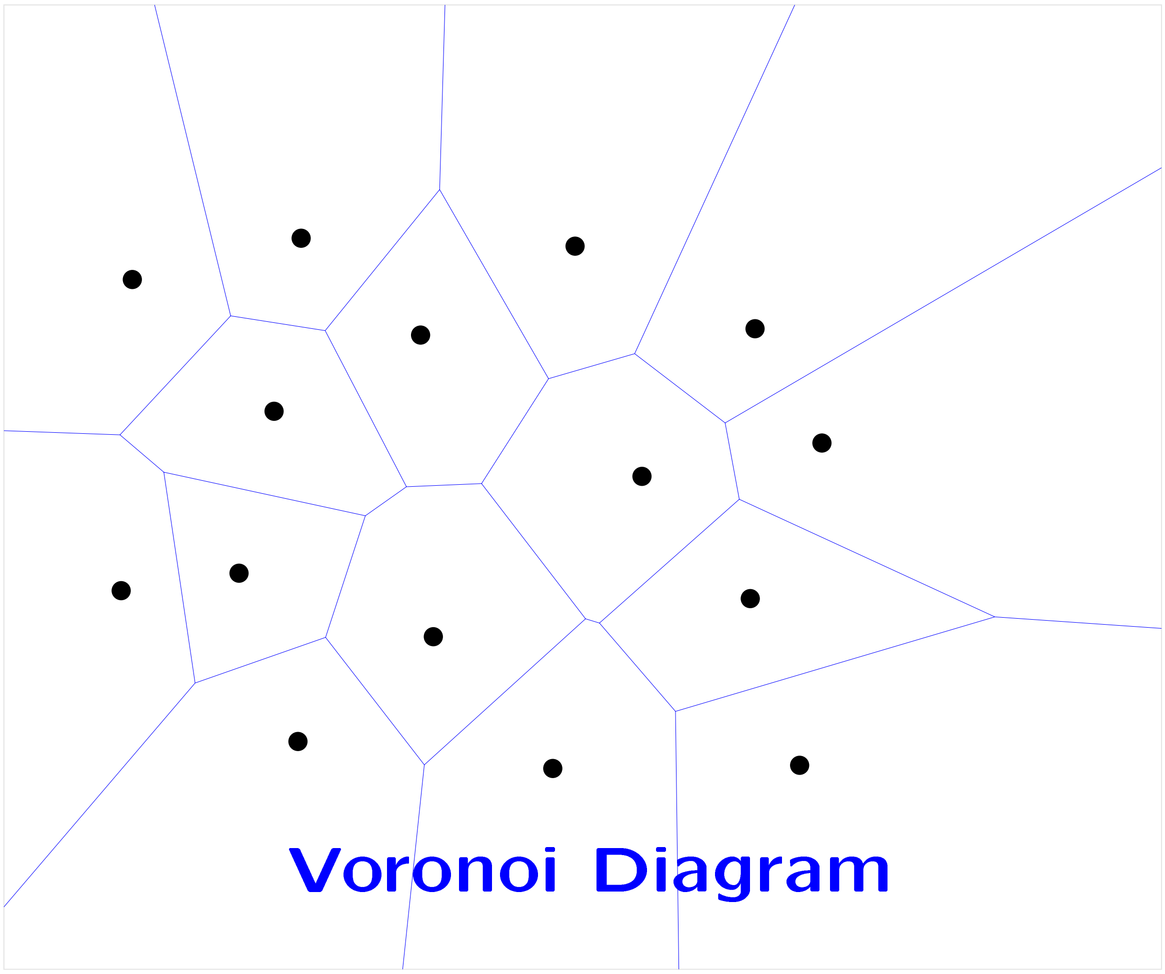
Nearest neighbor ?

Delaunay triangulation

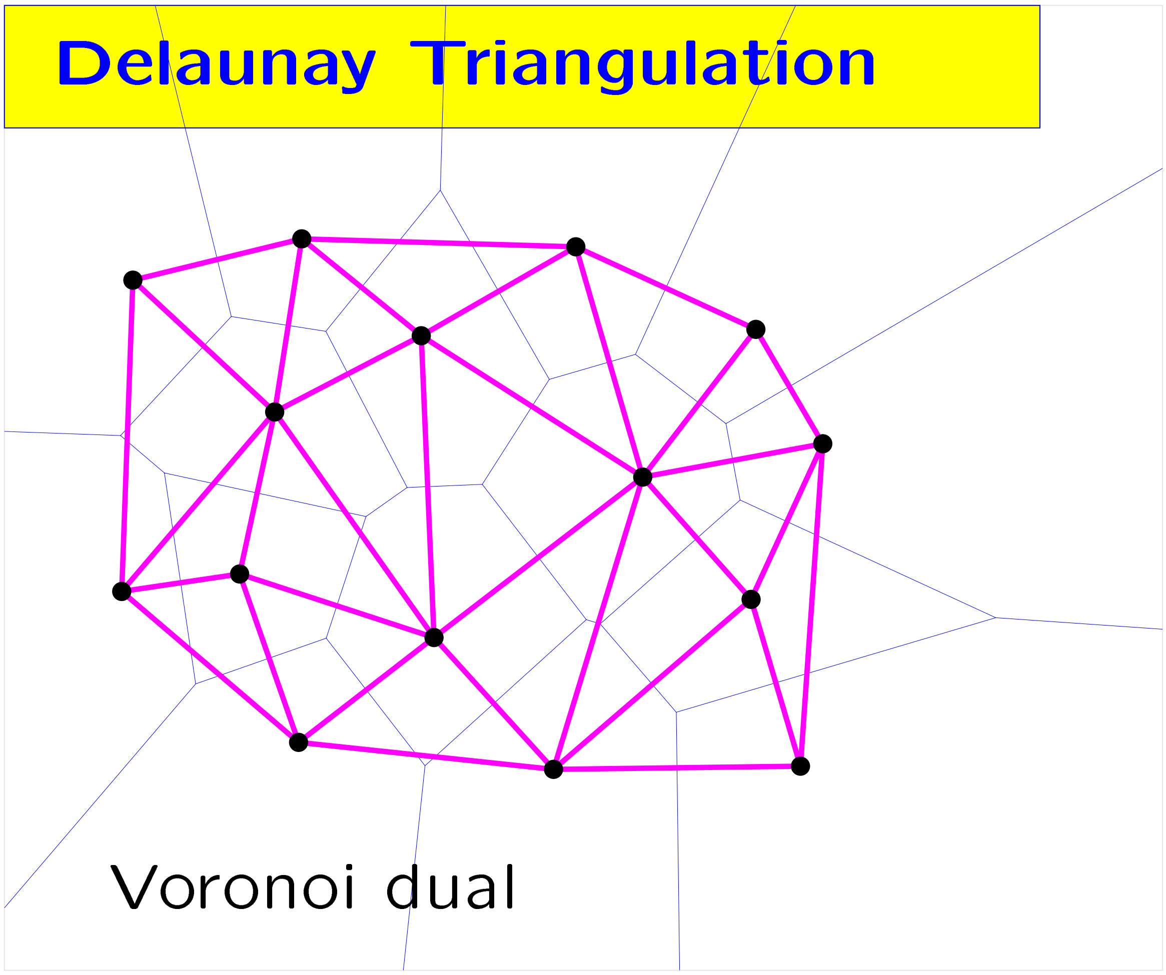




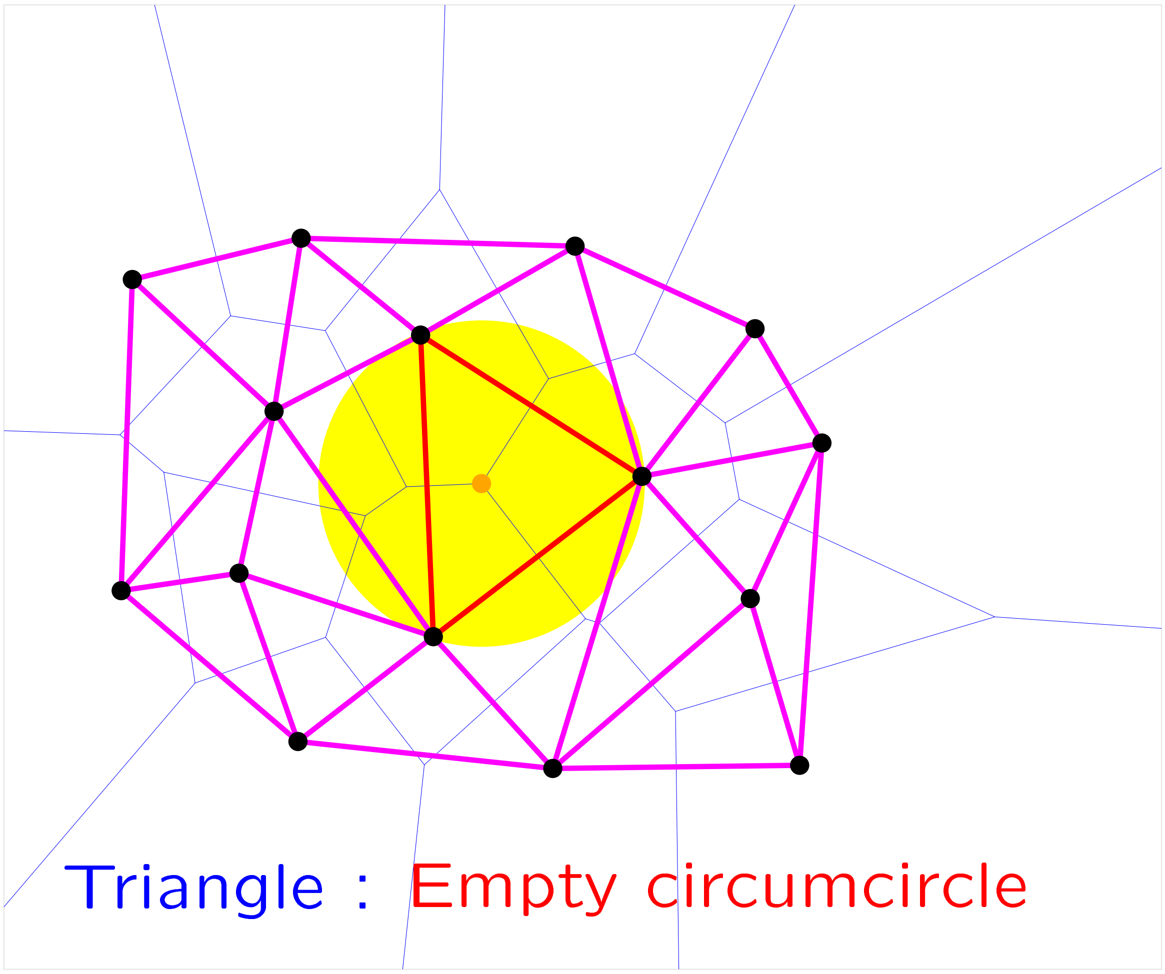
Nearest is p_i

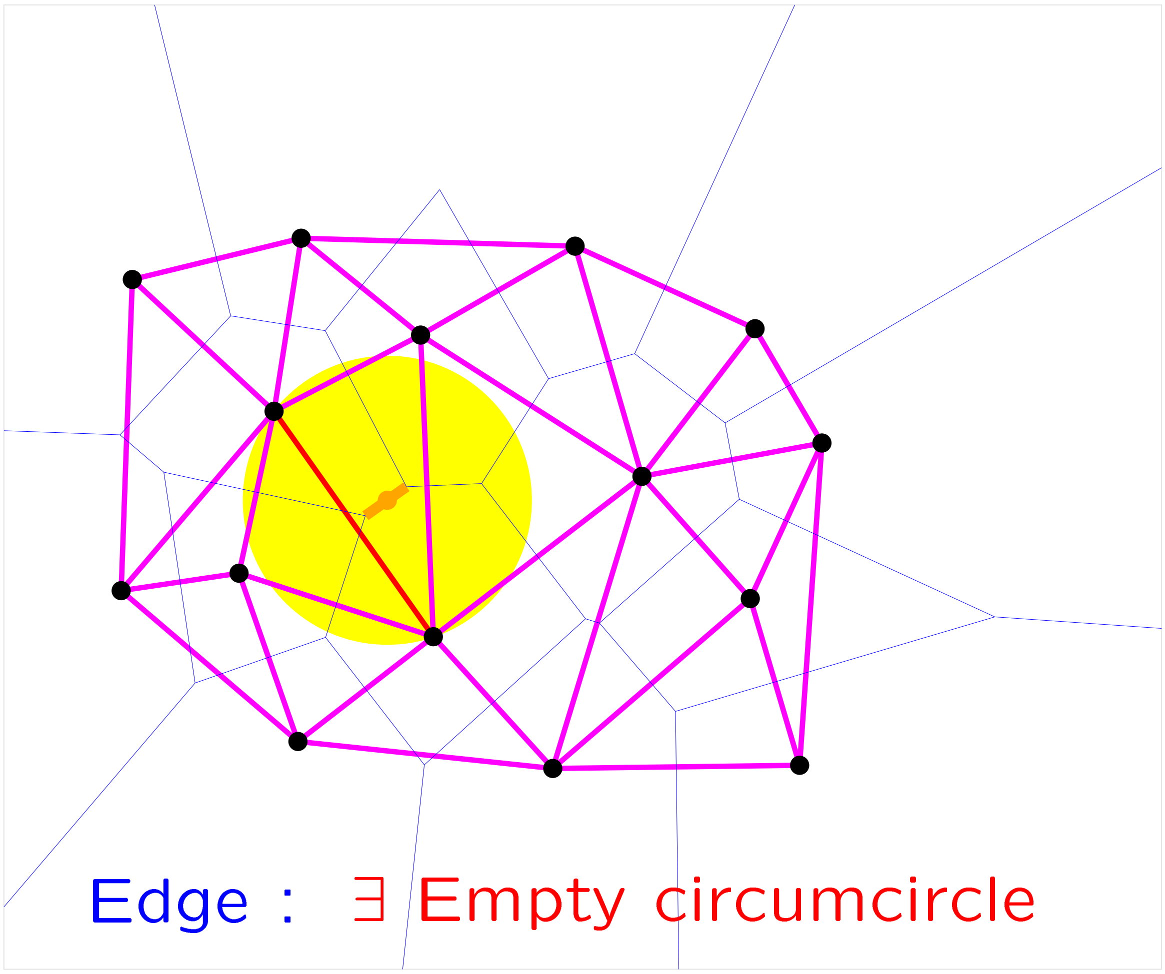

Voronoi Diagram

Delaunay Triangulation

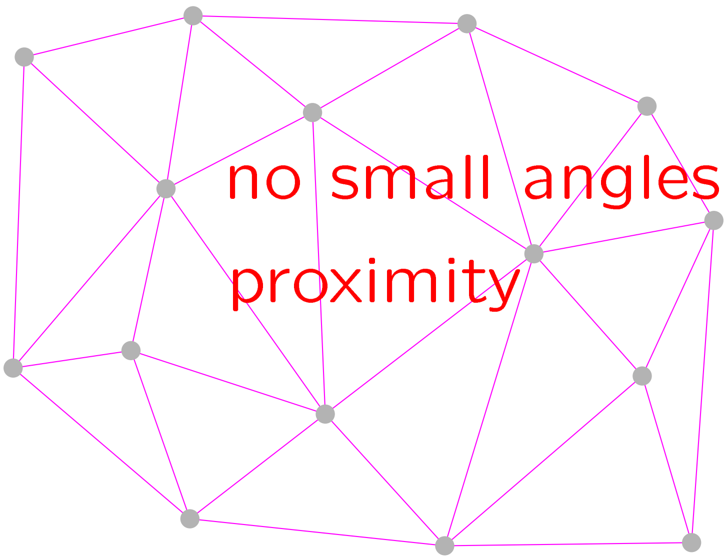


Voronoi dual




Edge : ∃ Empty circumcircle

Good properties



no small angles

proximity

Lower bound

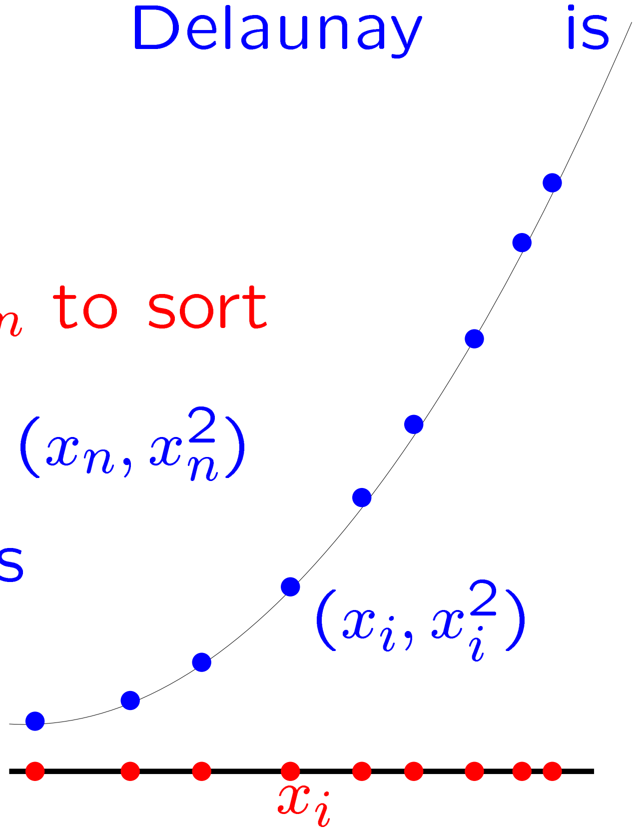
Theorem: Delaunay is $\Omega(n \log n)$.

Proof:

$x_1, x_2, \ldots, x_n$ to sort

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

n points



Theorem: Delaunay is $\Omega(n \log n)$.

Proof:

$x_1, x_2, \ldots, x_n$ to sort

$\downarrow$ $O(n)$

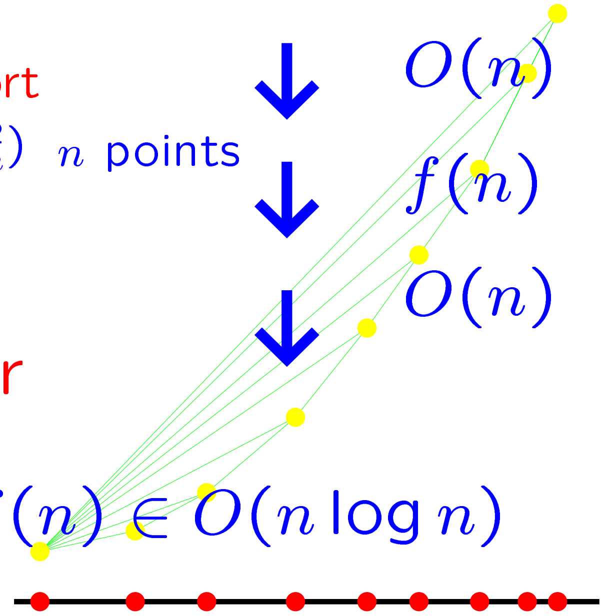
$(x_1, x_1^2), \ldots, (x_n, x_n^2)$ n points

$\downarrow$ $f(n)$

Delaunay

$\downarrow$ $O(n)$

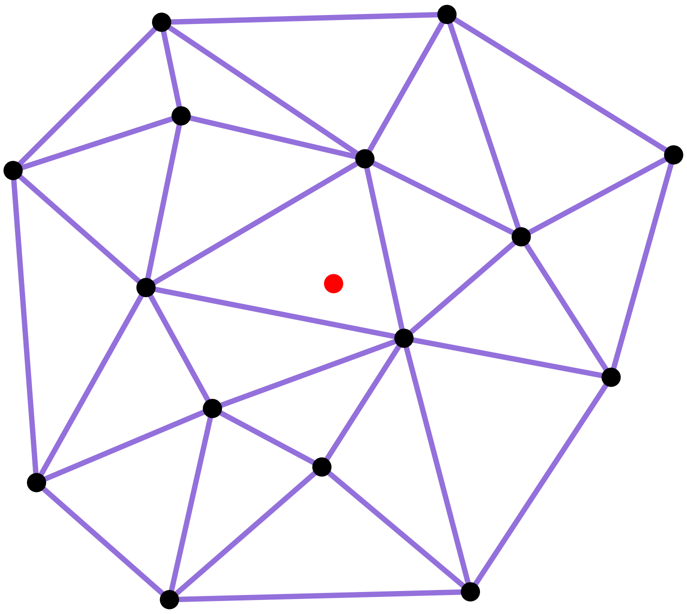
x order



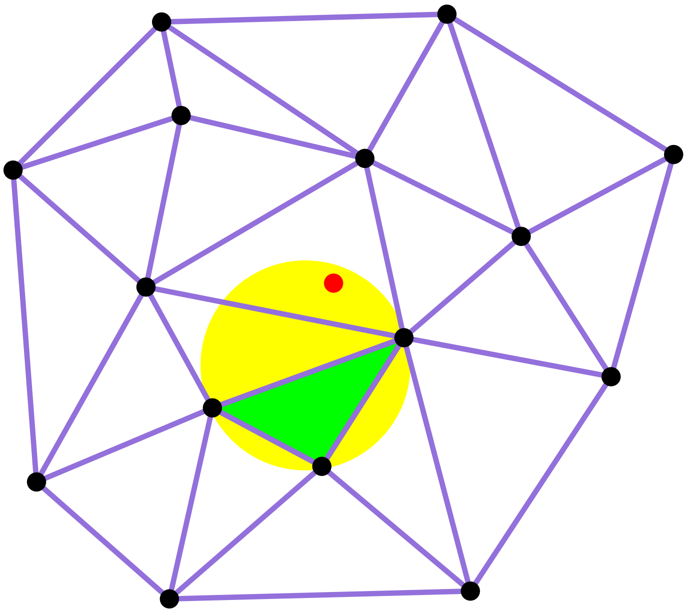
$$O(n) + f(n) \in O(n \log n)$$

First Algorithms

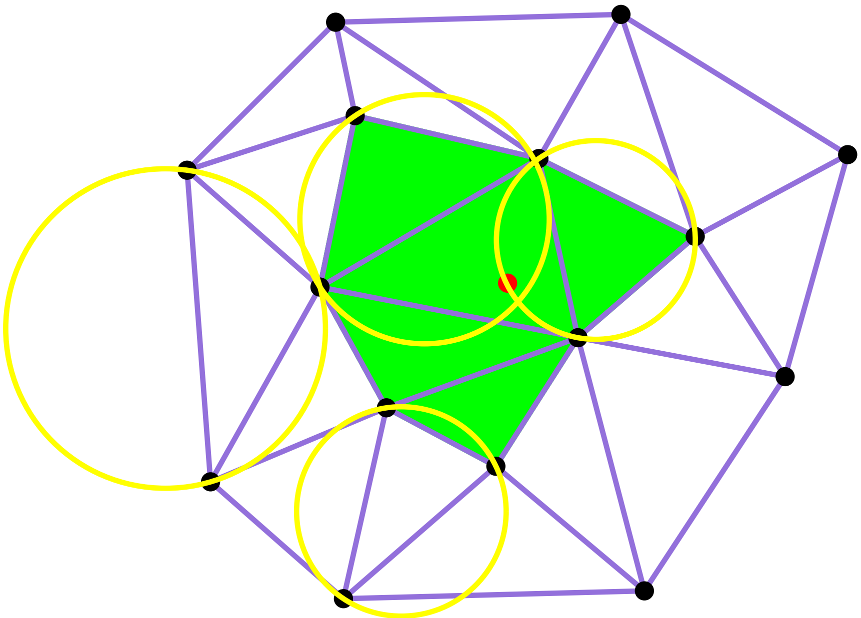
suboptimal

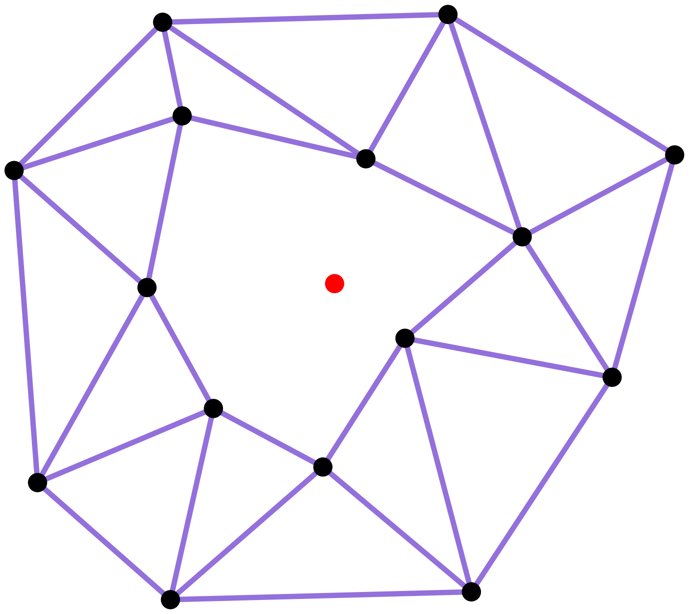


Incremental Algorithm

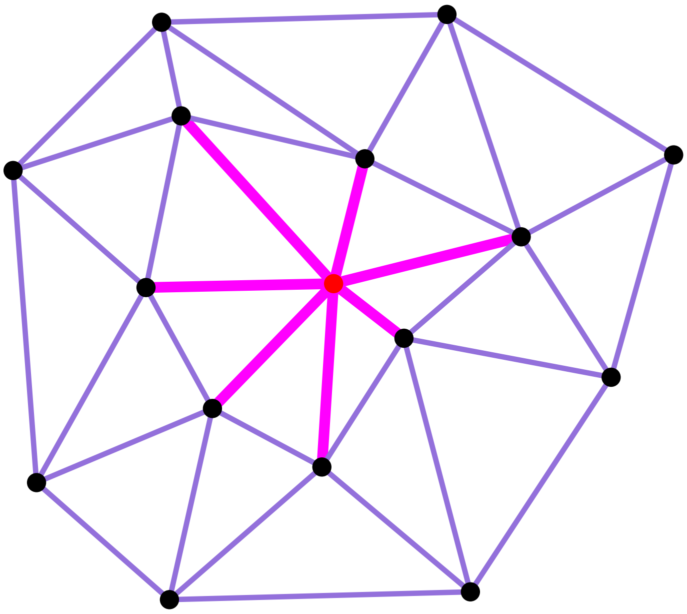


find all conflicting triangles





Delete them



Triangulate from p

Complexity

$O(n)$ for each insertion

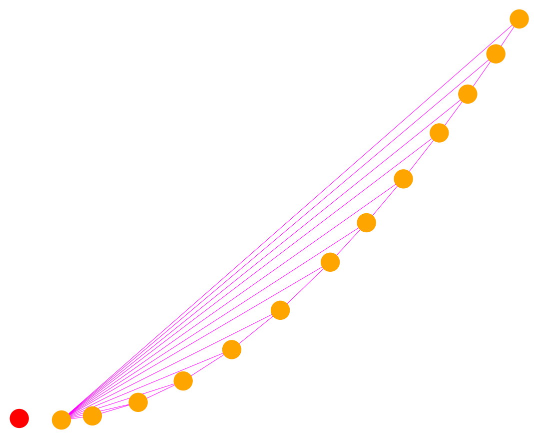
$O(n^2)$

Complexity

$O(n)$ for each insertion

$O(n^2)$

$\Omega(n^2)$

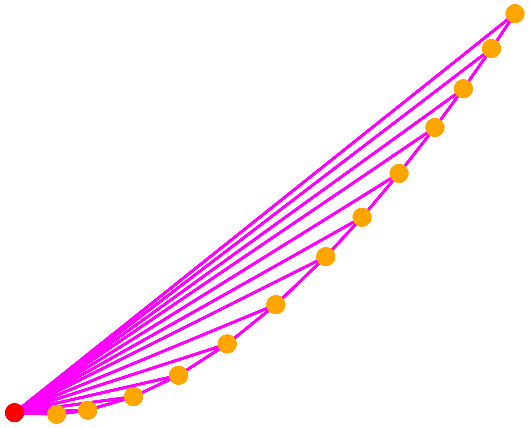


Complexity

$O(n)$ for each insertion

$O(n^2)$

$\Omega(n^2)$



Incremental algorithm

two stages

Location

where is the new point ?

Insertion

how to retriangulate

Incremental algorithm

two stages

Location

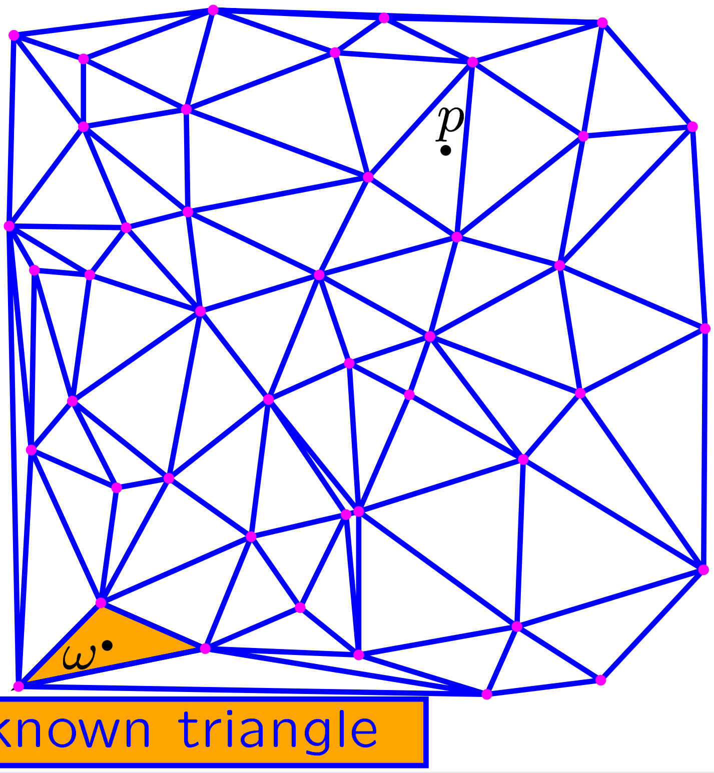
where is the new point ?

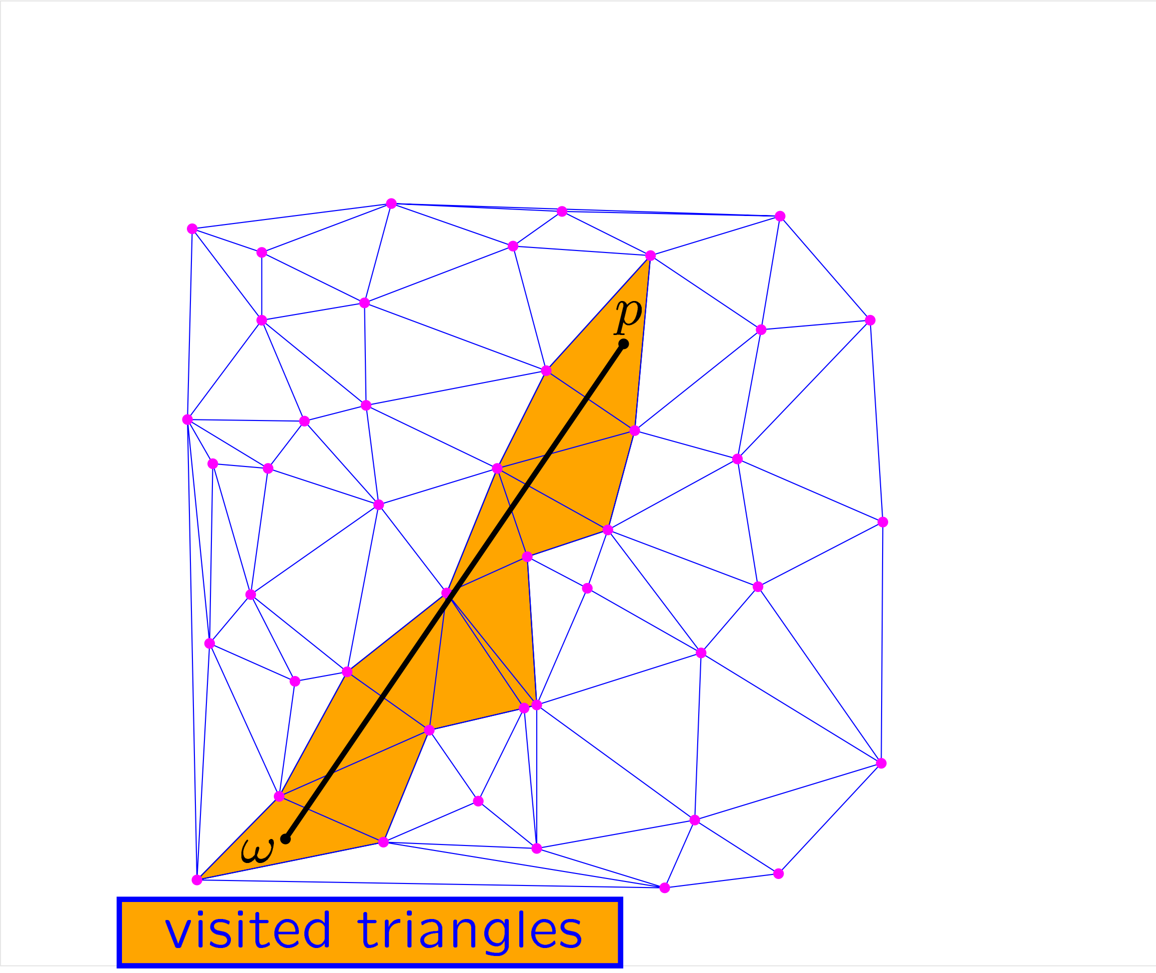
Insertion

how to retriangulate

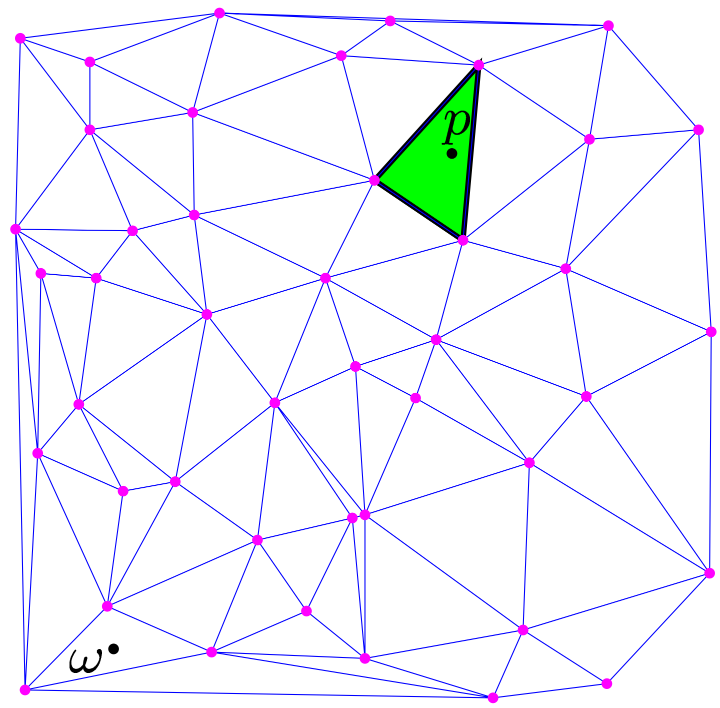
$\boxed{O(degree(p))}$

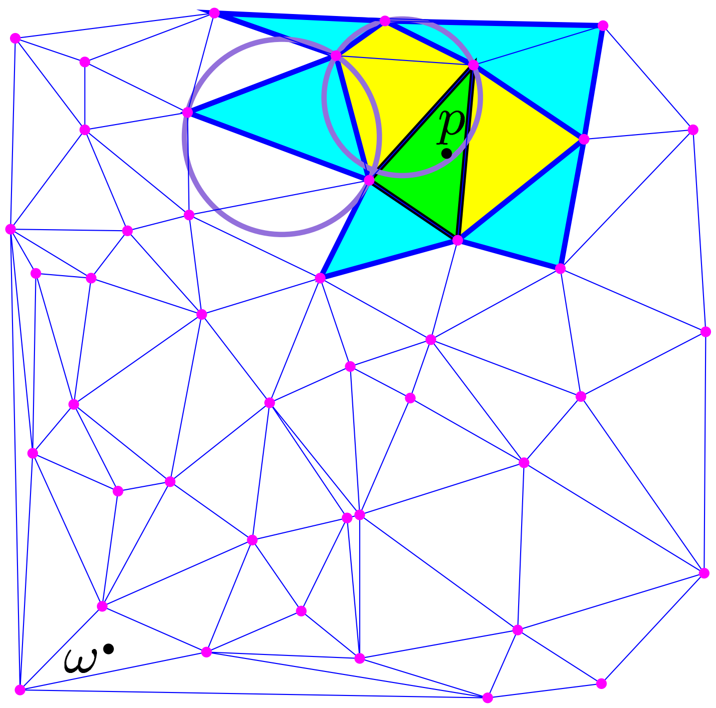
Location by march




p
ω
visited triangles

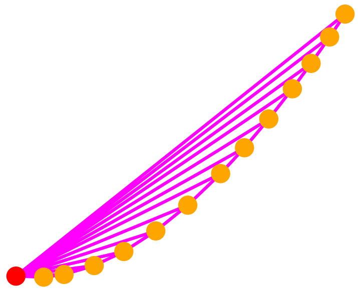
Delaunay triangulation




p
ω

Complexity

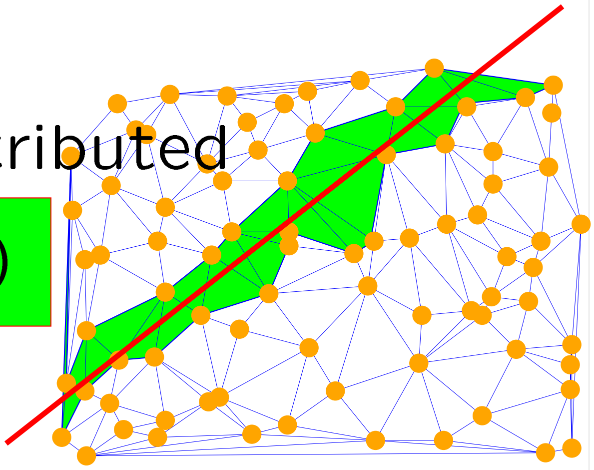
still $\boxed{\Omega(n^2)}$



On average

Evenly distributed

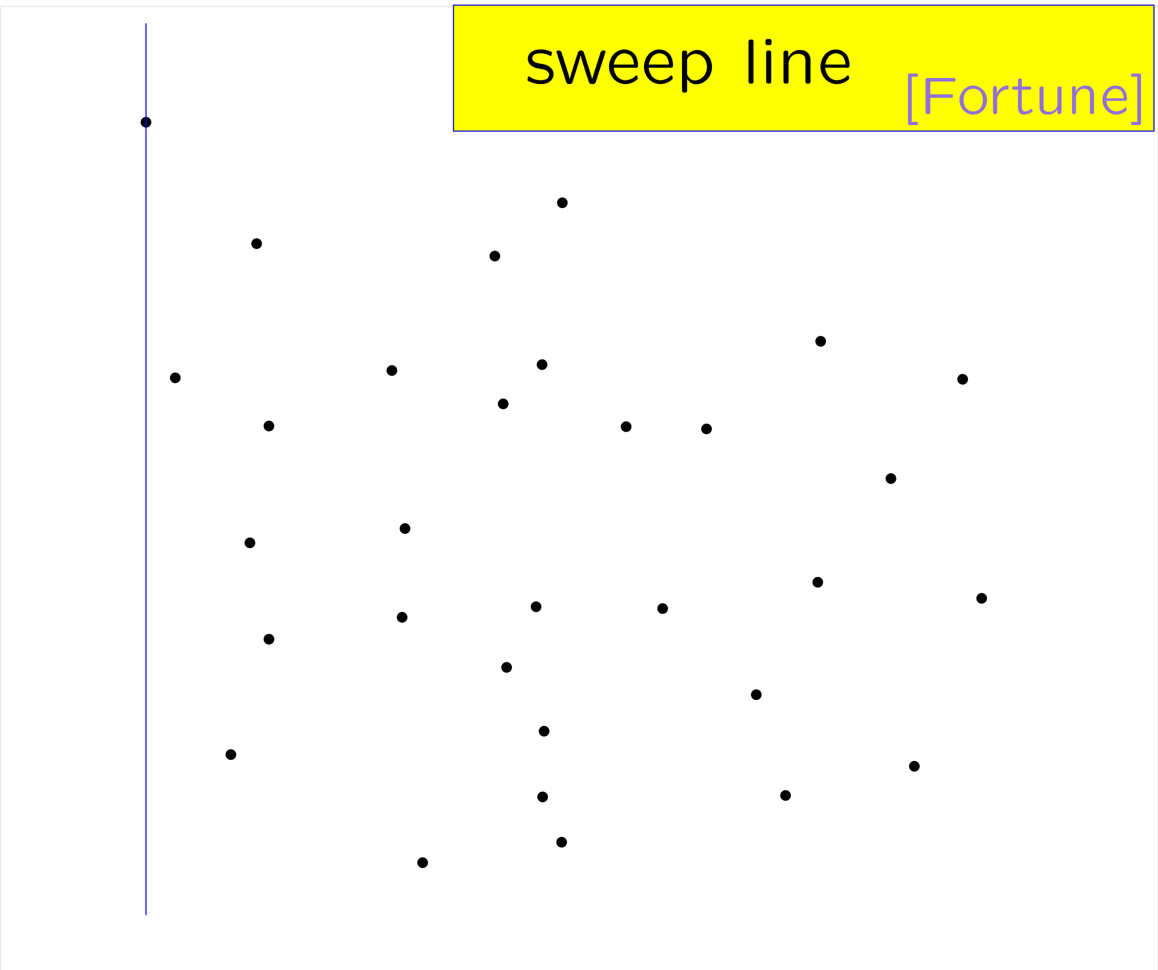
$\boxed{O(n\sqrt{n})}$



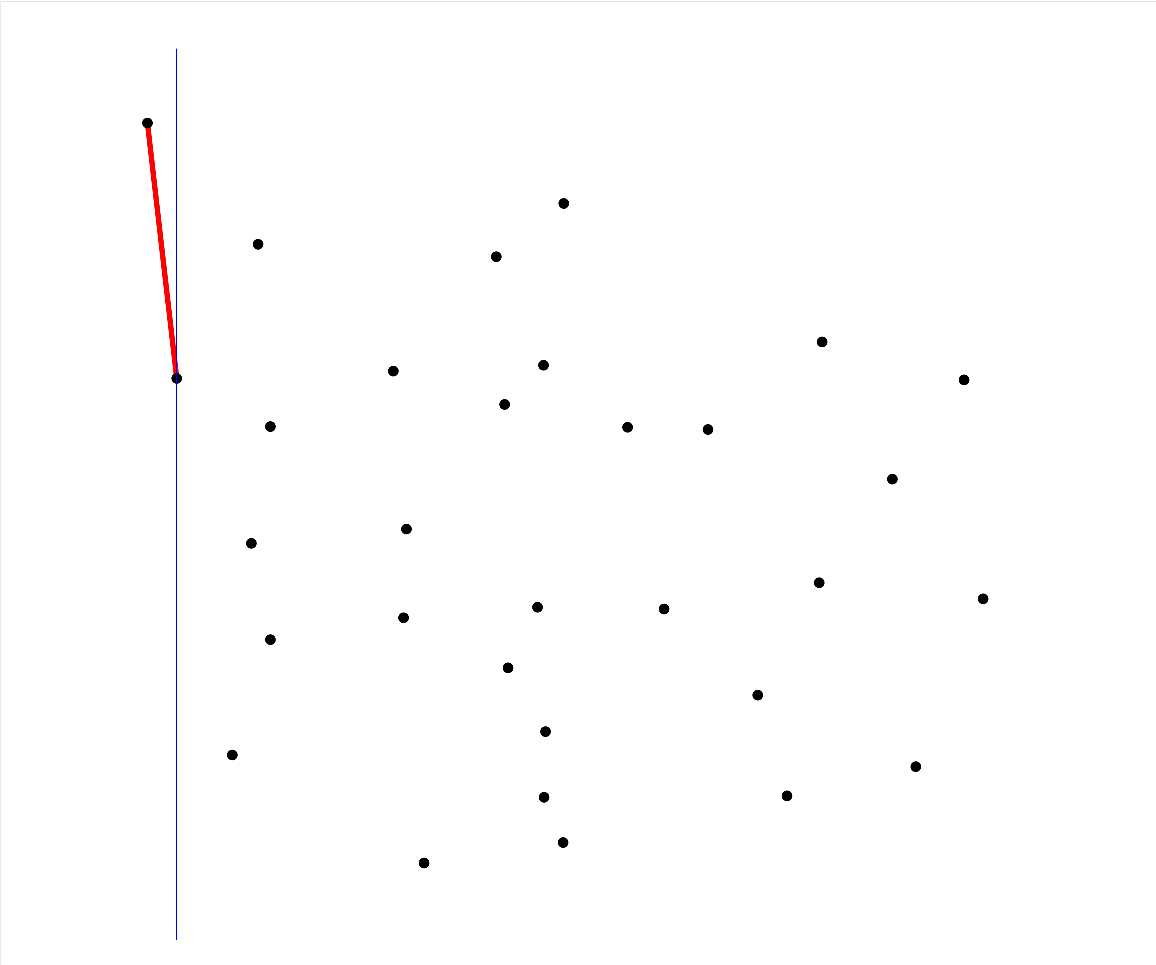
Classical Computational Geometry Solutions

find a $O(n \log n)$ worst case algorithm

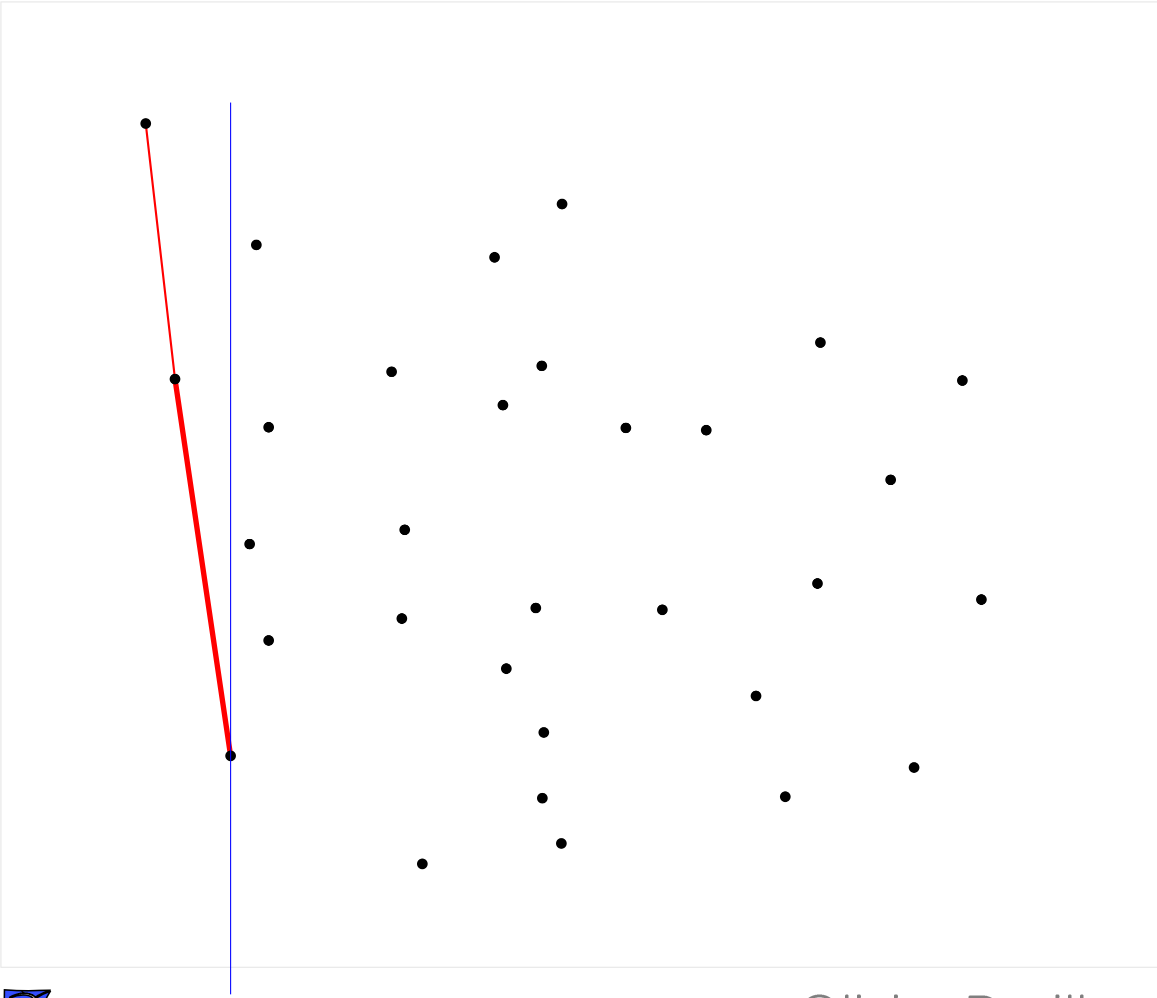
- sweep-line
- divide and conquer

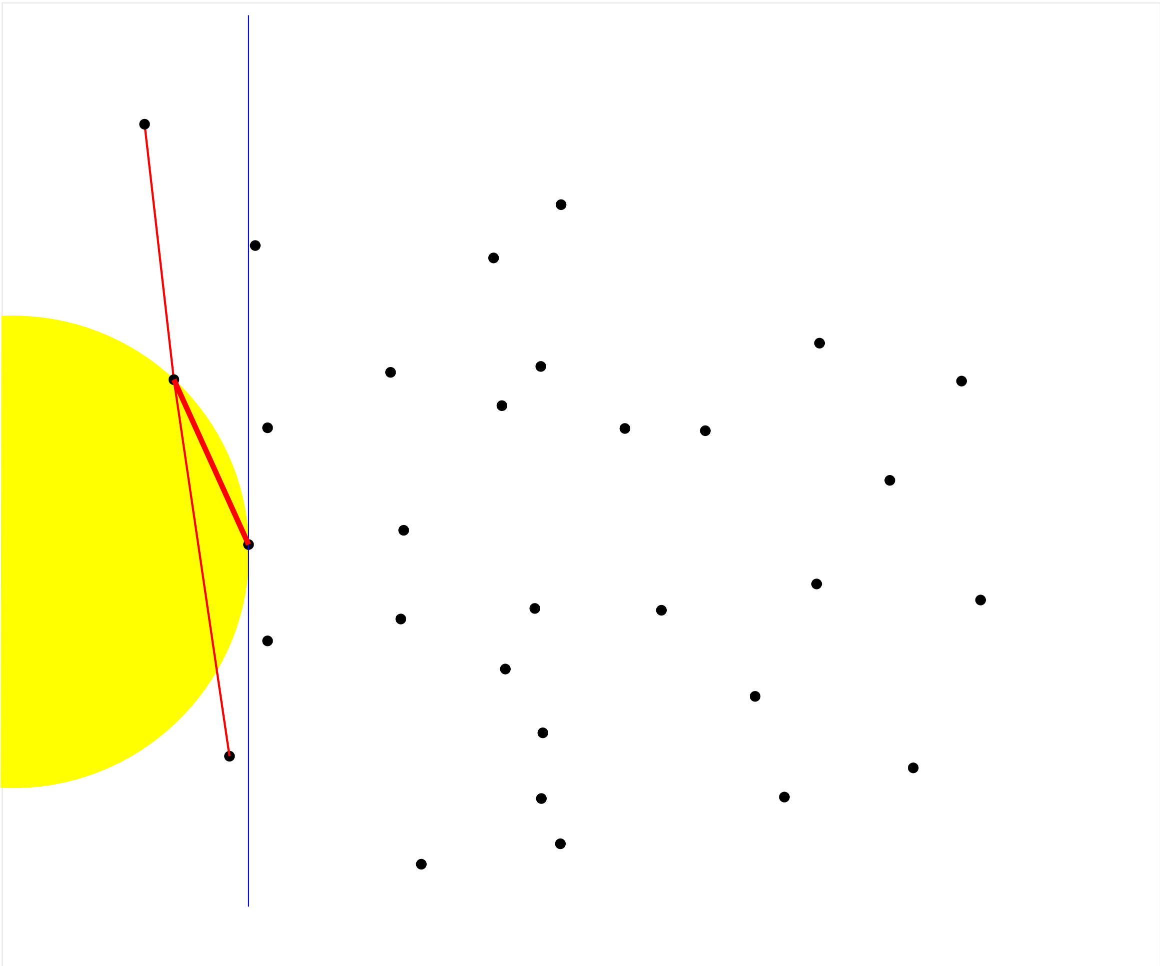

sweep line
[Fortune]

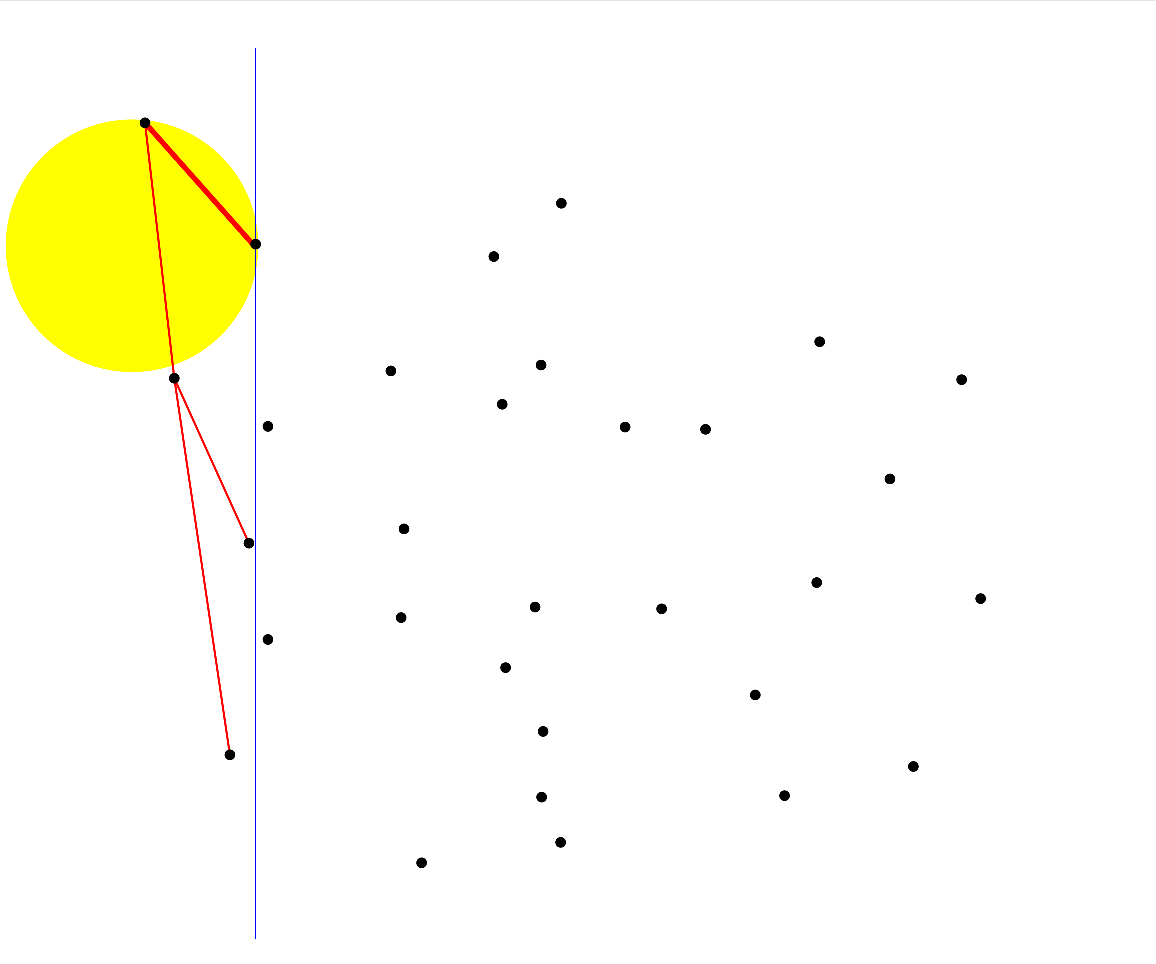
Delaunay triangulation

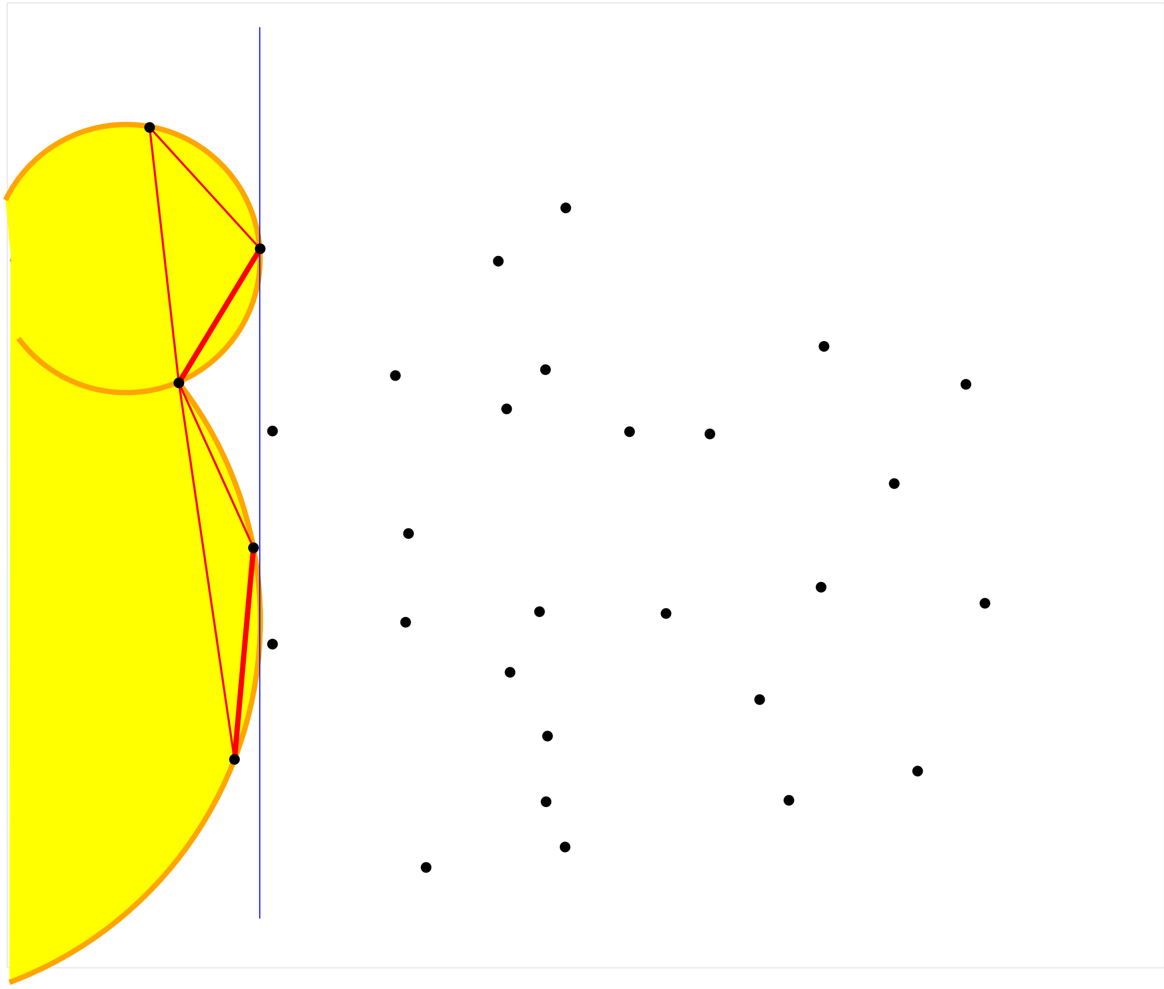


Olivier Devillers

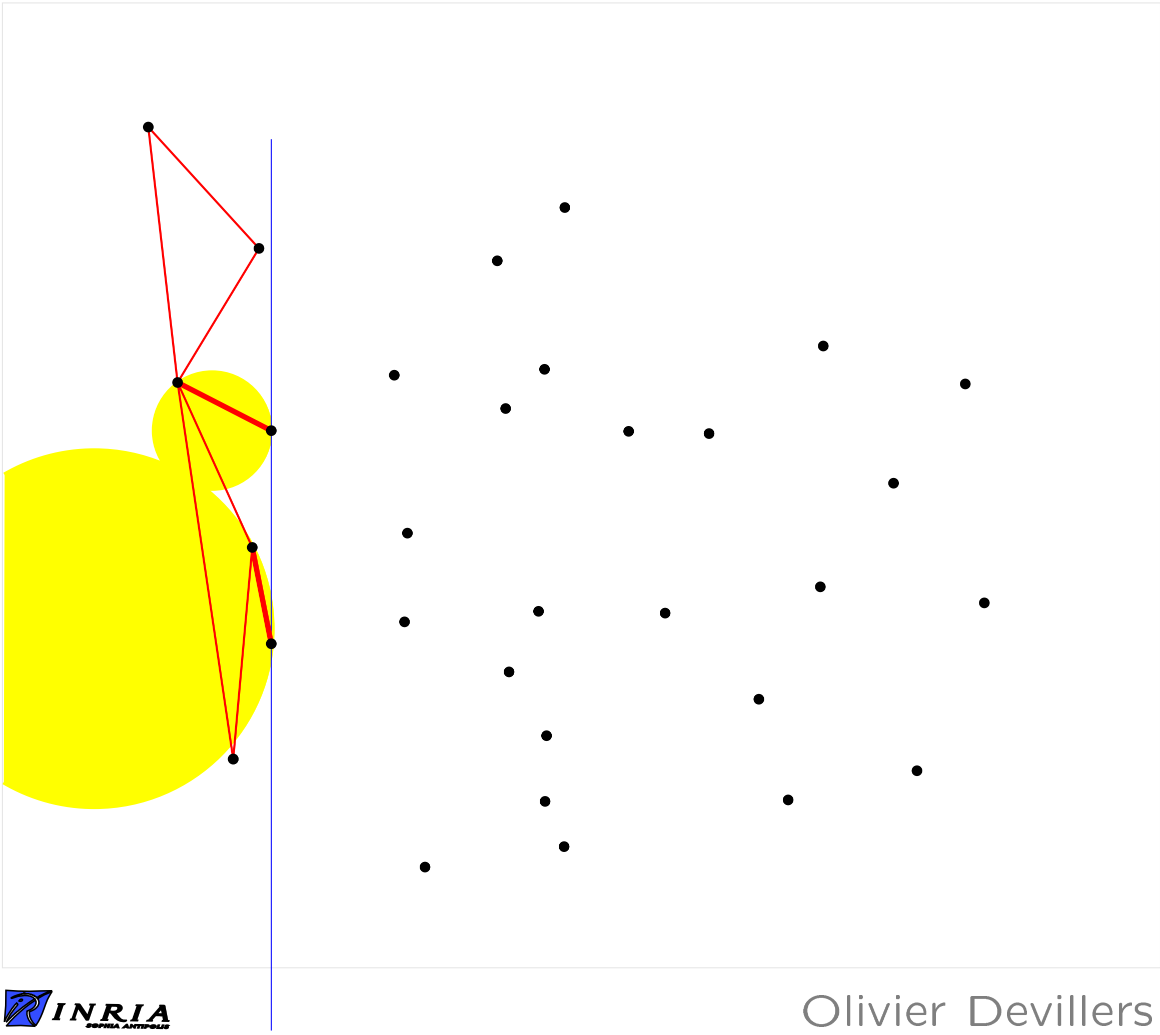






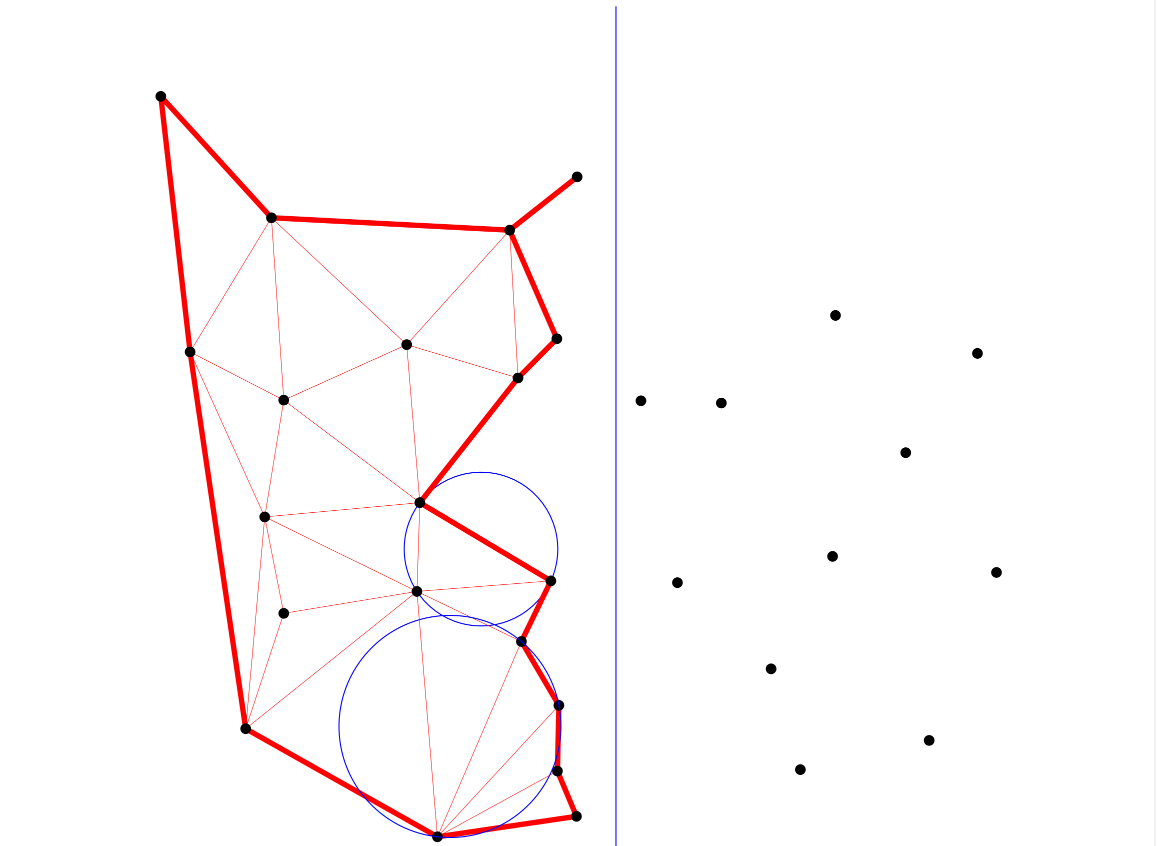


Delaunay triangulation



INRIA SOPHIA ANTIPOLIS

Olivier Devillers



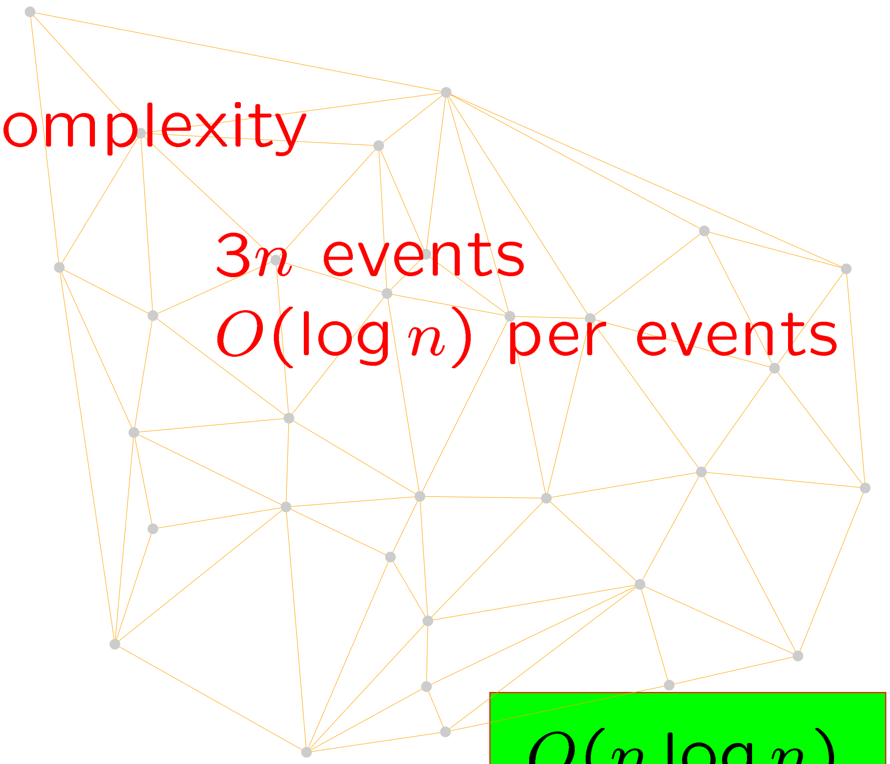
certified triangles and edges

empty circle to the left of the line

Complexity

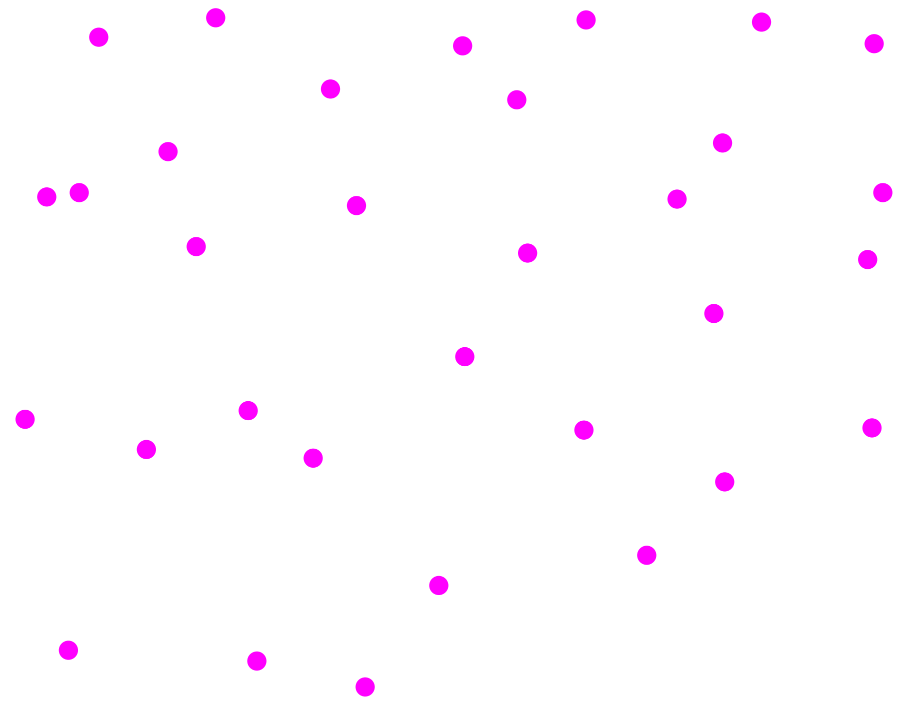
$3n$ events

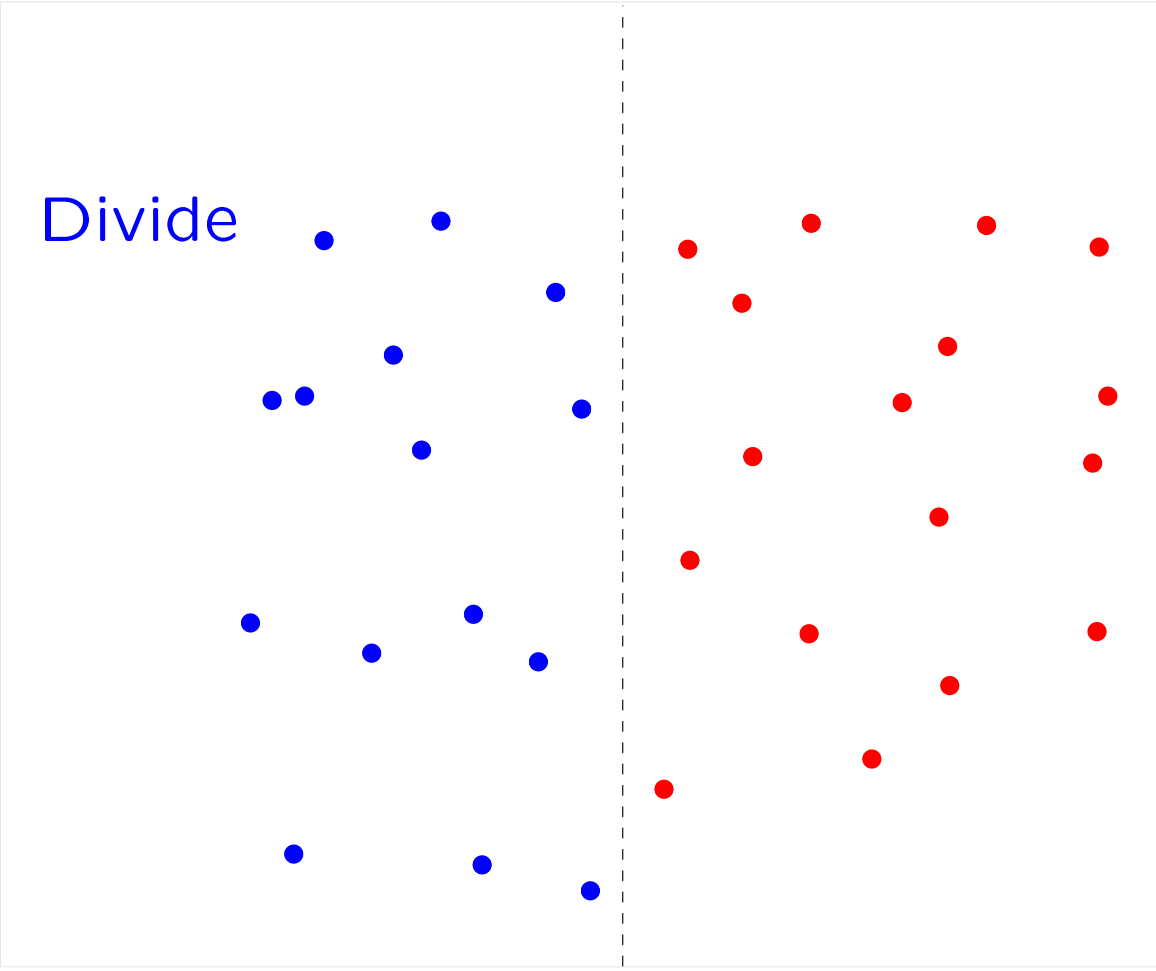
$O(\log n)$ per events



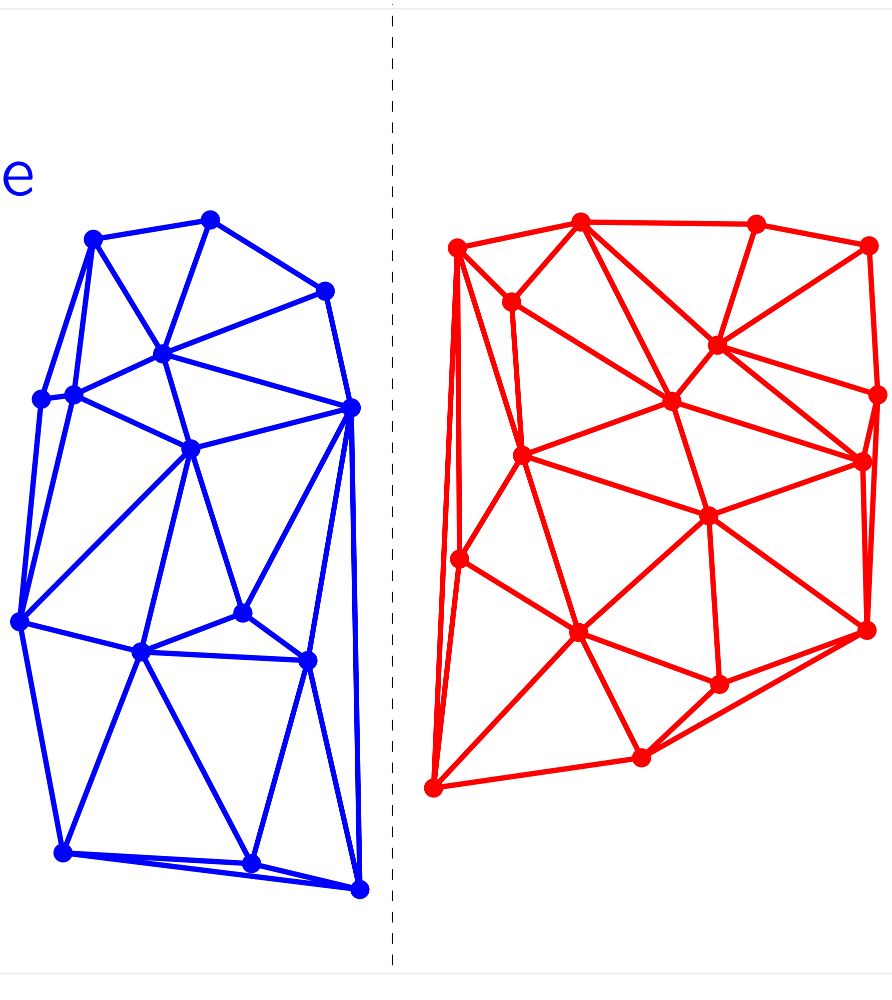
$O(n \log n)$

Divide and conquer [Lee]

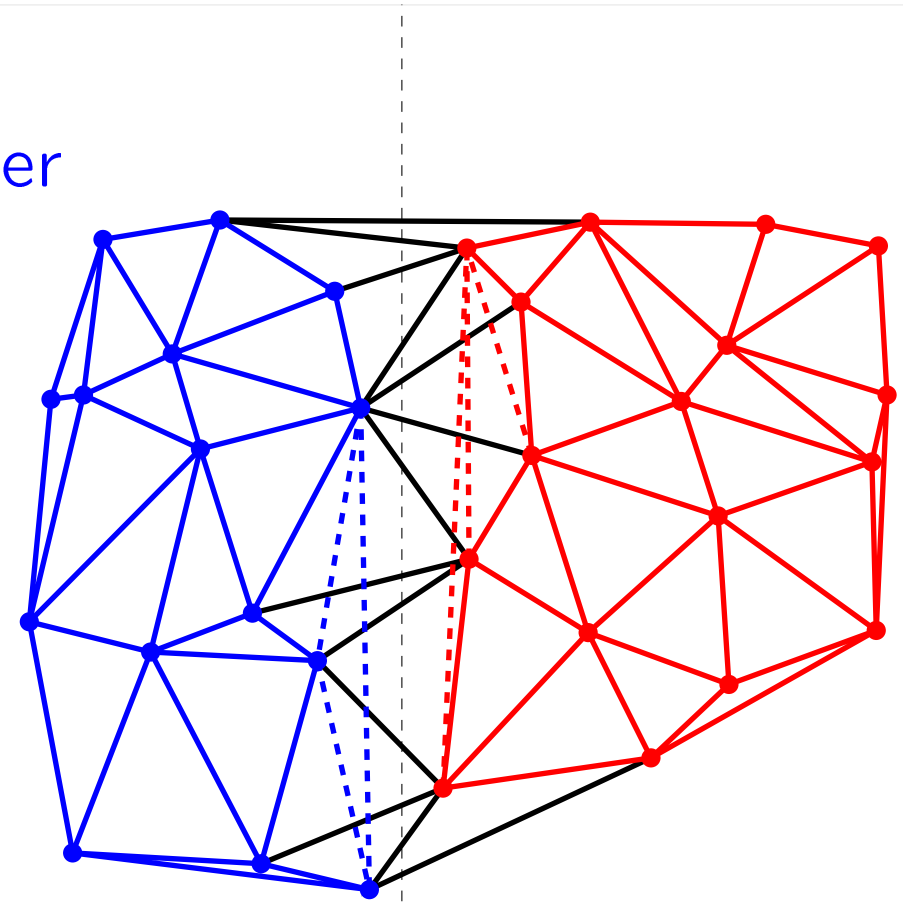


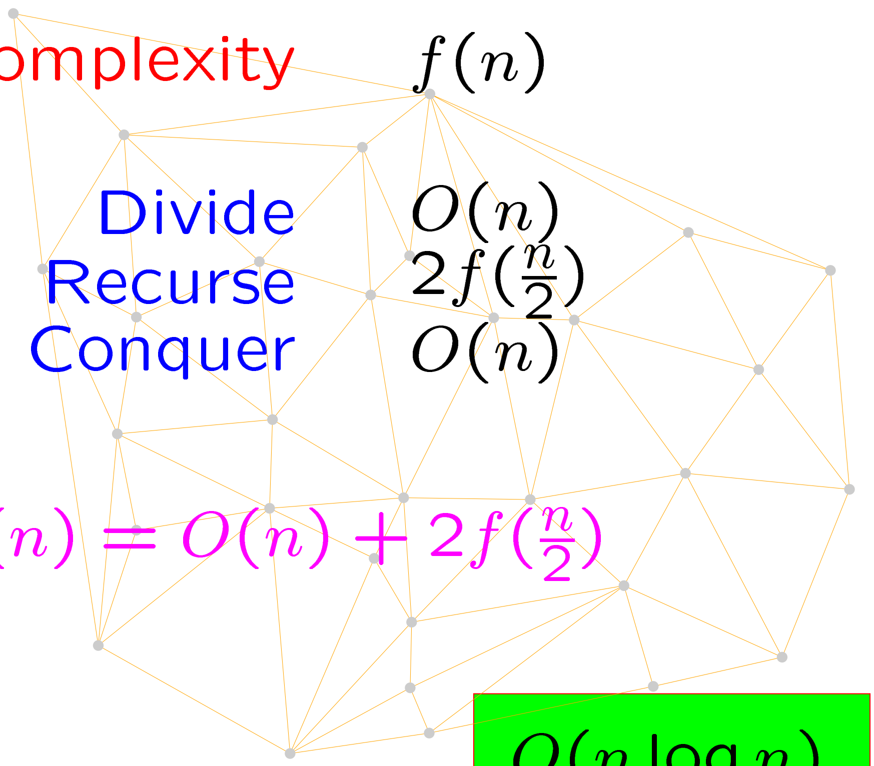

Divide

Recurse



Conquer





Complexity $f(n)$

Divide $O(n)$

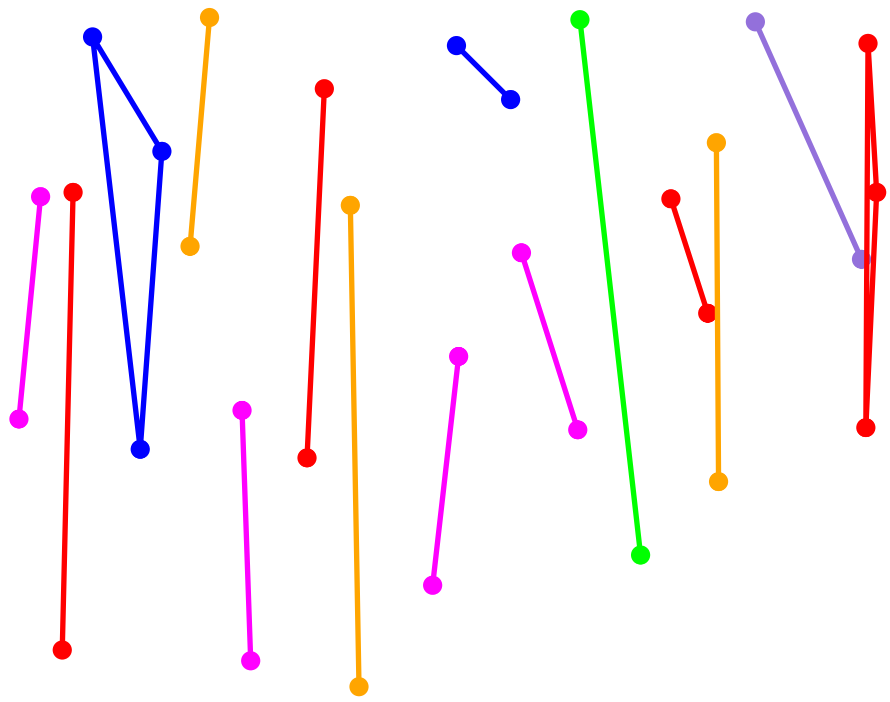
Recurse $2f(\frac{n}{2})$

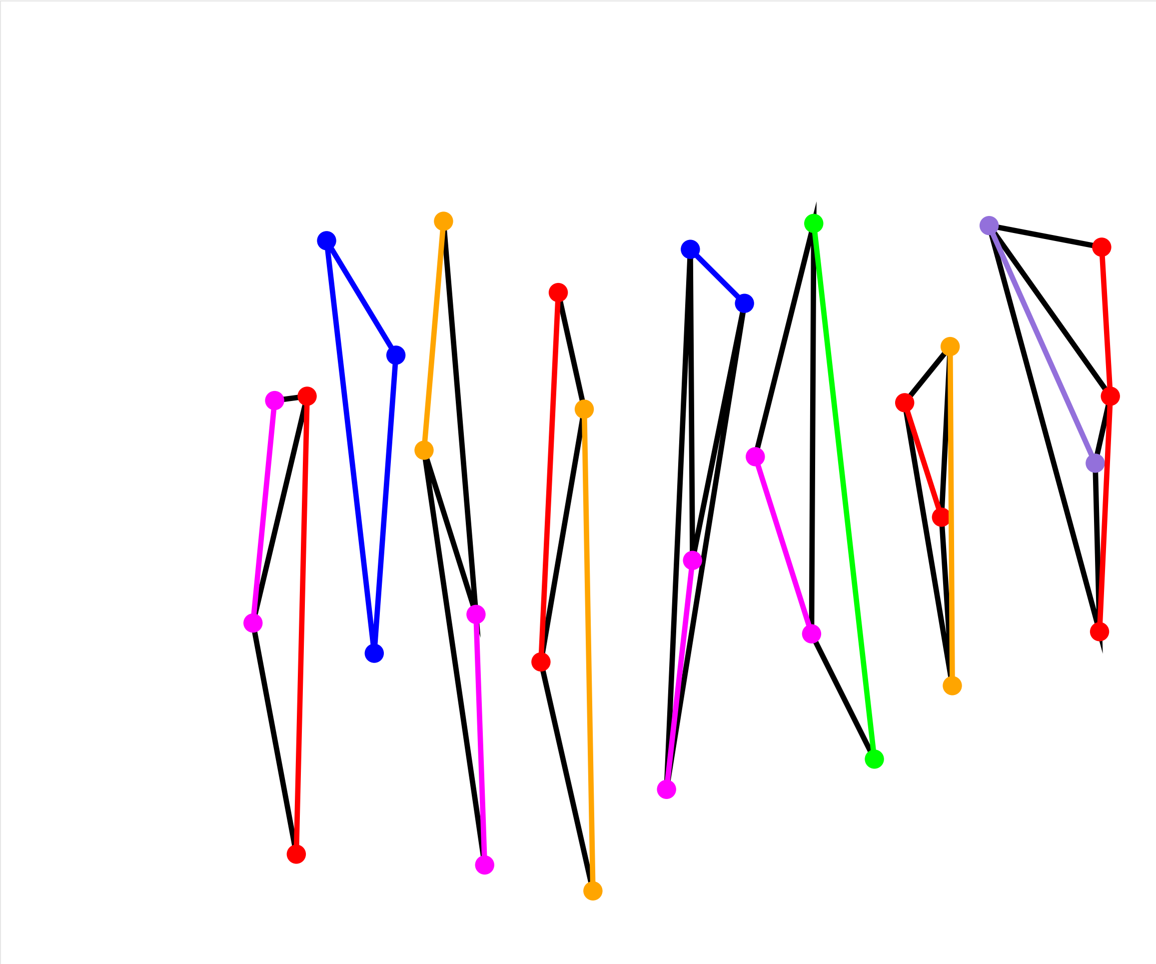
Conquer $O(n)$

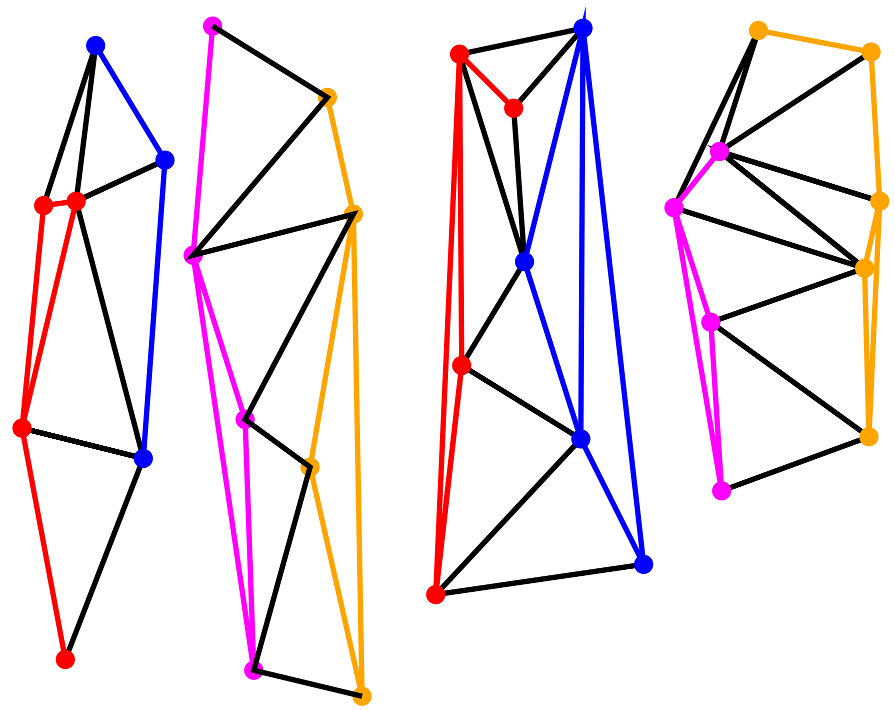
$$f(n) = O(n) + 2f(\tfrac{n}{2})$$

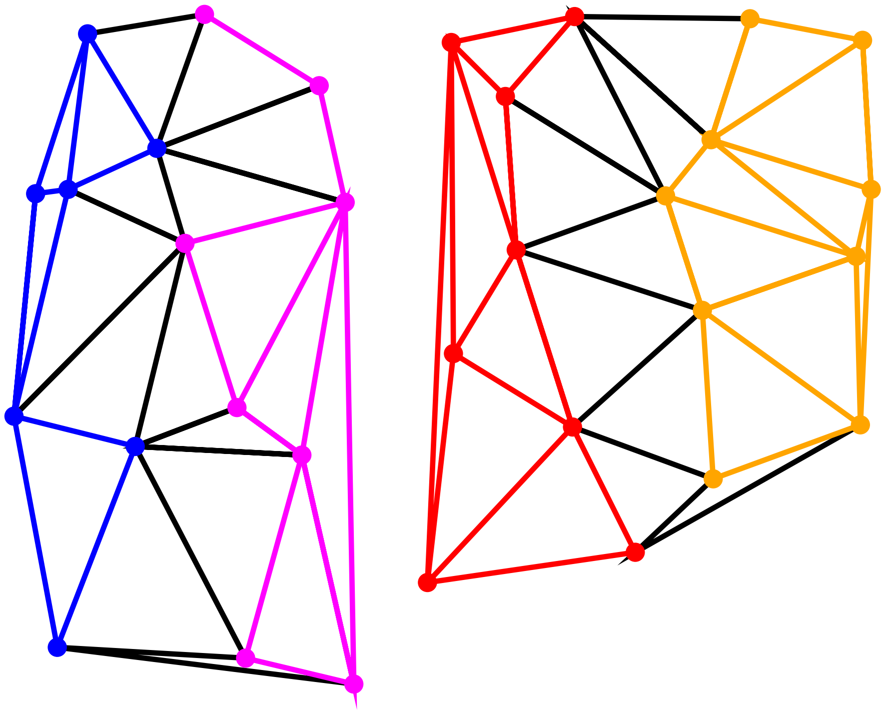
$$\boxed{O(n \log n)}$$

Dividing scheme

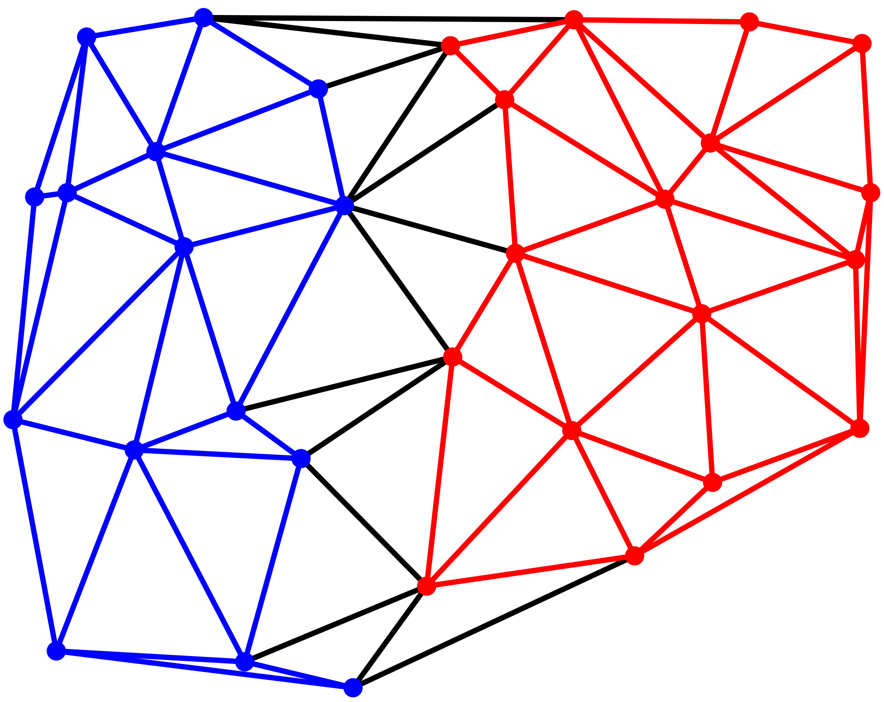






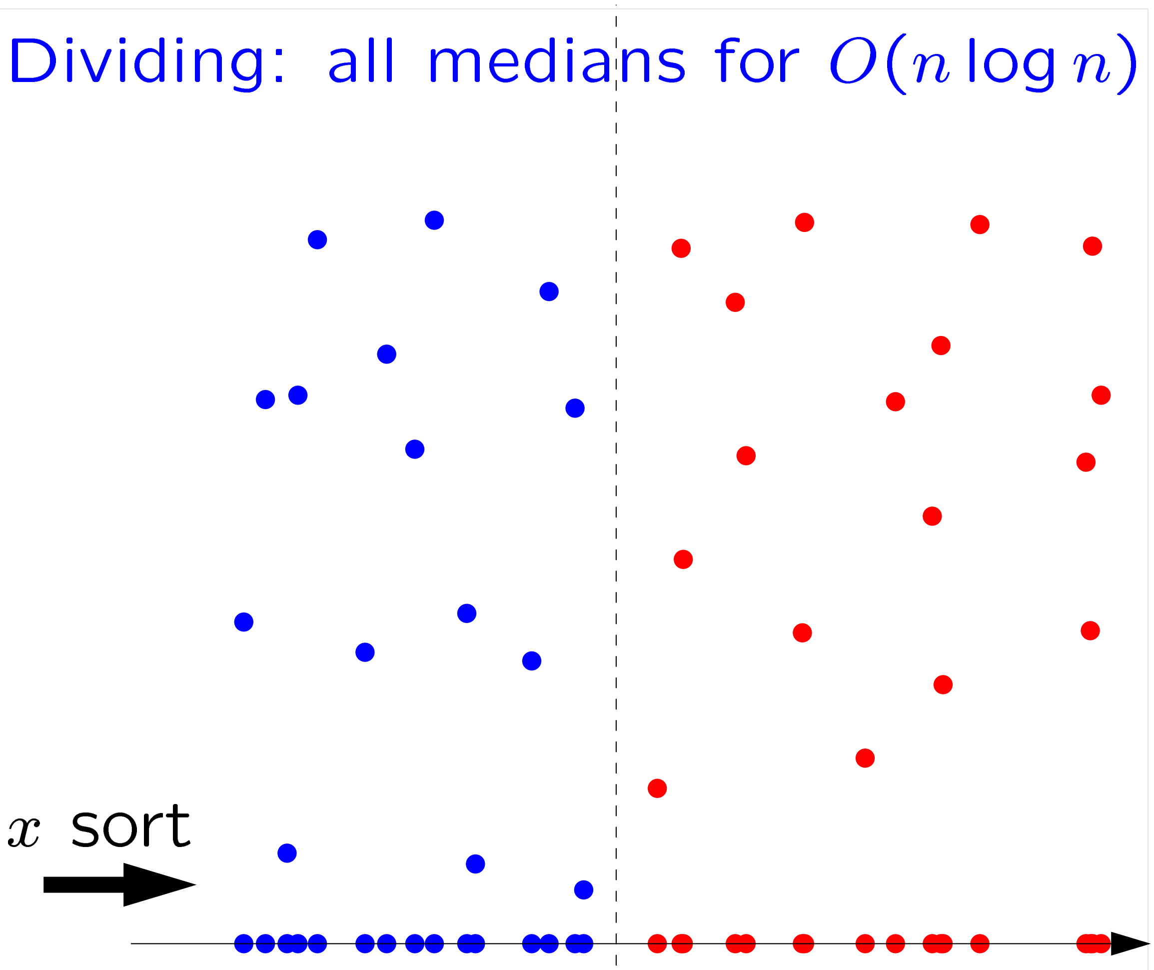


Delaunay triangulation

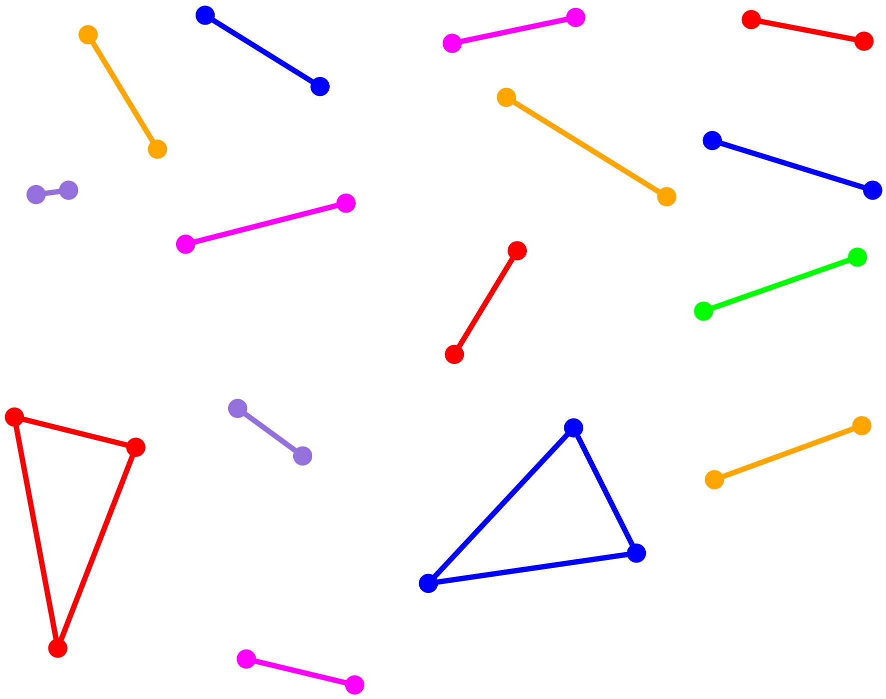


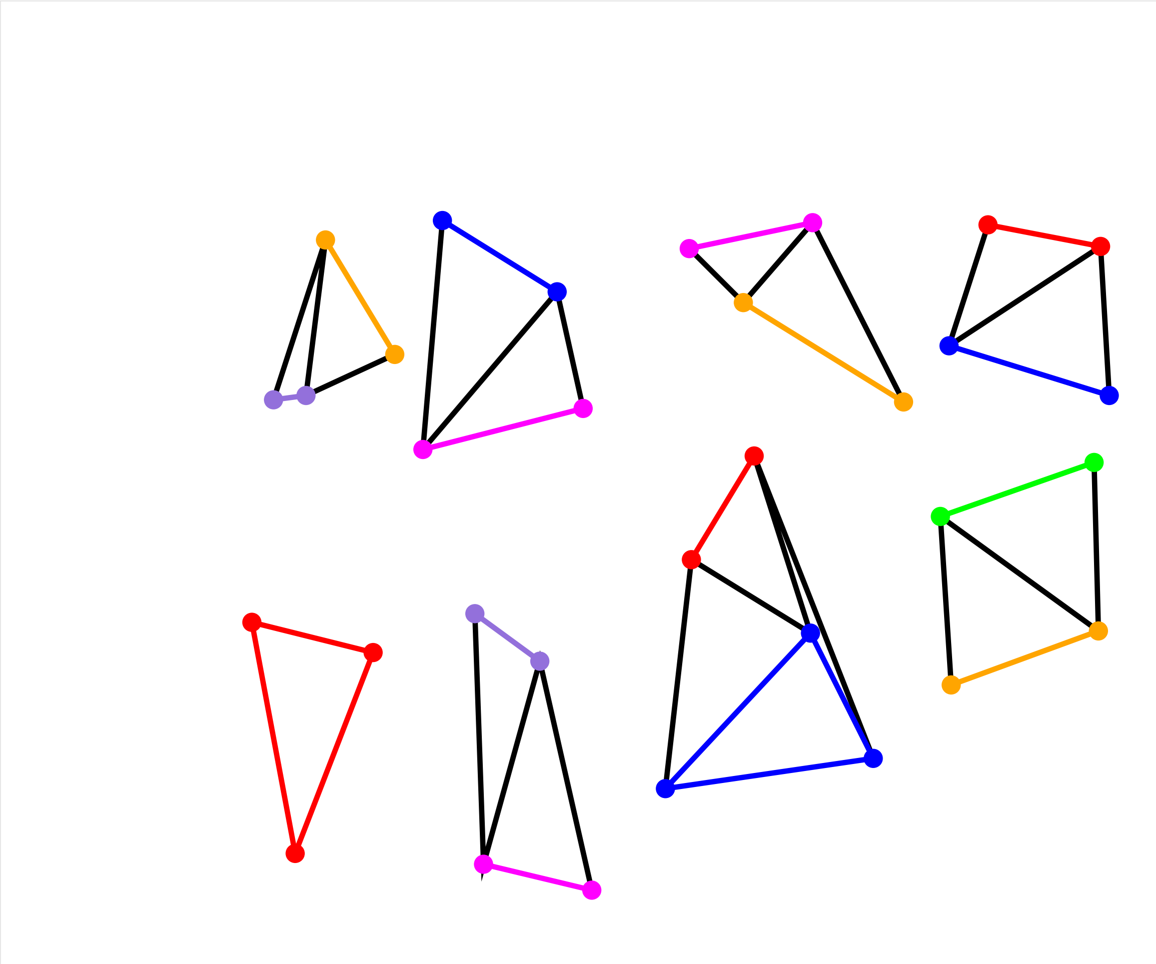
Olivier Devillers

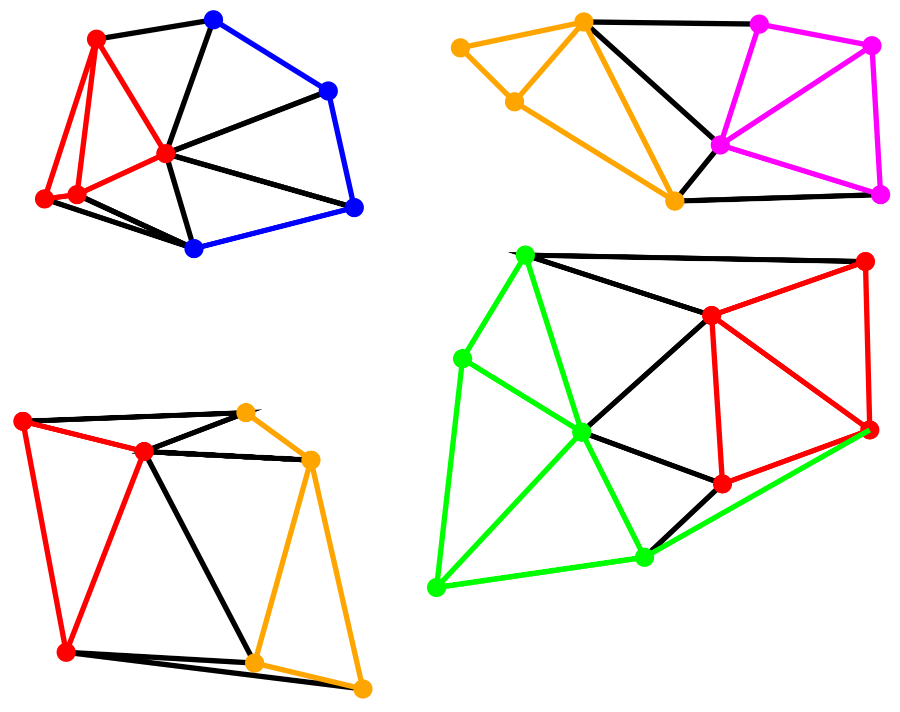
Dividing: all medians for $O(n \log n)$



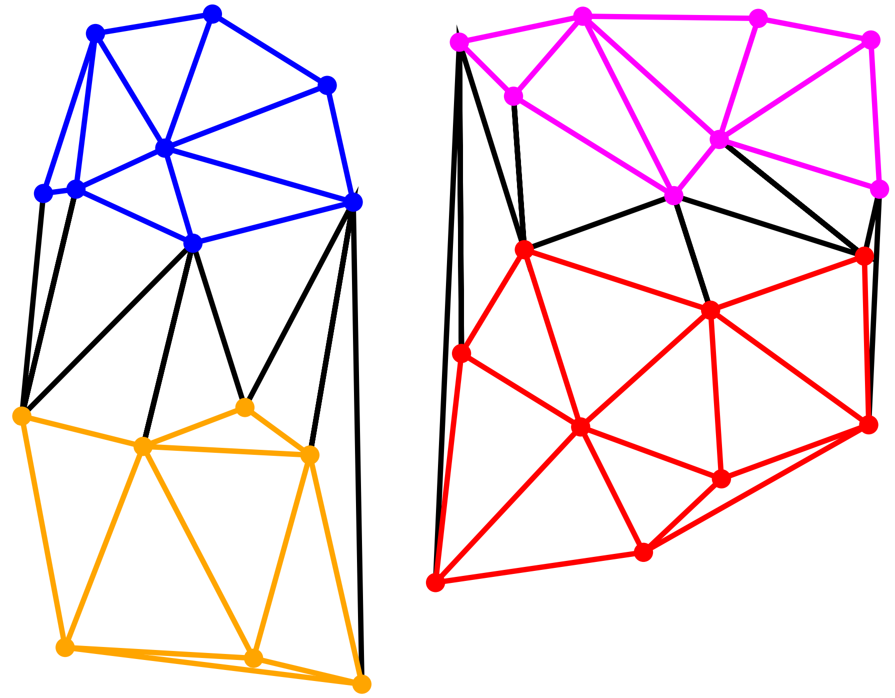
Dividing scheme



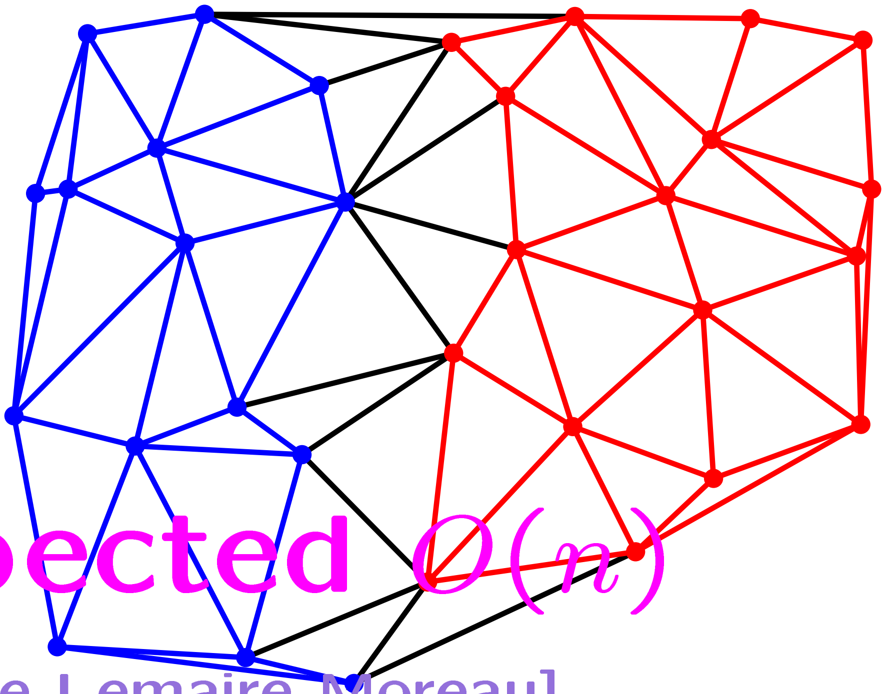




Delaunay triangulation



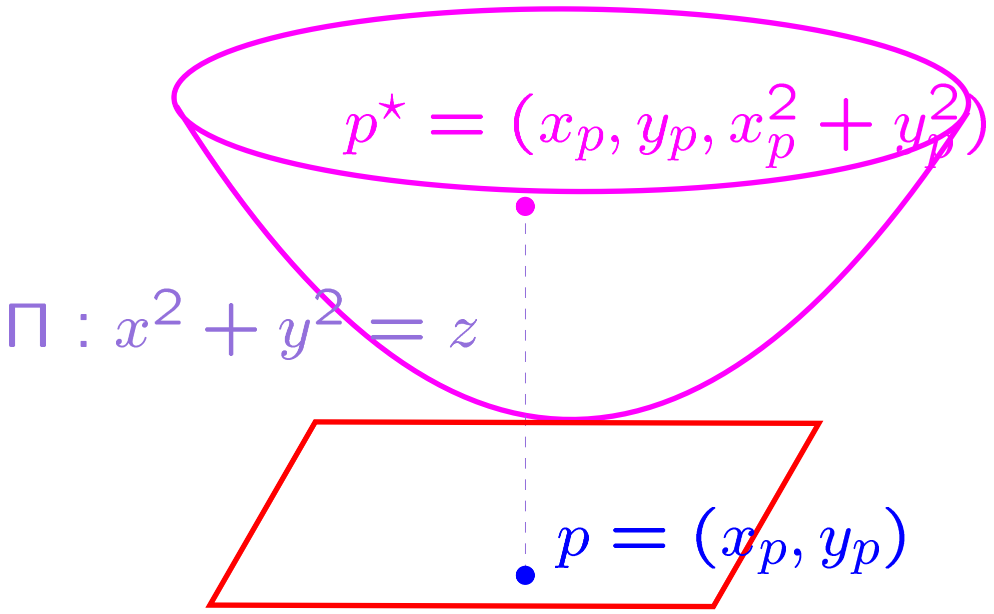
Finding the medians ?



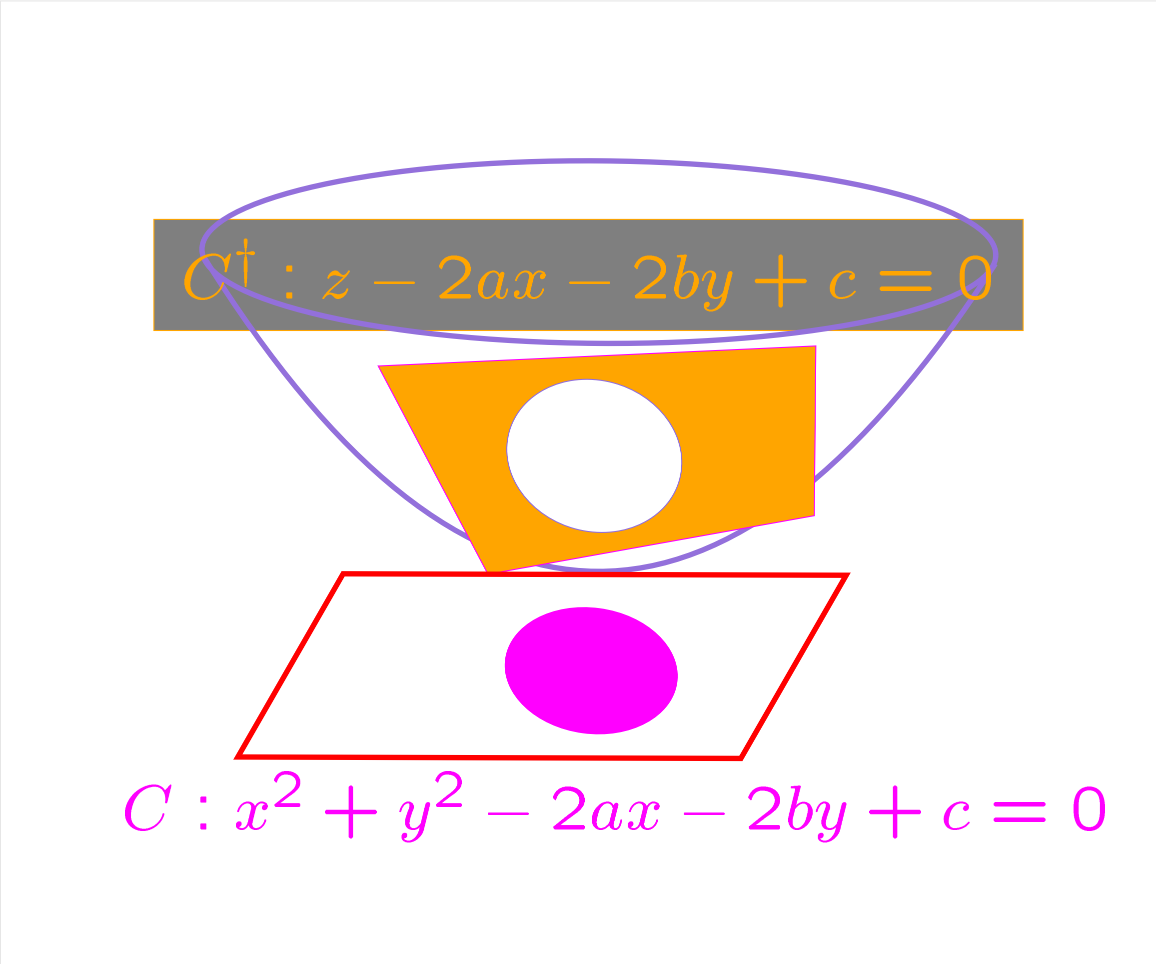
Expected $O(n)$

[Devroye Lemaire Moreau]

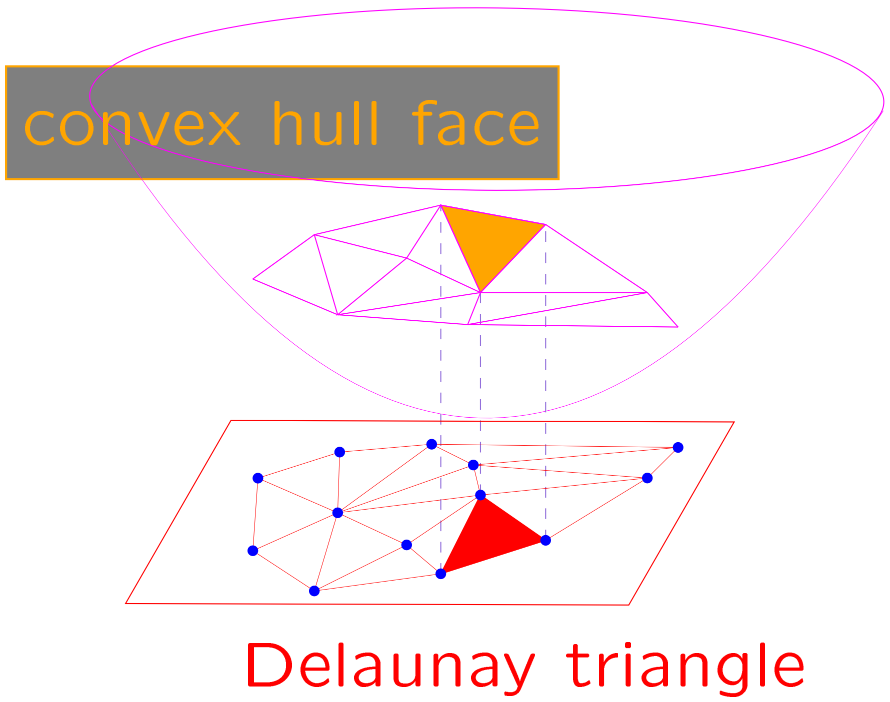
Delaunay and 3D convex hull



Project on the paraboloide



$C^\dagger : z - 2ax - 2by + c = 0$
$C : x^2 + y^2 - 2ax - 2by + c = 0$



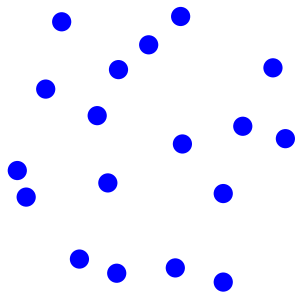
Convex hull algorithms apply !

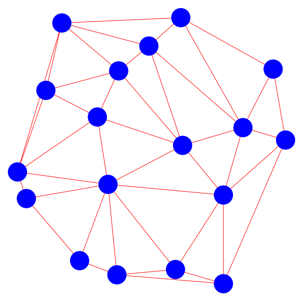
More practical Algorithms

Randomisation

Randomized algorithms

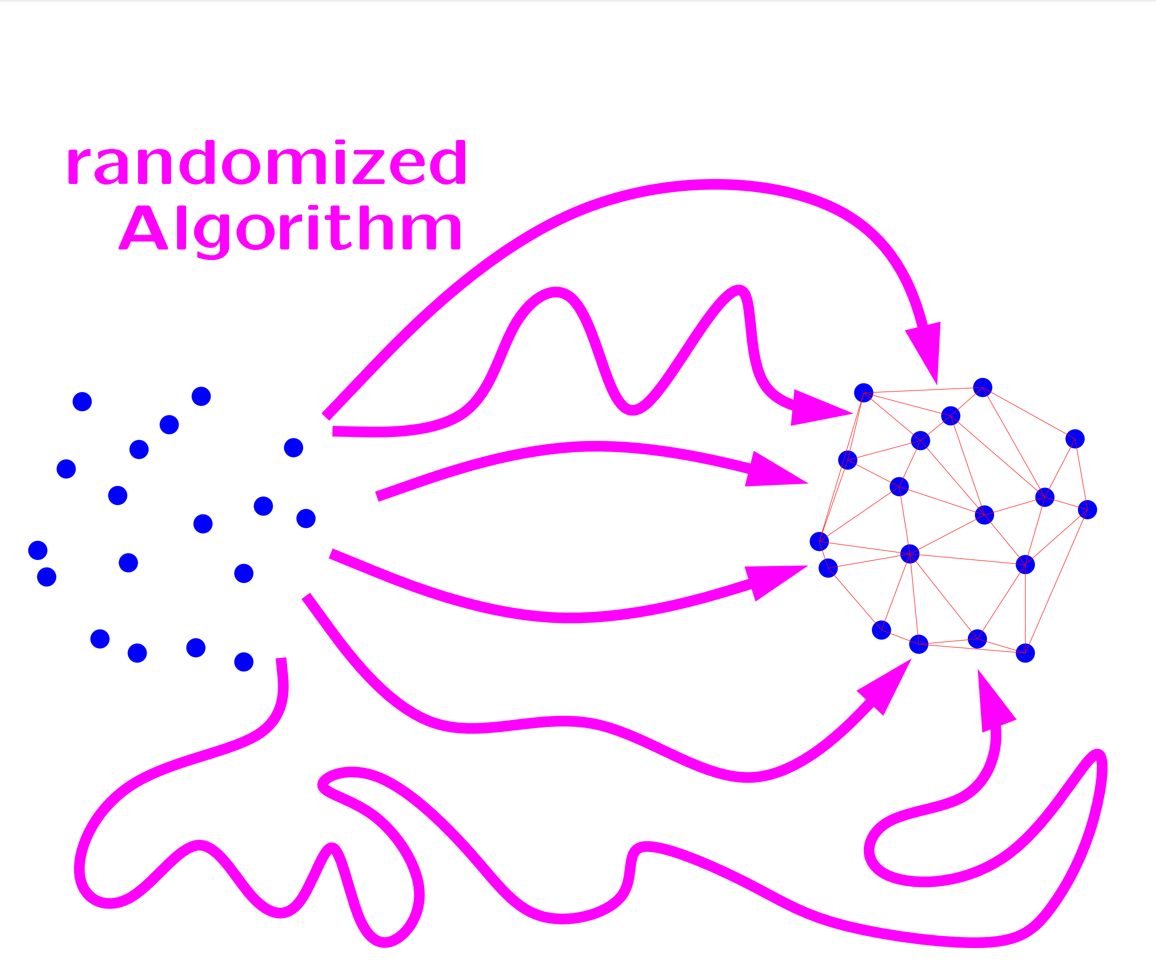
simpler

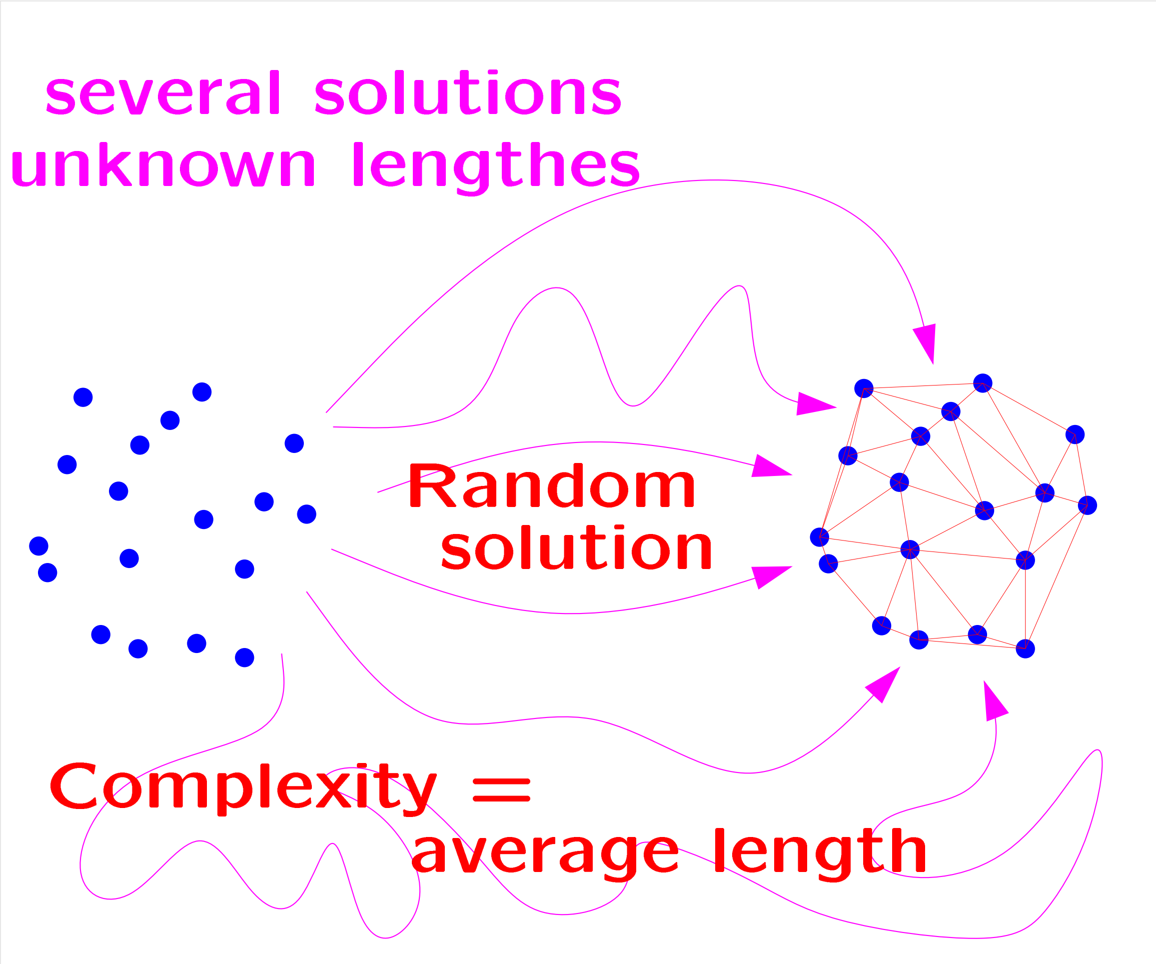




non deterministic


deterministic
Algorithm


randomized
Algorithm


several solutions
unknown lengthes
Random
solution
Complexity =
average length

How introduce randomness ?

divide and conquer

random sampling

incremental

random order

Randomized incremental constructions

Analyze the cost

of the last point

of a random point

Backward analysis [Seidel]

Incremental algorithm

two stages

Location

where is the new point ?

Insertion

how to retriangulate

$$\boxed{O(degree(p))}$$

Incremental algorithm

two stages

Location

where is the new point ?

Insertion

how to retriangulate

$O(degree(p))$

$O(1)$

Incremental algorithm

two stages

Location

where is the new point ?

Location structure

Insertion

how to retriangulate

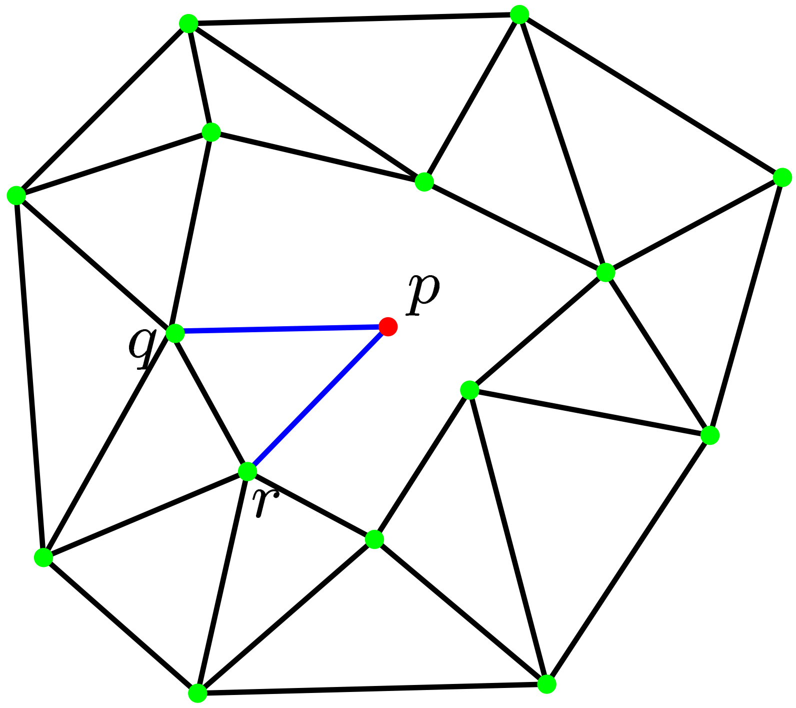
Delaunay tree [Boissonnat Teillaud]

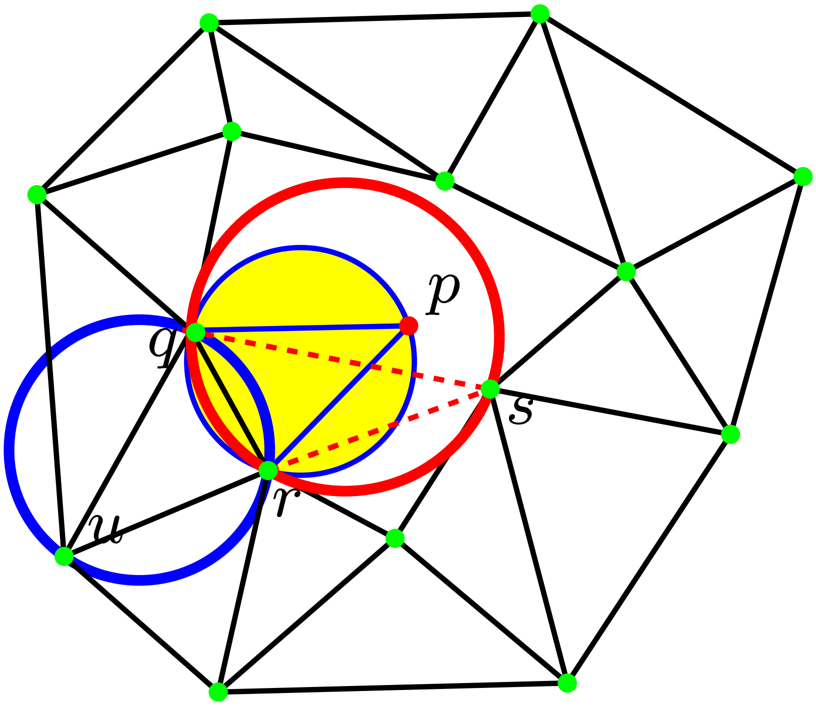
Remember the history
of the incremental
construction

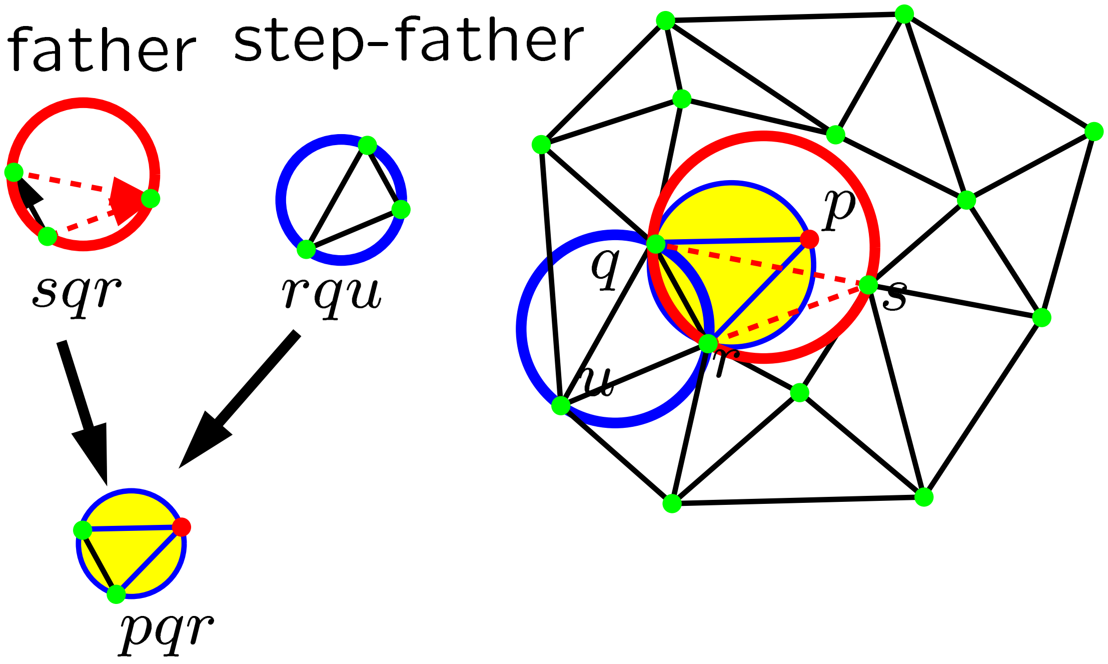
Locate a point p =
find all triangles conflicting p

which are Delaunay

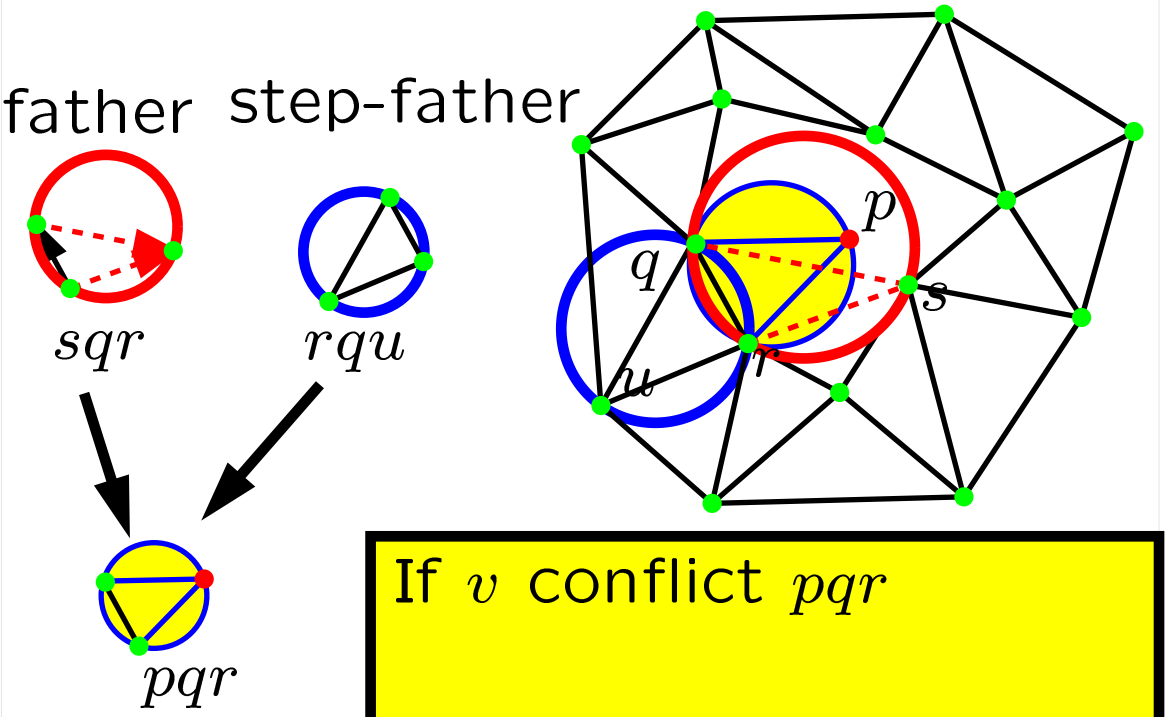
which have been Delaunay


p
q
r


p
q
s
r
u

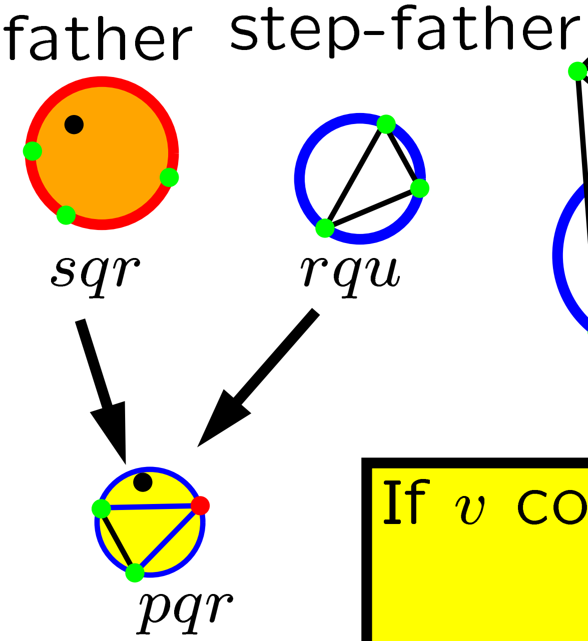

father
step-father
sqr
rqu
pqr
p
q
s
u
r

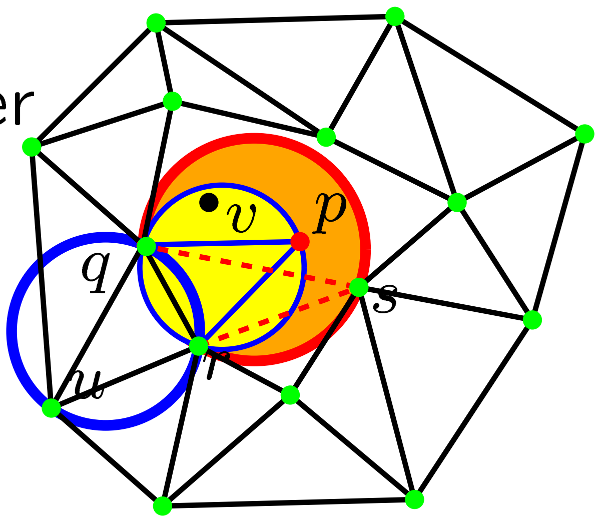
locating v



If v conflict pqr

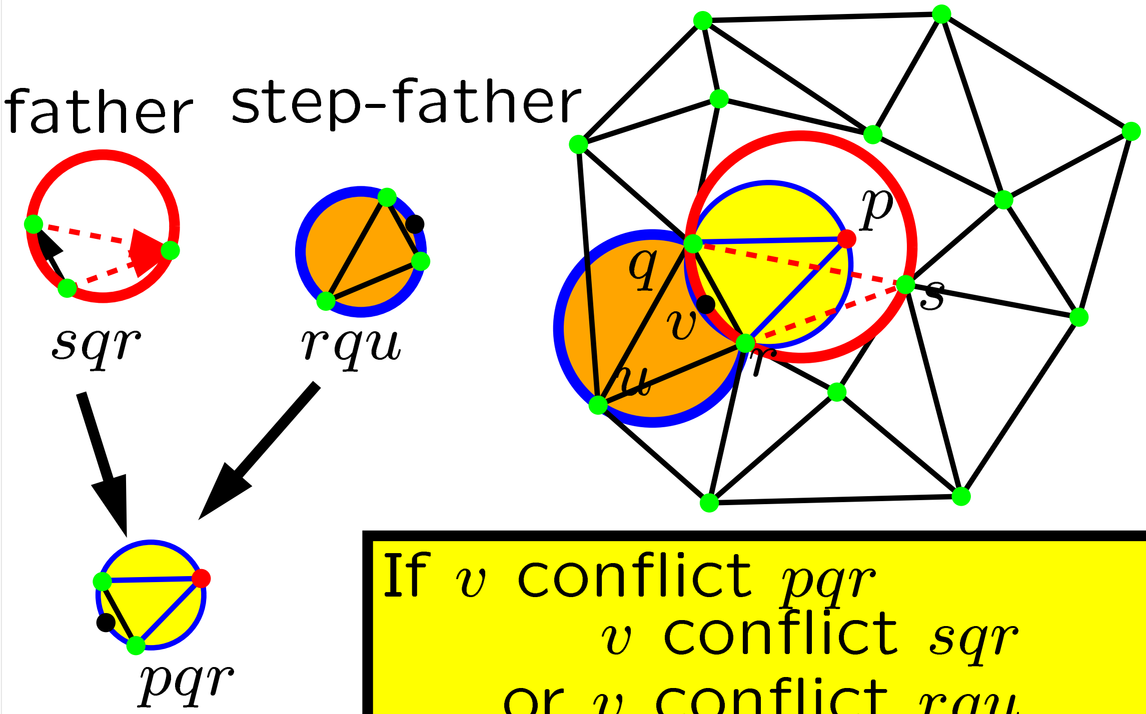
locating v





If v conflict pqr
v conflict sqr

locating v



If v conflict pqr
v conflict sqr
or v conflict rqu

Locate a point p =
find all triangles conflicting p

which are Delaunay

which have been Delaunay

Count them!

triangle conflicting x_n
created by the k^{th} point x_k

$x_k x_n$ is an edge of
Delaunay$(x_1, \ldots, x_k, x_n)$

expected 6 edges incident to x_k
Delaunay$(x_1, \ldots, x_k, x_n)$

$\boxed{\sum_k \frac{6}{k} = \log n}$

$\boxed{O(n \log n)}$

history of insertions

Delaunay tree

[Boissonnat Teillaud]

[Guibas Knuth Sharir]
[Cheong]
[Mehlhorn, Meiser, Klein]

$O(n \log n)$

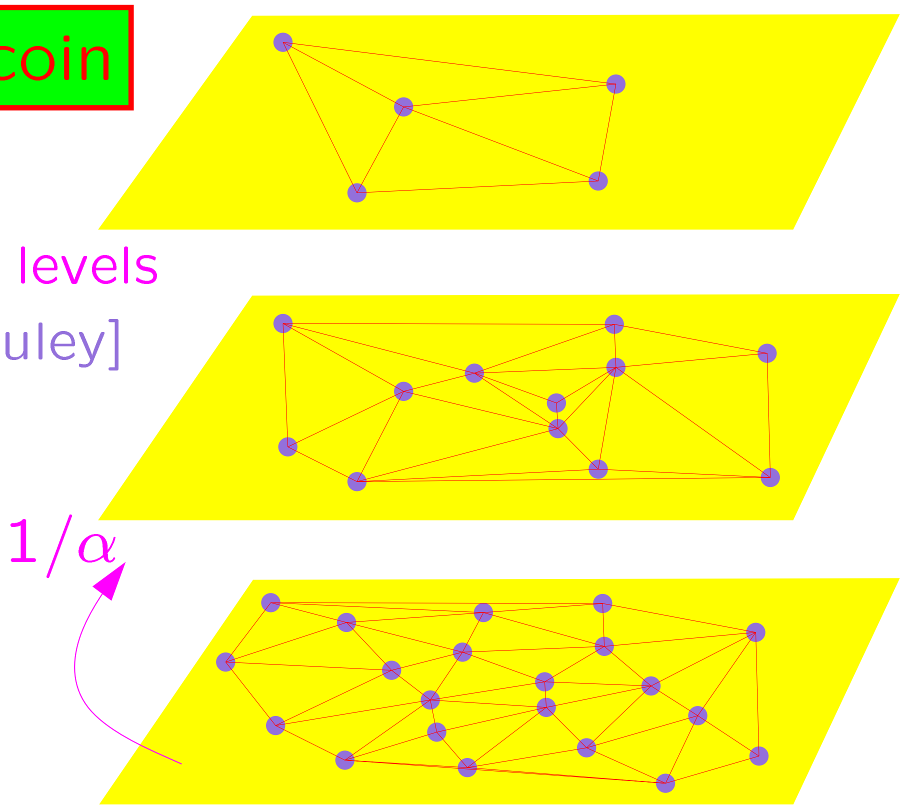
randomized
order sensitive

How introduce randomness ?

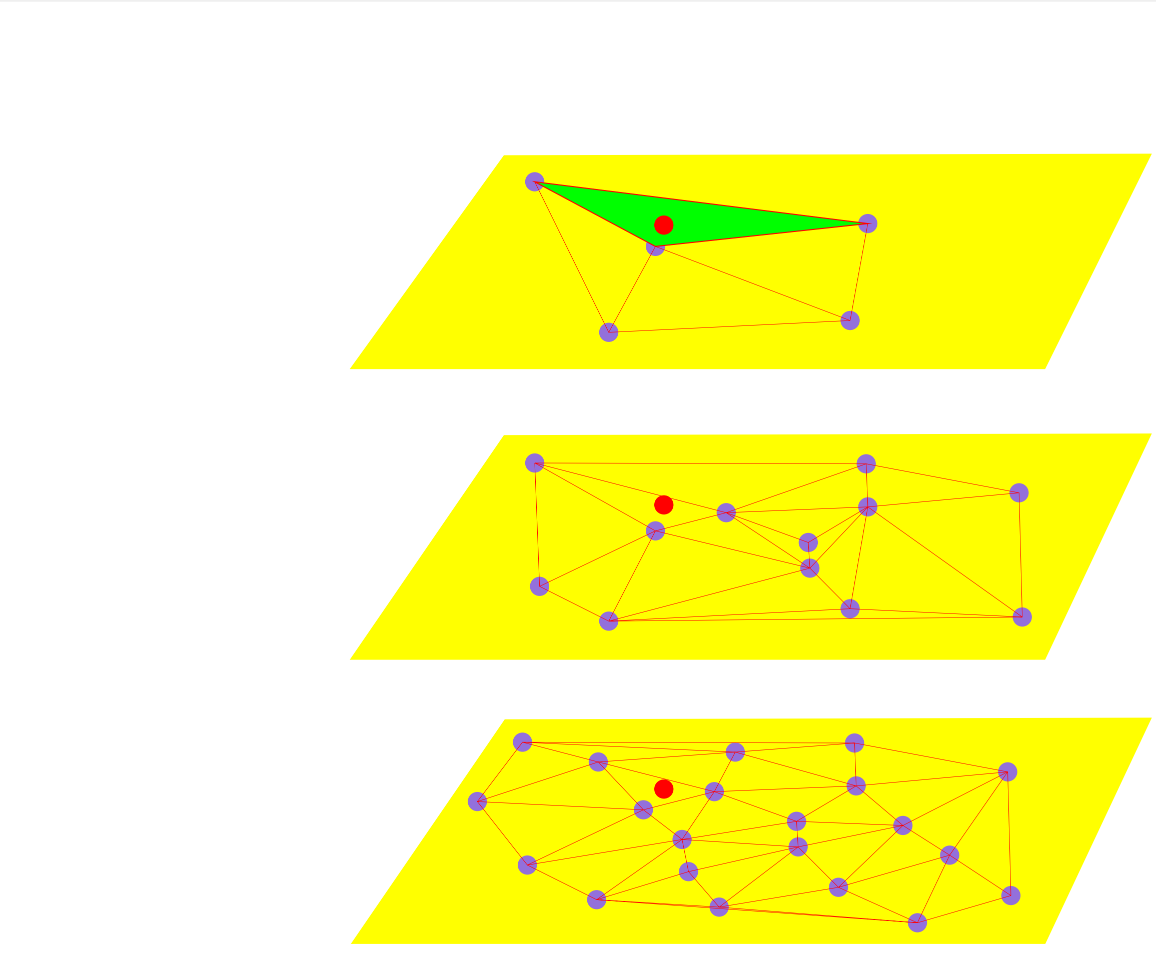
toss a coin

Bernoulli levels

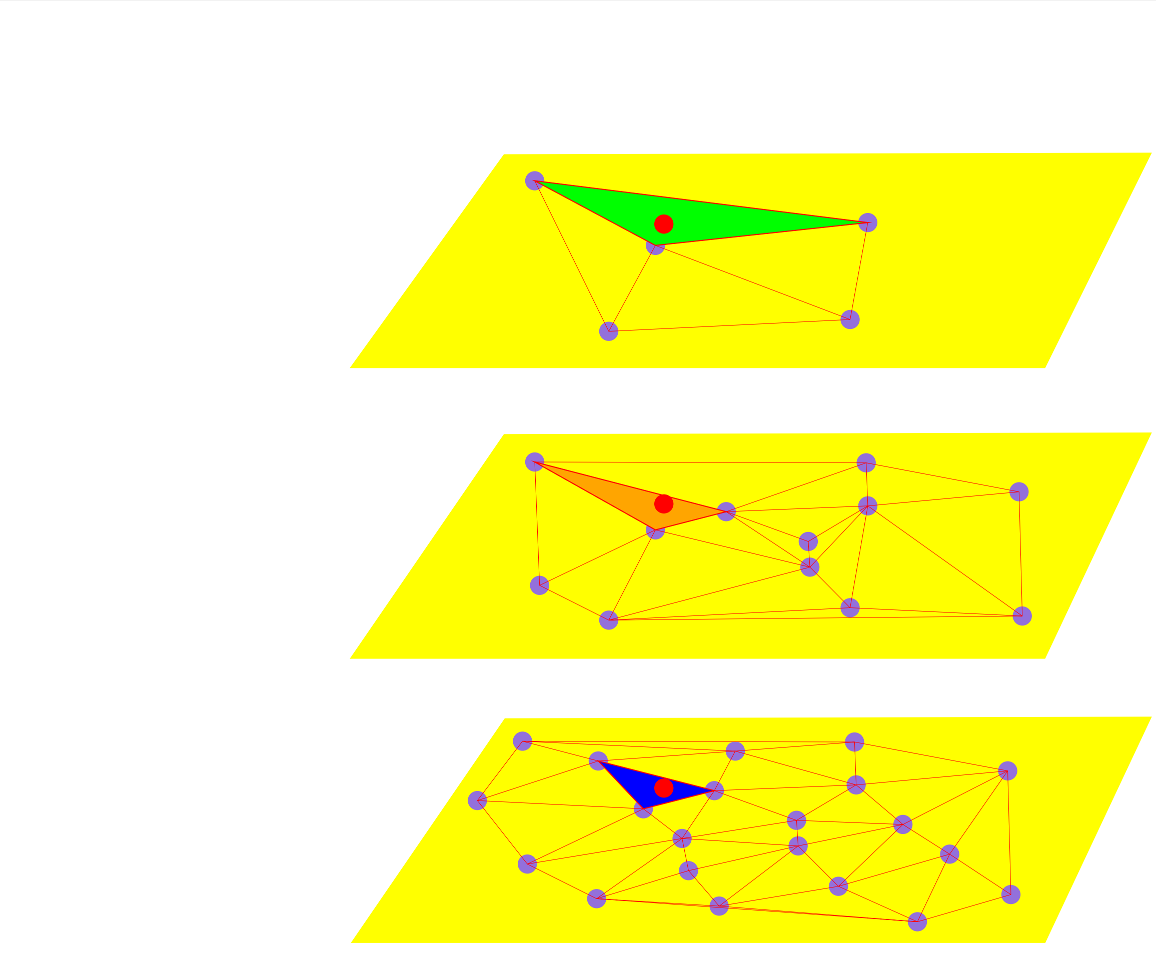
[Mulmuley]

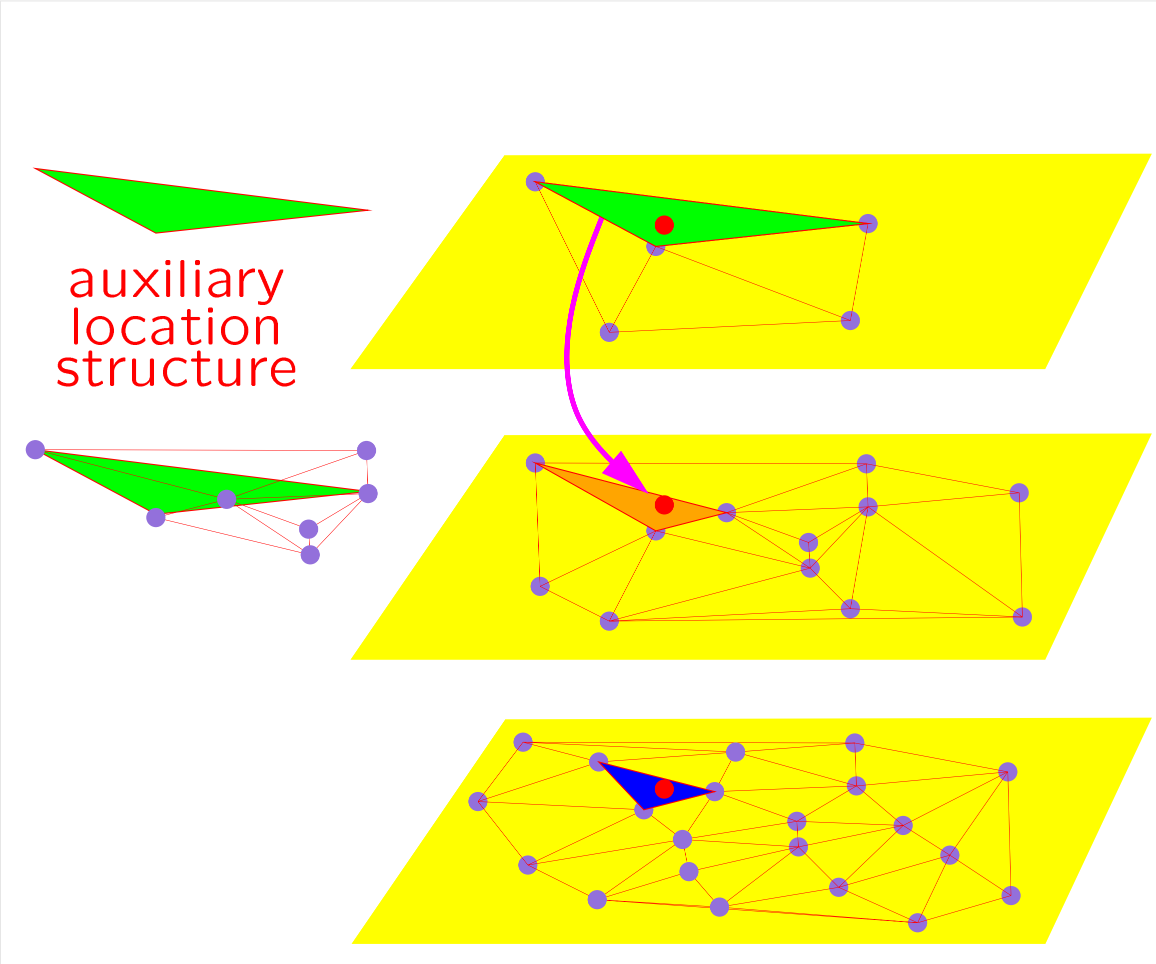


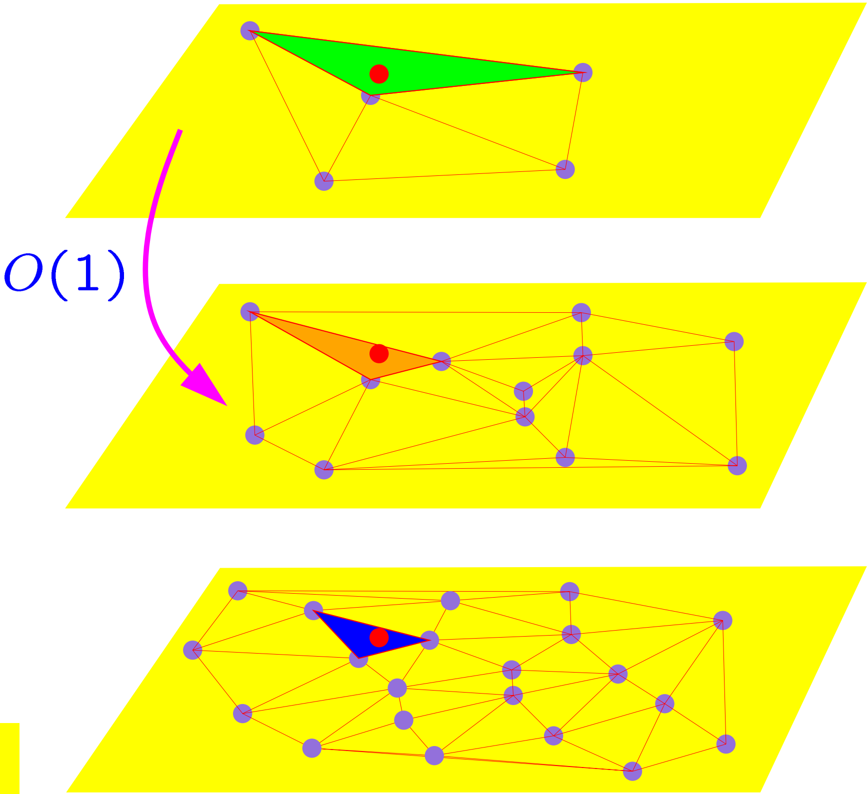
Delaunay triangulation



Delaunay triangulation



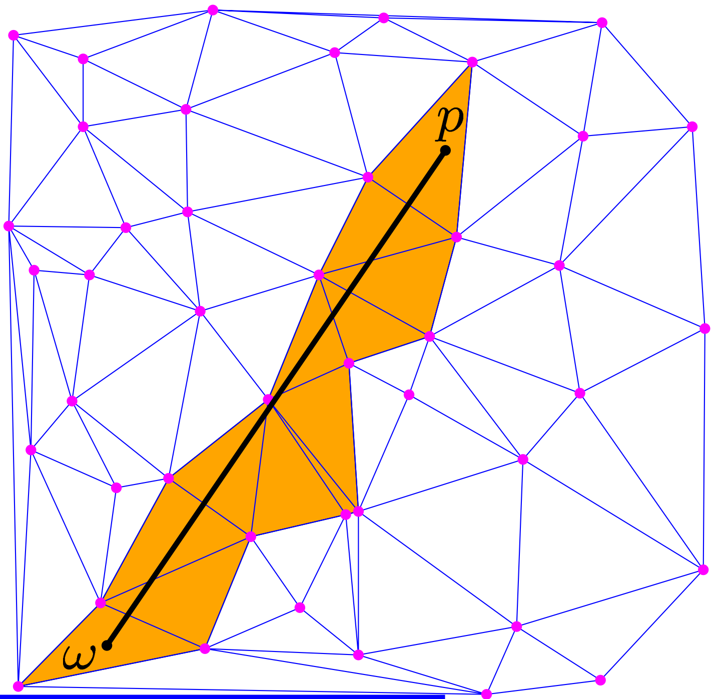

auxiliary
location
structure



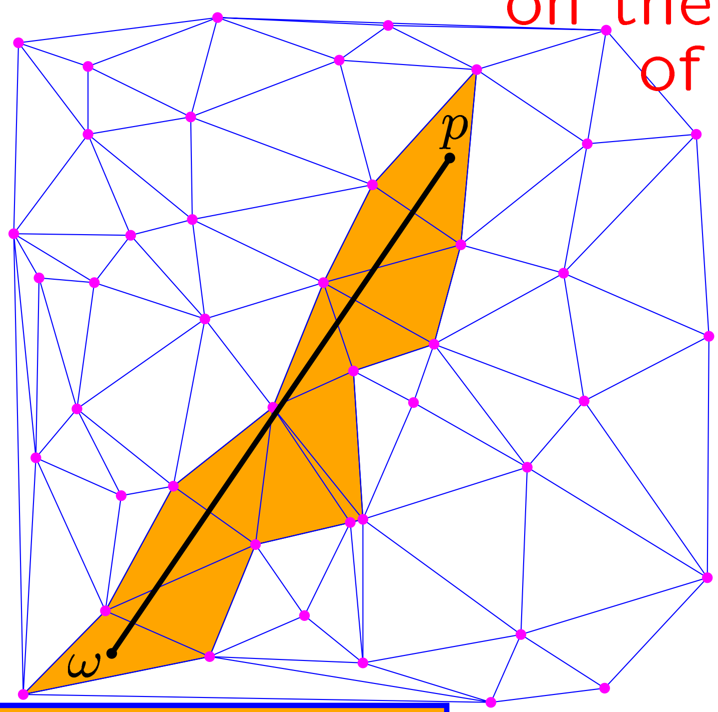
$O(\log n)$

levels

$O(n \log n)$



$O(\sqrt{n})$ visited triangles



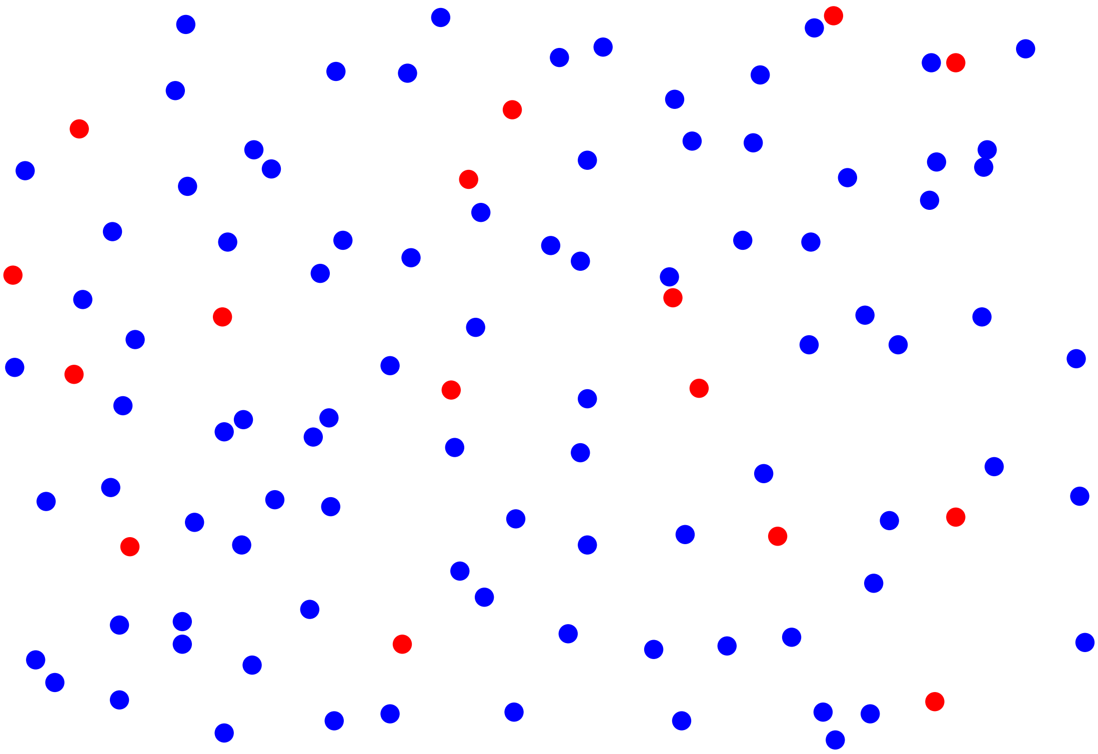
$O(\sqrt{n})$ visited triangles

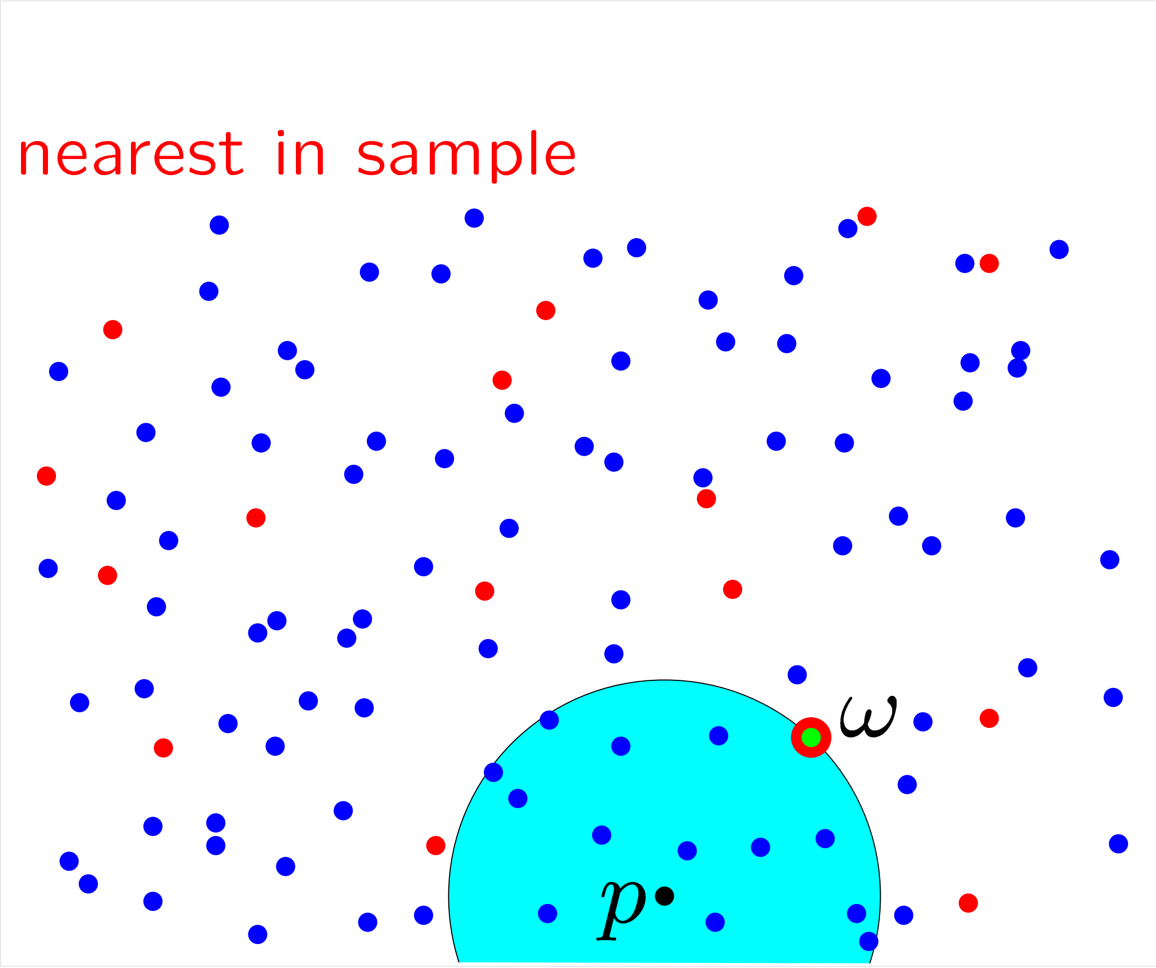
Find a good starting point ω

very simple algorithm

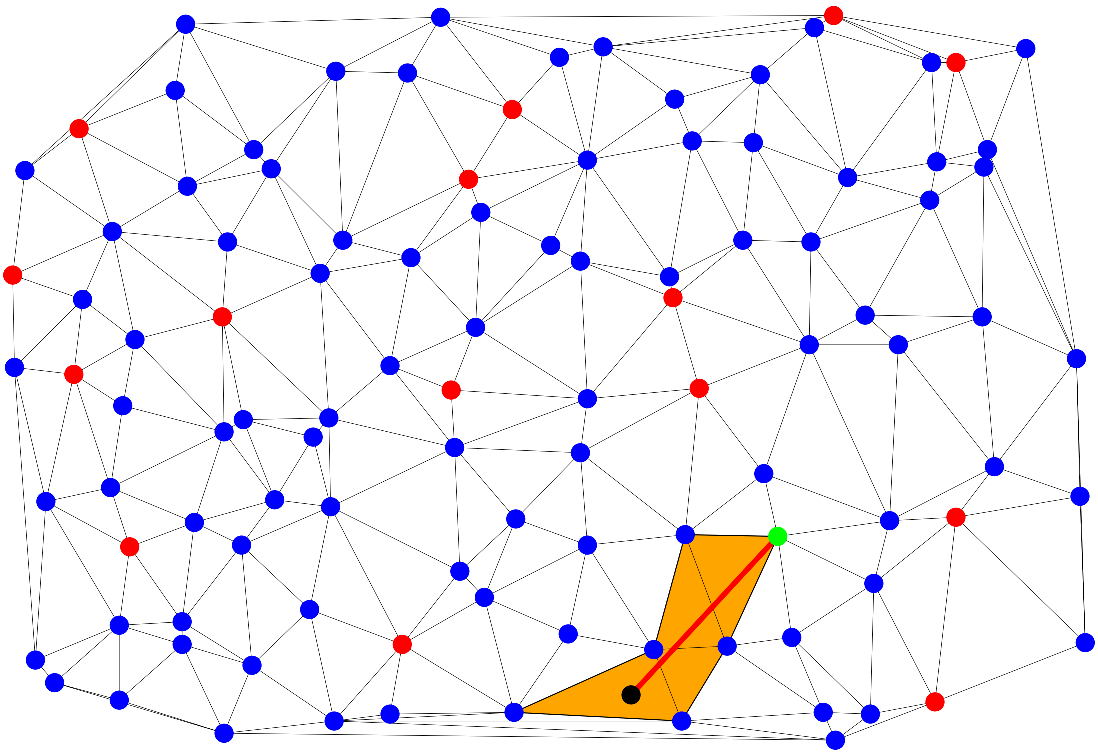
[Mücke Saias Zhu]

random sample of size k




nearest in sample
ω
p

Walk from ω



Analysis

Nearest in sample

O(k)

March

$$O(\sqrt{\frac{n}{k}})$$

$$\boxed{O(k+\sqrt{\frac{n}{k}})}$$

Analysis

$$O(k+\sqrt{\tfrac{n}{k}})$$

$$k = \sqrt[3]{n}$$

$$\boxed{O(\sqrt[3]{n})}$$

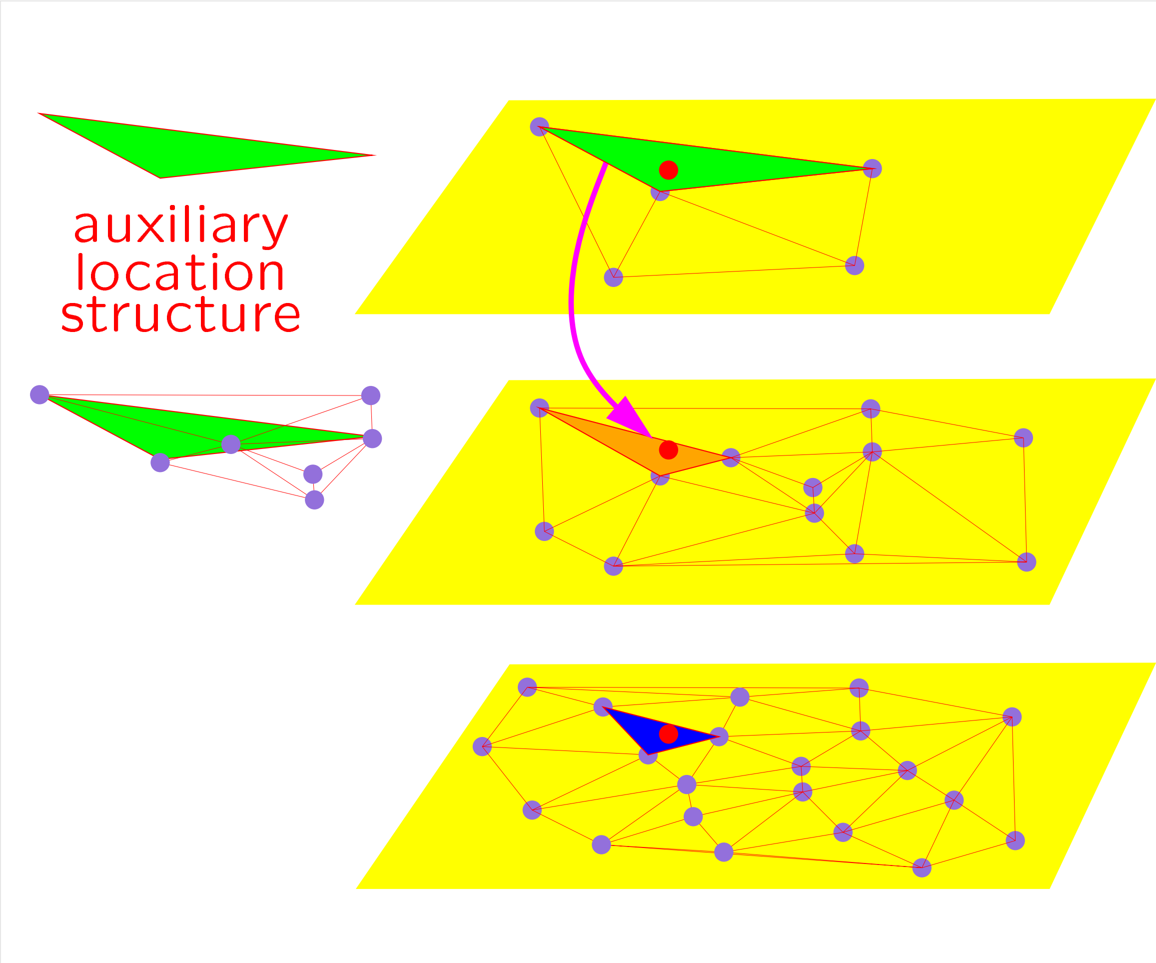
Delaunay $\boxed{O(n\sqrt[3]{n})}$

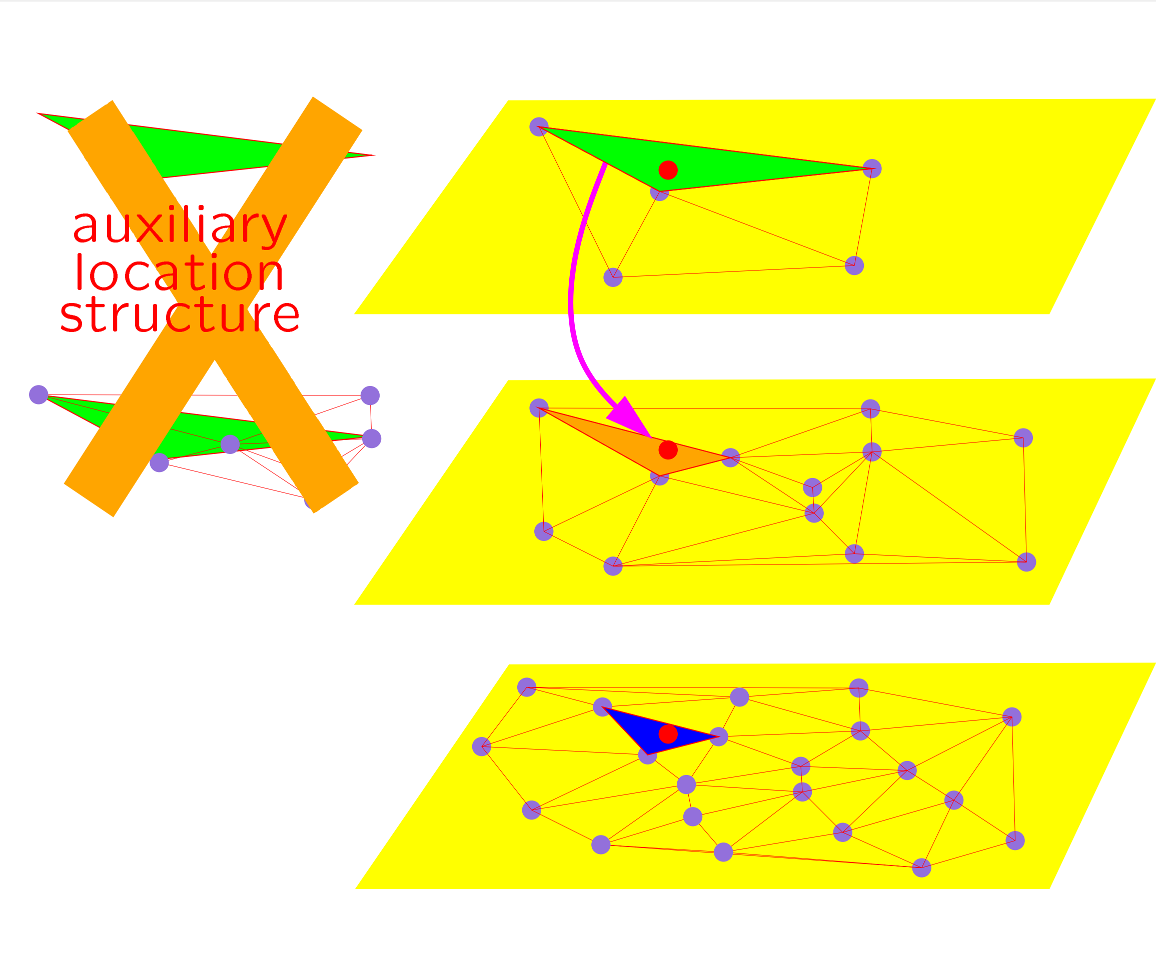
For random points only

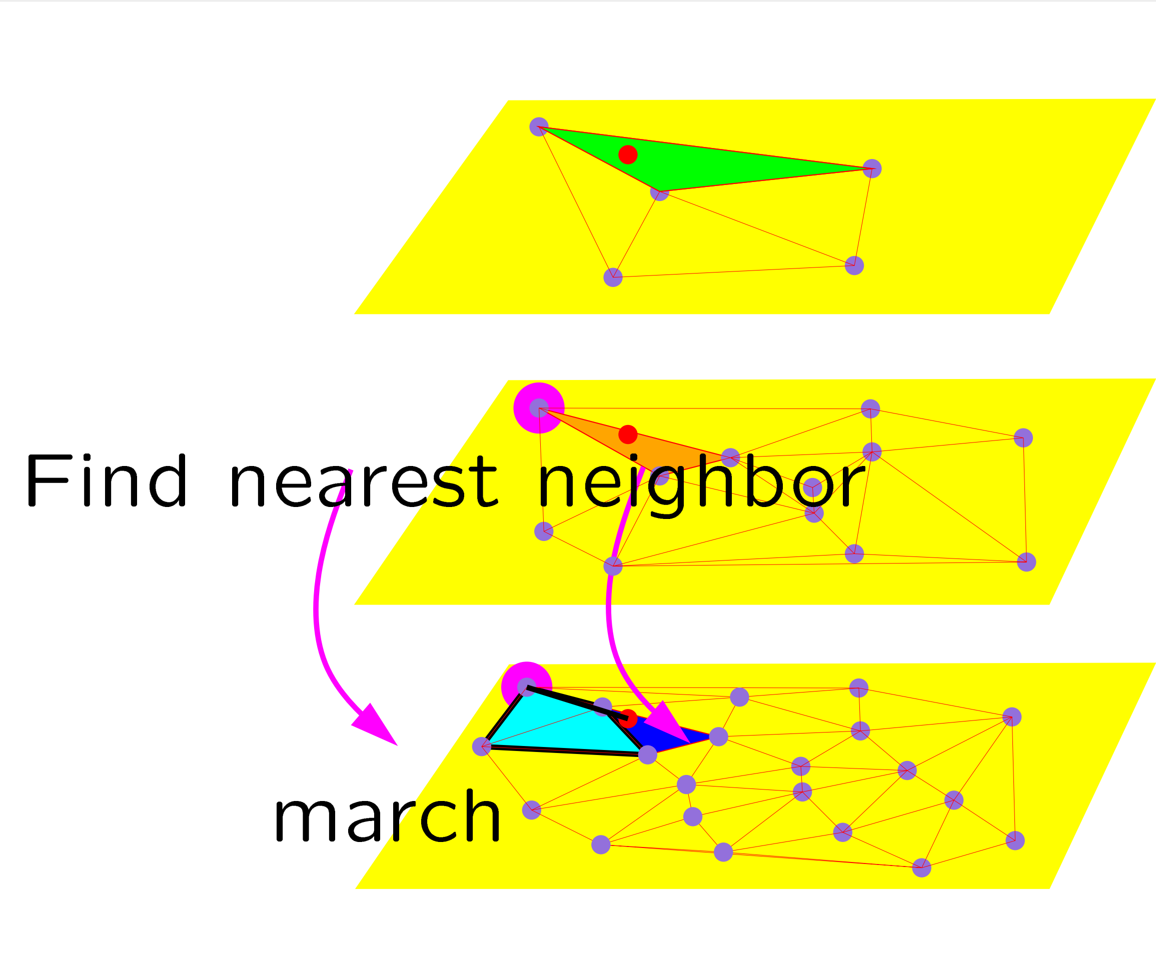
The Delaunay Hierarchy

[D]

very simple idea


auxiliary
location
structure


auxiliary
location
structure


Find nearest neighbor
march

very simple idea

difficult analysis

$O(n \log n)$ worst case

Good practical behavior
Random points

Small penalty memory

q query

v_i nearest vertex at level i

At level i

walk from v_{i+1} to v_i

- Turn around v_{i+1}
- Walk along $v_{i+1}q$
- Look at vertices "close" to q

Analysis

$$E(d_i^\circ(v_{i+1}) \leq 36$$

Random v at level $i+1$

average d_i° is 6

Nearest neighbor graph as degree ≤ 6

v can be v_i for at most 6 q

Analysis

Phase 2

Expected number of edges

$$O(\alpha)$$

Phase 3

Expected number of triangles

$$O(\alpha)$$

Analysis

$O(\alpha)$ per level

$O(n \log_\alpha n)$ time

$\left(1 + \frac{1}{\alpha-1}\right) n$ memory

Tuning α

$3 + \sqrt{\frac{\alpha}{\pi}}$ o. tests qv_{i+1} *4 + 5 f.p.o.

$\sqrt{\frac{\alpha}{\pi}}$ o. tests *7 f.p.o.

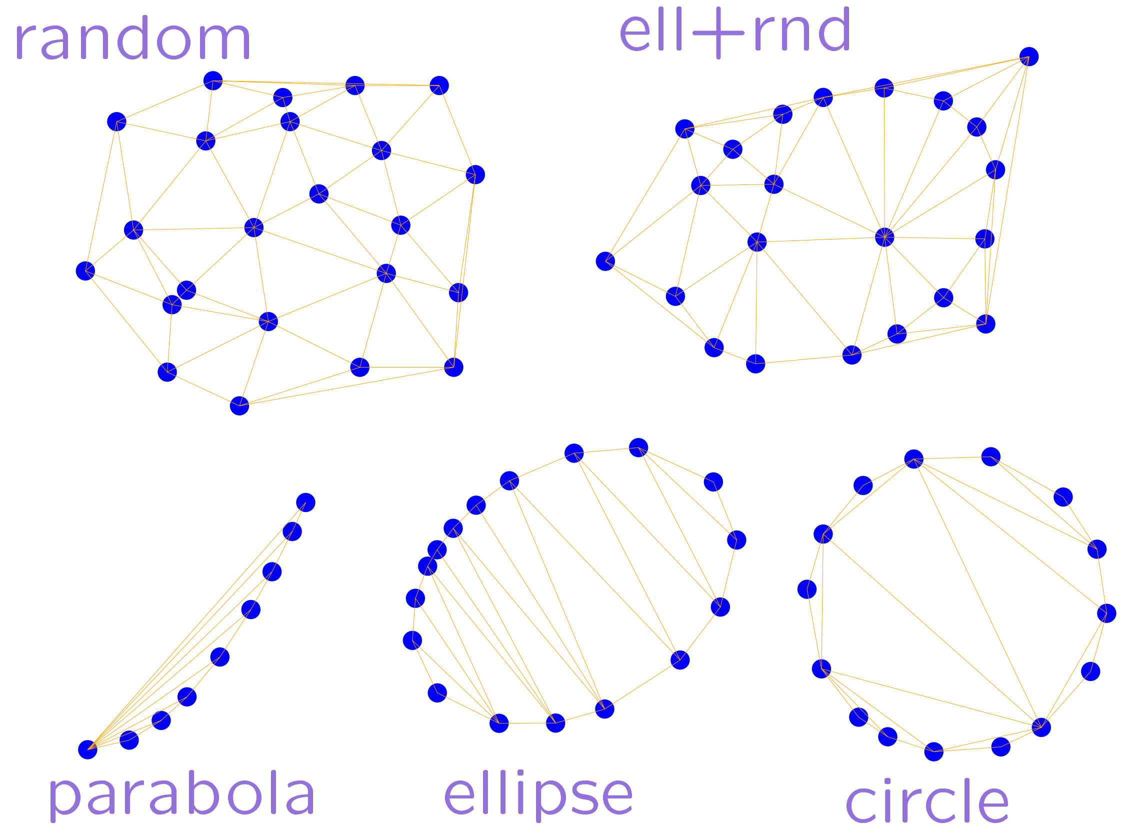
3 distance computations *5 f.p.o.

$$\boxed{32 + 6.2\sqrt{\alpha}}$$

$$\boxed{\frac{32+6.2\sqrt{\alpha}}{\log_2 \alpha} \log_2 n}$$

$$\boxed{\alpha \simeq 40}$$

Tests



time in seconds Sun Ultra 200

Delaunay only (`clock`)

500000 points

	walk	MSZ	$\star$	$\star$+MSZ
random	460	72	36	31
ell + rnd	930	760	57	49
ellipse	long	270	54	55
parabola	long	74	69	45
circle	39	44	36	36

Randomized incremental
versus
on line

[D & Guigue]

The shuffling buffer

$O(n^2 \log n)$

worst case

$O(n \log n)$

random order

$O(\frac{\log m}{m} n^2 \log n)$

m shuffled order