Modèles d'environnements & planification de trajectoire

Francis Colas

Olivier Devillers

Xavier Goaoc

Modèles d'environnements & planification de trajectoire

Francis Colas

Olivier Devillers

Xavier Goaoc

Modèles d'environnements & planification de trajectoire

Francis Colas

Olivier Devillers

Xavier Goaoc

- 24 heures de cours (12x2 heures)

  8x2 heures de GÉOMÉTRIE ALGORITHMIQUE

  4x2 heures de ROBOTIQUE

- Contrôle continu (40%) + examen (30 %) + exposé sur article (30 %)

  Pas de rattrapage pour le contrôle continu...

`https://members.loria.fr/Olivier.Devillers/master/`

1. (XG) Enveloppe convexe: définition et algorithmes.
2. (XG) Triangulation de Delaunay, définitions et premières propriétés.
3. (XG) Triangulation de Delaunay, algorithmes
4. (OD) Randomisation.
5. (OD) Robustesse aux erreurs numériques
6. (FC) Cartes en robotique
7. (OD) Reconstruction
8. (XG) Arrangements de courbes et de surfaces
9. (XG) Subdivision spatiale et planification de trajectoires
10. (FC) Espace de configuration
11. (FC) Planification déterministe
12. (FC) Planification stochastique

C'est quoi la géométrie algorithmique ?

Computational geometry

?

Computational geometry

Design geometric algorithms

Computational geometry

Design geometric algorithms

Study complexity

Computational geometry

Design geometric algorithms

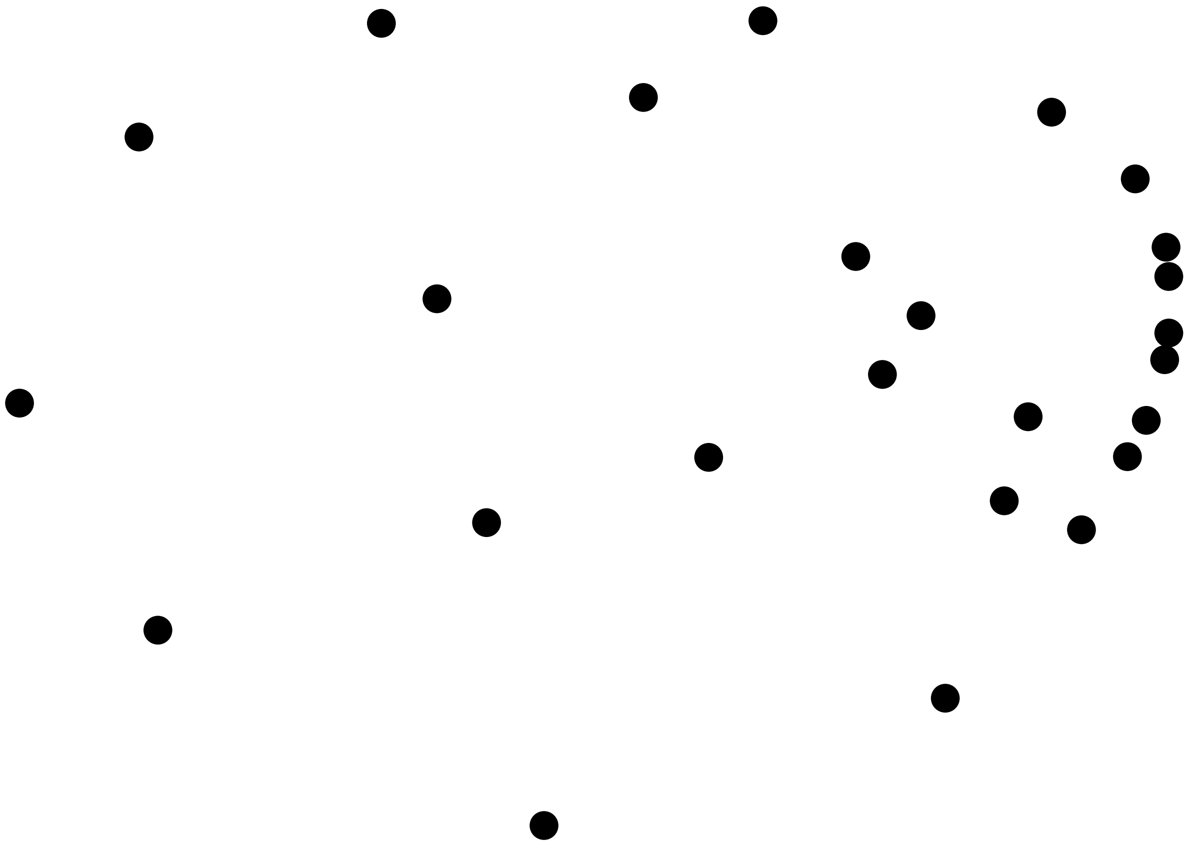
Study complexity

- Model of computation
- Worst-case or random analysis
- Lower bound
- Asymptotic analysis

Computational geometry problems

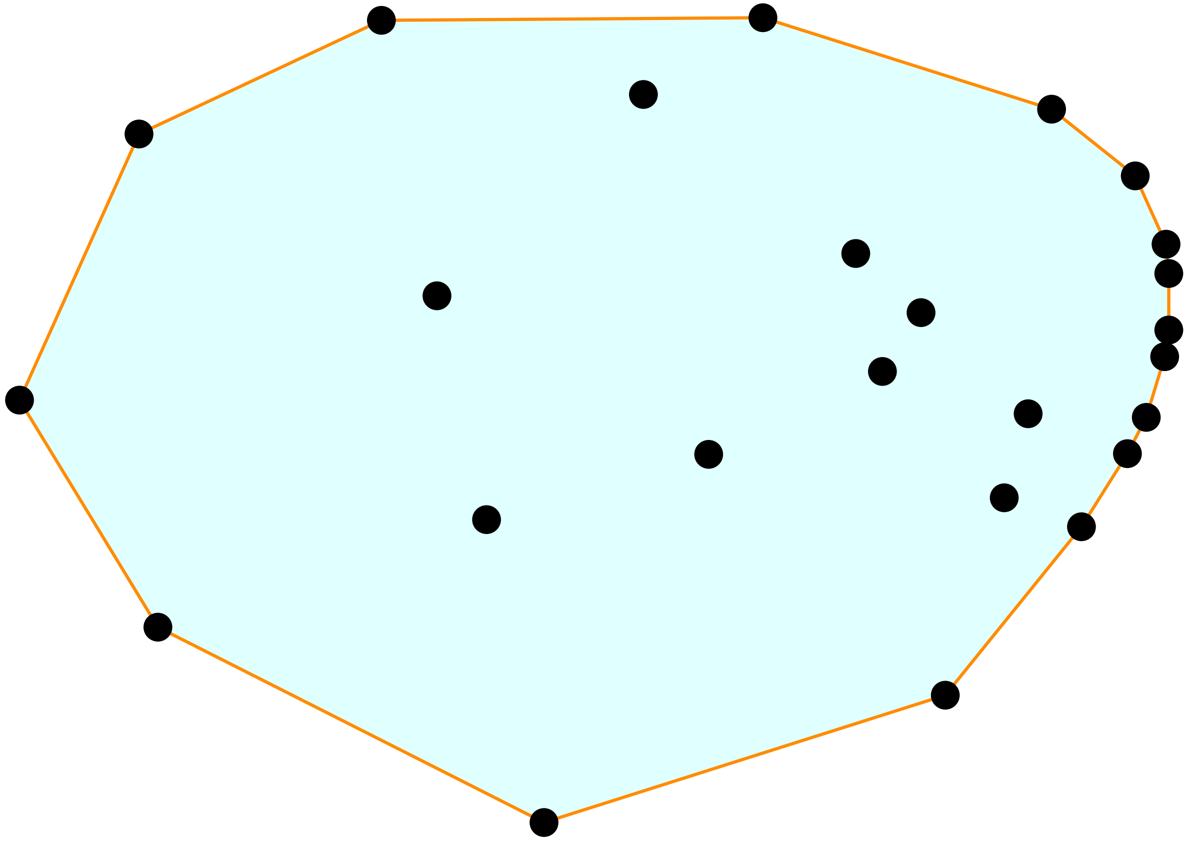
Computational geometry problems

Convex hull



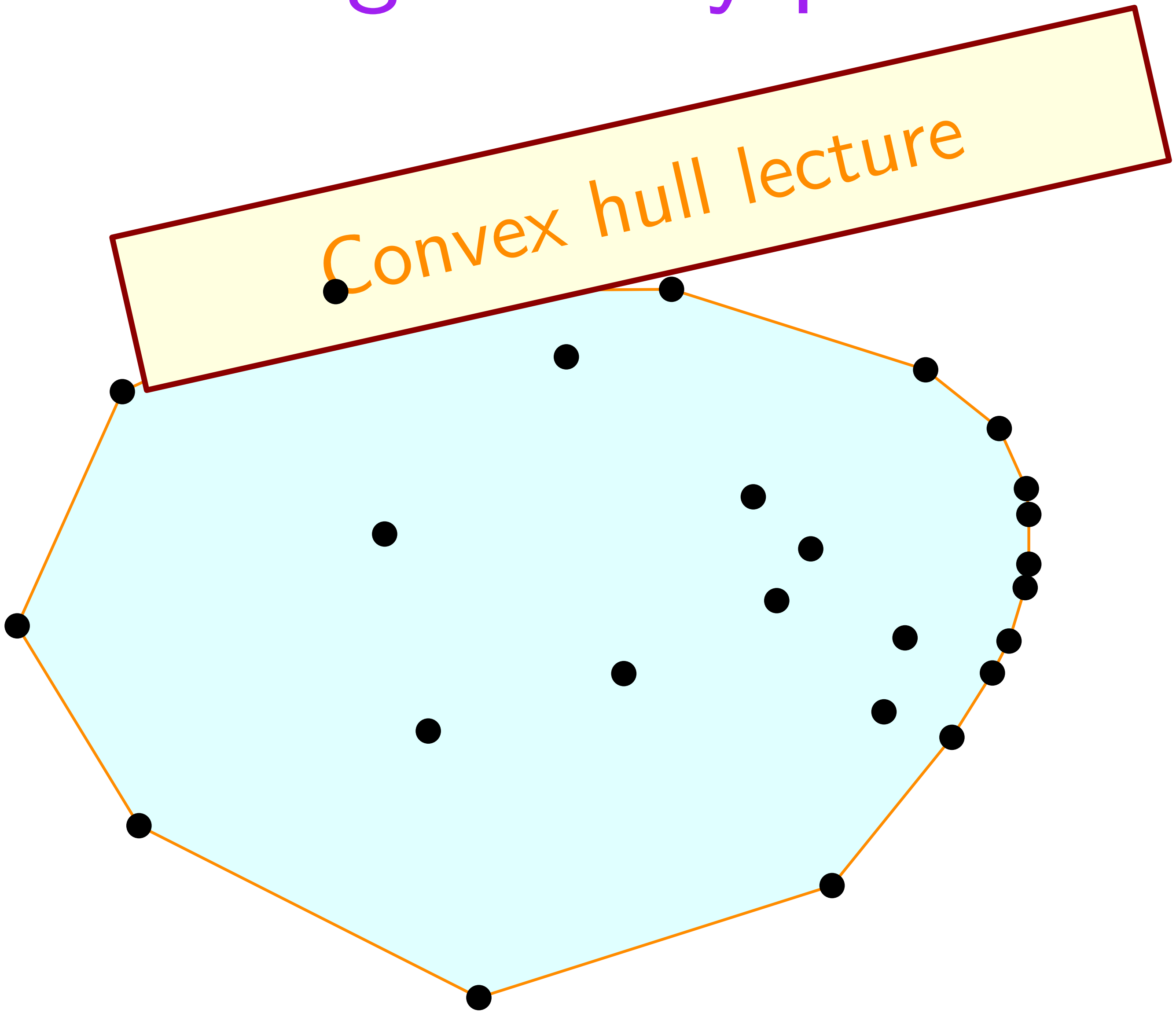
Computational geometry problems

Convex hull



Computational geometry problems

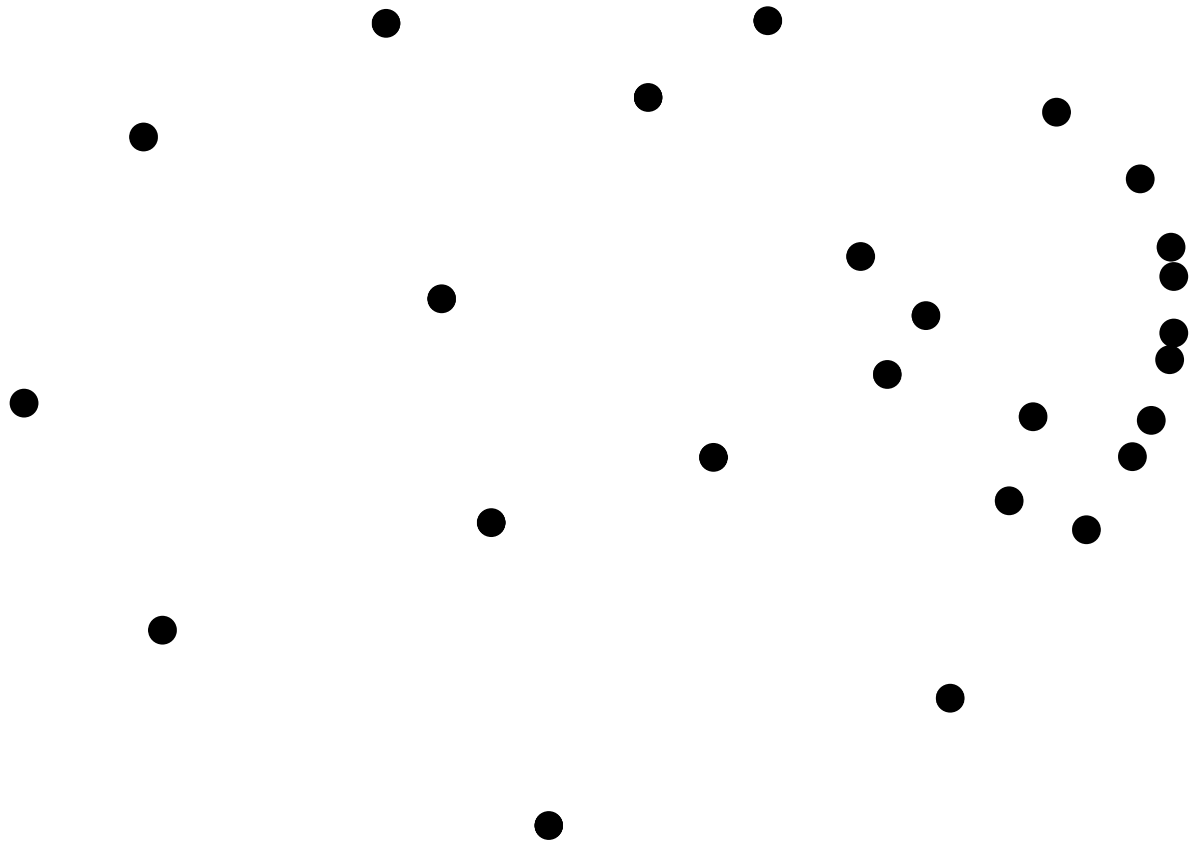
Convex hull



Computational geometry problems

Convex hull

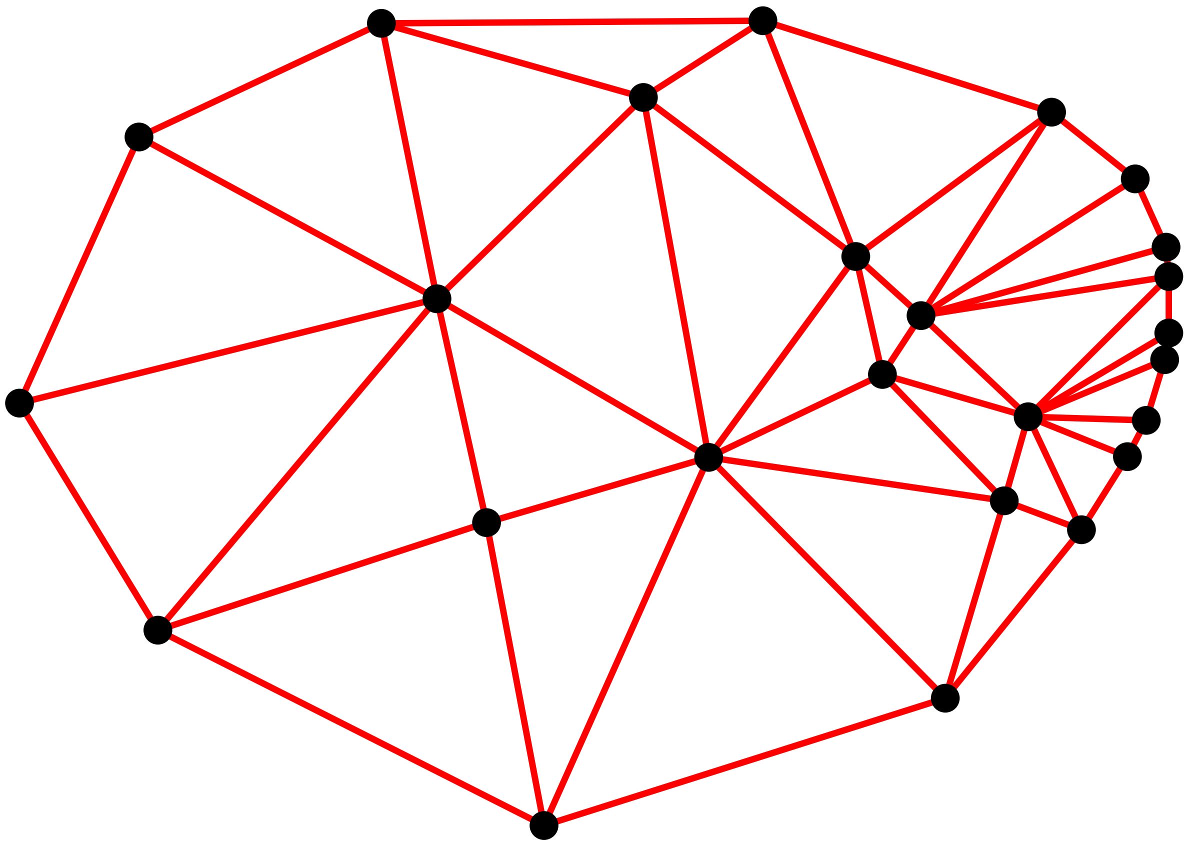
Delaunay triangulation / Voronoi diagrams



Computational geometry problems

Convex hull

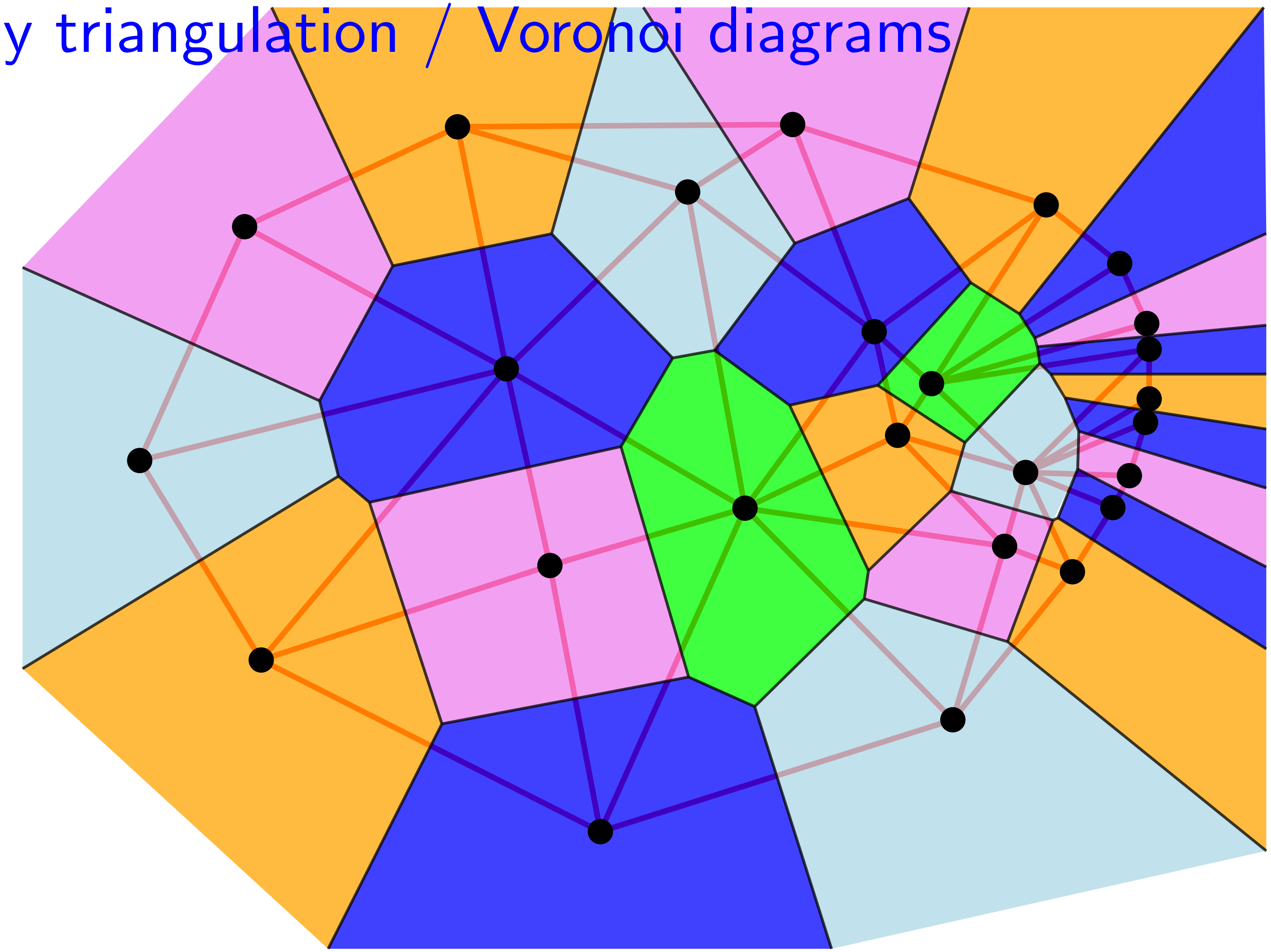
Delaunay triangulation / Voronoi diagrams



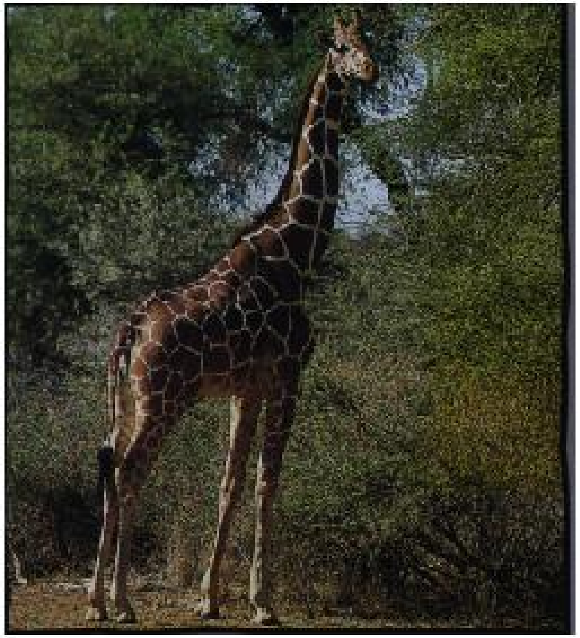
Computational geometry problems

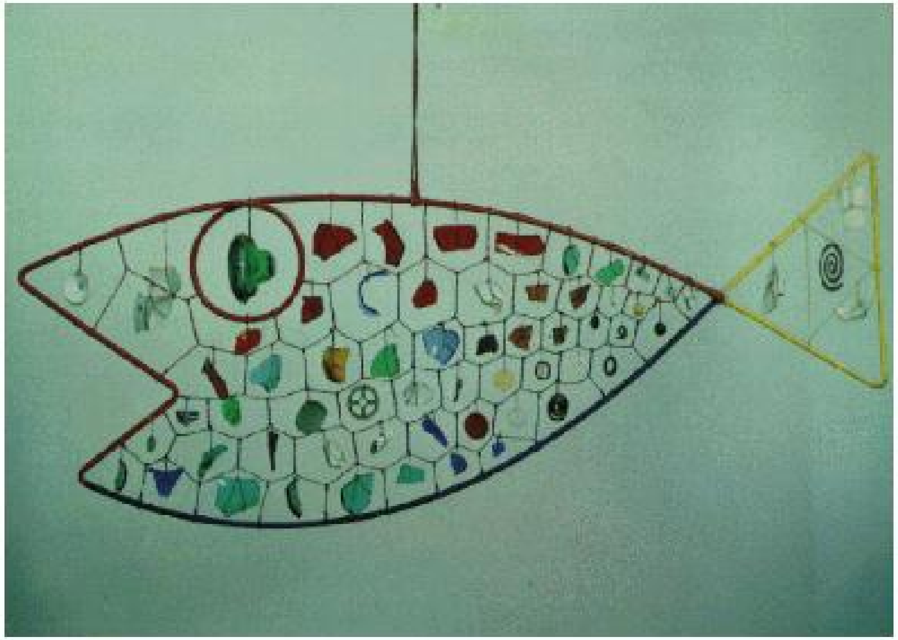
Convex hull

Delaunay triangulation / Voronoi diagrams



Computational geometry problems





Computational geometry problems

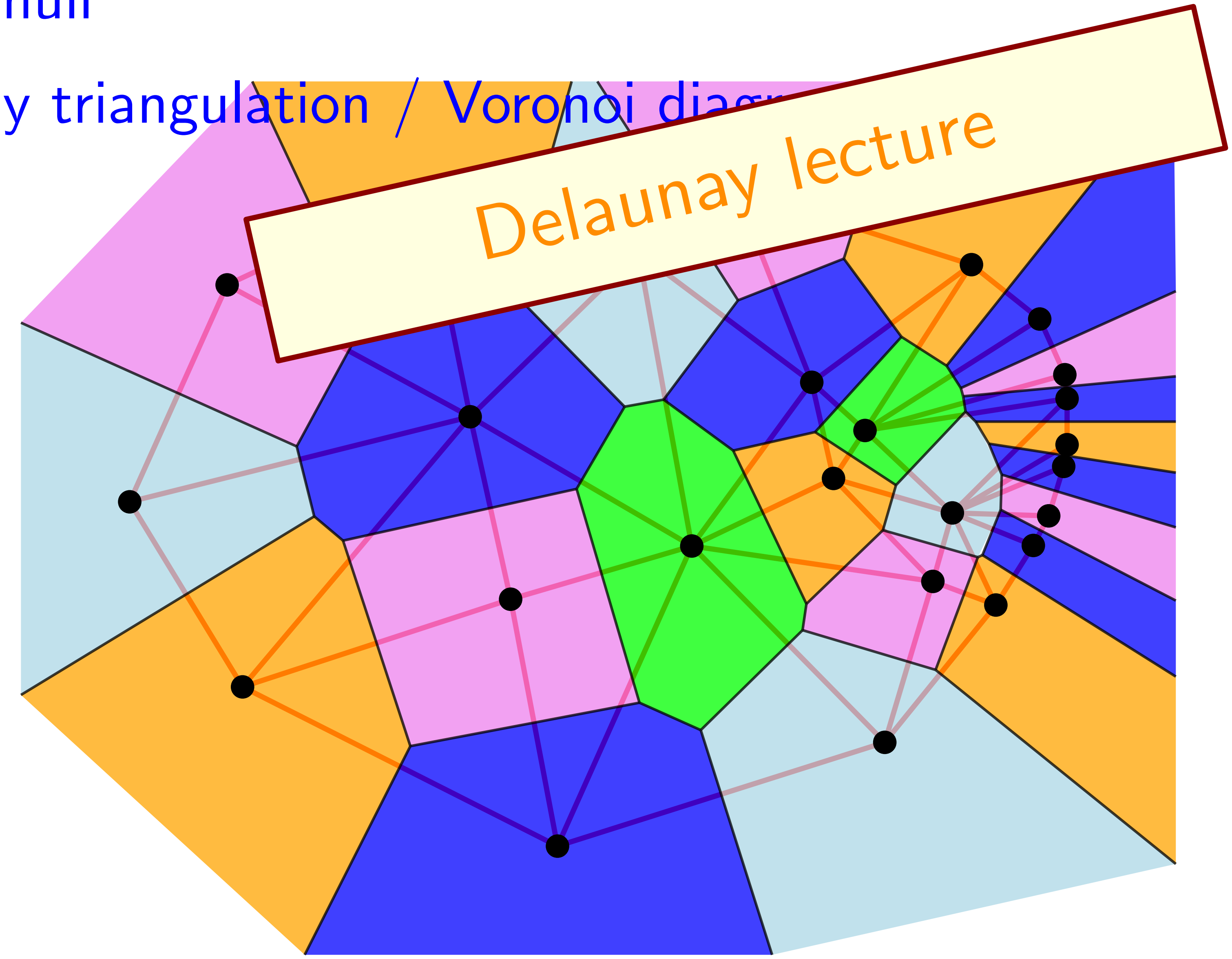
Computational geometry problems

Computational geometry problems

Computational geometry problems

Convex hull

Delaunay triangulation / Voronoi diagr

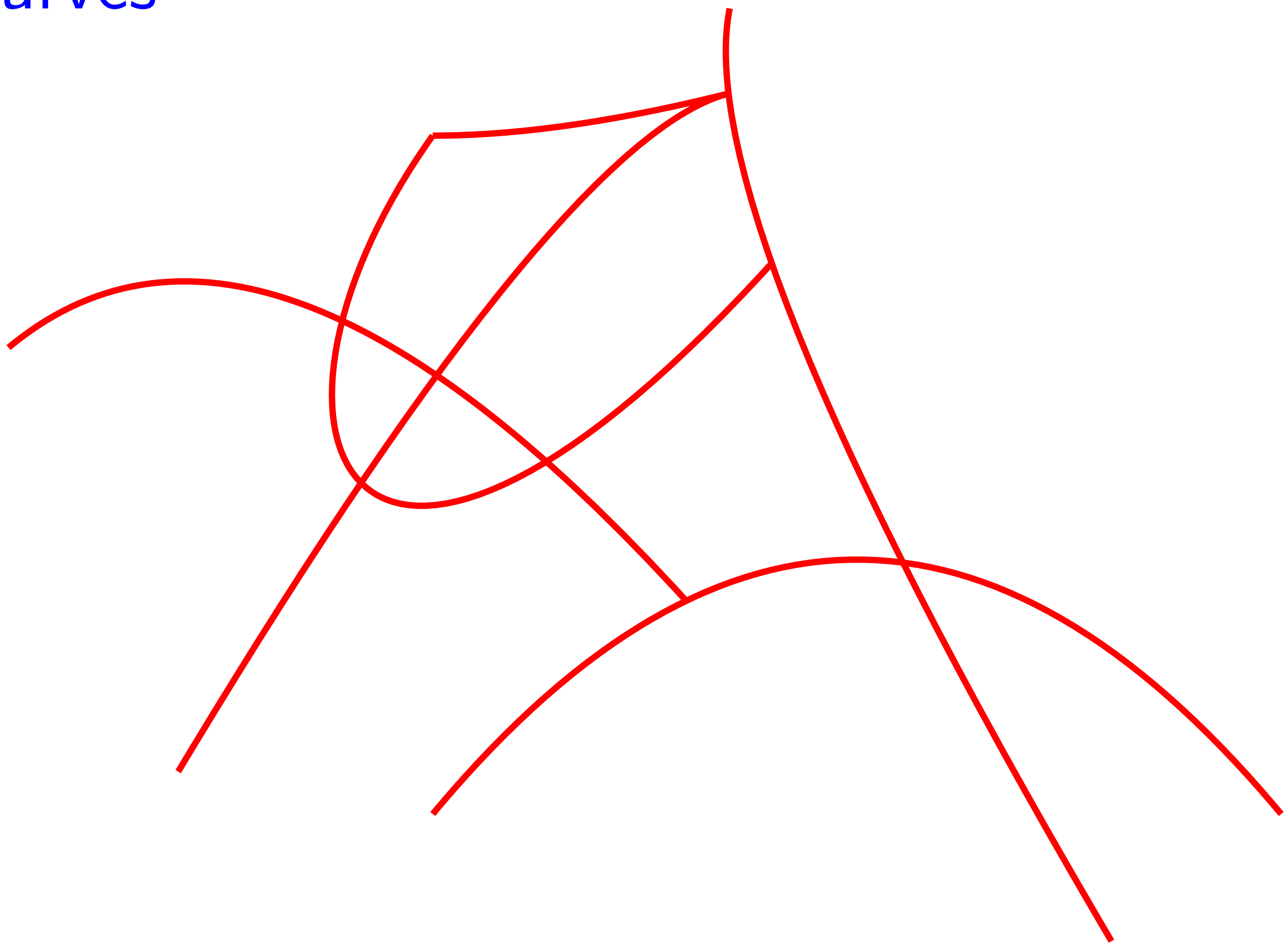


Computational geometry problems

Convex hull

Delaunay triangulation / Voronoi diagrams

Arrangement of curves

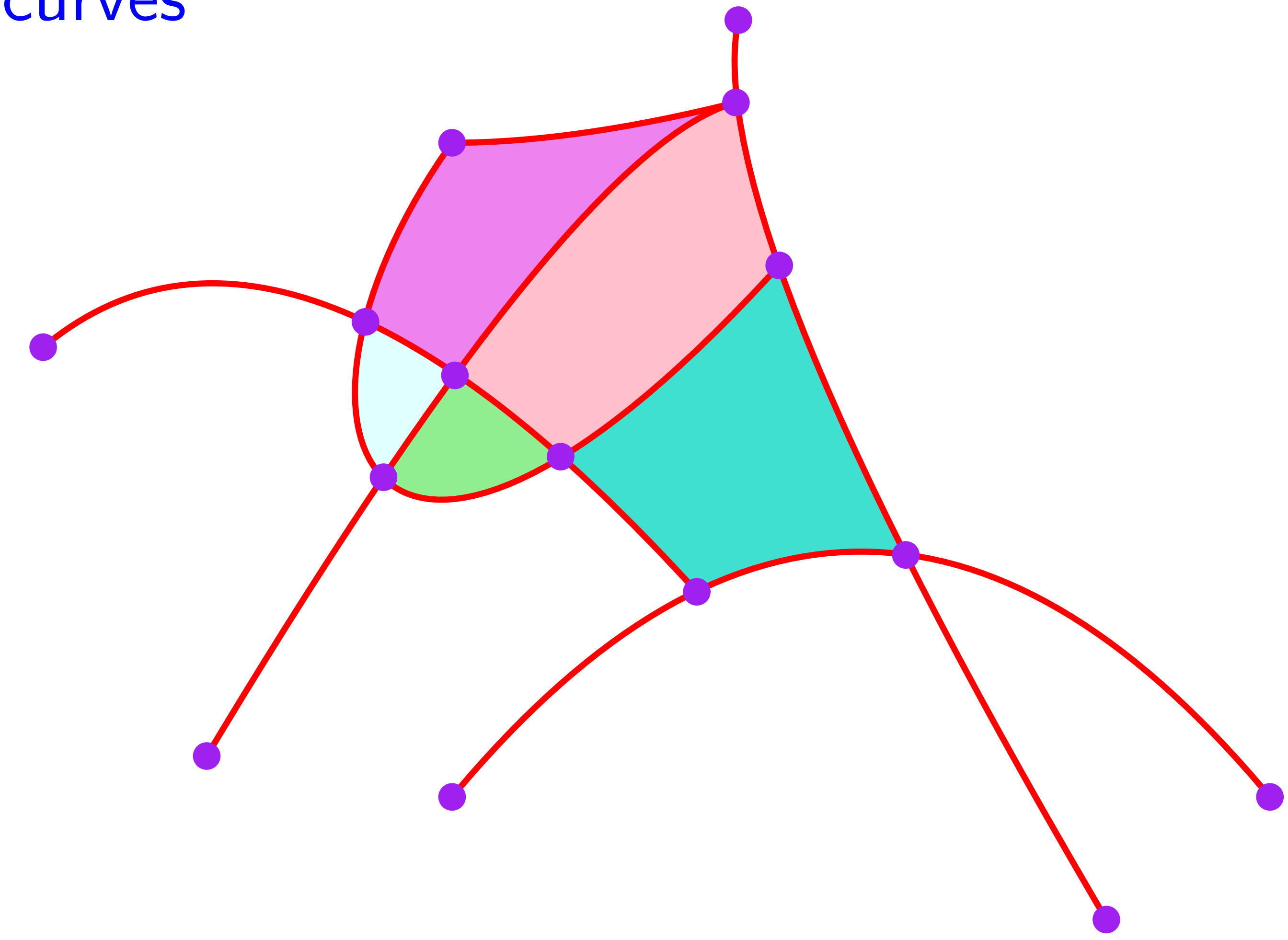


Computational geometry problems

Convex hull

Delaunay triangulation / Voronoi diagrams

Arrangement of curves



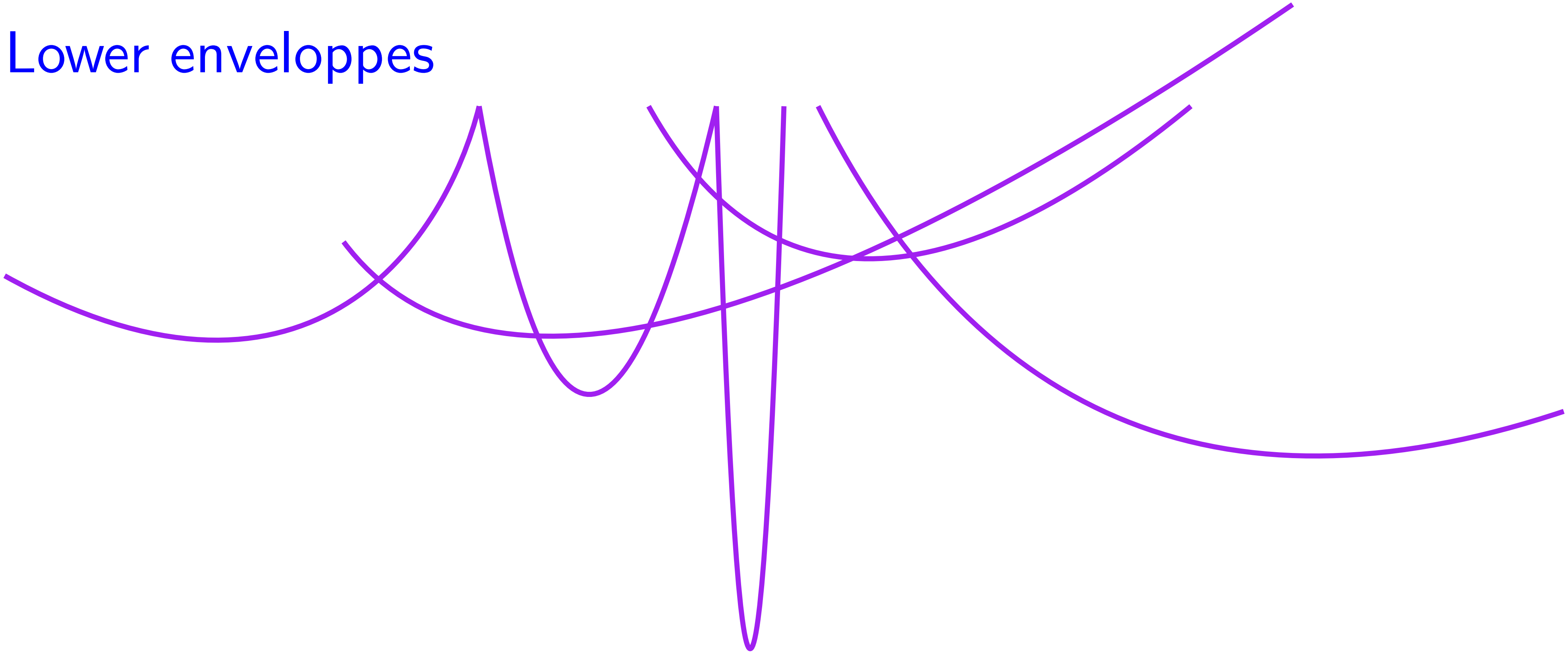
Computational geometry problems

Convex hull

Delaunay triangulation / Voronoi diagrams

Arrangement of curves

Lower enveloppes



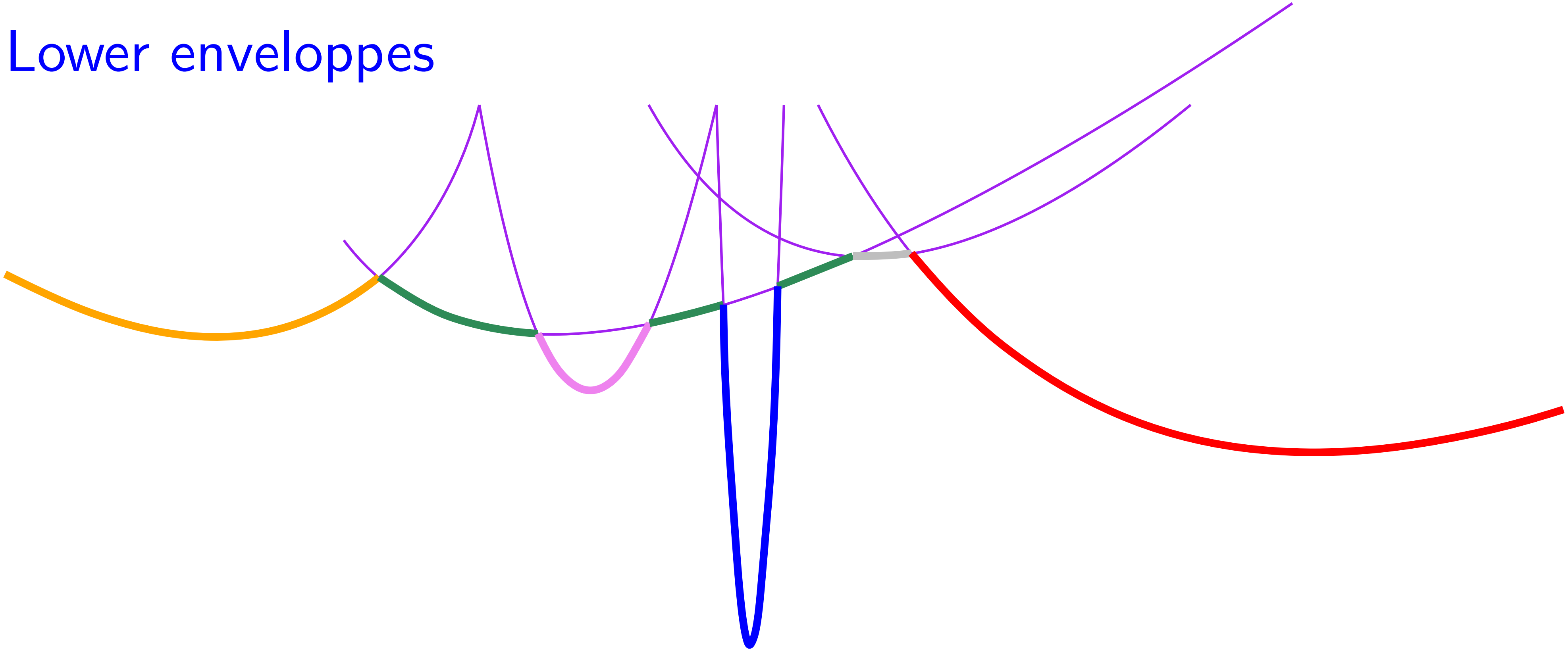
Computational geometry problems

Convex hull

Delaunay triangulation / Voronoi diagrams

Arrangement of curves

Lower enveloppes



Computational geometry problems

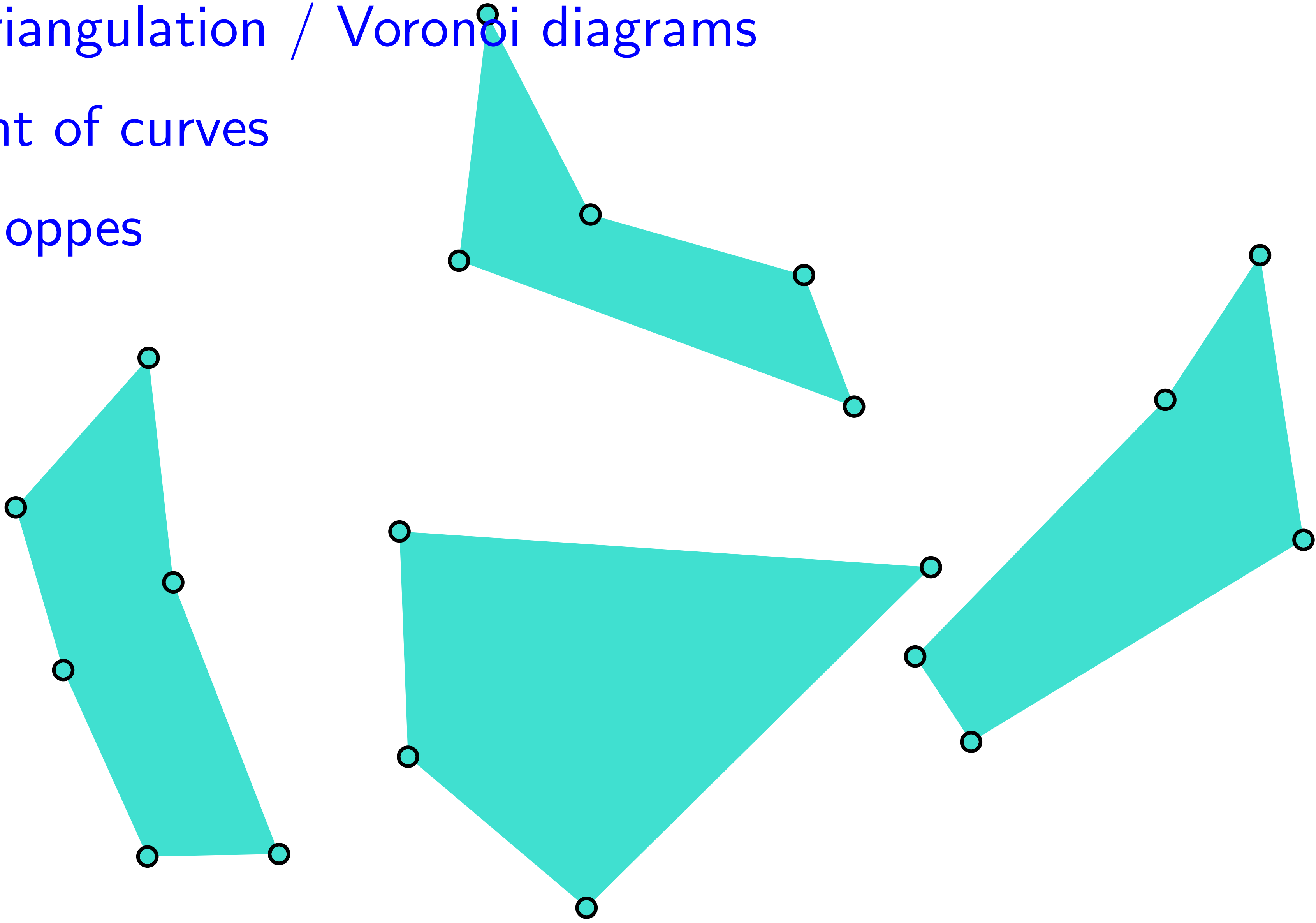
Convex hull

Delaunay triangulation / Voronoi diagrams

Arrangement of curves

Lower enveloppes

Visibility



Computational geometry problems

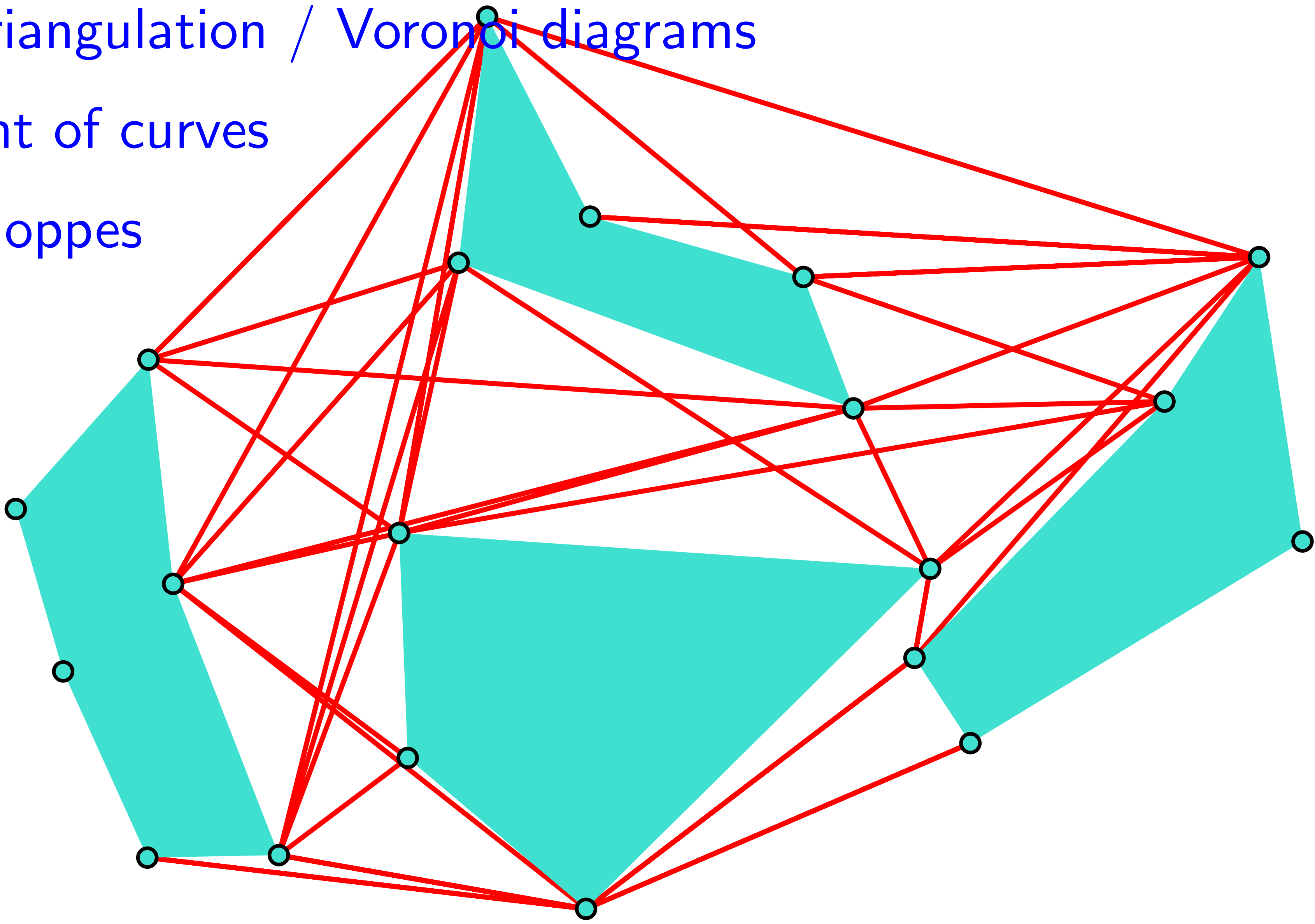
Convex hull

Delaunay triangulation / Voronoi diagrams

Arrangement of curves

Lower enveloppes

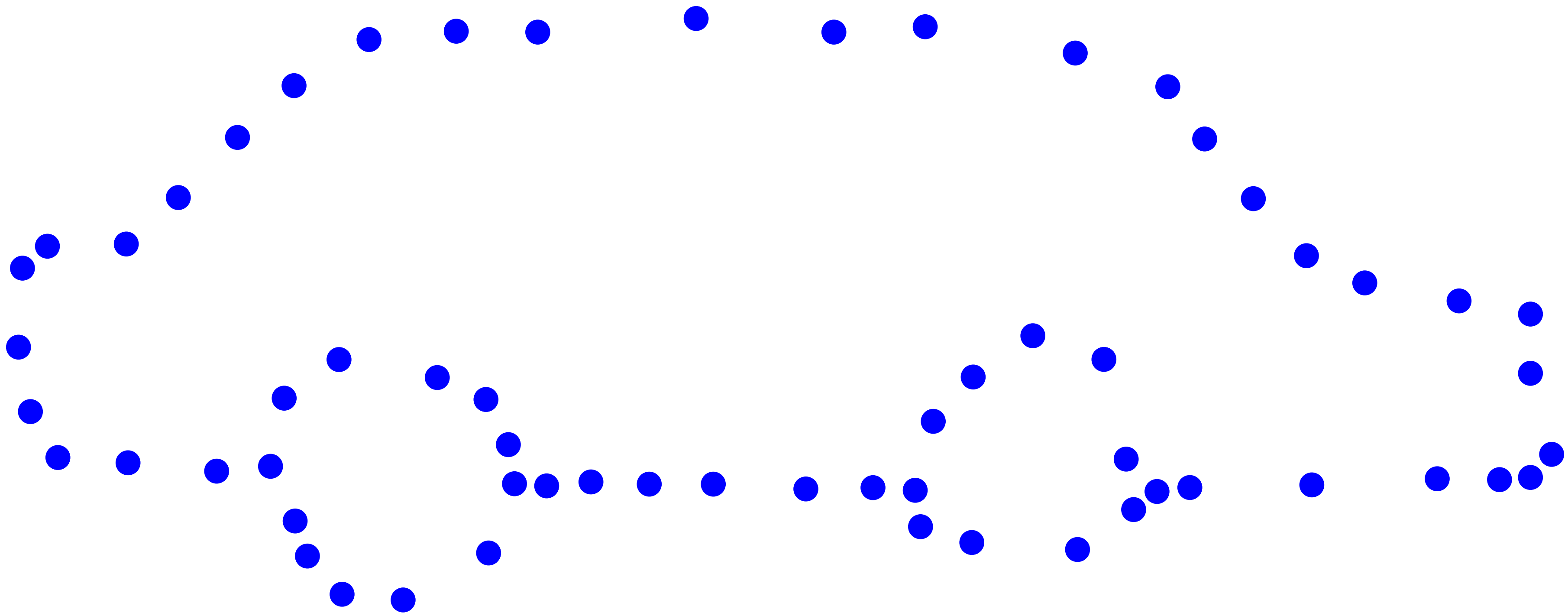
Visibility



Computational geometry usage

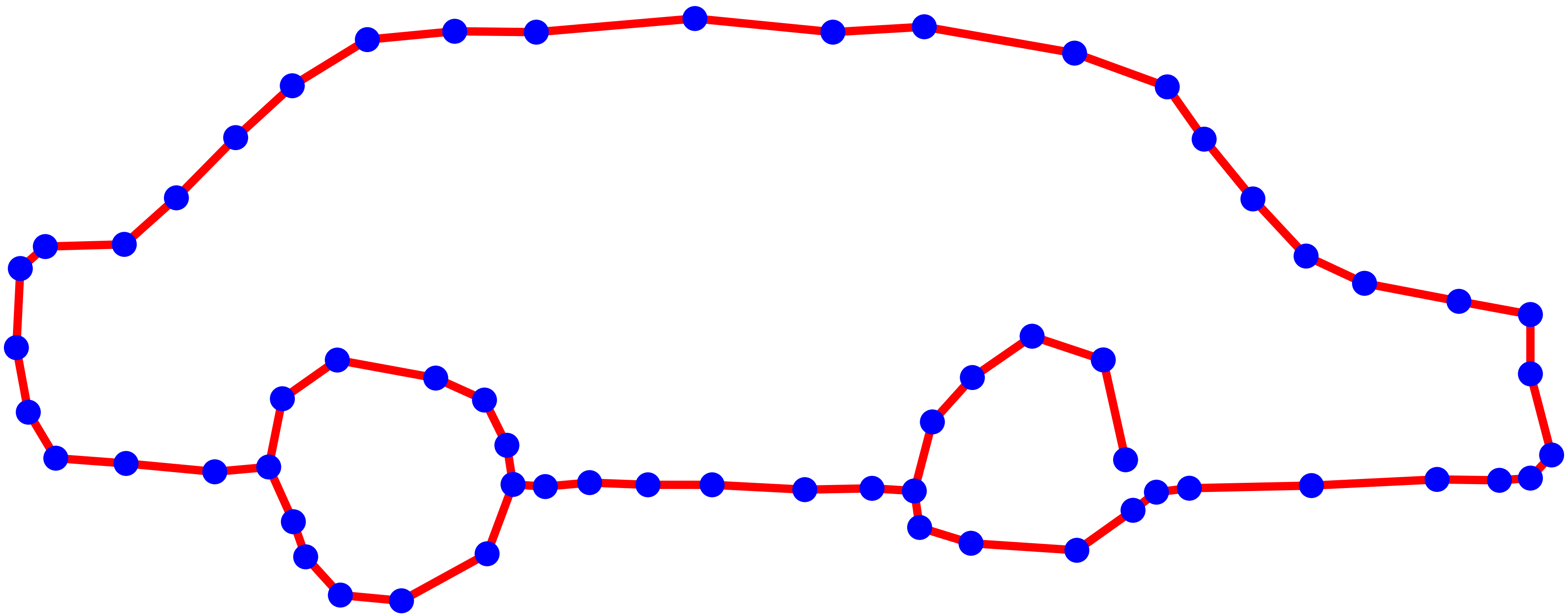
Computational geometry usage

Points to shape



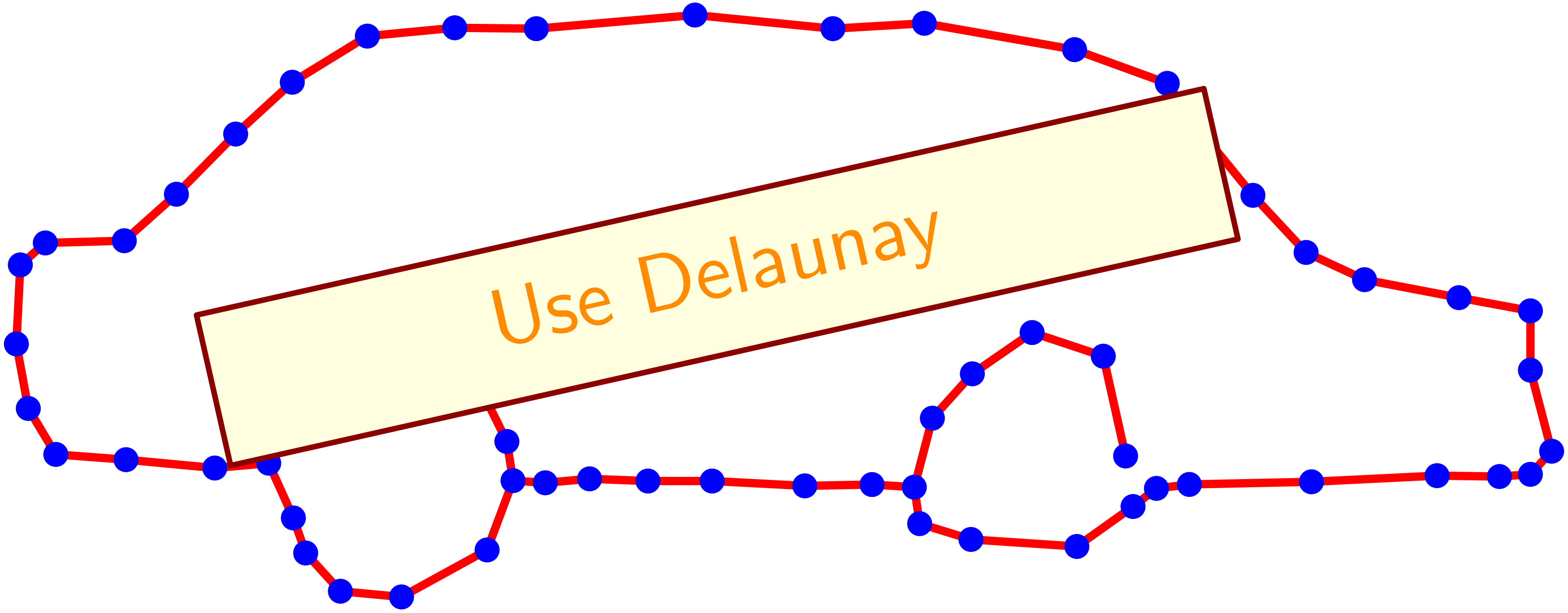
Computational geometry usage

Points to shape



Computational geometry usage

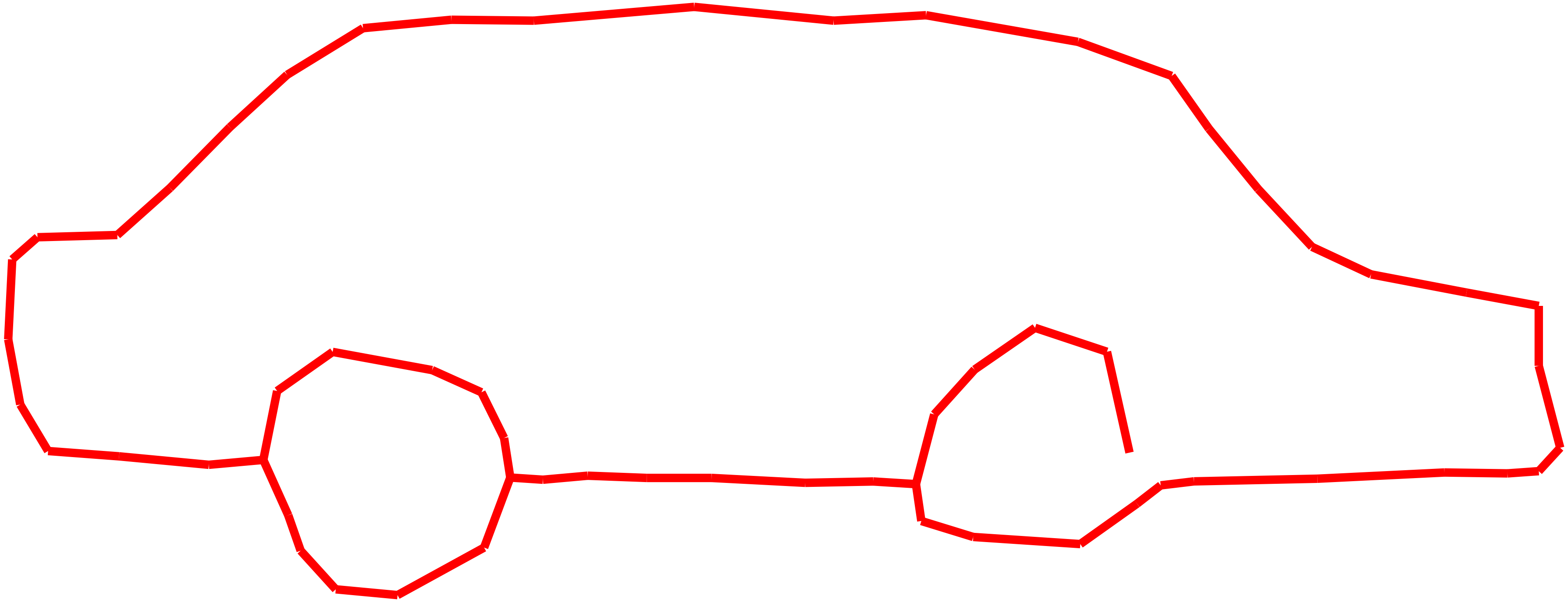
Points to shape



Computational geometry usage

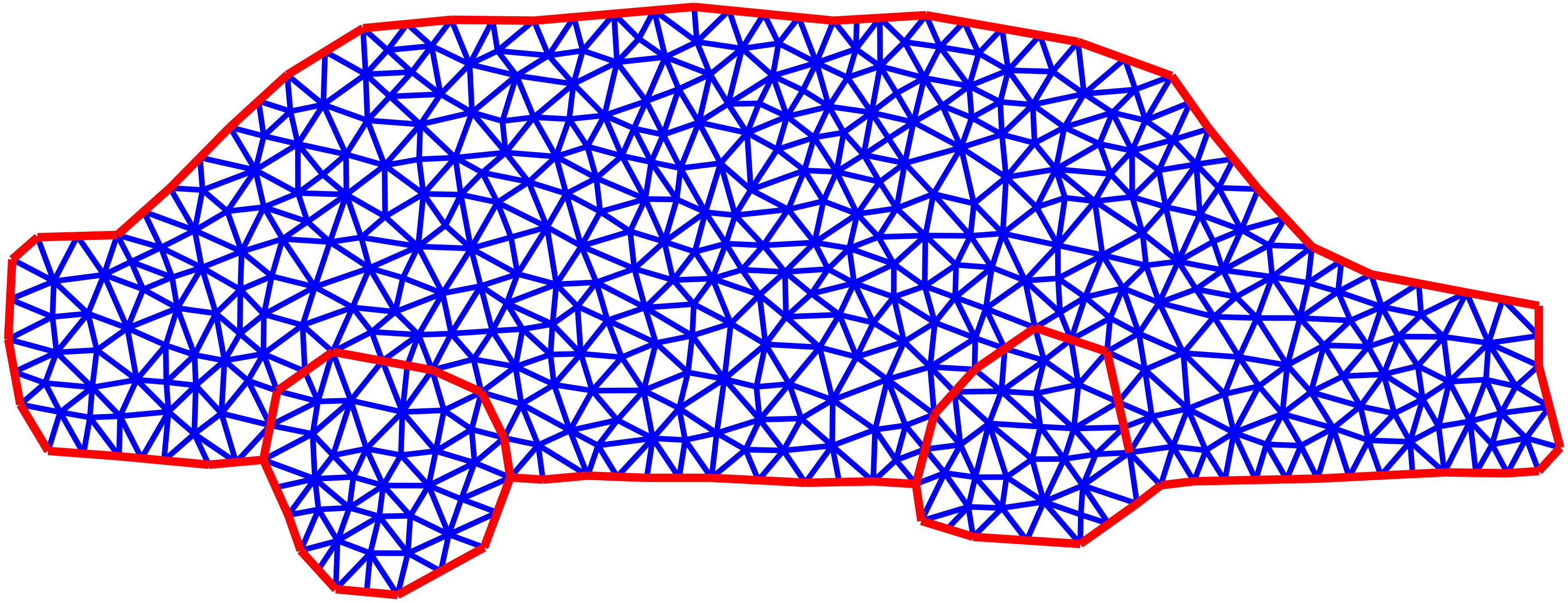
Computational geometry usage

Shape to mesh



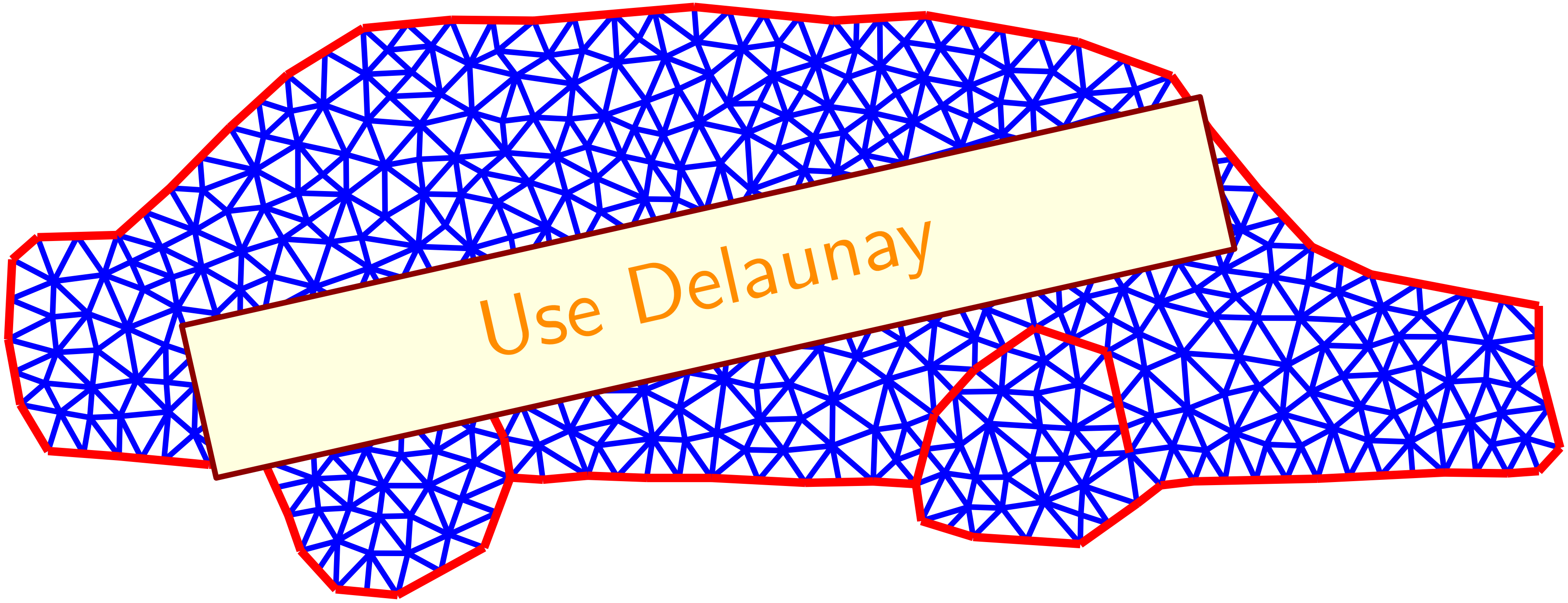
Computational geometry usage

Shape to mesh



Computational geometry usage

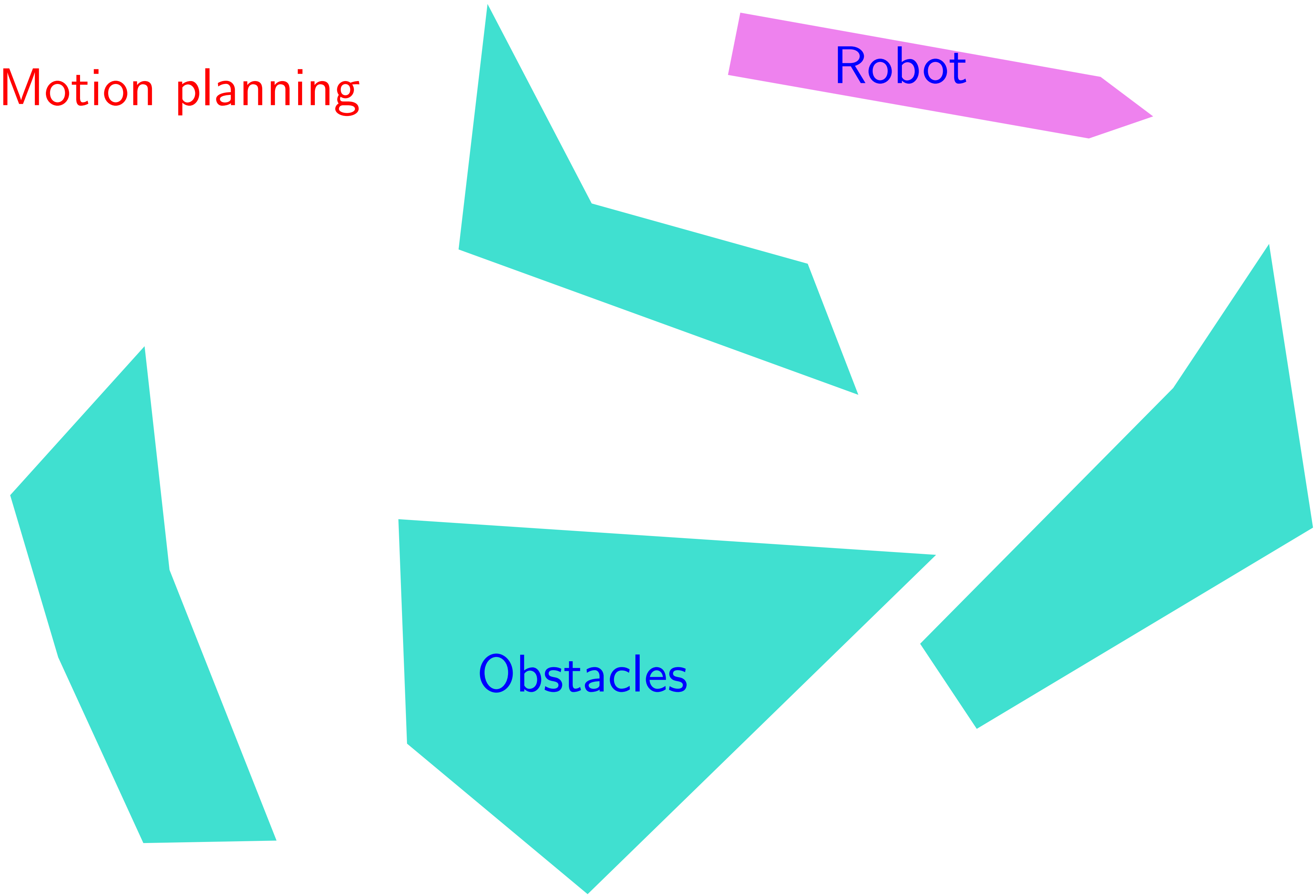
Shape to mesh



Computational geometry usage

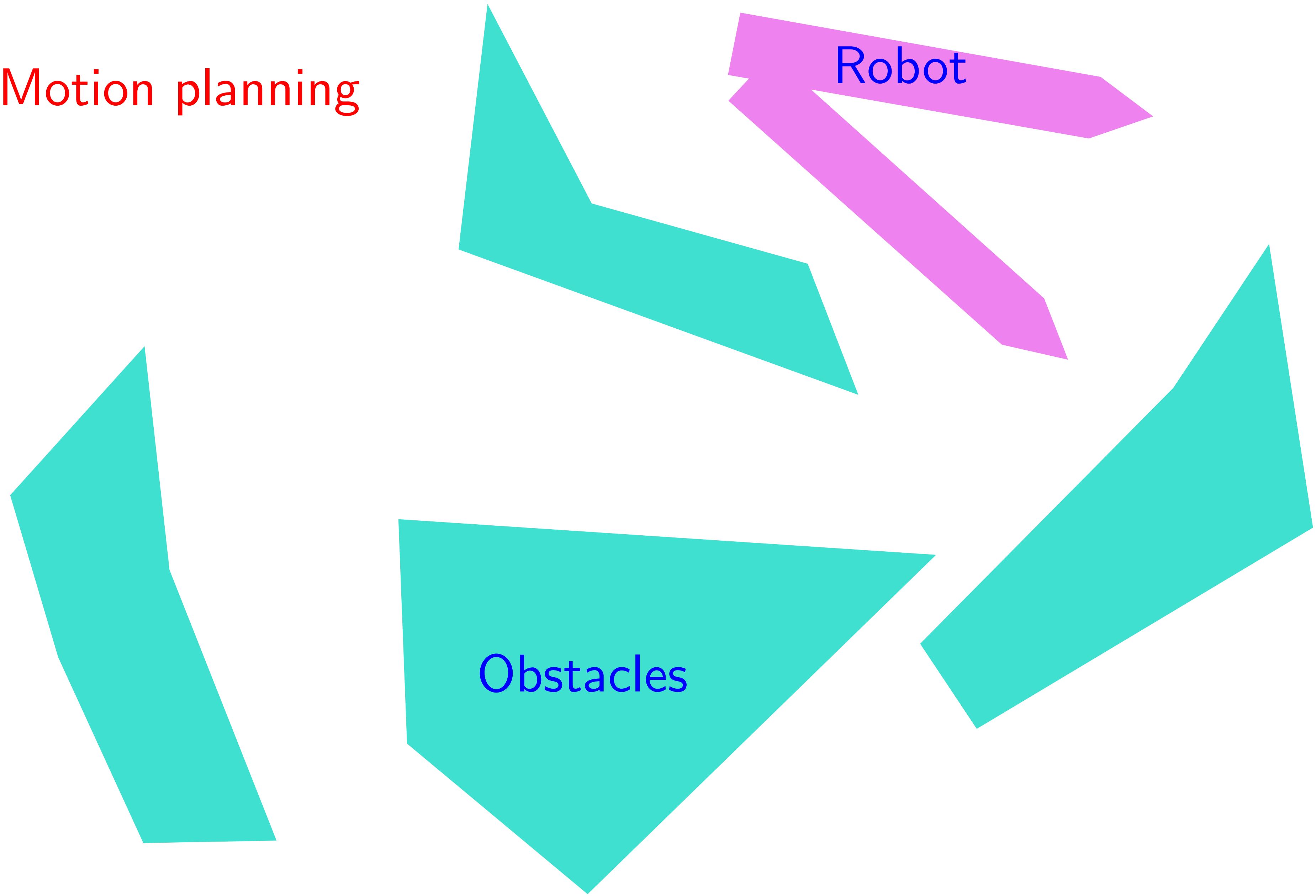
Computational geometry usage

Motion planning



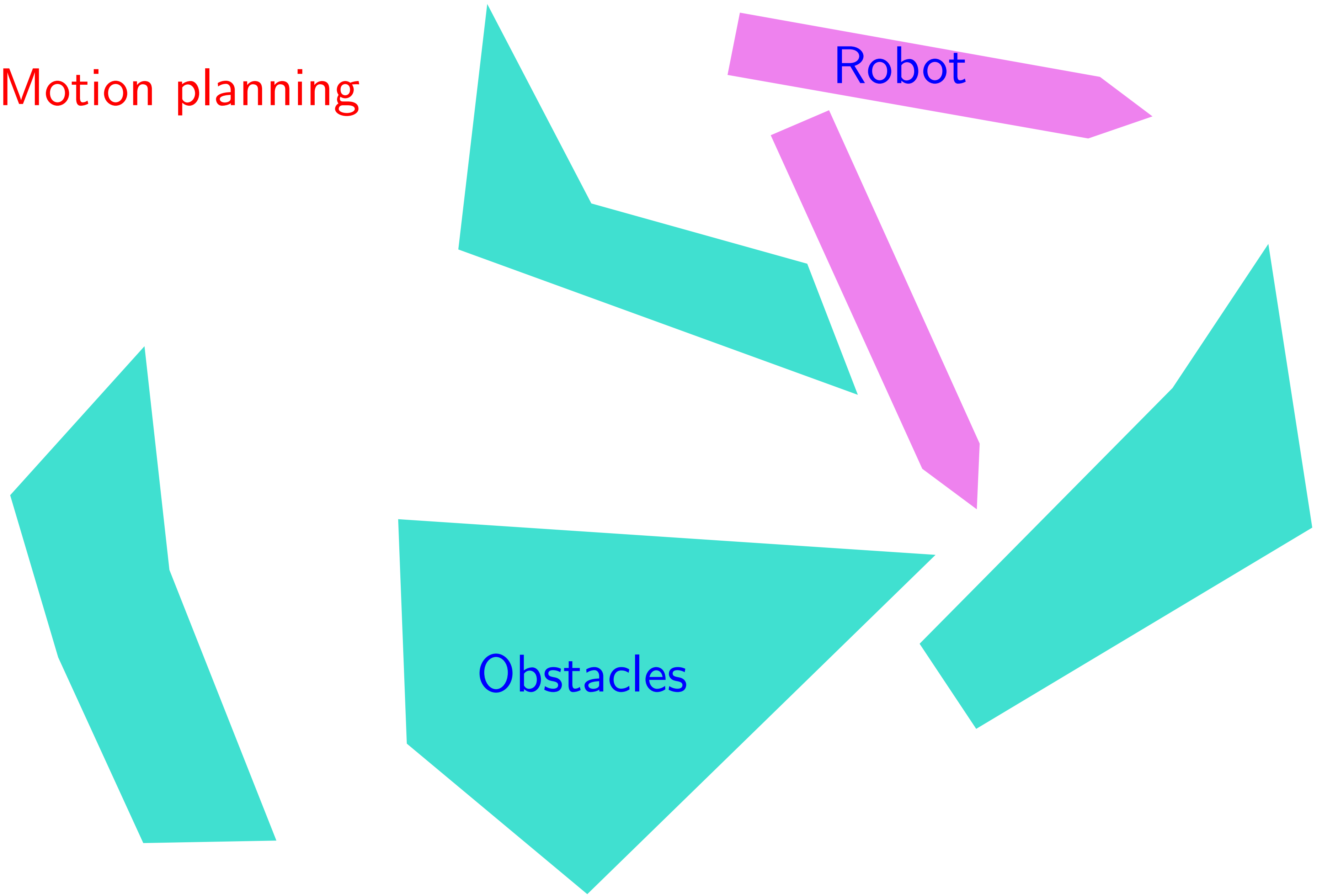
Computational geometry usage

Motion planning



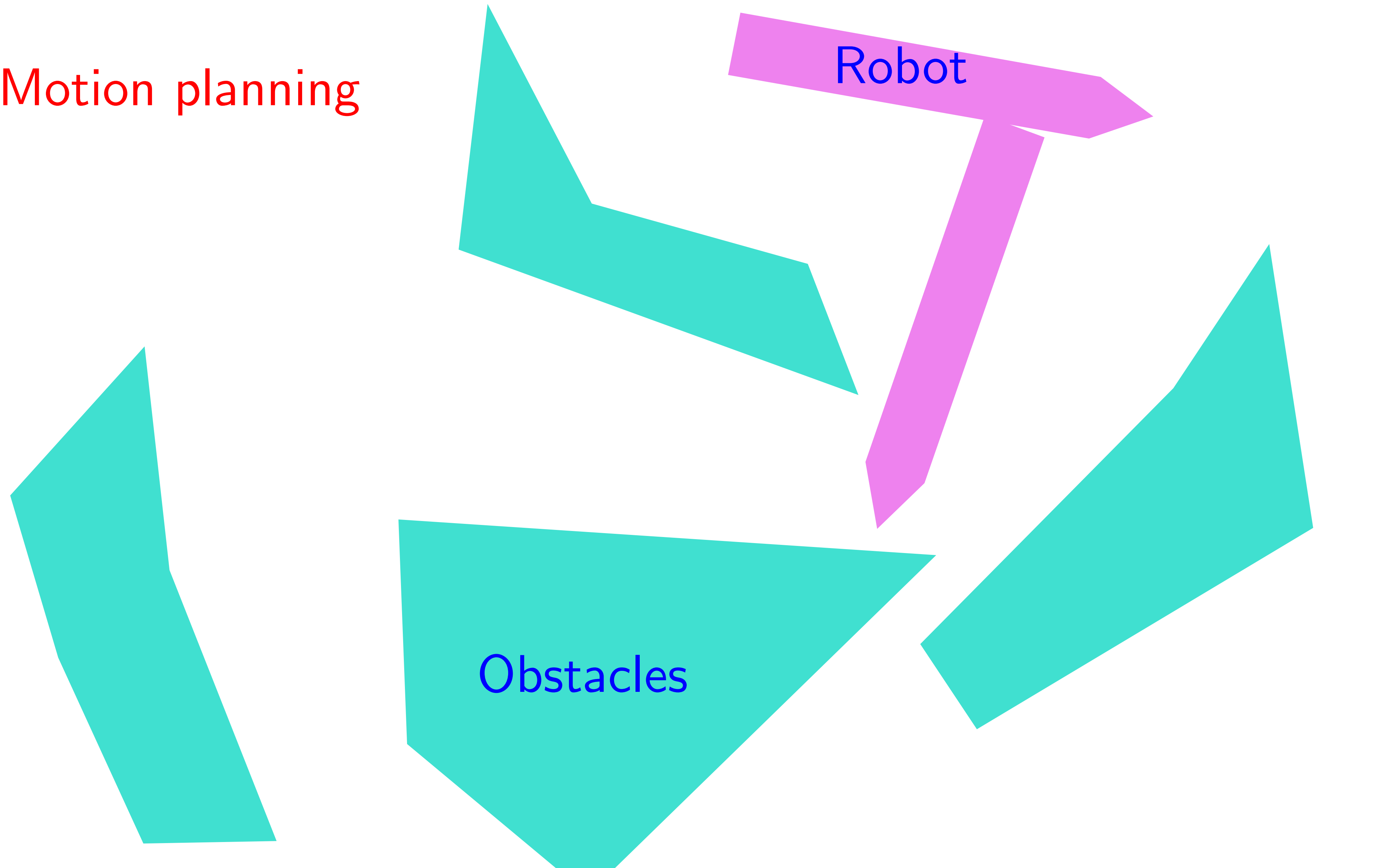
Computational geometry usage

Motion planning



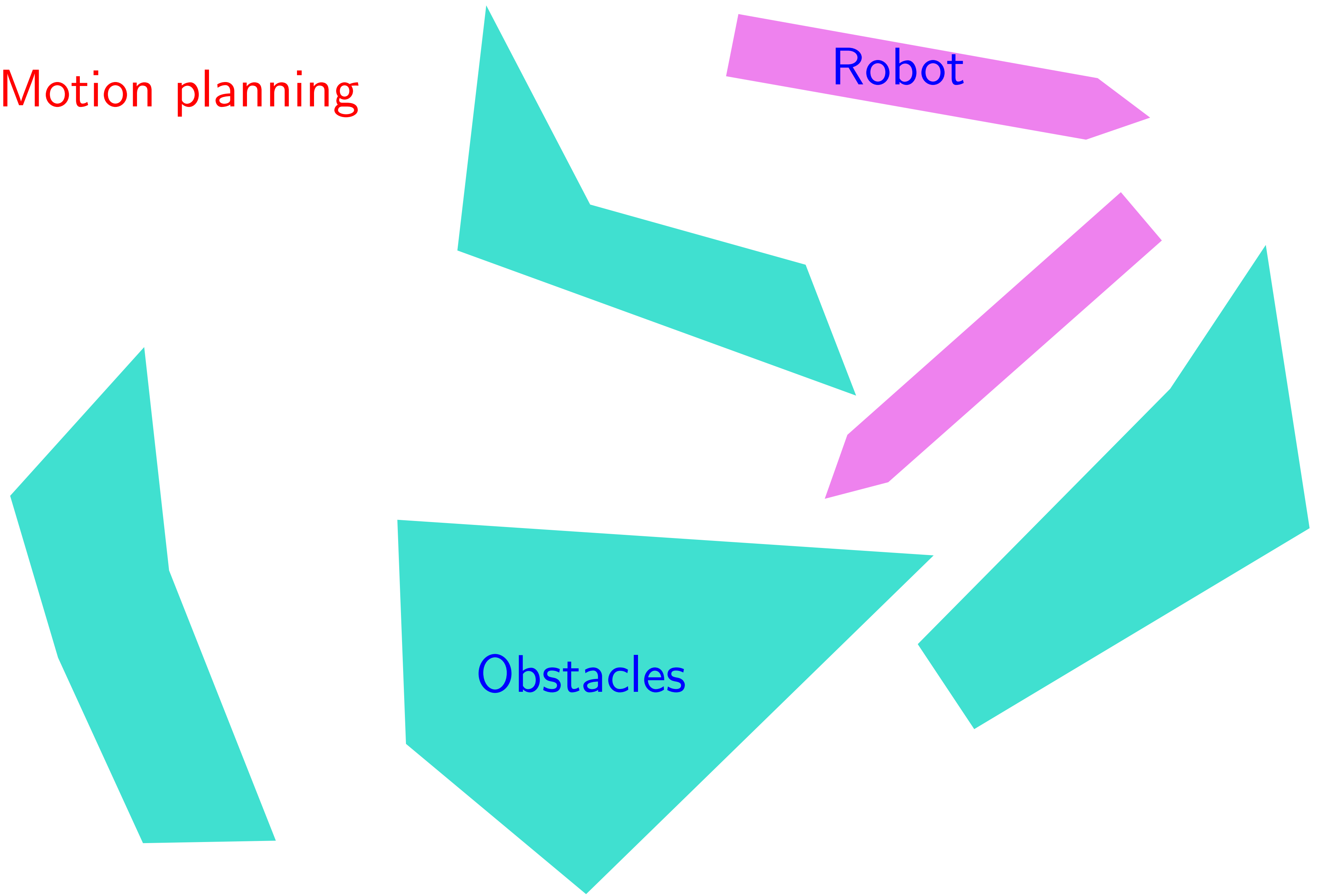
Computational geometry usage

Motion planning



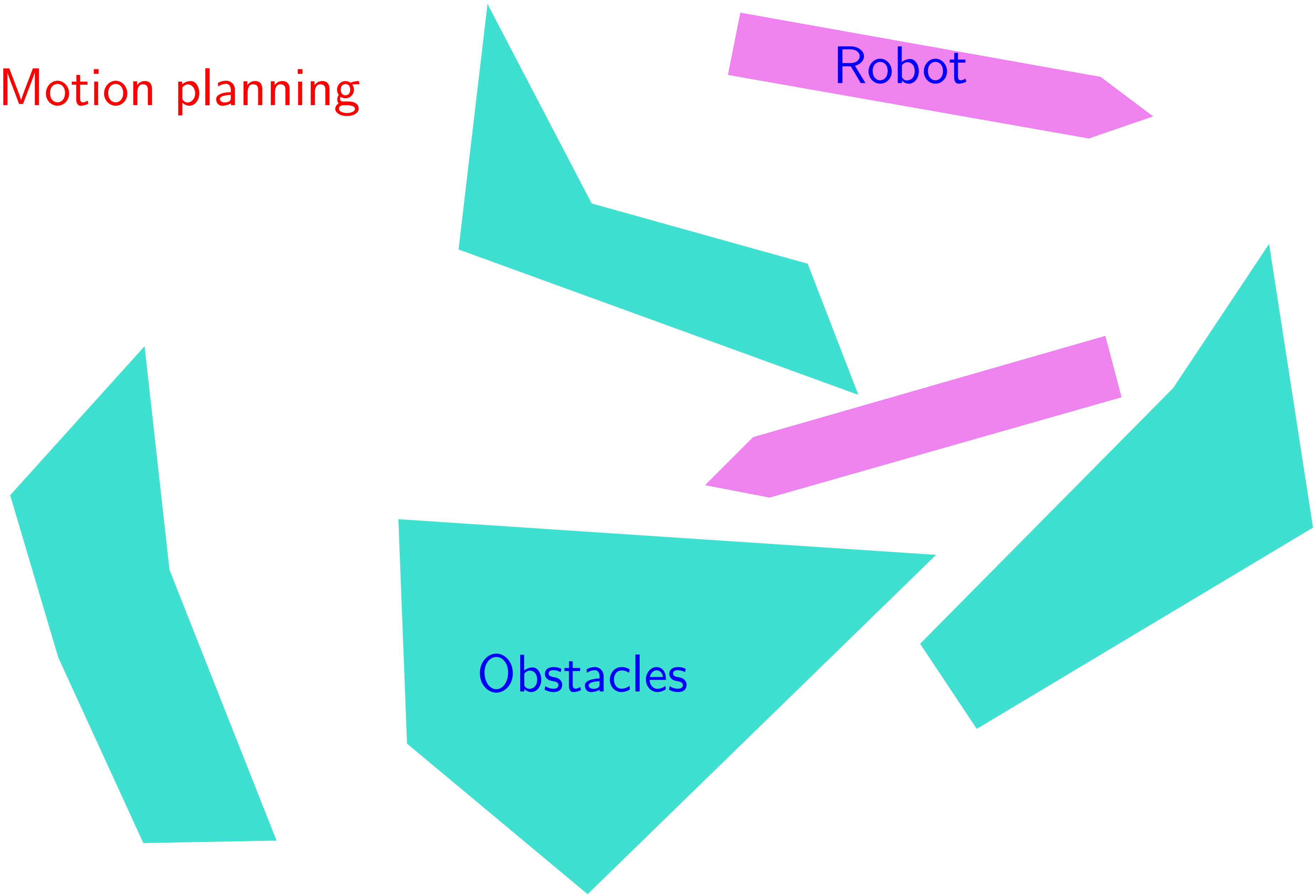
Computational geometry usage

Motion planning

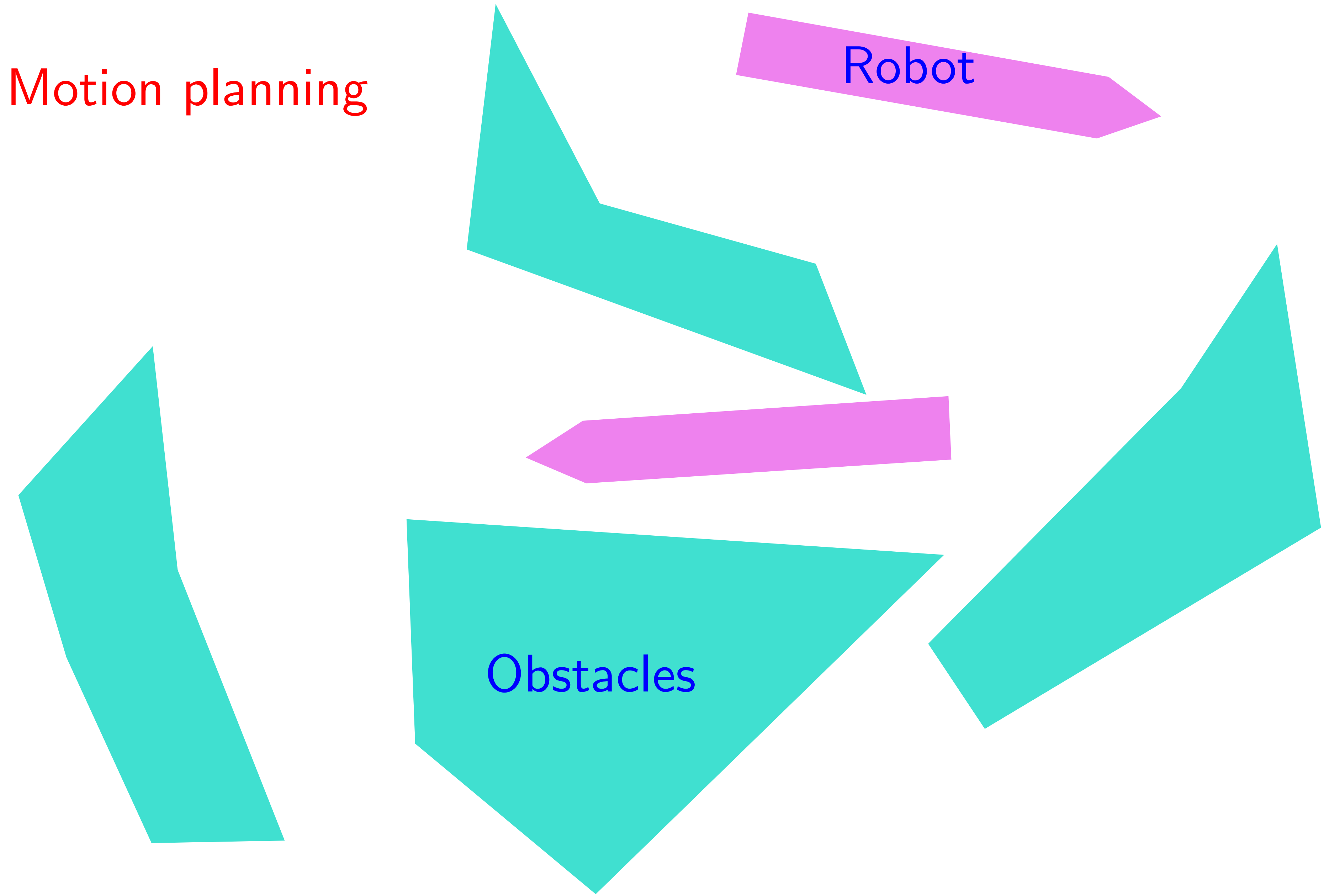


Computational geometry usage

Motion planning

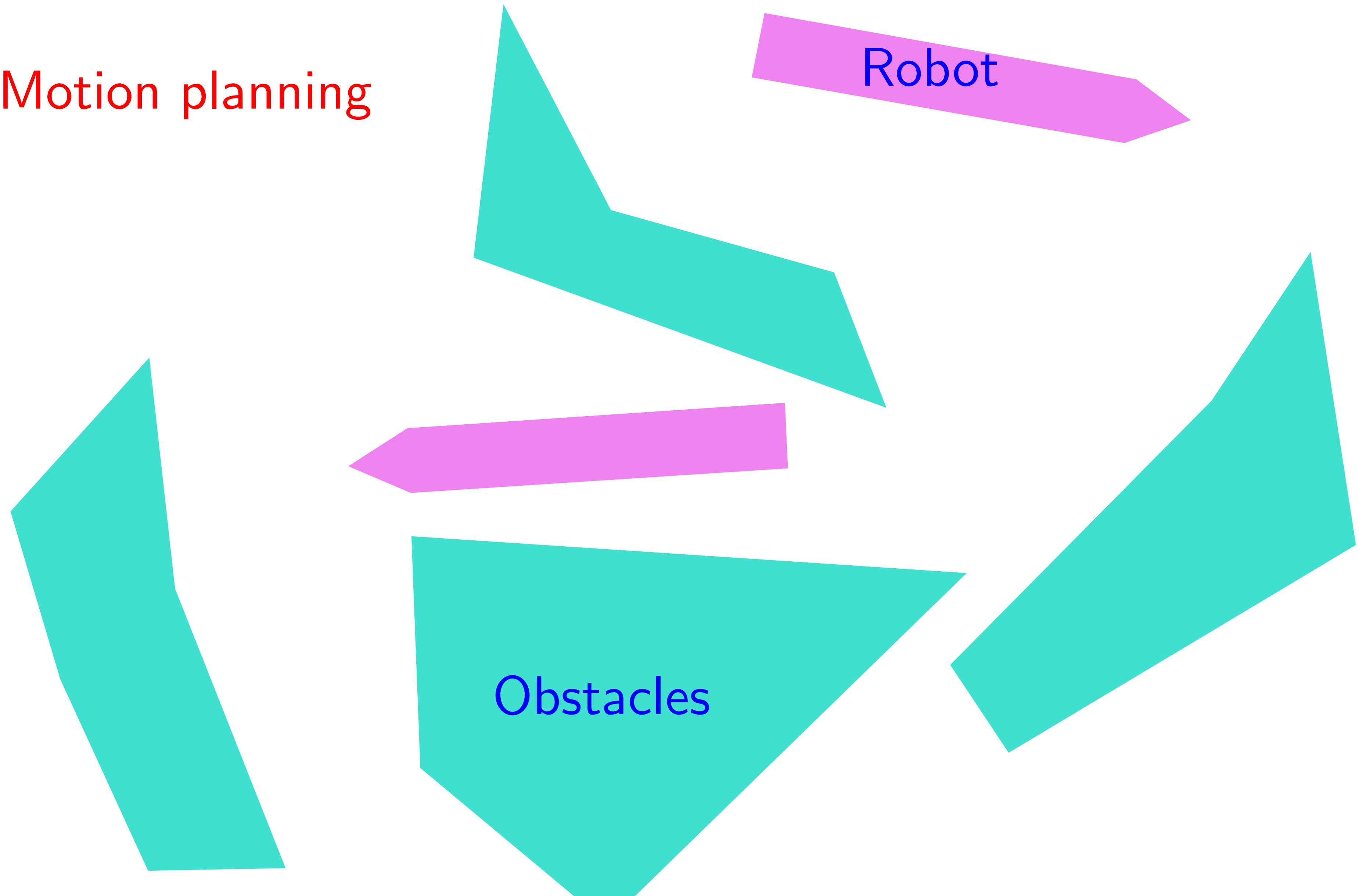


Computational geometry usage



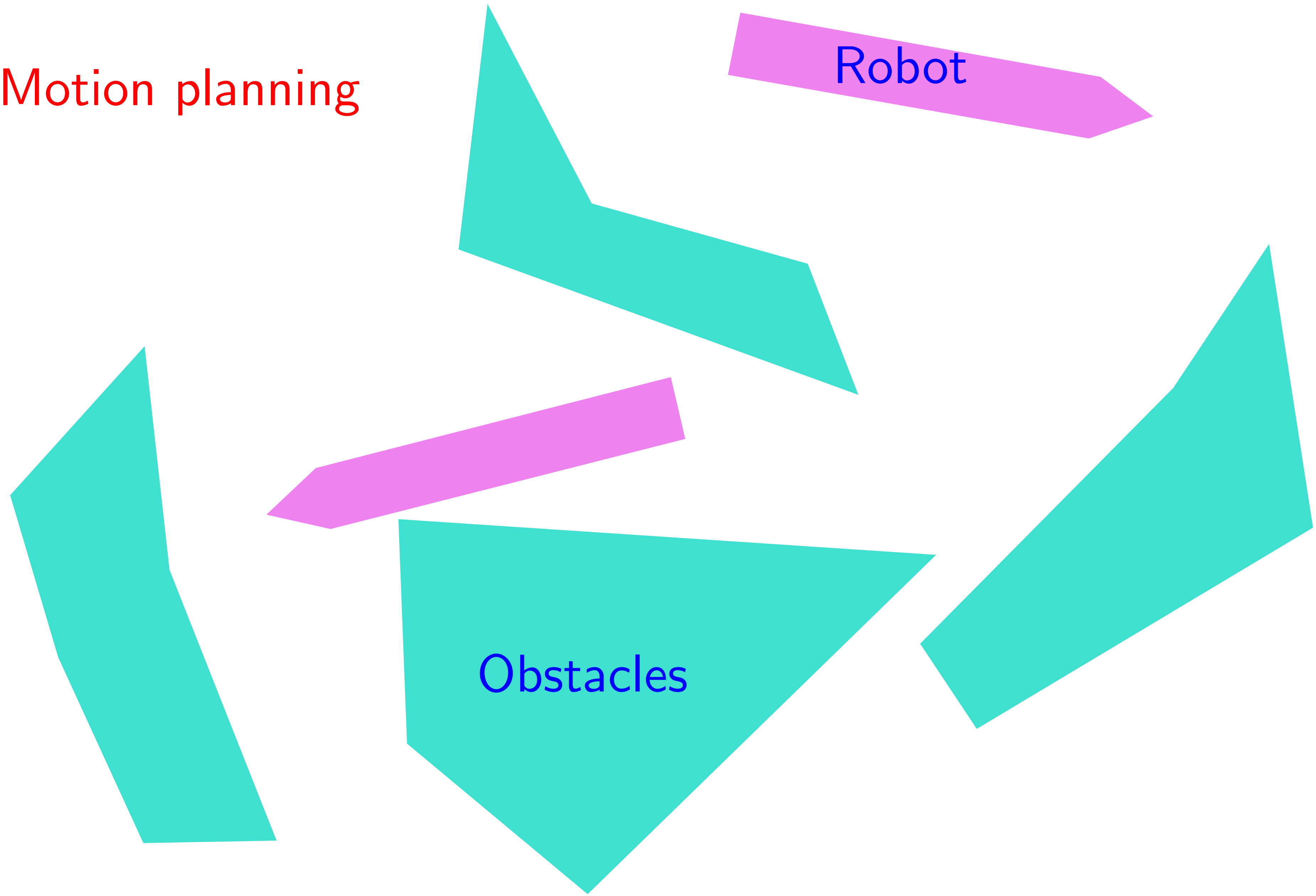
Computational geometry usage

Motion planning



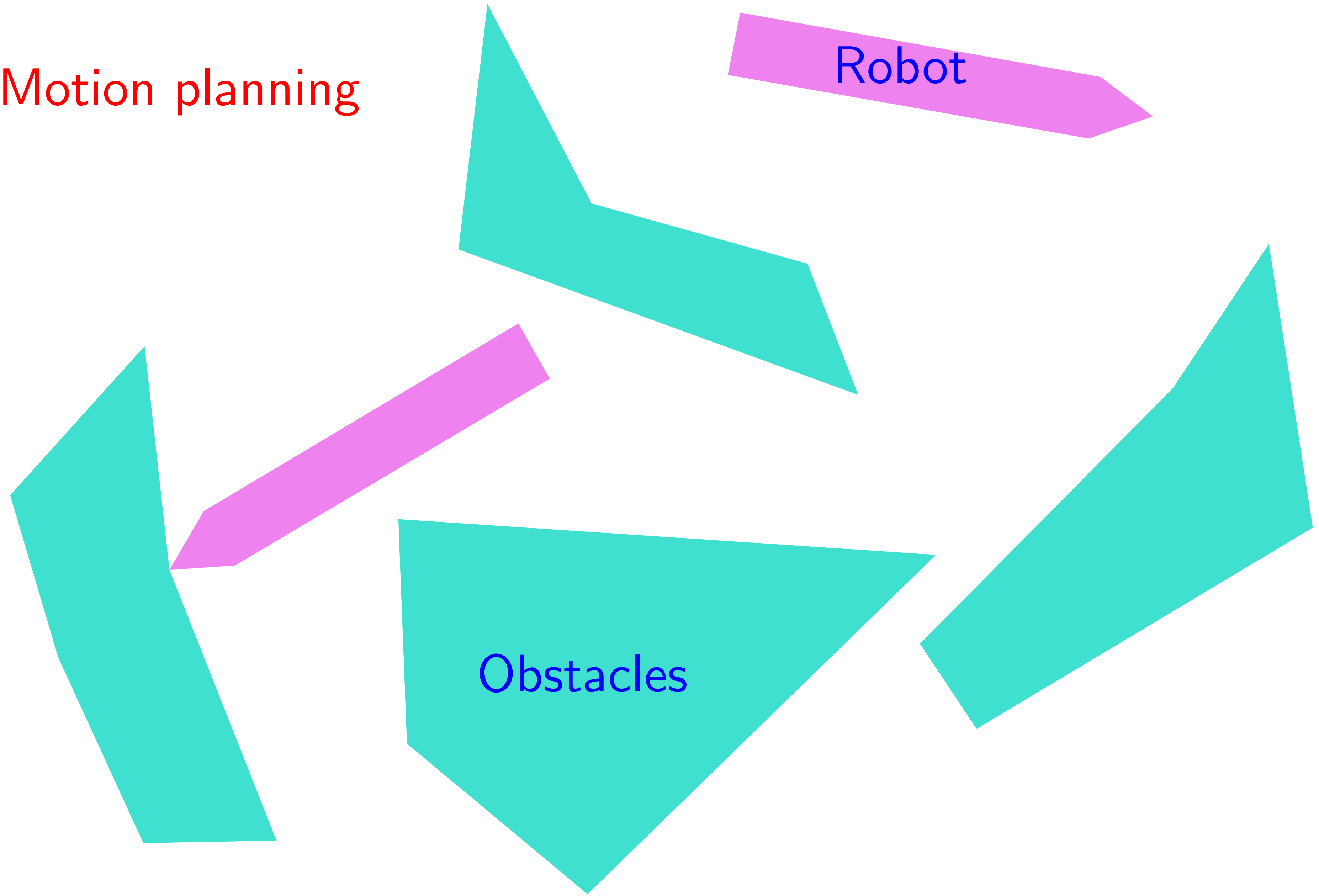
Computational geometry usage

Motion planning



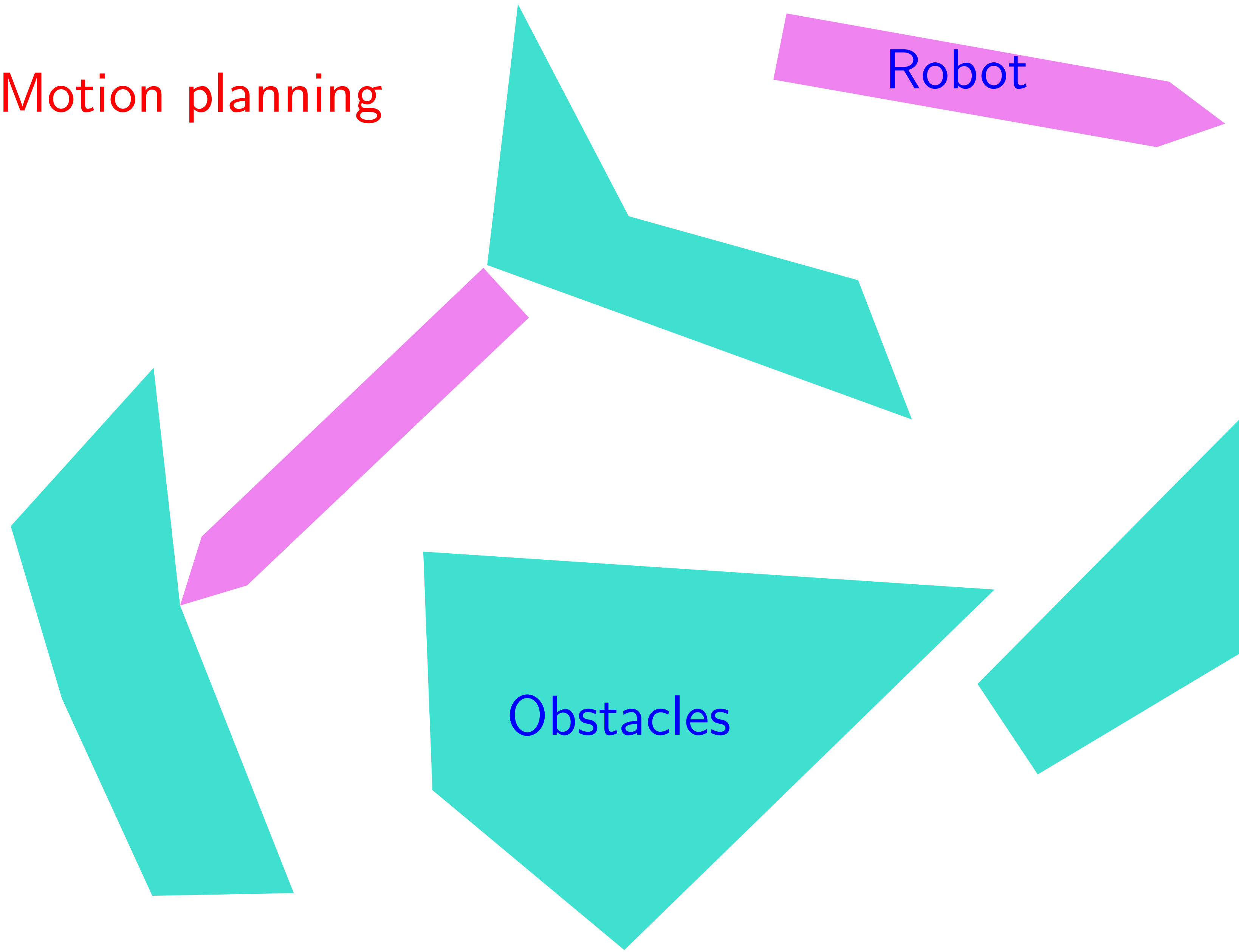
Computational geometry usage

Motion planning



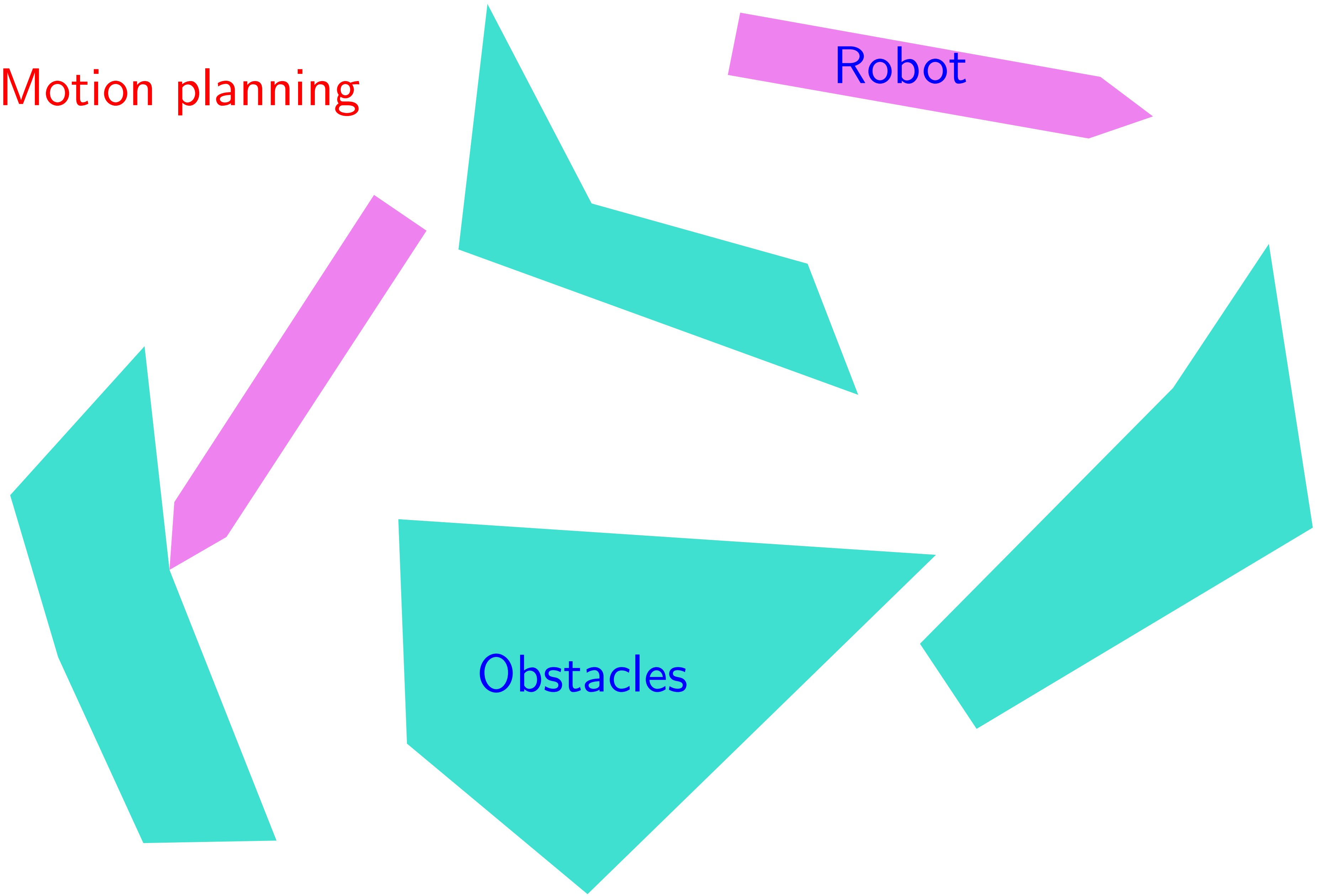
Computational geometry usage

Motion planning



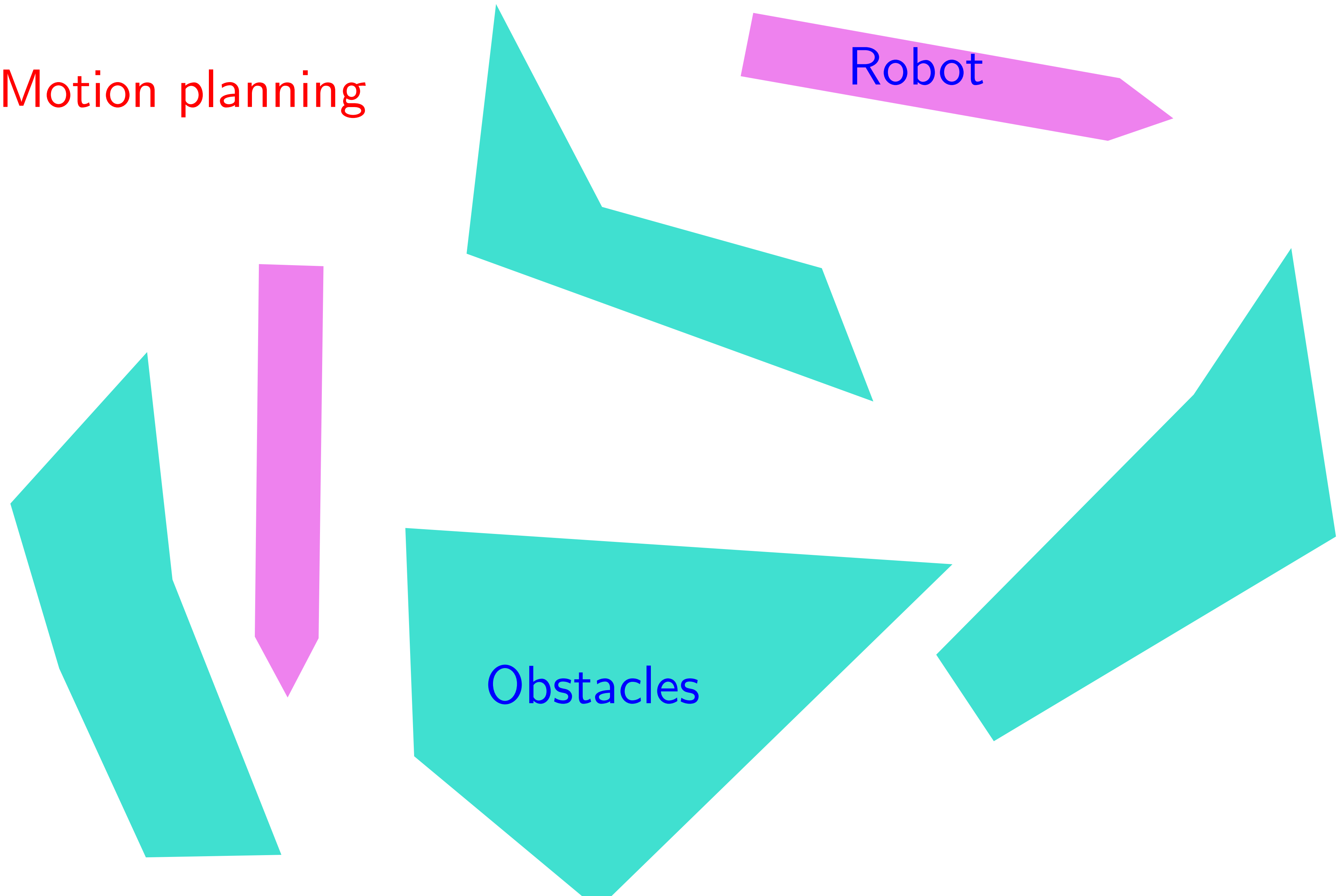
Computational geometry usage

Motion planning



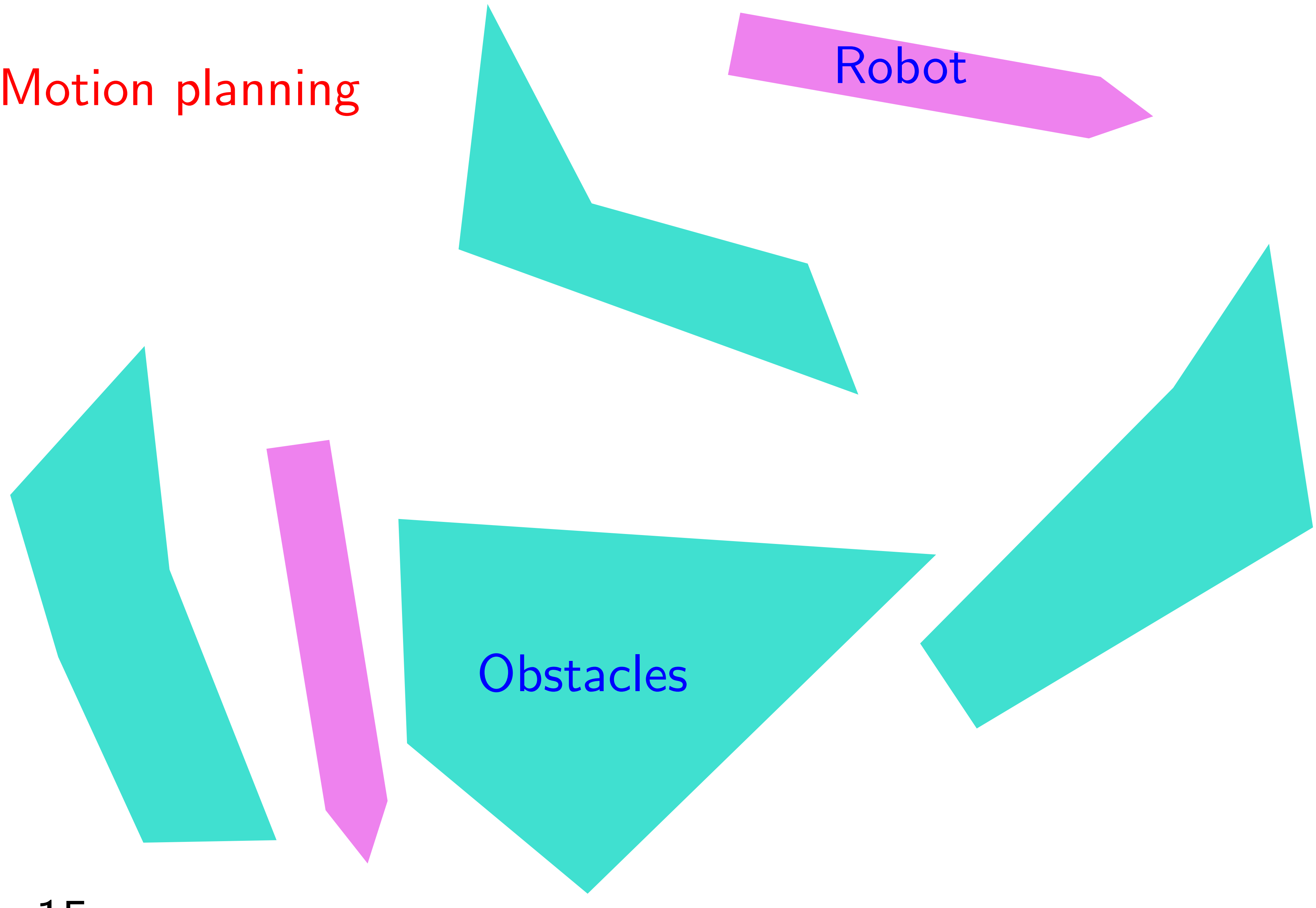
Computational geometry usage

Motion planning



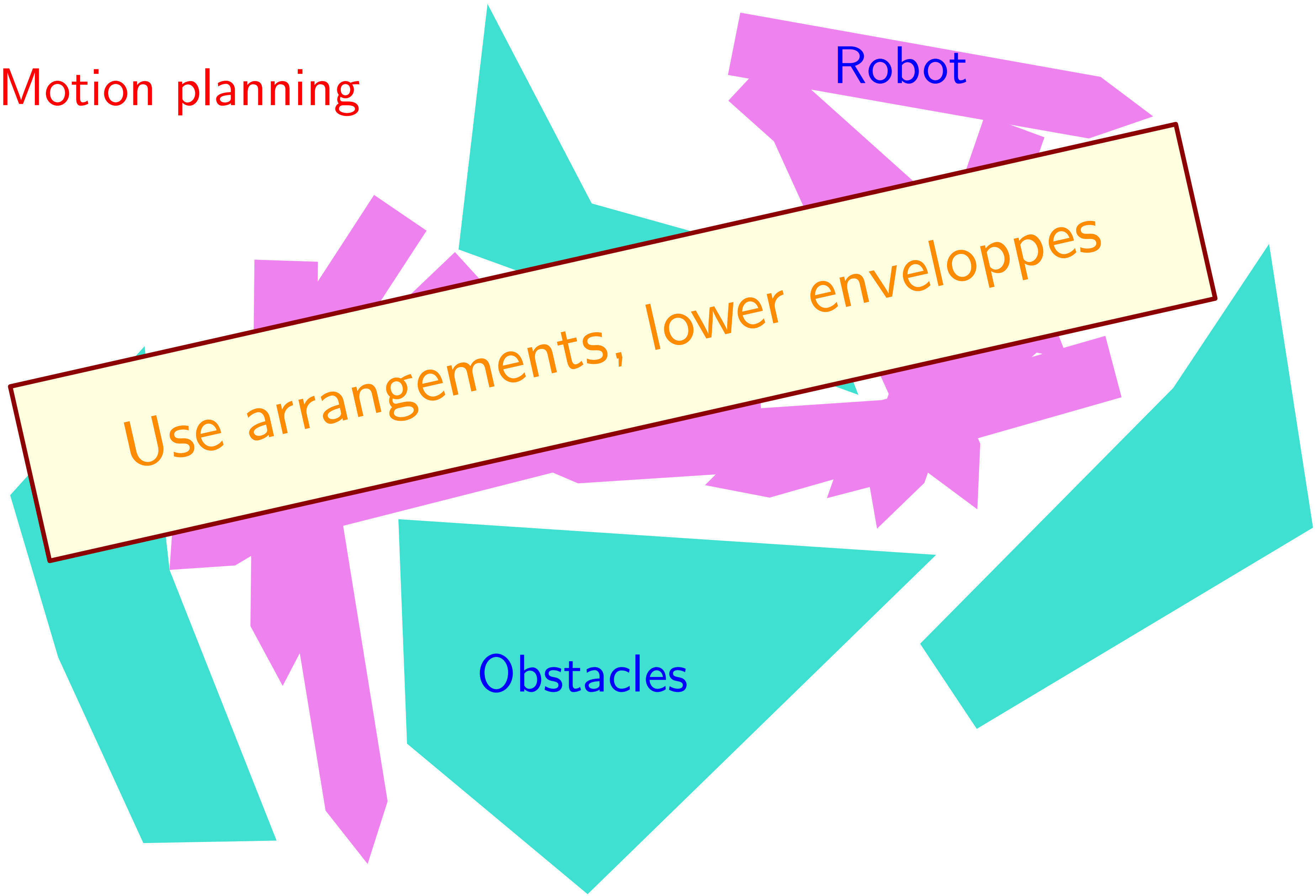
Computational geometry usage

Motion planning



Computational geometry usage

Motion planning



Use arrangements, lower envelopes

Computational geometry, 1975-1985

Complicated algorithms

Worst case complexities

Asymptotic complexities

Real RAM model

Lower bounds

General position hypothesis

Computational geometry, 1975-1985

Complicated algorithms	
Worst case complexities	Fit real life data
Asymptotic complexities	For n big enough
Real RAM model	Does it exist
Lower bounds	Real life data
General position hypothesis	Don't degeneracies exist

Computational geometry, 1975-1985

Complicated algorithms	
Worst case complexities	Fit real life data?
Asymptotic complexities	For n big enough?
Real RAM model	Does it exist?
Lower bounds	Real life data?
General position hypothesis	Don't degeneracies exist?

Computational geometry, 1975-1985

Complicated algorithms	Not used in practice
Worst case complexities	Fit real life data ?
Asymptotic complexities	For n big enough ?
Real RAM model	Does it exist ?
Lower bounds	Real life data ?
General position hypothesis	Don't degeneracies exist ?

Computational geometry, 1985-2000

Complicated algorithms

Worst case complexities

Asymptotic complexities

Real RAM model

Lower bounds

General position hypothesis

Computational geometry, 1985-2000

Simpler

~~Complicated~~ algorithms

Worst case complexities

Asymptotic complexities

Real RAM model

Lower bounds

General position hypothesis

Computational geometry, 1985-2000

Simpler

~~Complicated~~ algorithms

Worst case complexities ← randomized

Asymptotic complexities

Real RAM model

Lower bounds

General position hypothesis

Computational geometry, 1985-2000

Simpler

~~Complicated~~ algorithms

randomized

Worst case complexities

Asymptotic complexities

Real RAM model

Lower bounds

General position hypothesis

Randomization lecture

Computational geometry, 1985-2000

Complicated algorithms

Worst case complexities

Asymptotic complexities

Real RAM model — address robustness issues

Lower bounds

General position hypothesis — solve degeneracies

Computational geometry, 1985-2000

Complicated algorithms

Worst case complexities

Asymptotic

Robustness lecture

Real RAM model — address robustness issues

Lower bounds

General position hypothesis — solve degeneracies

Computational geometry, 1985-2000

Complicated algorithms

Worst case complexities

Asymptotic complexities   Just really code it

Real RAM model

Lower bounds

General position hypothesis

Computational geometry, 1985-2000

Complicated algorithms

Worst case complexities

Asymptotic complexities — Just really code it

Real RAM model

Lower bounds

General position hypothesis

CGAL

Computational geometry, 2000-

Complicated algorithms

Worst case complexities

Asymptotic complexities

Real RAM model

Lower bounds

General position hypothesis

Computational geometry, 2000-

Complicated algorithms

Worst case complexities | Probabilistic hypotheses

Asymptotic complexities

Real RAM model

Lower bounds

General position hypothesis

Computational geometry, 2000-

Complicated algorithms

Worst case complexities — Probabilistic hypotheses

Old (and recent) math literature

Asymptotic complexities

Real RAM model

Lower bounds

General position hypothesis

Computational geometry, 2000-

Complicated algorithms

Worst case complexities — Probabilistic hypotheses

Old (and recent) math literature

Asymptotic complexities

Real RAM model

Lower bounds

General position hypothesis

Computational geometry, 2000-

Complicated algorithms

Beyond the Euclidean realm

Worst case complexities

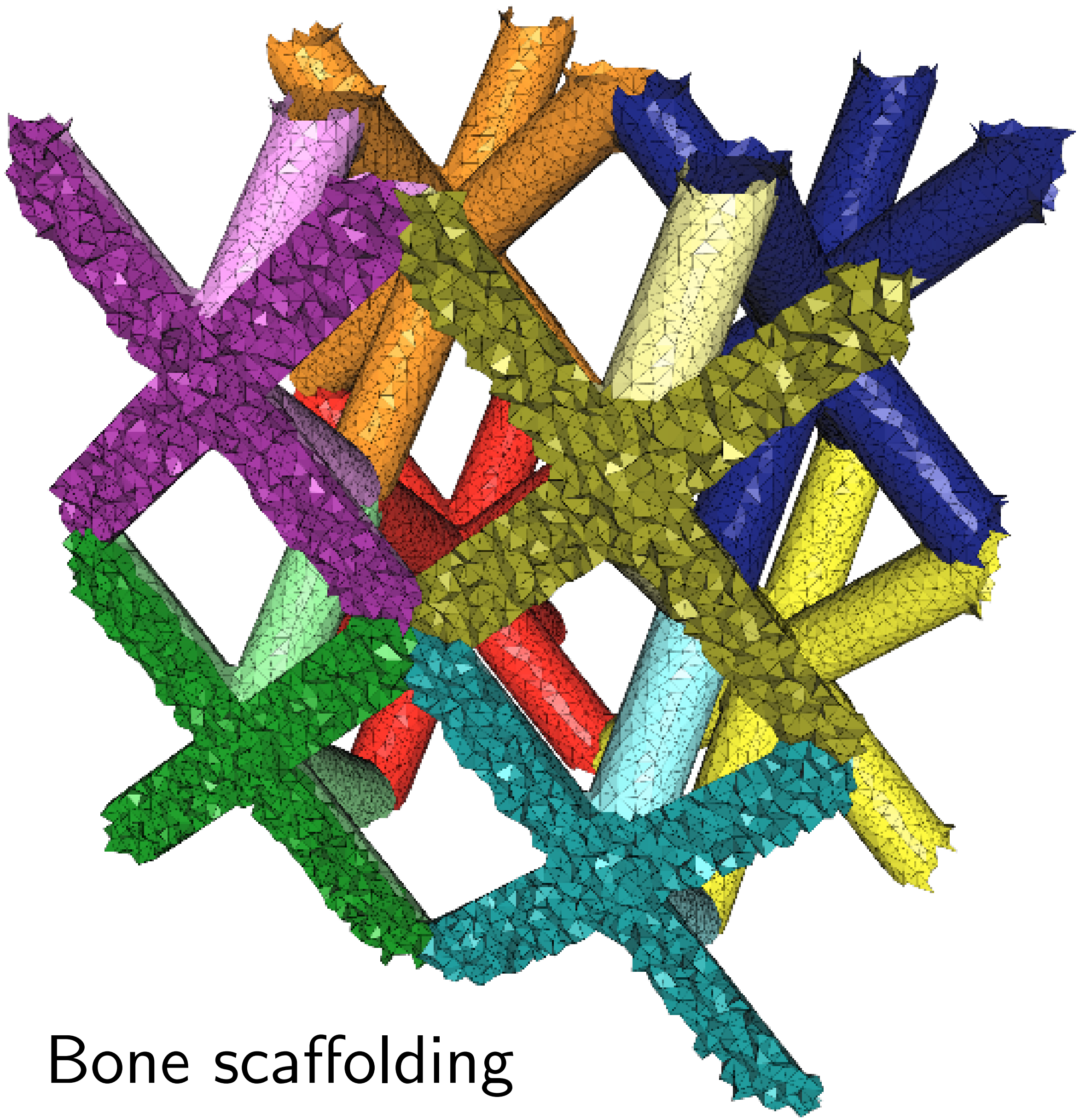
Asymptotic complexities

Real RAM model

Lower bounds

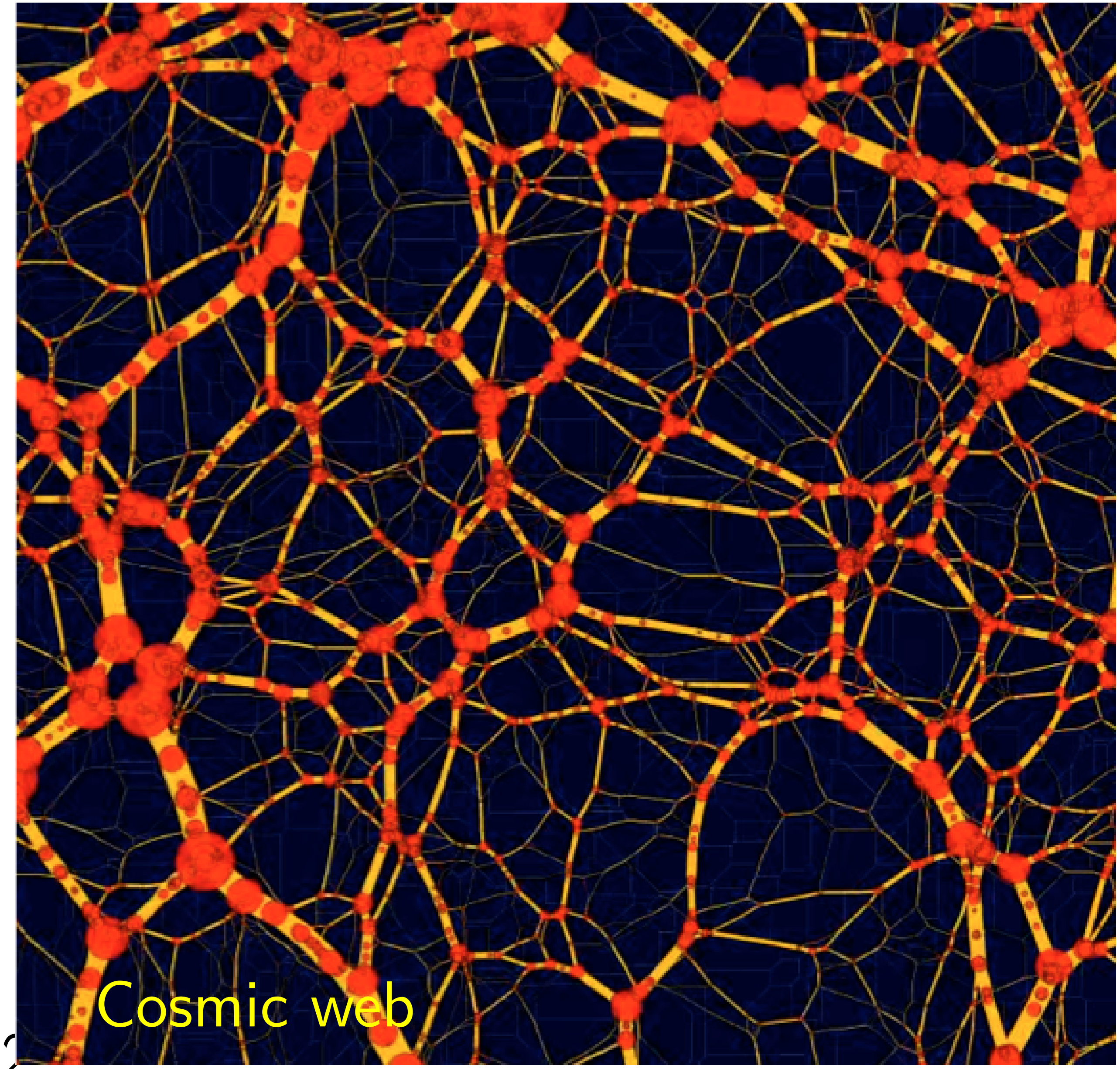
General position hypothesis

Computational geometry, 2000-

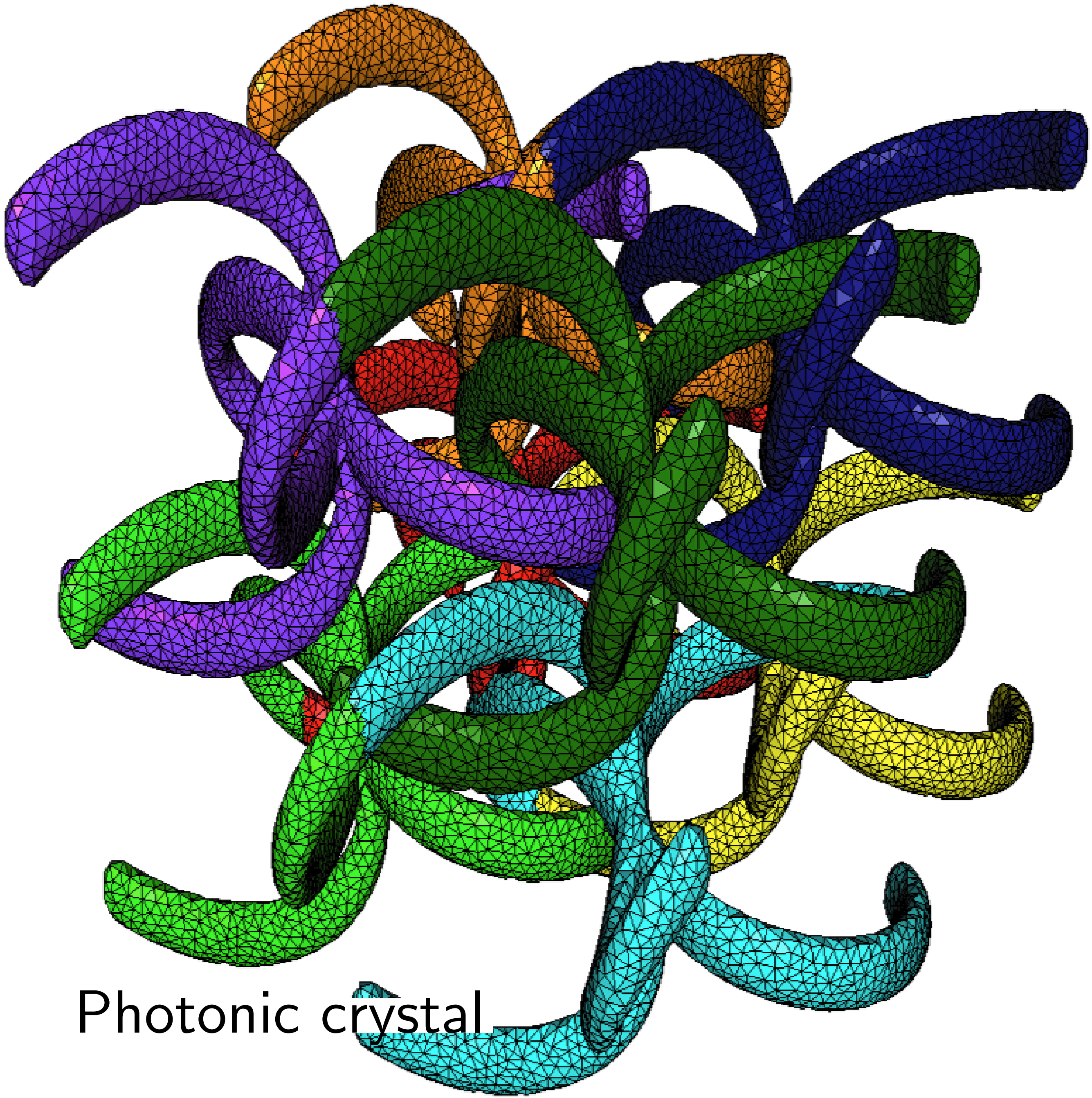


Bone scaffolding

Computational geometry, 2000-



Computational geometry, 2000-



Photonic crystal

Computational geometry, 2000-

Complicated algorithms

Beyond the Euclidean realm

Worst case complexities

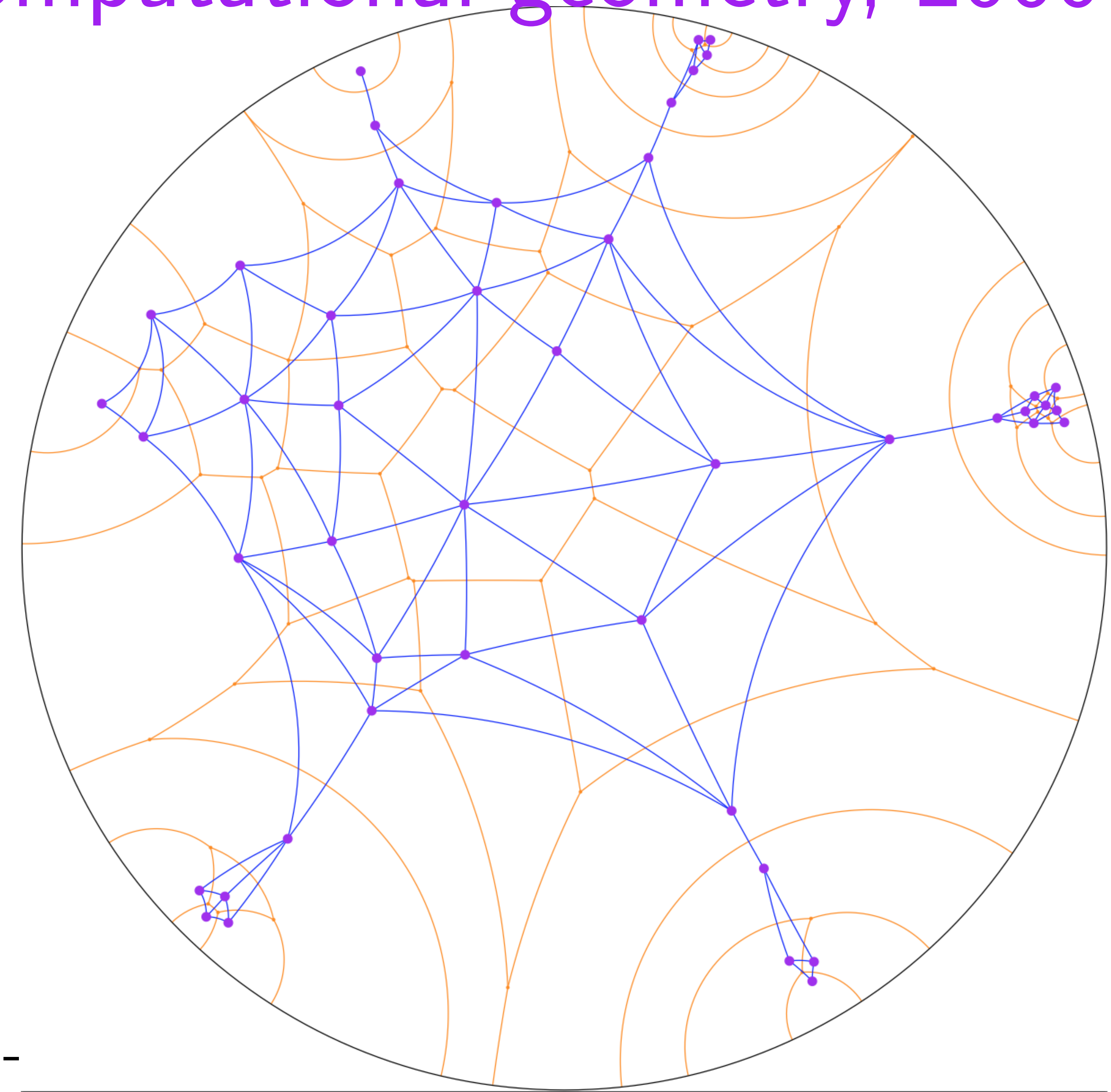
Asymptotic complexities

Real RAM model

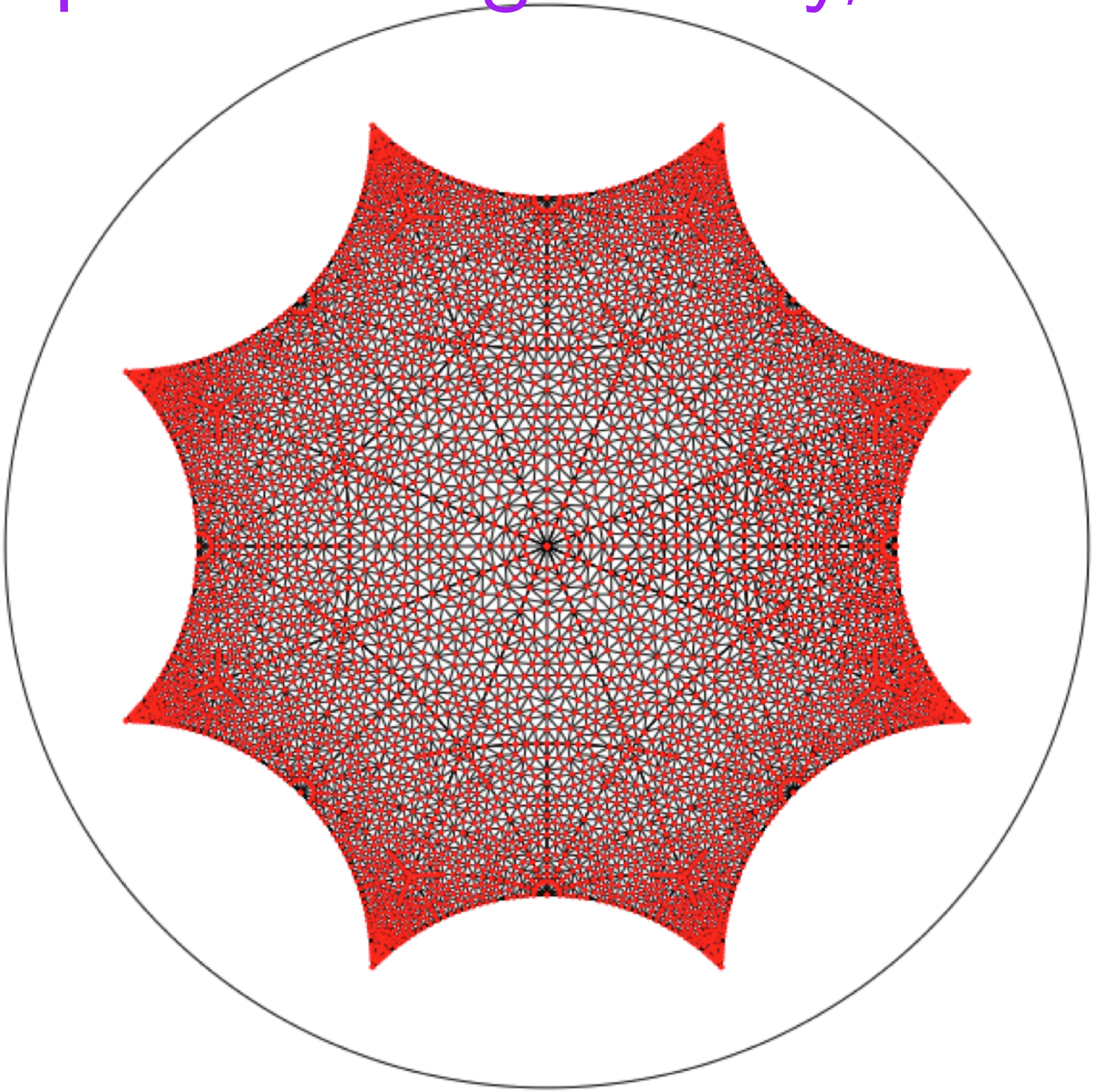
Lower bounds

General position hypothesis

Computational geometry, 2000-



Computational geometry, 2000-

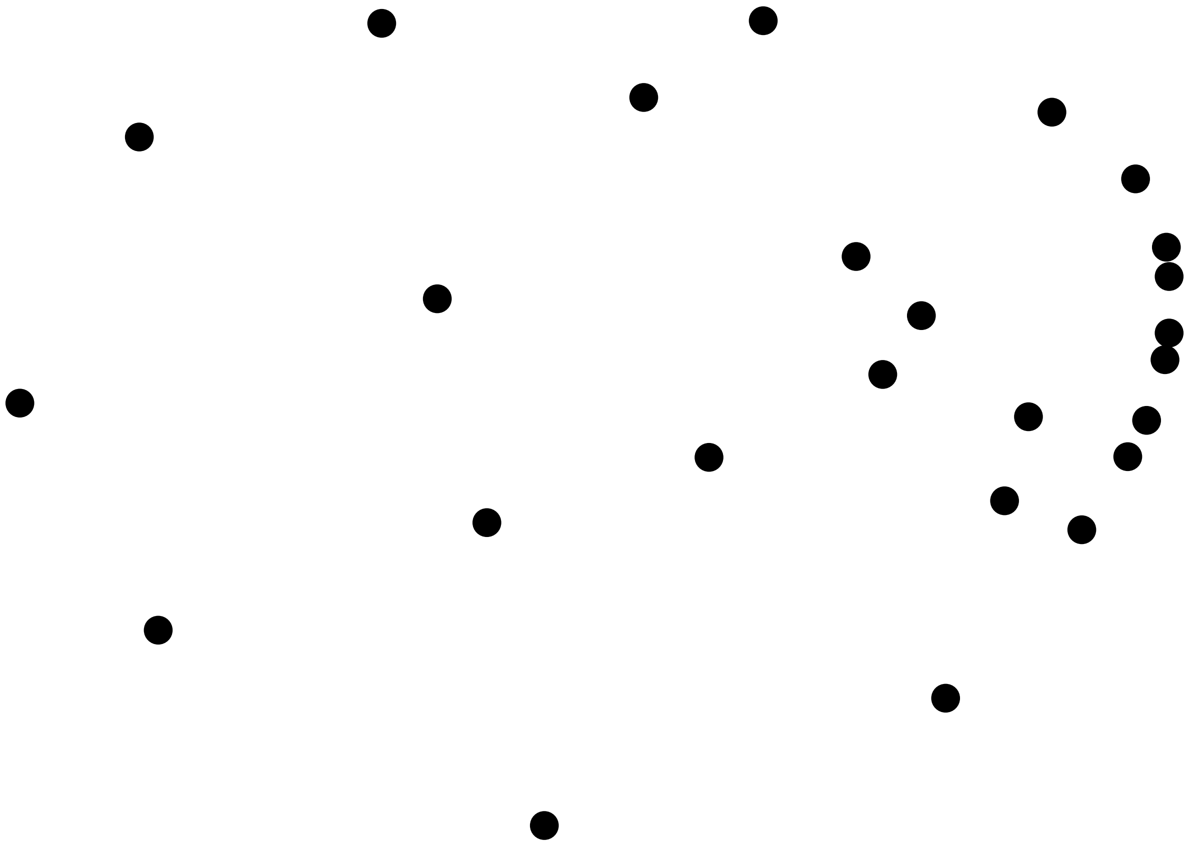


Le sujet d'aujourd'hui...

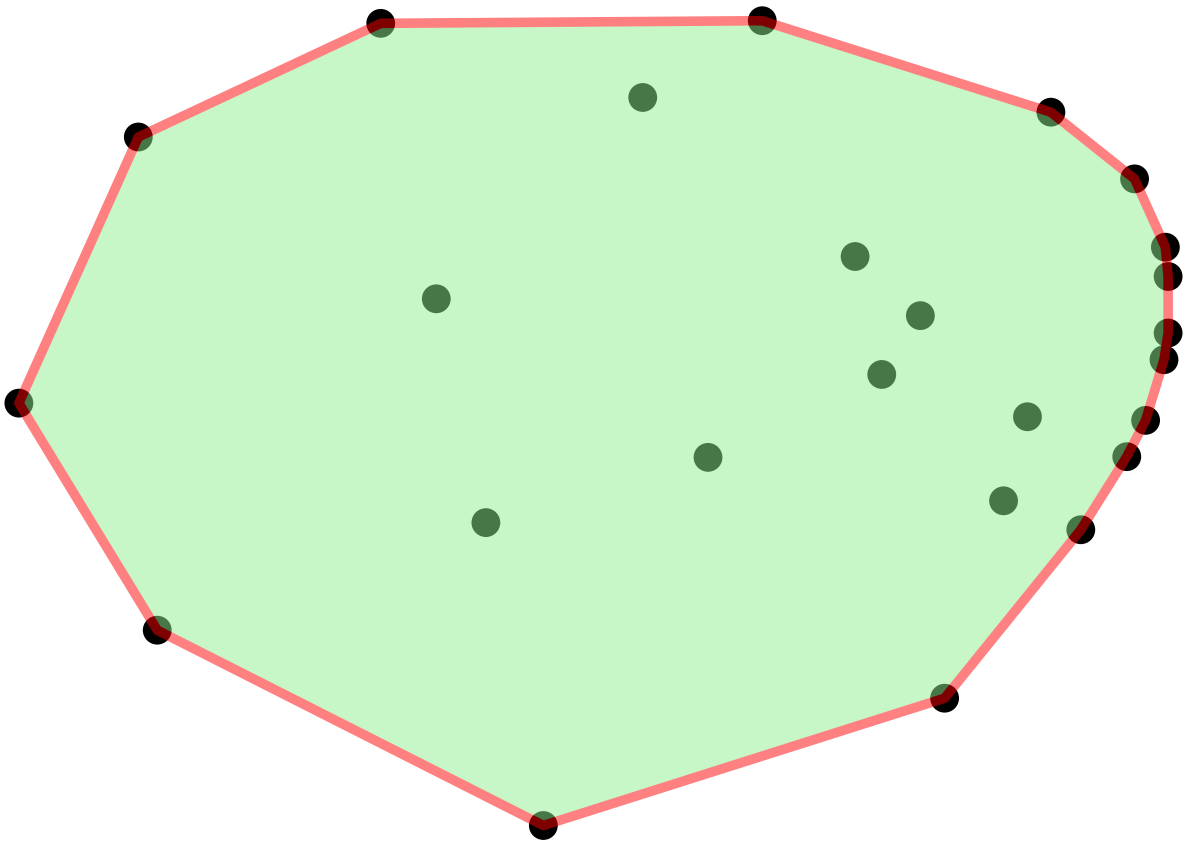
les enveloppes convexes

Convex hull

Convex hull



Convex hull



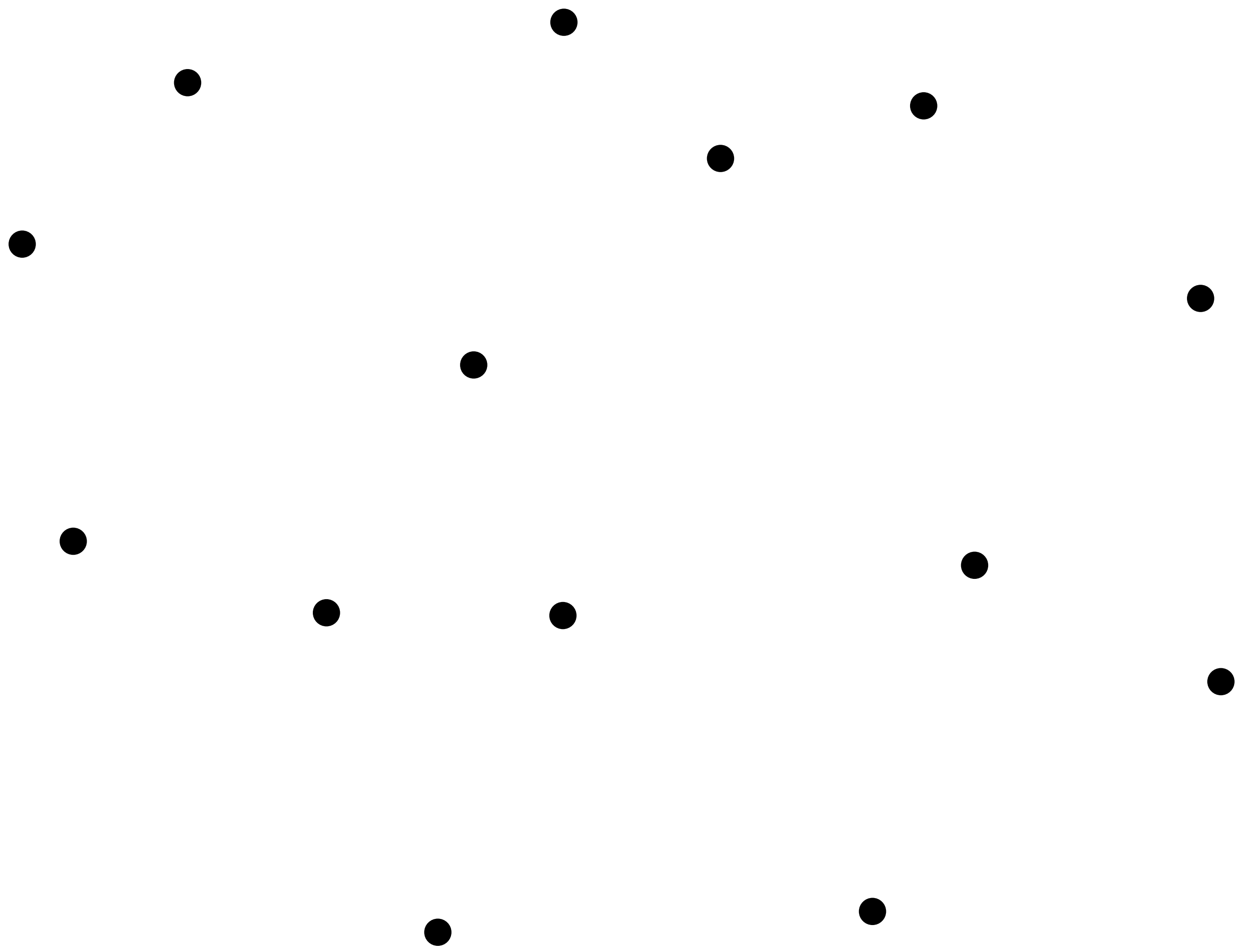
Convex hull

- Definition, extremal point
- Jarvis algorithm
- Orientation predicate
- Buggy degenerate example
- Real RAM model and general position hypothesis
- Graham algorithm
- Lower bound
- Three dimensions

Convex hull

Convex hull

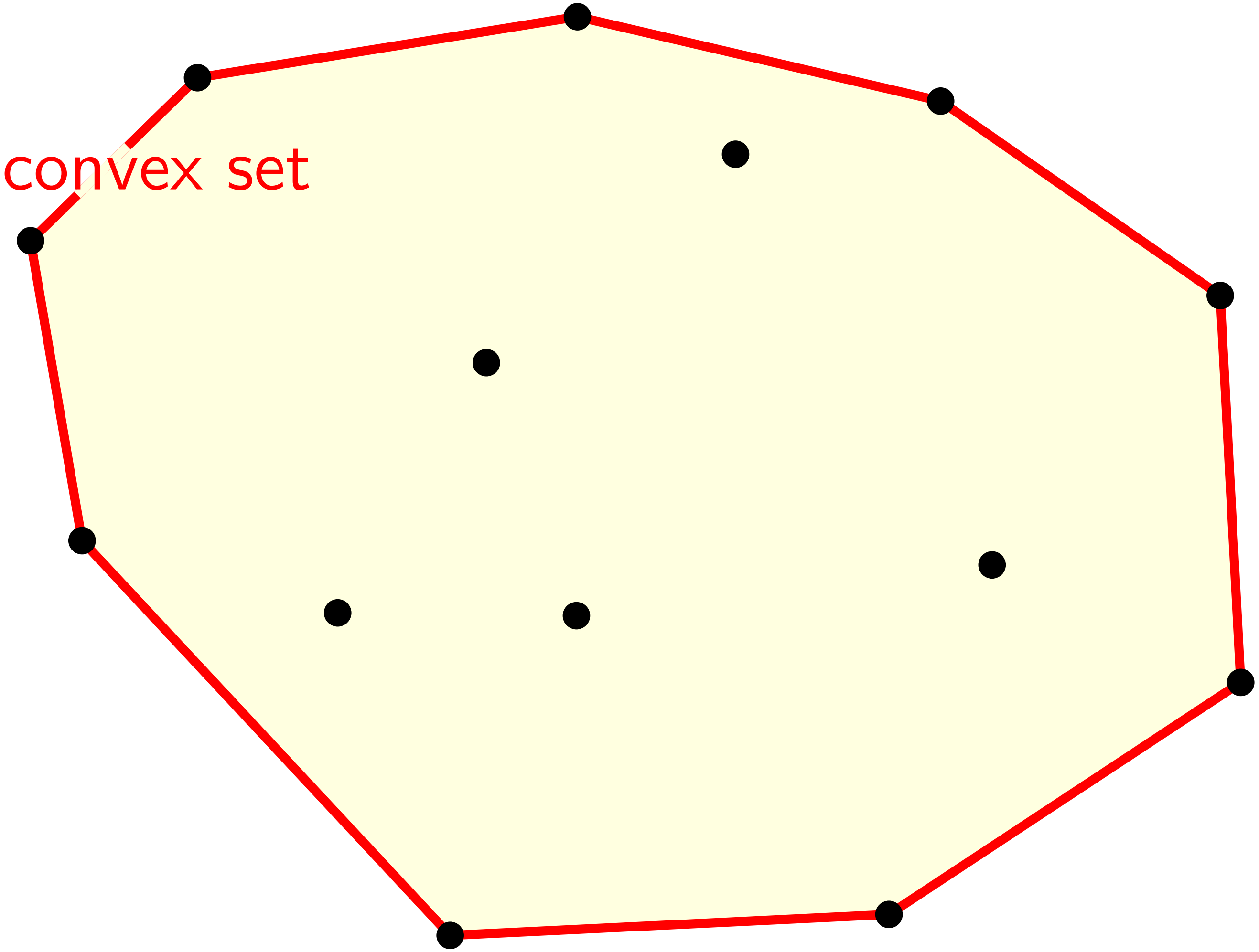
Set of points



Convex hull

Set of points

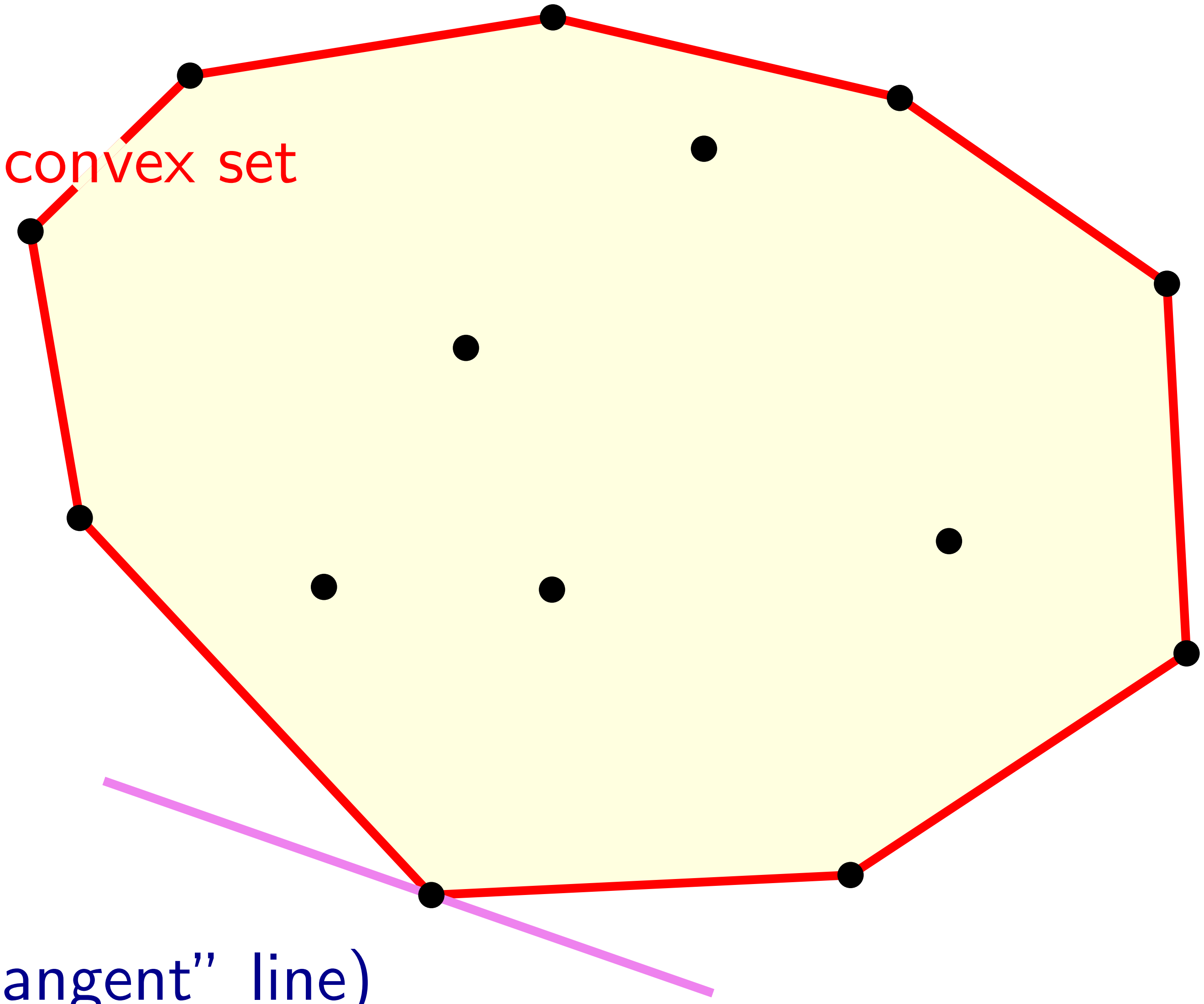
Smallest enclosing convex set



Convex hull

Set of points

Smallest enclosing convex set



Extremal point

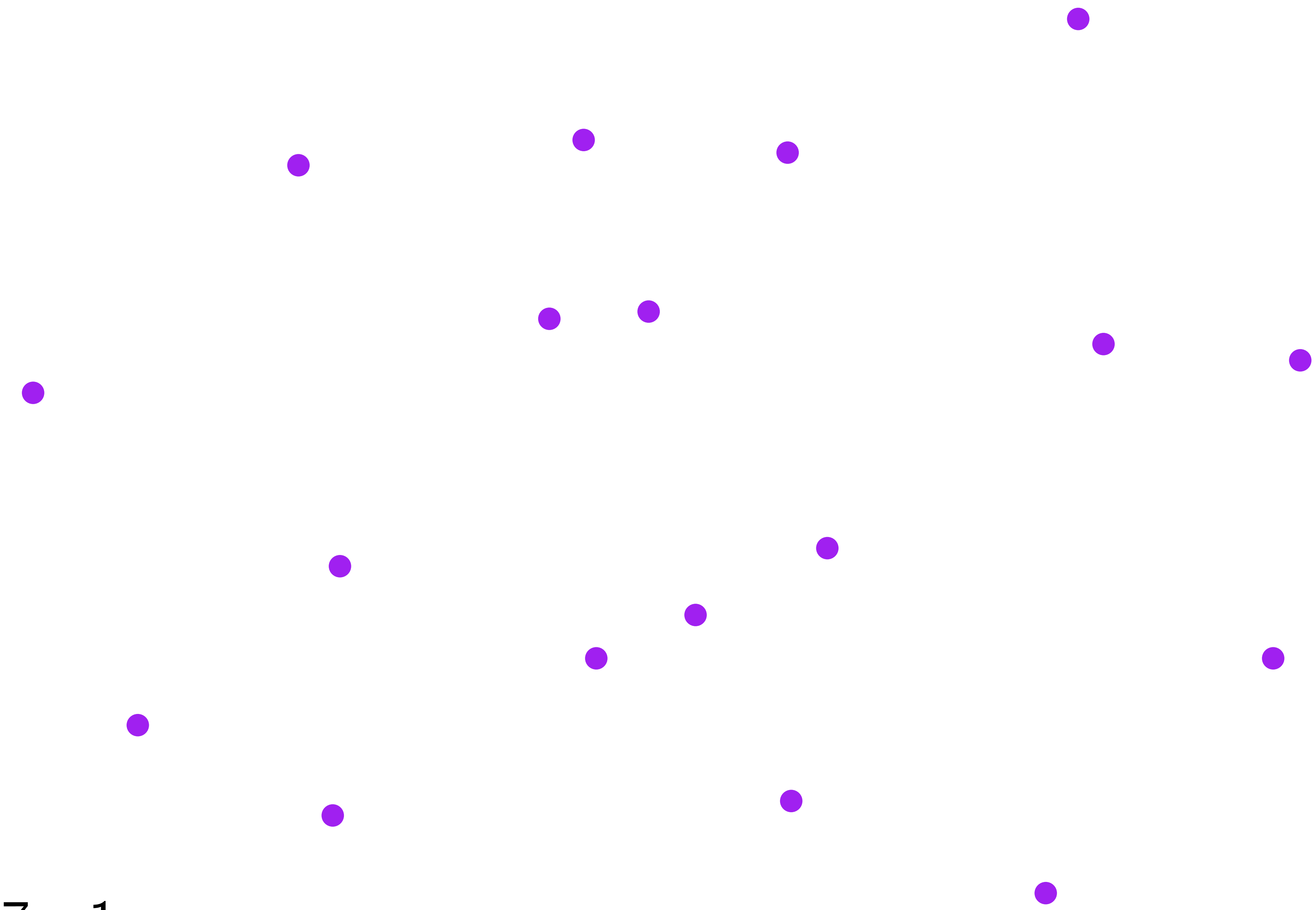
Supporting line (”tangent” line)

Un premier algorithme

(Jarvis, a.k.a. paquet cadeau)

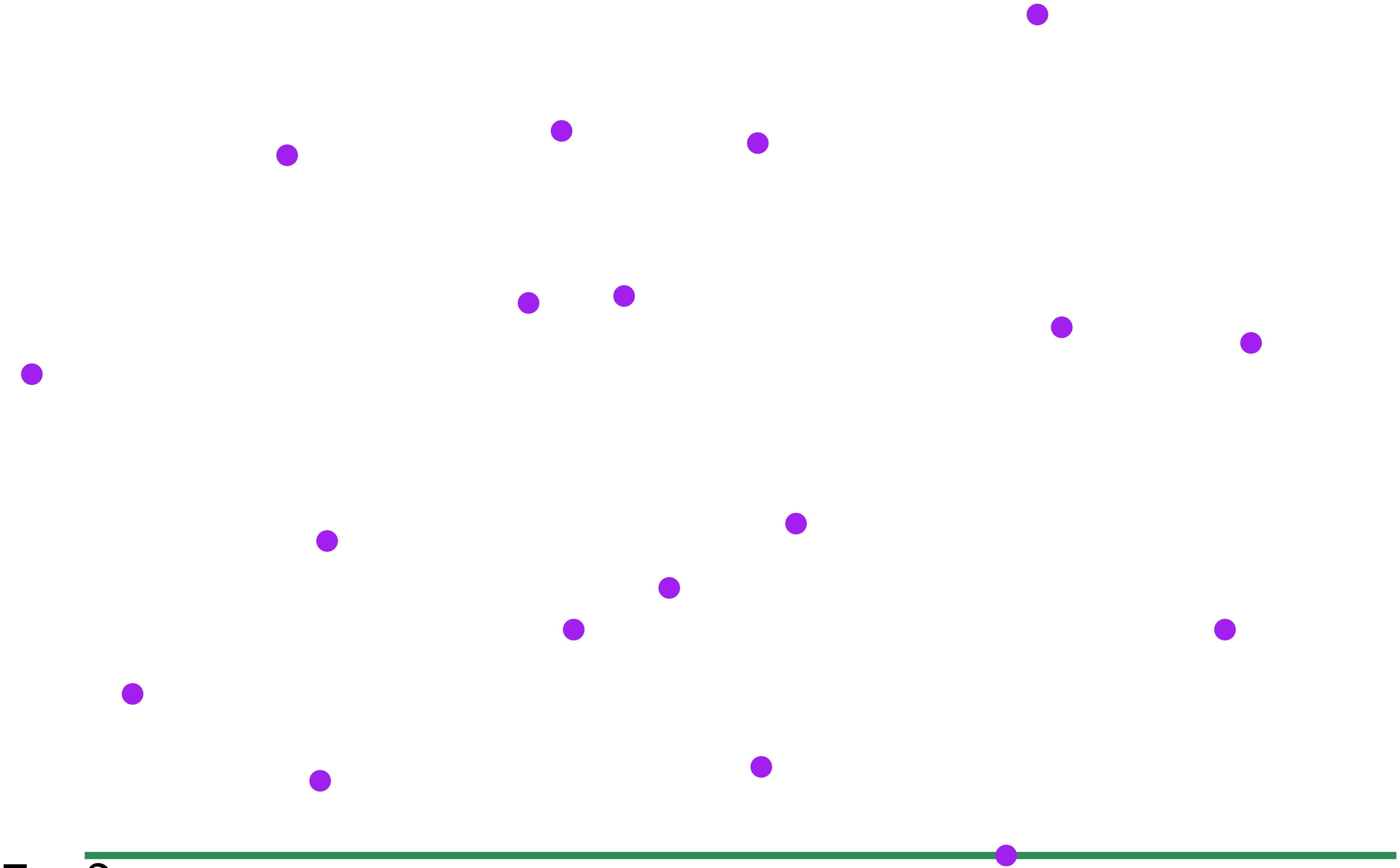
Convex hull

Jarvis algorithm



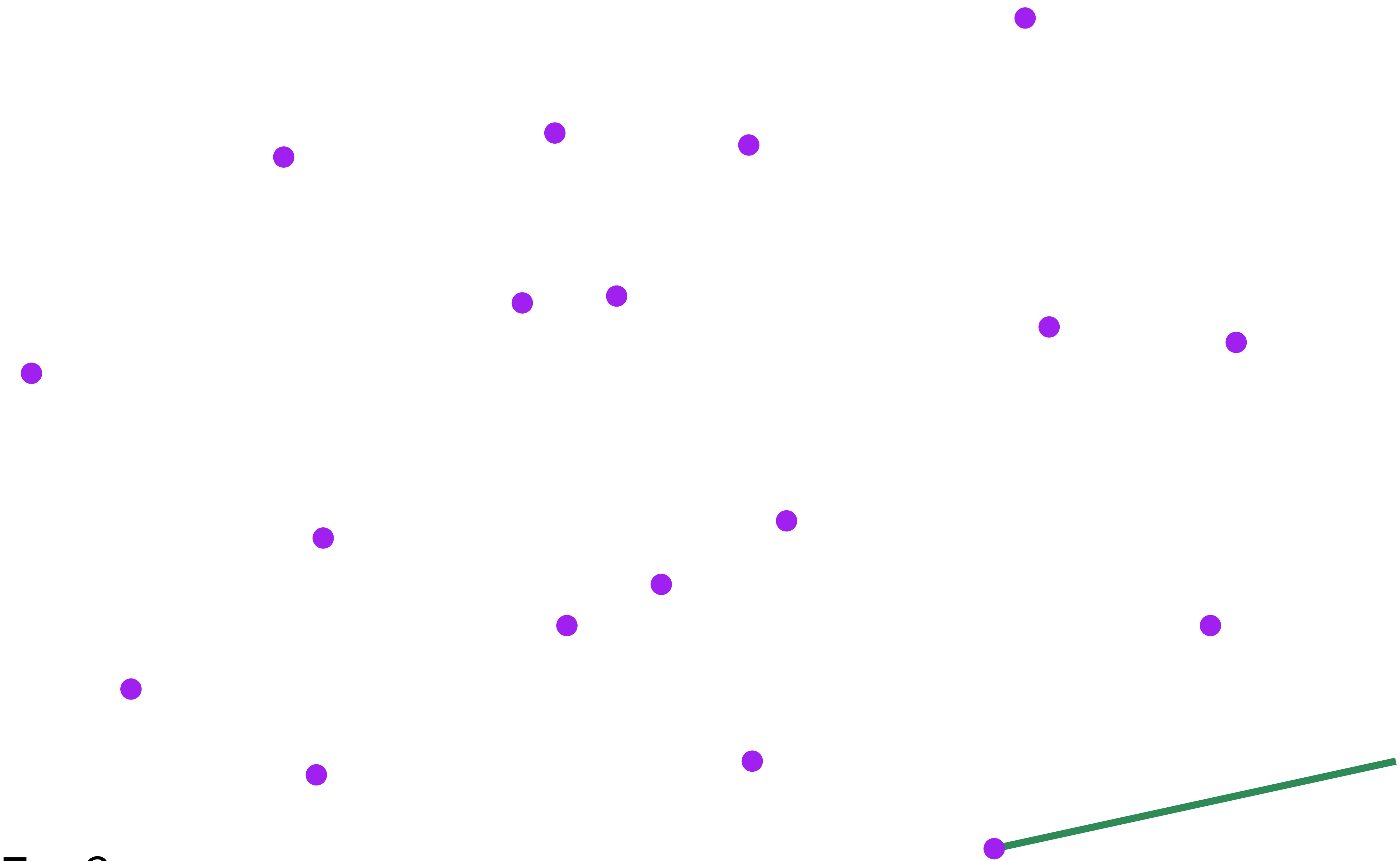
Convex hull

lowest point is extremal



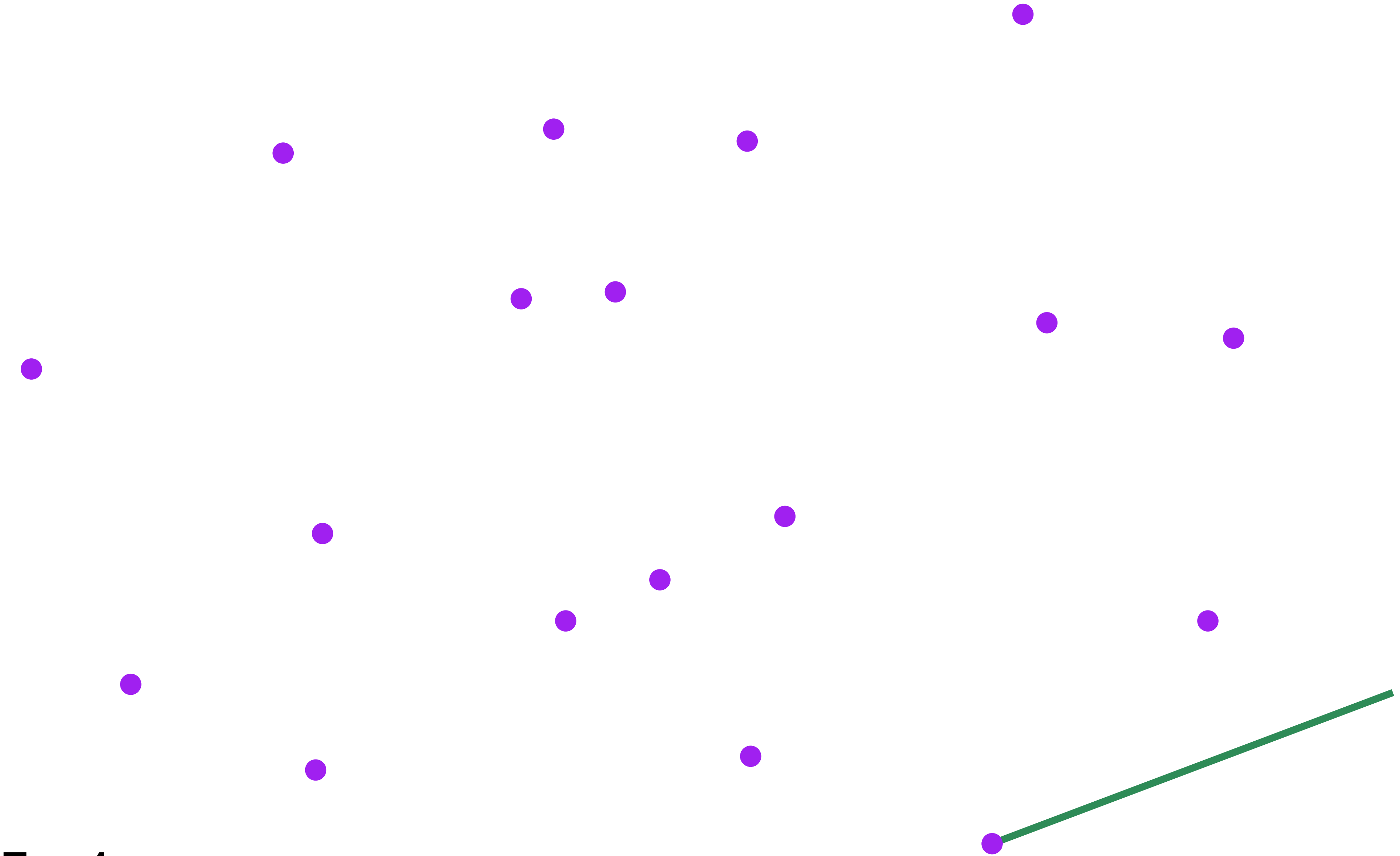
Convex hull

rotate



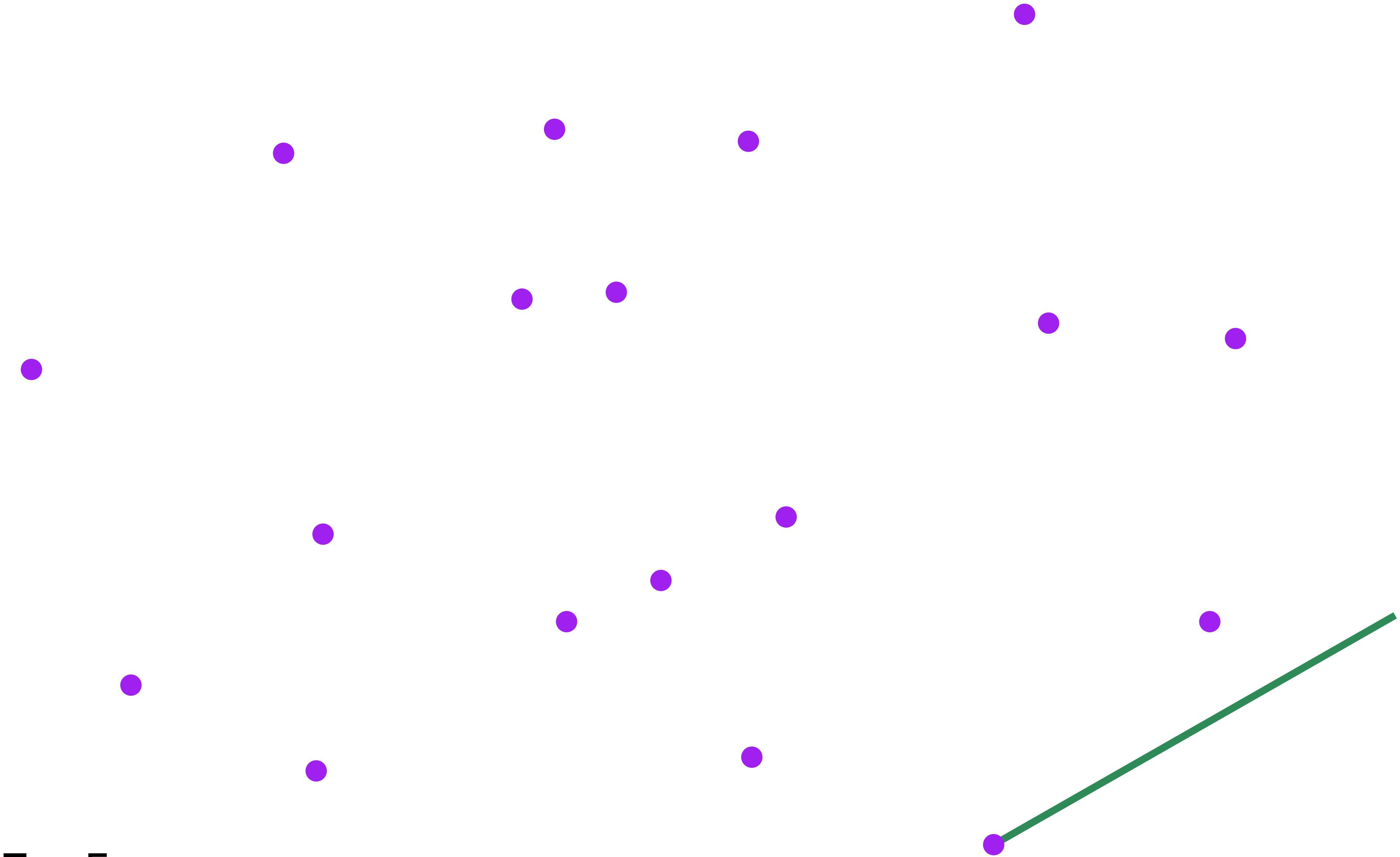
Convex hull

rotate



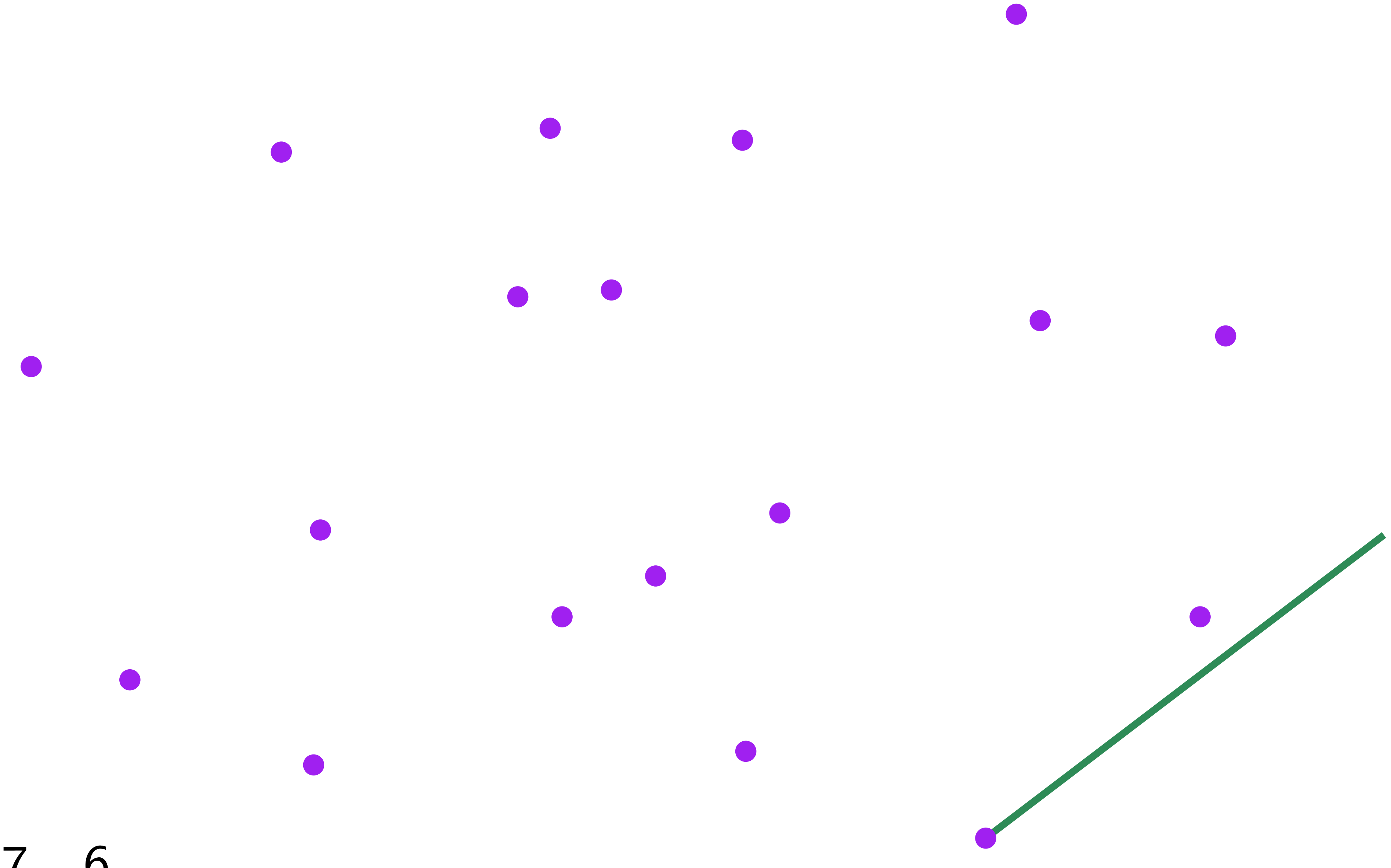
Convex hull

rotate



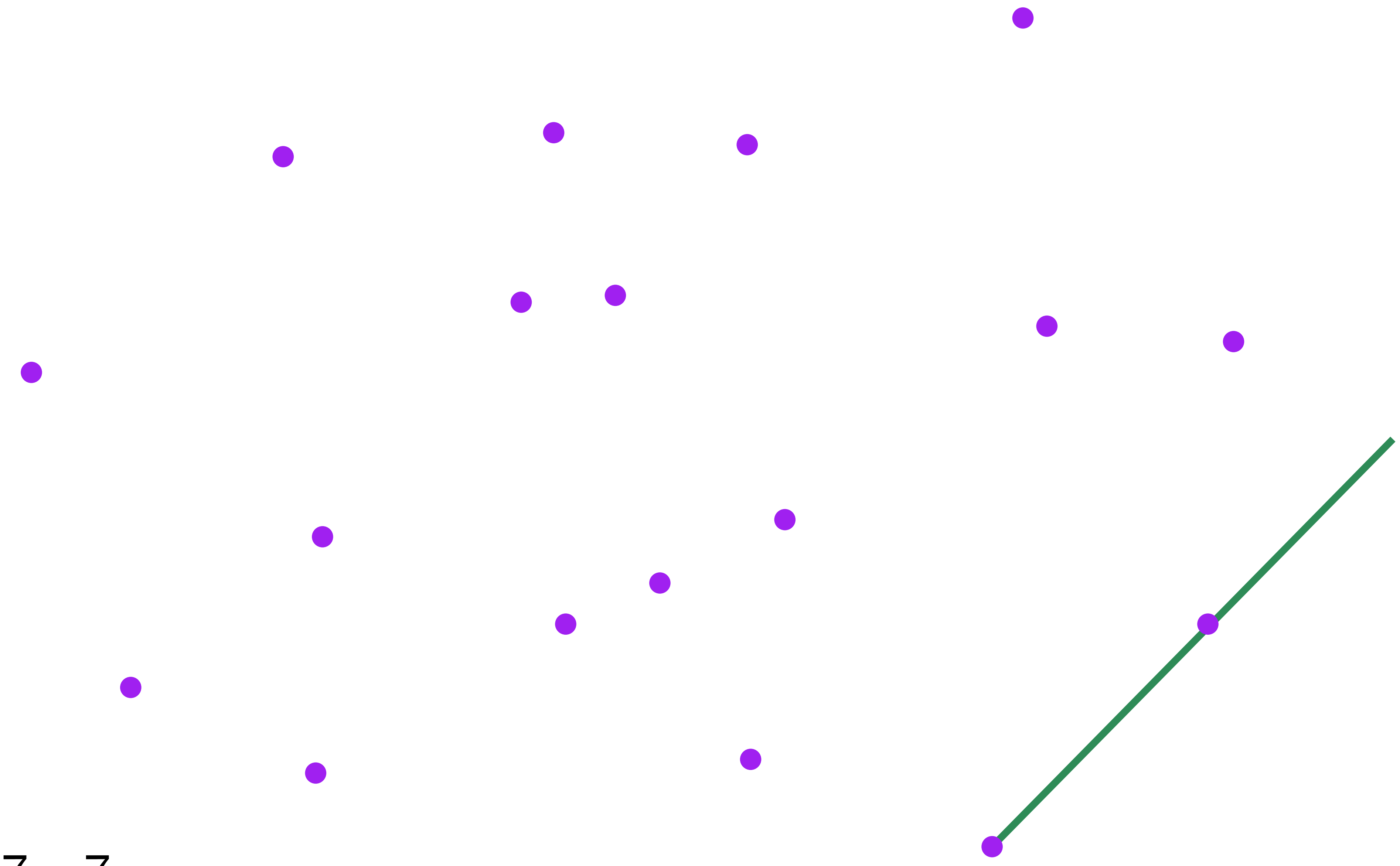
Convex hull

rotate



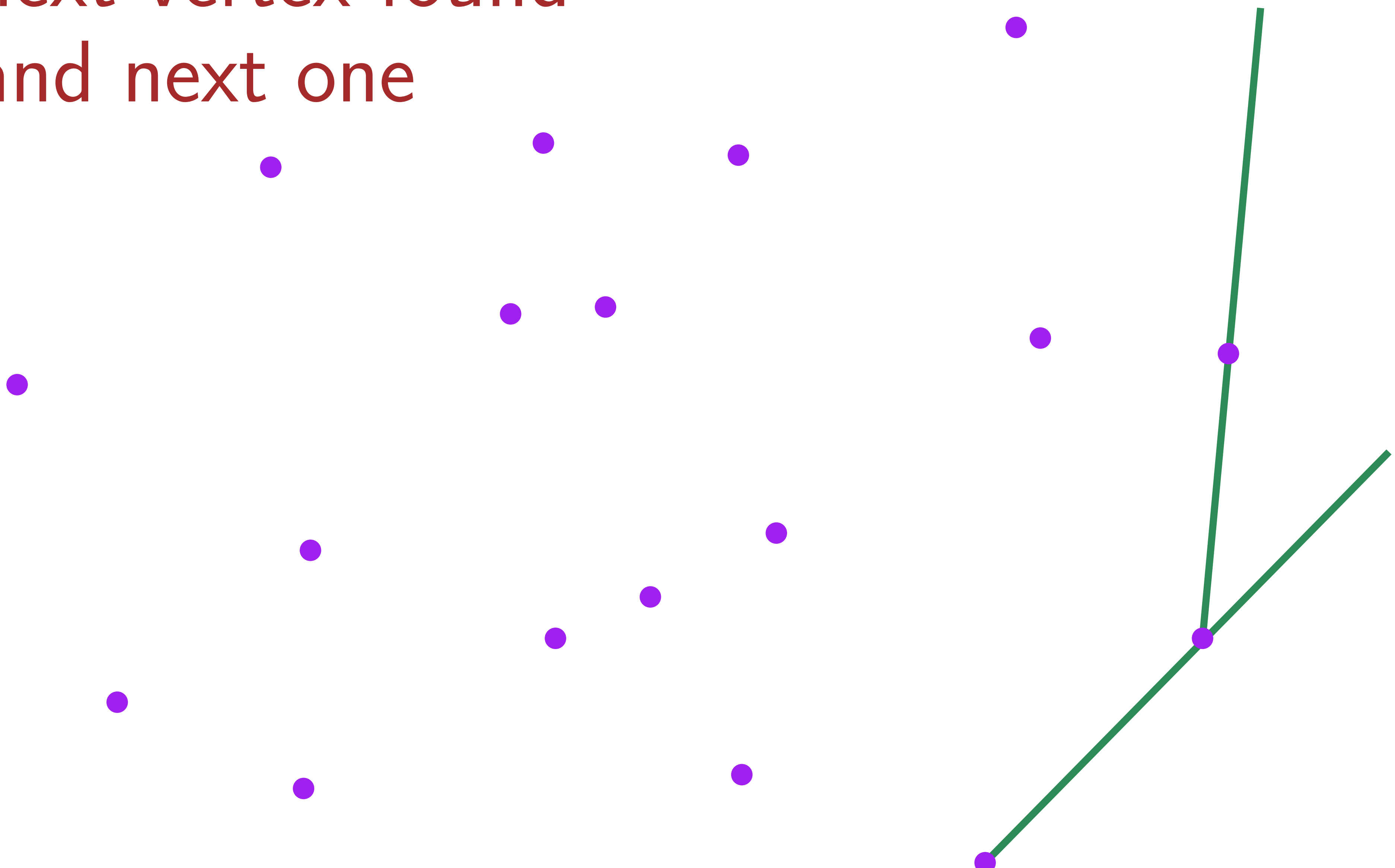
Convex hull

next vertex found



Convex hull

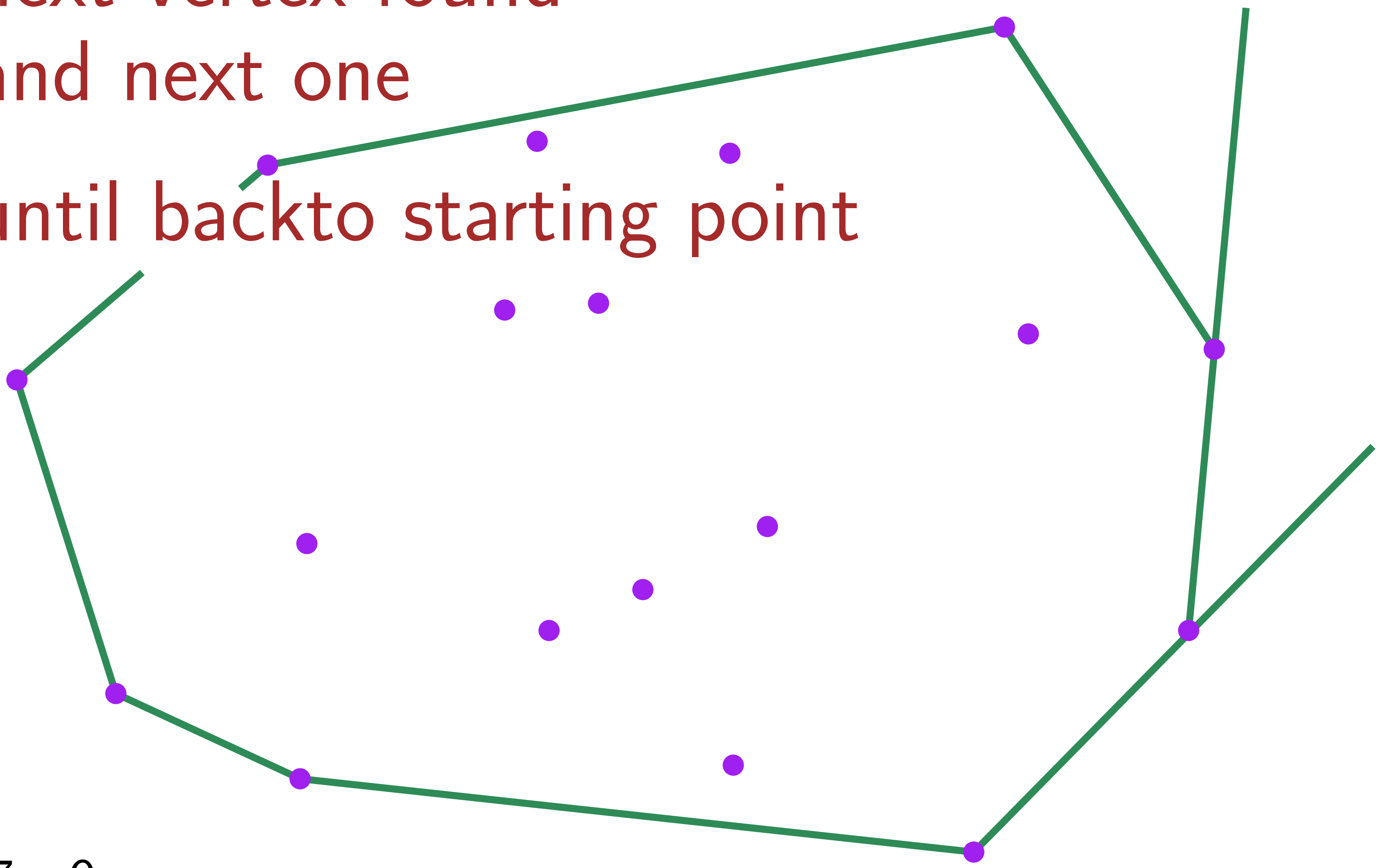
next vertex found
and next one



Convex hull

next vertex found
and next one

until backto starting point



Convex hull

Input : point set S

u = lowest point in S; $min = \infty$

For each $w \in S \setminus \{u\}$

  if $angle(ux, uw) < min$

    then $min = angle(ux, uw)$; $v = w$;

$u.next = v$;

Do

  $S = S \setminus \{v\}$

  For each $w \in S$

    $min = \infty$

    if $angle(v.pred\ v, vw) < min$

     then $min = angle(v.pred\ v, vw)$; $v.next = w$;

  $v = v.next$;

While $v \neq u$

Convex hull

Complexity ?

Jarvis algorithm

Input : point set S

$u =$ lowest point in S; $min = \infty$

For each $w \in S \setminus \{u\}$

  if $angle(ux, uw) < min$

    then $min = angle(ux, uw)$; $v = w$;

$u.next = v$;

Do

  $S = S \setminus \{v\}$

  For each $w \in S$

    $min = \infty$

    if $angle(v.pred\ v, vw) < min$

     then $min = angle(v.pred\ v, vw)$; $v.next = w$;

  $v = v.next$;

While $v \neq u$

Convex hull

Complexity ?

Jarvis algorithm

Input : point set S

$u =$ lowest point in S; $min = \infty$

For each $w \in S \setminus \{u\}$

if $angle(ux, uw) < min$

then $min = angle(ux, uw)$; $v = w$;

$O(n)$

$u.next = v$;

Do

$S = S \setminus \{v\}$

For each $w \in S$

$min = \infty$

if $angle(v.pred\ v, vw) < min$

then $min = angle(v.pred\ v, vw)$; $v.next = w$;

$v = v.next$;

While $v \neq u$

Convex hull

Jarvis algorithm

Complexity ?

Input : point set S

$u =$ lowest point in S; $min = \infty$

For each $w \in S \setminus \{u\}$

if $angle(ux, uw) < min$

then $min = angle(ux, uw)$; $v = w$;

$u.next = v$;

Do

$S = S \setminus \{v\}$

For each $w \in S$

$min = \infty$

if $angle(v.pred\ v, vw) < min$

then $min = angle(v.pred\ v, vw)$; $v.next = w$;

$v = v.next$;

While $v \neq u$

$O(n)$

Convex hull

Complexity ?

Jarvis algorithm

Input : point set S

$u =$ lowest point in S; $min = \infty$

For each $w \in S \setminus \{u\}$

if $angle(ux, uw) < min$

then $min = angle(ux, uw)$; $v = w$;

$u.next = v$;

Do

$S = S \setminus \{v\}$

For each $w \in S$

$min = \infty$

if $angle(v.pred\ v, vw) < min$

then $min = angle(v.pred\ v, vw)$; $v.next = w$;

$v = v.next$;

While $v \neq u$

Convex hull

Complexity ?

Jarvis algorithm

Input : point set S

u = lowest point in S; $min = \infty$

For each $w \in S \setminus \{u\}$

if $angle(ux, uw) < min$

then $min = angle(ux, uw)$; $v = w$;

$u.next = v$;

Do

$S = S \setminus \{v\}$

For each $w \in S$

$min = \infty$

if $angle(v.pred\ v, vw) < min$

then $min = angle(v.pred\ v, vw)$; $v.next = w$;

$v = v.next$;

While $v \neq u$

$O(n)$

Convex hull

Jarvis algorithm

Complexity ?

$O(n^2)$

Input : point set S

$u =$ lowest point in S; $min = \infty$

For each $w \in S \setminus \{u\}$

if $angle(ux, uw) < min$

then $min = angle(ux, uw)$; $v = w$;

$u.next = v$;

Do

$S = S \setminus \{v\}$

For each $w \in S$

$min = \infty$

if $angle(v.pred\ v, vw) < min$

then $min = angle(v.pred\ v, vw)$; $v.next = w$;

$v = v.next$;

While $v \neq u$

Convex hull

Complexity ?

Jarvis algorithm

Input : point set S

$u =$ lowest point in S; $min = \infty$

For each $w \in S \setminus \{u\}$

if $angle(ux, uw) < min$

then $min = angle(ux, uw)$; $v = w$;

$u.next = v$;

Do

$S = S \setminus \{v\}$

For each $w \in S$

$min = \infty$

if $angle(v.pred\ v, vw) < min$

then $min = angle(v.pred\ v, vw)$; $v.next = w$;

$v = v.next$;

While $v \neq u$

$O(n^2)$

$O(nh)$

Convex hull

Orientation predicate

Input : point set S

$u =$ lowest point in S; $min = \infty$

For each $w \in S \setminus \{u\}$

if $angle(ux, uw) < min$

then $min = angle(ux, uw)$; $v = w$;

$u.next = v$;

Do

$S = S \setminus \{v\}$

For each $w \in S$

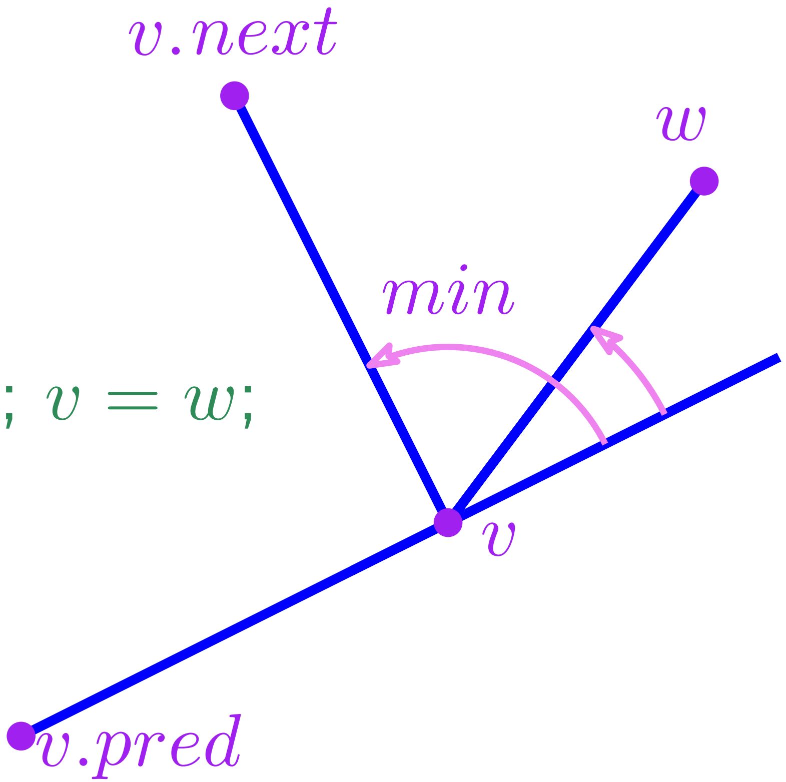
$min = \infty$

if $angle(v.pred\ v, vw) < min$

then $min = angle(v.pred\ v, vw)$; $v.next = w$;

$v = v.next$;

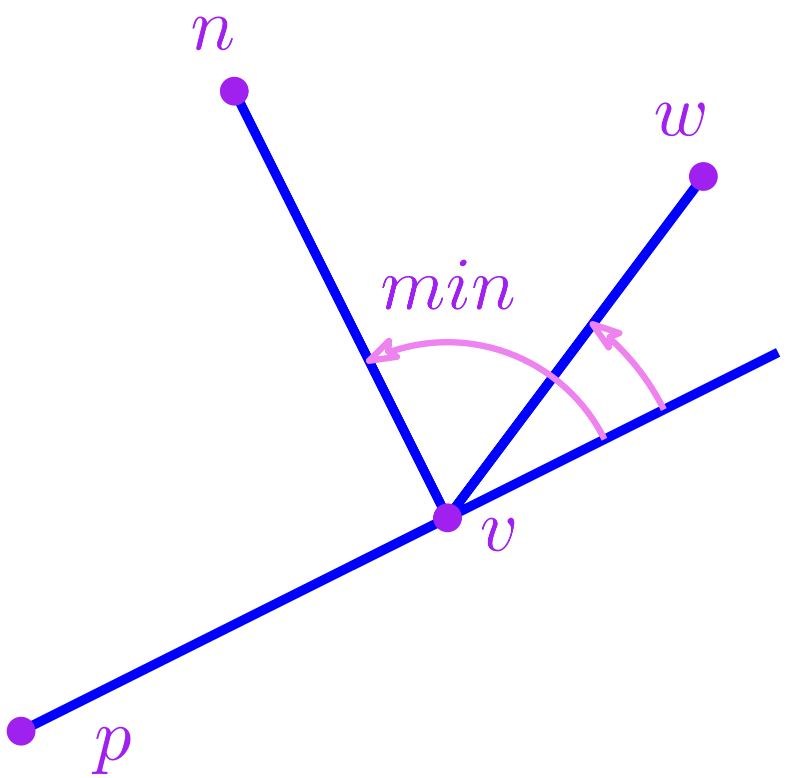
While $v \neq u$



Convex hull

Orientation predicate

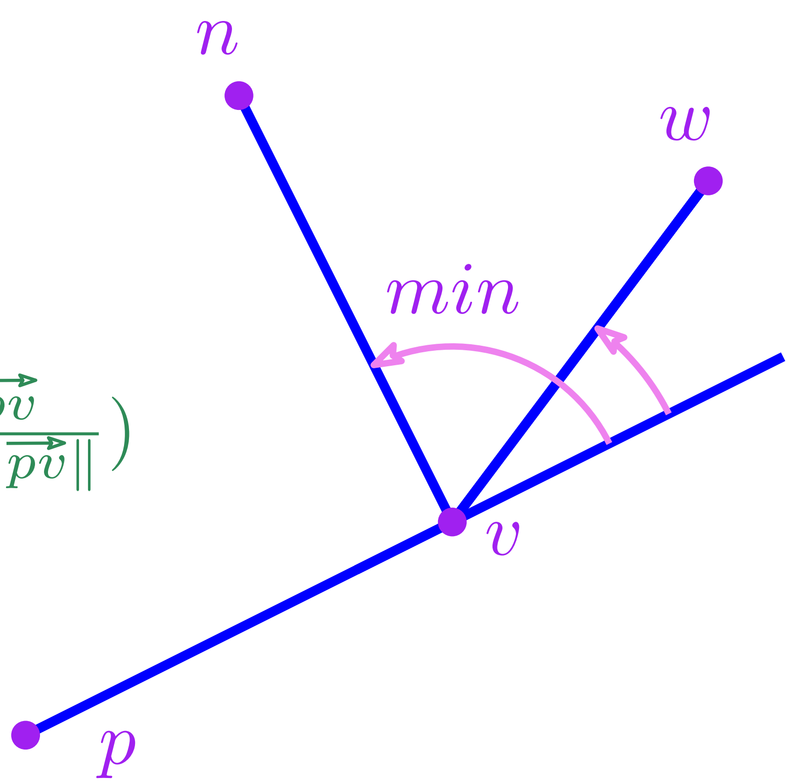
if $angle(pv, vw) < min$



Convex hull

Orientation predicate

if $angle(pv, vw) < min$



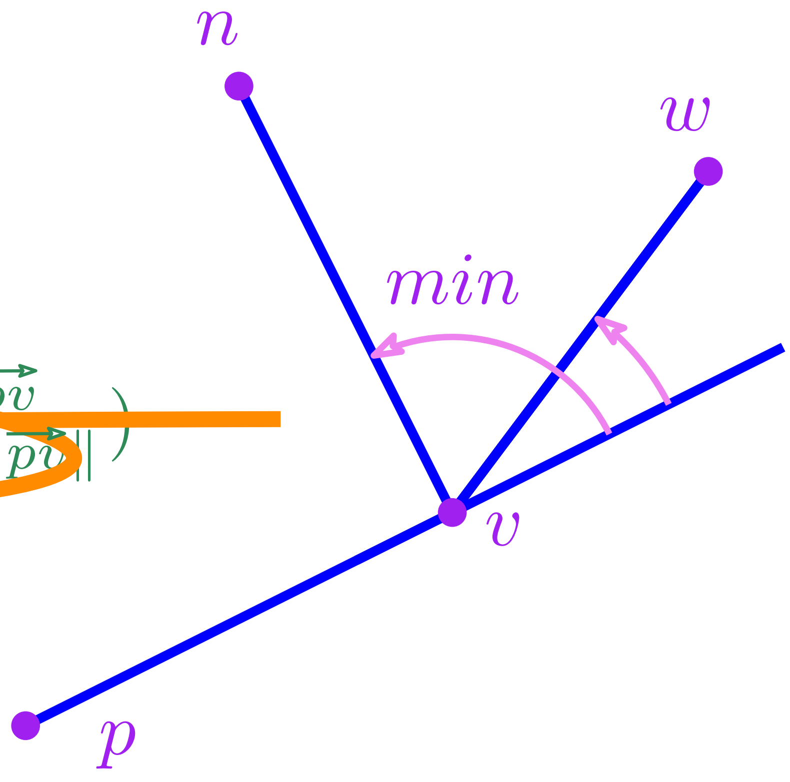
$$angle(pv, vw) = \arccos\left(\frac{\overrightarrow{vw} \cdot \overrightarrow{pv}}{\|\overrightarrow{vw}\| \cdot \|\overrightarrow{pv}\|}\right)$$

Convex hull

Orientation predicate

if $angle(pv, vw) < min$

$$angle(pv, vw) = \arccos\left(\frac{\overrightarrow{vw} \cdot \overrightarrow{pv}}{\|\overrightarrow{vw}\| \cdot \|\overrightarrow{pv}\|}\right)$$



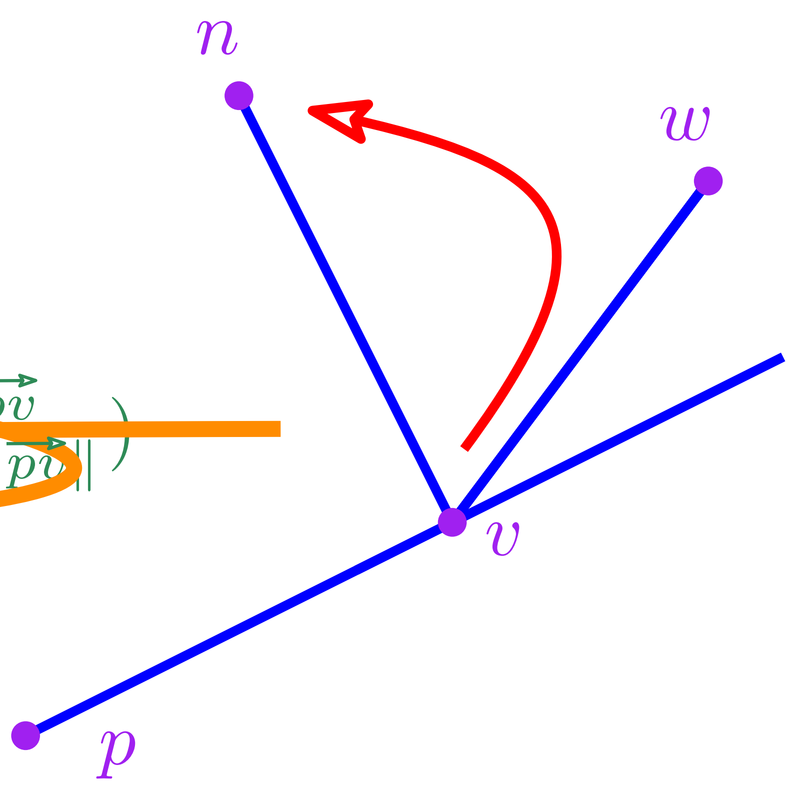
Convex hull

Orientation predicate

if $angle(pv, vw) < min$

$$angle(pv, vw) = \arccos\left(\frac{\vec{vw} \cdot \vec{pv}}{\|\vec{vw}\| \cdot \|\vec{pv}\|}\right)$$

if vwn turn left

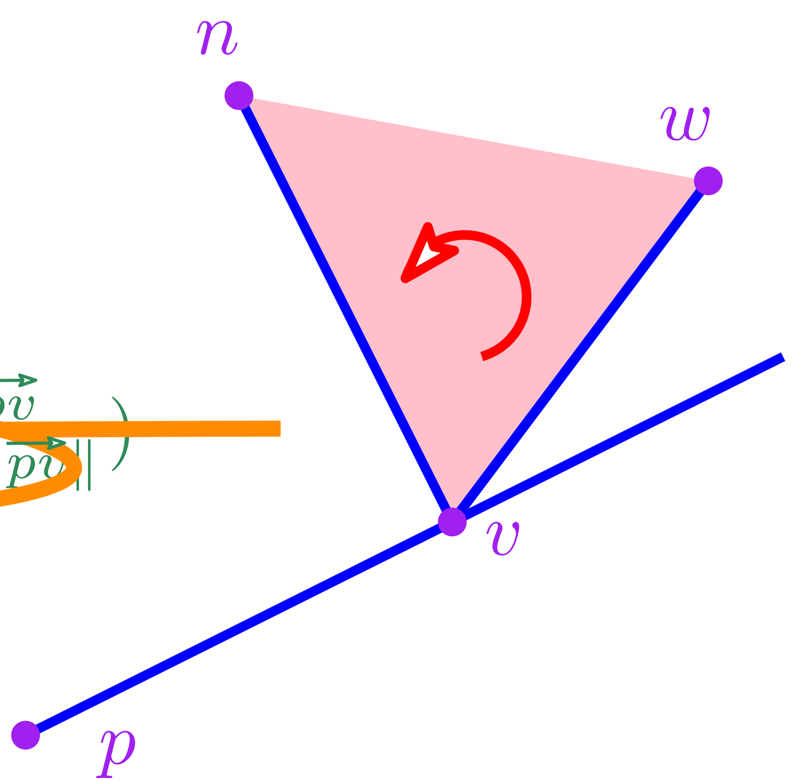


Convex hull

Orientation predicate

if $angle(pv, vw) < min$

$angle(pv, vw) = \arccos(\frac{\overrightarrow{vw} \cdot \overrightarrow{pv}}{\|\overrightarrow{vw}\| \cdot \|\overrightarrow{pv}\|})$



if vwn turn left

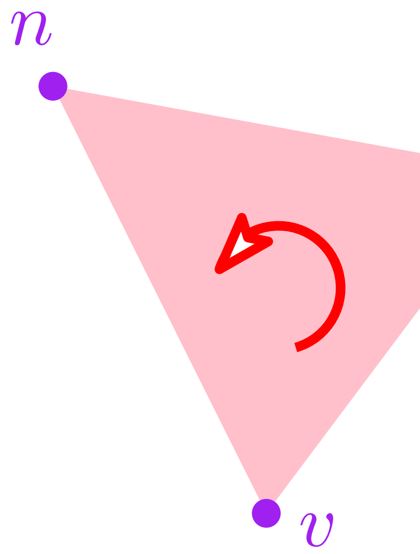
if triangle vwn counterclockwise (ccw)

if triangle vwn positively oriented

Convex hull

$vwn + ?$

Orientation predicate

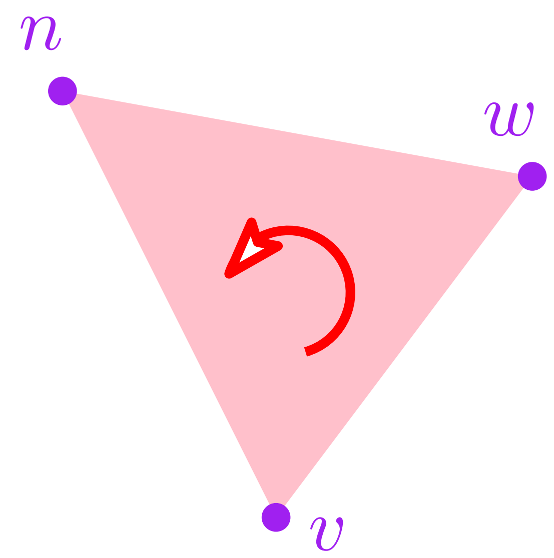


Convex hull

Orientation predicate

$vwn + ?$

$$\begin{vmatrix} x_w - x_v & x_n - x_v \\ y_w - y_v & y_n - y_v \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 \\ x_v & x_w & x_n \\ y_v & y_w & y_n \end{vmatrix} > 0$$

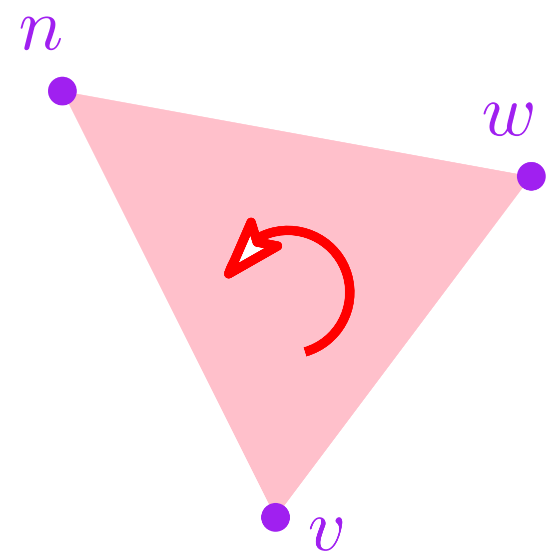


Convex hull

Orientation predicate

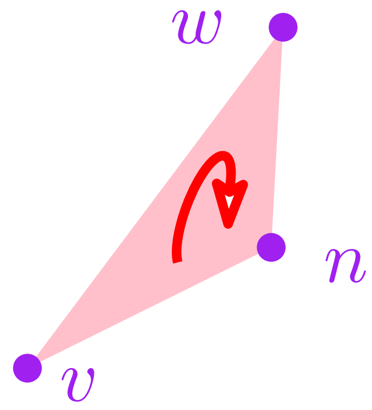
vwn + ?

$$\begin{vmatrix} x_w - x_v & x_n - x_v \\ y_w - y_v & y_n - y_v \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 \\ x_v & x_w & x_n \\ y_v & y_w & y_n \end{vmatrix} > 0$$



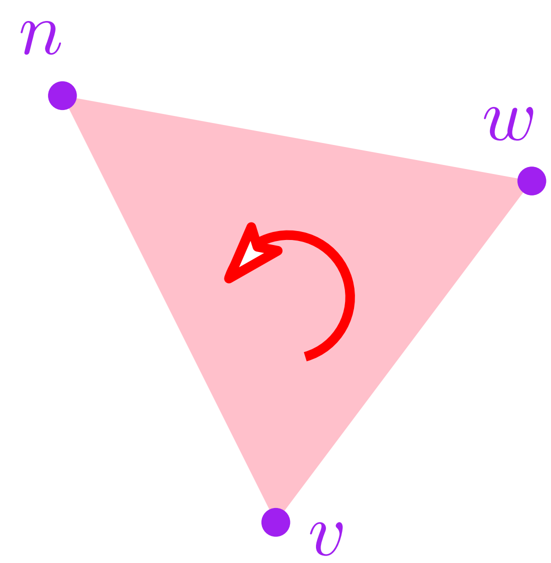
vwn - ?

$$\begin{vmatrix} 1 & 1 & 1 \\ x_v & x_w & x_n \\ y_v & y_w & y_n \end{vmatrix} < 0$$



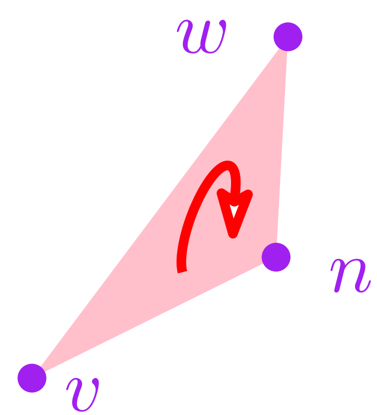
Convex hull

Orientation predicate



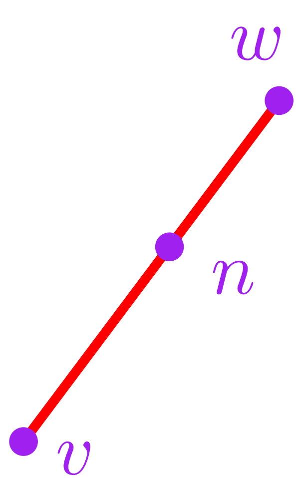
vwn + ?

$$\begin{vmatrix} x_w - x_v & x_n - x_v \\ y_w - y_v & y_n - y_v \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 \\ x_v & x_w & x_n \\ y_v & y_w & y_n \end{vmatrix} > 0$$



vwn - ?

$$\begin{vmatrix} 1 & 1 & 1 \\ x_v & x_w & x_n \\ y_v & y_w & y_n \end{vmatrix} < 0$$



vwn 0 ?

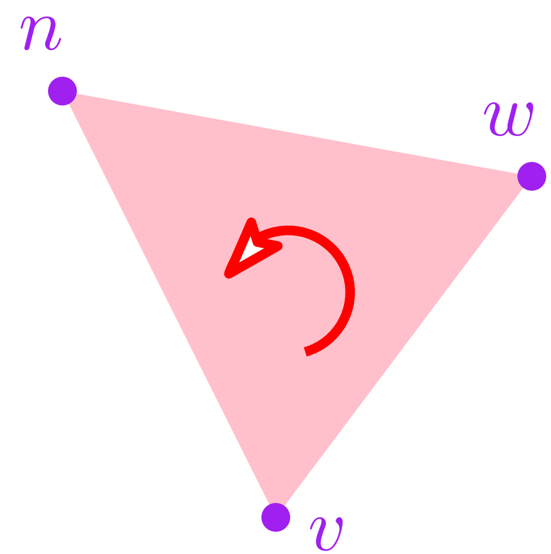
$$\begin{vmatrix} 1 & 1 & 1 \\ x_v & x_w & x_n \\ y_v & y_w & y_n \end{vmatrix} = 0$$

Convex hull

Orientation predicate

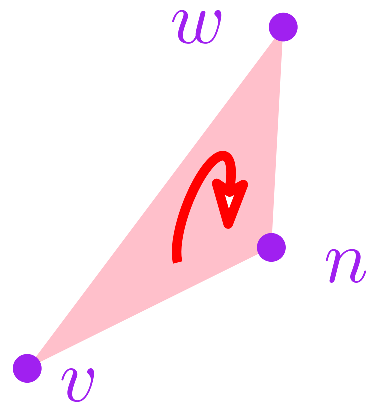
vwn + ?

$$\begin{vmatrix} x_w - x_v & x_n - x_v \\ y_w - y_v & y_n - y_v \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 \\ x_v & x_w & x_n \\ y_v & y_w & y_n \end{vmatrix} > 0$$



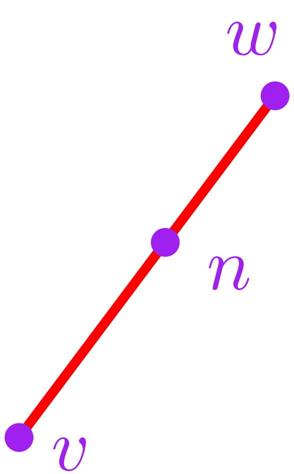
vwn - ?

$$\begin{vmatrix} 1 & 1 & 1 \\ x_v & x_w & x_n \\ y_v & y_w & y_n \end{vmatrix} < 0$$



vwn 0 ?

$$\begin{vmatrix} 1 & 1 & 1 \\ x_v & x_w & x_n \\ y_v & y_w & y_n \end{vmatrix} = 0$$



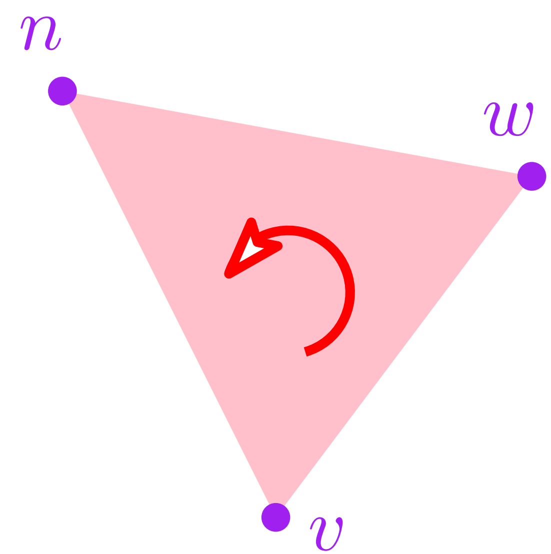
degenerate case

Convex hull

Orientation predicate

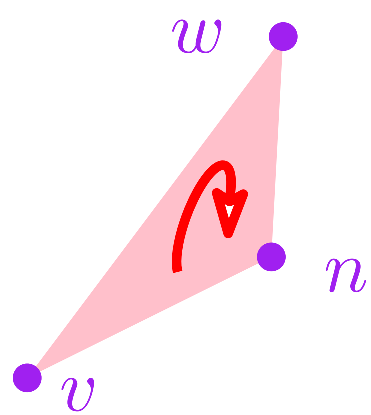
vwn + ?

$$\begin{vmatrix} x_w - x_v & x_n - x_v \\ y_w - y_v & y_n - y_v \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 \\ x_v & x_w & x_n \\ y_v & y_w & y_n \end{vmatrix} > 0$$



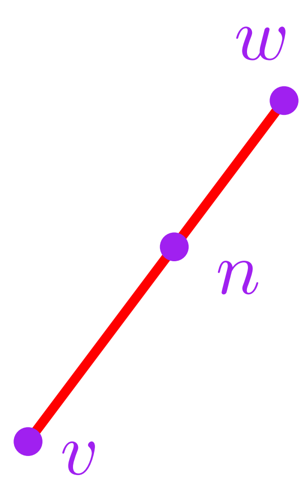
vwn - ?

$$\begin{vmatrix} 1 & 1 & 1 \\ x_v & x_w & x_n \\ y_v & y_w & y_n \end{vmatrix} < 0$$



vwn 0 ?

$$\begin{vmatrix} 1 & 1 & 1 \\ x_v & x_w & x_n \\ y_v & y_w & y_n \end{vmatrix} = 0$$



rounding errors

?

Convex hull

Input : point set S
$u = v =$ lowest point in S;

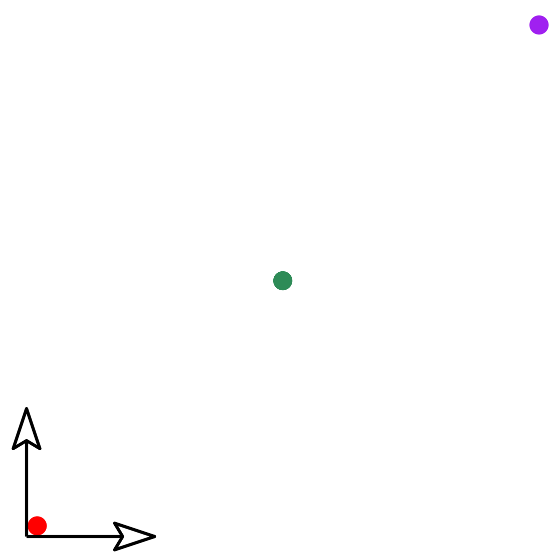
Do
    $n =$ first in S;
    For each $w \in S$
        if vwn CCW
            then $n = w$;
    $v.next = n$; $v = n$;
    $S = S \setminus \{v\}$
While $v \neq u$

Un premier contact avec les problèmes de robustesse

Convex hull

Orientation predicate

Rounding errors possible



$p = (\frac{1}{2} + x.u, \frac{1}{2} + y.u)$

$0 \leq x, y \leq 256,\ u = 2^{-53}$

$q = (12, 12)$

$r = (24, 24)$

Teaser robustness lecture

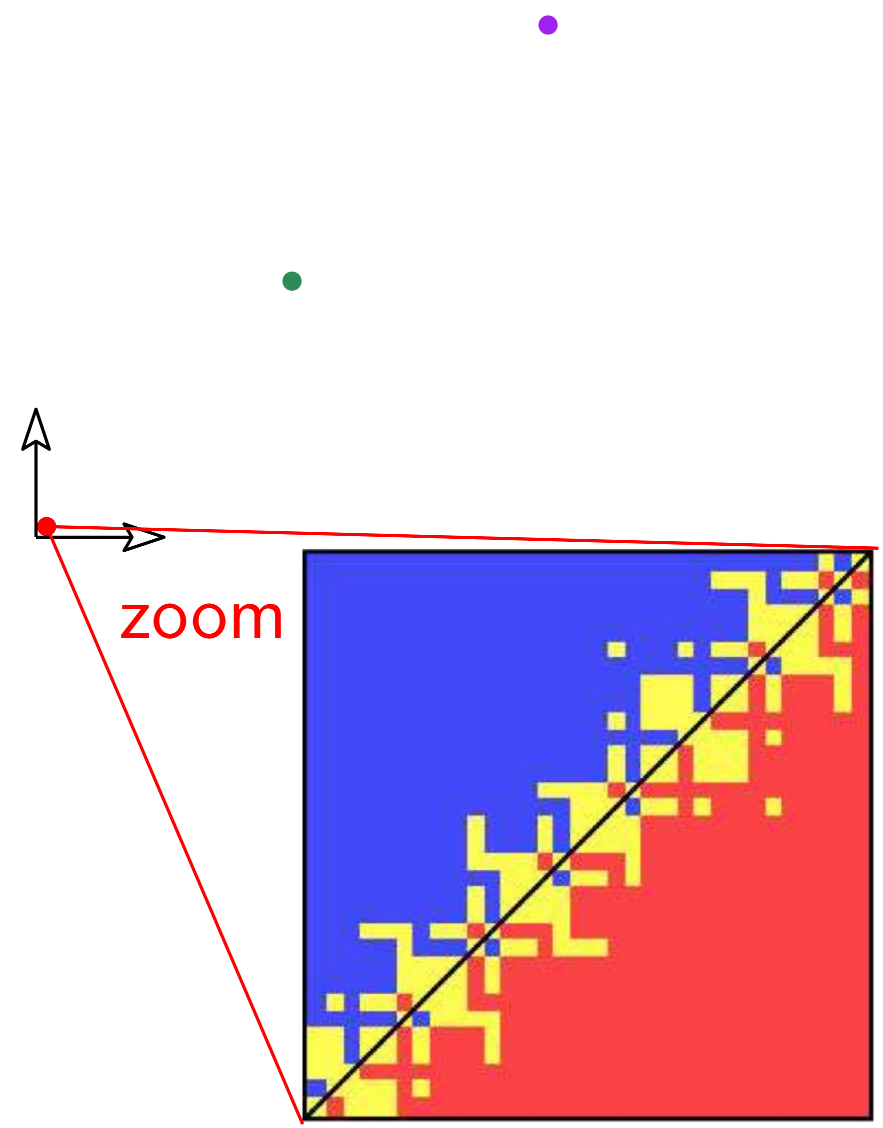
$orientation(p, q, r)$

evaluated with double

Convex hull

Orientation predicate

Rounding errors possible



$p = (\frac{1}{2} + x.u, \frac{1}{2} + y.u)$

$0 \leq x, y \leq 256, \; u = 2^{-53}$

$q = (12, 12)$

$r = (24, 24)$

Teaser robustness lecture

$orientation(p, q, r)$

evaluated with double

≤ 0

0

≥ 0

Convex hull

Teaser robustness lecture

Buggy degenerate example
(single precision)

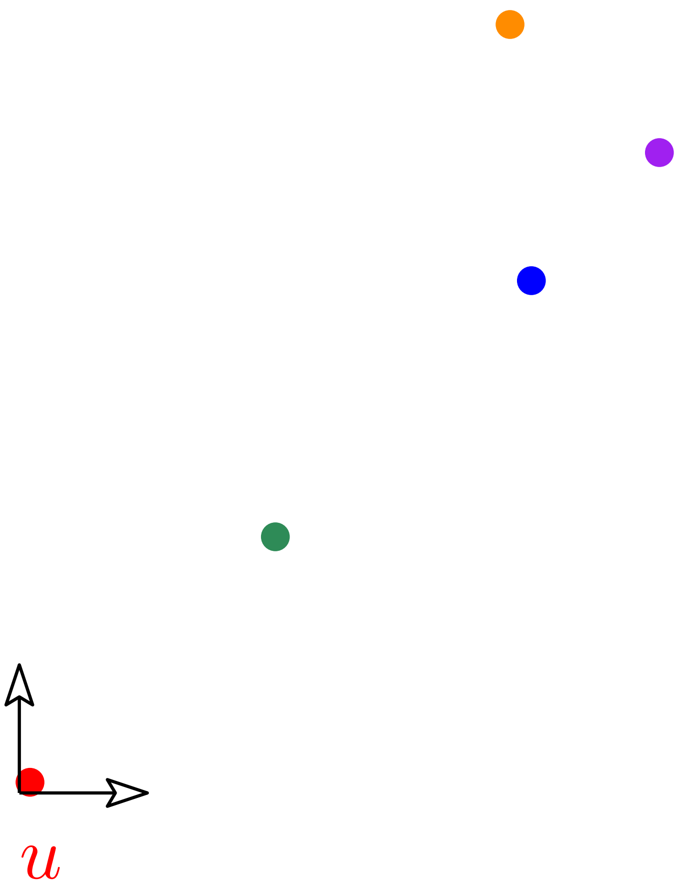
$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$



Convex hull

Teaser robustness lecture

Buggy degenerate example (single precision)

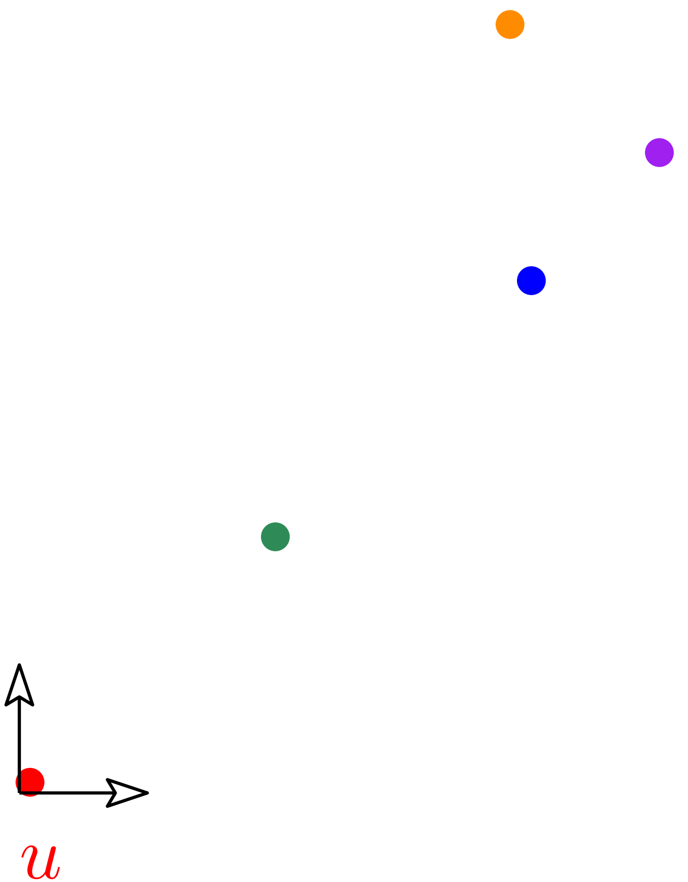
$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$



Input : point set S

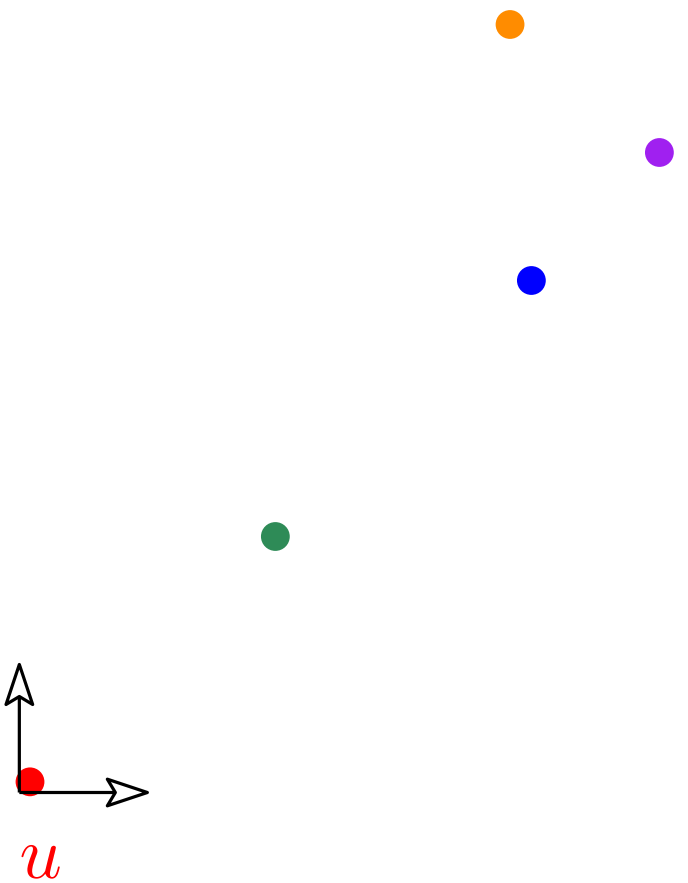
$u = v =$ lowest point in S;

Jarvis

Convex hull

Teaser robustness lecture

Buggy degenerate example (single precision)



$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$

Do
  $n =$ first in S;
  For each $w \in S$
    if vwn positive
     then $n = w$;
  $v.next = n$; $v = n$;
  $S = S \setminus \{v\}$
While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example

(single precision)

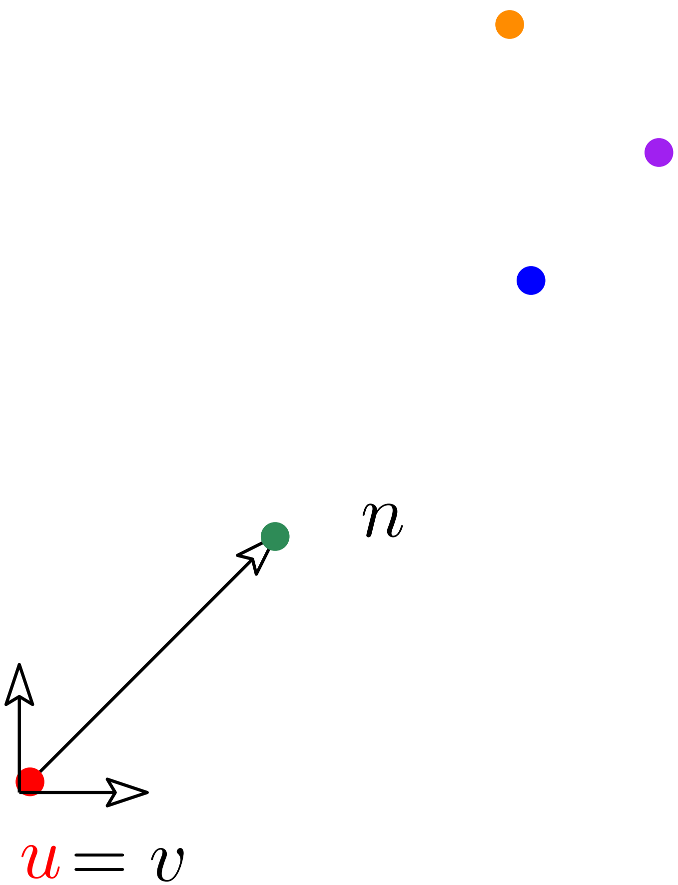
$$w_1 = (12, 12)$$

$$w_2 = (24, 24)$$

$$w_3 = (30, 30.000001)$$

$$w_4 = (23, 36)$$

$$w_5 = (0.5000029, 0.5000027)$$



Do

  $n =$ first in S;

  For each $w \in S$

    if vwn positive

     then $n = w$;

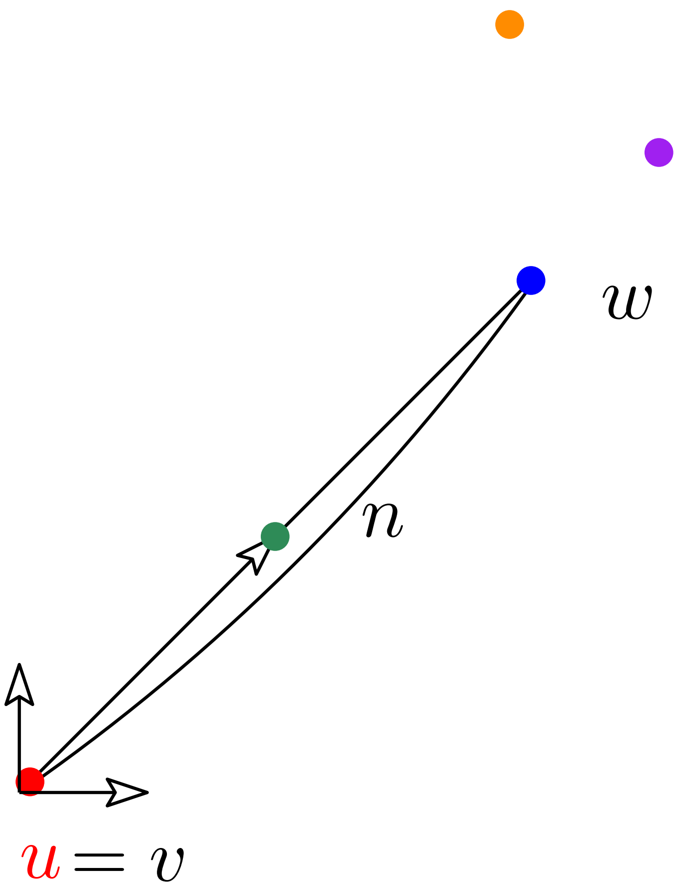
  $v.next = n$; $v = n$;

  $S = S \setminus \{v\}$

While $v \neq u$

Convex hull

Teaser robustness lecture



Buggy degenerate example (single precision)

$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$

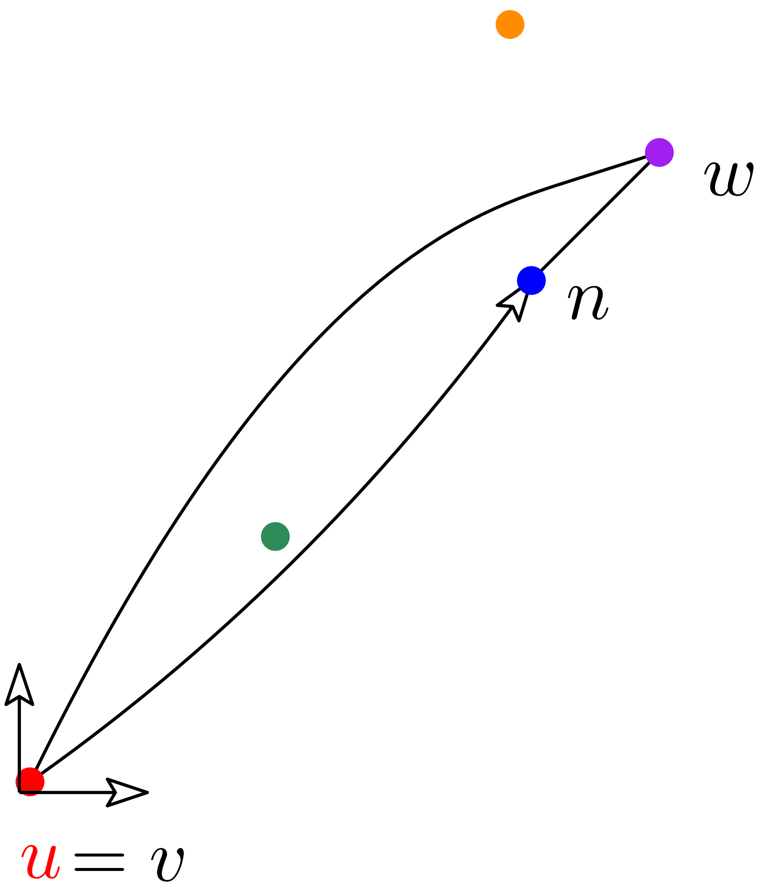
Do
- n = first in S;
- For each $w \in S$
  - if vwn positive
    - then $n = w$;
- $v.next = n$; $v = n$;
- $S = S \setminus \{v\}$

While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example (single precision)



$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

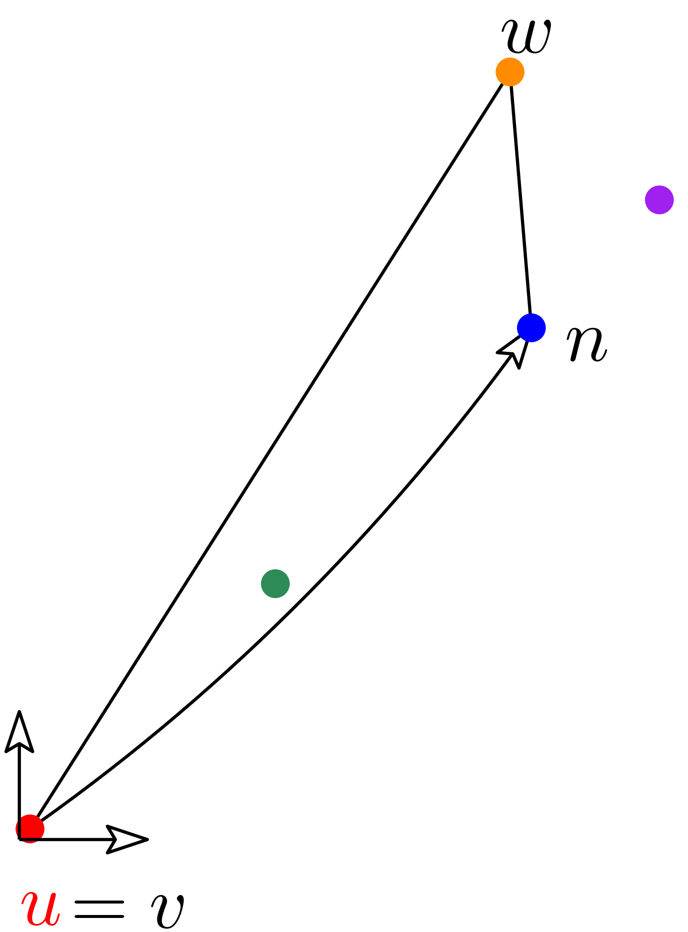
$w_5 = (0.5000029, 0.5000027)$

Do
- $n =$ first in S;
- For each $w \in S$
  - if vwn positive
    - then $n = w$;
- $v.next = n$; $v = n$;
- $S = S \setminus \{v\}$

While $v \neq u$

Convex hull

Teaser robustness lecture



Buggy degenerate example

(single precision)

$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$

Do

$n =$ first in S;

For each $w \in S$

if vwn positive

then $n = w$;

$v.next = n$; $v = n$;

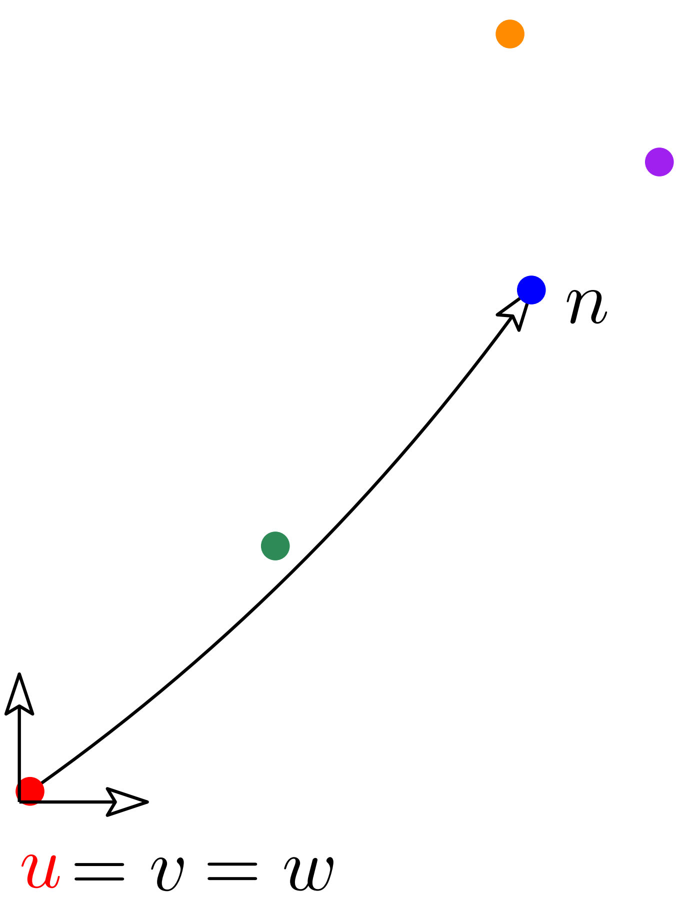
$S = S \setminus \{v\}$

While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example (single precision)



$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$

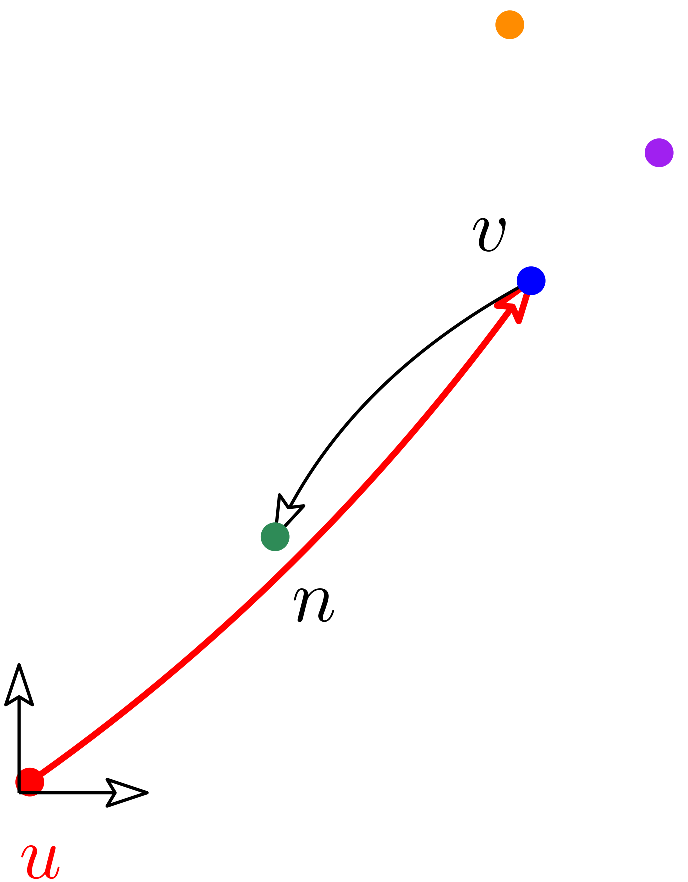
Do
- $n =$ first in S;
- For each $w \in S$
  - if vwn positive
    - then $n = w$;
- $v.next = n$; $v = n$;
- $S = S \setminus \{v\}$

While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example
(single precision)



$w_1 = (12, 12)$

~~$w_2 = (24, 24)$~~

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$

Do
  $n =$ first in S;
  For each $w \in S$
    if vwn positive
      then $n = w$;
  $v.next = n$; $v = n$;
  $S = S \setminus \{v\}$
While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example
(single precision)

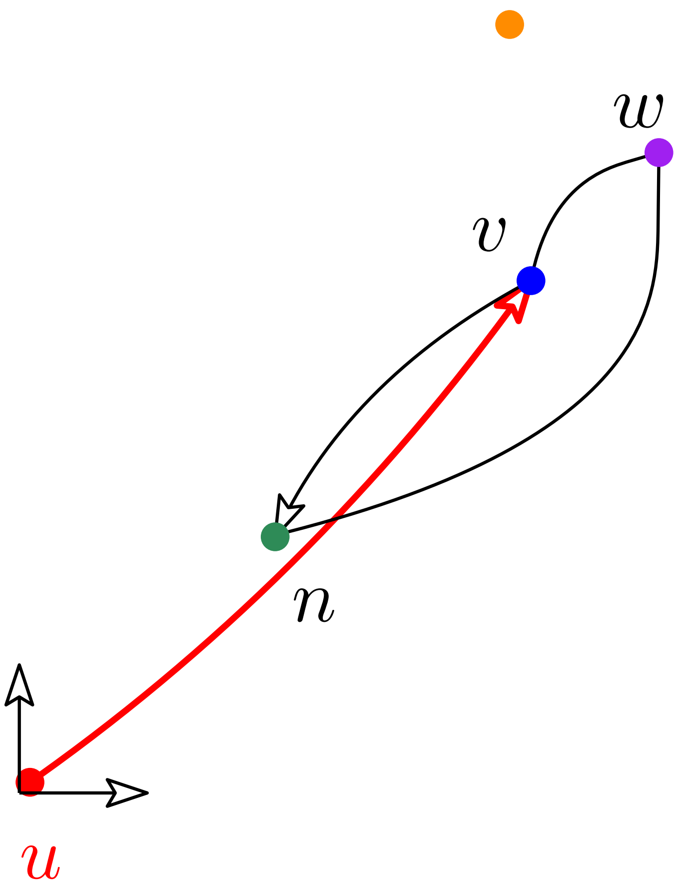
$w_1 = (12, 12)$

~~$w_2 = (24, 24)$~~

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$



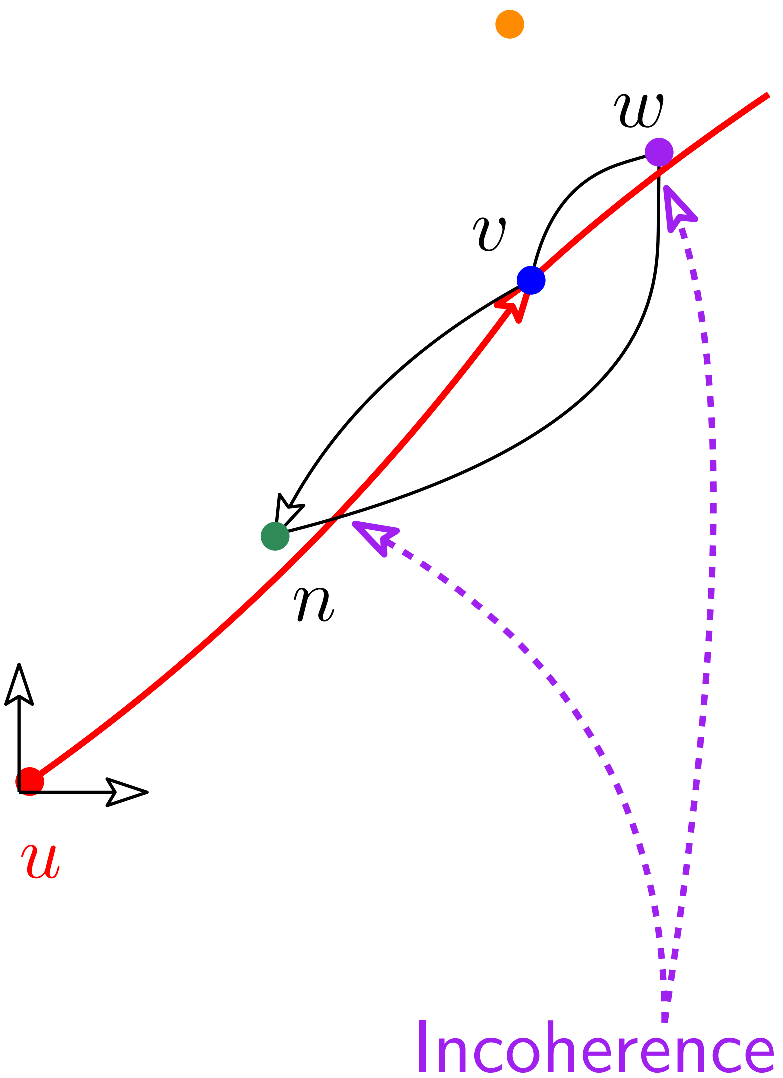
Do
 $n =$ first in S;
 For each $w \in S$
  if vwn positive
   then $n = w$;
 $v.next = n$; $v = n$;
 $S = S \setminus \{v\}$
While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example

(single precision)



$w_1 = (12, 12)$

~~$w_2 = (24, 24)$~~

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$

Do

$n =$ first in S;

For each $w \in S$

if vwn positive

then $n = w$;

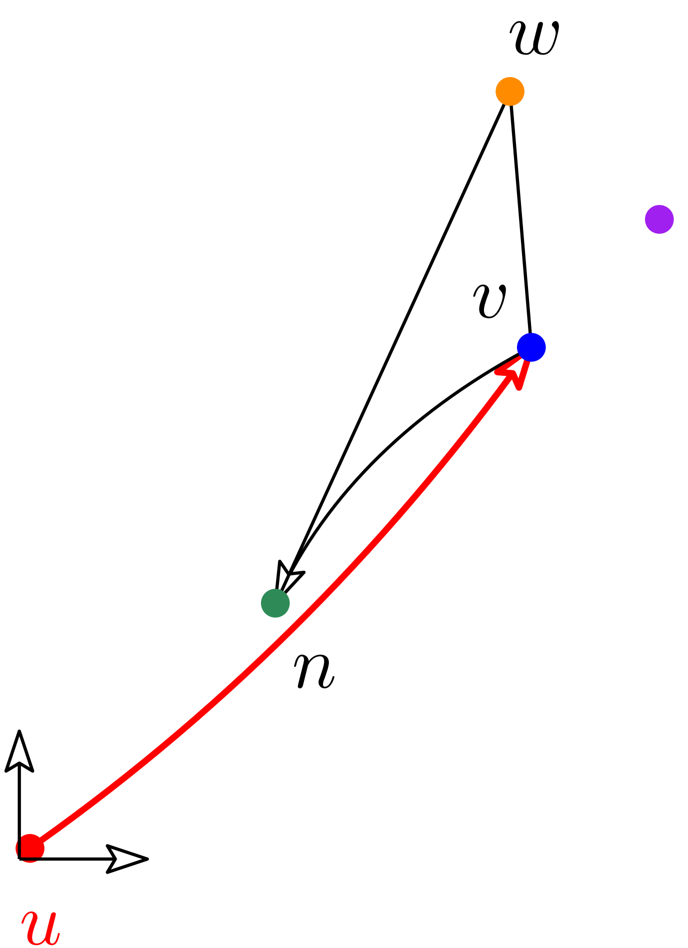
$v.next = n$; $v = n$;

$S = S \setminus \{v\}$

While $v \neq u$

Convex hull

Teaser robustness lecture



Buggy degenerate example
(single precision)

$w_1 = (12, 12)$

~~$w_2 = (24, 24)$~~

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$

Do
  $n =$ first in S;
  For each $w \in S$
    if vwn positive
     then $n = w$;
  $v.next = n$; $v = n$;
  $S = S \setminus \{v\}$
While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example
(single precision)

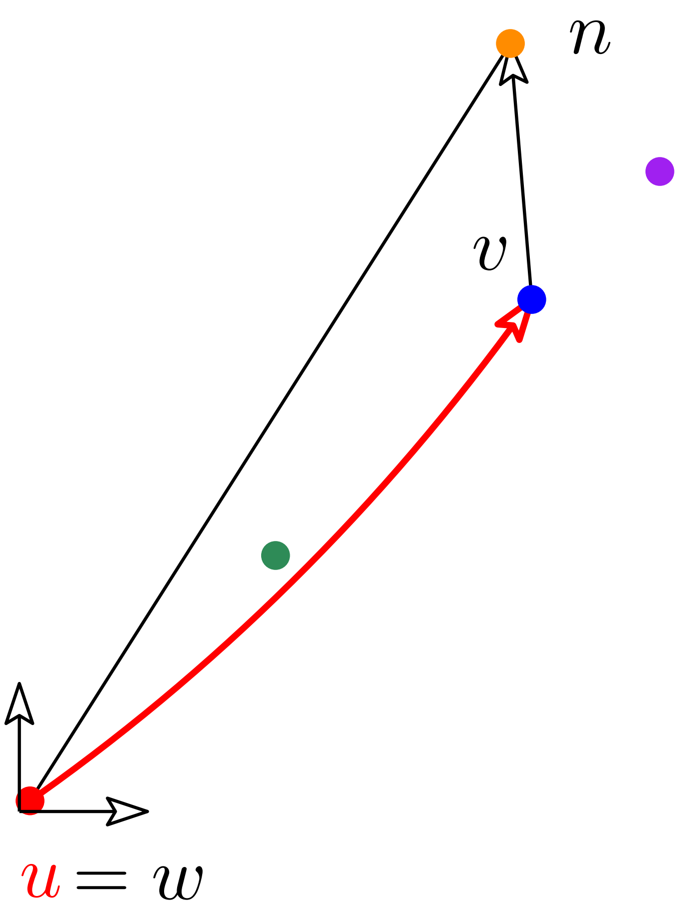
$w_1 = (12, 12)$

~~$w_2 = (24, 24)$~~

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$

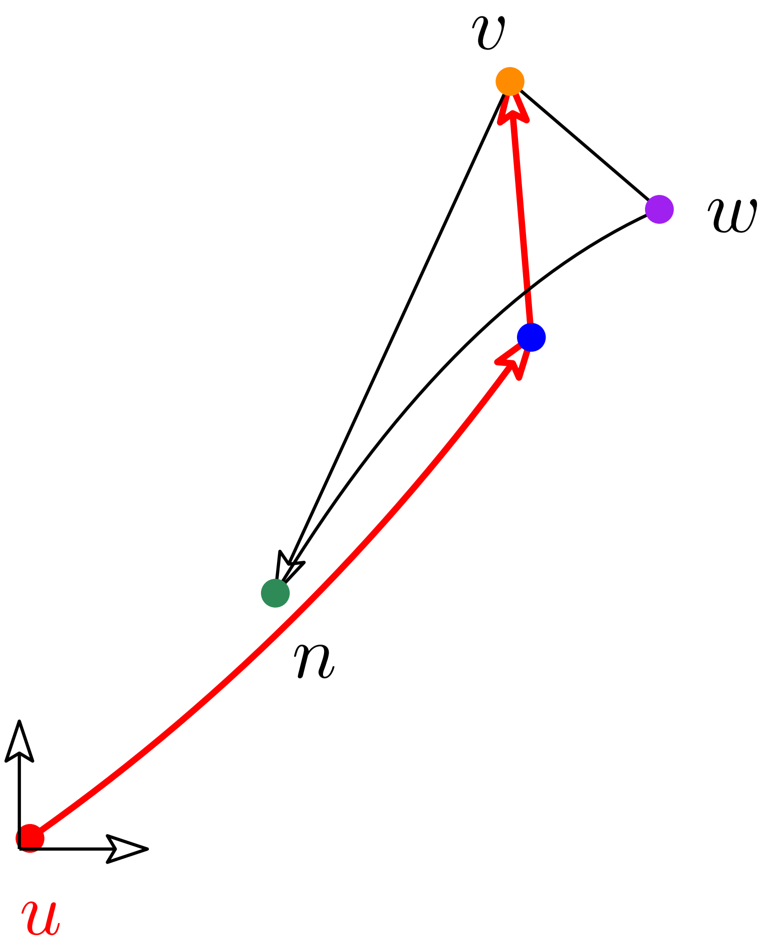


Do
    $n =$ first in S;
    For each $w \in S$
        if vwn positive
            then $n = w$;
    $v.next = n$; $v = n$;
    $S = S \setminus \{v\}$
While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example (single precision)



$w_1 = (12, 12)$

~~$w_2 = (24, 24)$~~

$w_3 = (30, 30.000001)$

~~$w_4 = (23, 36)$~~

$w_5 = (0.5000029, 0.5000027)$

Do

  $n =$ first in S;

  For each $w \in S$

    if vwn positive

     then $n = w$;

  $v.next = n$; $v = n$;

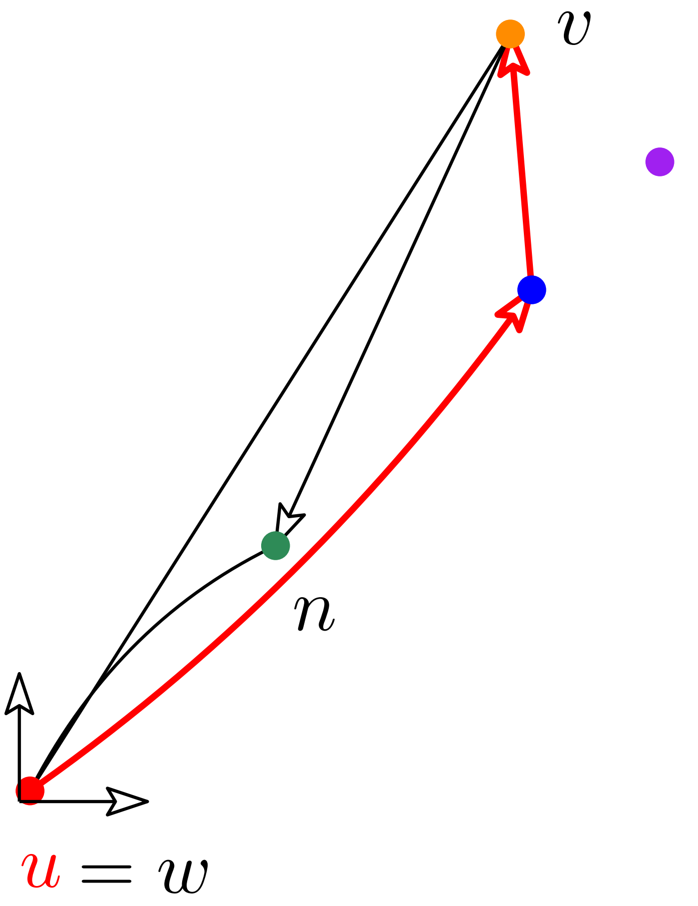
  $S = S \setminus \{v\}$

While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example (single precision)



$w_1 = (12, 12)$

~~$w_2 = (24, 24)$~~

$w_3 = (30, 30.000001)$

~~$w_4 = (23, 36)$~~

$w_5 = (0.5000029, 0.5000027)$

Do
  $n =$ first in S;
  For each $w \in S$
    if vwn positive
      then $n = w$;
  $v.next = n$; $v = n$;
  $S = S \setminus \{v\}$
While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example
(single precision)

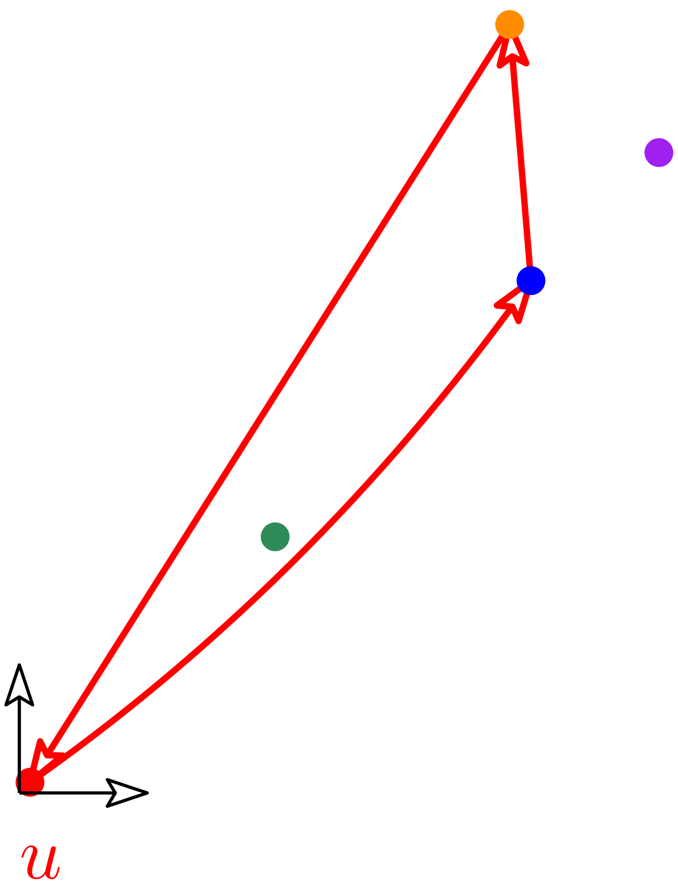
~~$w_1 = (12, 12)$~~

~~$w_2 = (24, 24)$~~

$w_3 = (30, 30.000001)$

~~$w_4 = (23, 36)$~~

$u = \quad w_5 = (0.5000029, 0.5000027)$



Do

$n =$ first in S;

For each $w \in S$

if vwn positive

then $n = w$;

$v.next = n$; $v = n$;

$S = S \setminus \{v\}$

While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example (single precision)

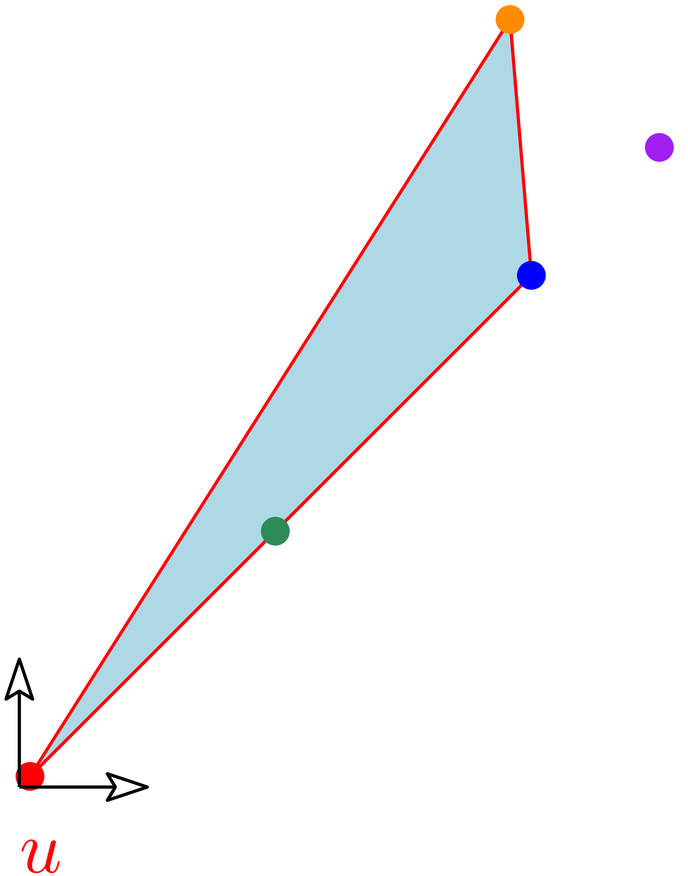
$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$



Result is really wrong

Sous-jacent :

le modèle de calcul

Convex hull

Real Random Access Memory model

Assume exact computation on real numbers

constant time for single operations: $+, -, \sqrt{}, \sin \ldots$

Convex hull

Real RAM model and general position hypothesis

Real Random Access Memory model

Assume exact computation on real numbers

constant time for single operations: $+, -, \sqrt{}, \sin \ldots$

General position hypotheses

Predicate: Input $\longmapsto \{-1, 0, 1\}$

Convex hull

Real Random Access Memory model

Assume exact computation on real numbers

constant time for single operations: $+, -, \sqrt{\ }, \sin \ldots$

General position hypotheses

Predicate: Input $\longmapsto \{-1, 0, 1\}$

2D convex hull: no three points colinear

Convex hull

Real RAM model and general position hypothesis

Real Random Access Memory model

Assume exact computation on real numbers

constant time for single operations: $+, -, \sqrt{}, \sin \ldots$

General position hypotheses

Predicate: Input $\longmapsto \{-1, 0, 1\}$

2D convex hull: no three points colinear

possibly: no 2 points with same x

Convex hull

Geometric algorithms are **designed** and **studied** in the Real RAM model.

Convex hull

Geometric algorithms are **designed** and **studied** in the Real RAM model.

They are **implemented** in a constant-word RAM model.

Convex hull

Geometric algorithms are **designed** and **studied** in the Real RAM model.

They are **implemented** in a constant-word RAM model.

Robustness issues arise because the two models are different.

Convex hull

Geometric algorithms are **designed** and **studied** in the Real RAM model.

They are **implemented** in a constant-word RAM model.

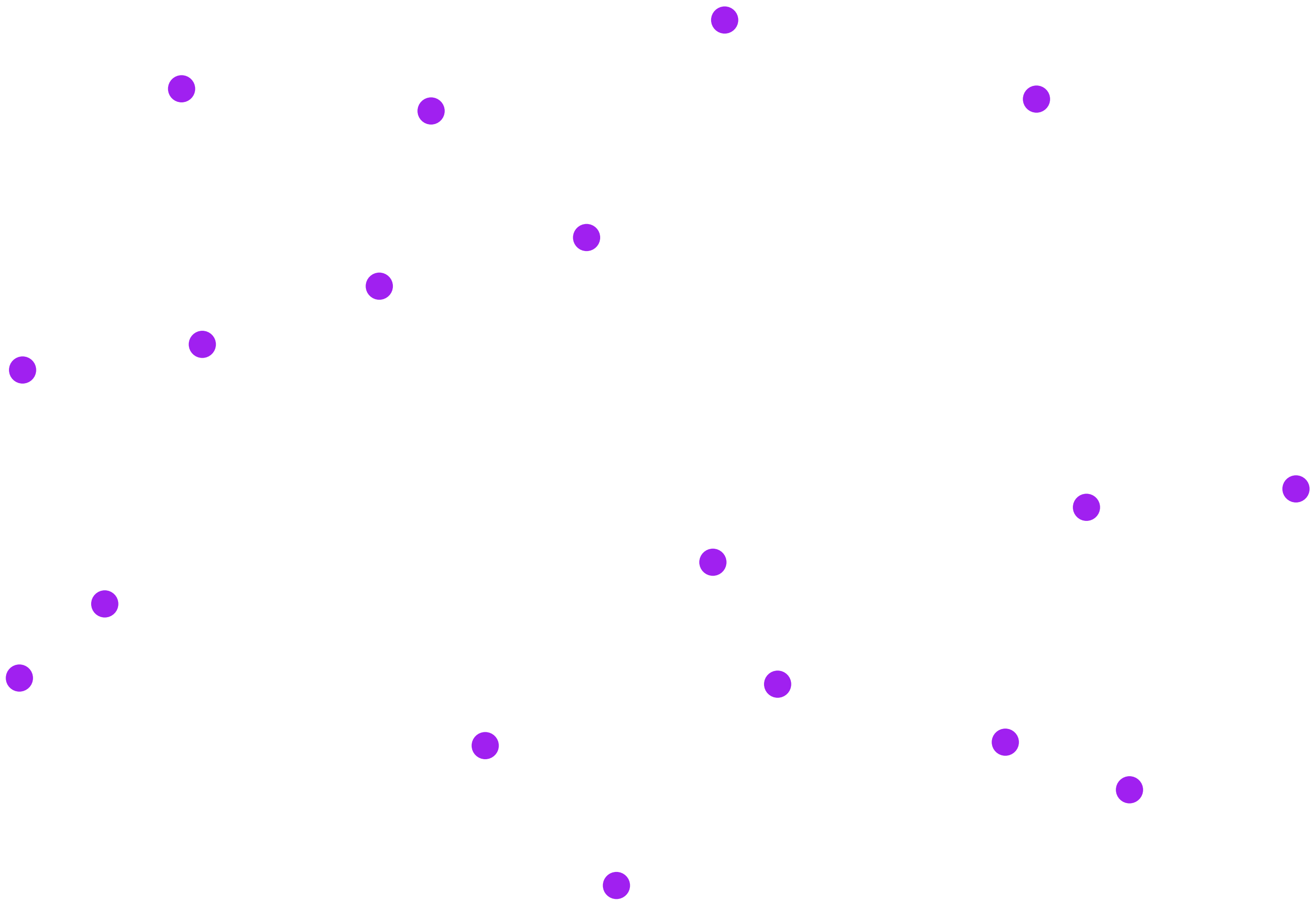
Robustness issues arise because the two models are different.

For now... keep the model in mind.

Un second algorithme

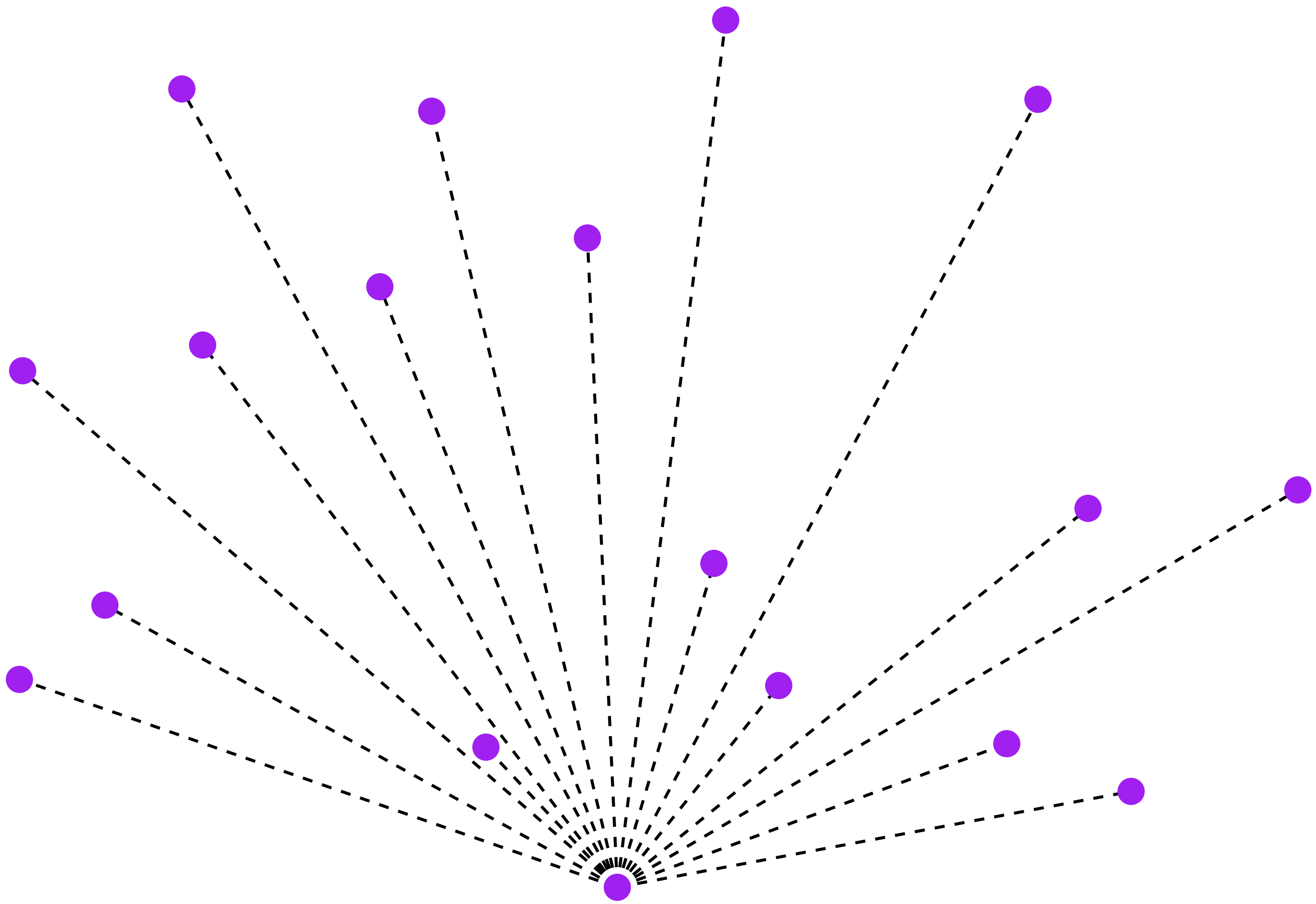
(Graham scan)

Convex hull



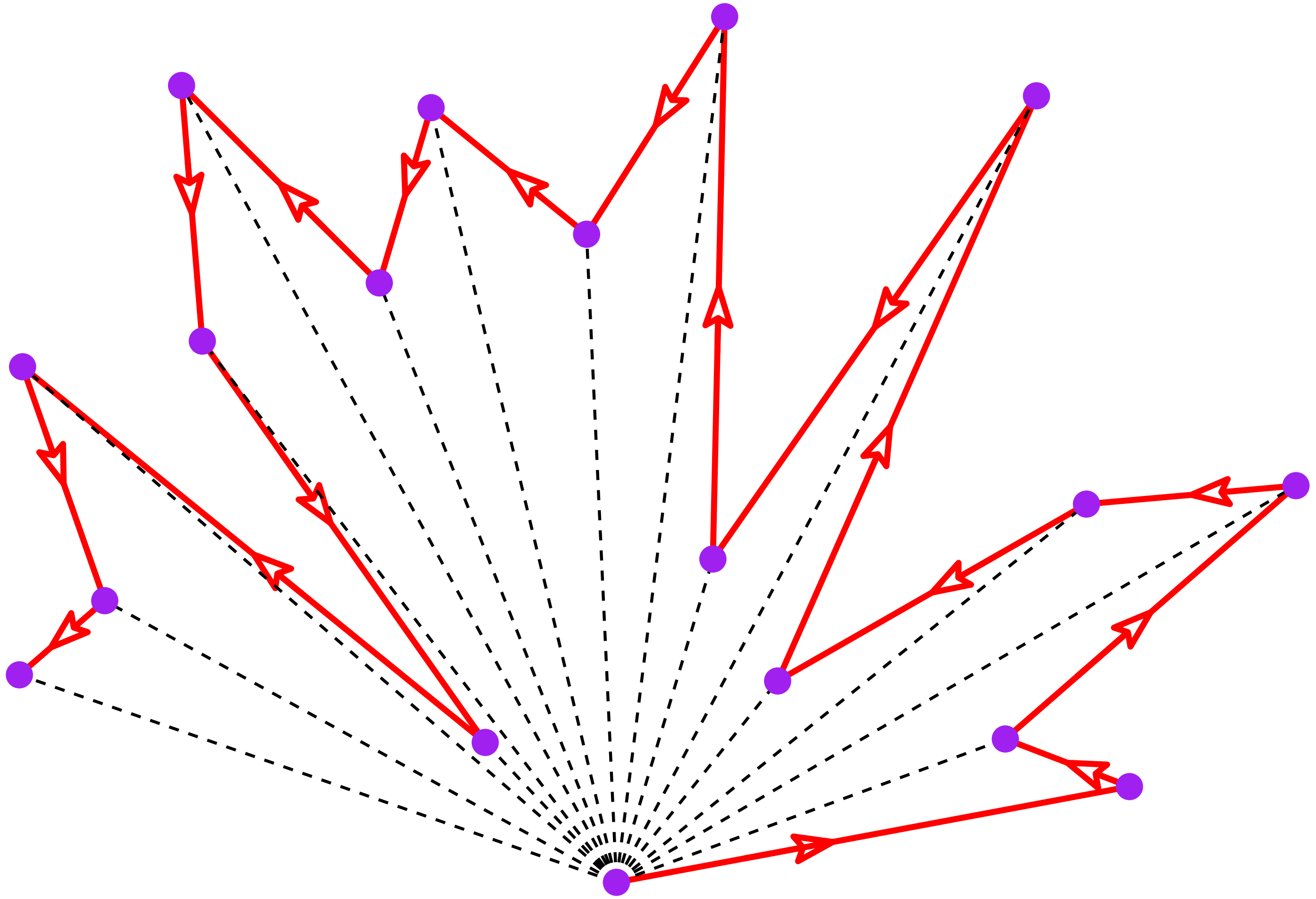
Convex hull

sort around a point (e.g. lowest point)



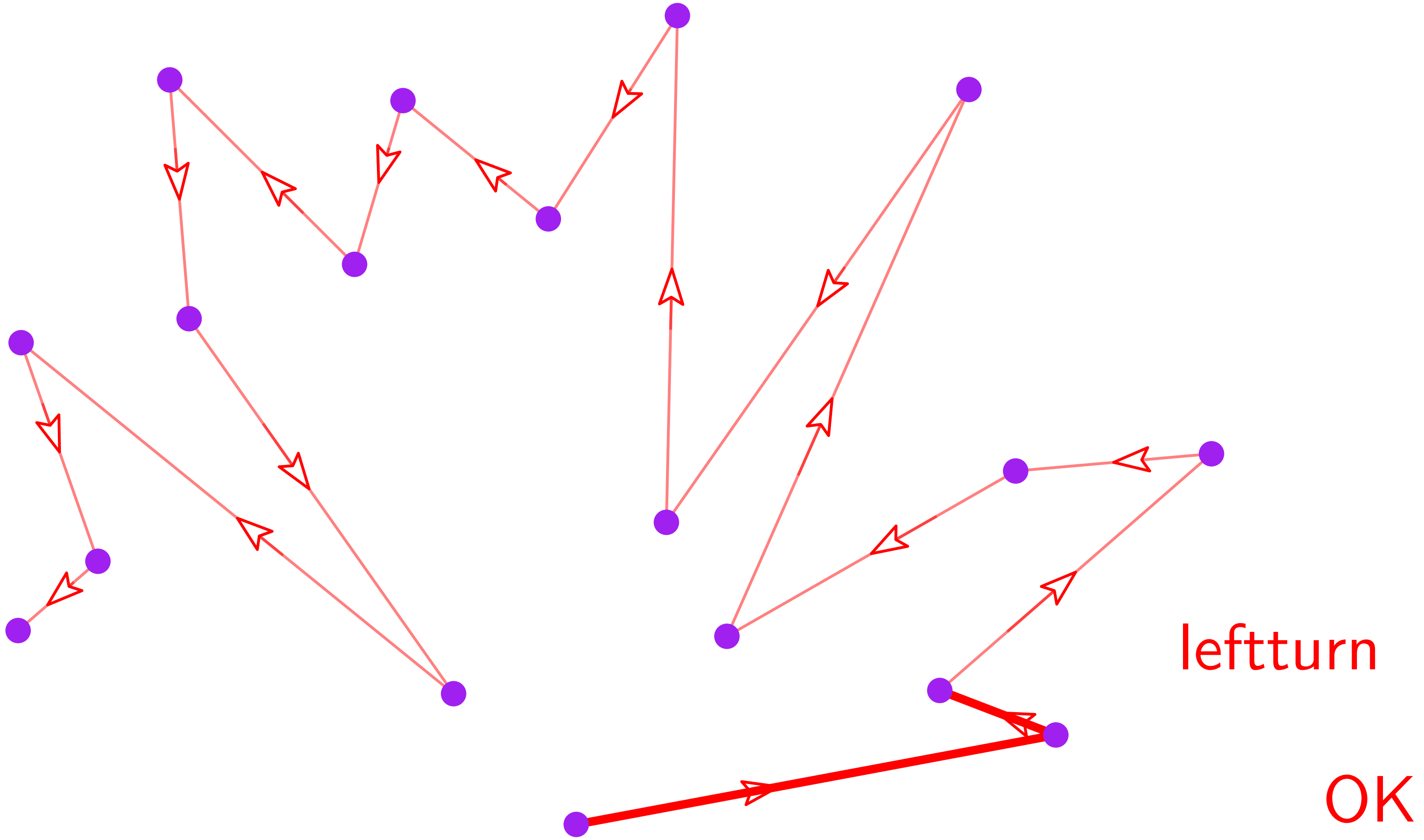
Convex hull

sort around a point (e.g. lowest point)

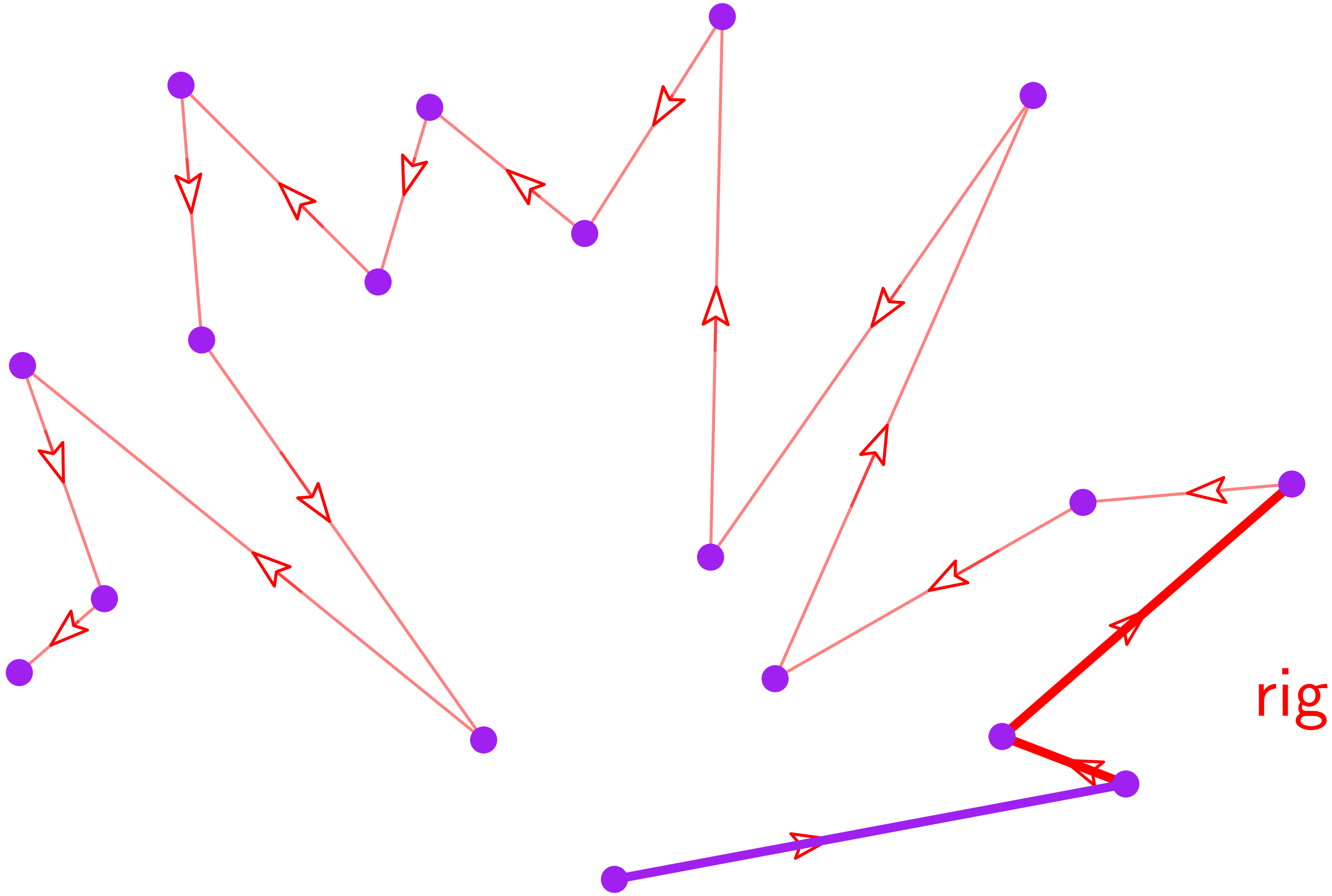


Convex hull

Graham algorithm

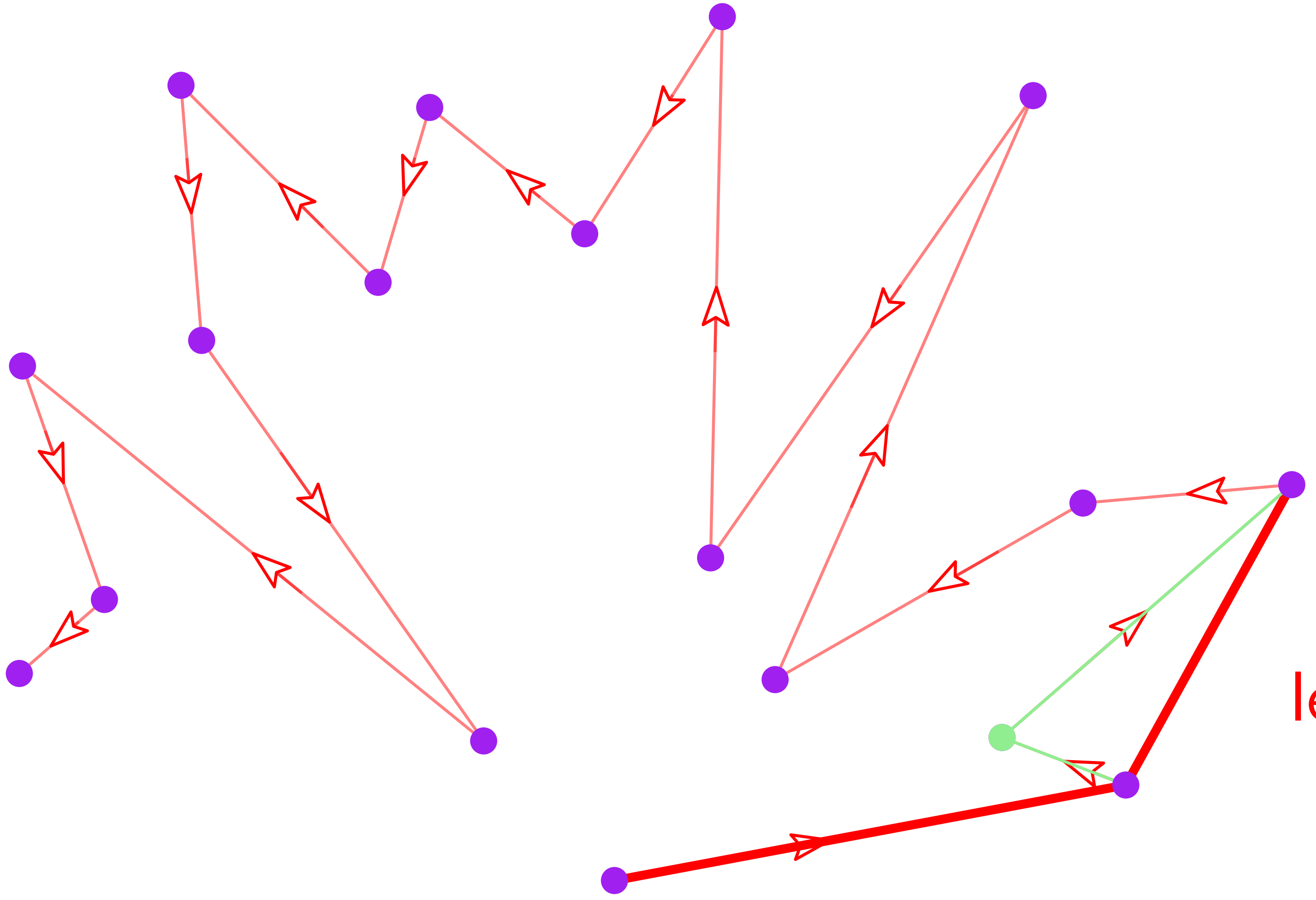


Convex hull

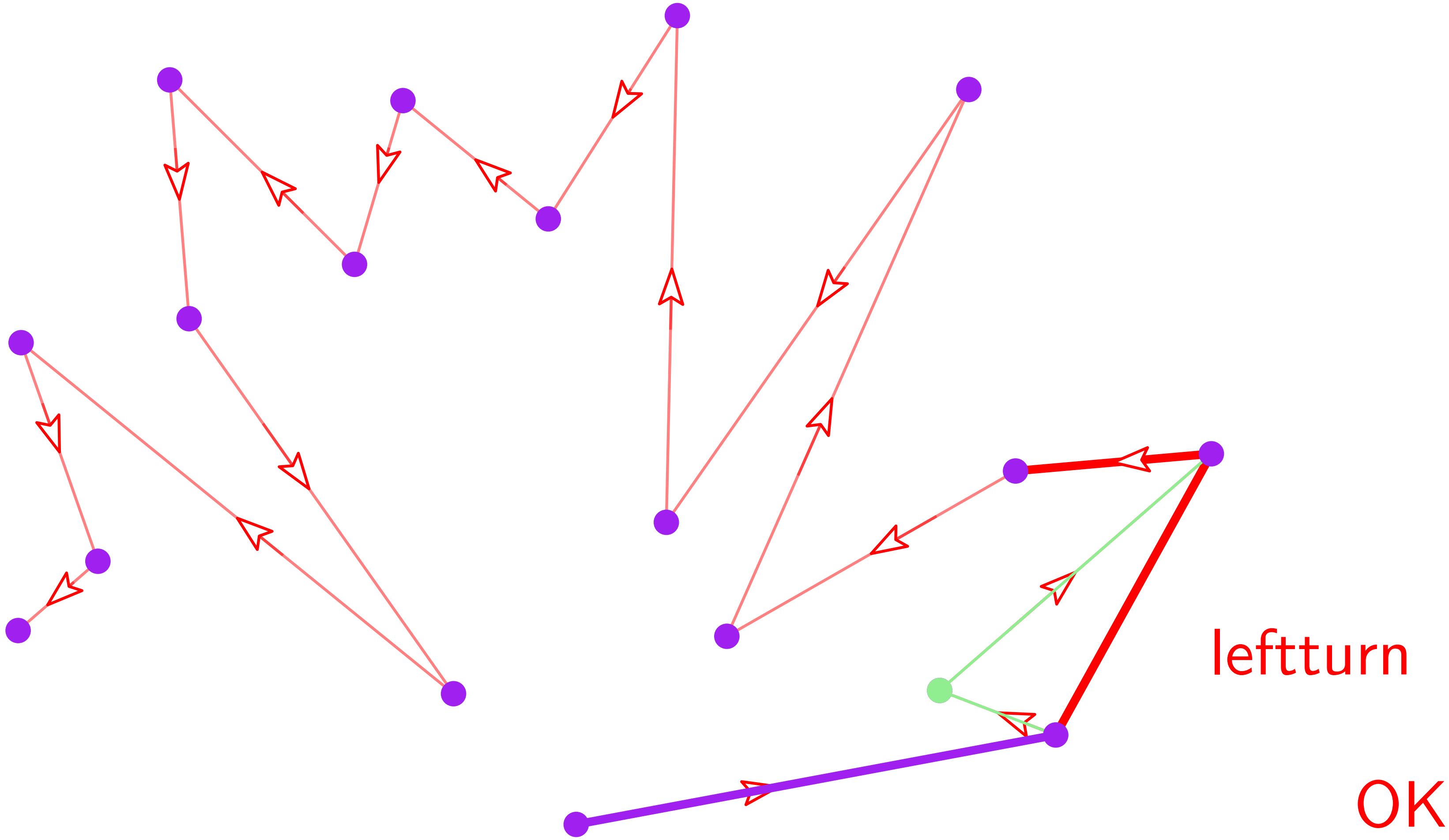


Convex hull

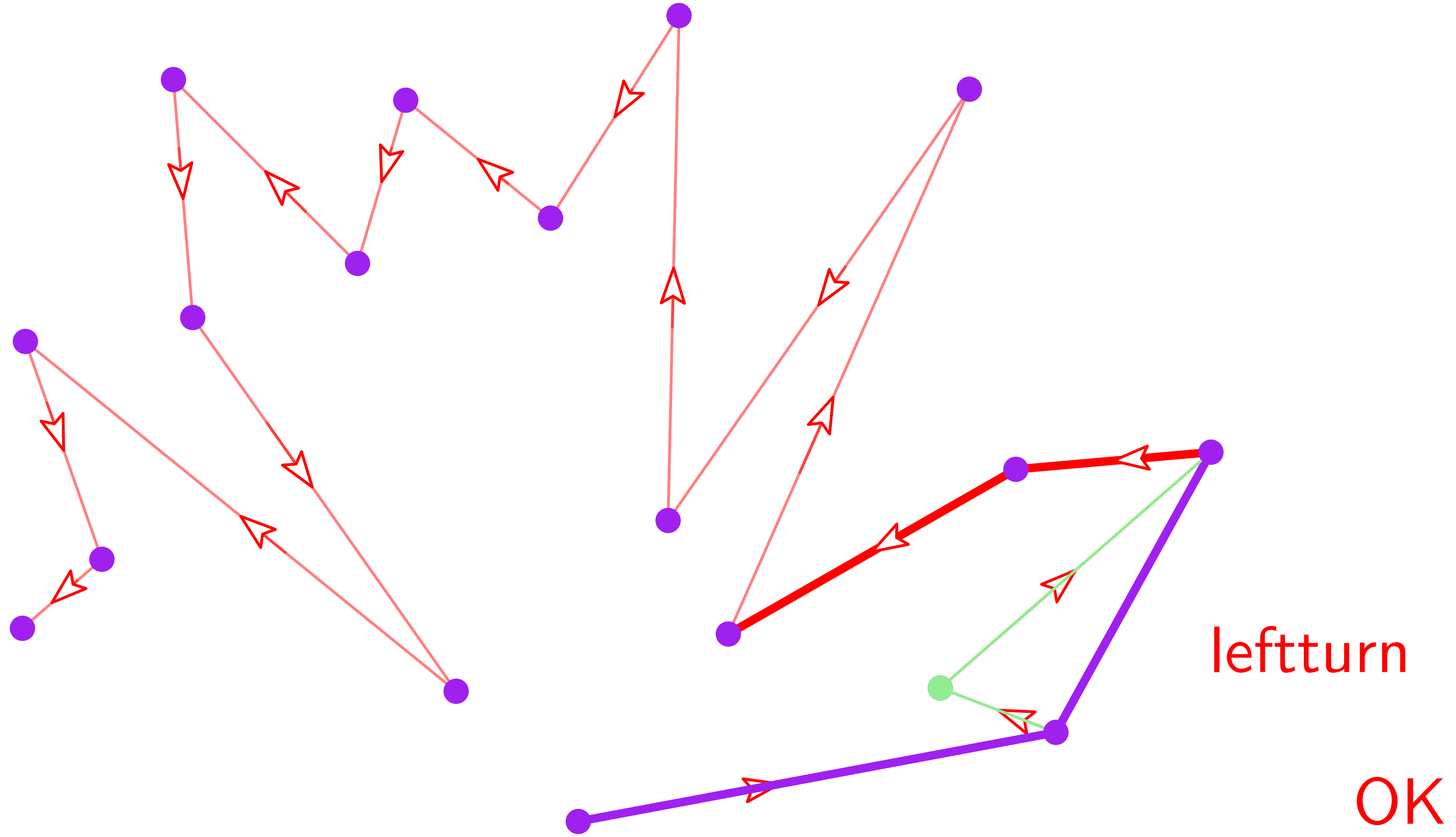
Graham algorithm



Convex hull

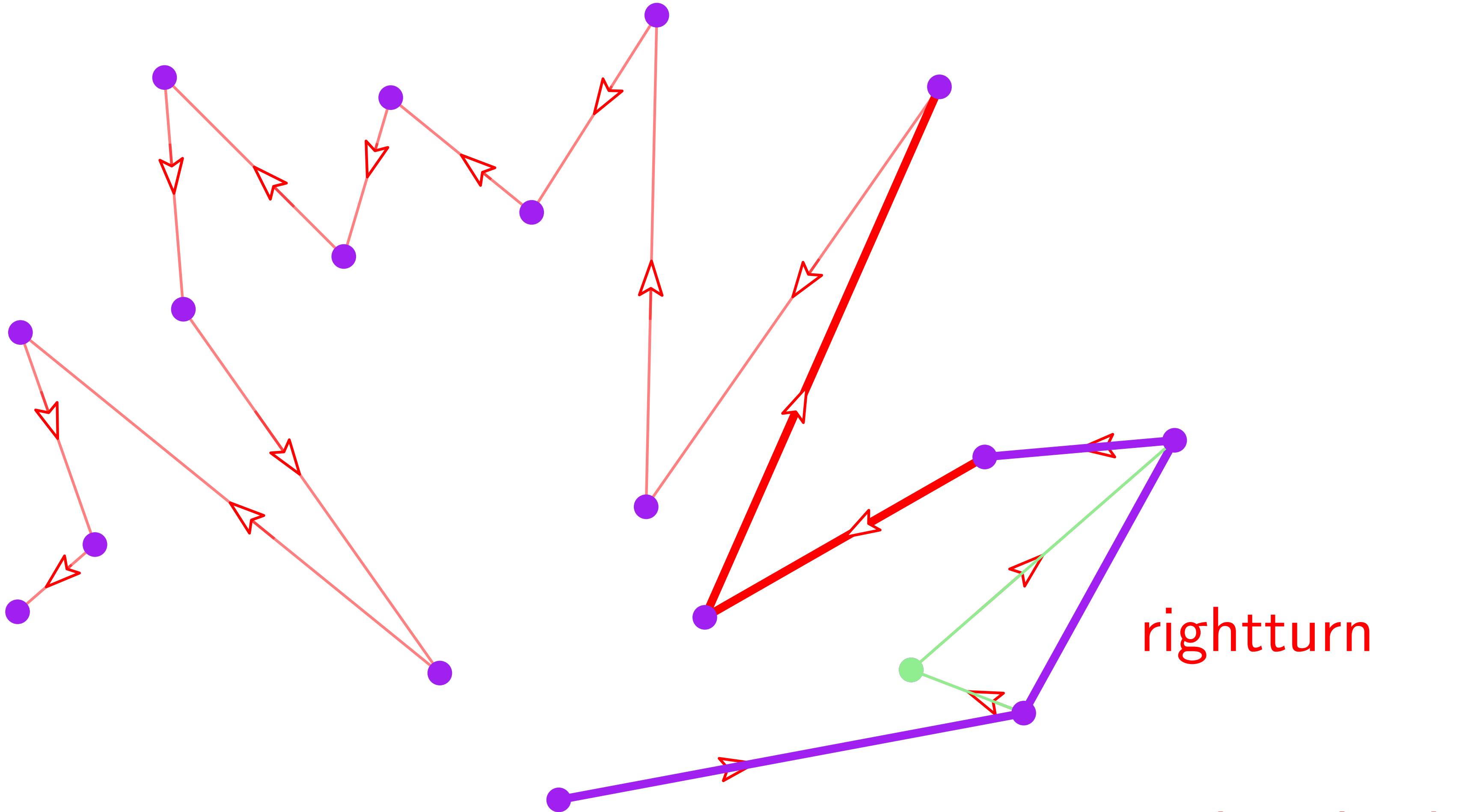


Convex hull

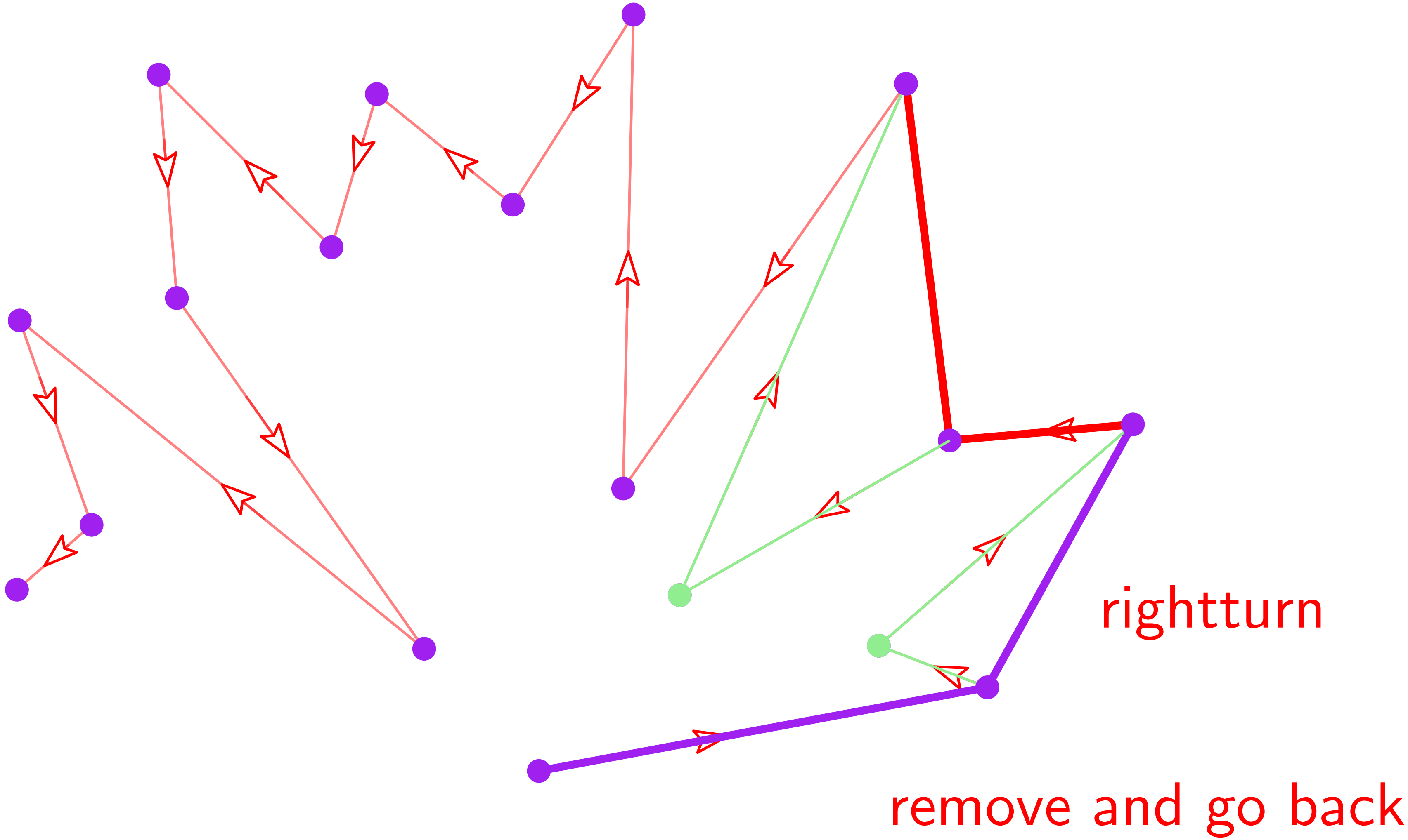


Convex hull

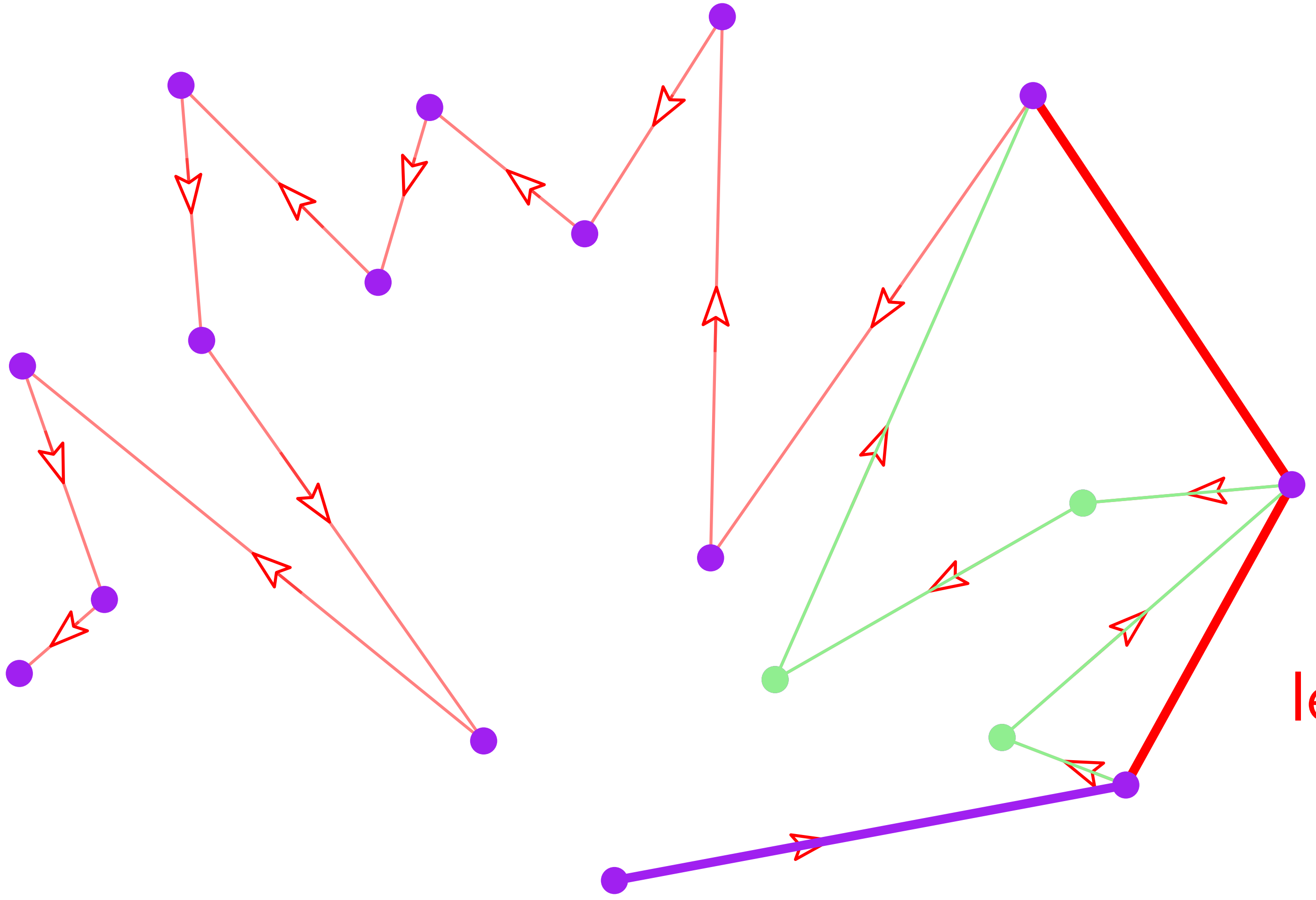
Graham algorithm



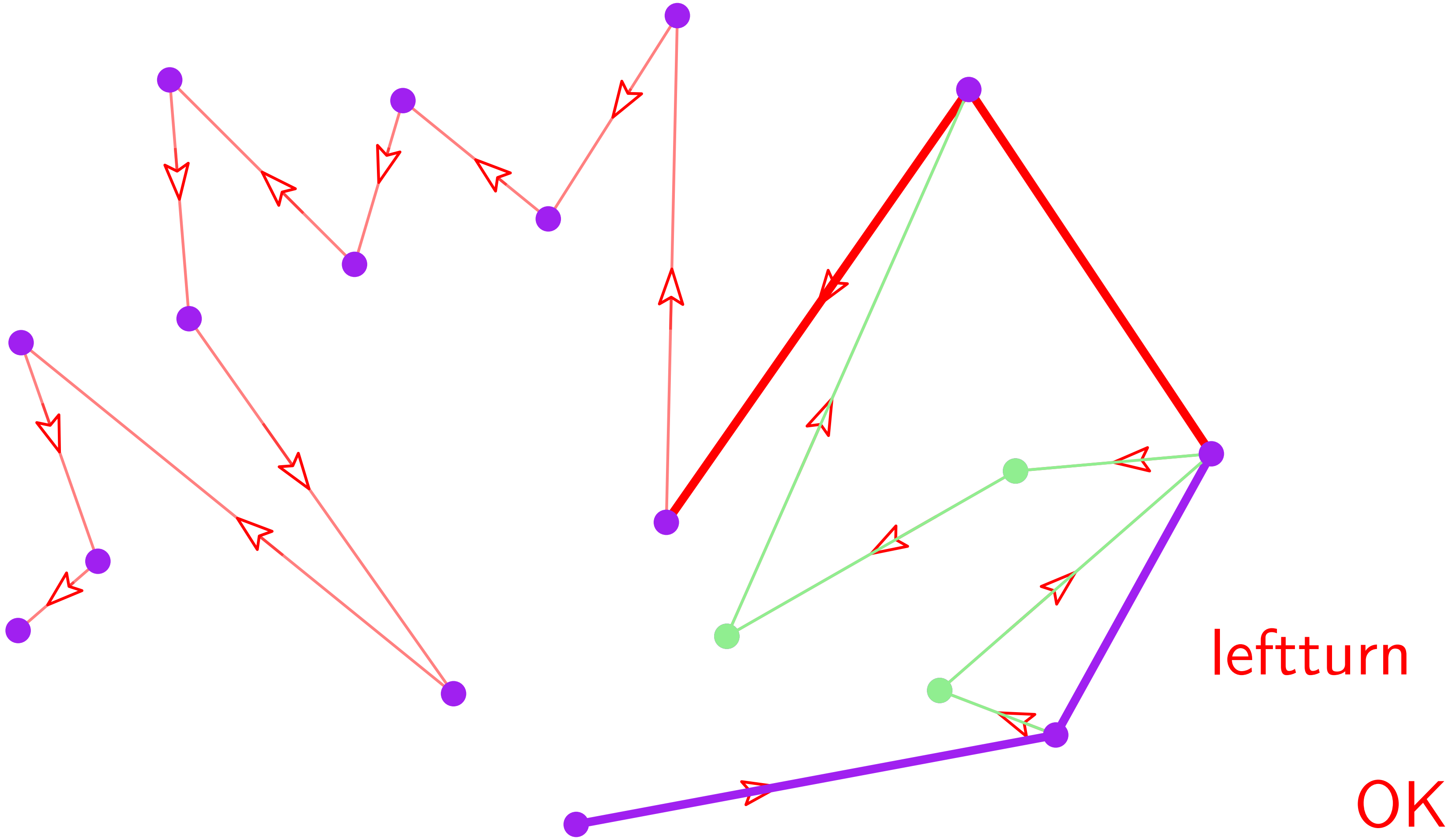
Convex hull



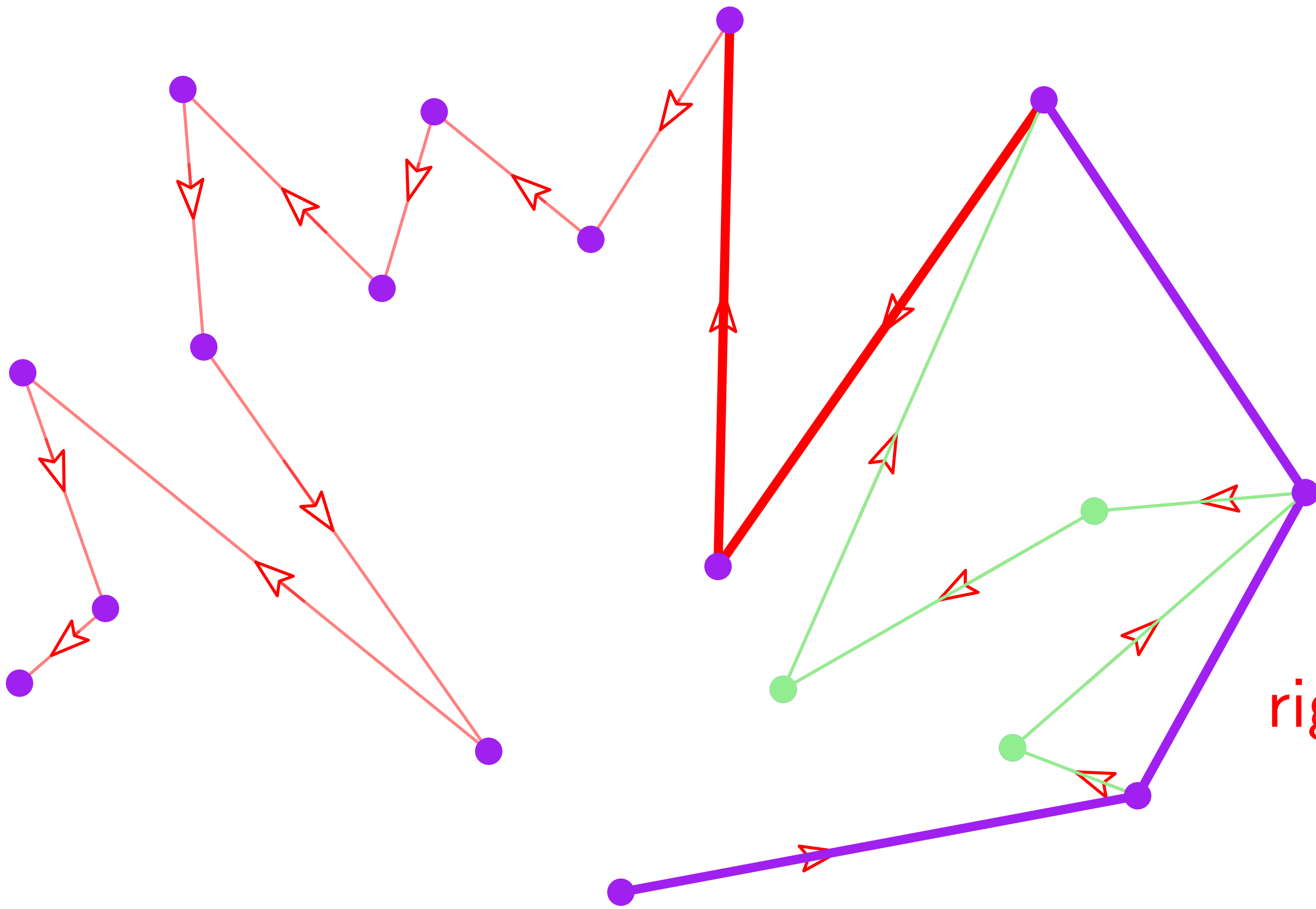
Convex hull



Convex hull



Convex hull

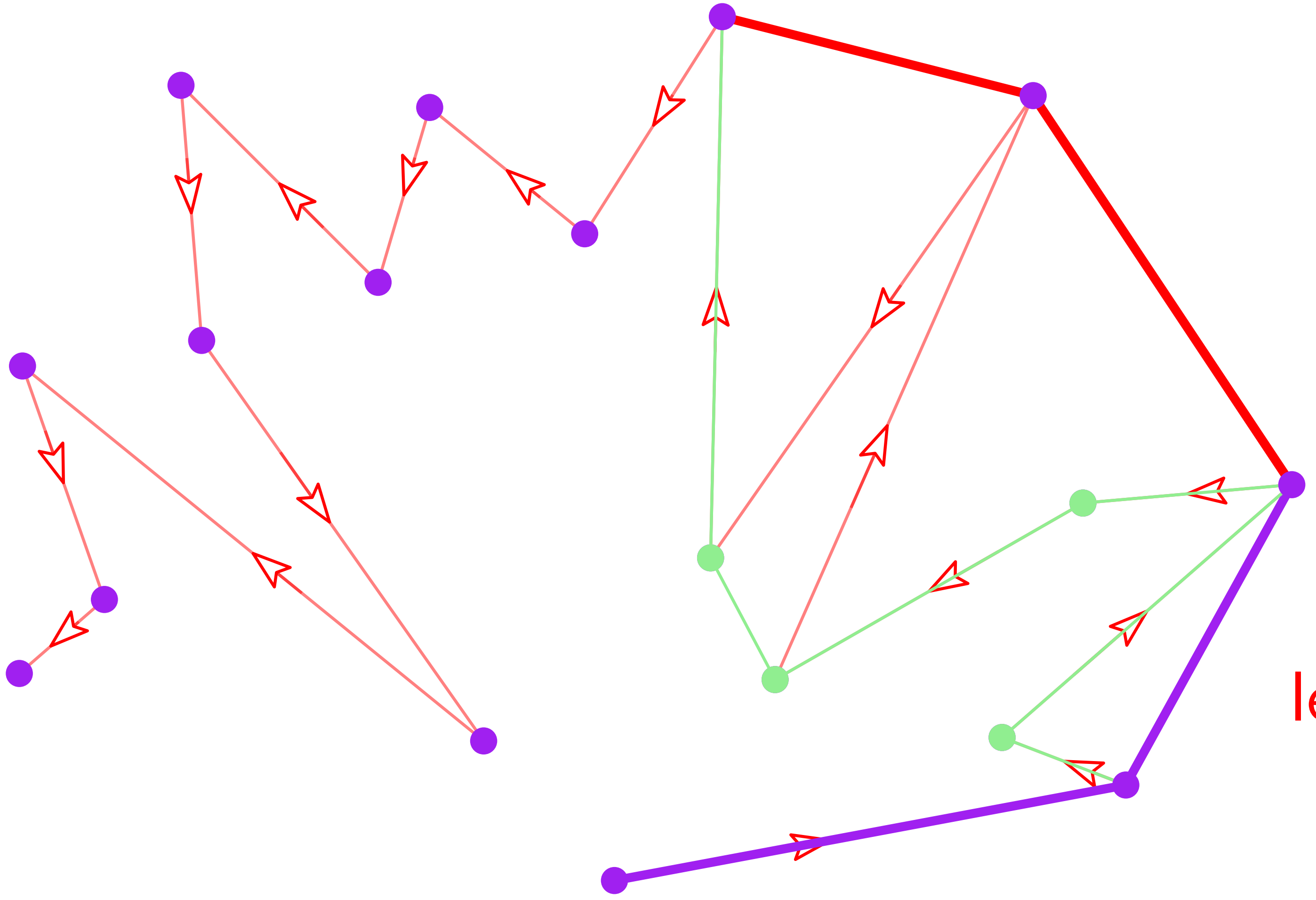


rightturn

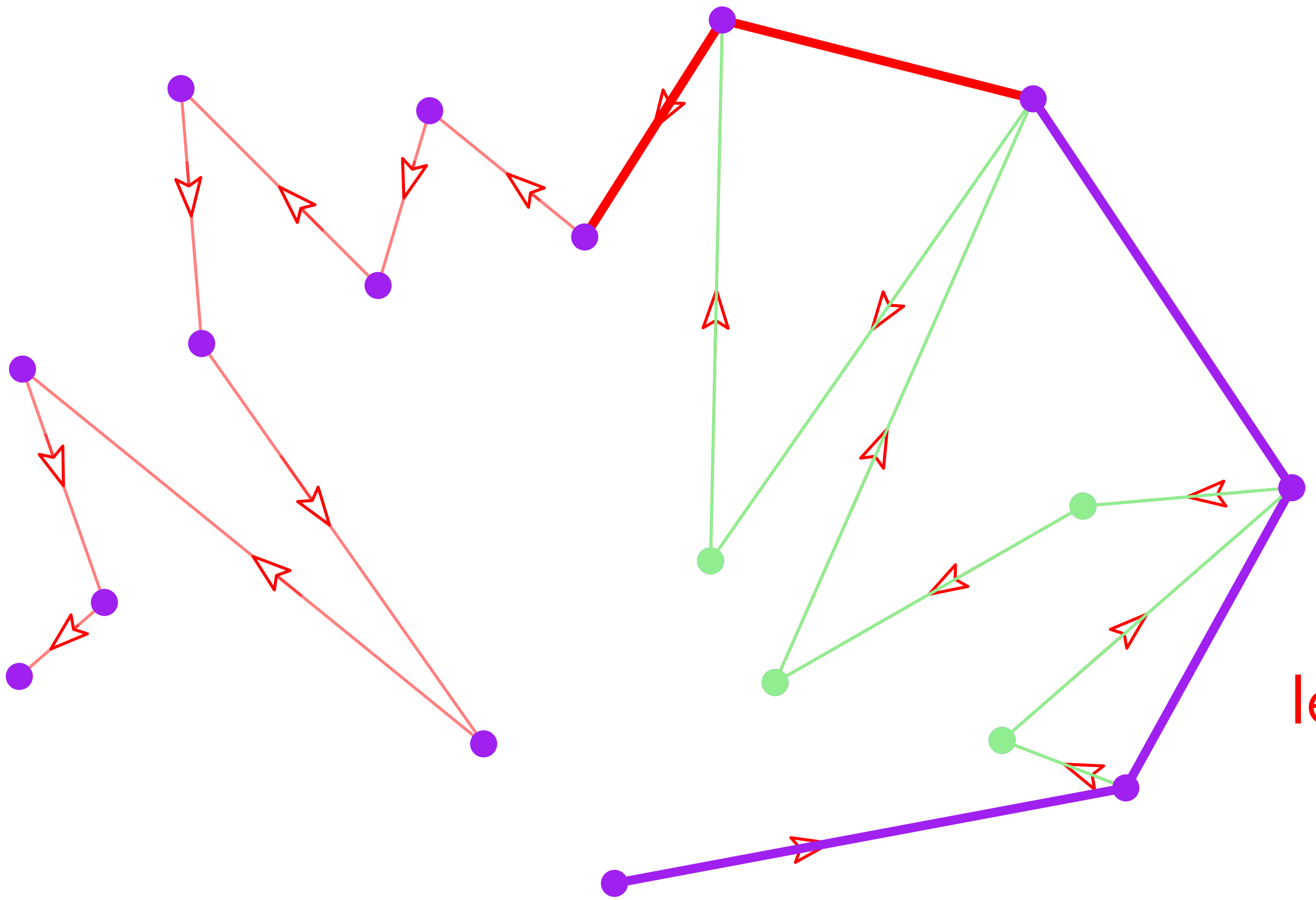
remove and go back

Convex hull

Graham algorithm



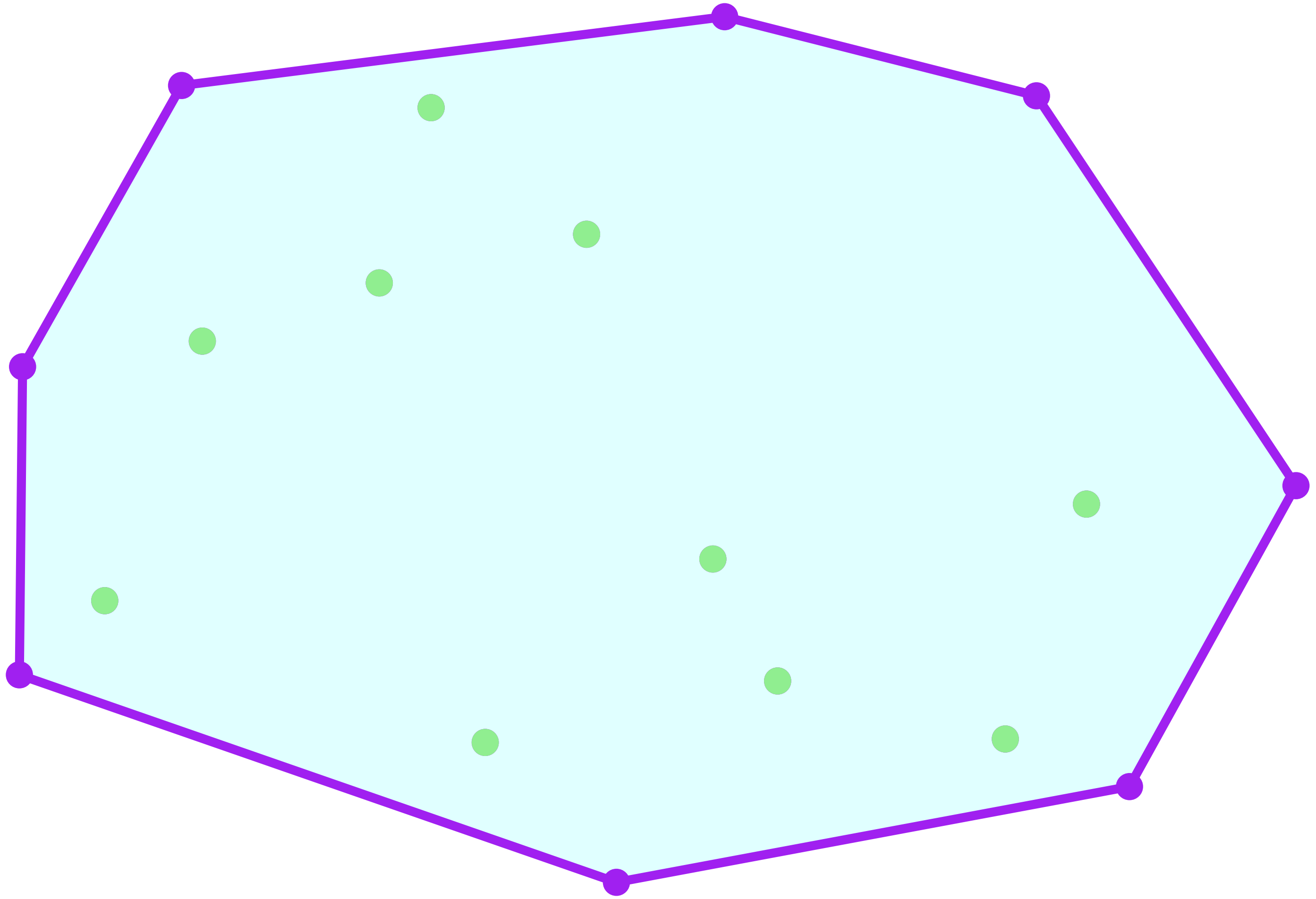
Convex hull



leftturn

OK

Convex hull



Convex hull

Complexity

Graham algorithm

Input: point set S

u lowest point of S;

sort S around u in a circular list including u;

$v = u$;

while $v.next \neq u$

    if $(v, v.next, v.next.next)$ ccw

        $v = v.next$;

    else

        $v.next = v.next.next$; $v.next.previous = v$;

        if $v \neq u$ $v = v.previous$;

Convex hull

Complexity

Graham algorithm

Input: point set S

u lowest point of S;

sort S around u in a circular list including u;

$v = u$;

$O(n)$

while $v.next \neq u$

 if $(v, v.next, v.next.next)$ ccw

  $v = v.next$;

 else

  $v.next = v.next.next$; $v.next.previous = v$;

  if $v \neq u$ $v = v.previous$;

Convex hull

Complexity

Graham algorithm

Input: point set S
u lowest point of S;
sort S around u in a circular list including u;
$v = u$;

$O(n \log n)$

while $v.next \neq u$
    if $(v, v.next, v.next.next)$ ccw
        $v = v.next$;
    else
        $v.next = v.next.next$; $v.next.previous = v$;
        if $v \neq u$ $v = v.previous$;

Convex hull

Complexity

Graham algorithm

Input: point set S

u lowest point of S;

sort S around u in a circular list including u;

$v = u$;

while $v.next \neq u$

if $(v, v.next, v.next.next)$ ccw

$v = v.next$;

else

$v.next = v.next.next$; $v.next.previous = v$;

if $v \neq u$ $v = v.previous$;

Convex hull

Complexity

Input: point set S

u lowest point of S;

sort S around u in a circular list including u;

$v = u$;

while $v.next \neq u$

  if $(v, v.next, v.next.next)$ ccw

    $v = v.next$;

  else

    $v.next = v.next.next$; $v.next.previous = v$;

    if $v \neq u$ $v = v.previous$;

delete one point

at most n times

Convex hull

Complexity

Input: point set S
u lowest point of S;
sort S around u in a circular list including u;
$v = u$;

while $v.next \neq u$
    if $(v, v.next, v.next.next)$ ccw
        $v = v.next;$
    else
        $v.next = v.next.next;$ $v.next.previous = v;$
        if $v \neq u$ $\quad v = v.previous;$

distance to u decreases
at most n times

delete one point
at most n times

Convex hull

Complexity

Input: point set S
u lowest point of S;
sort S around u in a circular list including u;
$v = u$;

$O(n)$

while $v.next \neq u$
  if $(v, v.next, v.next.next)$ ccw
    $v = v.next;$
  else
    $v.next = v.next.next; v.next.previous = v;$
    if $v \neq u \quad v = v.previous;$

distance to u decreases
at most n times

delete one point
at most n times

Convex hull

Complexity

Graham algorithm

Input: point set S
u lowest point of S;
sort S around u in a circular list including u;
$v = u$;

$O(n \log n)$

while $v.next \neq u$
  if $(v, v.next, v.next.next)$ ccw
    $v = v.next$;
  else
    $v.next = v.next.next$; $v.next.previous = v$;
    if $v \neq u$ $v = v.previous$;

$$n^2 \to n \log n \to \ldots ?$$

Convex hull

Problem lower bound is $\Omega(f(n))$

Iff there is NO algorithm

solving all size n problems

using less than $Cf(n)$ operations

$\forall n$

C constant independent of n

Convex hull

Problem lower bound is $\Omega(f(n))$ in a model of computation

Iff there is NO algorithm in that model

solving all size n problems

using less than $Cf(n)$ operations of that model

$\forall n$

C constant independent of n

Sorting

Input: n real (positive) numbers

Sorting

Input: n real (positive) numbers

Output: sorting permutation

Sorting

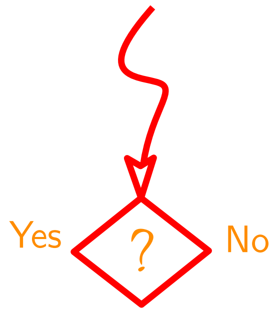
Input: n real (positive) numbers

Output: sorting permutation

Monitoring execution

Sorting

Input: n real (positive) numbers

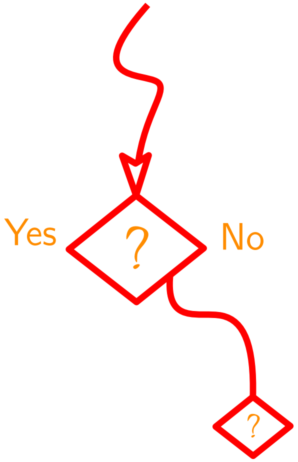


Output: sorting permutation

Monitoring execution

Sorting

Input: n real (positive) numbers



Output: sorting permutation

Monitoring execution

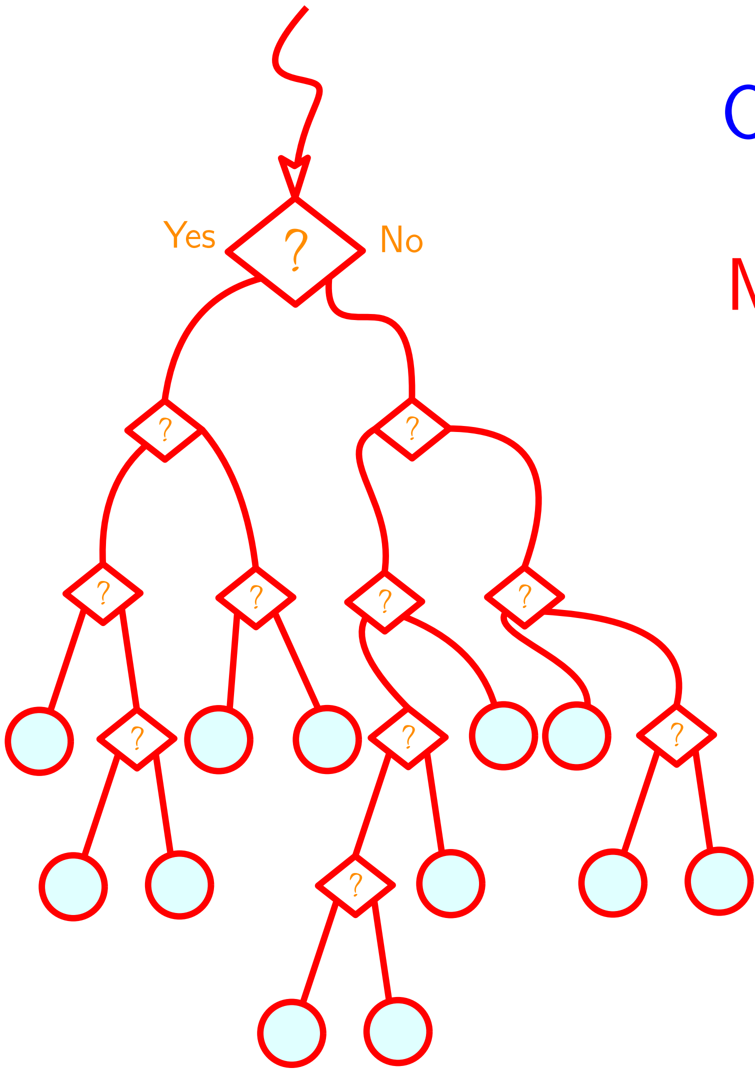
Sorting

Lower bound

Input: n real (positive) numbers

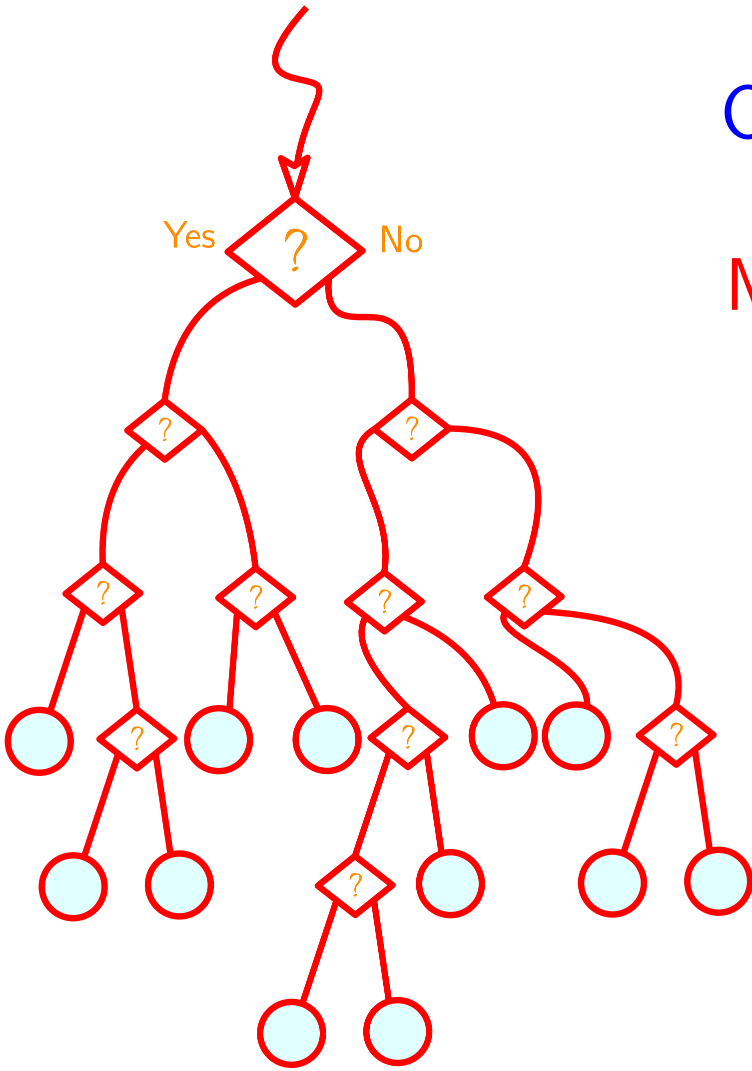
Output: sorting permutation

Monitoring execution



Sorting

Input: n real (positive) numbers



Output: sorting permutation

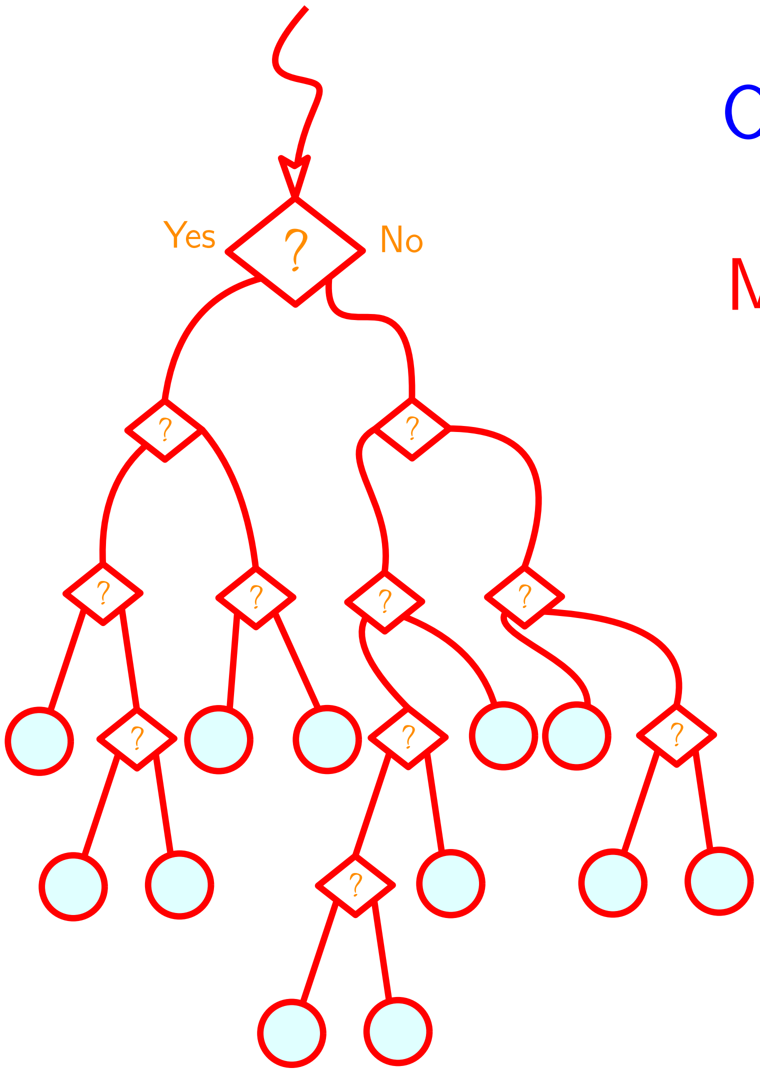
Monitoring execution

$\sharp$ leaves $\geq$ $\sharp$ permutations

Sorting

Lower bound

Input: n real (positive) numbers



Output: sorting permutation

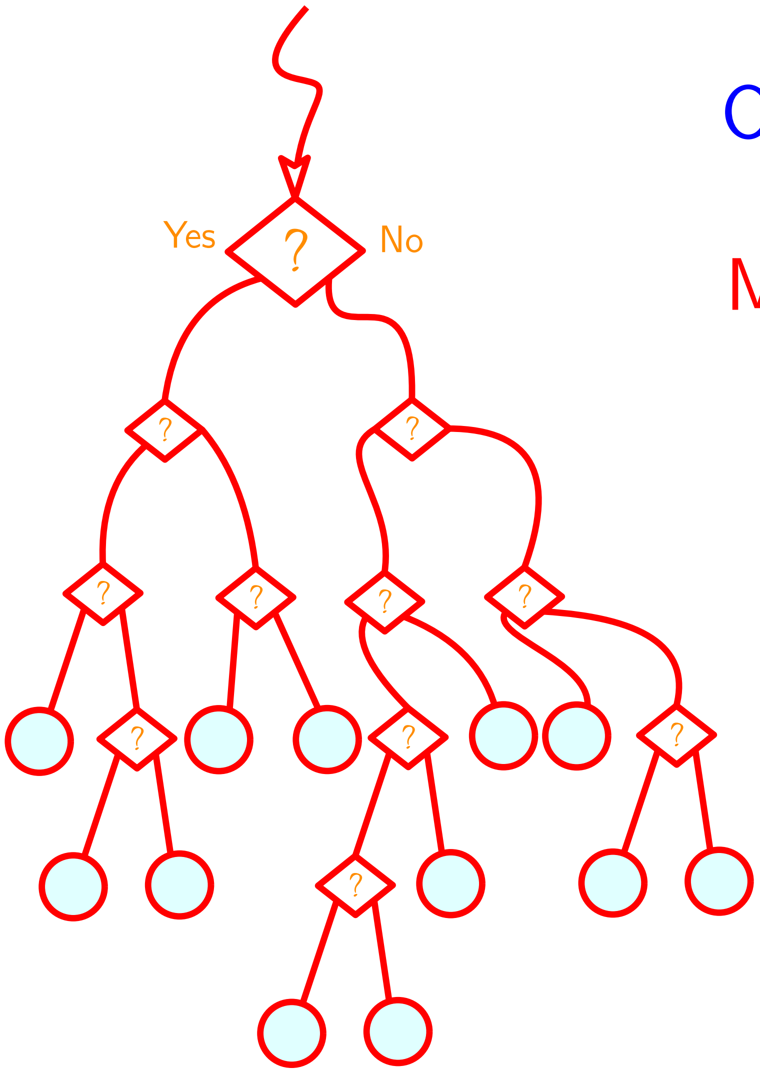
Monitoring execution

$\sharp$ leaves $\geq$ $\sharp$ permutations

There are $n!$ permutations

Sorting

Input: n real (positive) numbers



Output: sorting permutation

Monitoring execution

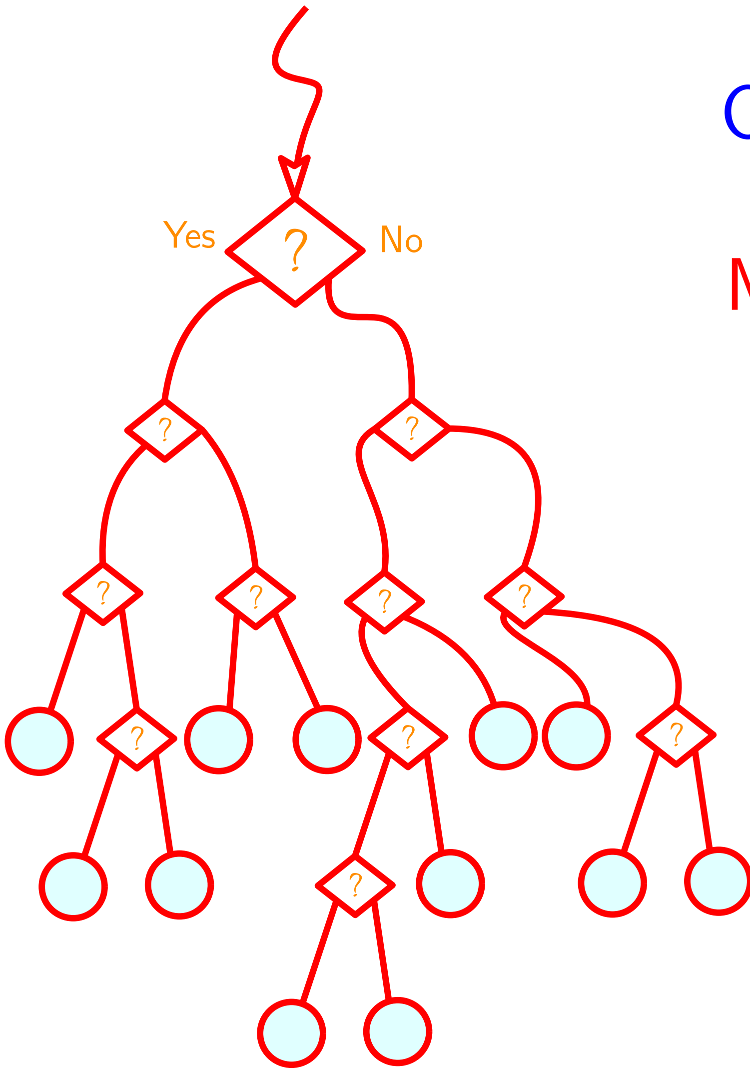
$\sharp$ leaves $\geq$ $\sharp$ permutations

There are $n!$ permutations

Tree height is at least $\log_2 \sharp$ leaves

Sorting

Input: n real (positive) numbers



Output: sorting permutation

Monitoring execution

$\sharp$ leaves $\geq$ $\sharp$ permutations

There are $n!$ permutations

Tree height is at least $\log_2 \sharp$ leaves

$\sharp$ comparisons $\leq \log_2 n! \simeq n \log_2 n$

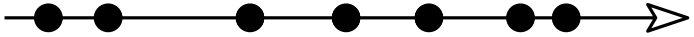
Convex hull

Input: n 2D points (real coordinates)

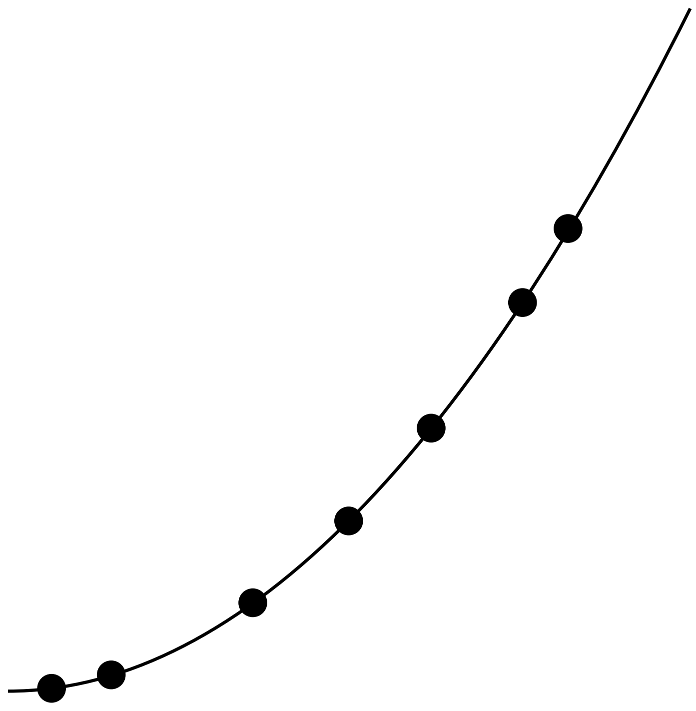
Output: list of points along the convex hull

Convex hull

A stupid algorithm for sorting numbers

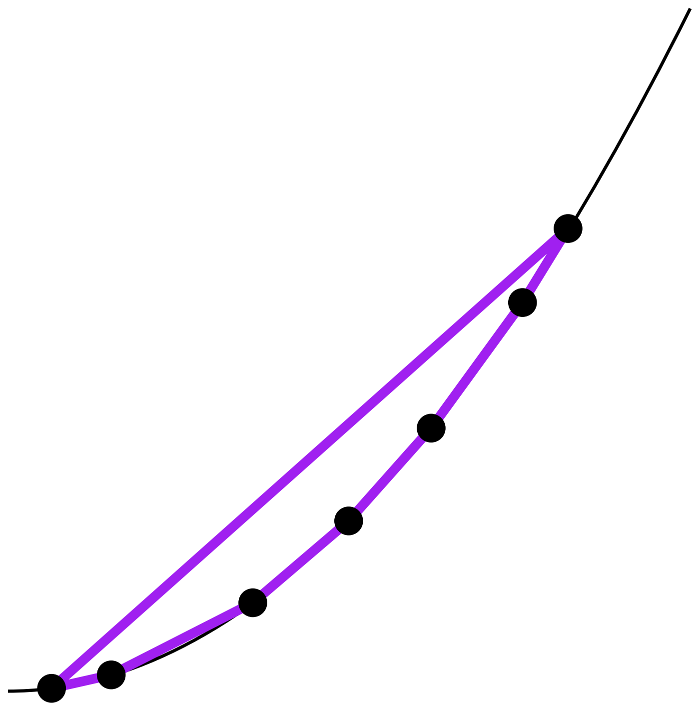


Convex hull



project on parabola

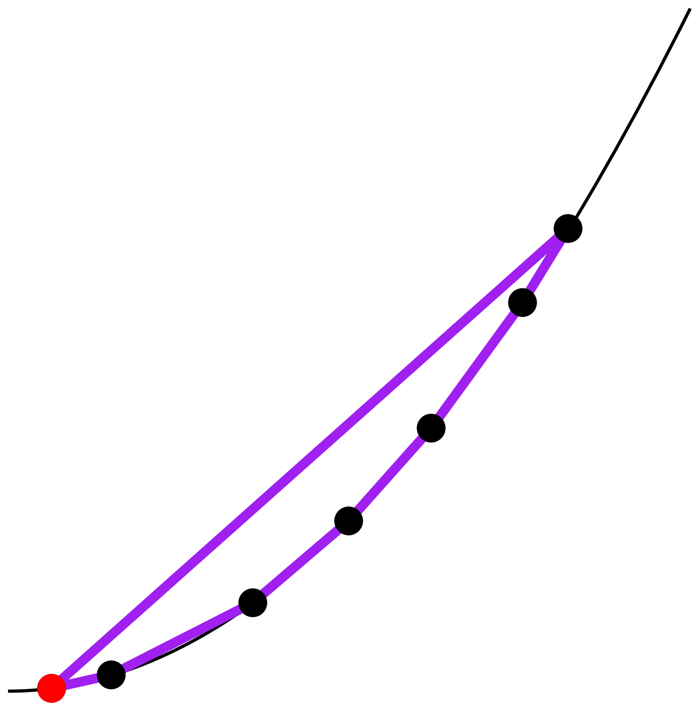
Convex hull



project on parabola

compute convex hull

Convex hull

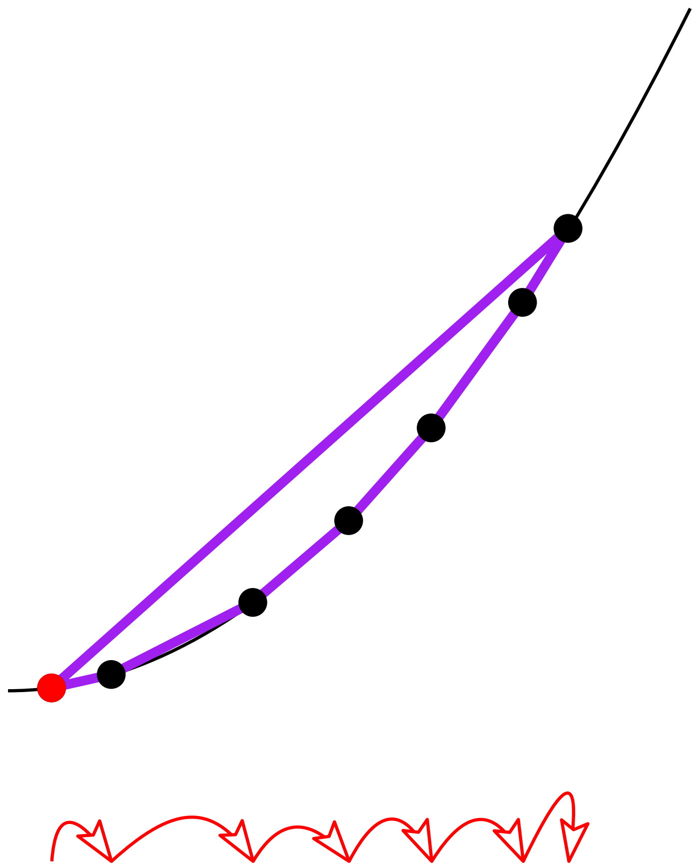


project on parabola

compute convex hull

find lowest point

Convex hull



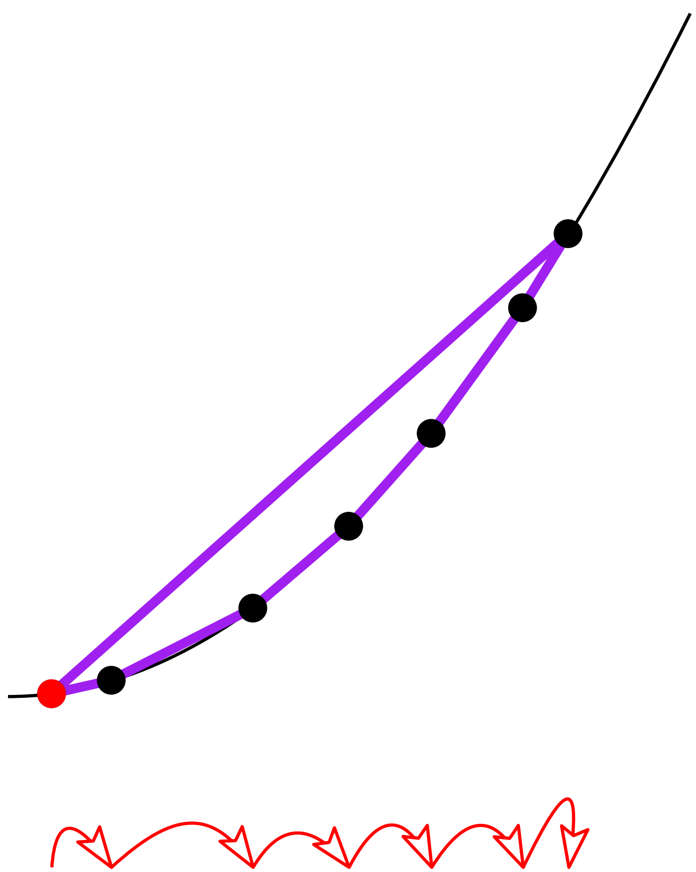
project on parabola

compute convex hull

find lowest point

enumerate x coordinates
in ccw CH order

Convex hull



$O(n)$ project on parabola

$f(n)$ compute convex hull

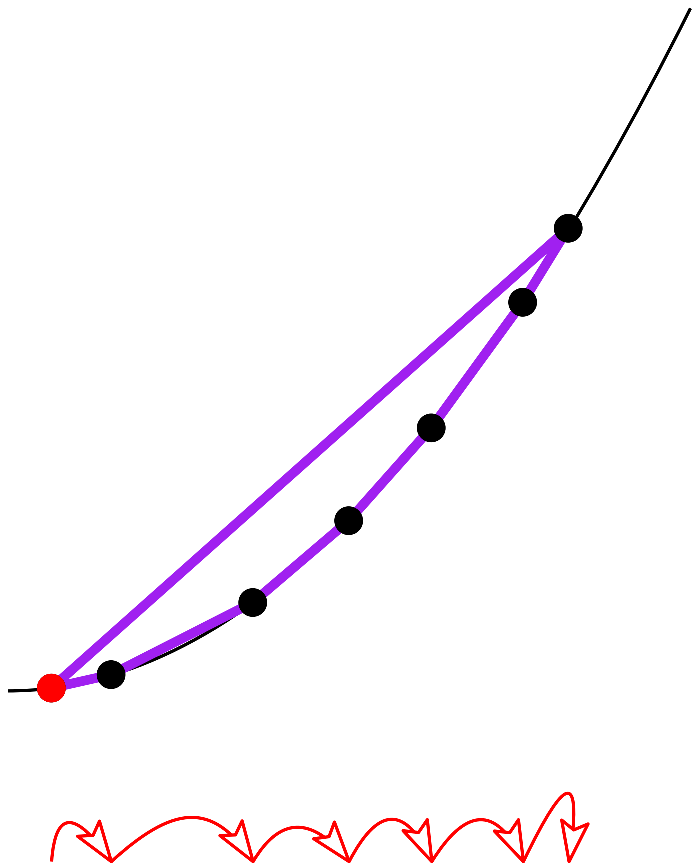
$O(n)$ find lowest point

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

Lower bound on sorting

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

Convex hull



$O(n)$ project on parabola

$f(n)$ compute convex hull

$O(n)$ find lowest point

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

Lower bound on sorting

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

Et en 3D ?

Convex hull

Euler relation

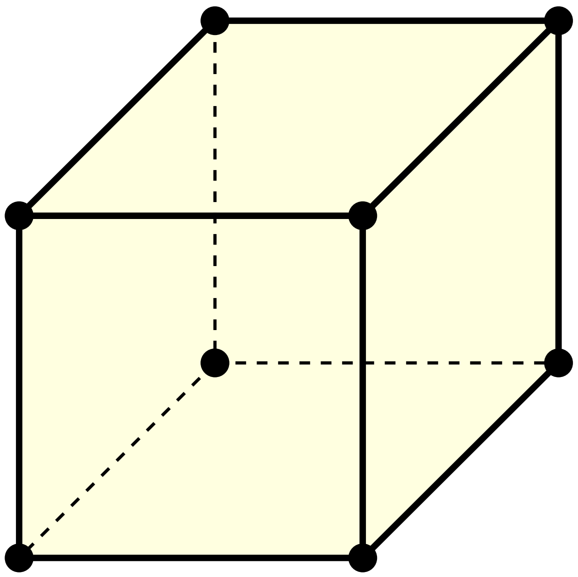
Polytope boundary

Vertices Edges Faces

Convex hull

Euler relation

Polytope boundary



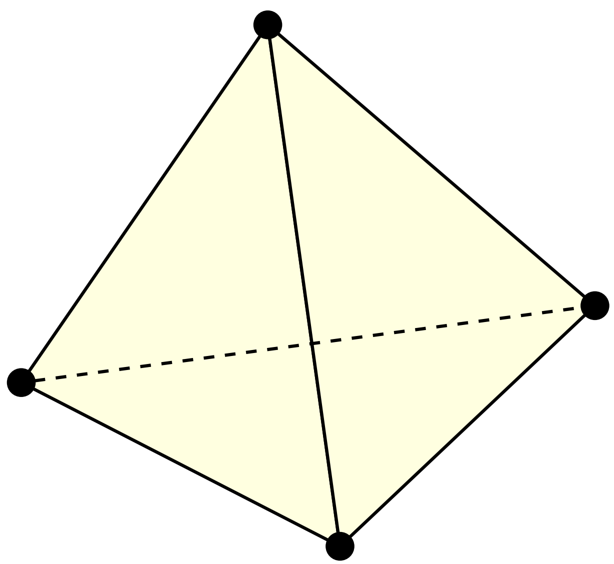
Vertices Edges Faces

$$8 \quad - \quad 12 \quad + \quad 6 \qquad = 2$$

Convex hull

Euler relation

Polytope boundary

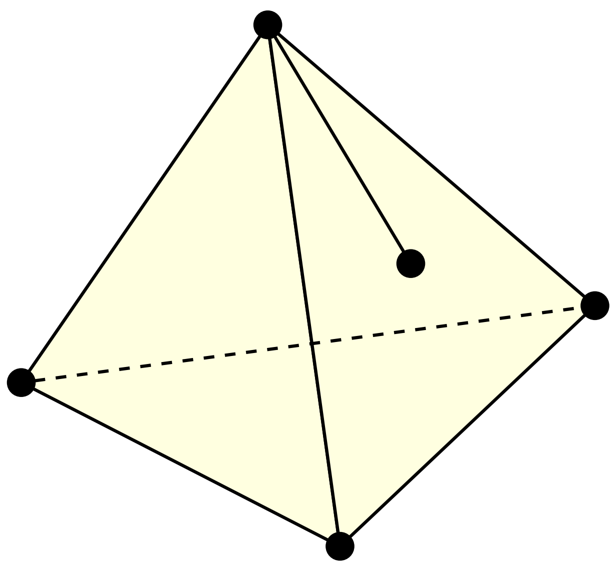


Vertices		Edges		Faces	
8	$-$	12	$+$	6	$= 2$
4	$-$	6	$+$	4	$= 2$

Convex hull

Euler relation

Polytope boundary

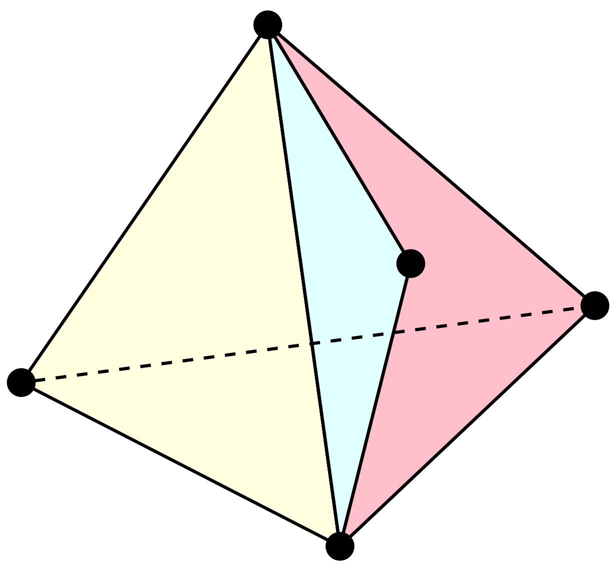


Vertices		Edges		Faces	
8	$-$	12	$+$	6	$= 2$
4	$-$	6	$+$	4	$= 2$
$+1$	$-$	$+1$	$+$	0	$= +0$

Convex hull

Euler relation

Polytope boundary

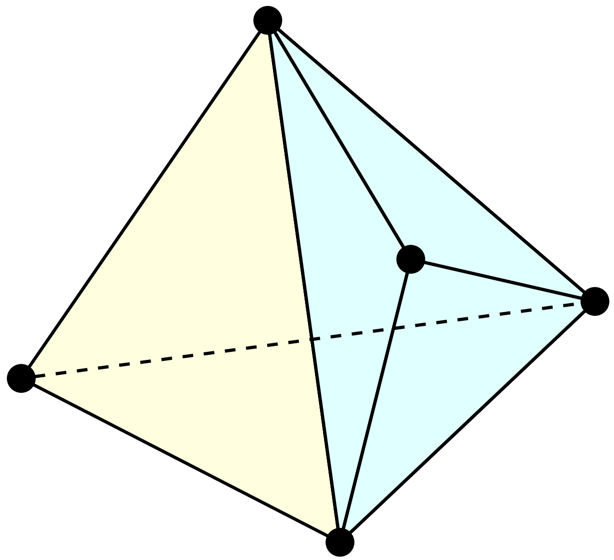


Vertices		Edges		Faces	
8	$-$	12	$+$	6	$= 2$
4	$-$	6	$+$	4	$= 2$
$+1$	$-$	$+1$	$+$	0	$= +0$
0	$-$	$+1$	$+$	$+1$	$= +0$

Convex hull

Euler relation

Polytope boundary



Vertices		Edges		Faces	
8	$-$	12	$+$	6	$=2$
4	$-$	6	$+$	4	$=2$
$+1$	$-$	$+1$	$+$	0	$=+0$
0	$-$	$+1$	$+$	$+1$	$=+0$

Convex hull

Euler relation

Polytope boundary

Vertices Edges Faces

$$n \quad - \quad e \quad + \quad f \quad = 2$$

Convex hull

Euler relation

Polytope boundary

Vertices		Edges		Faces	
n	$-$	e	$+$	f	$= 2$

triangular faces $3f = 2e$

Convex hull

Euler relation

Polytope boundary

Vertices Edges Faces

$$n \quad - \quad e \quad + \quad f \qquad = 2$$

triangular faces $3f = 2e$

$$f = 2n - 4$$

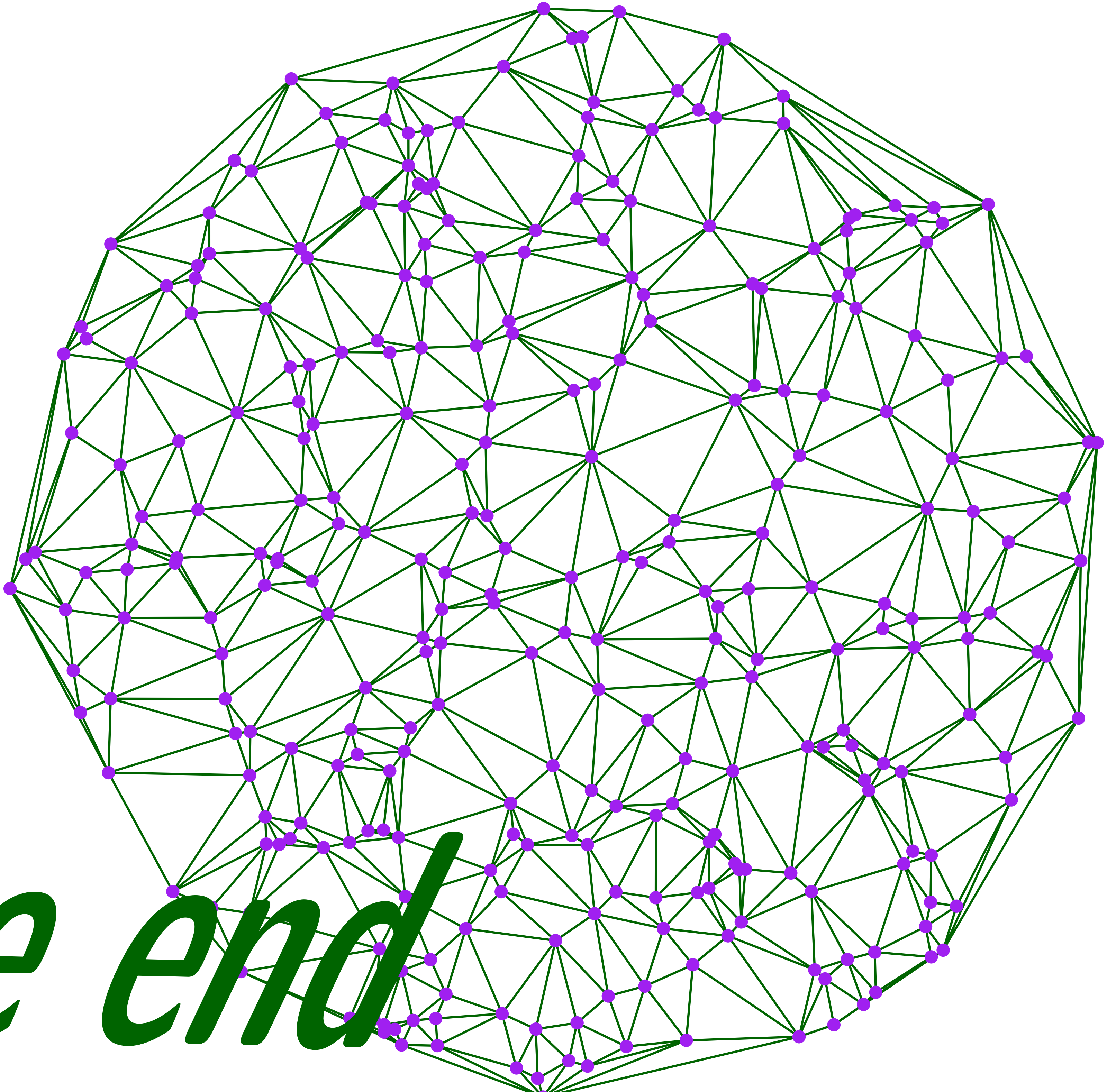
$$e = 3n - 6$$

Convex hull

Linear size

$O(n \log n)$ divide and conquer algorithm

$O(nh)$ gift wraping algorithm



The end