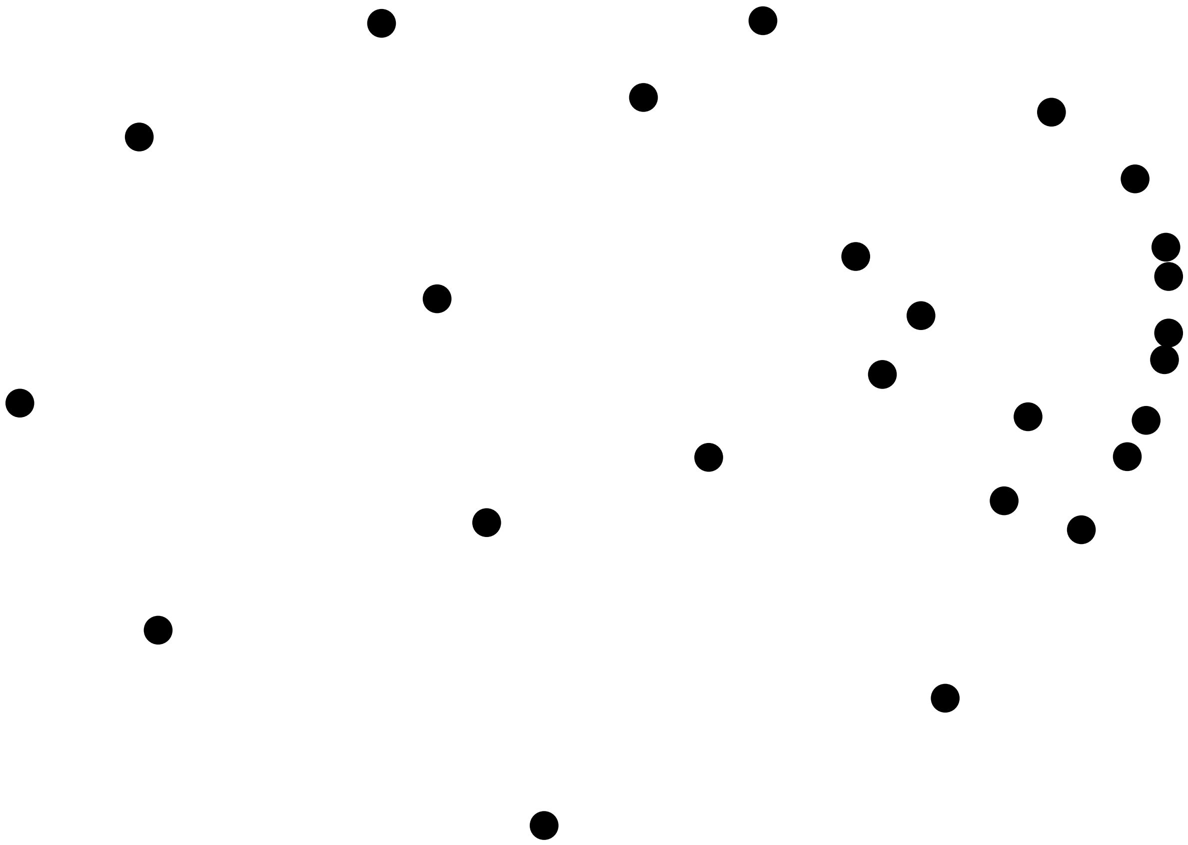
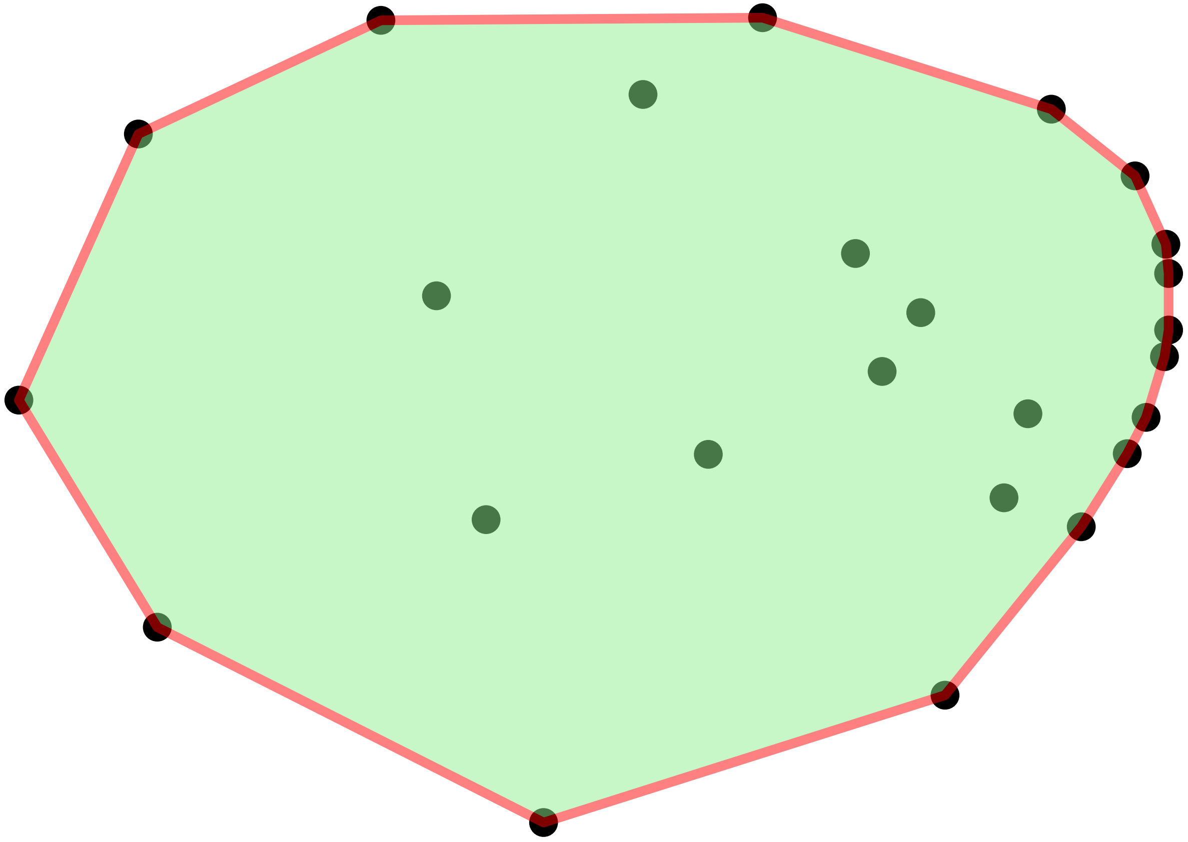
Convex hull

Convex hull



Convex hull



Convex hull

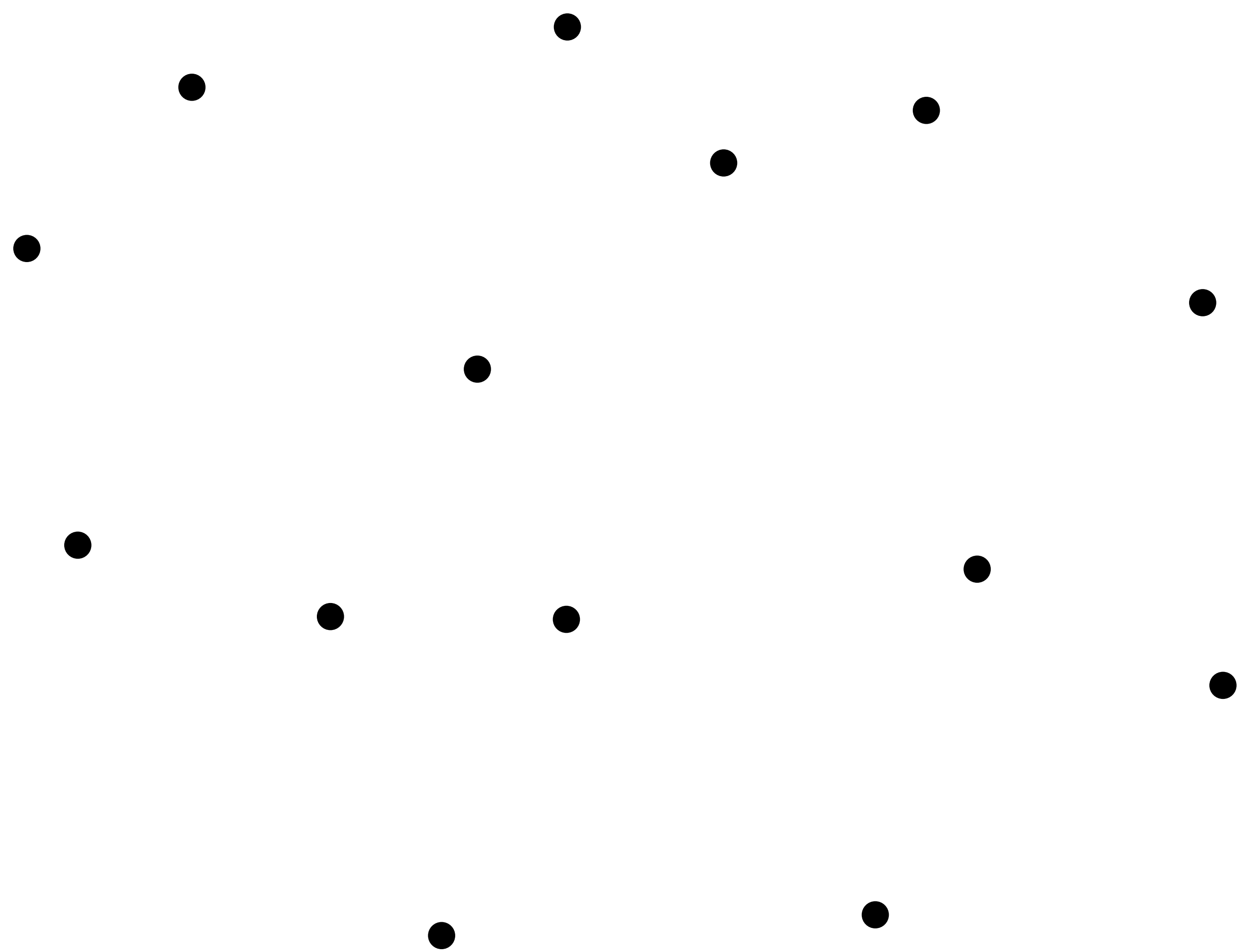
- Definition, extremal point
- Jarvis algorithm
- Orientation predicate
- Buggy degenerate example
- Real RAM model and general position hypothesis
- Graham algorithm
- Lower bound
- Other results
- Higher dimensions
- Another lower bound
- A simple randomized algorithm for linear programming

Convex hull

Convex hull

Definition, extremal point

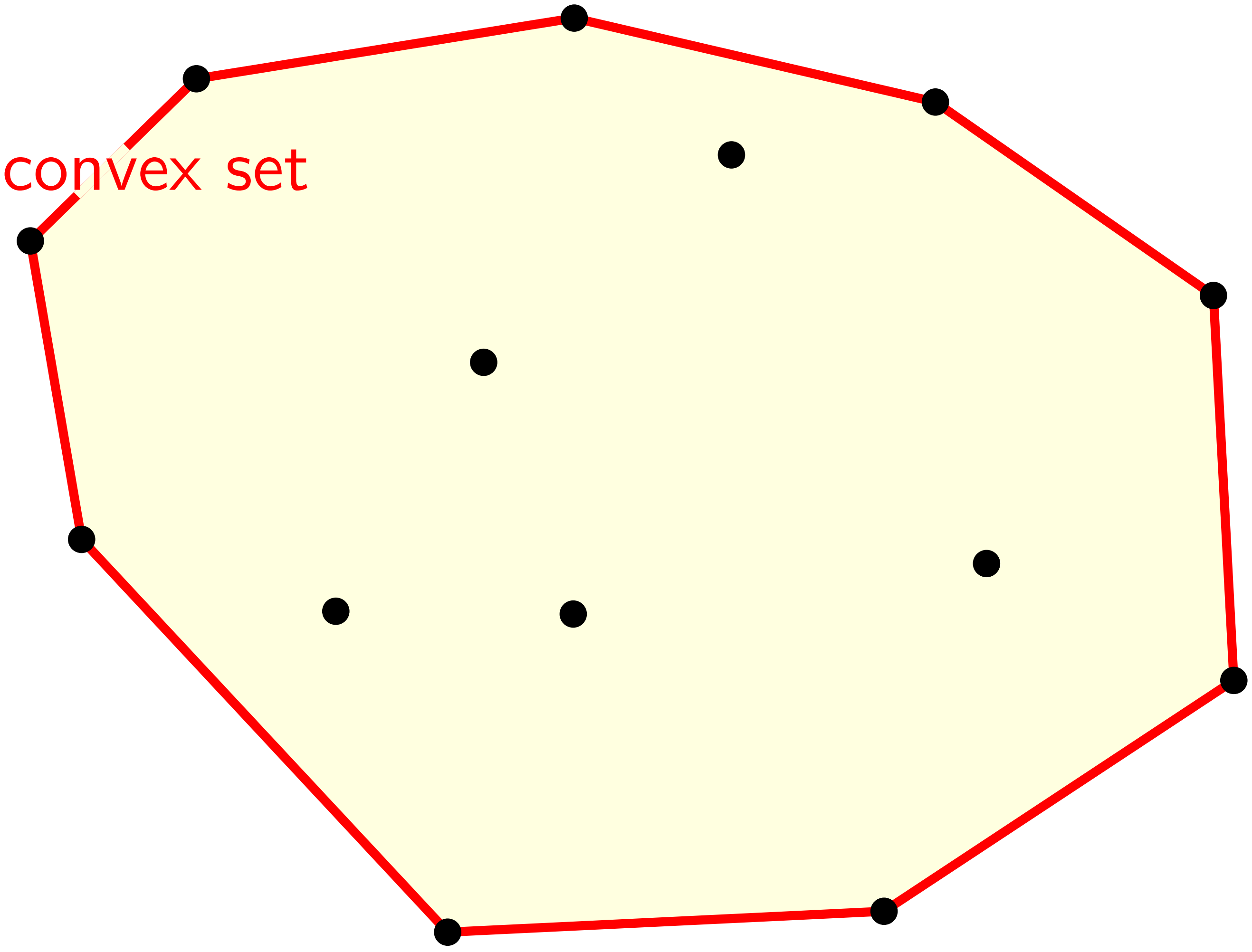
Set of points



Convex hull

Set of points

Smallest enclosing convex set

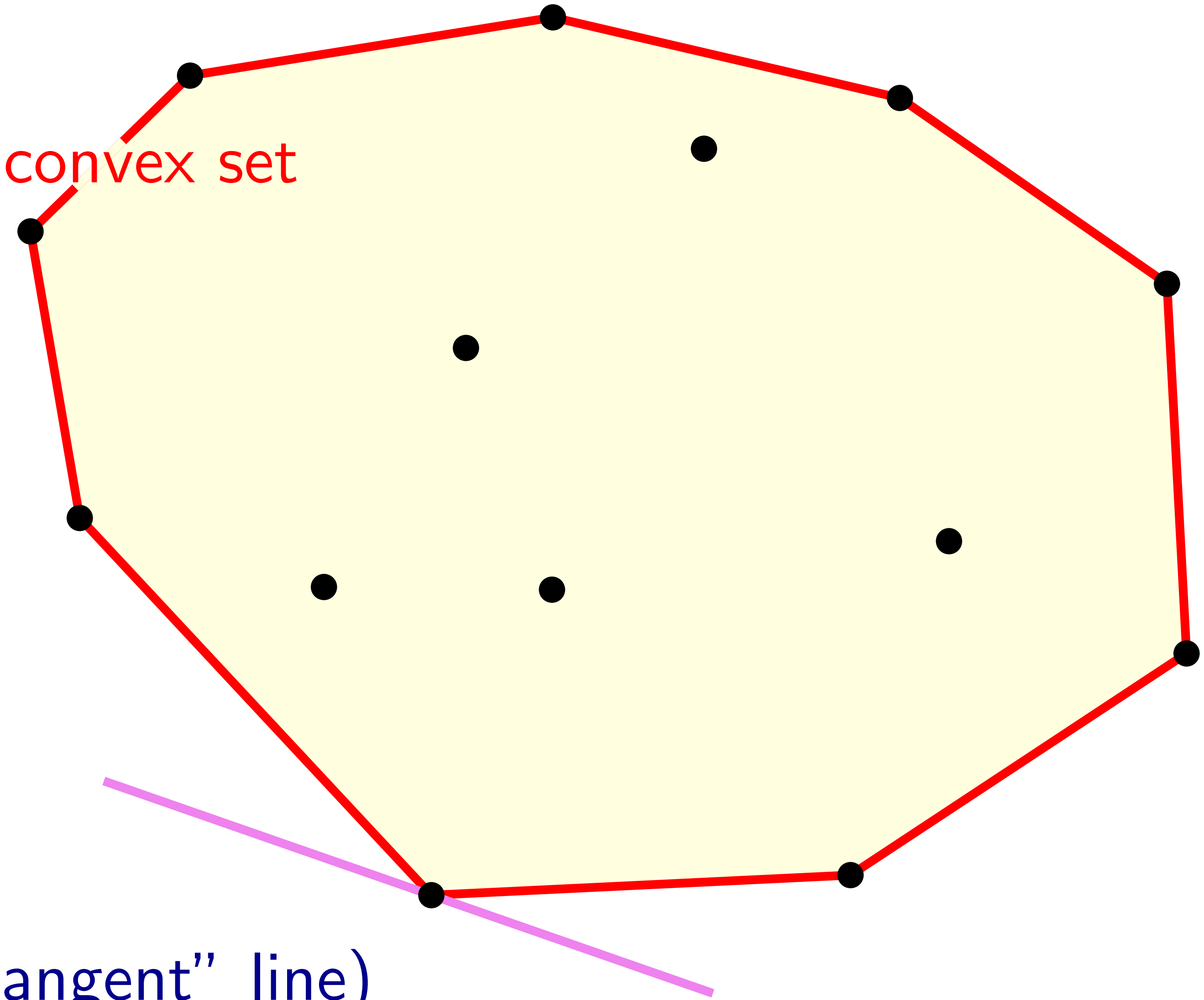


Convex hull

Definition, extremal point

Set of points

Smallest enclosing convex set

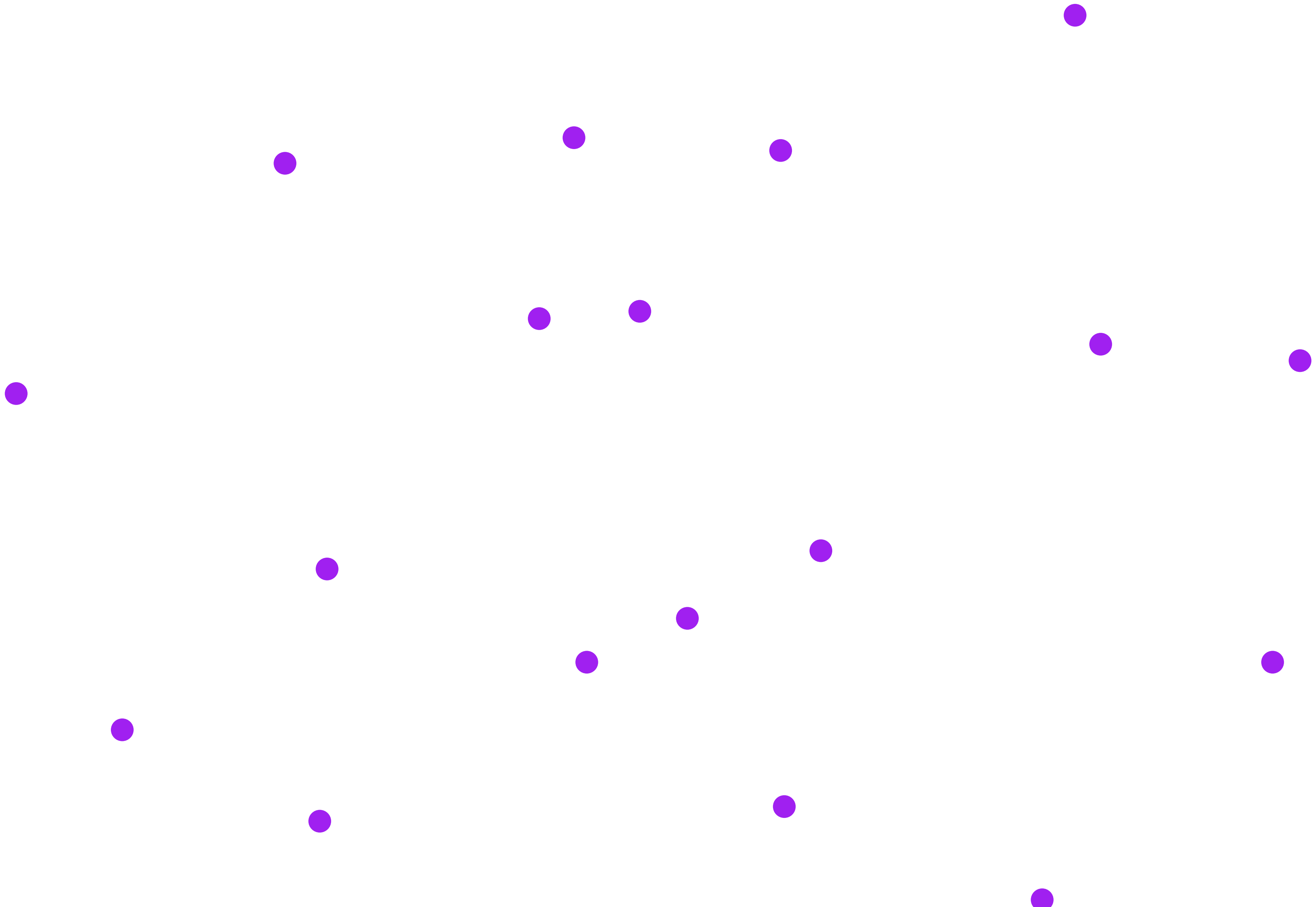


Extremal point

Supporting line ("tangent" line)

Convex hull

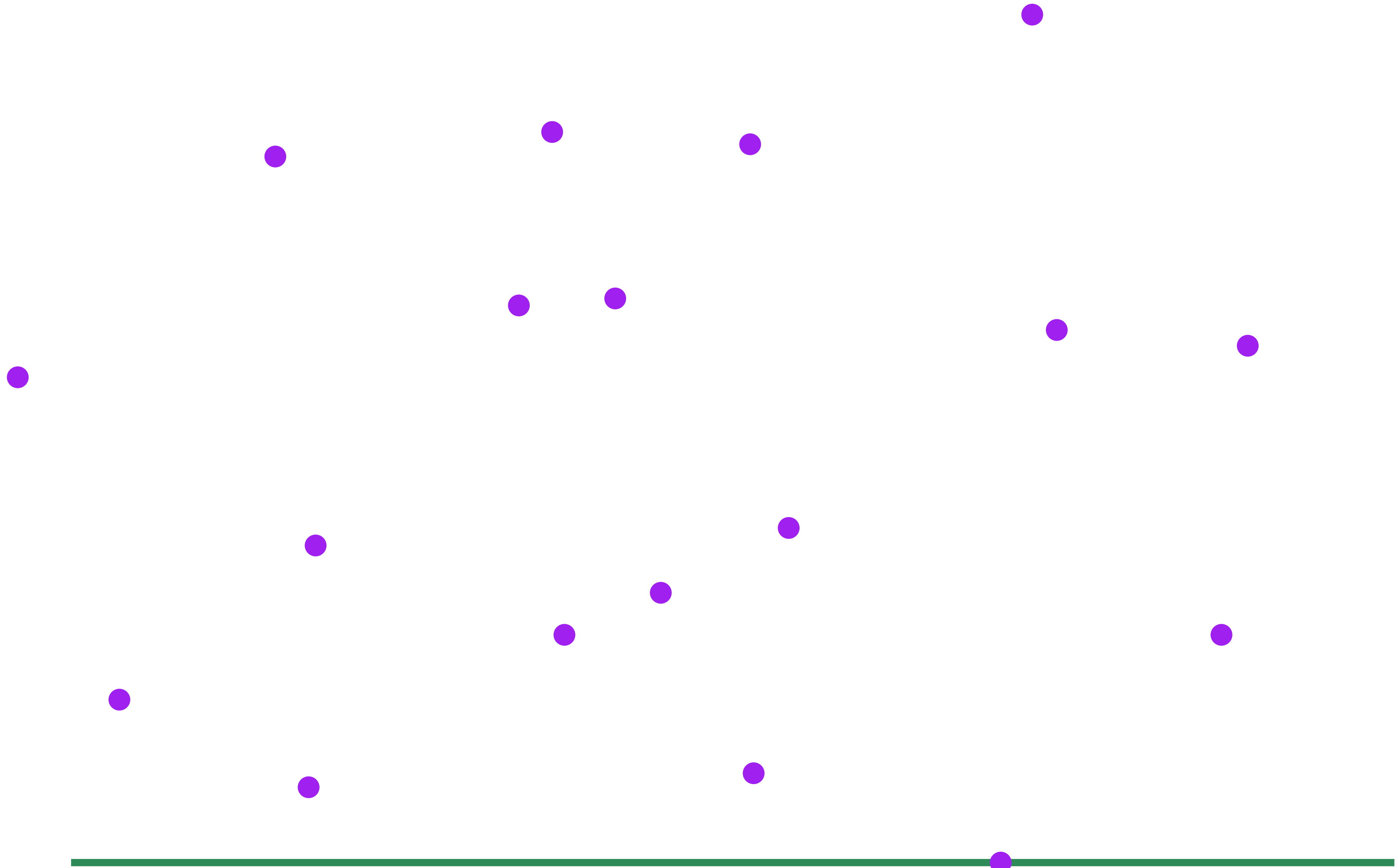
Jarvis algorithm



Convex hull

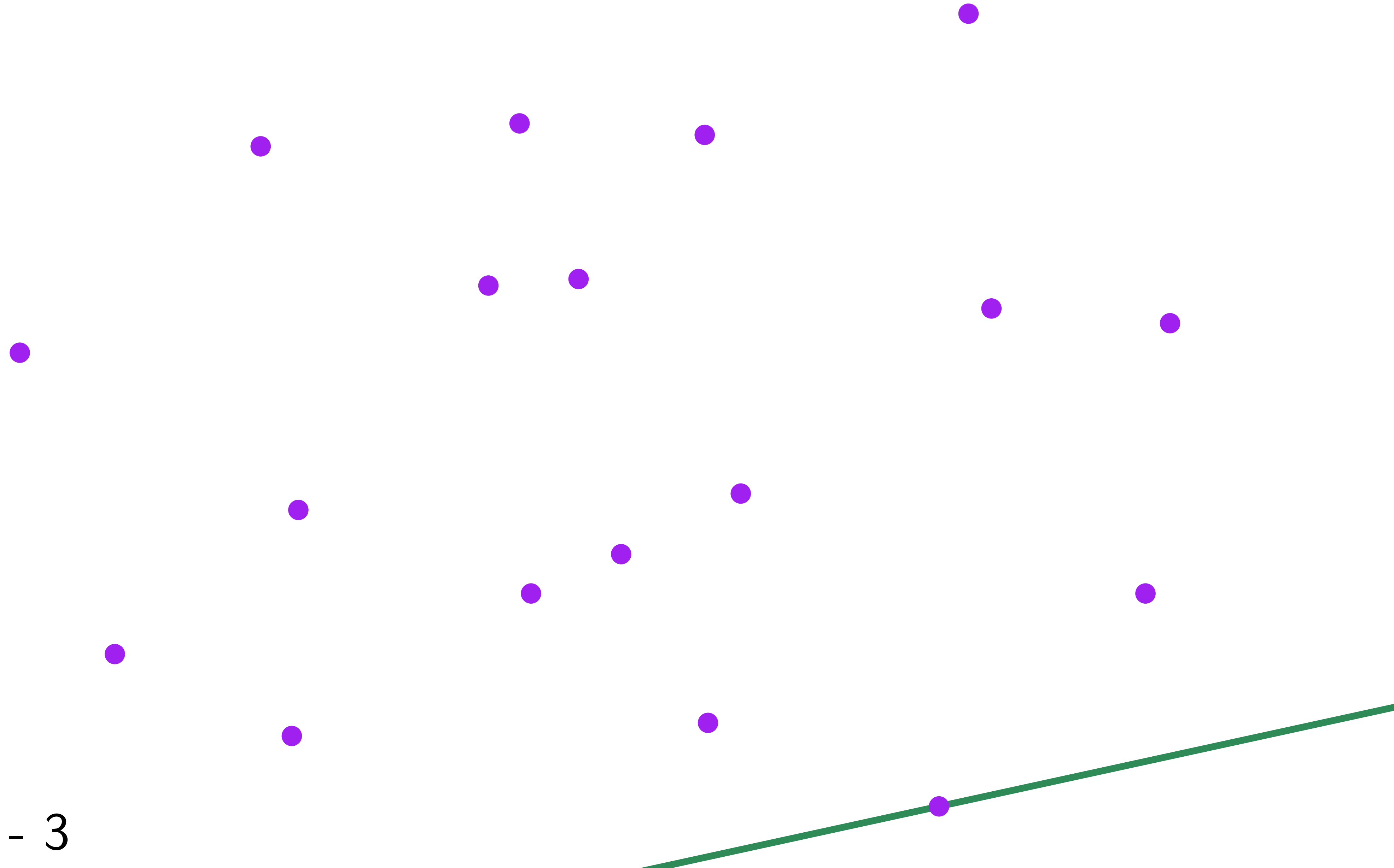
Jarvis algorithm

lowest point is extremal



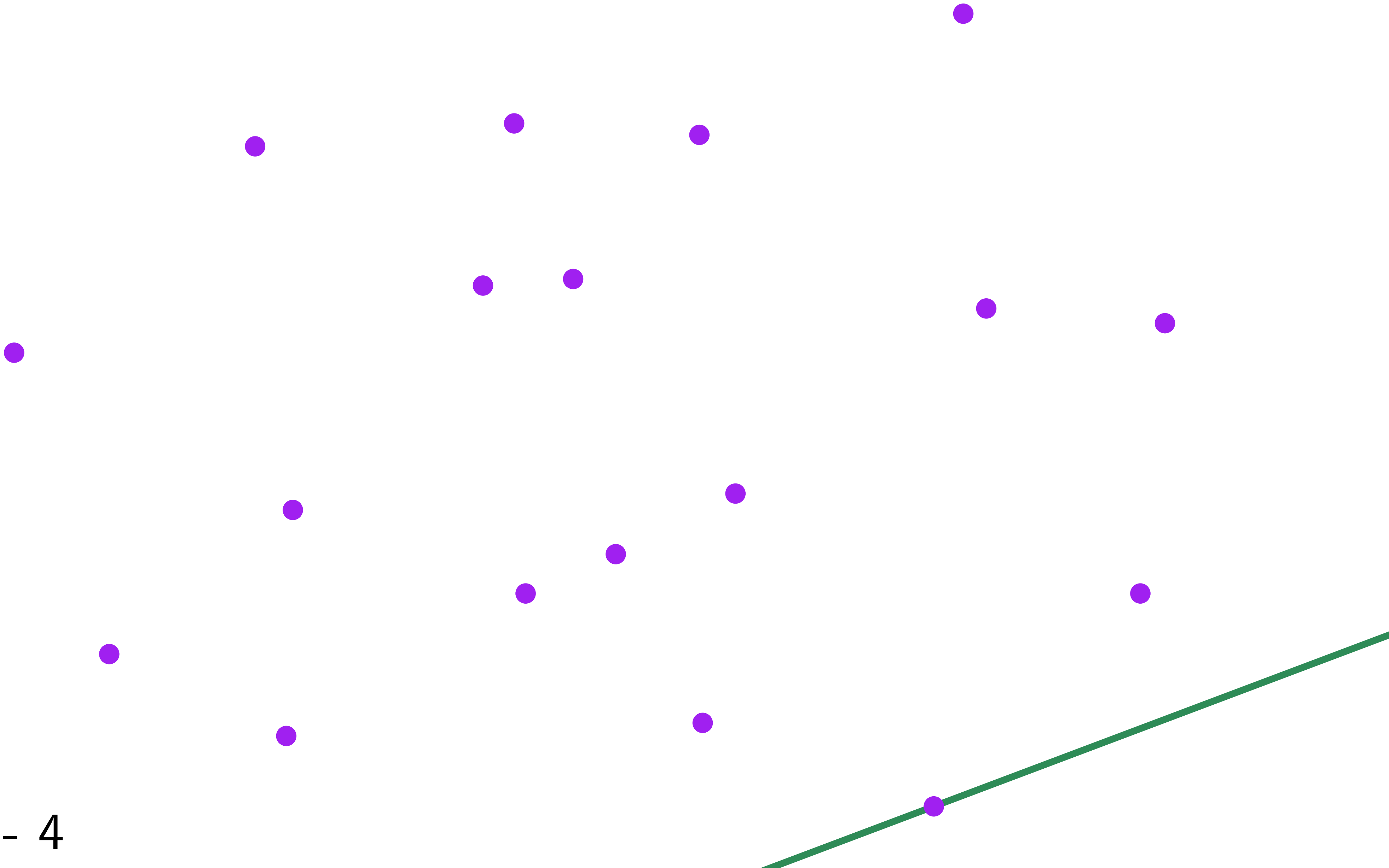
Convex hull

rotate



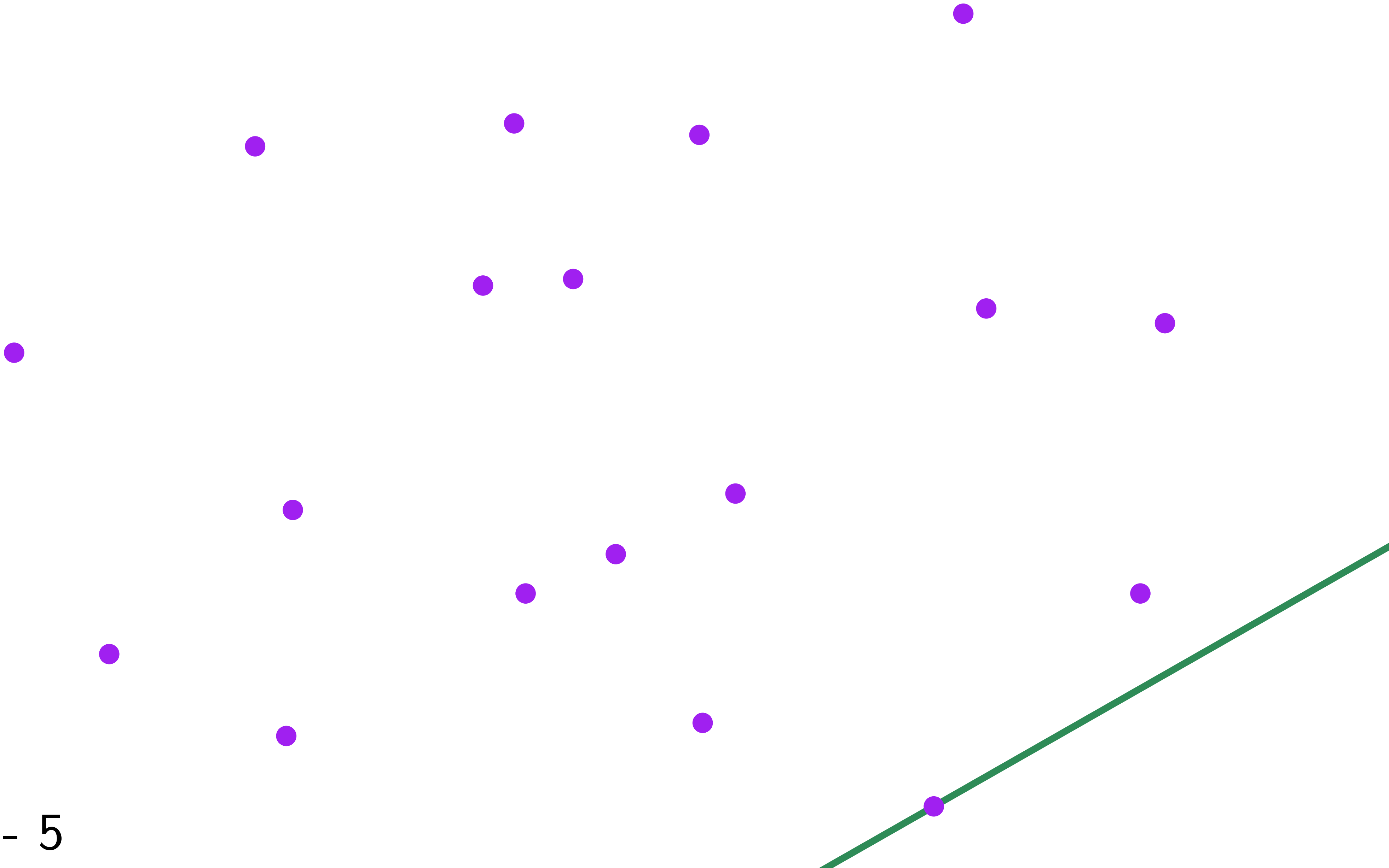
Convex hull

rotate



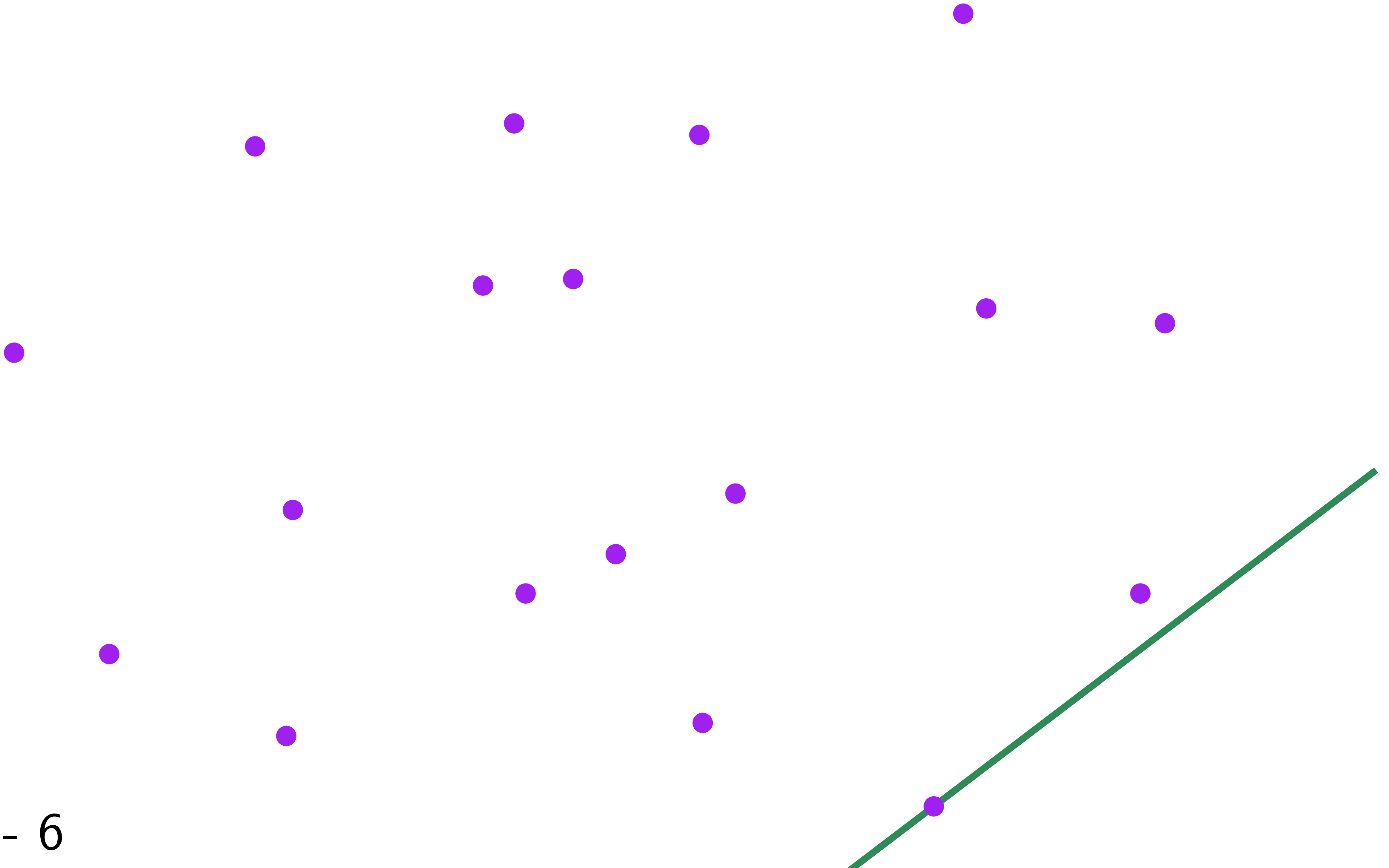
Convex hull

rotate



Convex hull

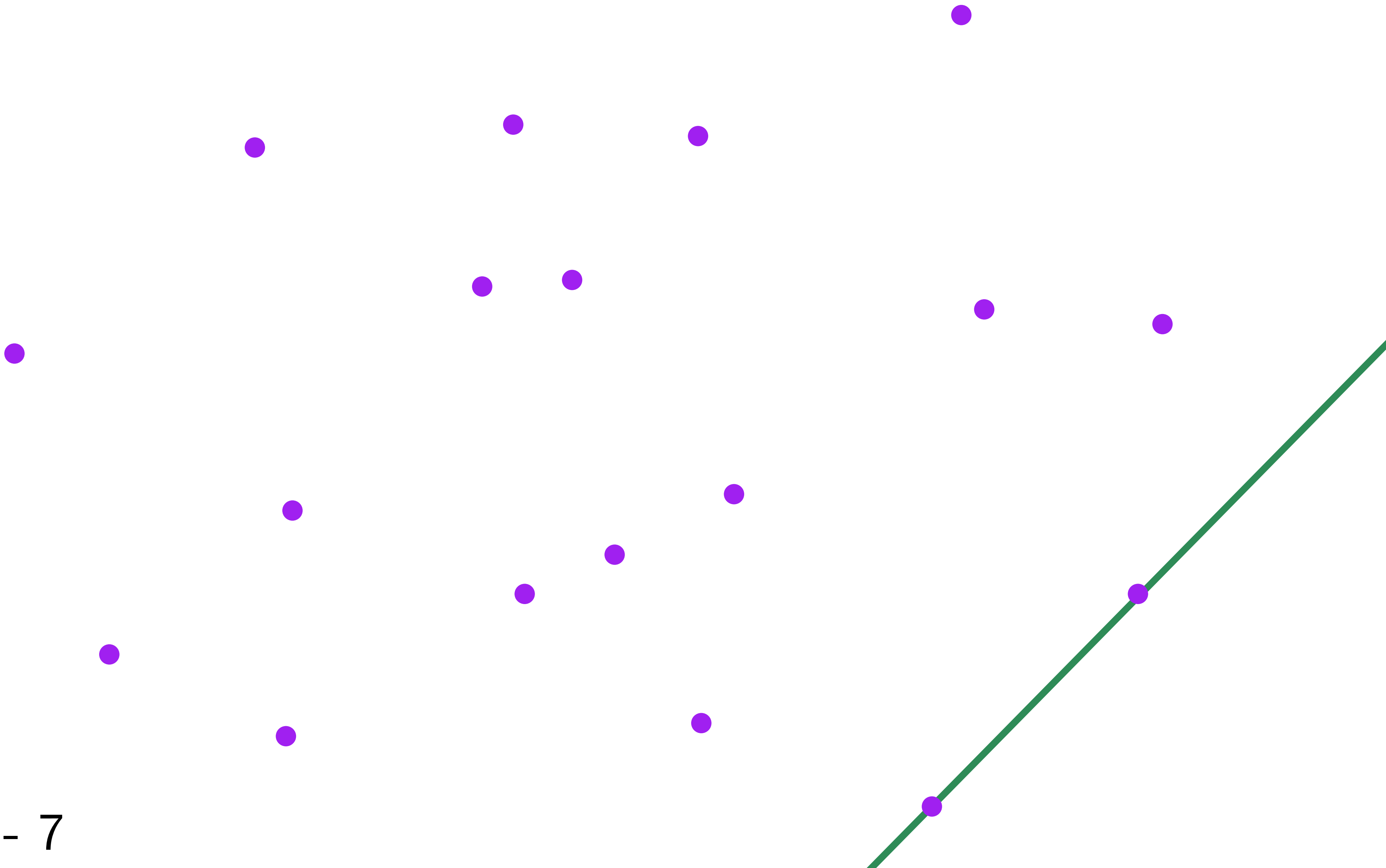
rotate



Convex hull

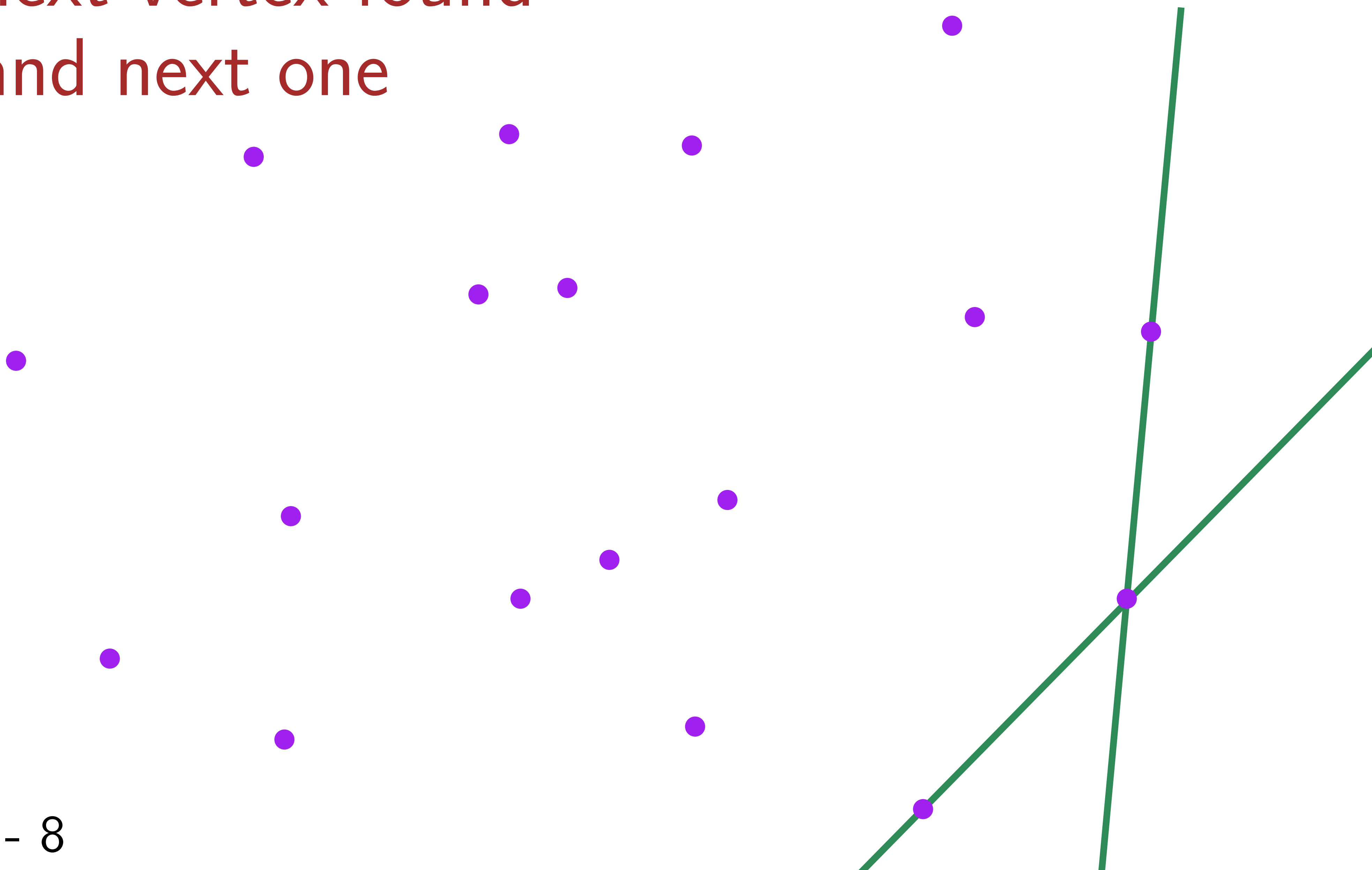
Jarvis algorithm

next vertex found



Convex hull

next vertex found
and next one



Convex hull

Jarvis algorithm

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

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

    $min = \infty$

    For each $w \in S$

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

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

    $v = v.next$;

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

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

    $min = \infty$

    For each $w \in S$

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

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

    $v = v.next$;

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

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

$min = \infty$

For each $w \in S$

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

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

$v = v.next$;

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

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

$min = \infty$

For each $w \in S$

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

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

$v = v.next$;

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

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

  $min = \infty$

  For each $w \in S$

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

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

  $v = v.next$;

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

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

$min = \infty$

For each $w \in S$

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

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

$v = v.next$;

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

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

    $min = \infty$

    For each $w \in S$

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

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

    $v = v.next$;

While $v \neq u$

$O(n^2)$

Convex hull

Complexity ?

Jarvis algorithm

Input : point set S

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

For each $w \in S \setminus \{u\}$ $O(n^2)$

if $angle(ux, uw) < min$

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

$u.next = v$; $O(nh)$

Do

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

$min = \infty$

For each $w \in S$

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

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

$v = v.next$;

While $v \neq u$

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 $w \in S$

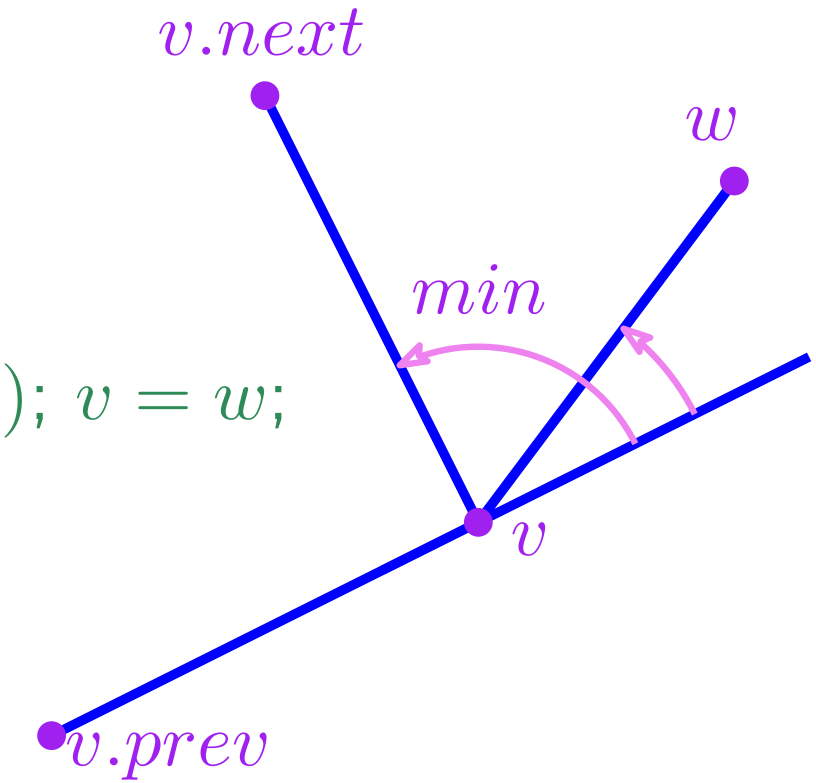
    $min = \infty$

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

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

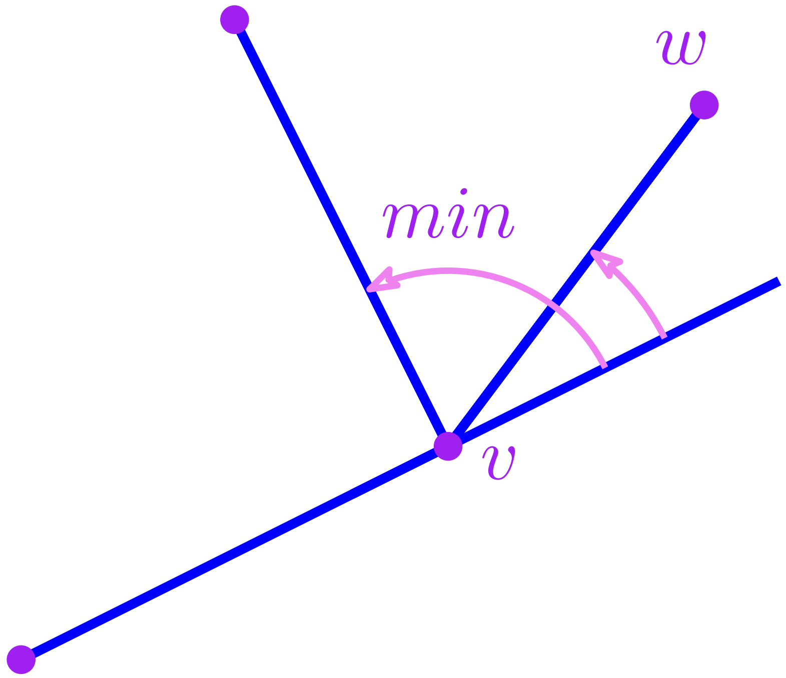
  $v = v.next$;

While $v \neq u$



Convex hull

if $angle(pv, vw) < min$

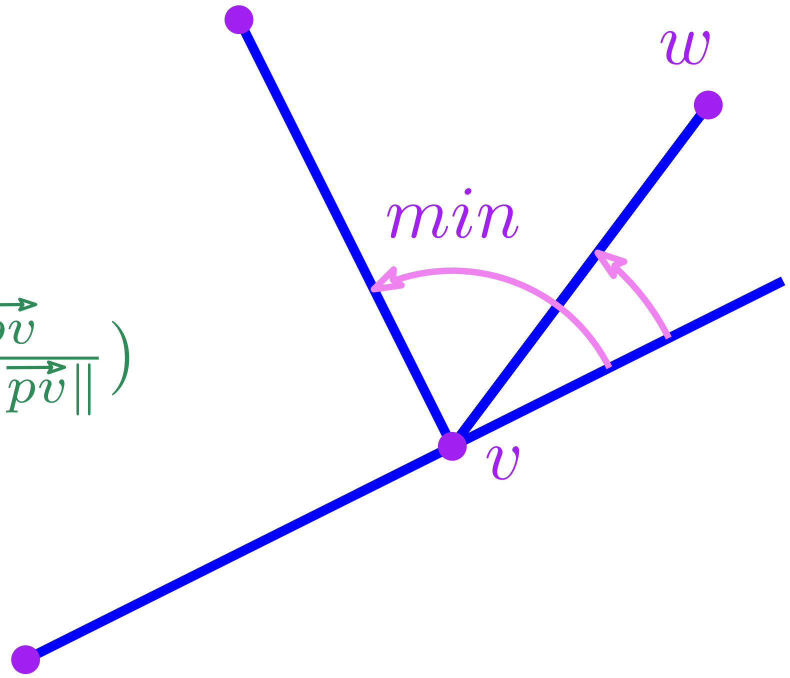


Convex hull

Orientation predicate

if $angle(pv, vw) < min$

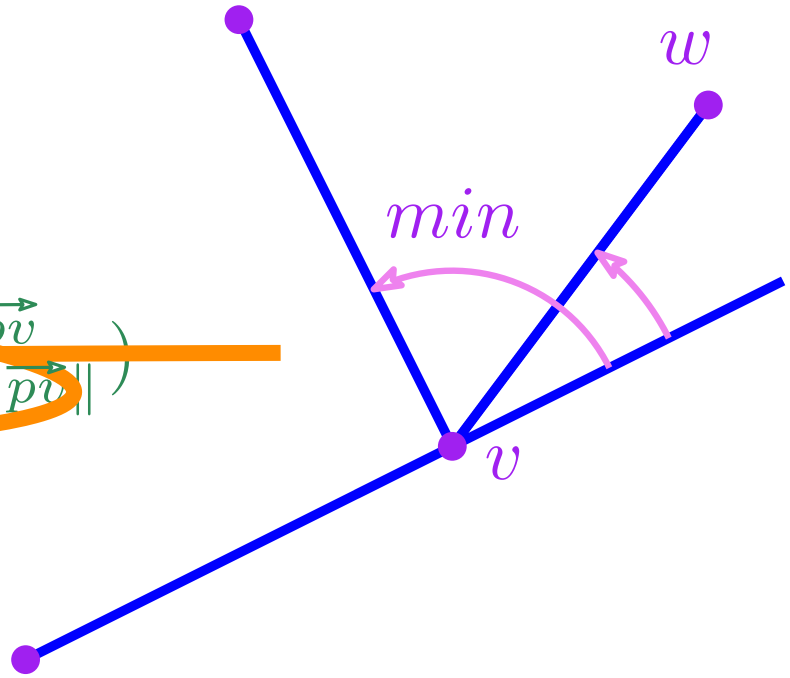
$$angle(pv, vw) = \arccos(\frac{\overrightarrow{vw} \cdot \overrightarrow{pv}}{\|\overrightarrow{vw}\| \cdot \|\overrightarrow{pv}\|})$$



Convex hull

if $angle(pv, vw) < min$

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



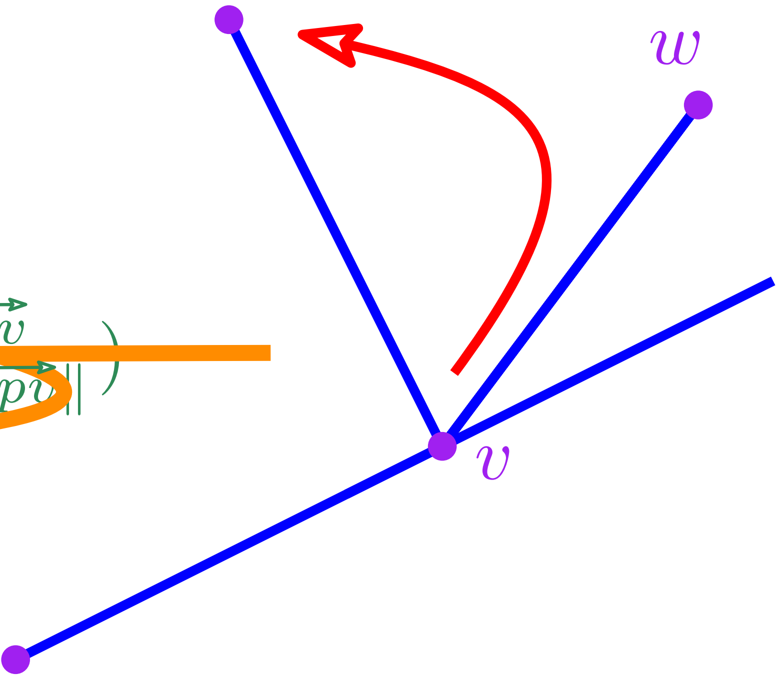
Convex hull

Orientation predicate

if $angle(pv, vw) < min$

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

if vwn turn left

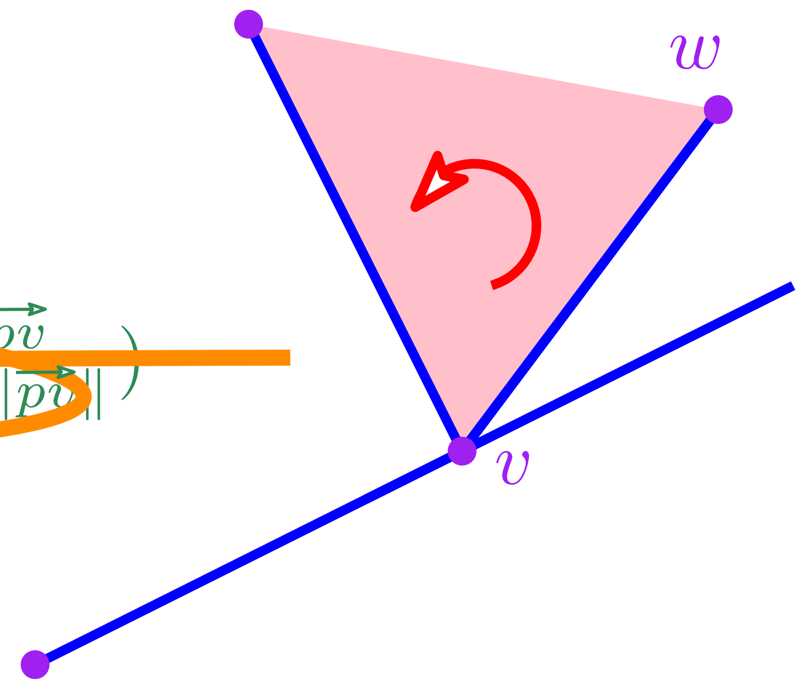


Convex hull

Orientation predicate

if $angle(pv, vw) < min$

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



if vwn turn left

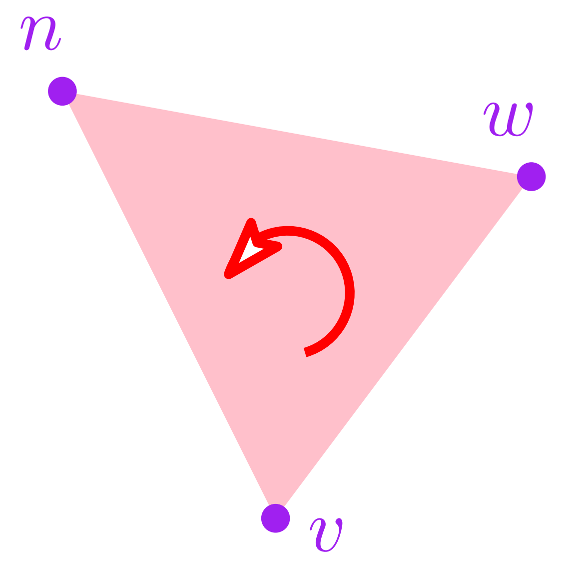
if triangle vwn counterclockwise (ccw)

if triangle vwn positively oriented

Convex hull

Orientation predicate

$vwn + ?$

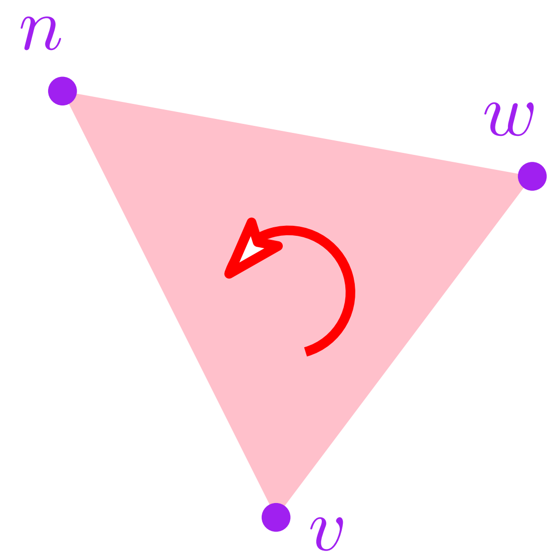


Convex hull

Orientation predicate

$vwn + ?$

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

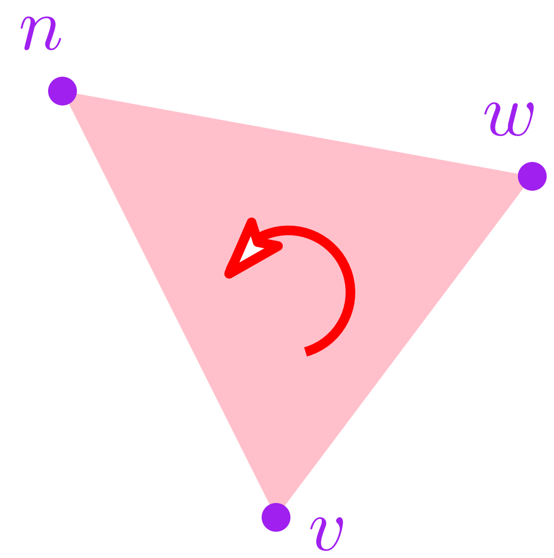


Convex hull

Orientation predicate

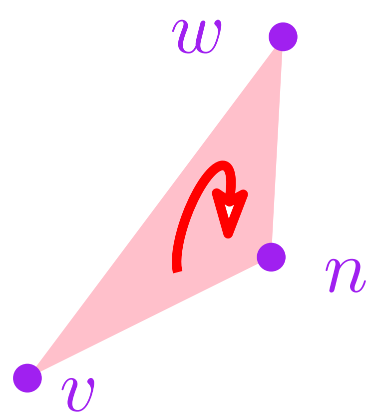
vwn + ?

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



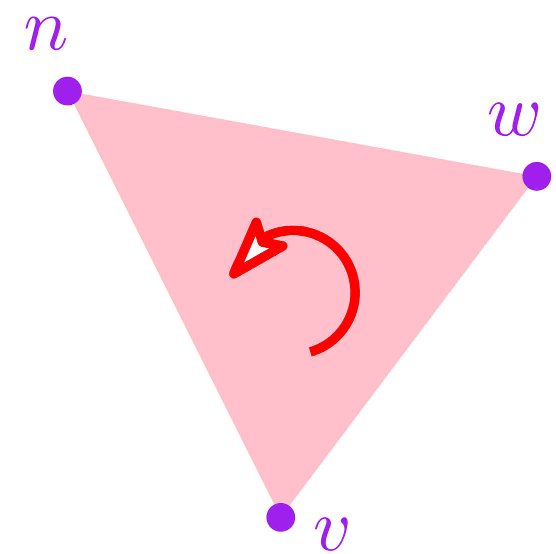
vwn - ?

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



Convex hull

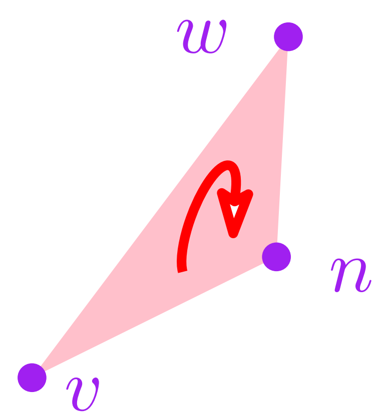
Orientation predicate



vwn + ?

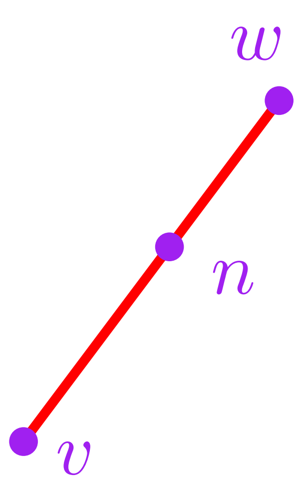
$$\begin{vmatrix} x_w - x_v & x_n - x_v \\ y_w - y_v & y_n - y_v \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 \\ x_v & x_w & x_n \\ y_v & y_w & y_n \end{vmatrix} > 0$$

vwn - ?



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

vwn 0 ?



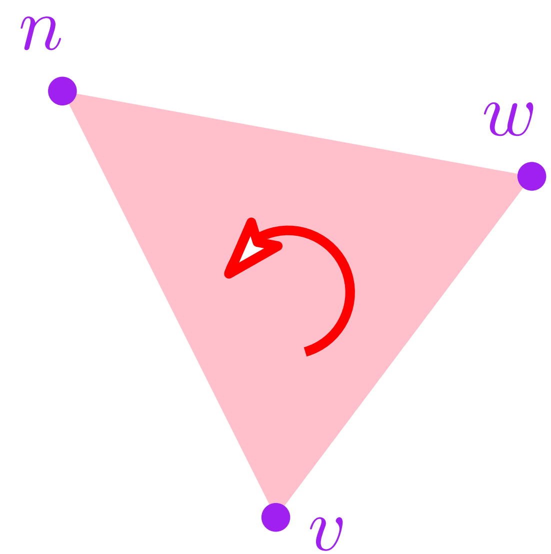
$$\begin{vmatrix} 1 & 1 & 1 \\ x_v & x_w & x_n \\ y_v & y_w & y_n \end{vmatrix} = 0$$

Convex hull

Orientation predicate

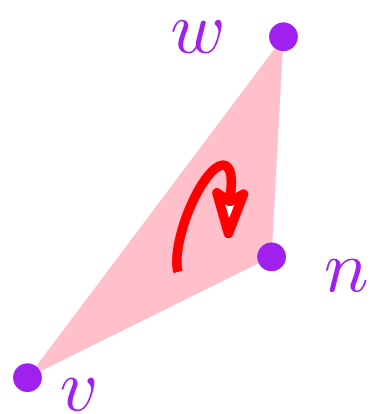
vwn + ?

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



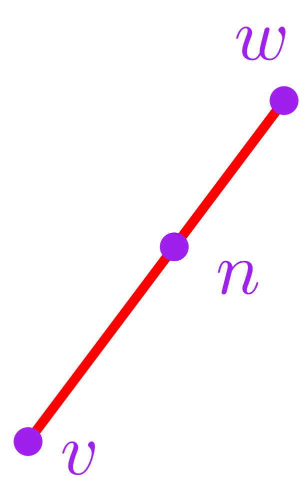
vwn - ?

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



vwn 0 ?

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



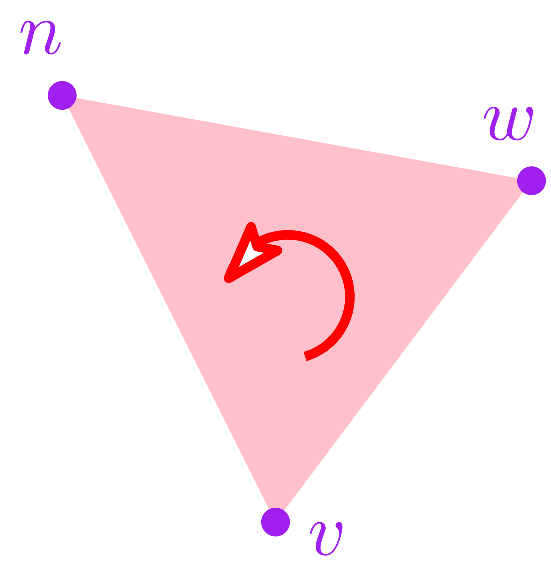
degenerate case

Convex hull

Orientation predicate

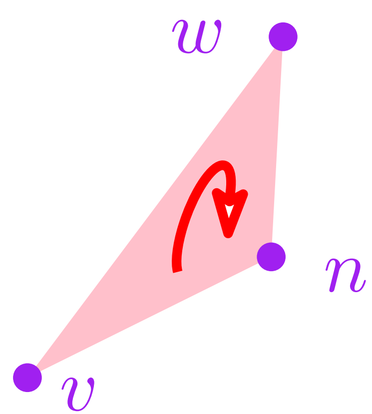
vwn + ?

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



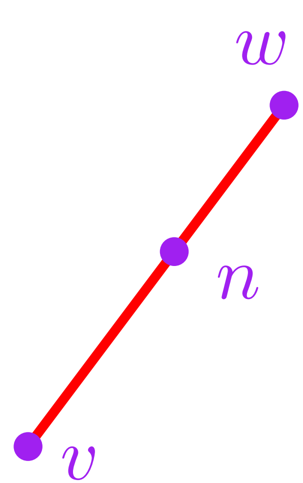
vwn - ?

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



vwn 0 ?

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



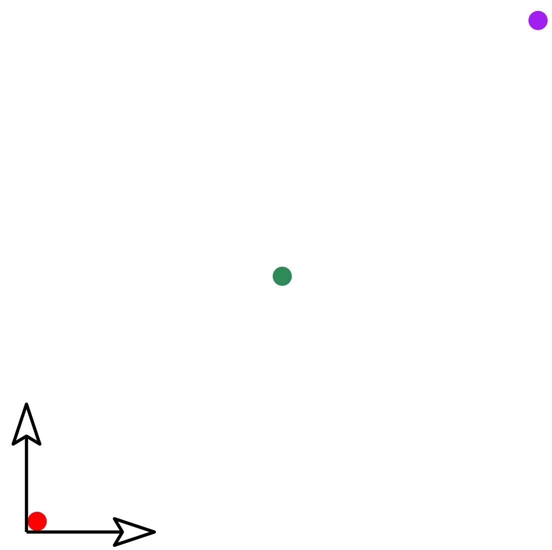
rounding errors

?

Convex hull

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

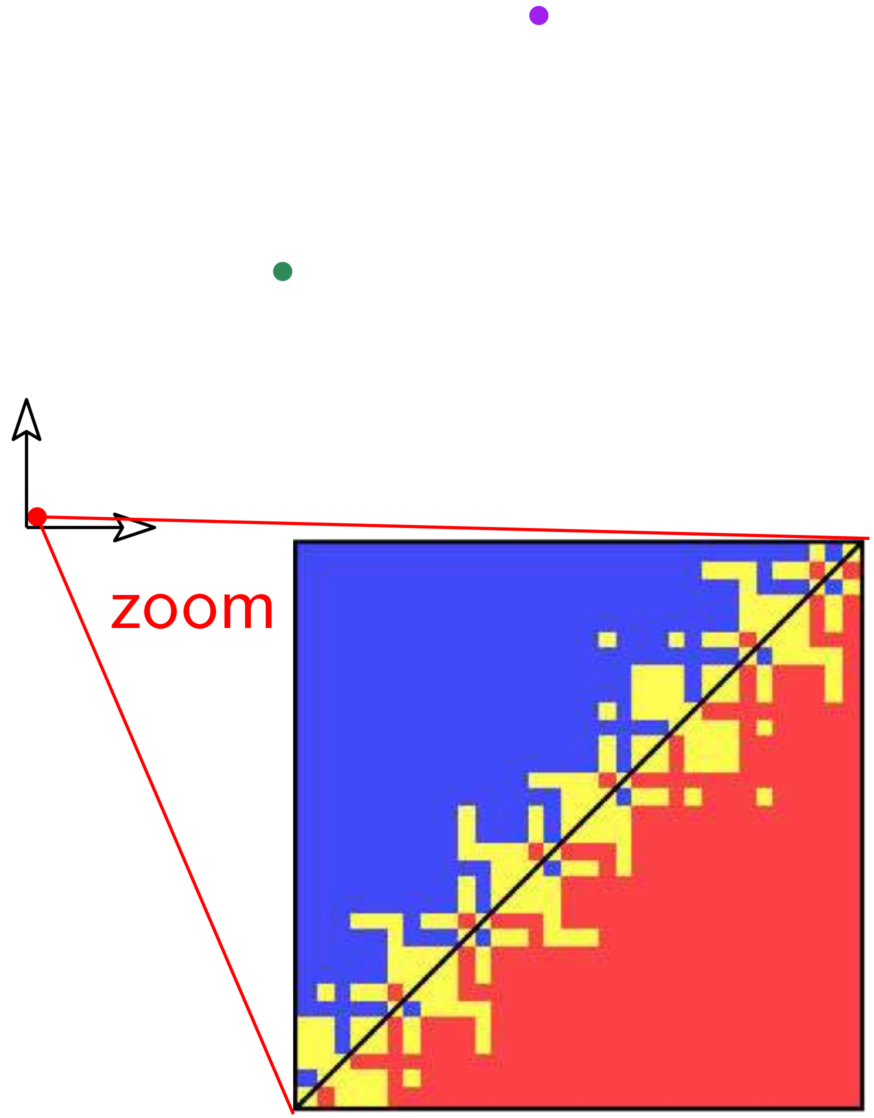
$p = (\frac{1}{2} + x.u, \frac{1}{2} + y.u)$

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

$q = (12, 12)$

$r = (24, 24)$

Teaser robustness lecture



$orientation(p, q, r)$

evaluated with double

≤ 0

0

≥ 0

Convex hull

Teaser robustness lecture

Buggy degenerate example
(single precision)

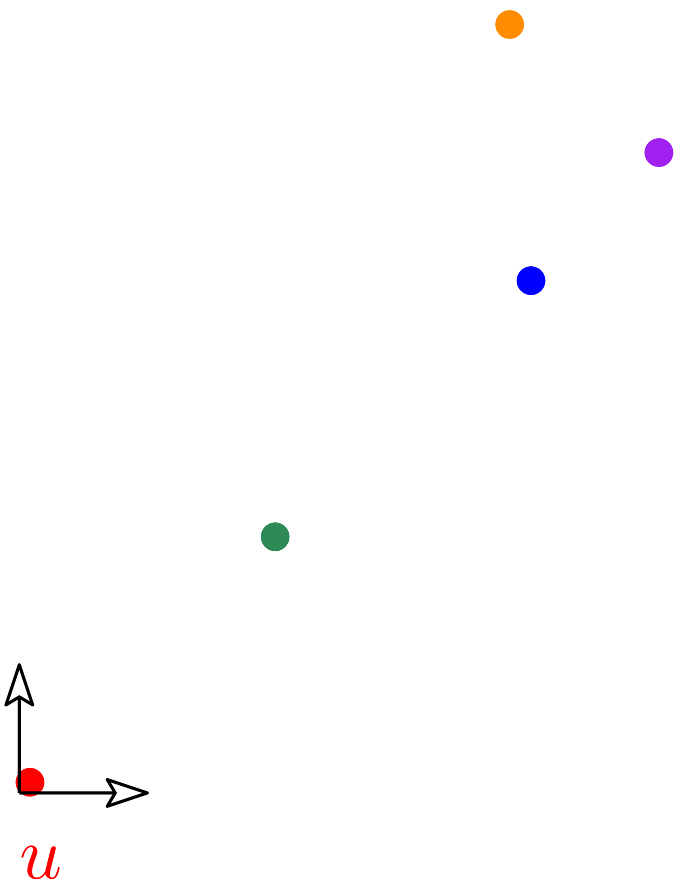
$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$



Convex hull

Teaser robustness lecture

Buggy degenerate example
(single precision)

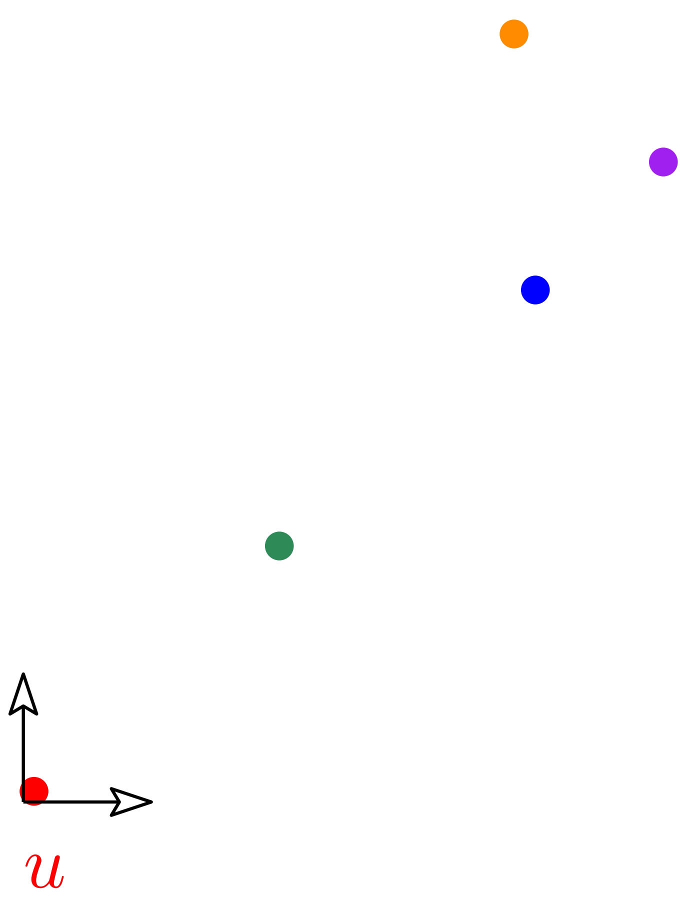
$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$



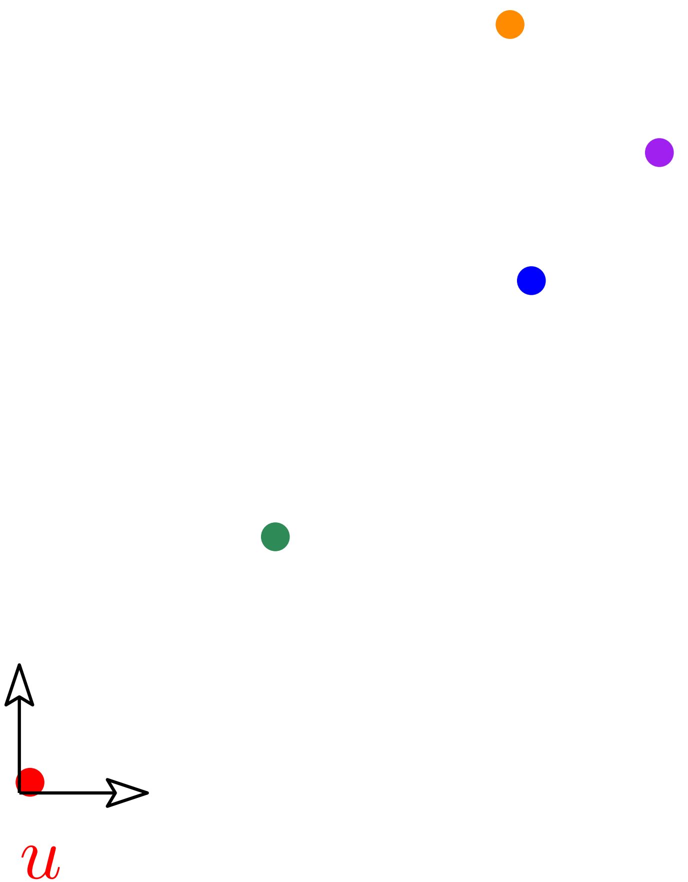
Input : point set S

$u = v =$ 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)

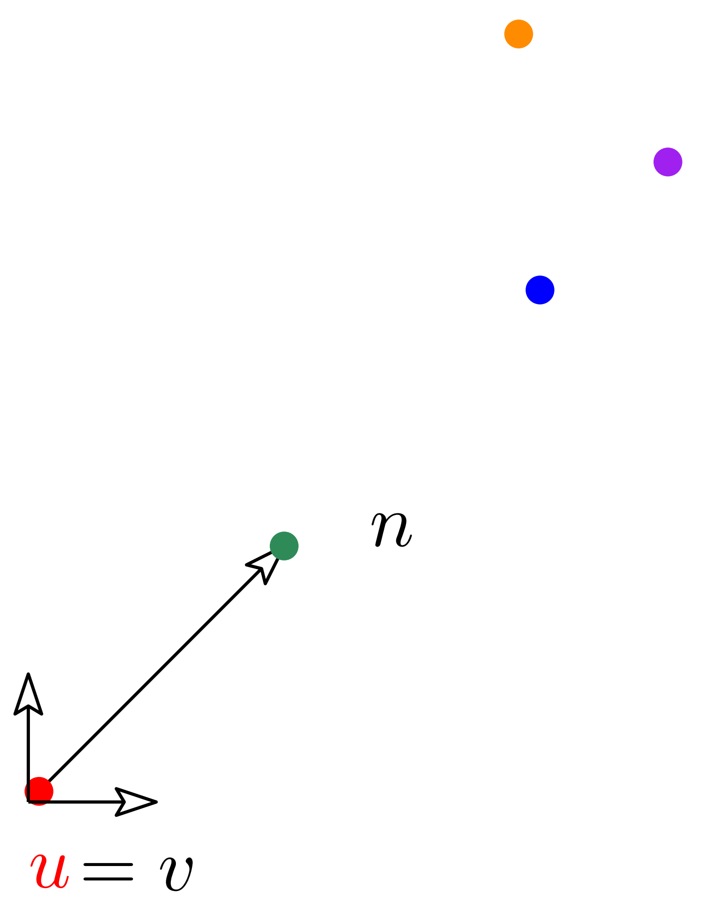
$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$



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)

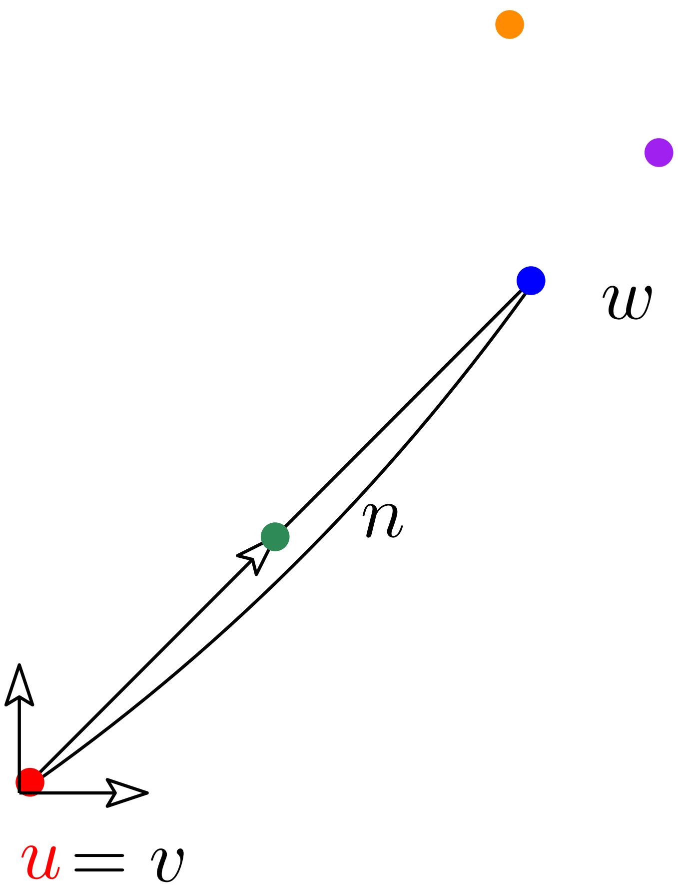
$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$



Do

$n =$ first in S;

For each $w \in S$

if vwn positive

then $n = w$;

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

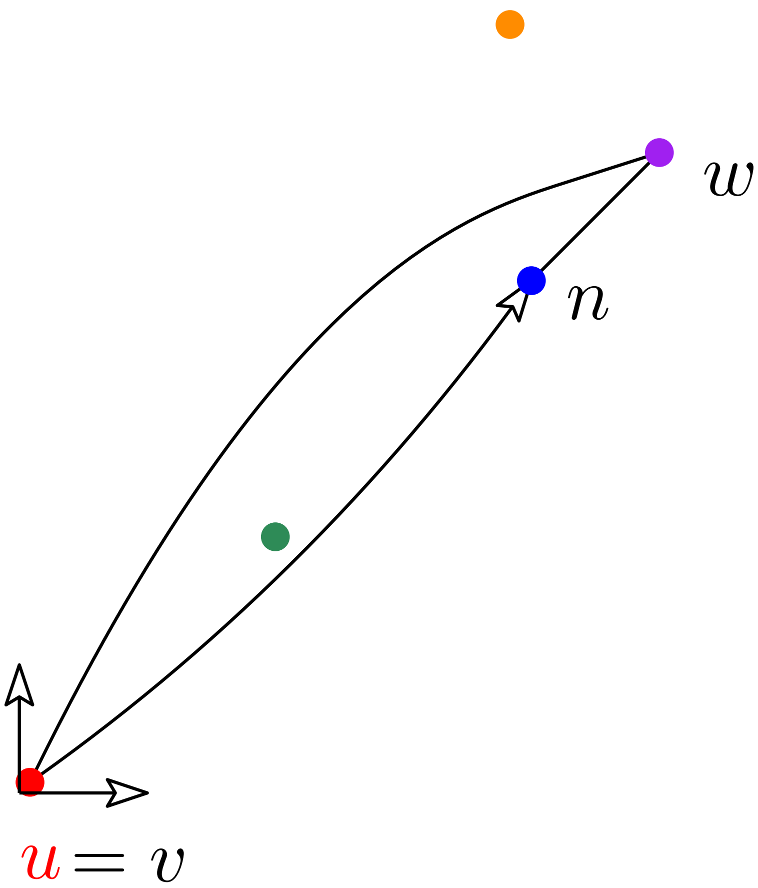
$S = S \setminus \{v\}$

While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example (single precision)



$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$

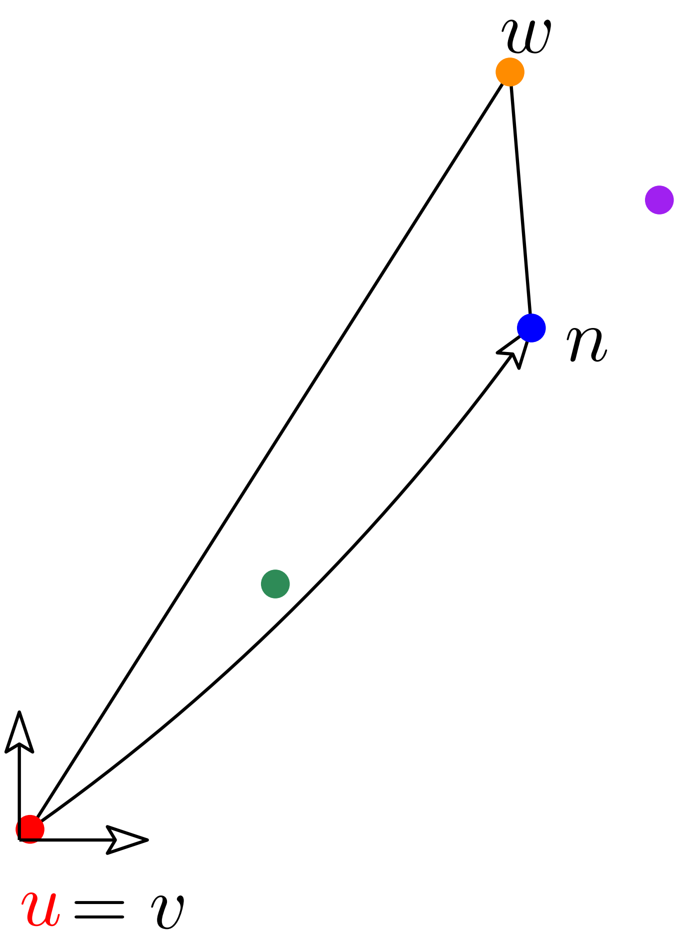
Do
- $n =$ first in S;
- For each $w \in S$
  - if vwn positive
    - then $n = w$;
- $v.next = n$; $v = n$;
- $S = S \setminus \{v\}$

While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example
(single precision)



$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

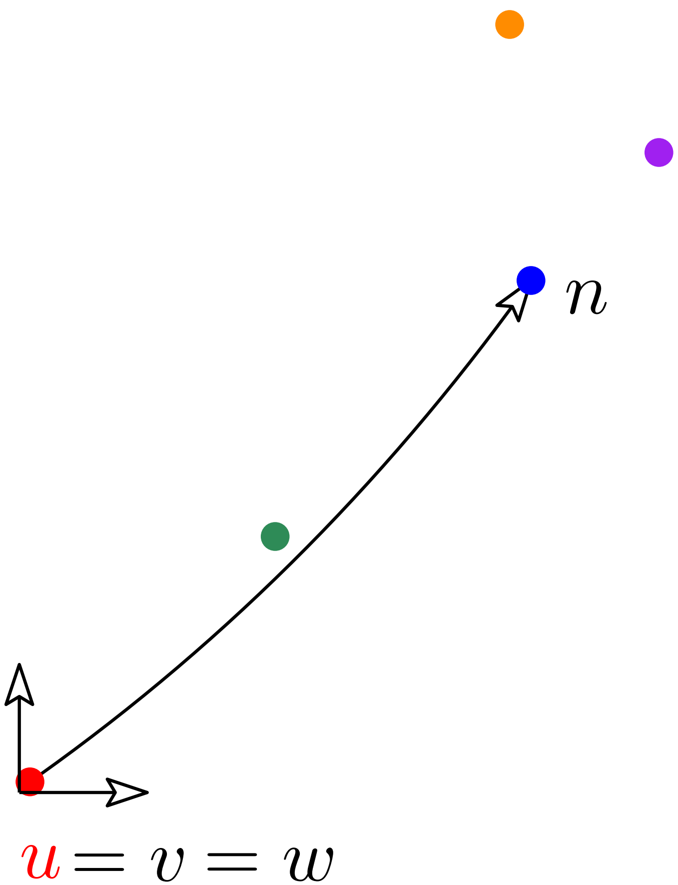
$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$

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)

$w_1 = (12, 12)$

$w_2 = (24, 24)$

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$

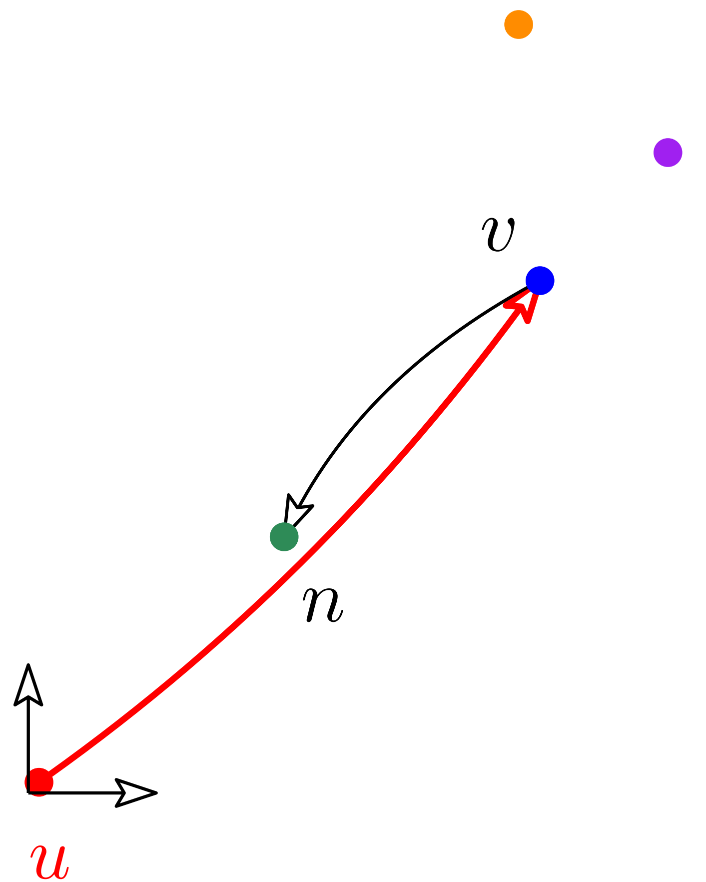
Do
- $n =$ first in S;
- For each $w \in S$
  - if vwn positive
    - then $n = w$;
- $v.next = n$; $v = n$;
- $S = S \setminus \{v\}$

While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example
(single precision)



$w_1 = (12, 12)$

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

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

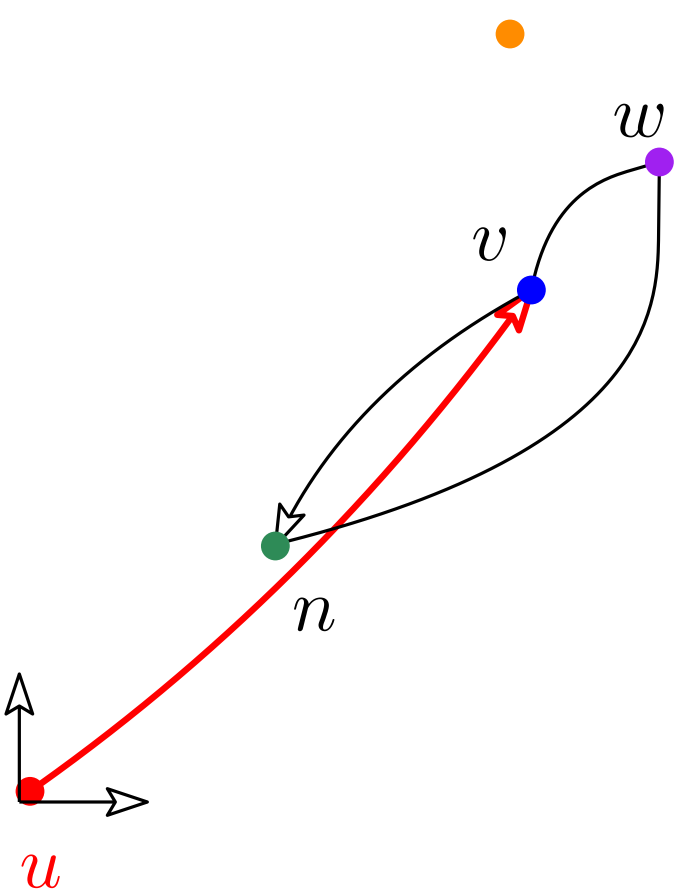
$w_5 = (0.5000029, 0.5000027)$

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)

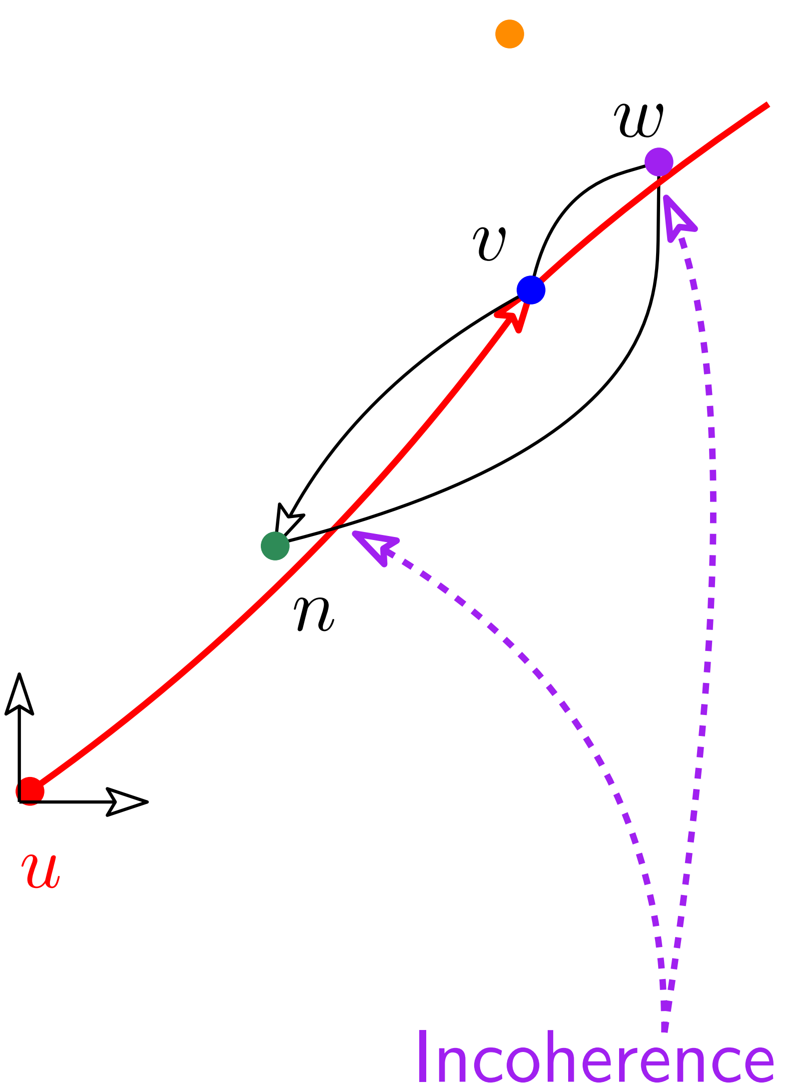
$w_1 = (12, 12)$

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

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$

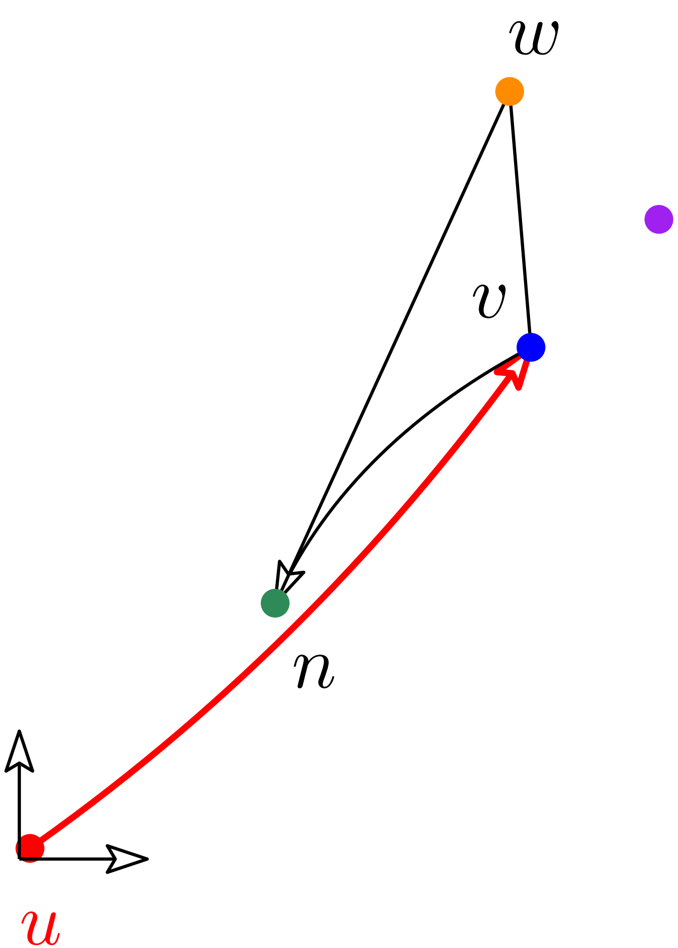


Do
- $n =$ first in S;
- For each $w \in S$
  - if vwn positive
    - then $n = w$;
- $v.next = n$; $v = n$;
- $S = S \setminus \{v\}$

While $v \neq u$

Convex hull

Teaser robustness lecture



Buggy degenerate example (single precision)

$w_1 = (12, 12)$

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

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

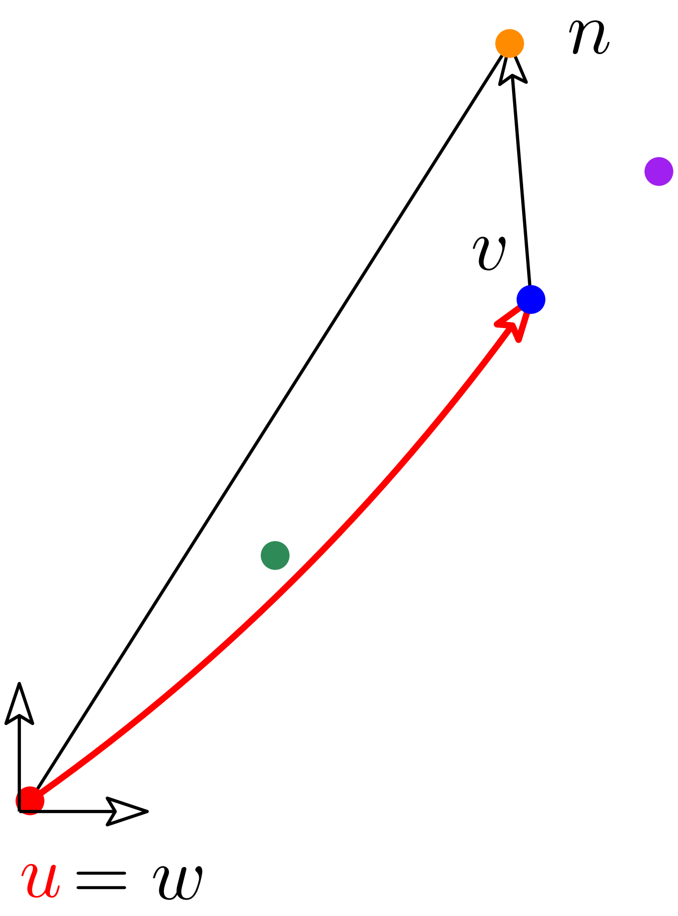
$w_5 = (0.5000029, 0.5000027)$

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)



$w_1 = (12, 12)$

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

$w_3 = (30, 30.000001)$

$w_4 = (23, 36)$

$w_5 = (0.5000029, 0.5000027)$

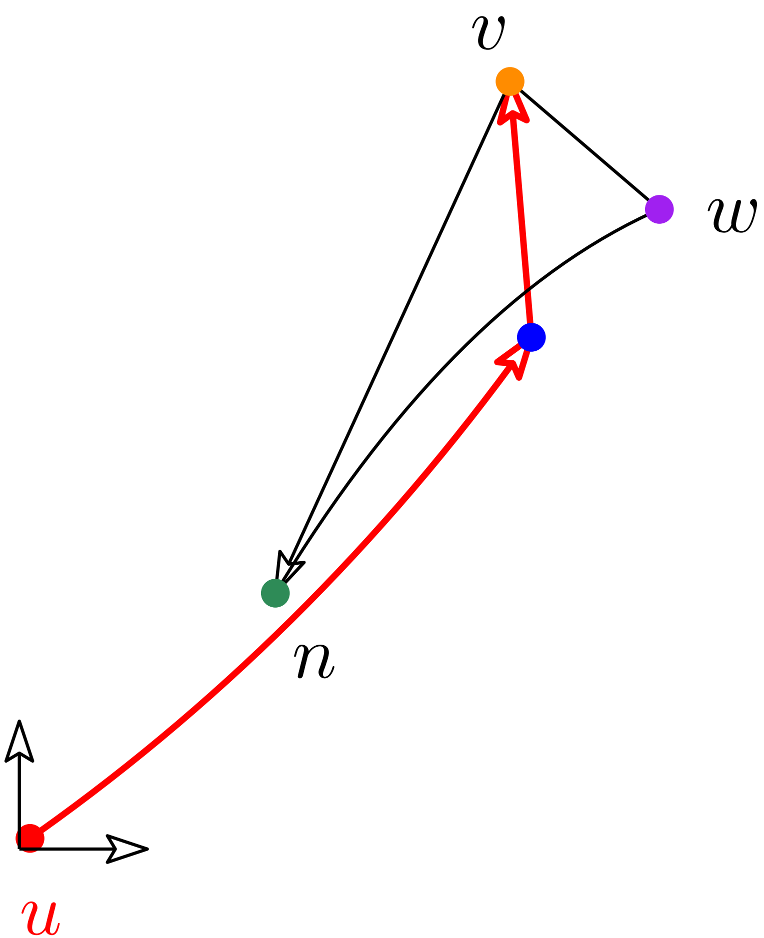
Do
- $n =$ first in S;
- For each $w \in S$
  - if vwn positive
    - then $n = w$;
- $v.next = n$; $v = n$;
- $S = S \setminus \{v\}$

While $v \neq u$

Convex hull

Teaser robustness lecture

Buggy degenerate example (single precision)



$w_1 = (12, 12)$

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

$w_3 = (30, 30.000001)$

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

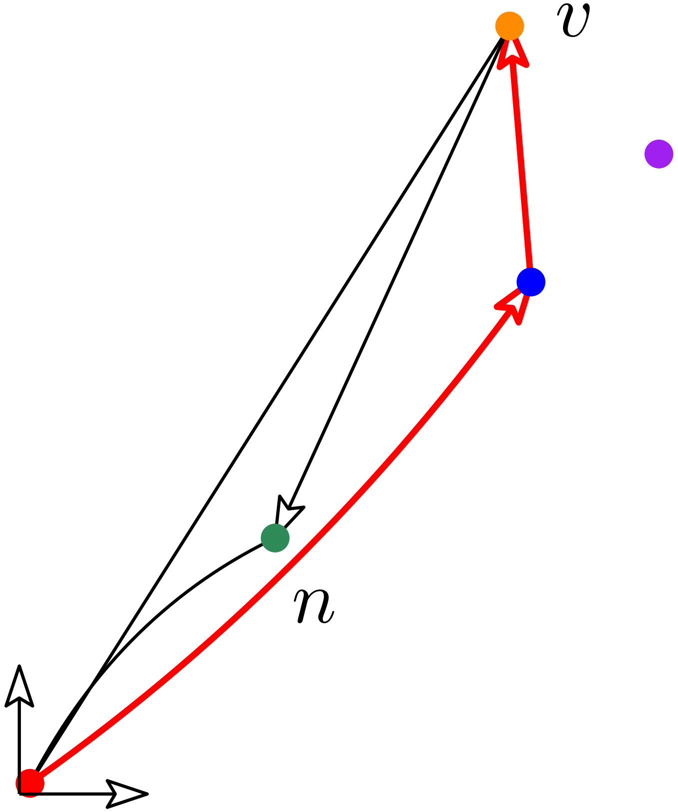
$w_5 = (0.5000029, 0.5000027)$

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)

$w_1 = (12, 12)$

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

$w_3 = (30, 30.000001)$

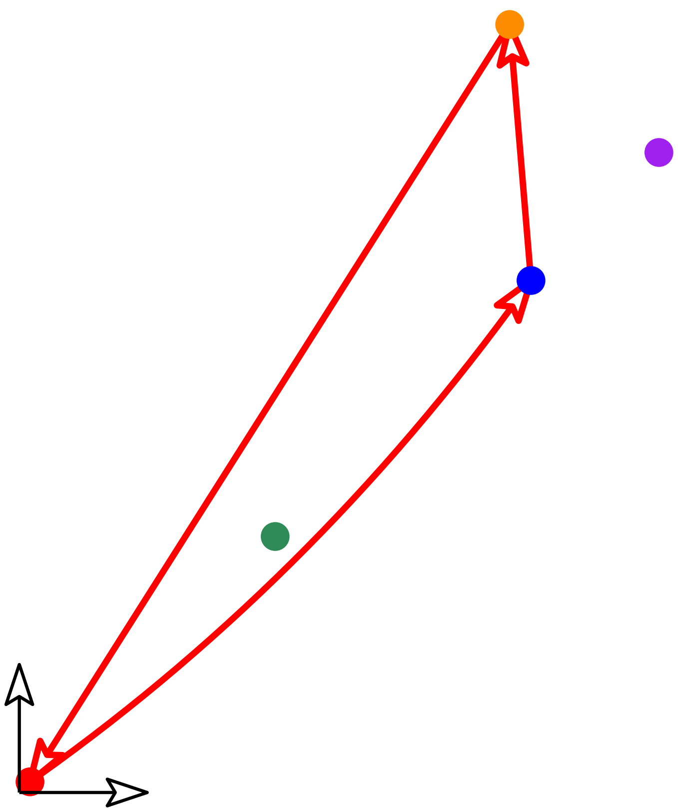
~~$w_4 = (23, 36)$~~

$w_5 = (0.5000029, 0.5000027)$

```
Do
    n = first in S;
    For each w ∈ S
        if vwn positive
          then n = w;
    v.next = n; v = n;
    S = S \ {v}
While v ≠ 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

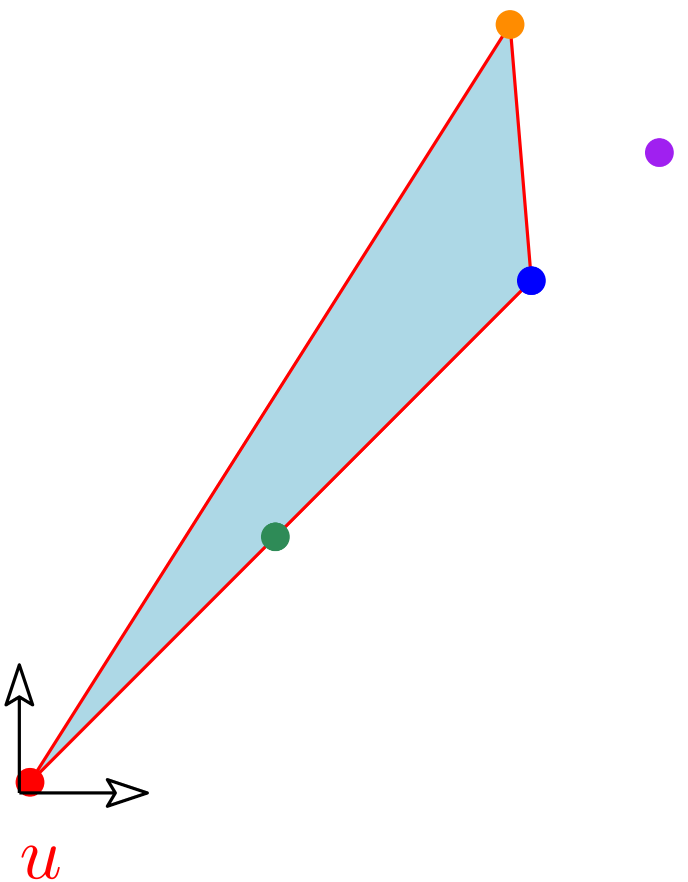
    $n =$ first in S;
    For each $w \in S$
        if vwn positive
            then $n = w$;
    $v.next = n$; $v = n$;
    $S = S \setminus \{v\}$

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

Convex hull

Real Random Access Memory model

Assume exact computation on real numbers

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

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 $\longmapsto \{-1, 0, 1\}$

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: $+, -, \sqrt{\ }, \sin \ldots$

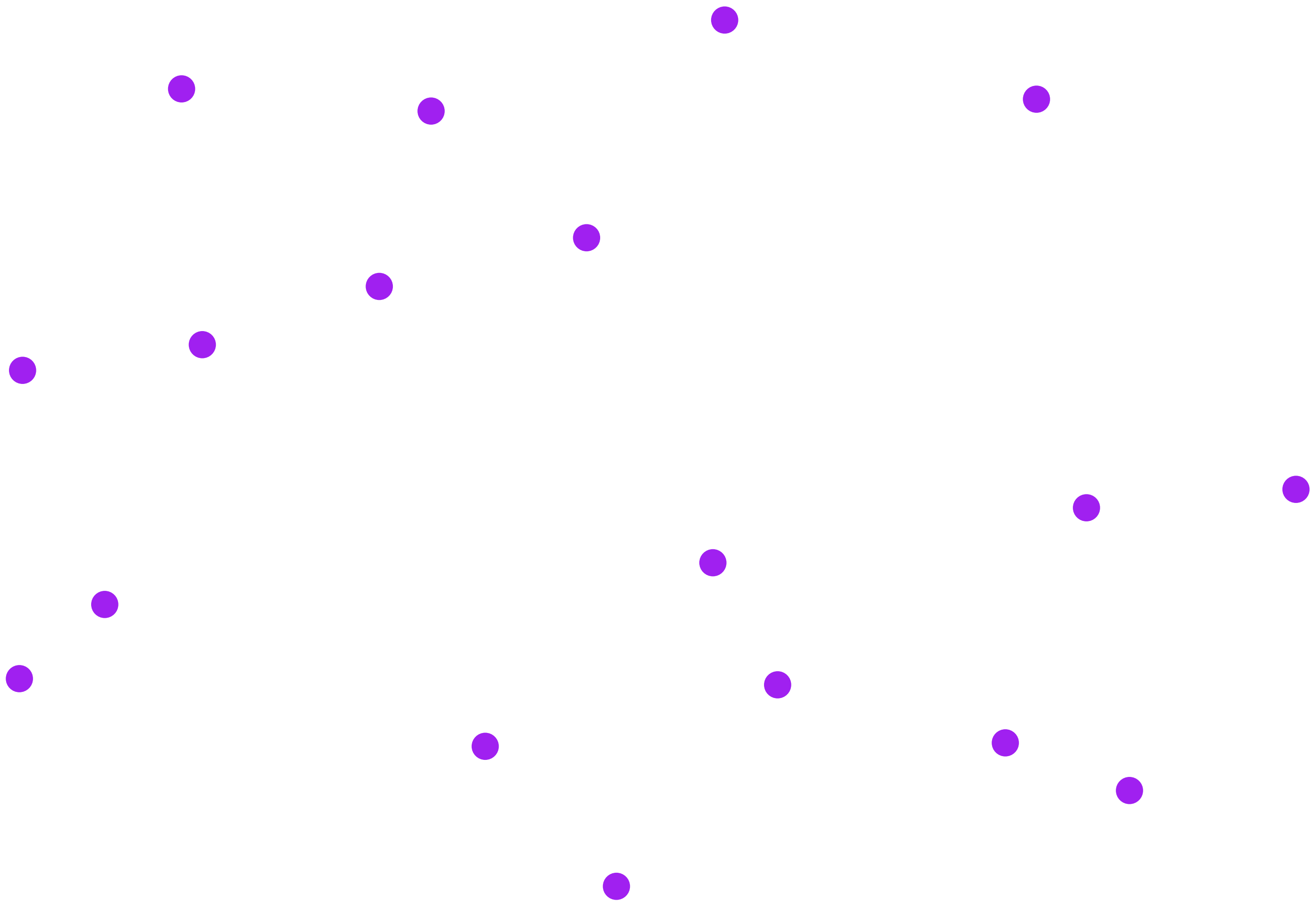
General position hypotheses

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

2D convex hull: no three points colinear

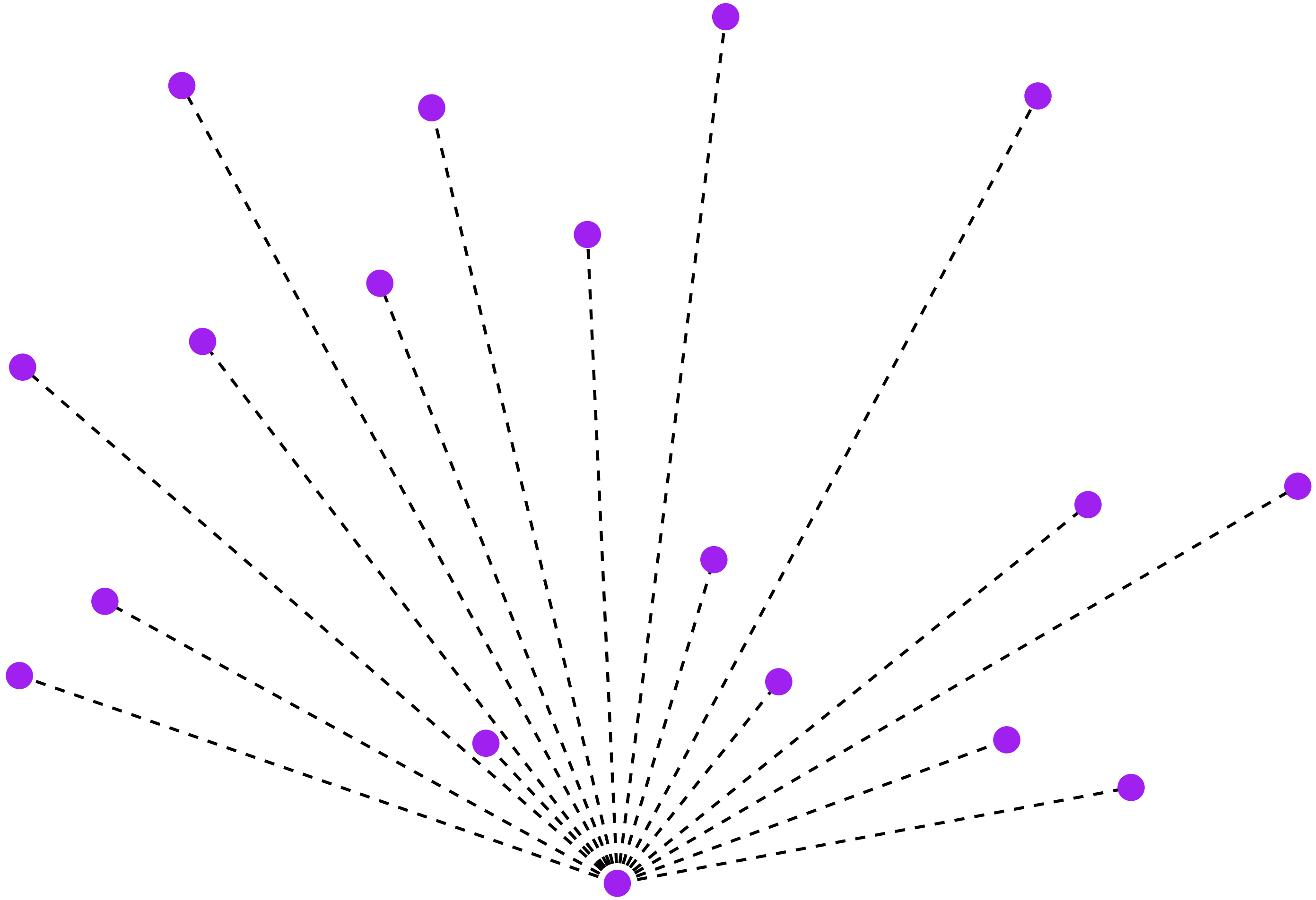
possibly: no 2 points with same x

Convex hull



Convex hull

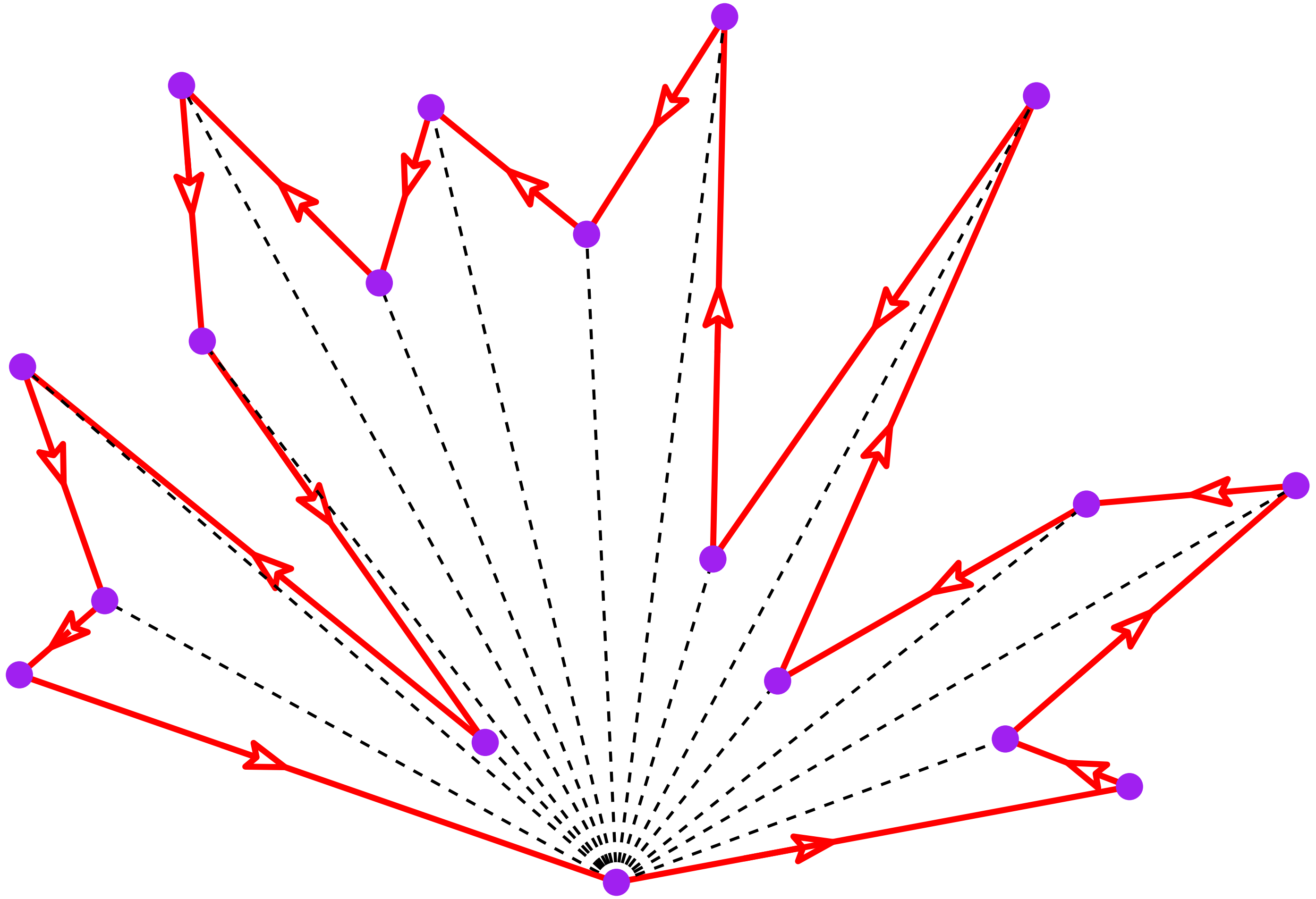
sort around a point (e.g. lowest point)



Convex hull

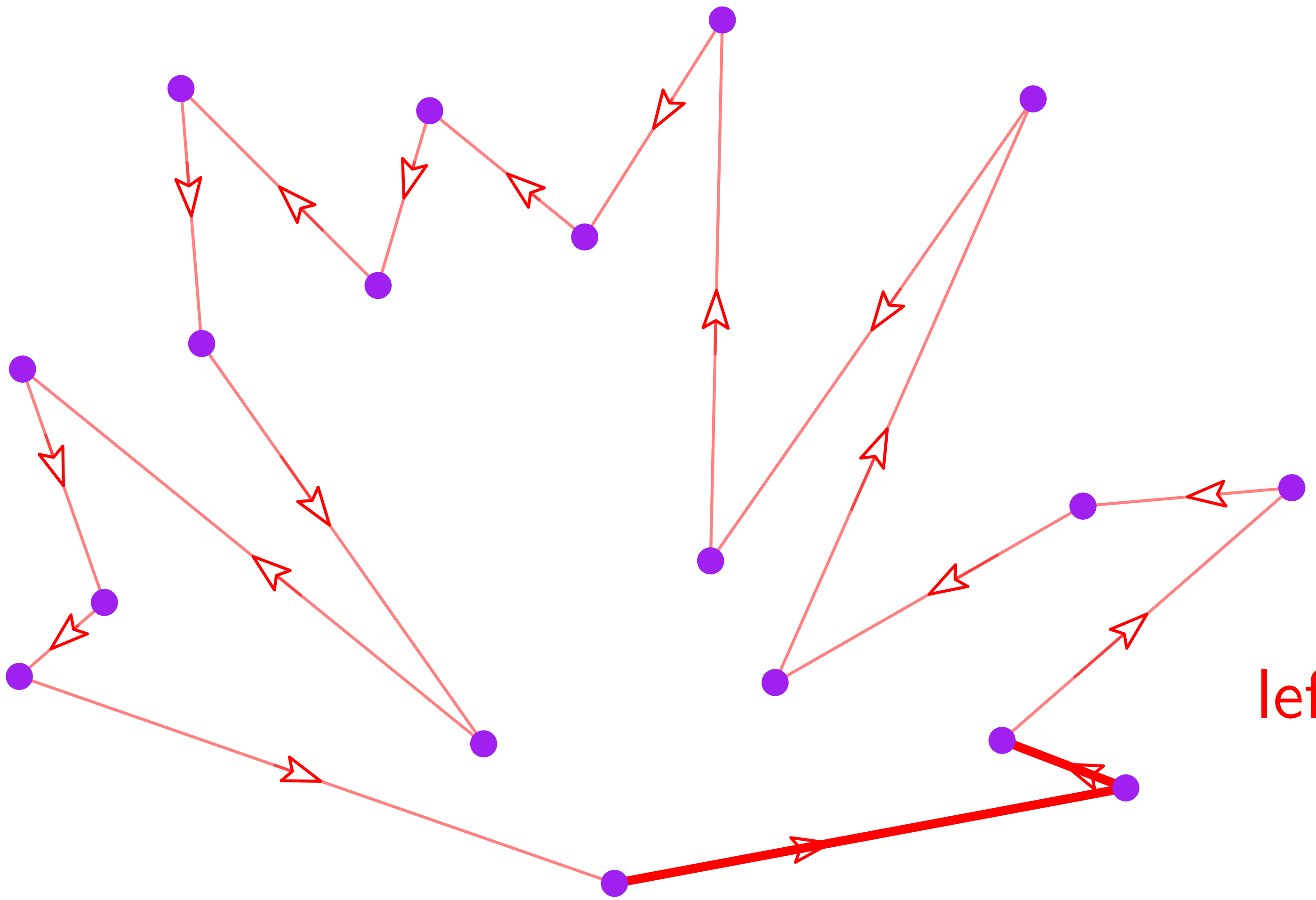
Graham algorithm

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

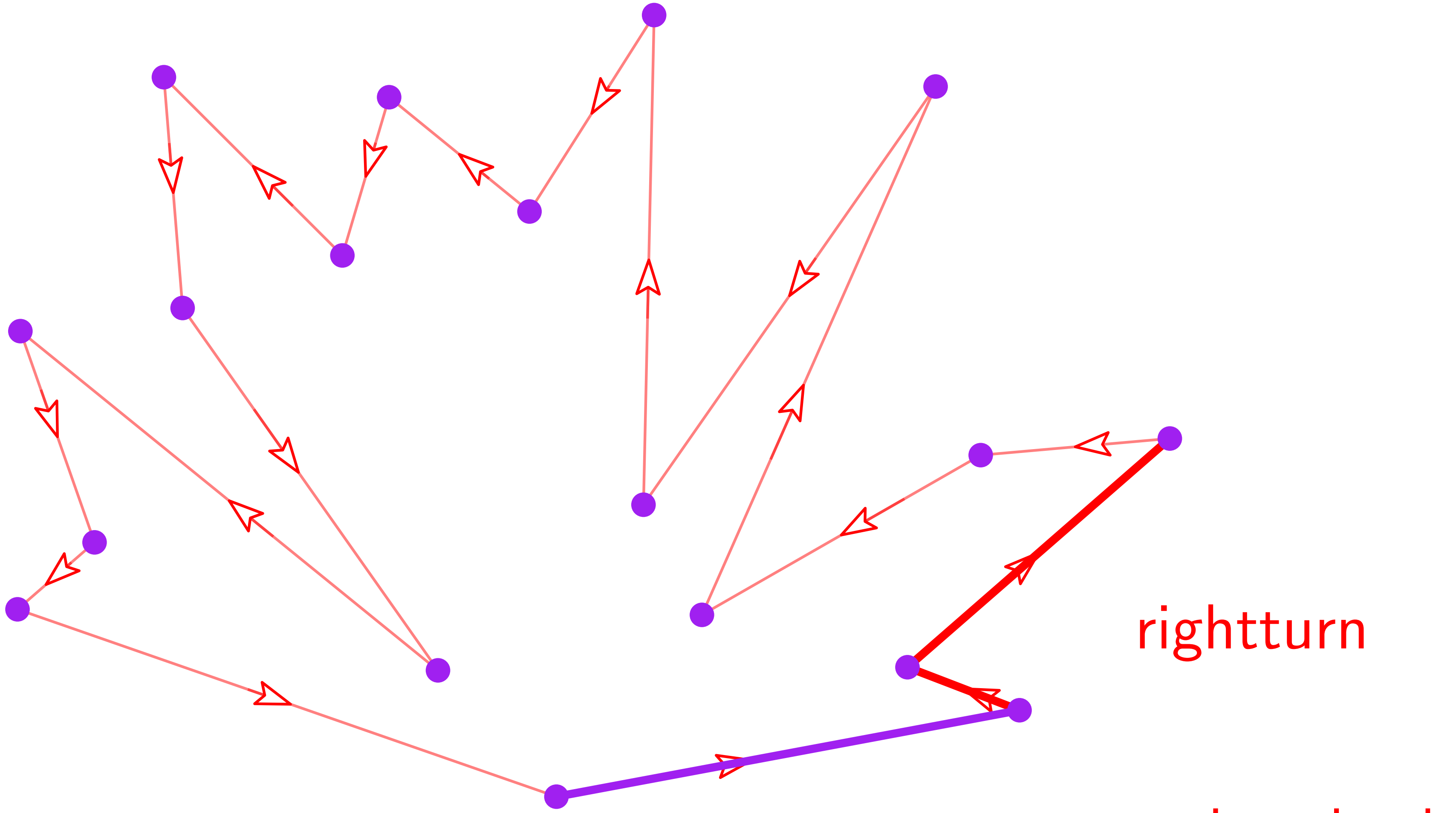


Convex hull

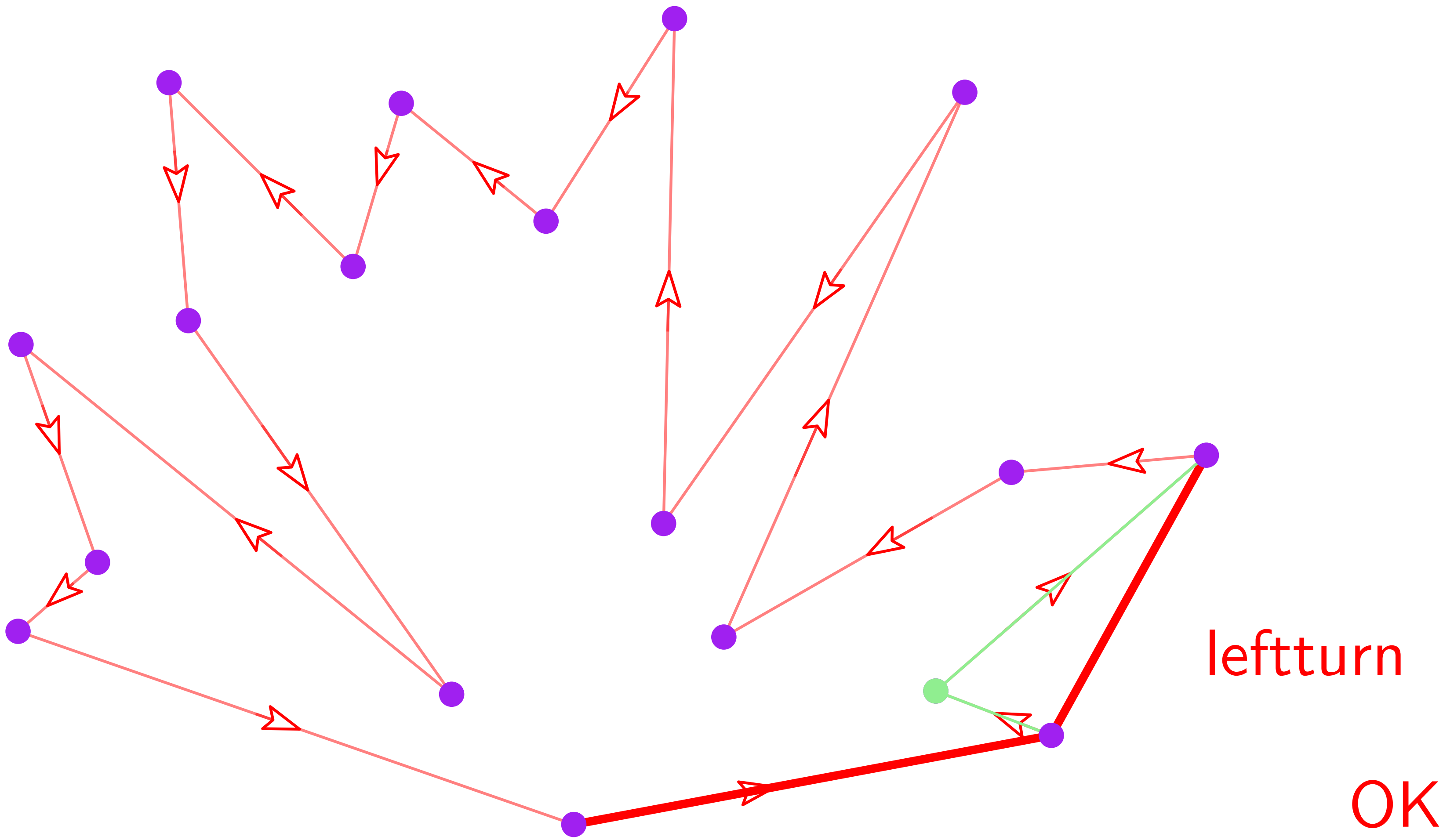
Graham algorithm



Convex hull

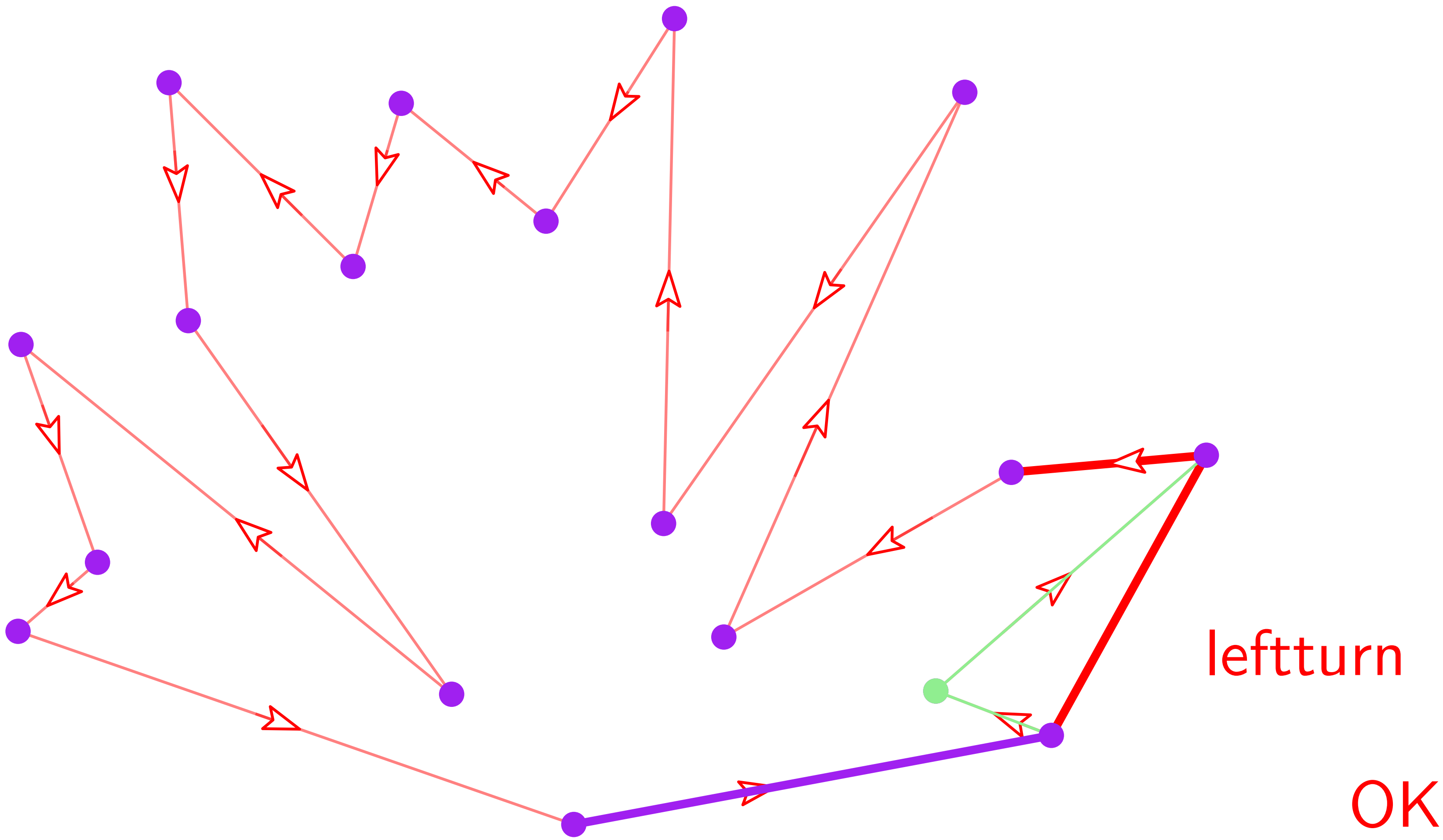


Convex hull



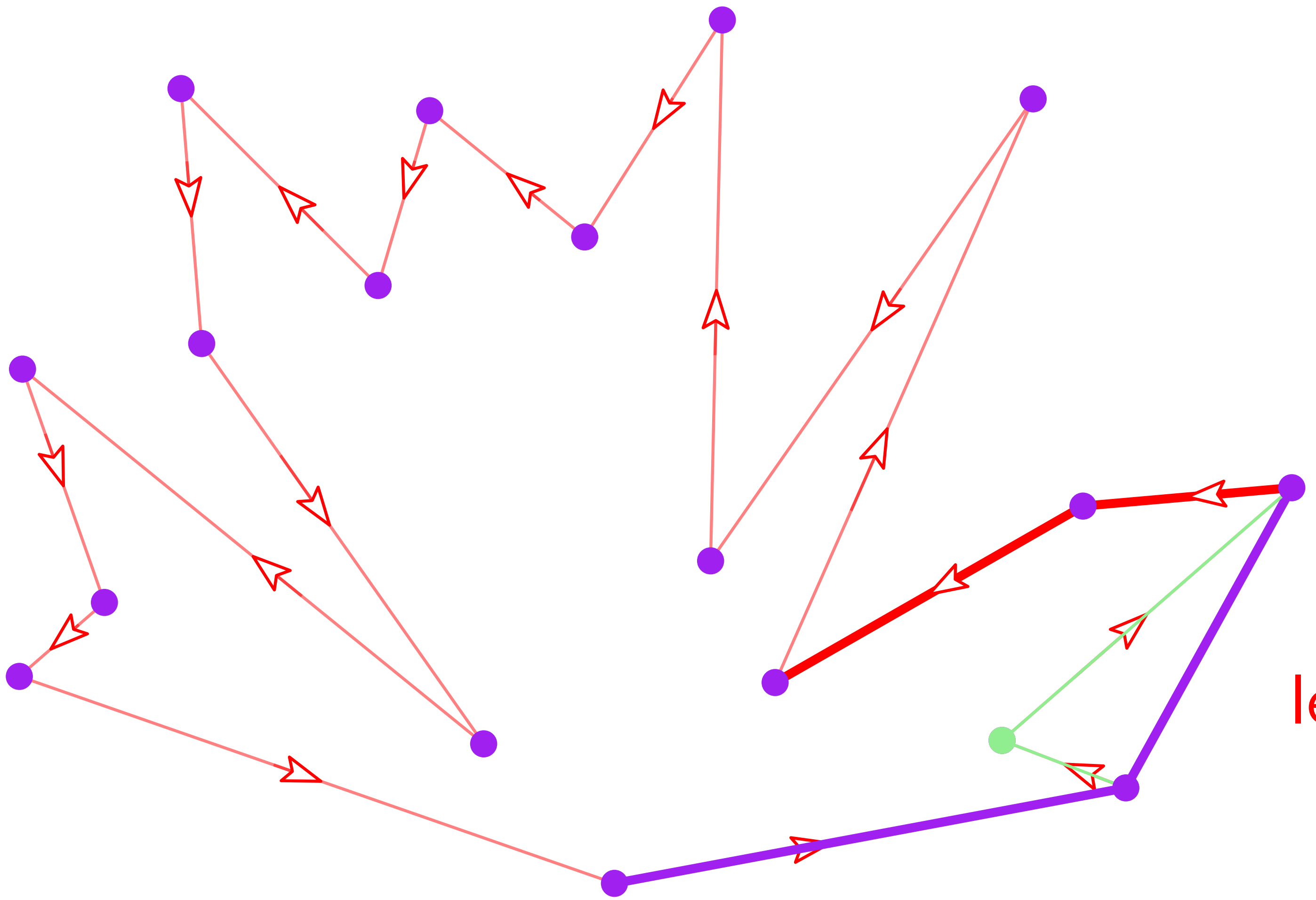
Convex hull

Graham algorithm



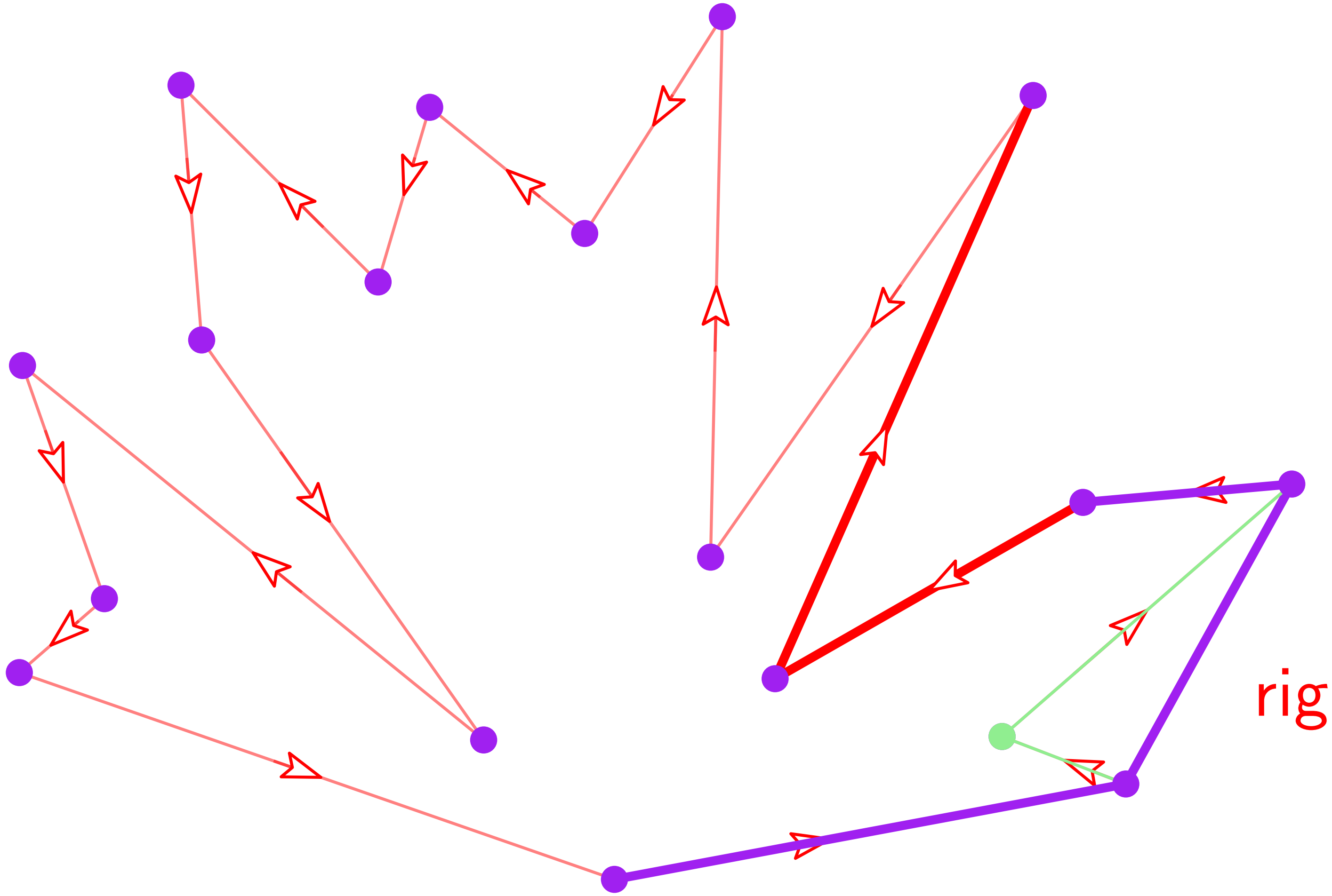
Convex hull

Graham algorithm



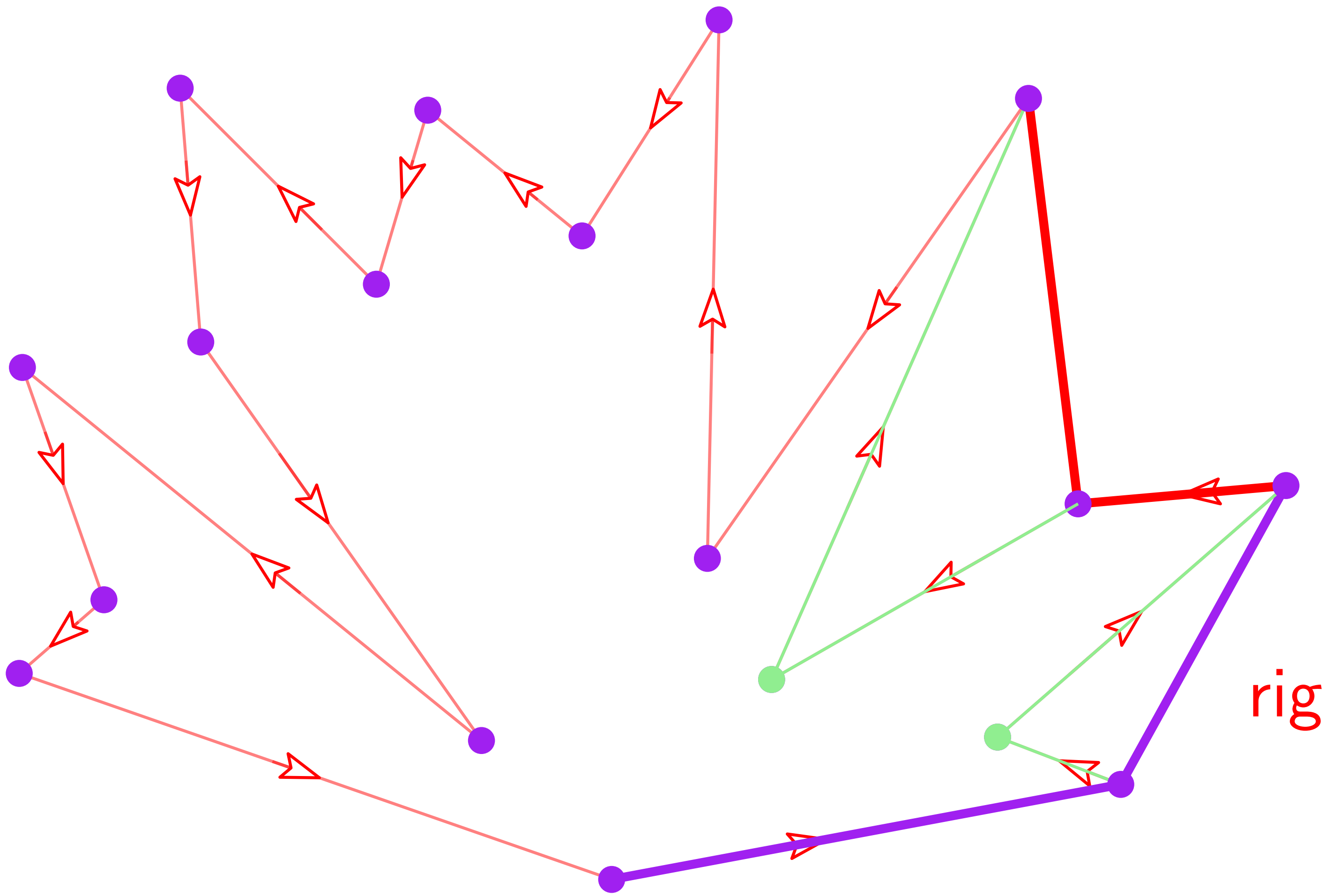
Convex hull

Graham algorithm



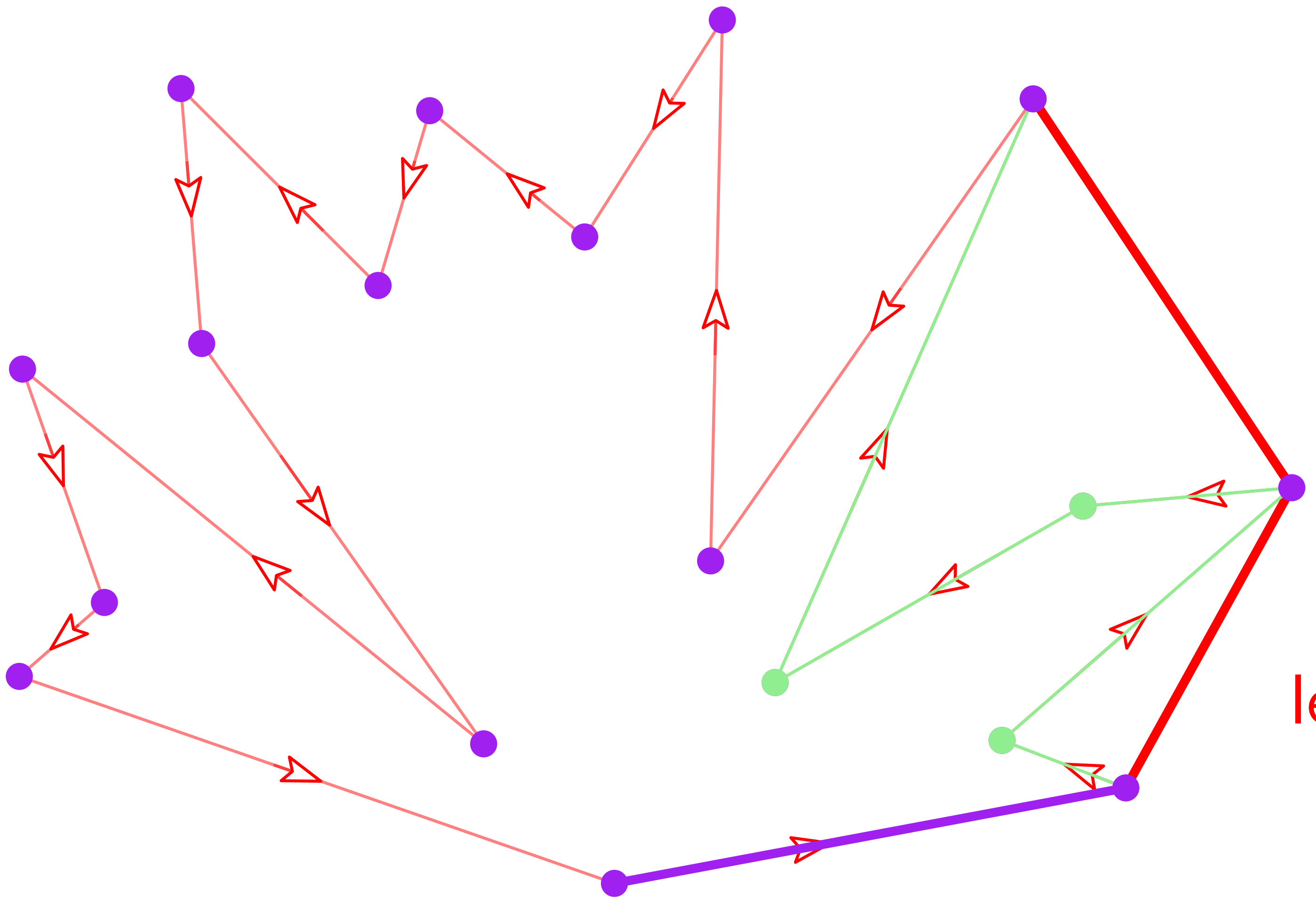
Convex hull

Graham algorithm



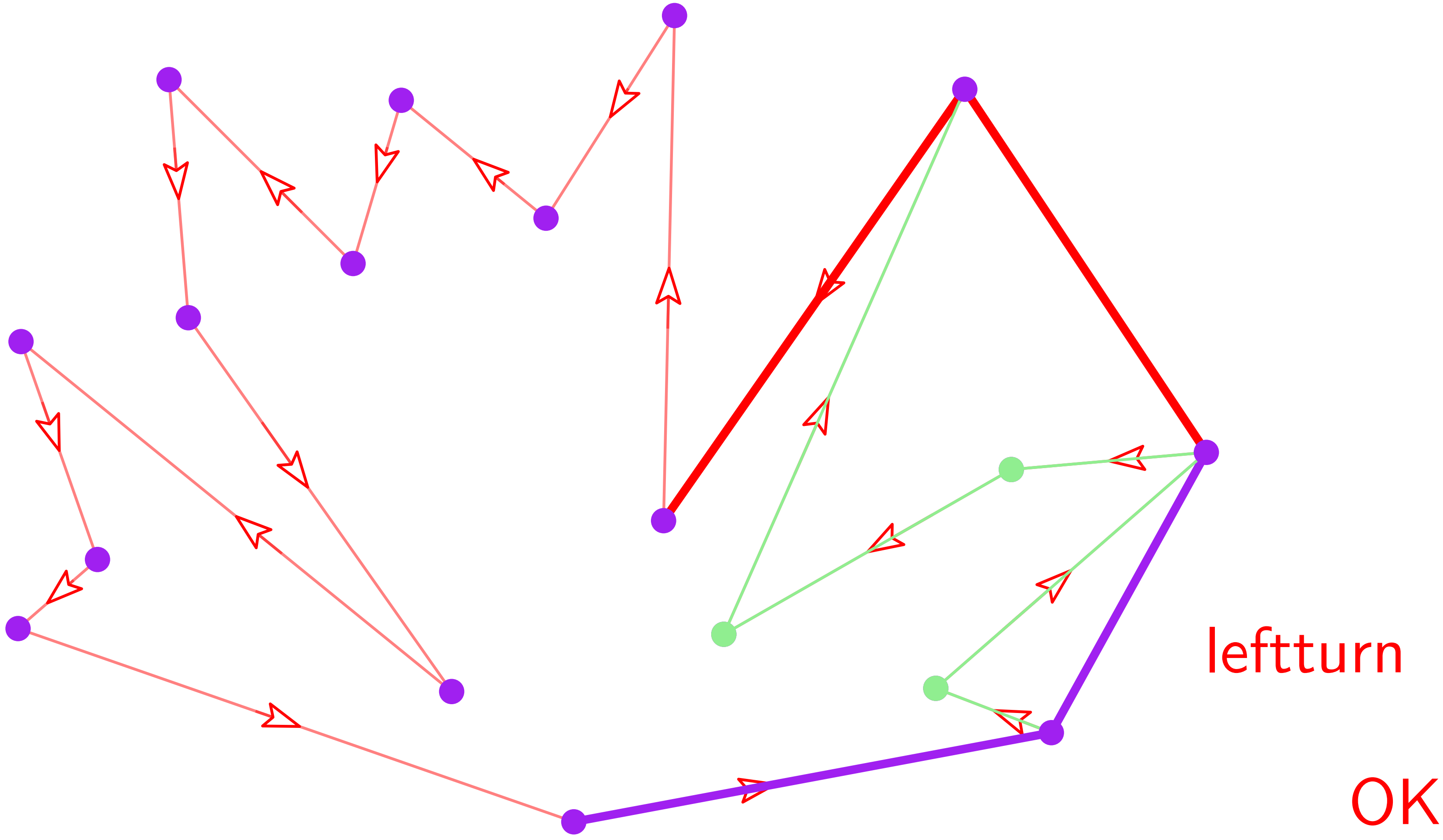
Convex hull

Graham algorithm



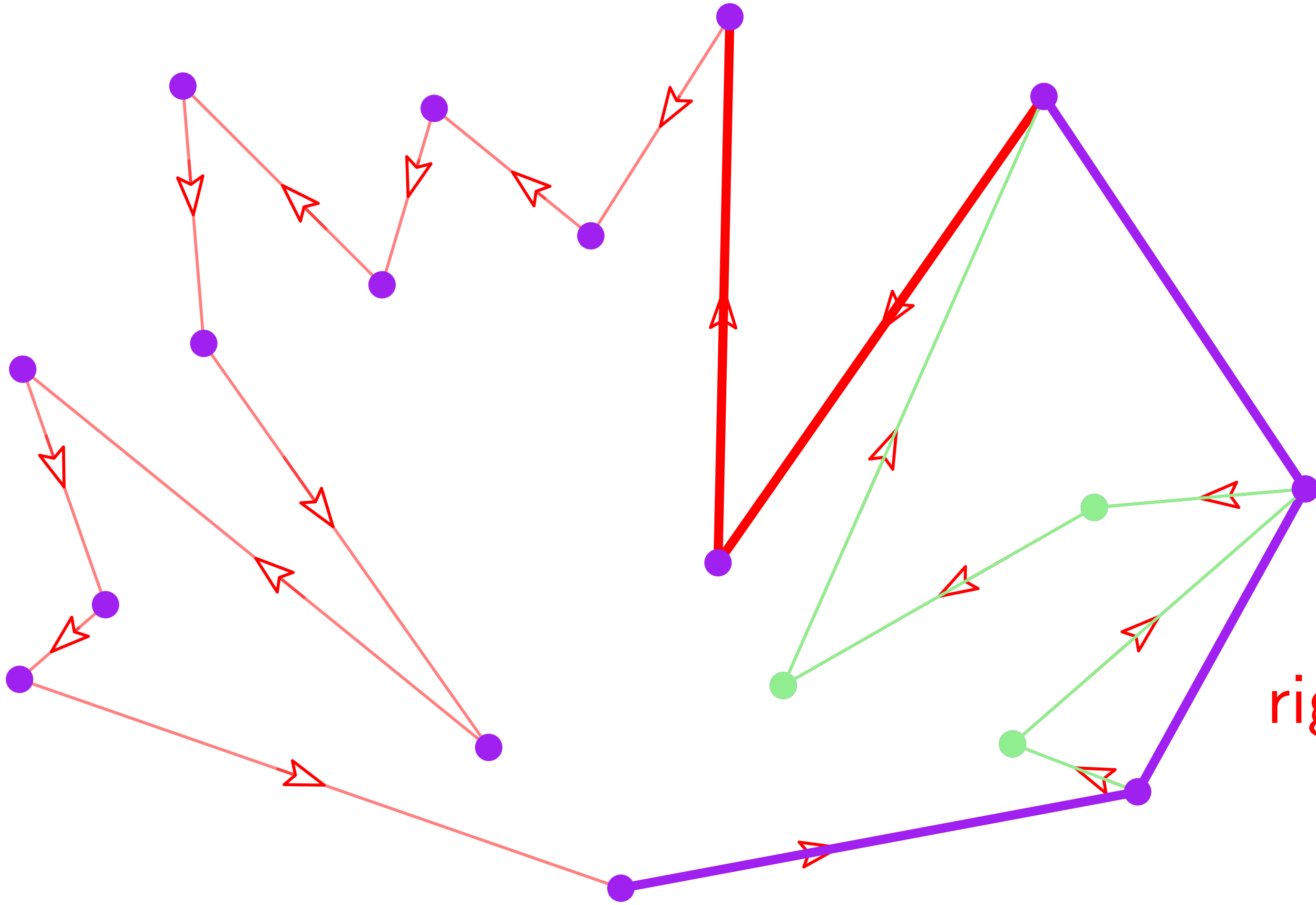
Convex hull

Graham algorithm



Convex hull

Graham algorithm

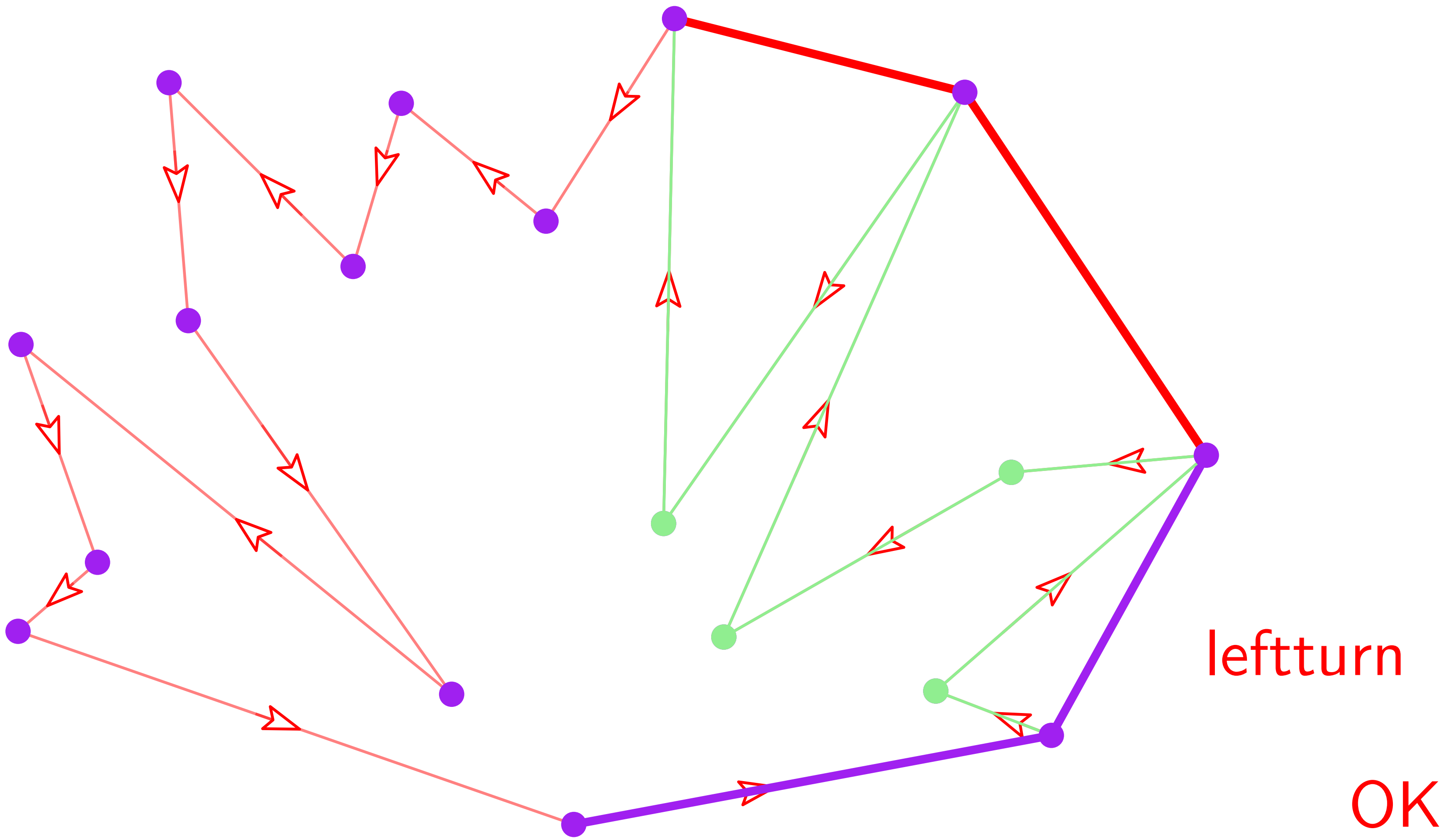


rightturn

remove and go back

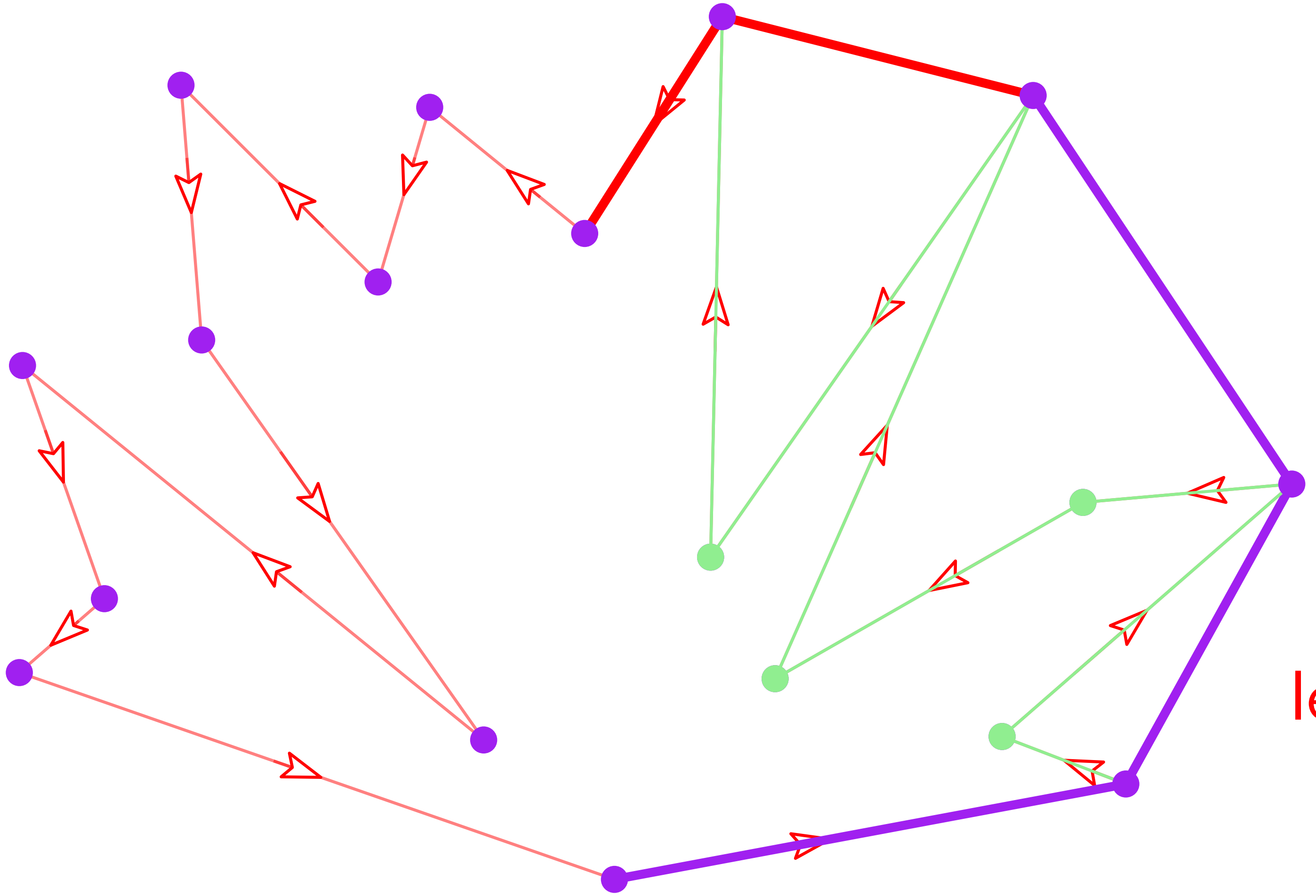
Convex hull

Graham algorithm



Convex hull

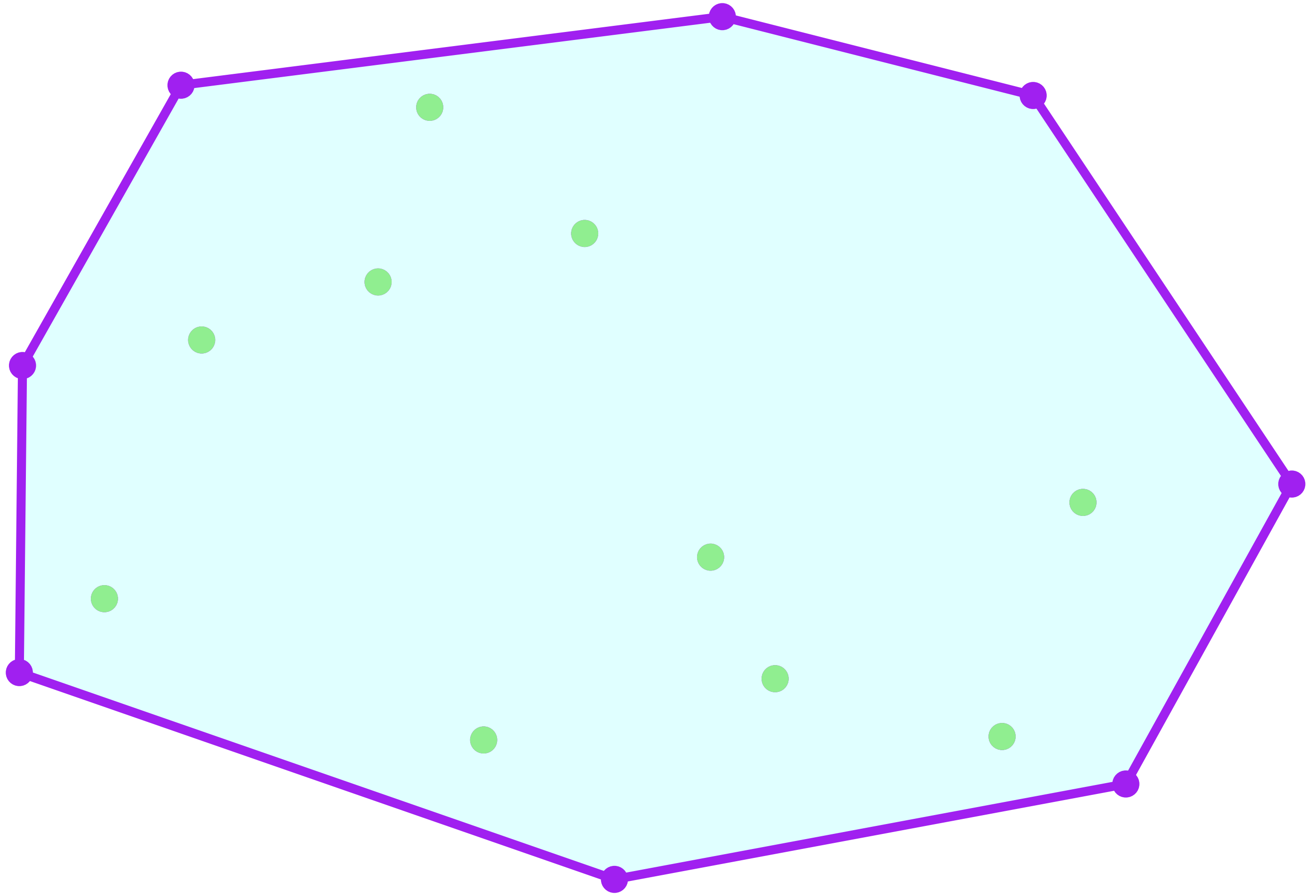
Graham algorithm



leftturn

OK

Convex hull



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

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

$O(n)$

Convex hull

Complexity

Graham algorithm

Input: point set S

u lowest point of S;

sort S around u in a circular list including u; $O(n \log n)$

$v = u$;

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

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

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

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

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;

$v = u$;

$O(n \log n)$

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

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

Sorting

Input: n real (positive) numbers

Sorting

Input: n real (positive) numbers

Output: sorting permutation

Sorting

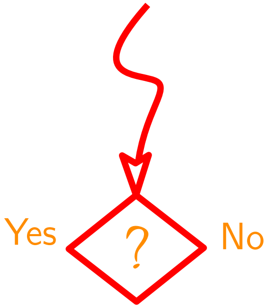
Input: n real (positive) numbers

Output: sorting permutation

Monitoring execution

Sorting

Input: n real (positive) numbers

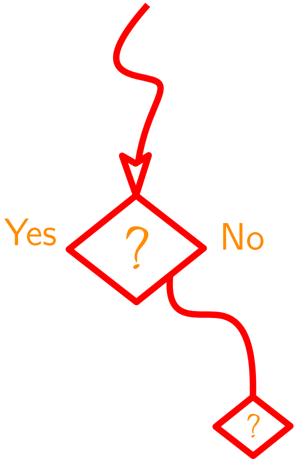


Output: sorting permutation

Monitoring execution

Sorting

Input: n real (positive) numbers



Output: sorting permutation

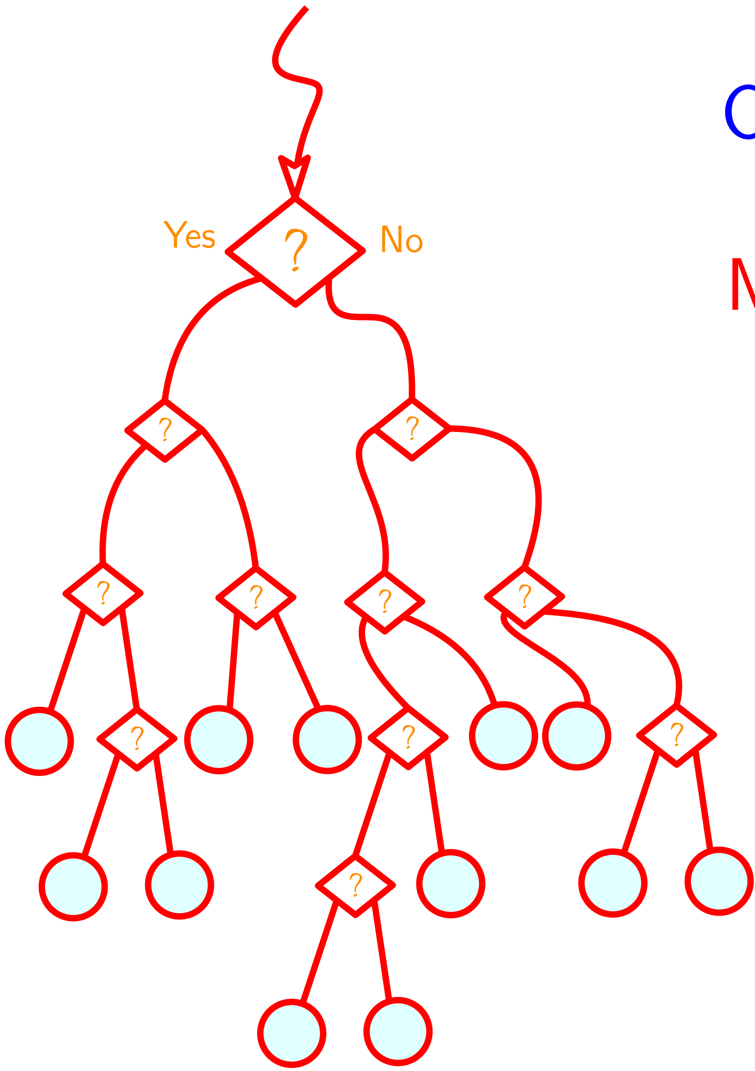
Monitoring execution

Sorting

Input: n real (positive) numbers

Output: sorting permutation

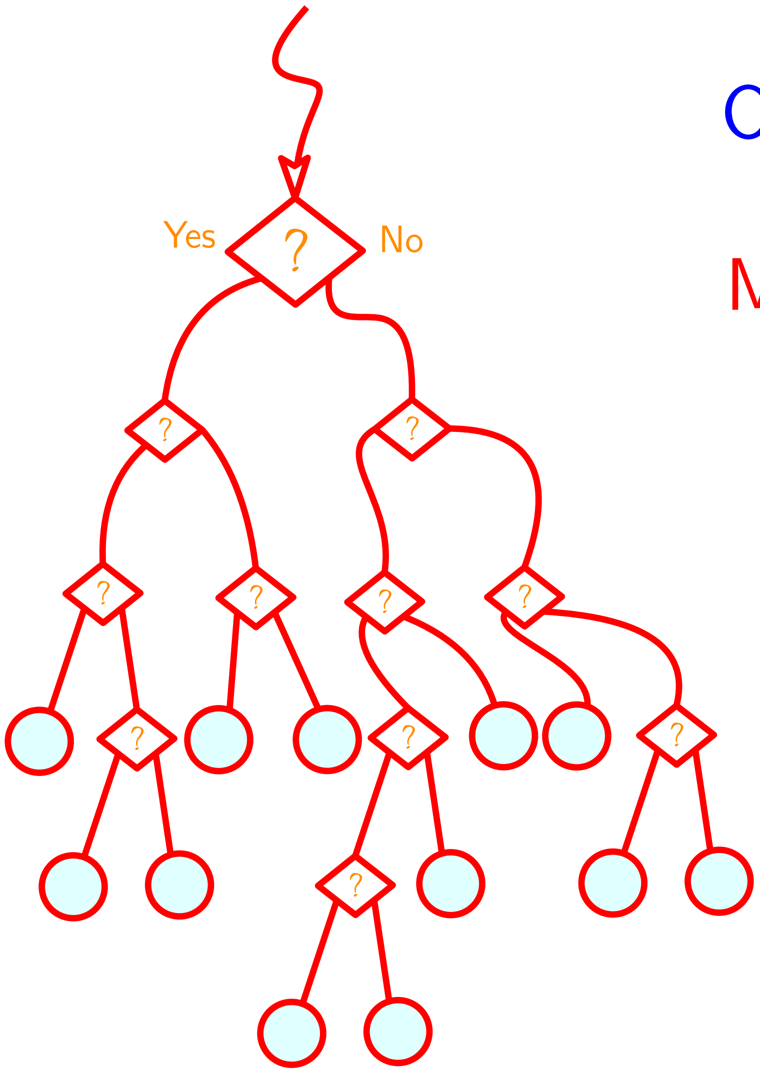
Monitoring execution



Sorting

Lower bound

Input: n real (positive) numbers



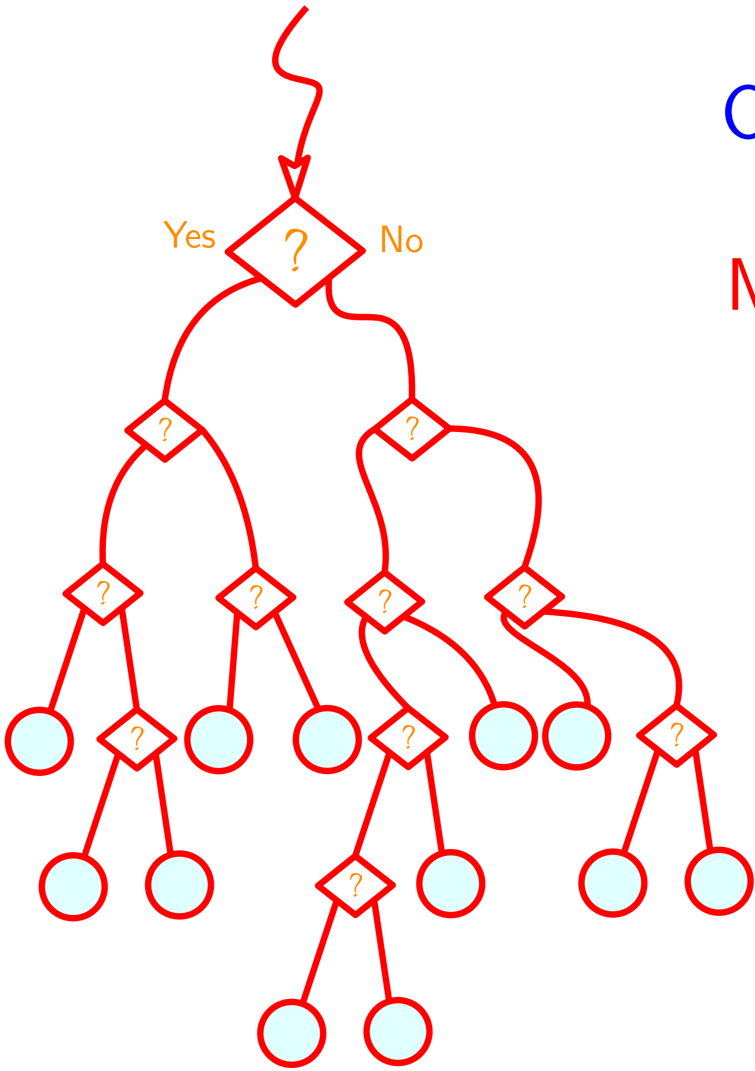
Output: sorting permutation

Monitoring execution

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

Sorting

Input: n real (positive) numbers



Output: sorting permutation

Monitoring execution

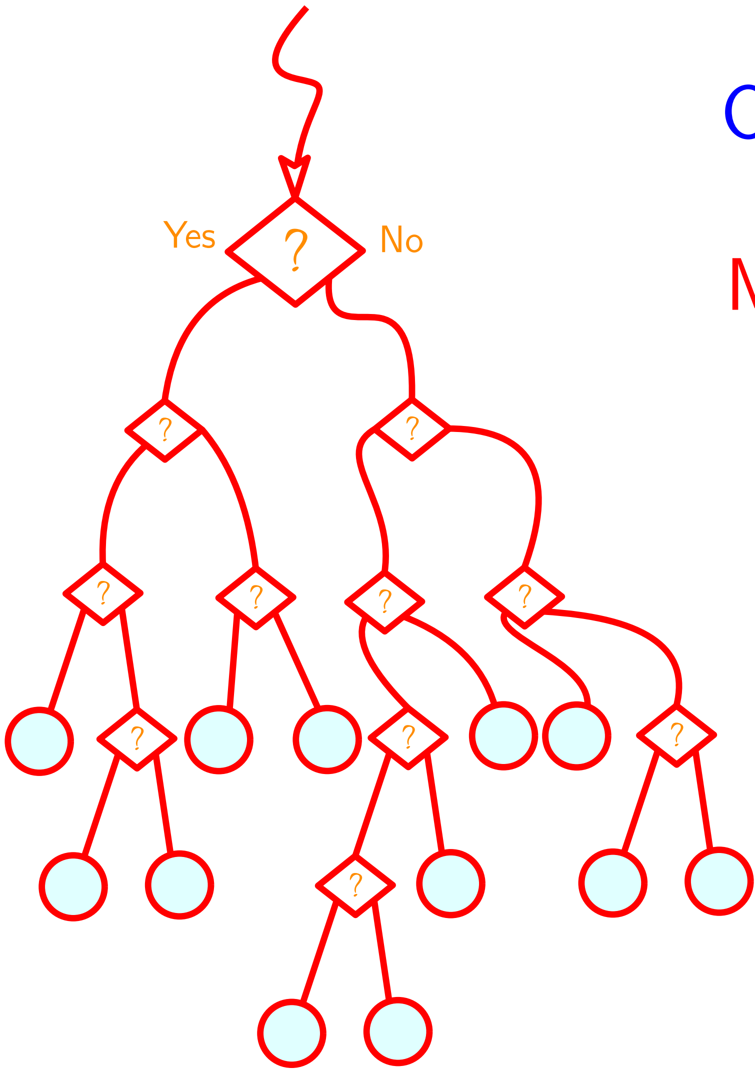
$\sharp$ leaves $\geq$ $\sharp$ permutations

There are $n!$ permutations

Sorting

Lower bound

Input: n real (positive) numbers



Output: sorting permutation

Monitoring execution

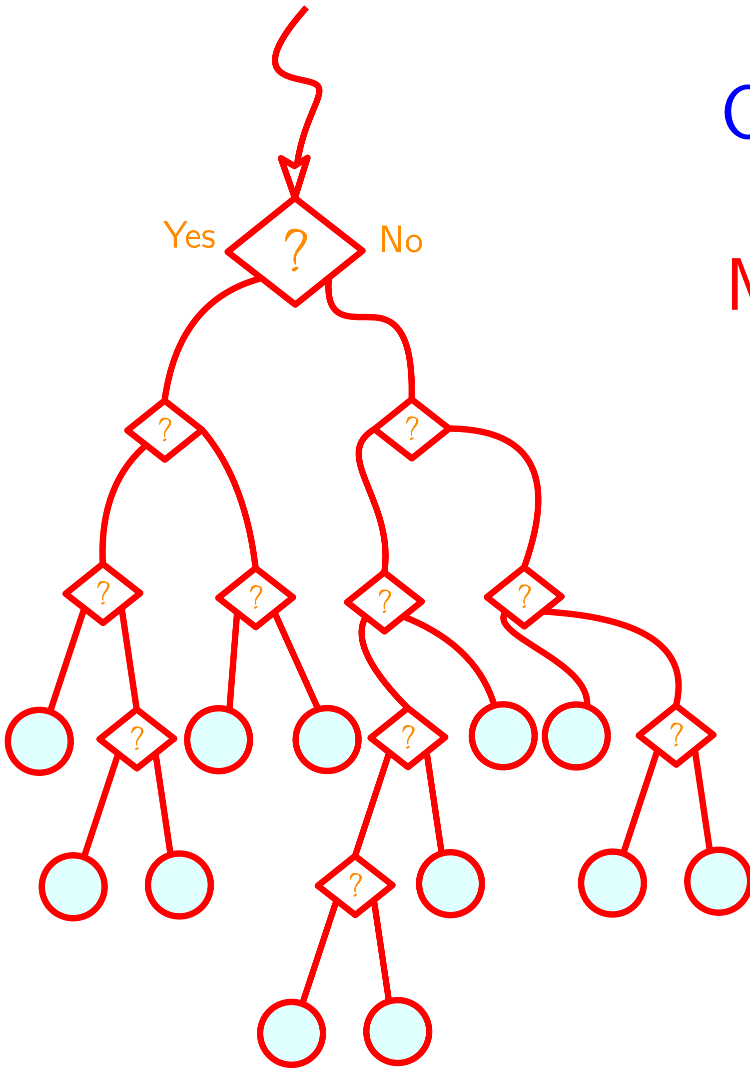
$\sharp$ leaves $\geq$ $\sharp$ permutations

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)

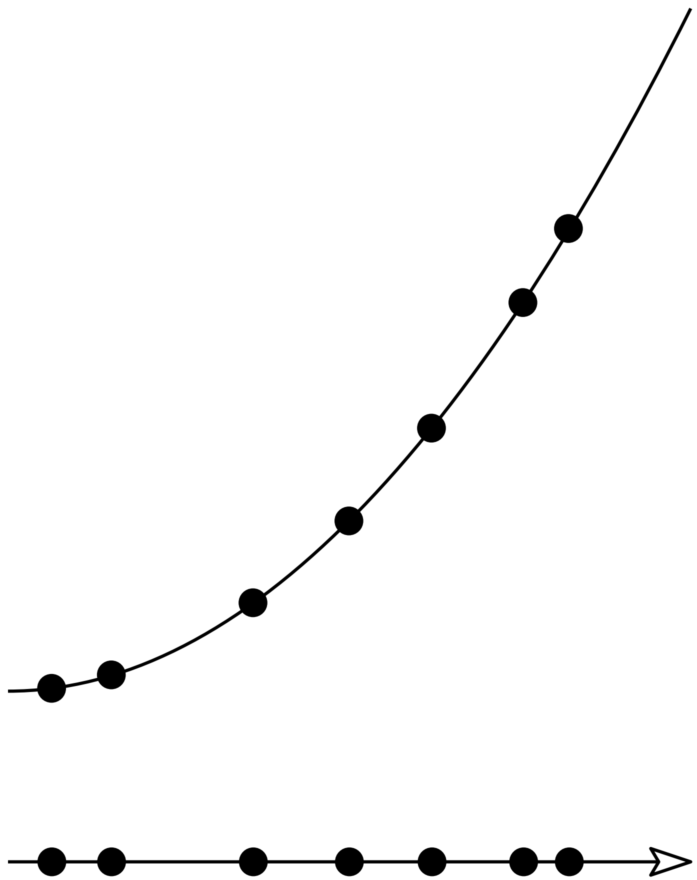
Output: list of points along the convex hull

Convex hull

A stupid algorithm for sorting numbers

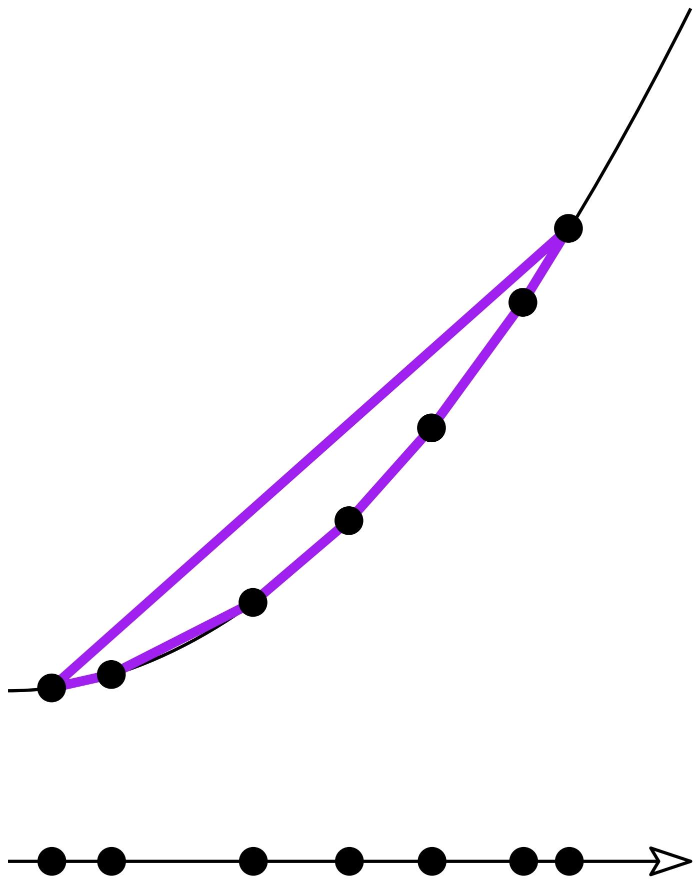
Convex hull

Lower bound



project on parabola

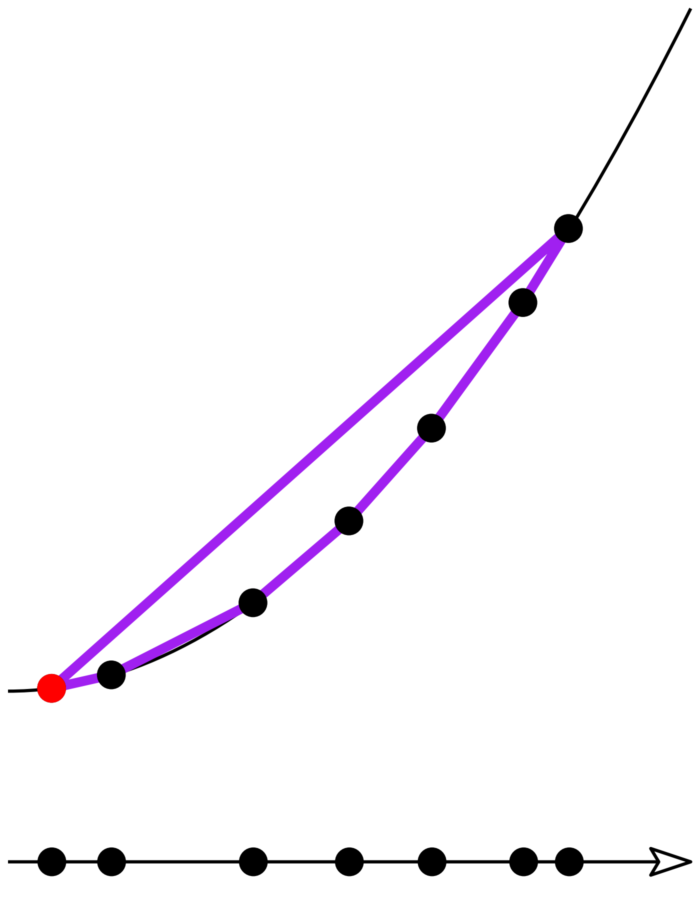
Convex hull



project on parabola

compute convex hull

Convex hull

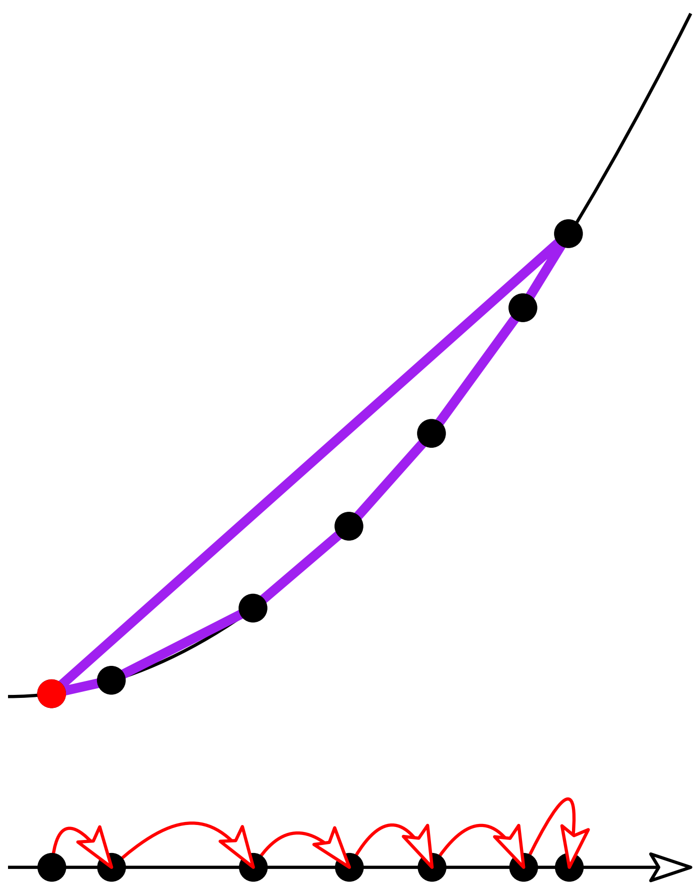


project on parabola

compute convex hull

find lowest point

Convex hull



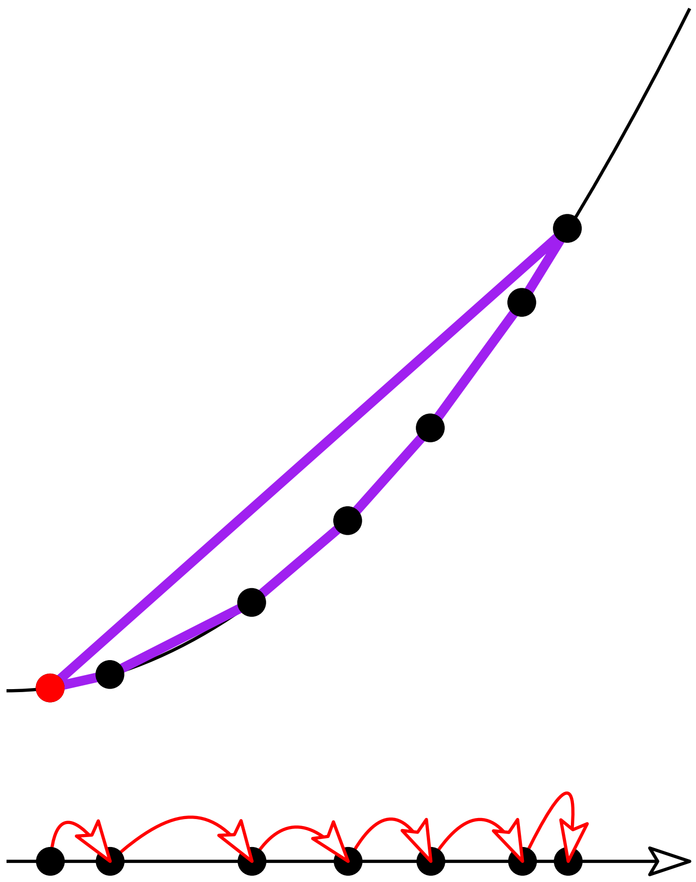
project on parabola

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

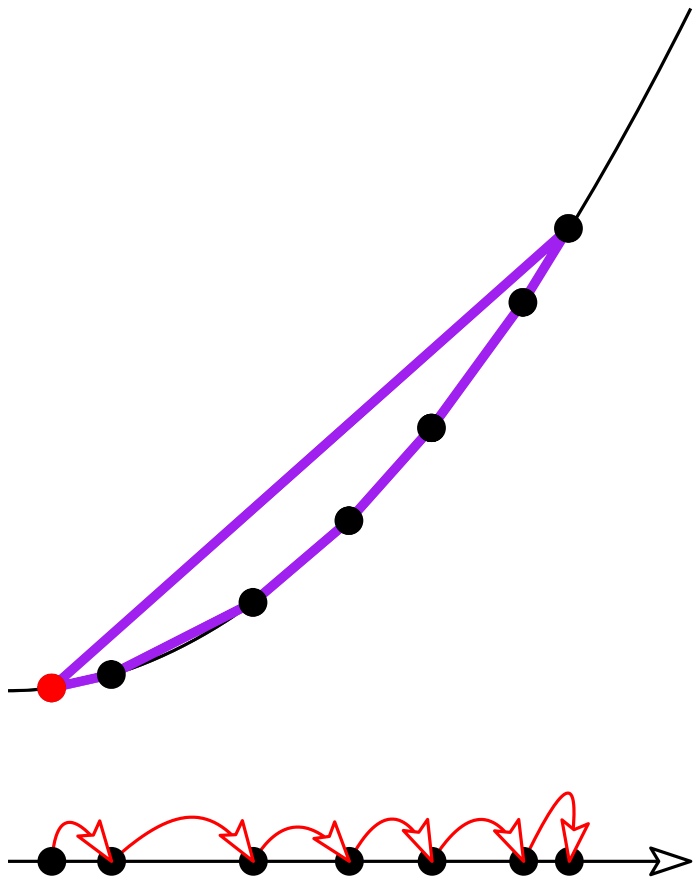
$O(n)$ find lowest point

$O(n)$ 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

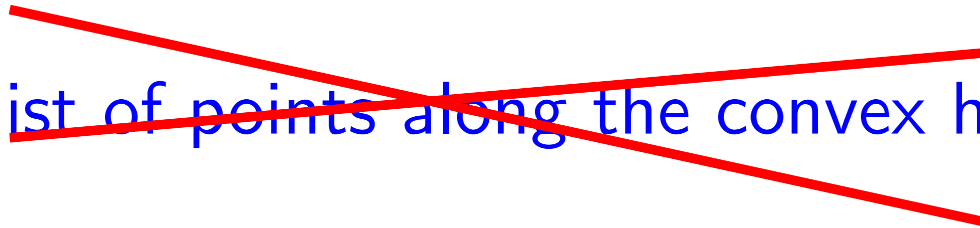
Lower bound on sorting

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

Convex hull

Input: n points in $\mathbb{R}^2$ (real coordinates)

Output: list of points along the convex hull



Convex hull

Input: n points in $\mathbb{R}^2$ (real coordinates)

~~Output: list of points along the convex hull~~

Output: list of extreme points (not ordered)

Convex hull

Input: n points in $\mathbb{R}^2$ (real coordinates)

~~Output: list of points along the convex hull~~

Output: list of extreme points (not ordered)

Weaker output: are all points extreme (strictly)

Convex hull

Input: n points in $\mathbb{R}^2$ (real coordinates)

~~Output: list of points along the convex hull~~

Output: list of extreme points (not ordered)

Weaker output: are all points extreme (strictly)

i.e., split $\mathbb{R}^{2n}$ in two parts

- "all points are extreme" part $= S$
- complementary part

Theorem [Ben-Or]: Any decision tree algorithm that solve the membership in a set S problem has lower bound $\log_2 \sharp(S)$ where $\sharp(S)$ is the number of connected component of S.

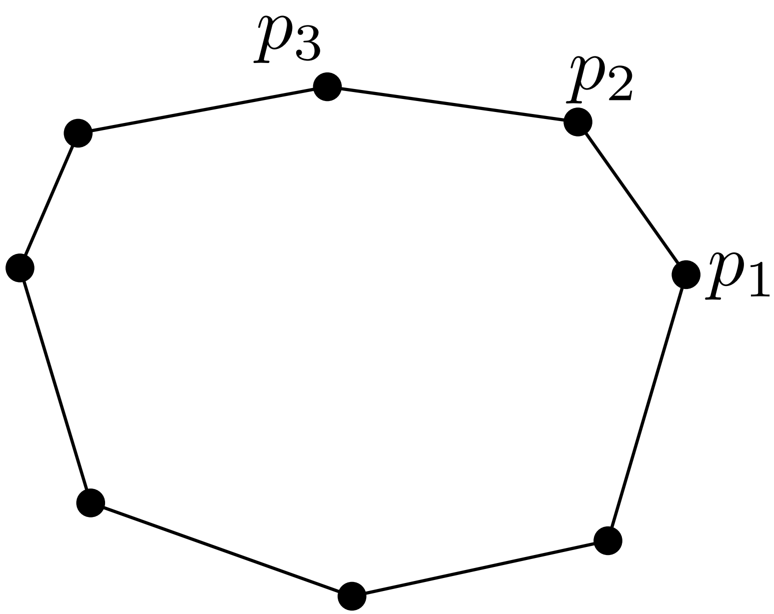
Just prove that S has enough connected components

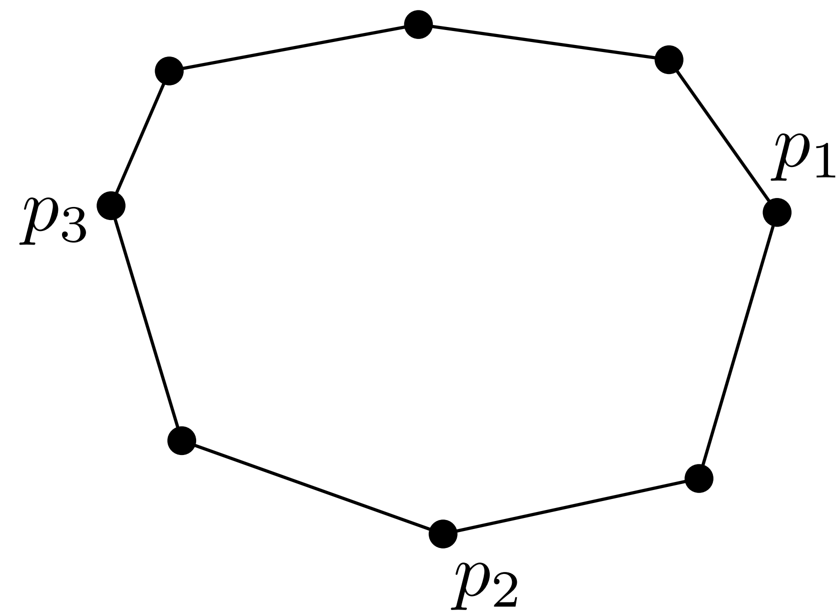
i.e., split $\mathbb{R}^{2n}$ in two parts

- "all points are extreme" part $= S$
- complementary part

Convex hull

Just prove that S has enough connected components





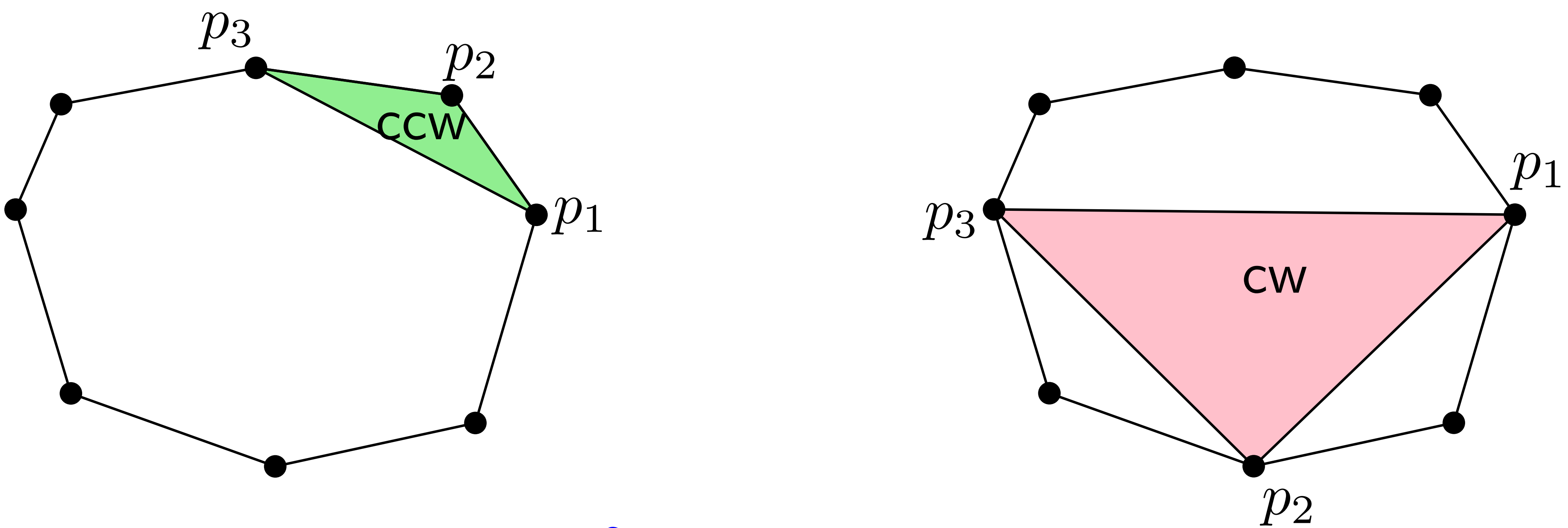
i.e., split $\mathbb{R}^{2n}$ in two parts

- “all points are extreme” part $= S$
- complementary part

Convex hull

Another lower bound

Just prove that S has enough connected components

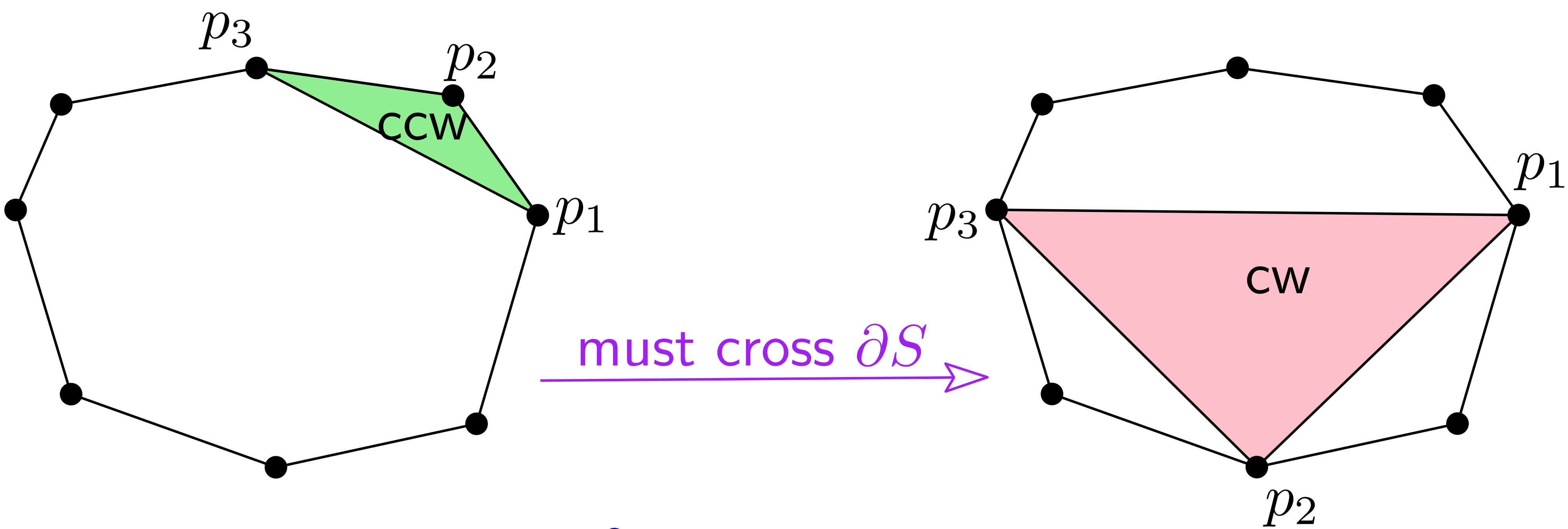


i.e., split $I\!R^{2n}$ in two parts

- "all points are extreme" part $= S$
- complementary part

Convex hull

Just prove that S has enough connected components

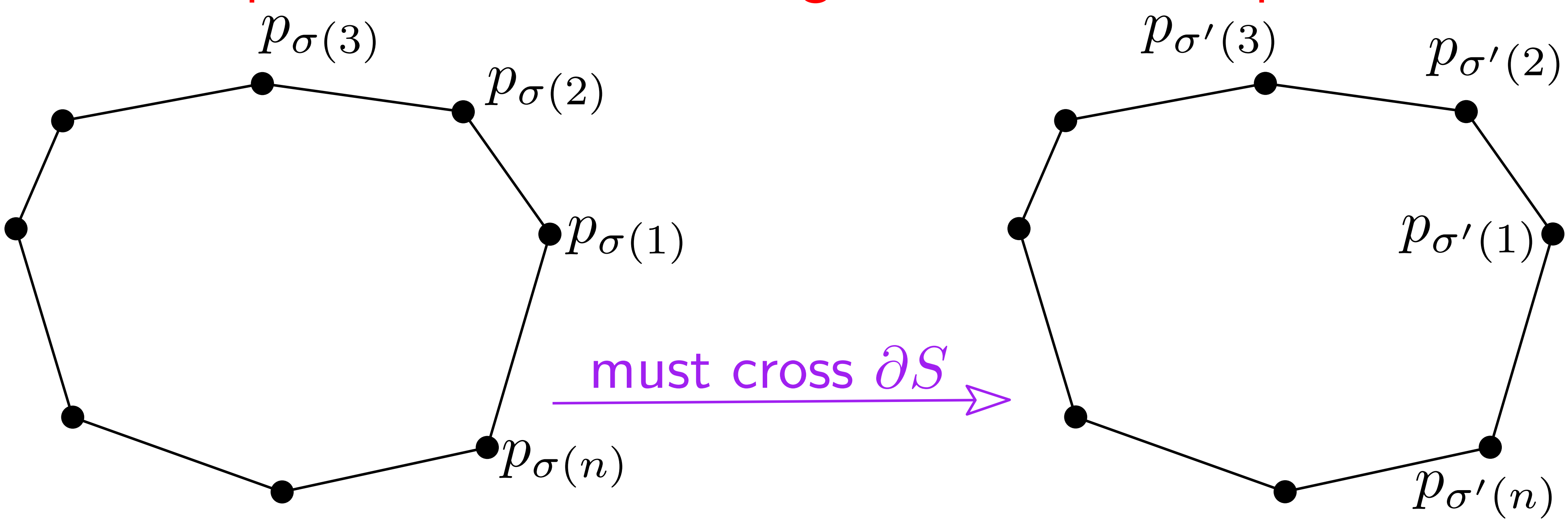


i.e., split $\mathbb{R}^{2n}$ in two parts

- "all points are extreme" part $= S$
- complementary part

Convex hull

Just prove that S has enough connected components

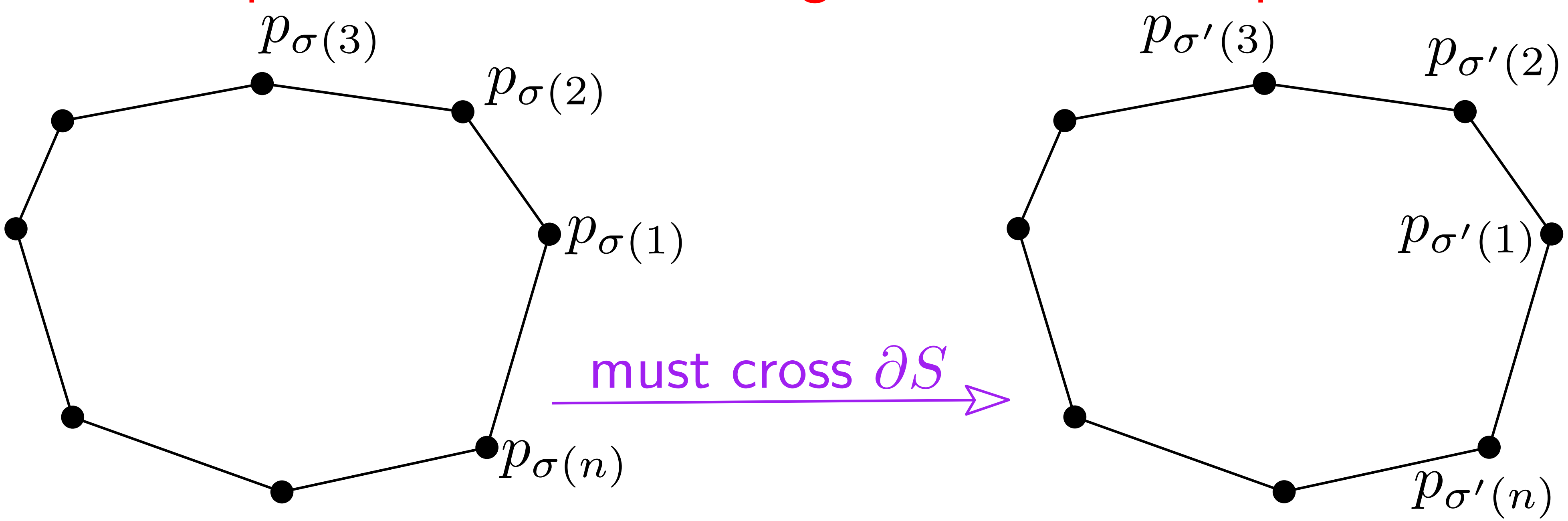


i.e., split $I\!R^{2n}$ in two parts

- "all points are extreme" part $= S$
- complementary part

Convex hull

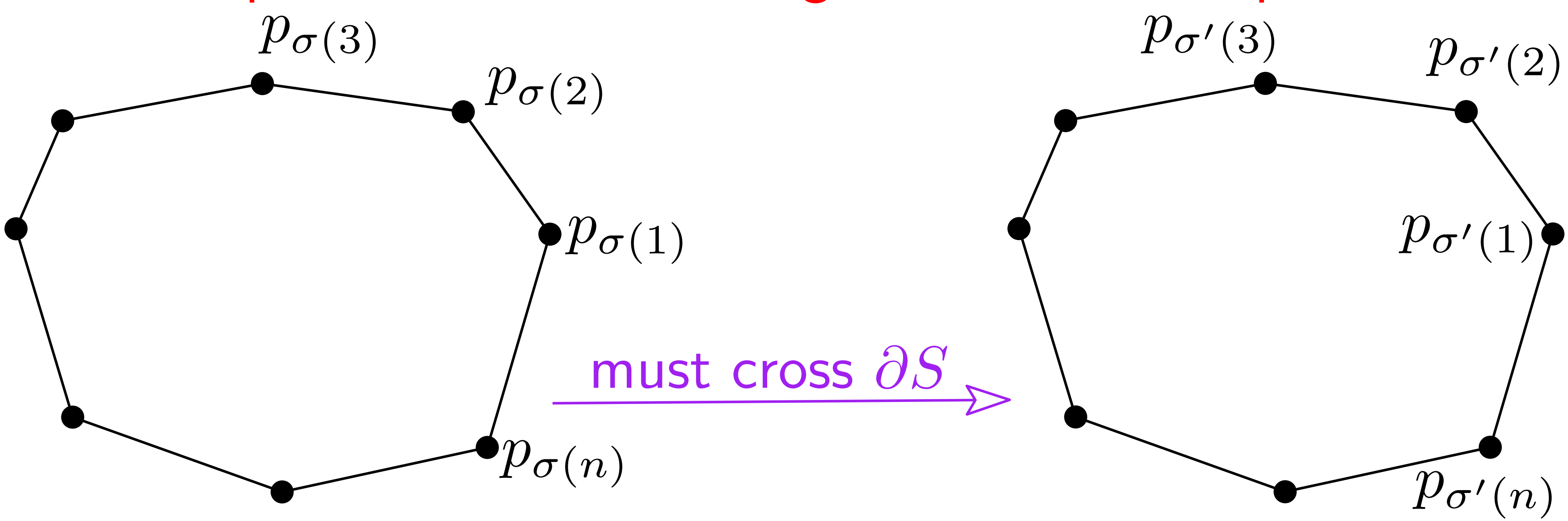
Just prove that S has enough connected components



One connected component per (non circular) permutation

Convex hull

Just prove that S has enough connected components



One connected component per (non circular) permutation

$$\Omega(\log(n-1)!) = \Omega(n \log n)$$

Convex hull

Expected size of the convex hull

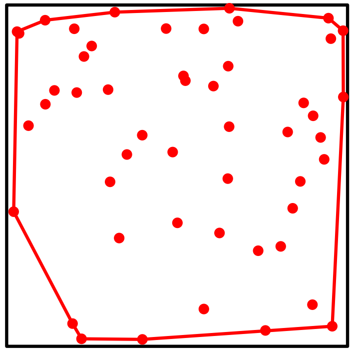
Convex hull

Other results

expected

Expected size of the convex hull

n random points in a square $\Theta(\log n)$



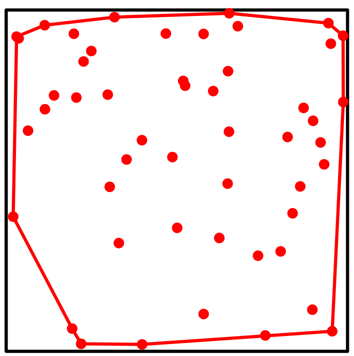
Convex hull

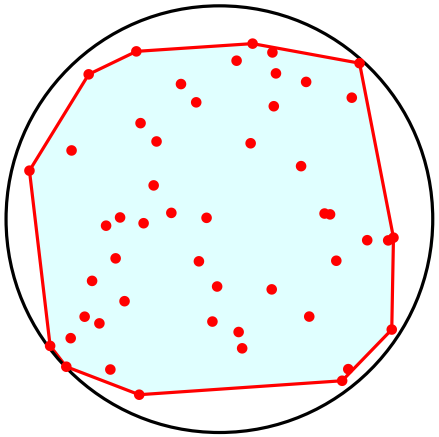
Other results

expected

Expected size of the convex hull

n random points in a square	$\Theta(\log n)$
n random points in a disk	$\Theta(n^{\frac{1}{3}})$





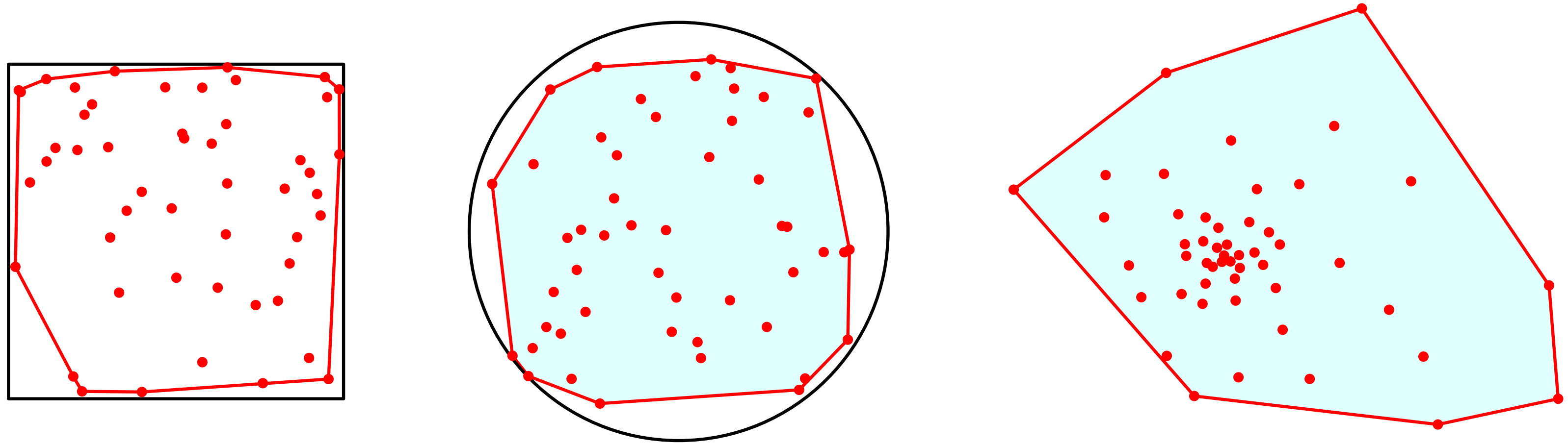
Convex hull

Other results

expected

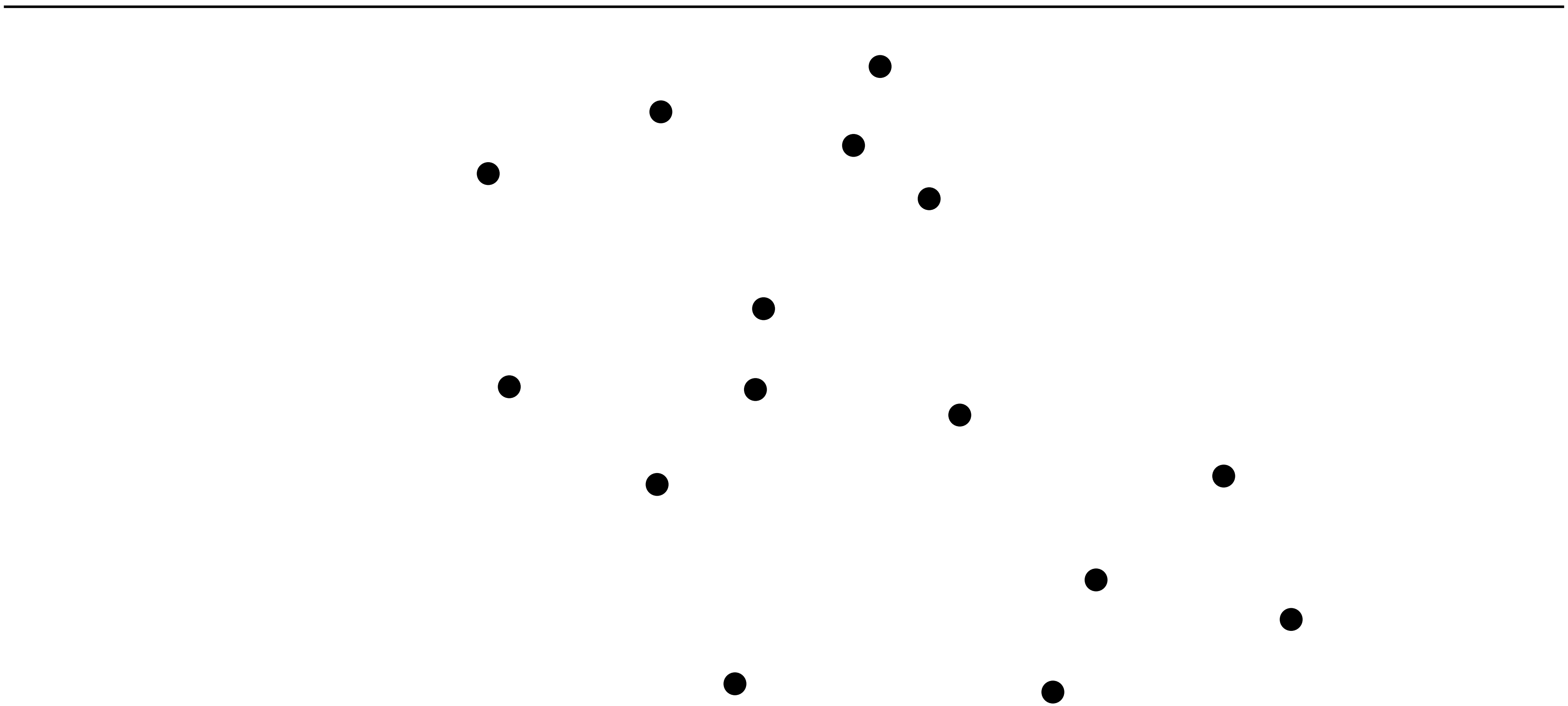
Expected size of the convex hull

n random points in a square	$\Theta(\log n)$
n random points in a disk	$\Theta(n^{\frac{1}{3}})$
n random points, Gaussian distribution	$\Theta(\sqrt{\log n})$



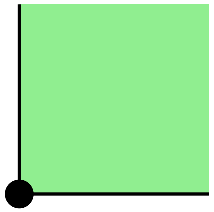
Maximal points

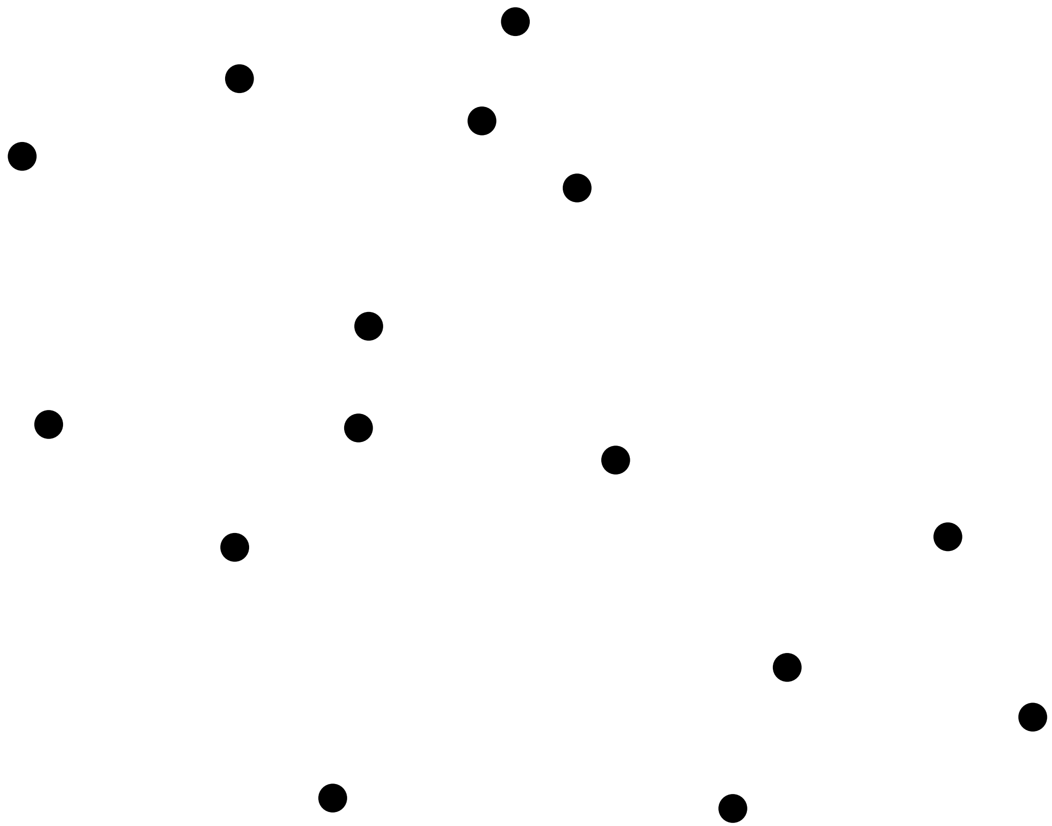
In a set of points



Maximal points

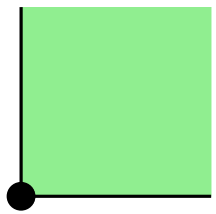
In a set of points

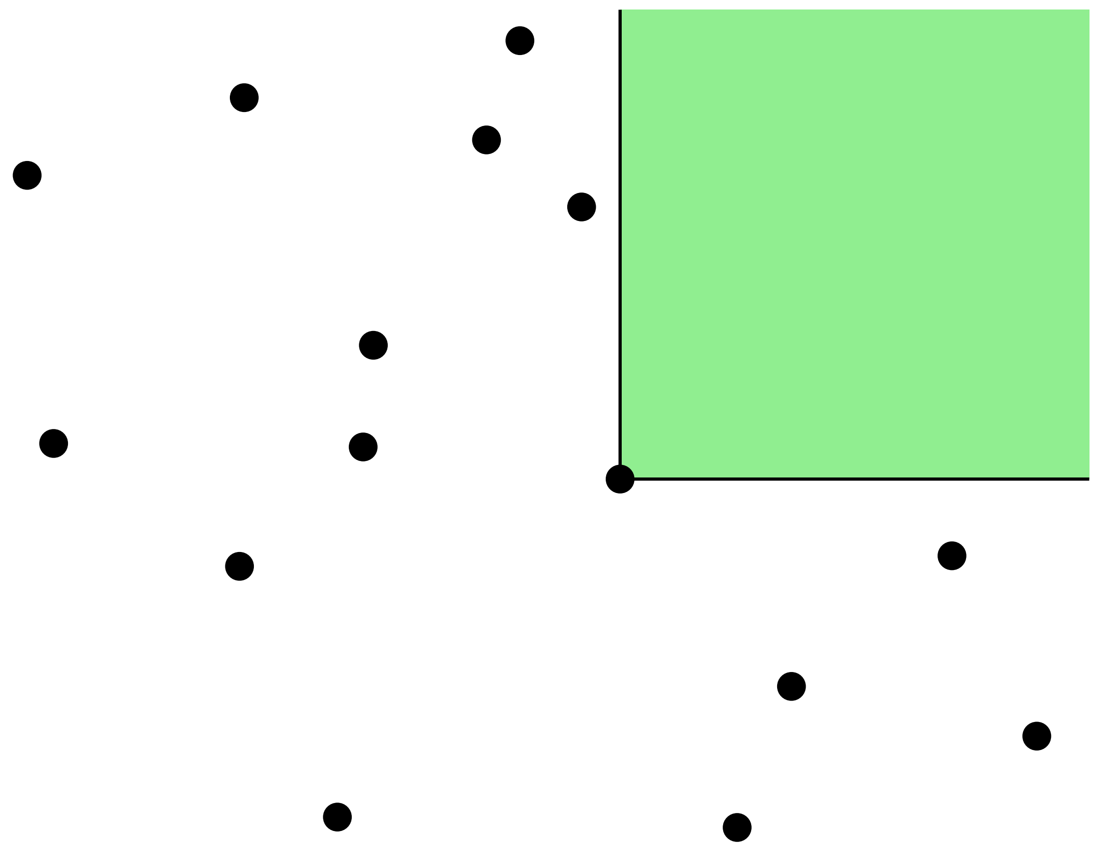
p is NE maximal if  empty



Maximal points

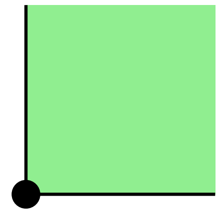
In a set of points

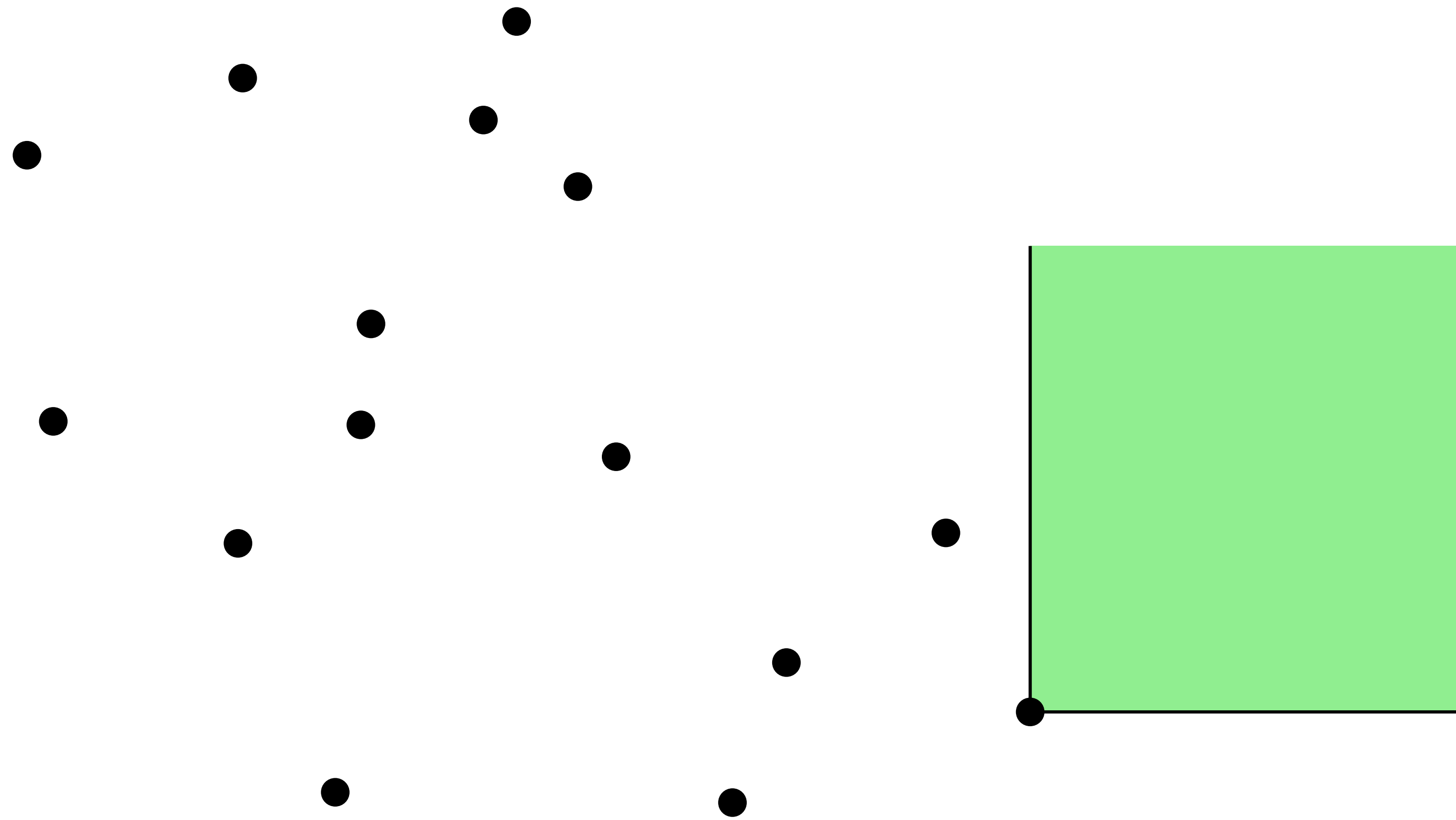
p is NE maximal if  empty



Maximal points

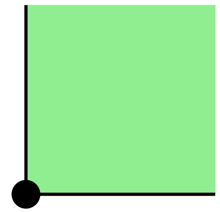
In a set of points

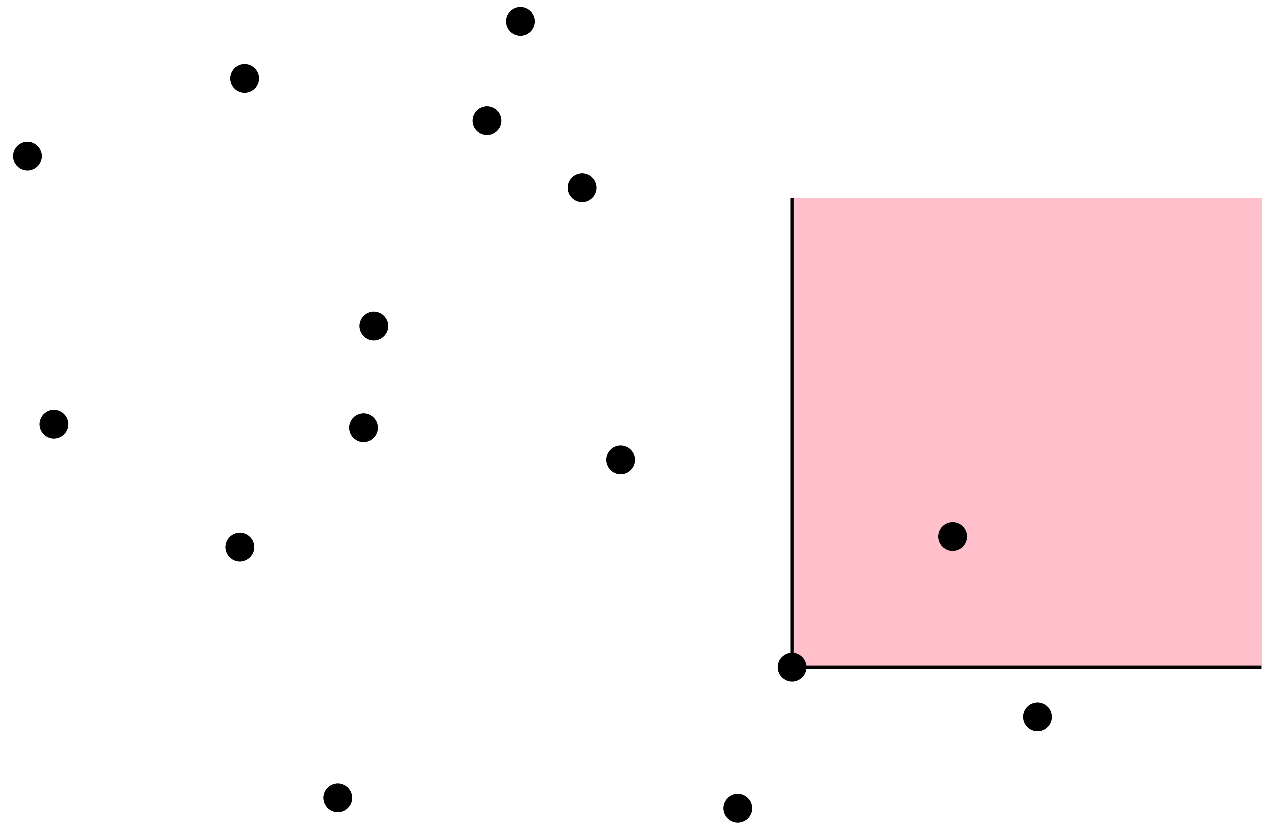
p is NE maximal if  empty



Maximal points

In a set of points

p is NE maximal if  empty



Maximal points

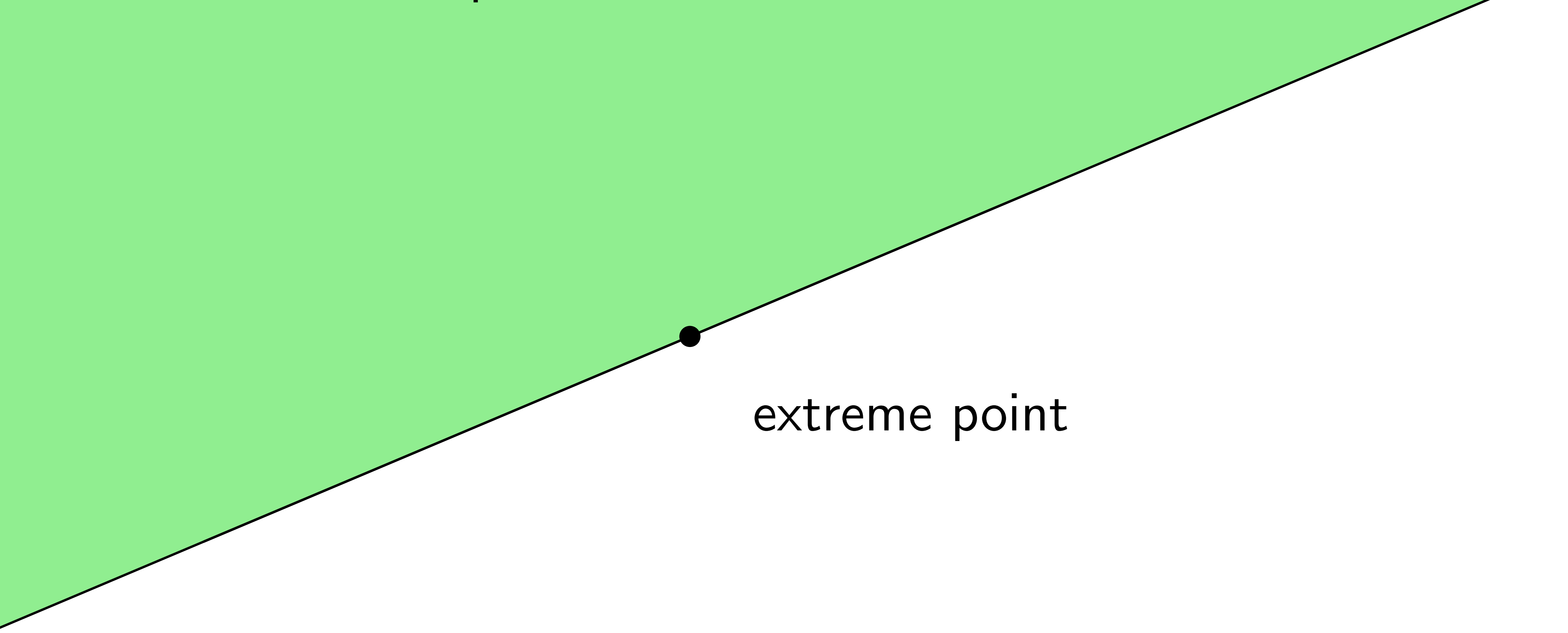
contains extreme points

An extreme point is NE, NW,SW, or SE maximal

Maximal points

contains extreme points

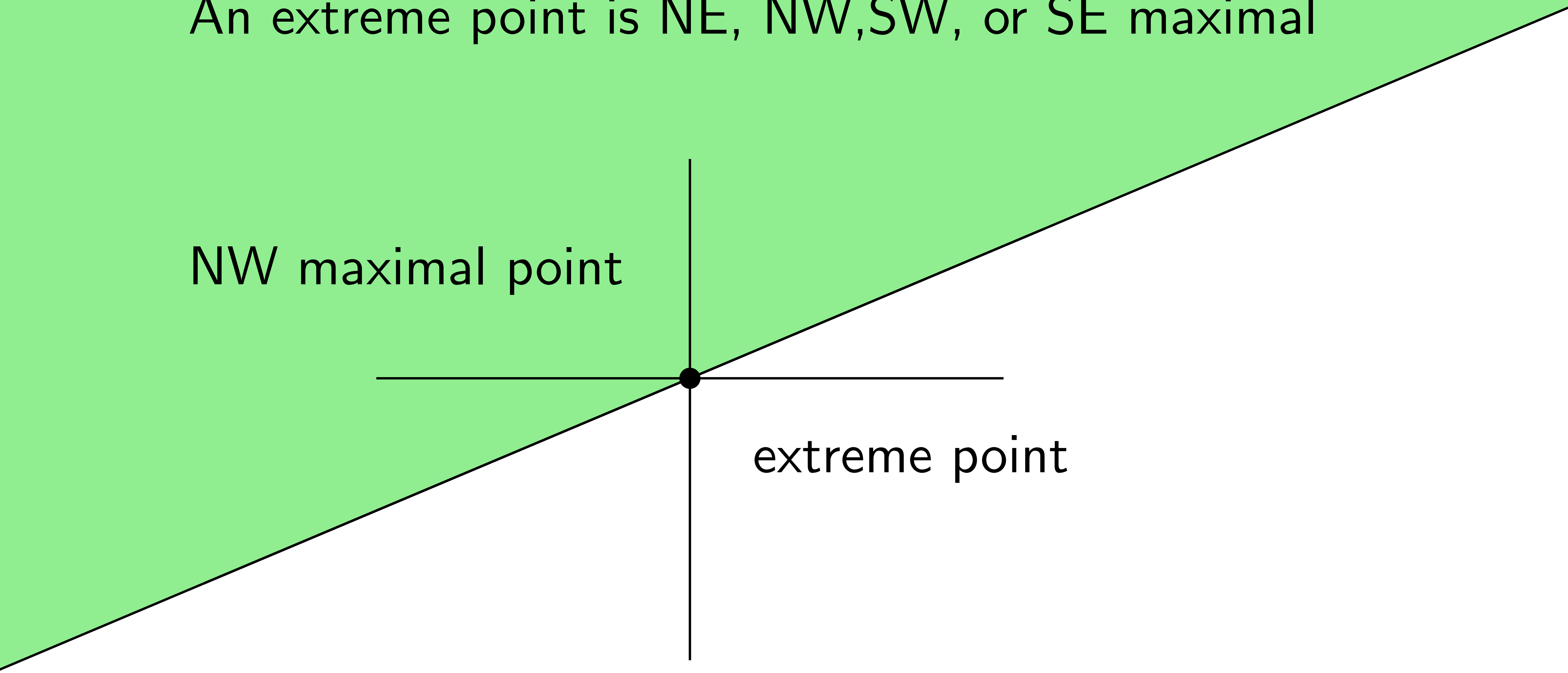
An extreme point is NE, NW,SW, or SE maximal



Maximal points

contains extreme points

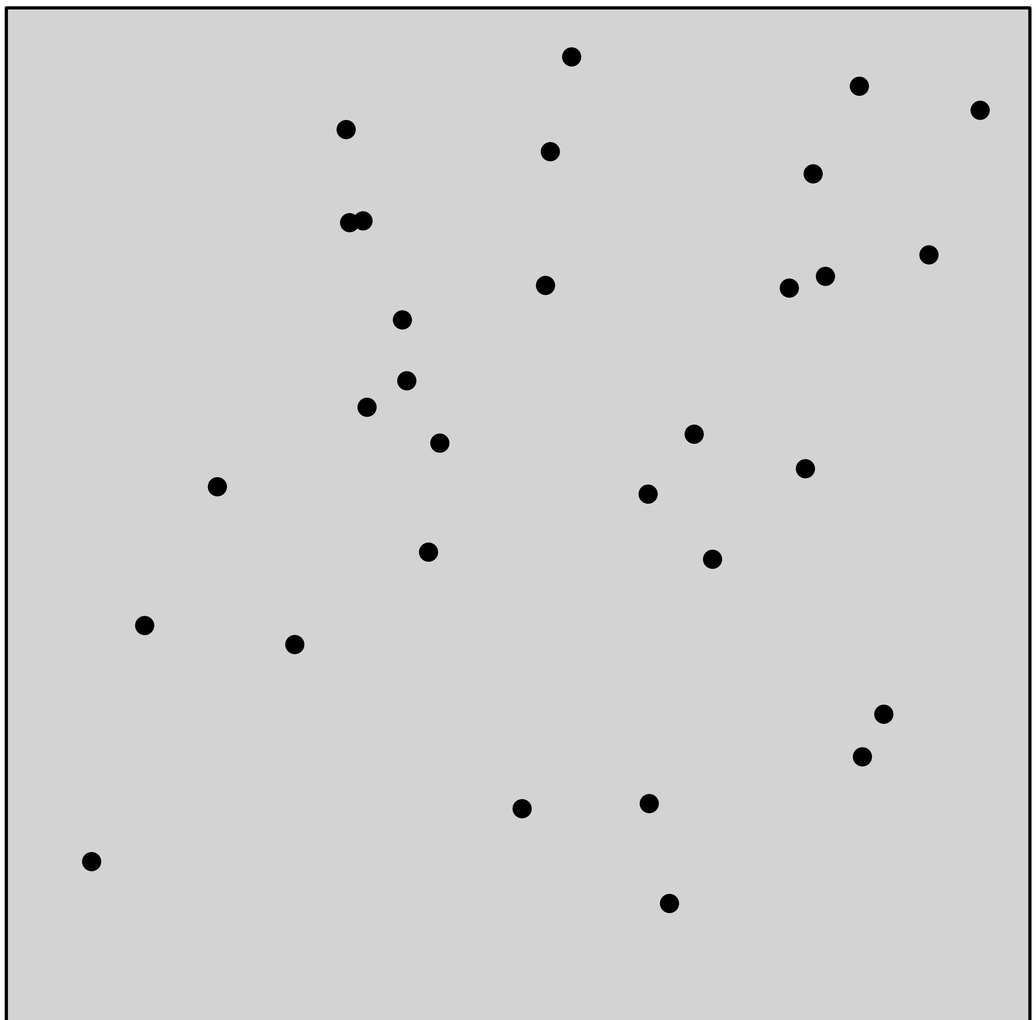
An extreme point is NE, NW,SW, or SE maximal



Maximal points

expected number ?

n random points in a square   NW maximal ?

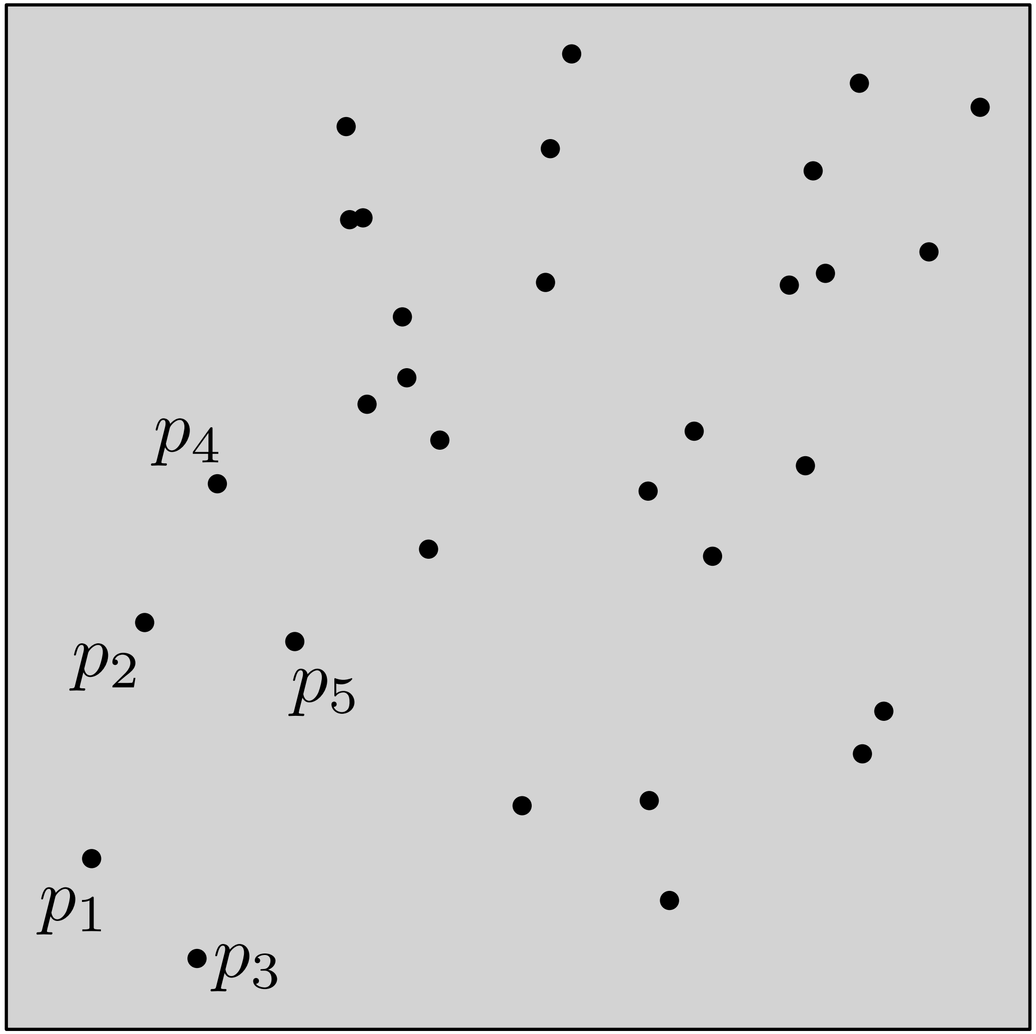


Maximal points

expected number ?

n random points in a square

NW maximal ?

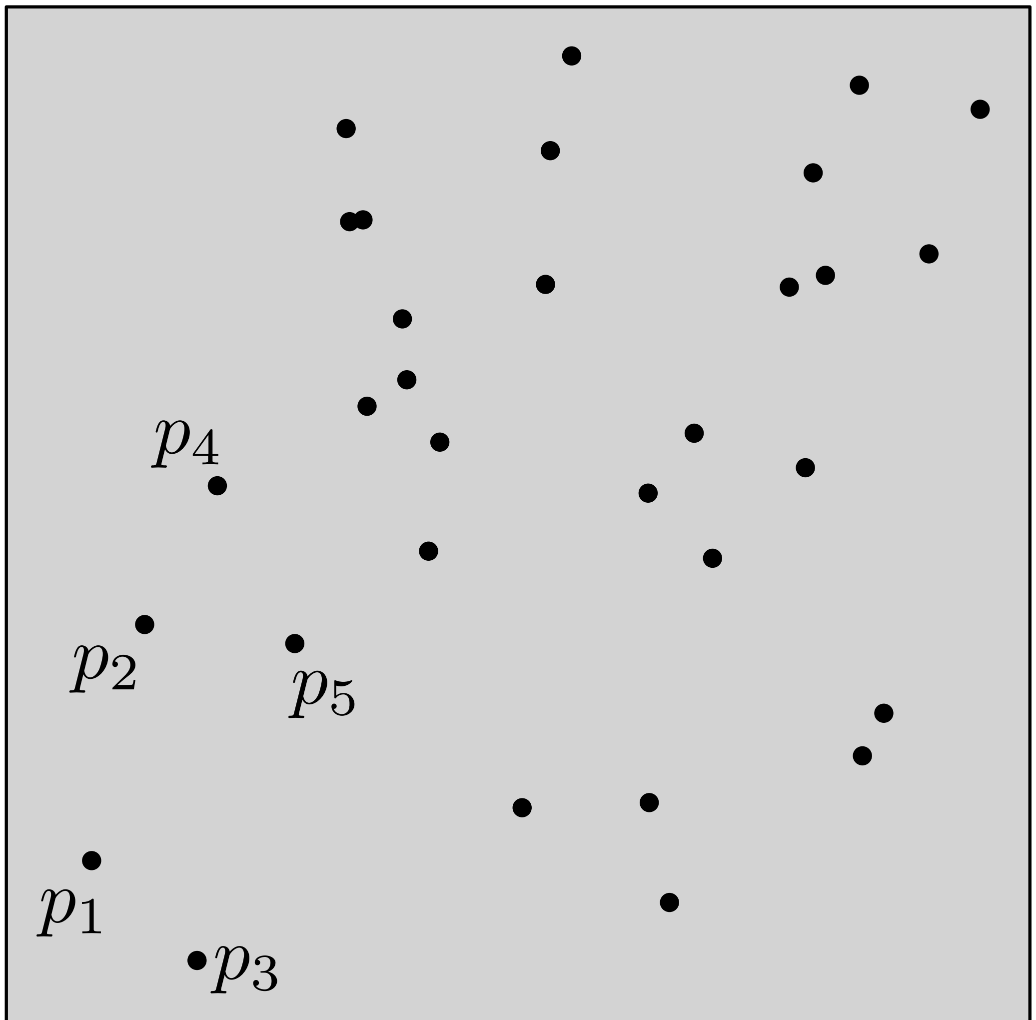


Number by increasing x

Maximal points

expected number ?

n random points in a square



NW maximal ?

Number by increasing x

p_i maximal

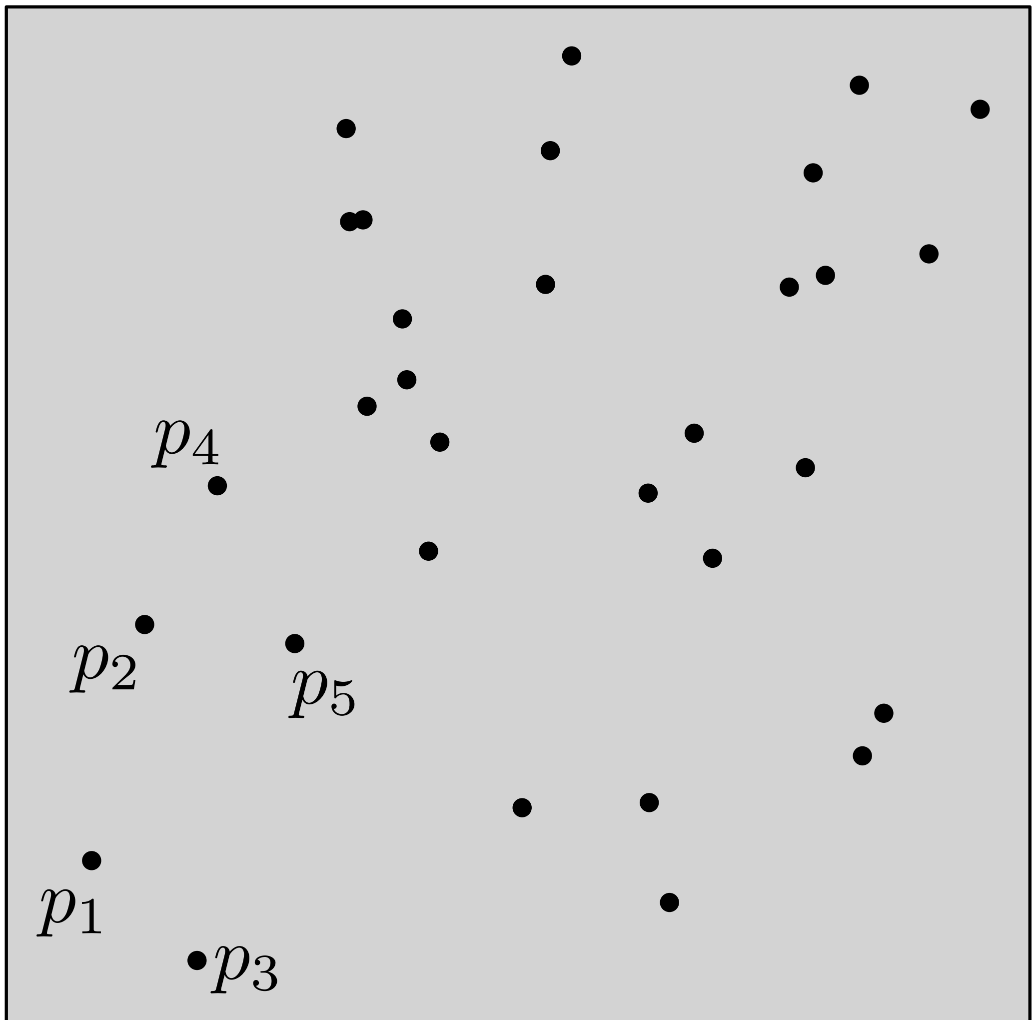
if $y_{p_i} > y_{p_j}$ for $j < i$

Maximal points

expected number ?

n random points in a square

NW maximal ?



Number by increasing x

p_i maximal

if $y_{p_i} > y_{p_j}$ for $j < i$

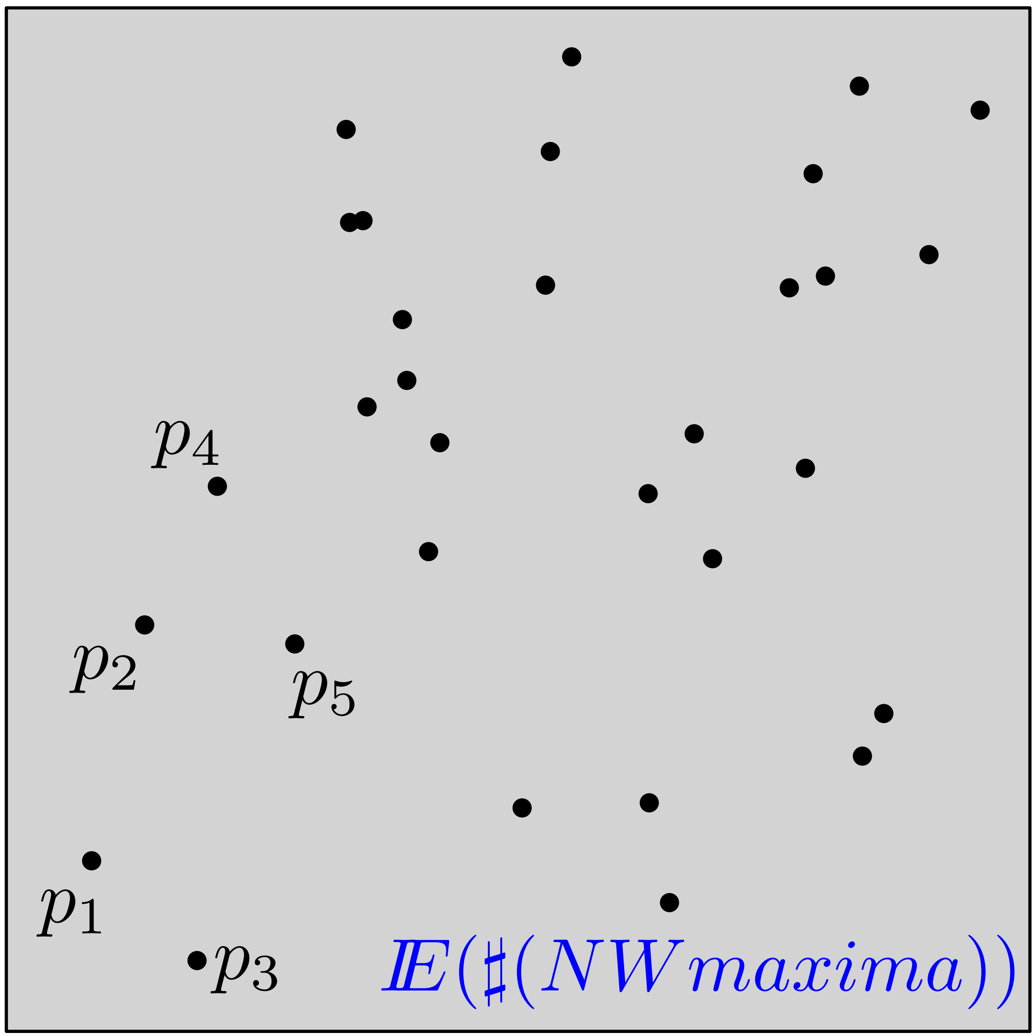
it happens with proba $\frac{1}{j}$

Maximal points

expected number ?

n random points in a square

NW maximal ?



Number by increasing x

p_i maximal

if $y_{p_i} > y_{p_j}$ for $j < i$

it happens with proba $\frac{1}{j}$

$$\mathbb{E}(\sharp(NW\,maxima)) = \sum \frac{1}{j} \simeq \log n$$

Maximal points

expected number ?

n random points in a square

NW maximal ?

Number by increasing x

p_i maximal

if $y_{p_i} > y_{p_j}$ for $j < i$

it happens with proba $\frac{1}{j}$

Convex hull

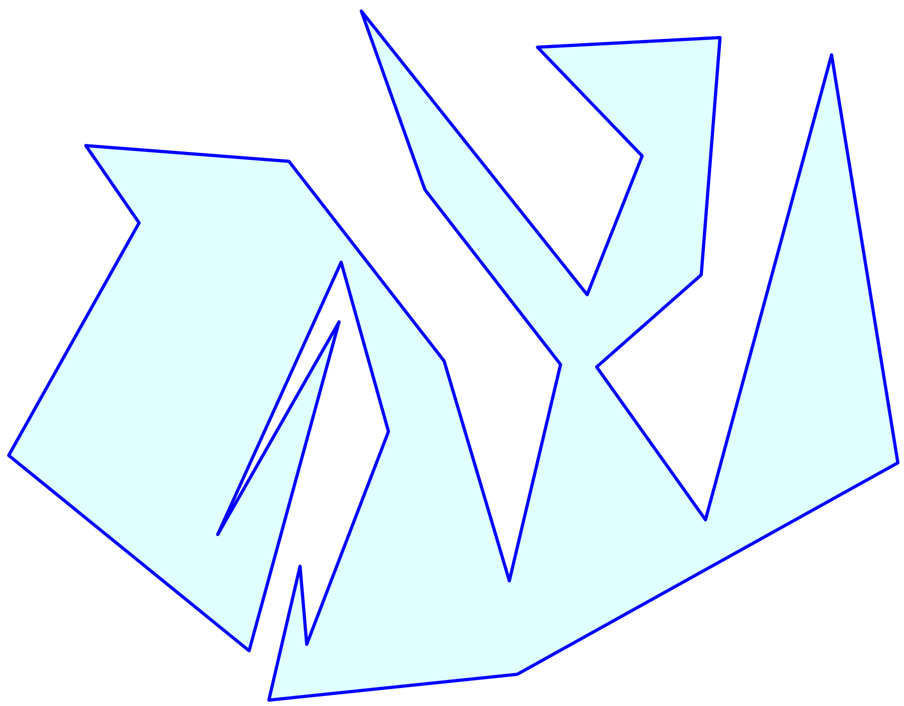
$$\mathbb{E}(\sharp(CH)) \leq \mathbb{E}(\sharp(maximas)) \simeq 4 \log n$$

p_4 p_2 p_5 p_1 p_3

$$\mathbb{E}(\sharp(NW\,maxima)) = \sum \frac{1}{j} \simeq \log n$$

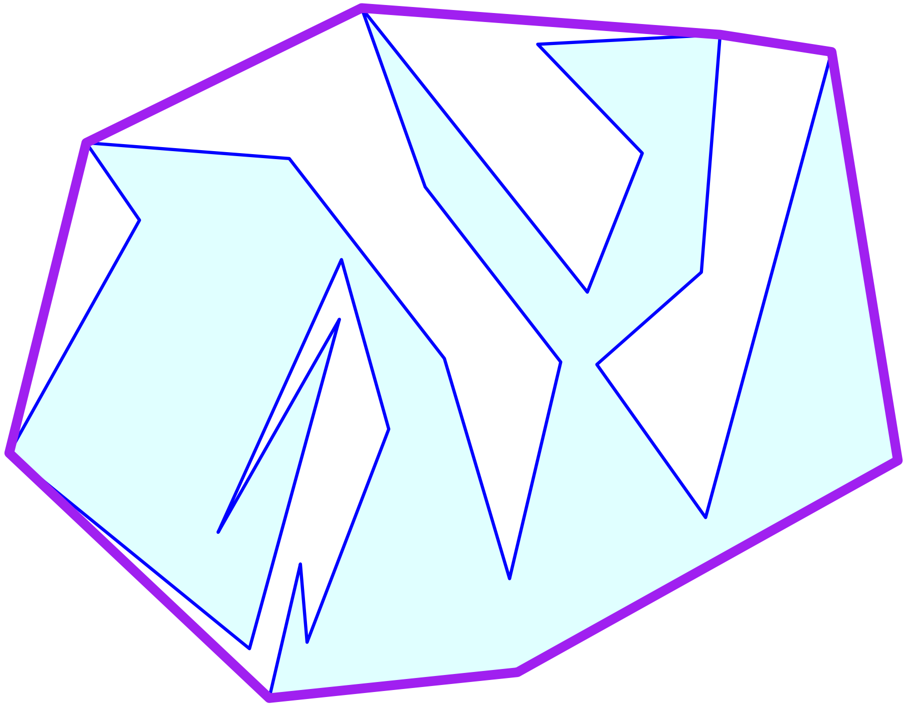
Convex hull

Simple polygon



Convex hull

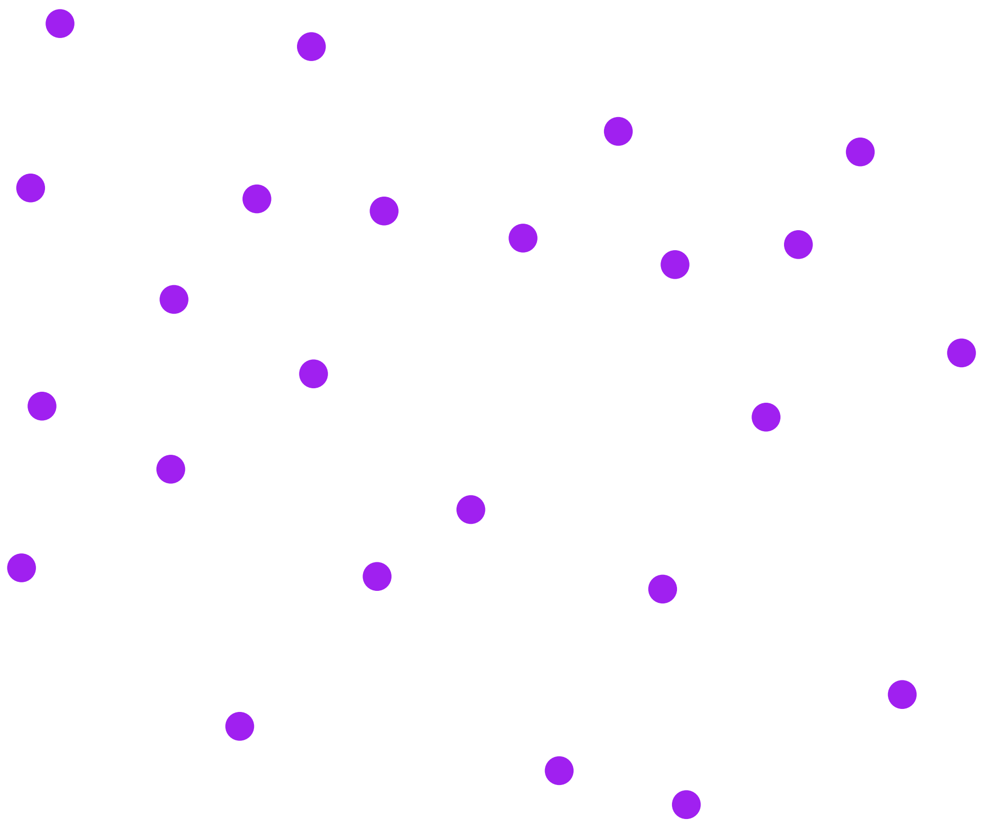
Simple polygon



$O(n)$

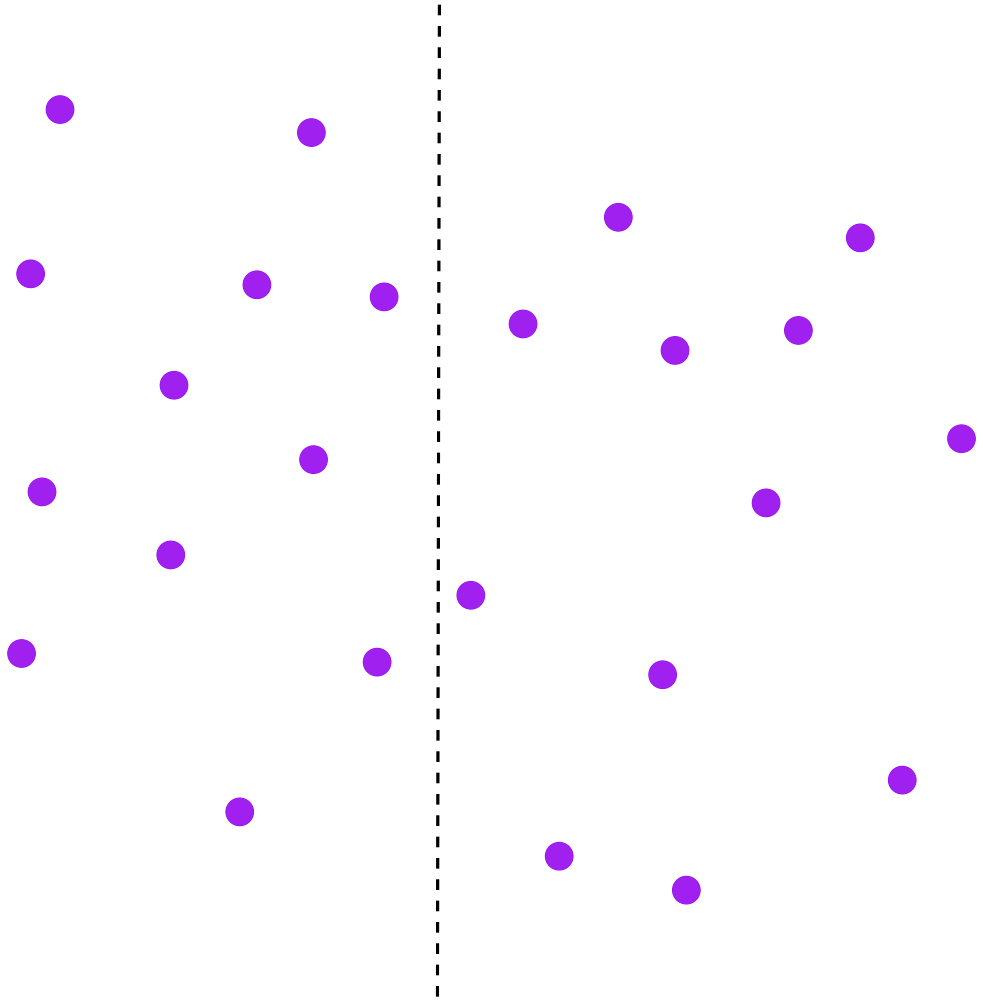
Convex hull

Divide and conquer



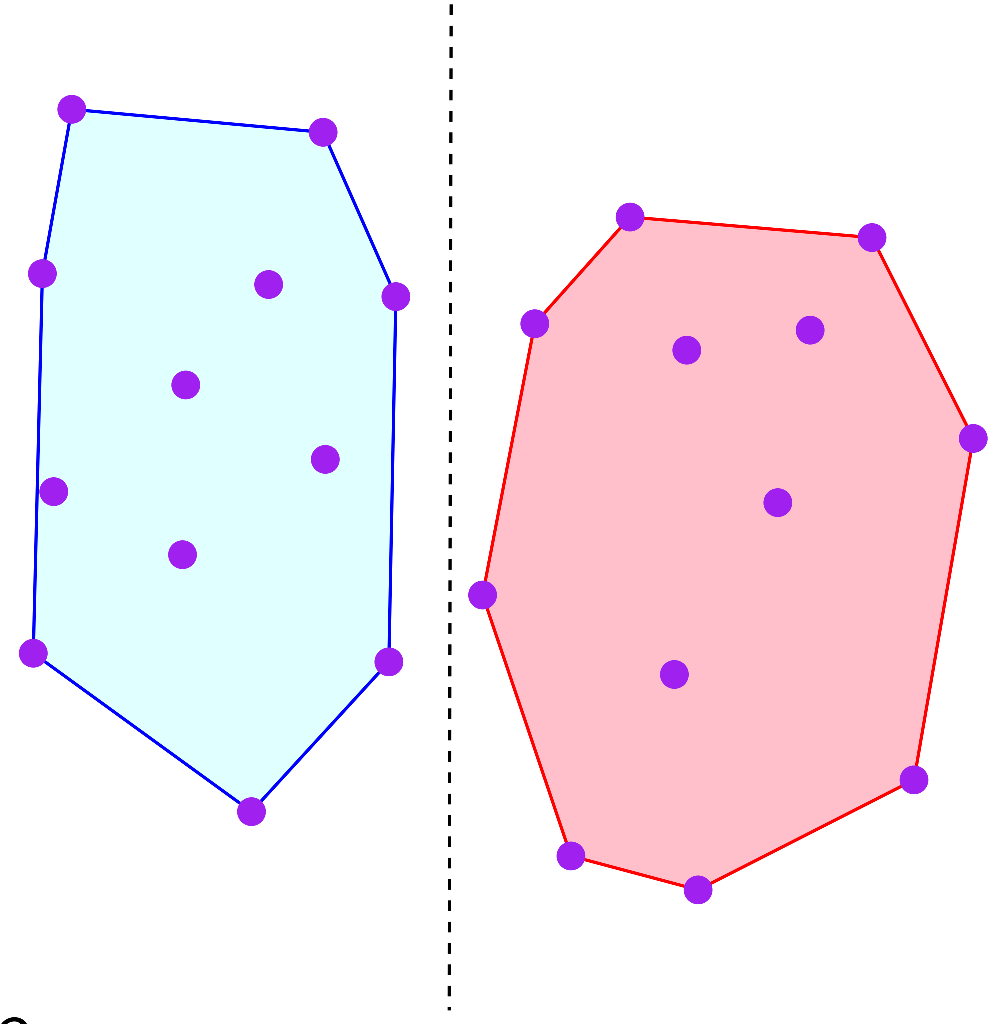
Convex hull

Divide and conquer



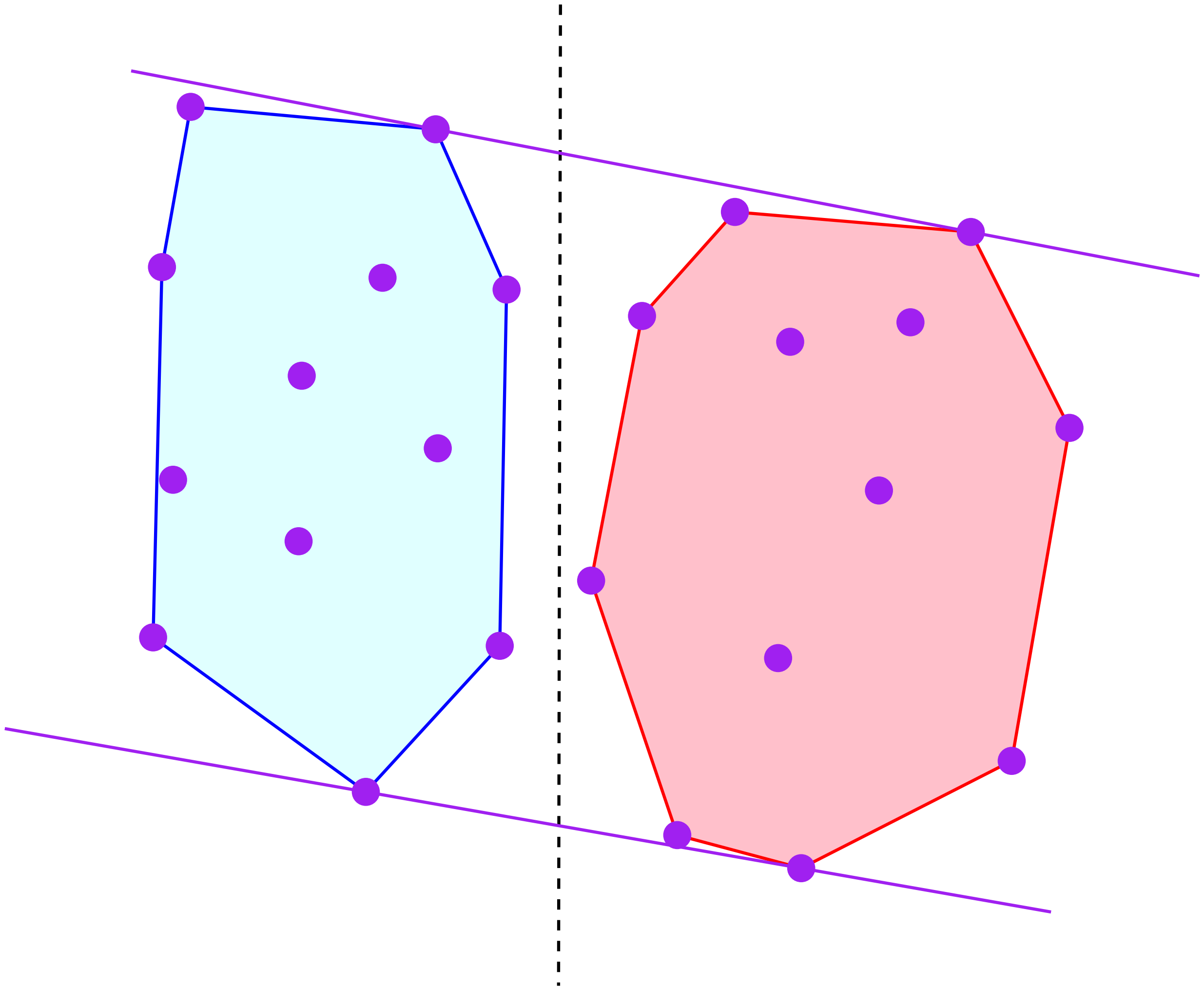
Convex hull

Divide and conquer



Convex hull

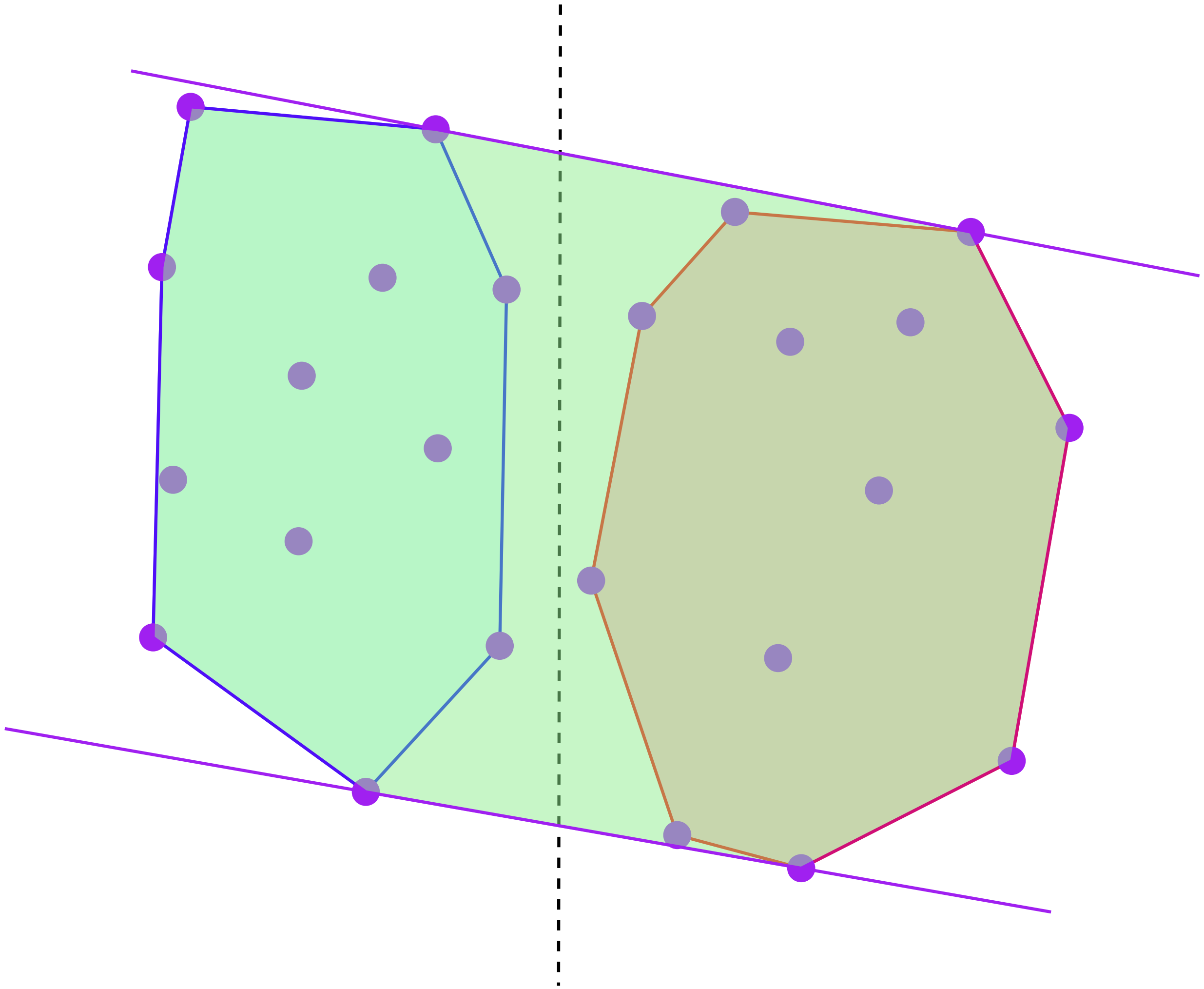
Divide and conquer



Convex hull

Other results

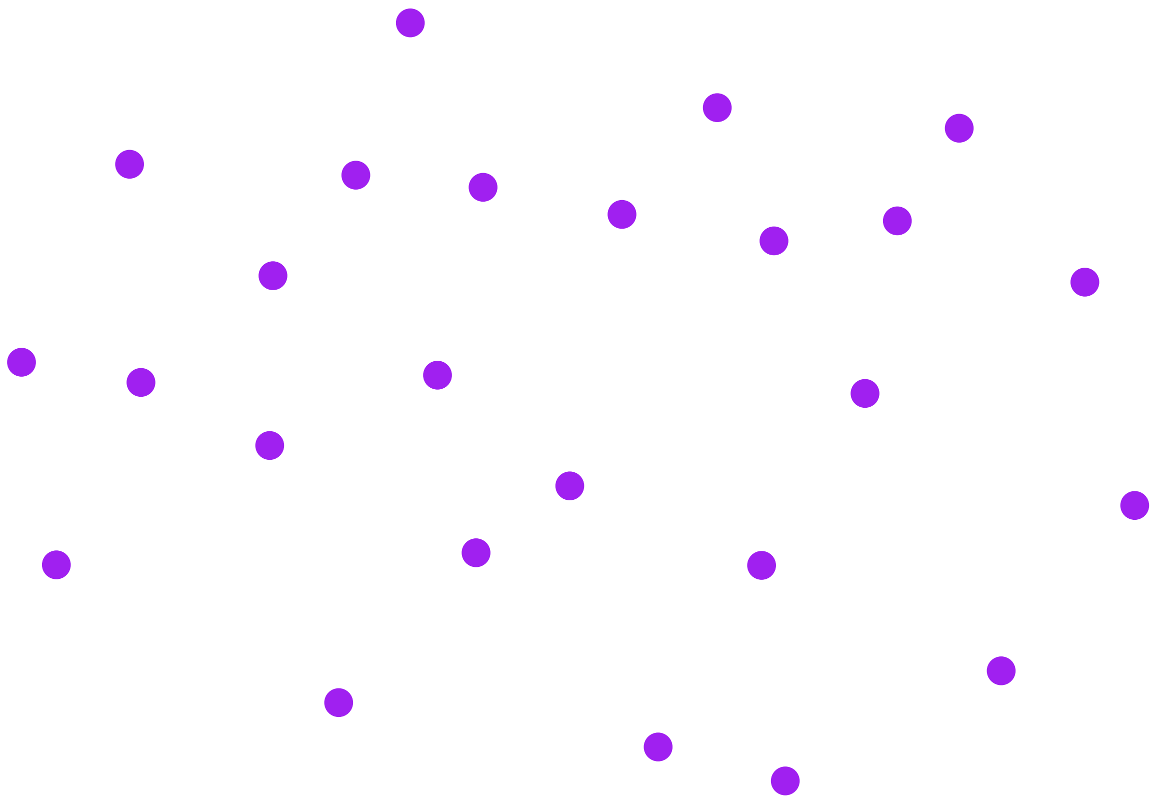
Divide and conquer



$O(n \log n)$

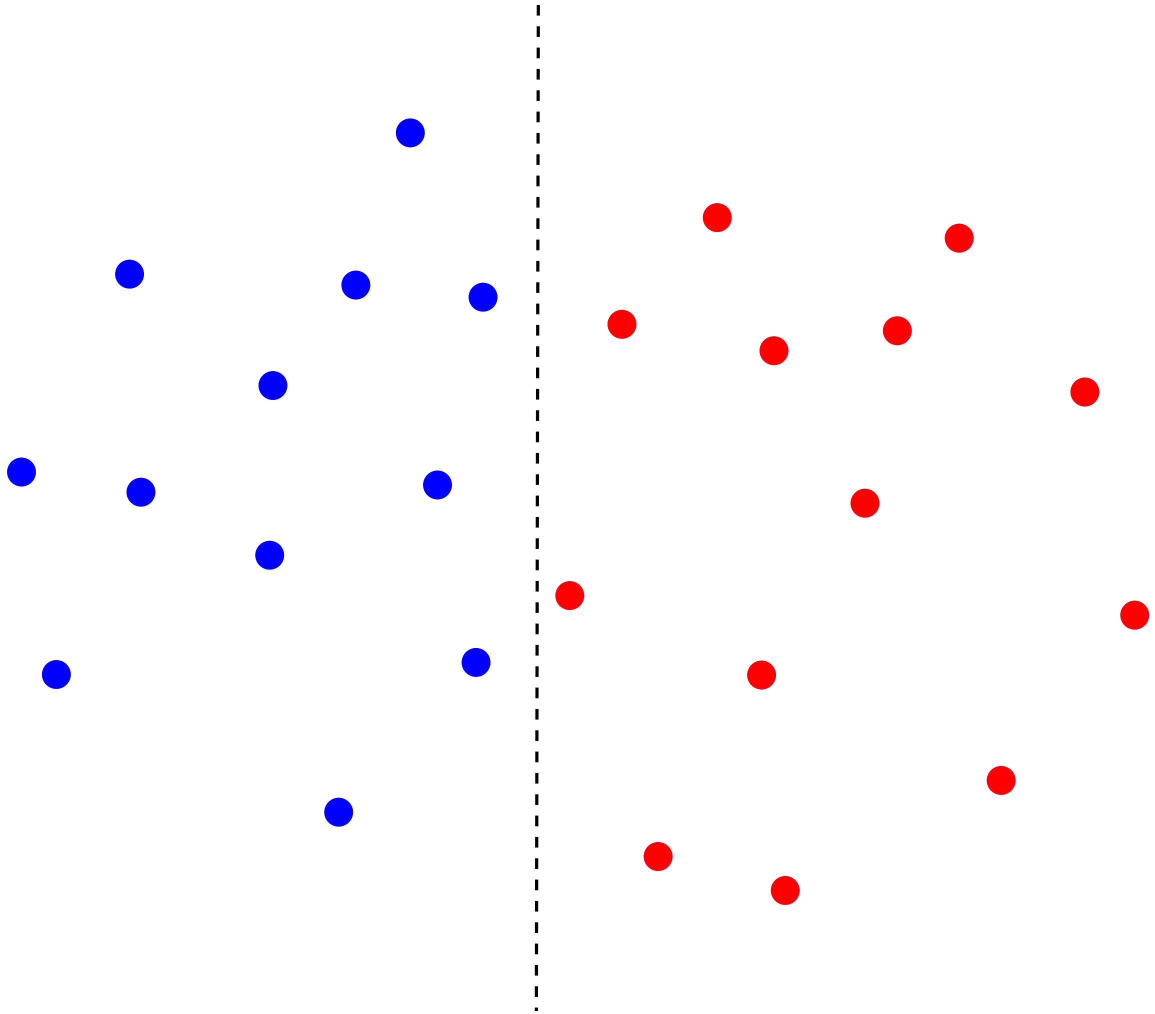
Convex hull

Mariage before conquest



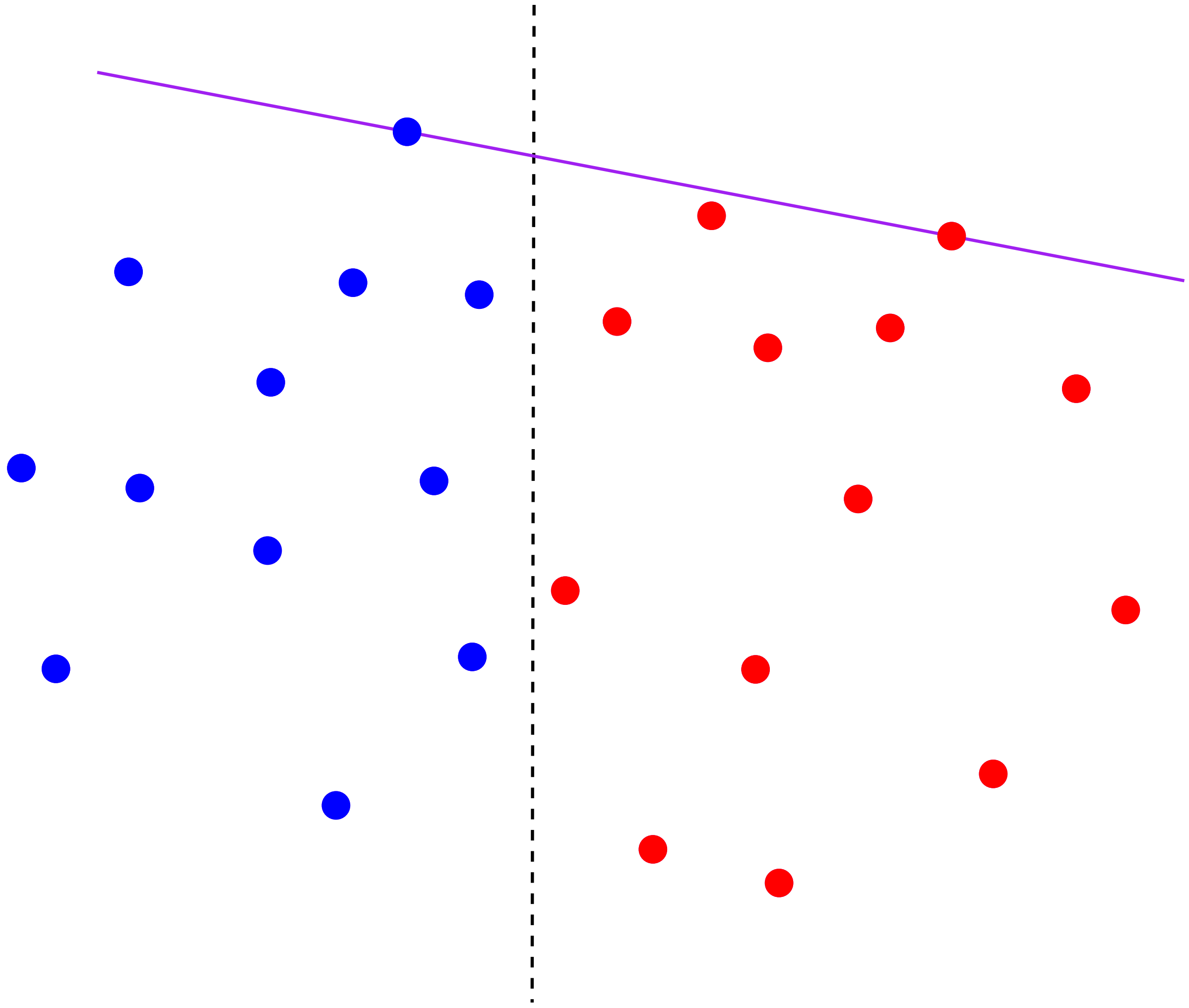
Convex hull

Mariage before conquest



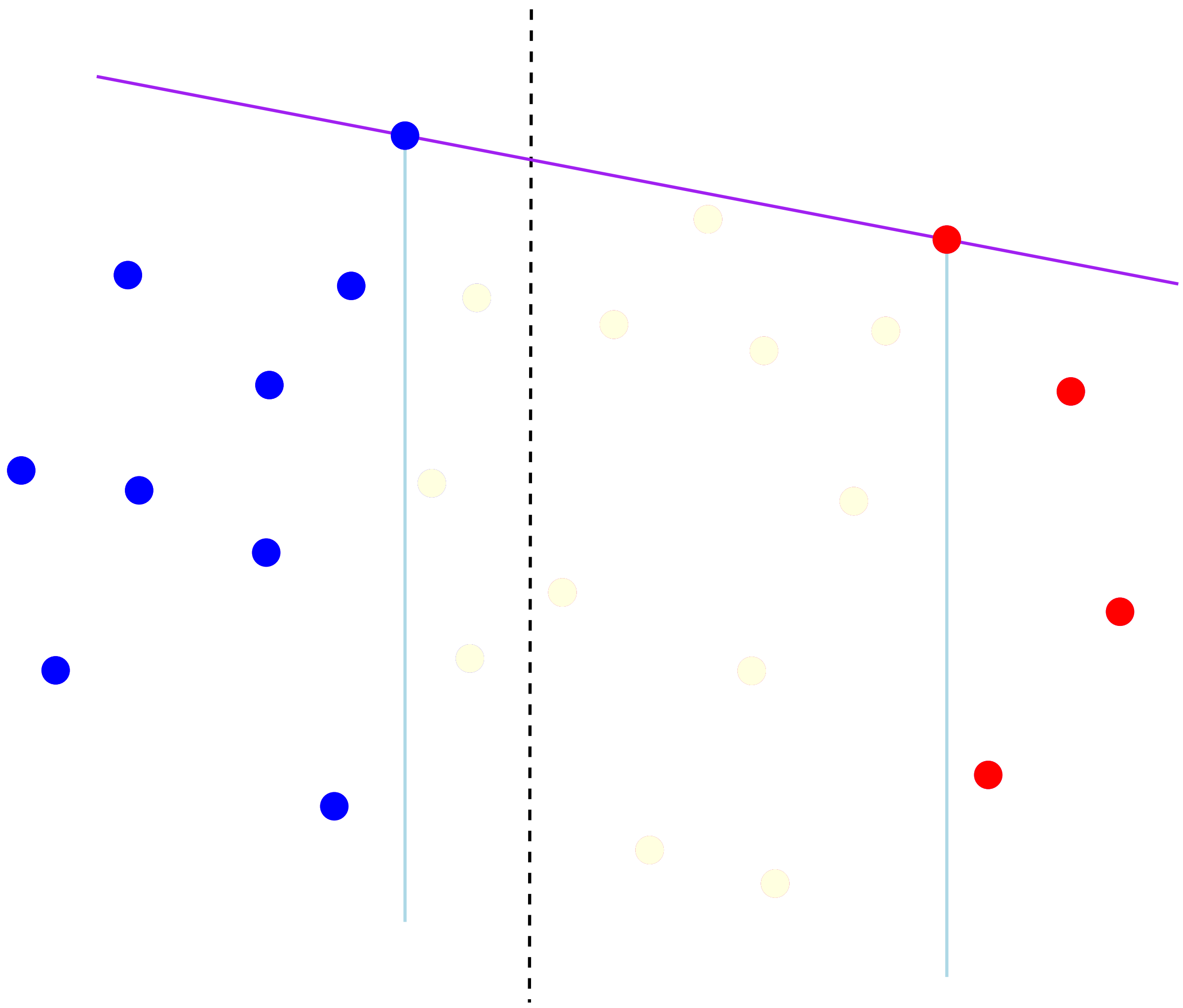
Convex hull

Mariage before conquest



Convex hull

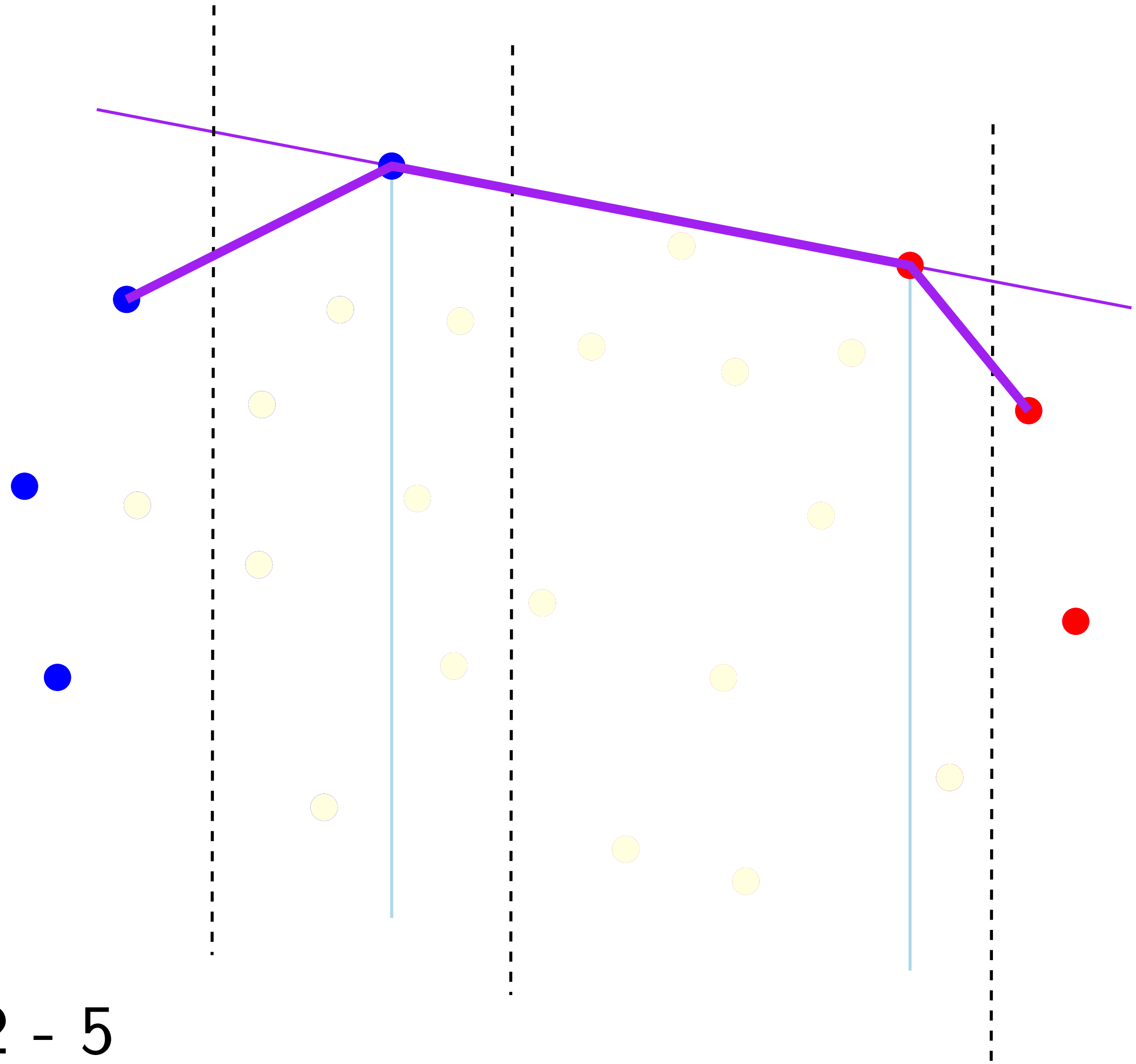
Mariage before conquest



Convex hull

Other results

Mariage before conquest

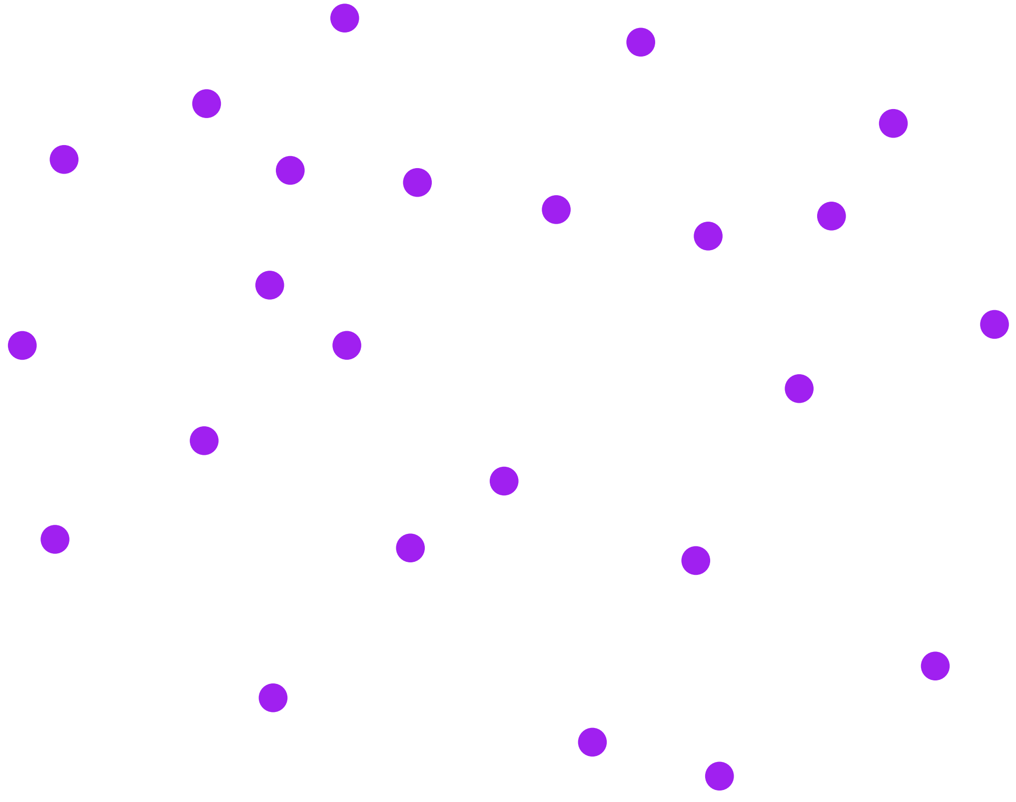


recursion

$O(n \log h)$

Convex hull

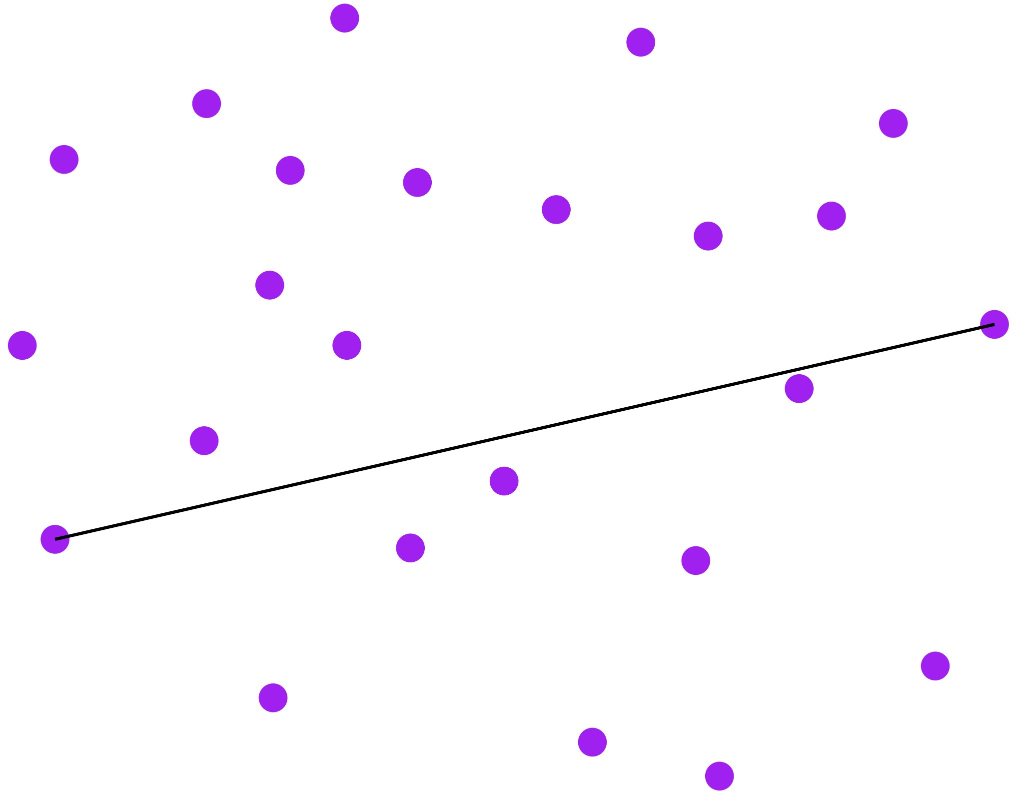
Quickhull



Convex hull

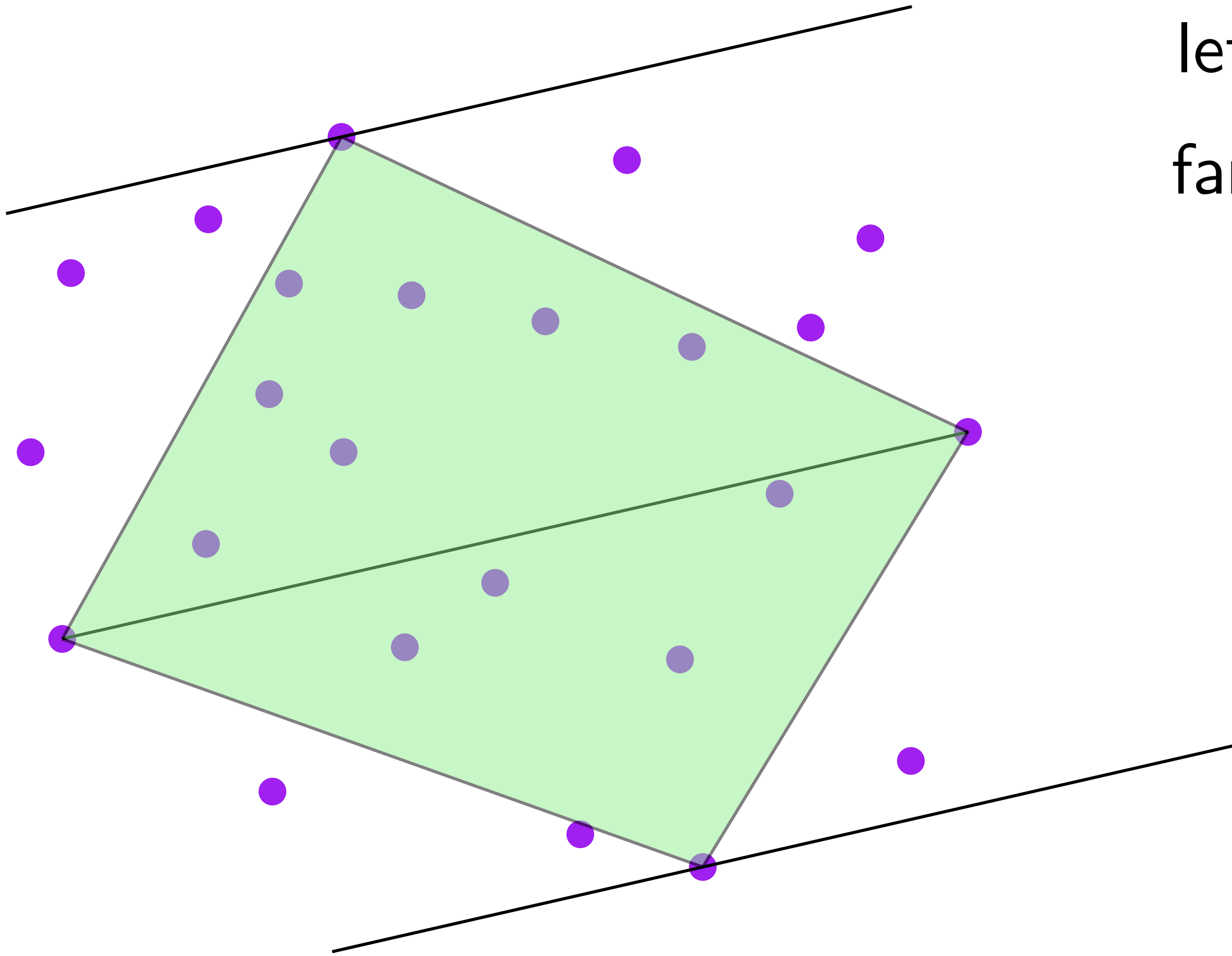
Quickhull

leftmost rightmost



Convex hull

Quickhull

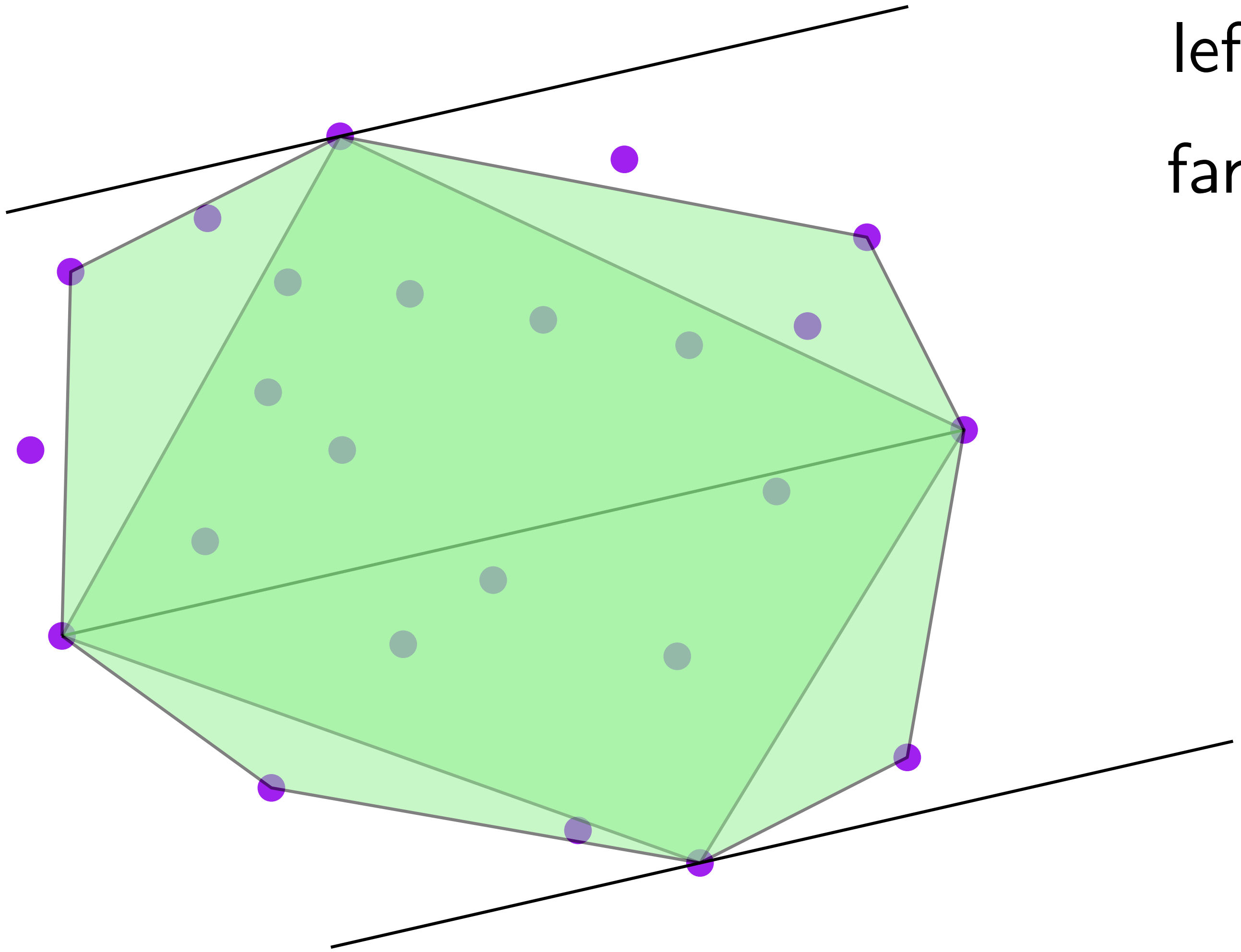


leftmost rightmost

farthest points

Convex hull

Quickhull

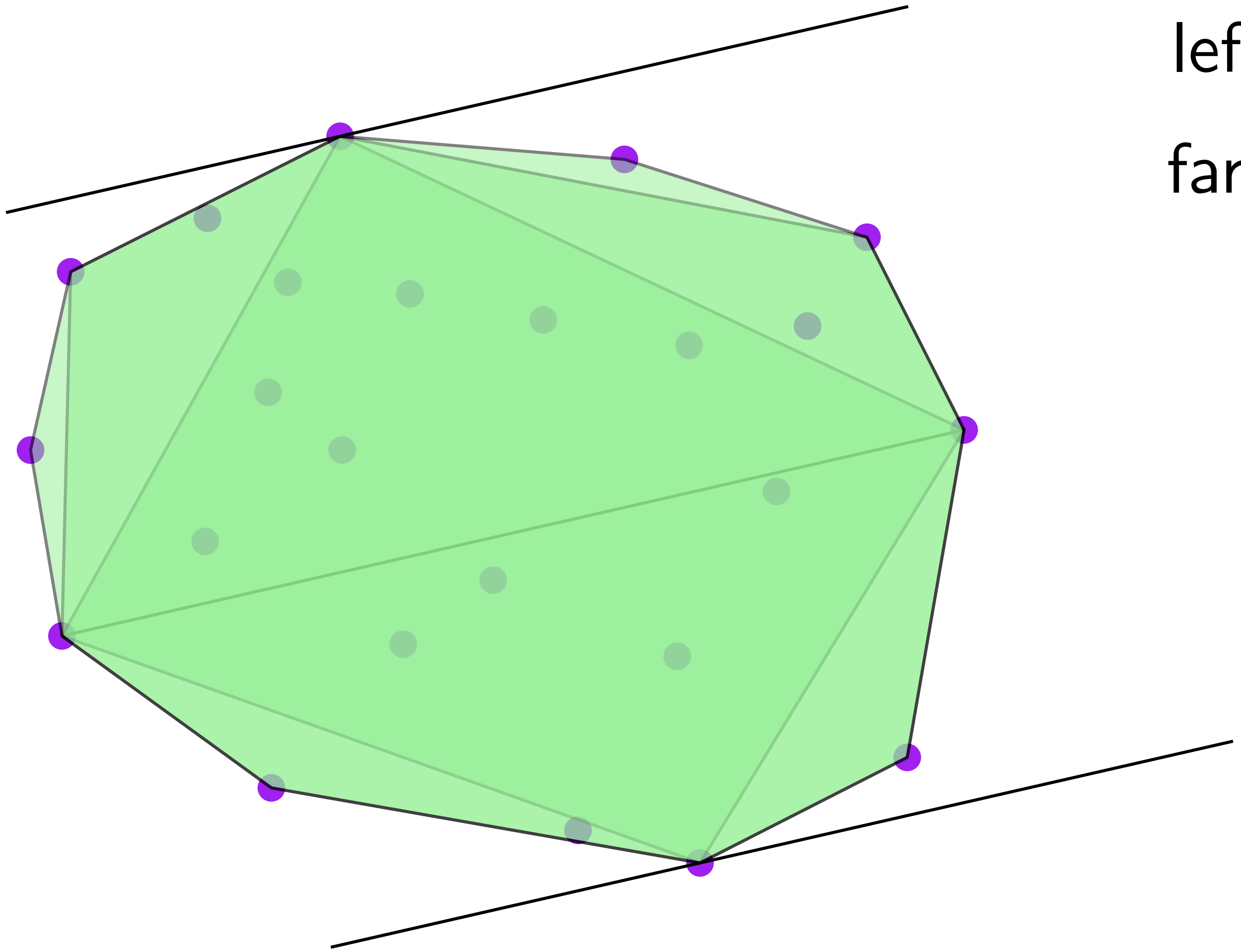


leftmost rightmost

farthest points

Convex hull

Quickhull

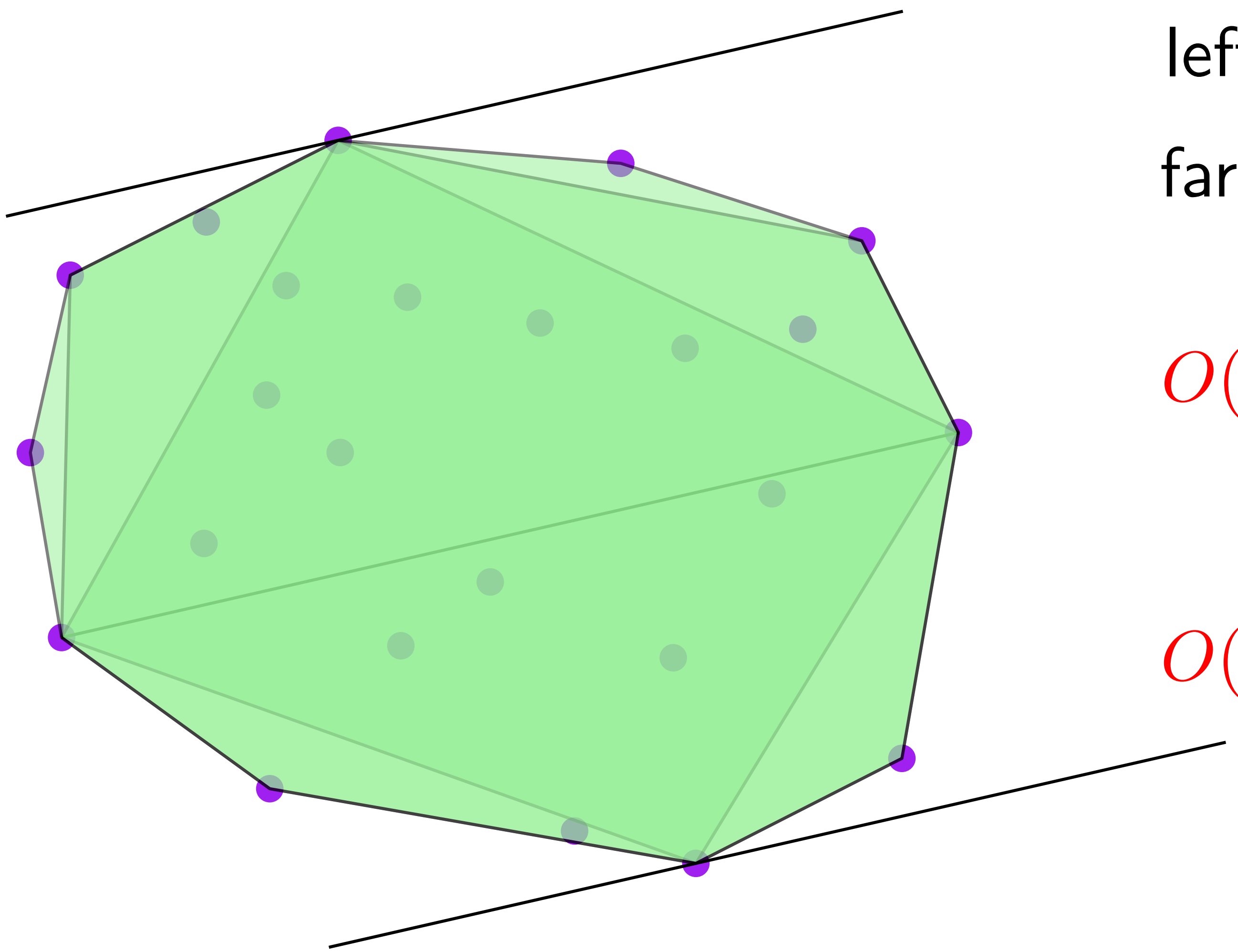


leftmost rightmost

farthest points

Convex hull

Quickhull



leftmost rightmost

farthest points

$O(n \log n)$ expected

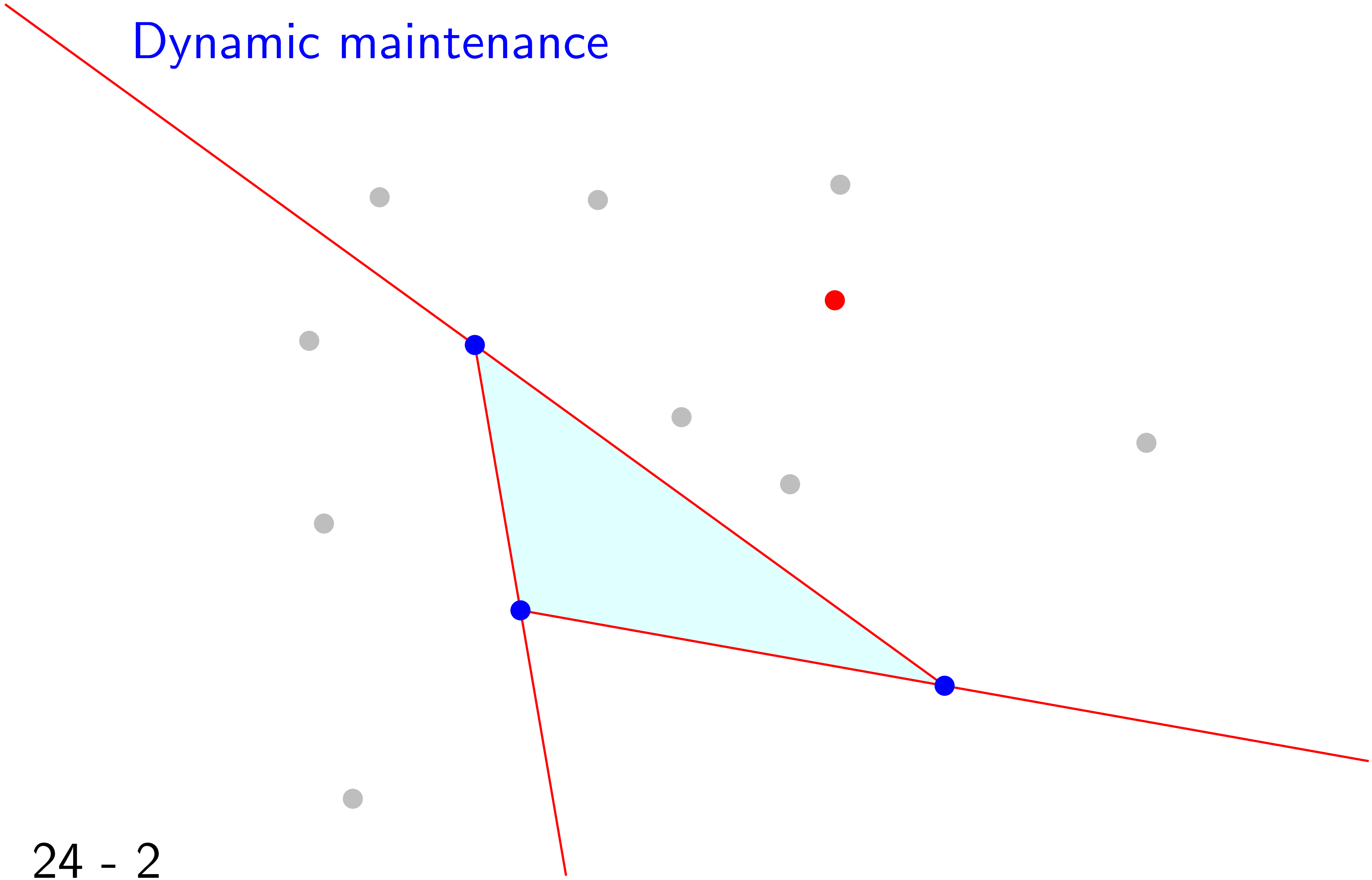
$O(n^2)$ worst-case

Convex hull

Dynamic maintenance

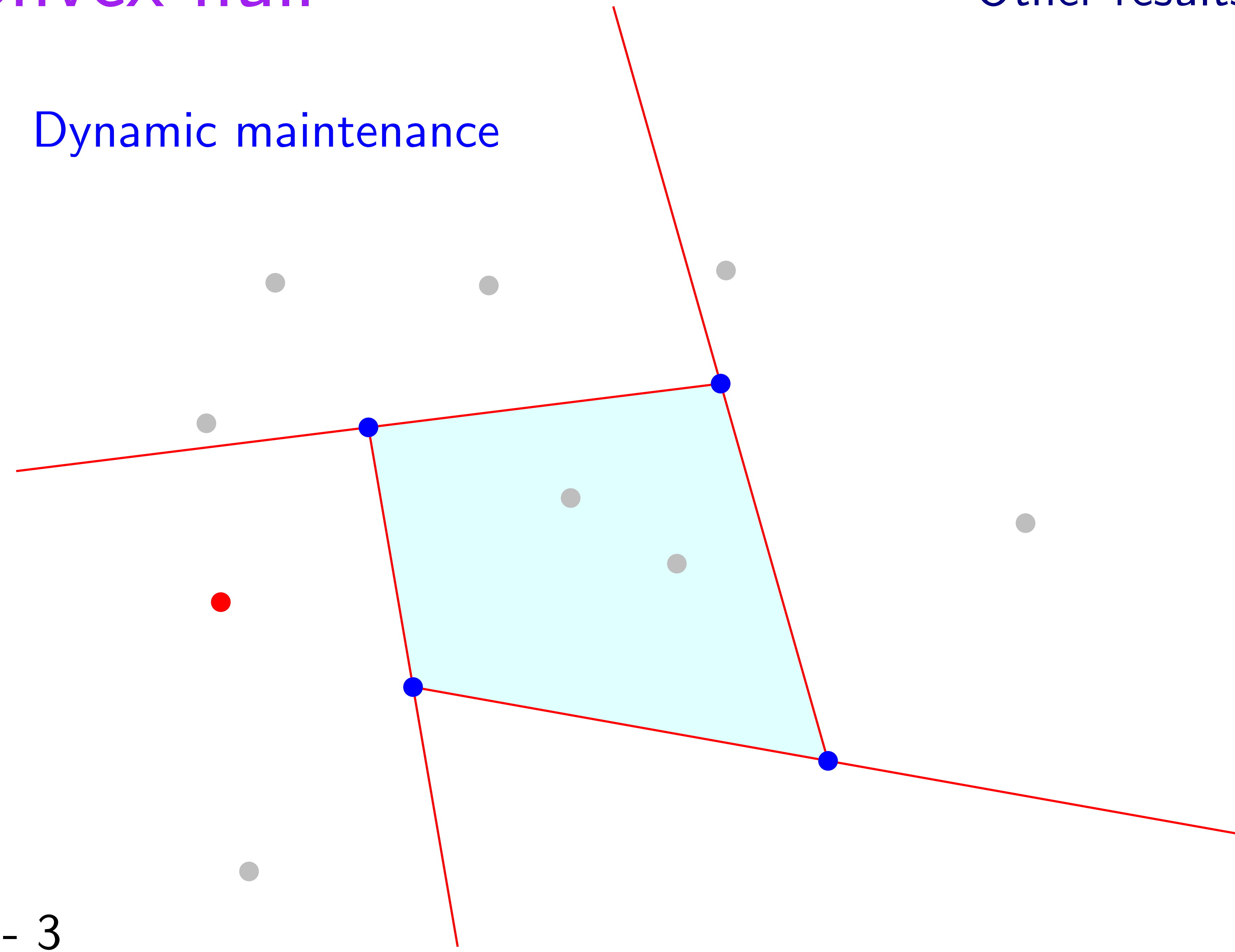
Convex hull

Dynamic maintenance



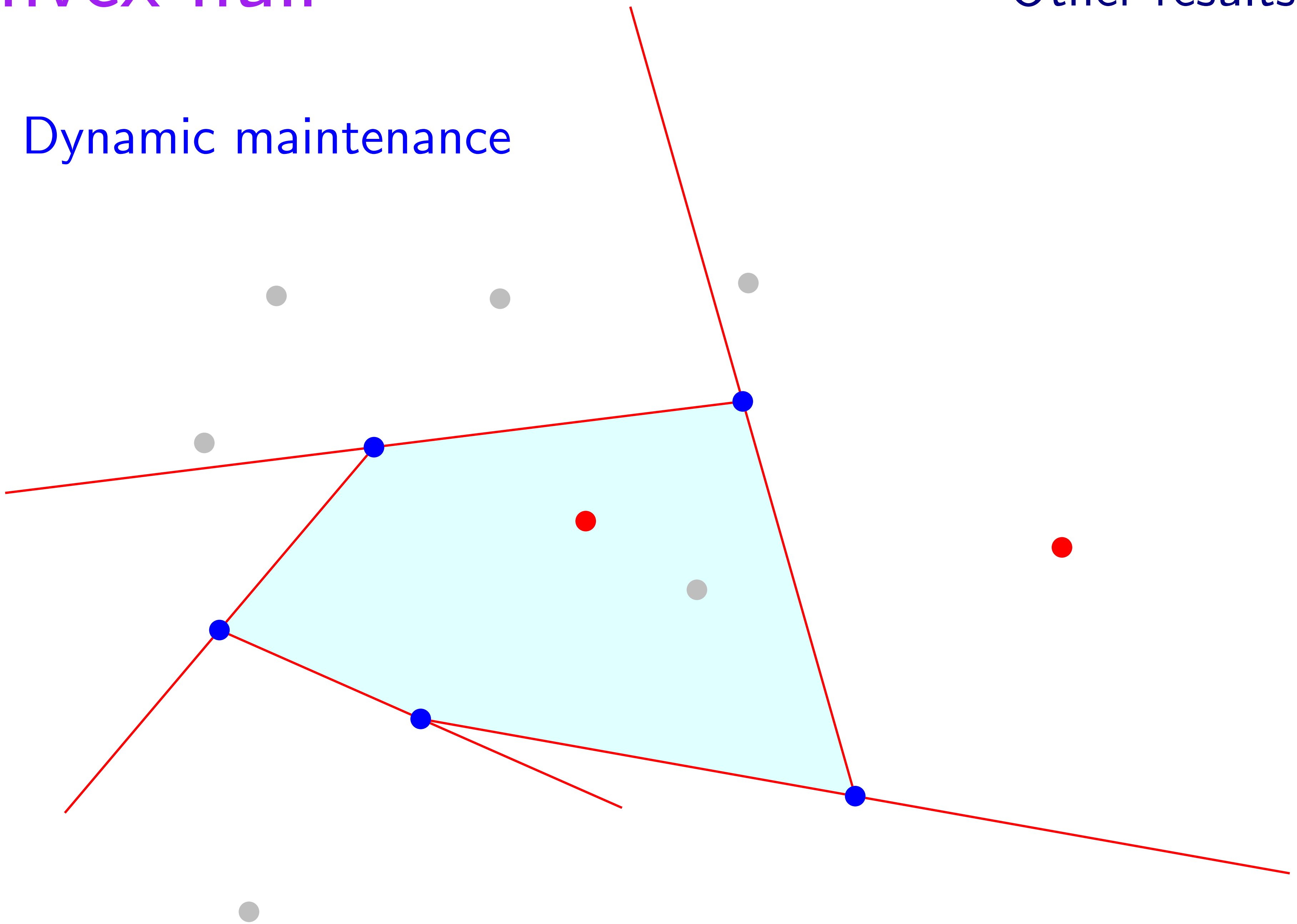
Convex hull

Dynamic maintenance



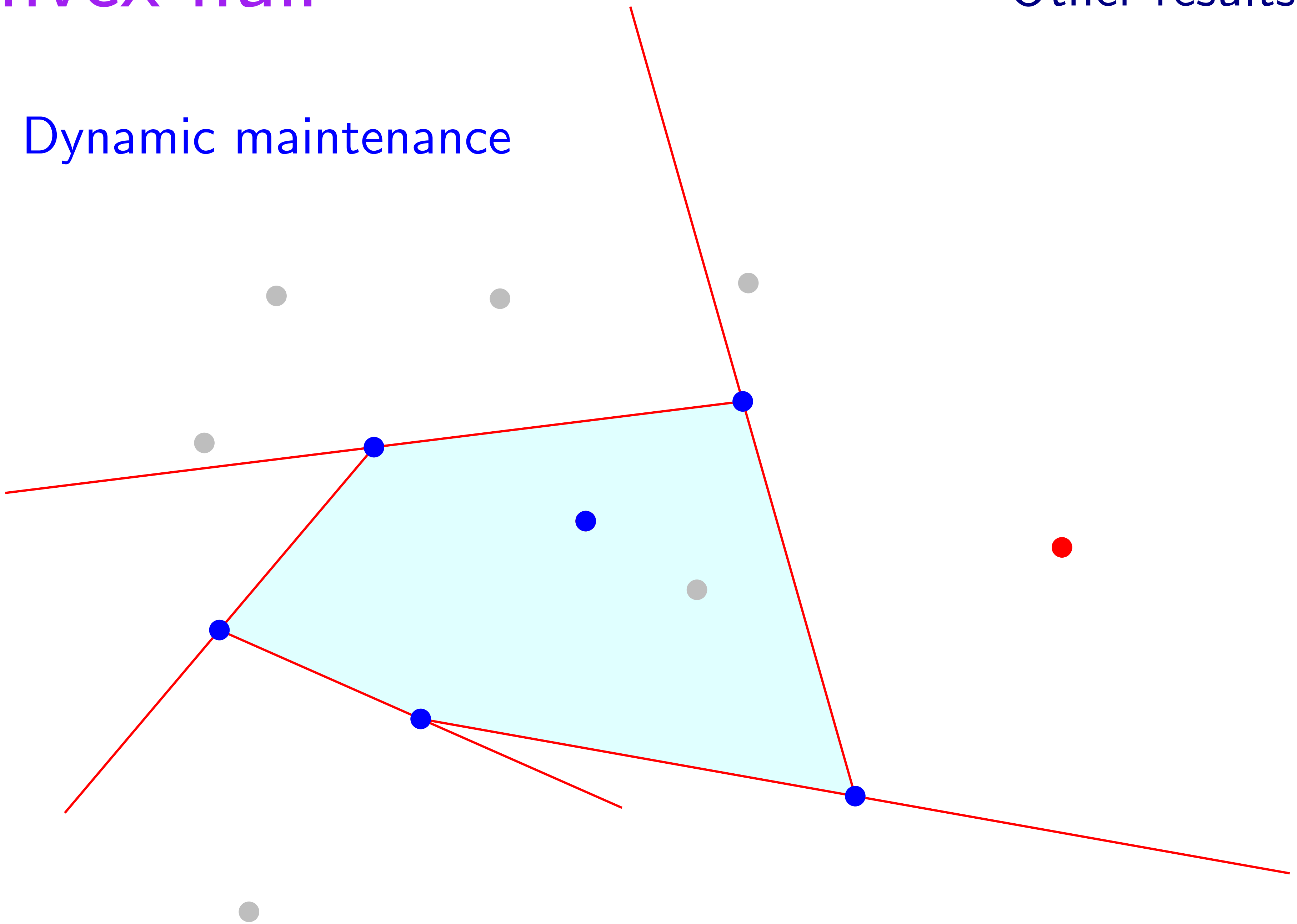
Convex hull

Dynamic maintenance



Convex hull

Dynamic maintenance

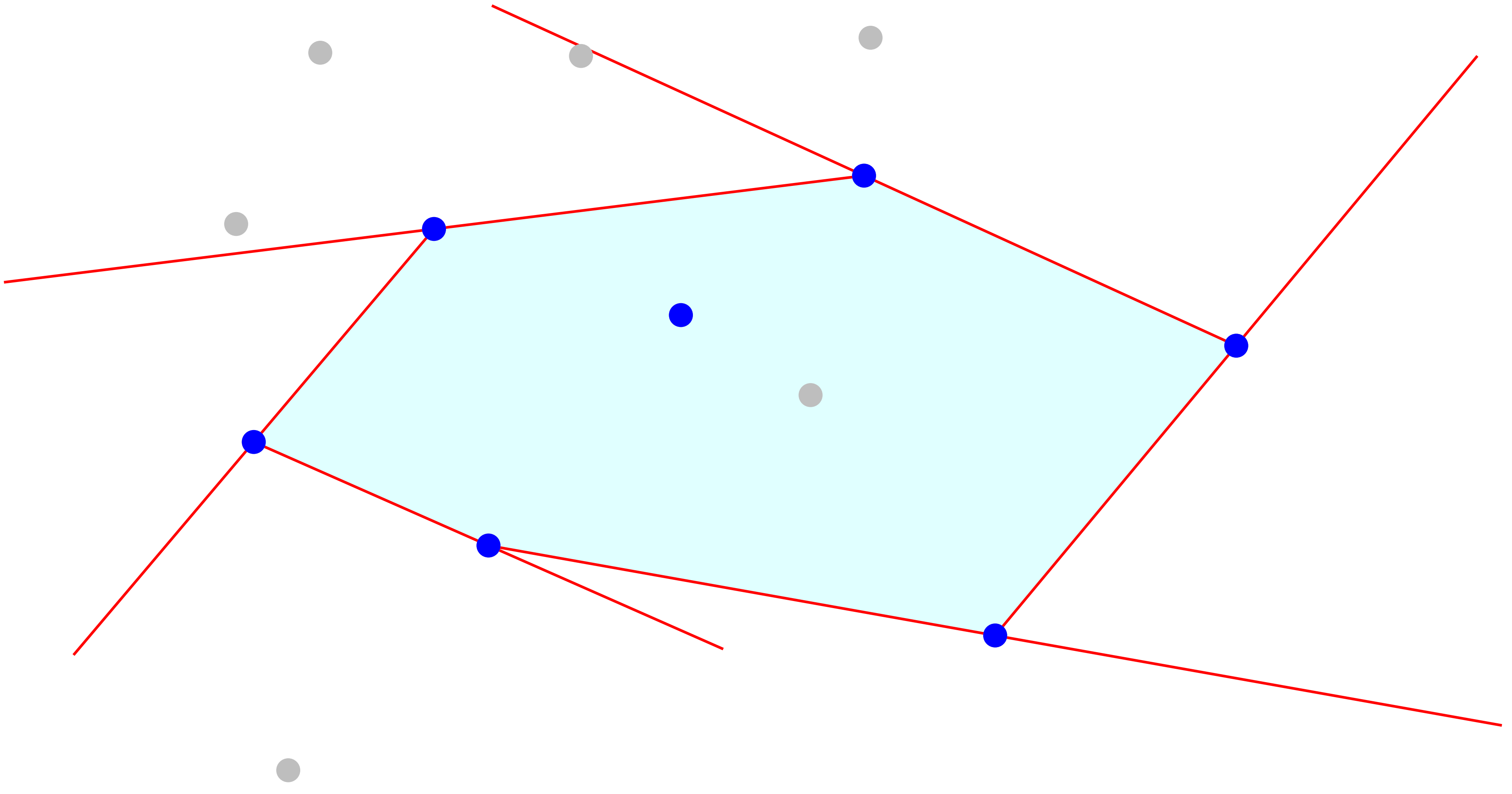


Convex hull

Other results

Dynamic maintenance

$O(\log n)$ per insertion



Convex hull

Euler relation

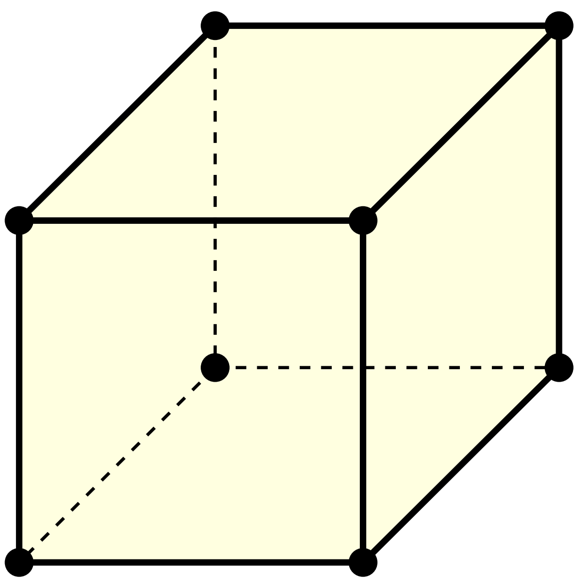
Polytope boundary

Vertices Edges Faces

Convex hull

Euler relation

Polytope boundary

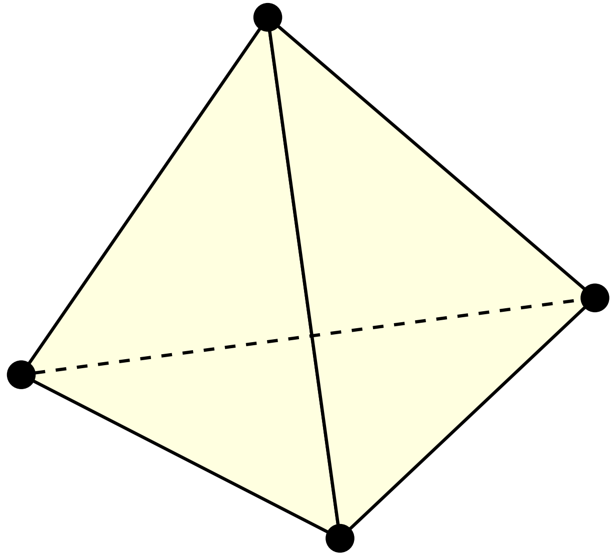


Vertices		Edges		Faces	
8	$-$	12	$+$	6	$= 2$

Convex hull

Euler relation

Polytope boundary

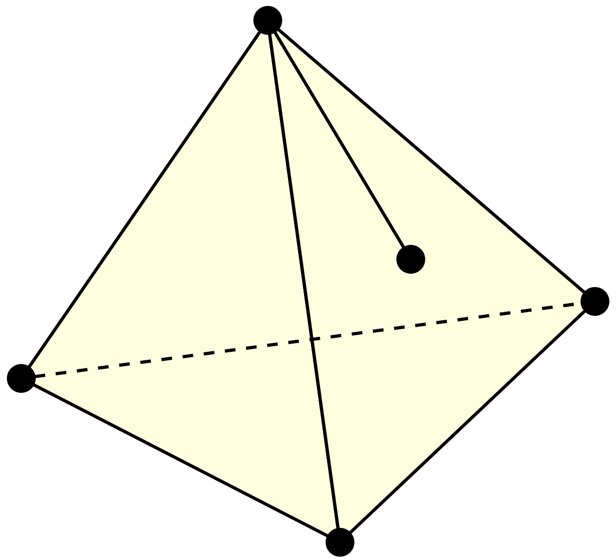


Vertices		Edges		Faces	
8	$-$	12	$+$	6	$= 2$
4	$-$	6	$+$	4	$= 2$

Convex hull

Euler relation

Polytope boundary

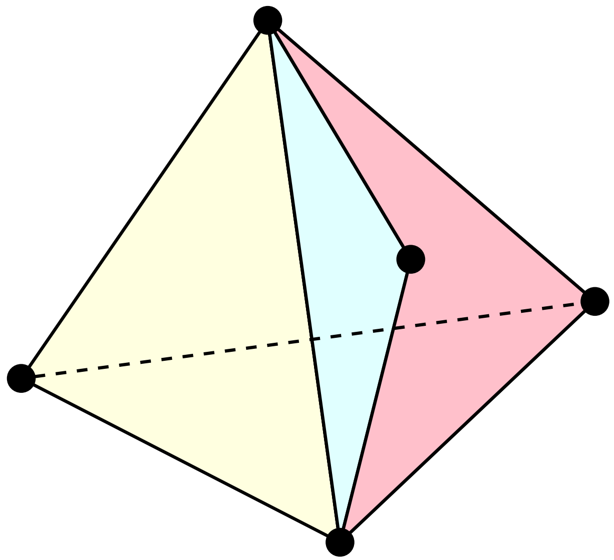


	Vertices		Edges		Faces	
	8	$-$	12	$+$	6	$= 2$
	4	$-$	6	$+$	4	$= 2$
	$+1$	$-$	$+1$	$+$	0	$= +0$

Convex hull

Euler relation

Polytope boundary

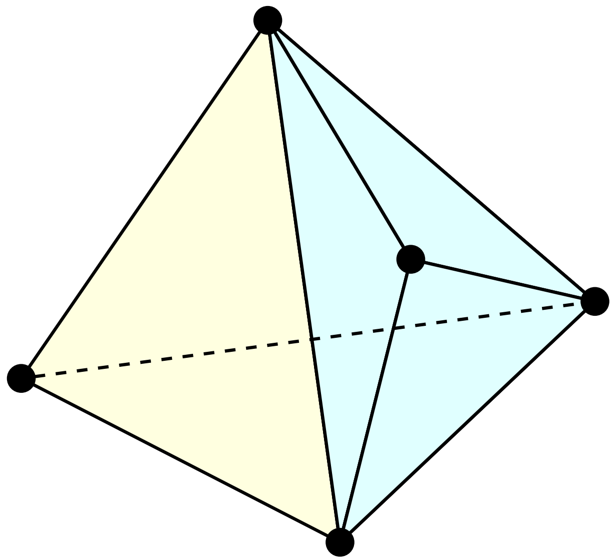


Vertices		Edges		Faces	
8	$-$	12	$+$	6	$= 2$
4	$-$	6	$+$	4	$= 2$
$+1$	$-$	$+1$	$+$	0	$= +0$
0	$-$	$+1$	$+$	$+1$	$= +0$

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 $\quad 3f = 2e$

$$f = 2n - 4$$

$$e = 3n - 6$$

Convex hull

Linear size

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

$O(nh)$ gift wraping algorithm

Convex hull

Dehn Sommerville relations $f_i = \sharp(\text{faces of dim } i)$

Euler: $$f_0 - f_1 + f_2 - \ldots f_{d-1} = (-1)^{d-1} + 1$$

Convex hull

Dehn Sommerville relations $f_i = \sharp(\text{faces of dim } i)$

Euler: $$f_0 - f_1 + f_2 - \ldots f_{d-1} = (-1)^{d-1} + 1$$

$$\sum_j = k^{d-1} - 1^j \left(\begin{array}{c} j+1 \\ k+1 \end{array} \right) f_j = (-1)^{d-1} f_k$$

$$-1 \leq k \leq d-2 \qquad f_{-1} = f_d = 1$$

$\left\lfloor \dfrac{d+1}{2} \right\rfloor$ independent equations

Convex hull

Dehn Sommerville relations $f_i = \sharp(\text{faces of dim } i)$

If $f_0, f_1, \ldots f_{\lfloor \frac{d-1}{2} \rfloor}$ are known

$f_{\lfloor \frac{d+1}{2} \rfloor}, \ldots f_{d-1}$ can be deduced

Convex hull

Dehn Sommerville relations $f_i = \sharp(\text{faces of dim } i)$

If $f_0, f_1, \ldots f_{\lfloor \frac{d-1}{2} \rfloor}$ are known

$f_{\lfloor \frac{d+1}{2} \rfloor}, \ldots f_{d-1}$ can be deduced

$$f_{\lfloor \frac{d-1}{2} \rfloor} = O(n^{\lfloor \frac{d+1}{2} \rfloor})$$

$$\implies \forall i \;\; f_i = O(n^{\lfloor \frac{d+1}{2} \rfloor})$$

Convex hull

Dehn Sommerville relations $f_i = \sharp(\text{faces of dim } i)$

If $f_0, f_1, \ldots f_{\lfloor \frac{d-1}{2} \rfloor}$ are known

$f_{\lfloor \frac{d+1}{2} \rfloor}, \ldots f_{d-1}$ can be deduced

$$f_{\lfloor \frac{d-1}{2} \rfloor} = O(n^{\lfloor \frac{d+1}{2} \rfloor})$$

$$\Longrightarrow \forall i \;\; f_i = O(n^{\lfloor \frac{d+1}{2} \rfloor})$$

$\exists$ an optimal algorithm

Linear programming

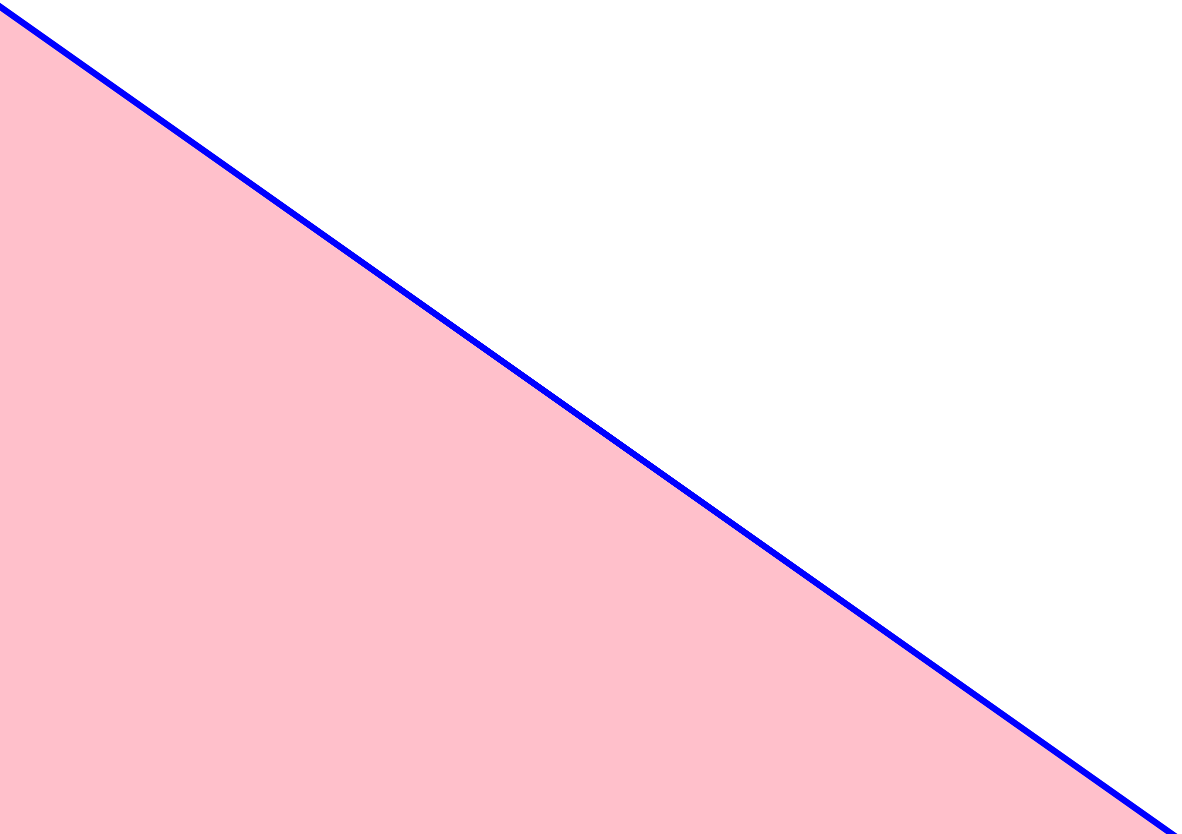
A simple algorithm [Seidel]

n linear constraints (half-spaces)

Linear programming

A simple algorithm [Seidel]

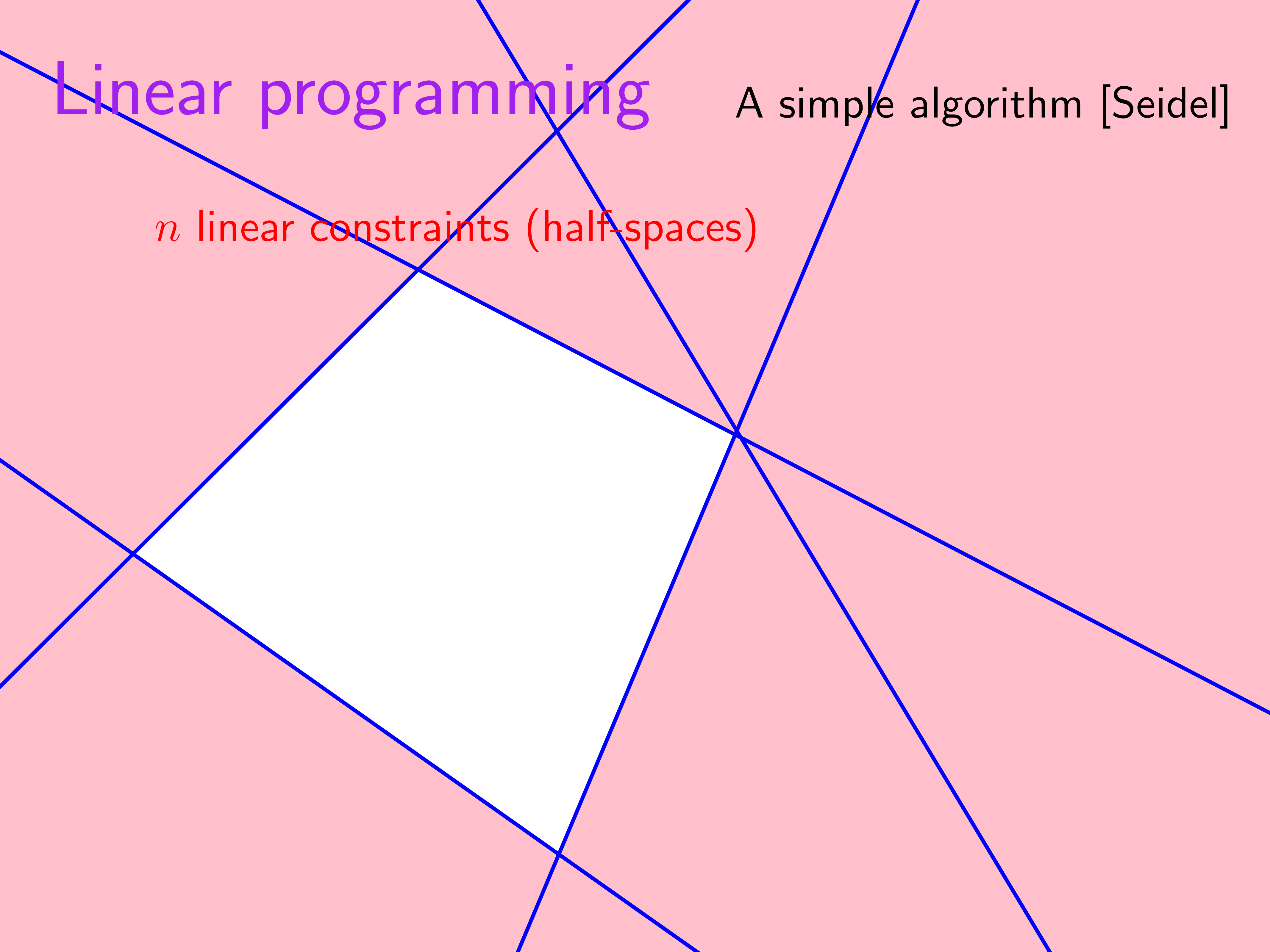
n linear constraints (half-spaces)



Linear programming

A simple algorithm [Seidel]

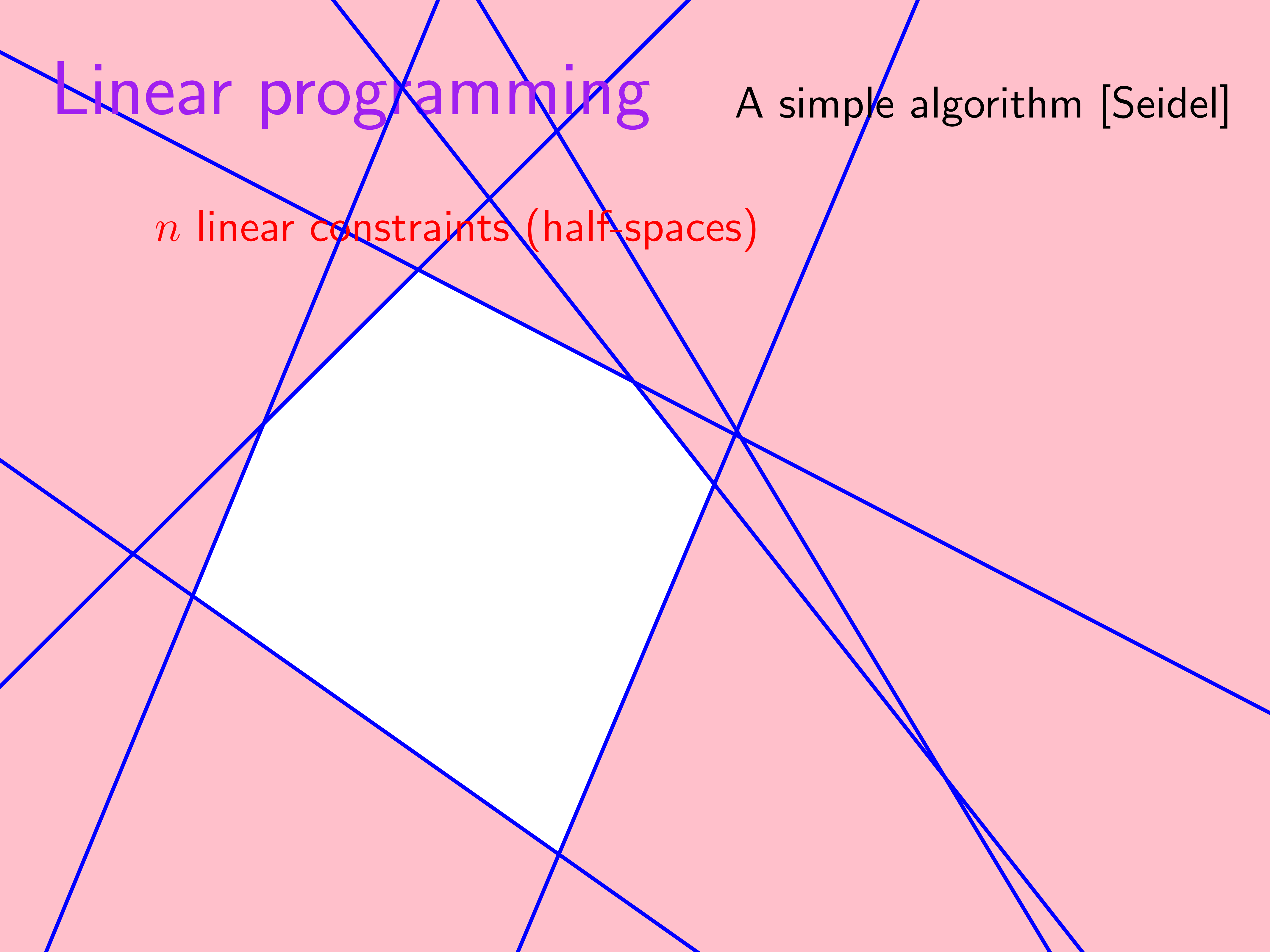
n linear constraints (half-spaces)



Linear programming

A simple algorithm [Seidel]

n linear constraints (half-spaces)

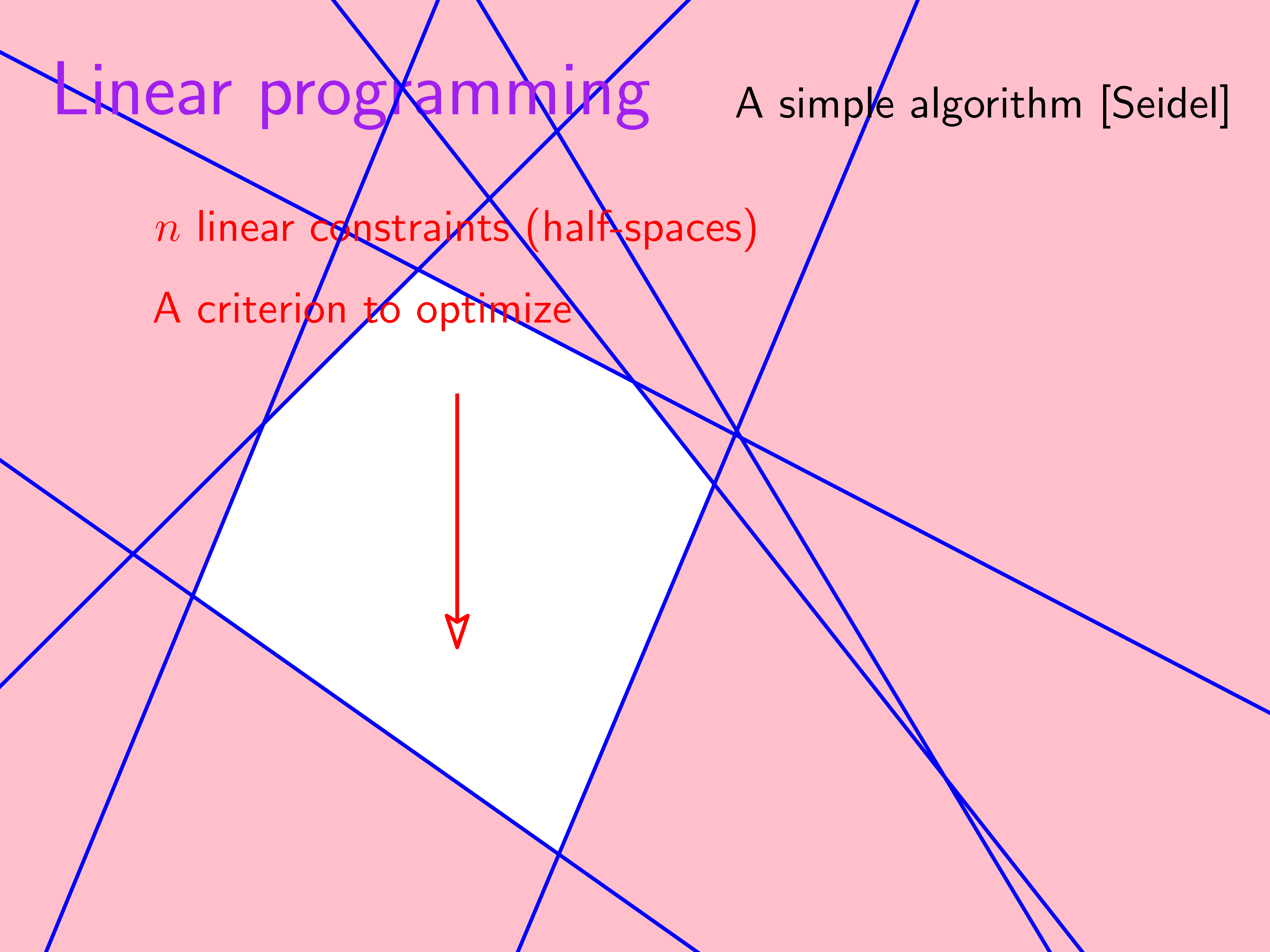


Linear programming

A simple algorithm [Seidel]

n linear constraints (half-spaces)

A criterion to optimize

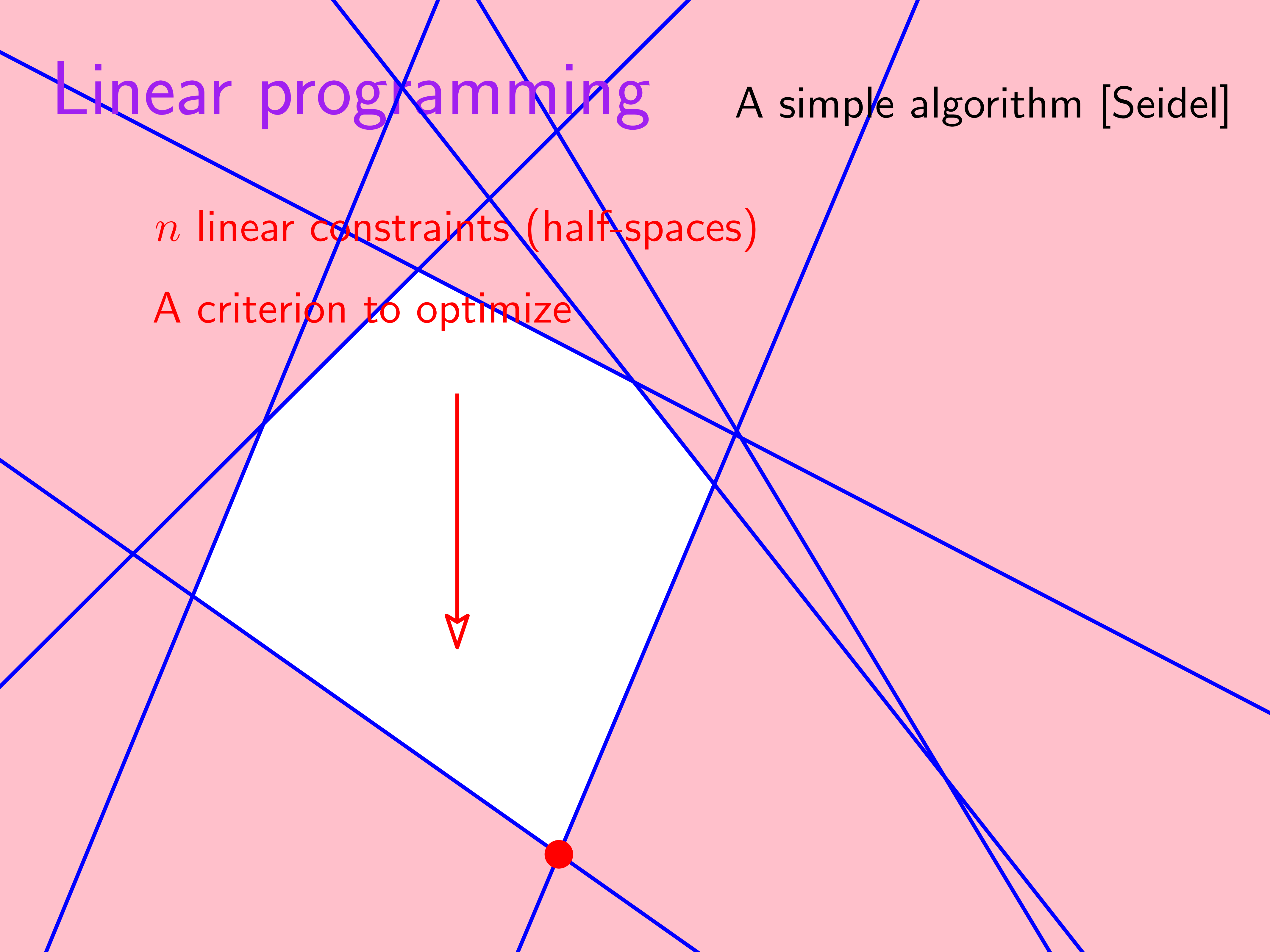


Linear programming

A simple algorithm [Seidel]

n linear constraints (half-spaces)

A criterion to optimize



Linear programming

A simple algorithm [Seidel]

One dimension

Admissible solutions is an interval

Maintain incrementally

Linear programming

A simple algorithm [Seidel]

One dimension

Admissible solutions is an interval

Maintain incrementally

Easy $O(n)$

Linear programming
A simple algorithm [Seidel]

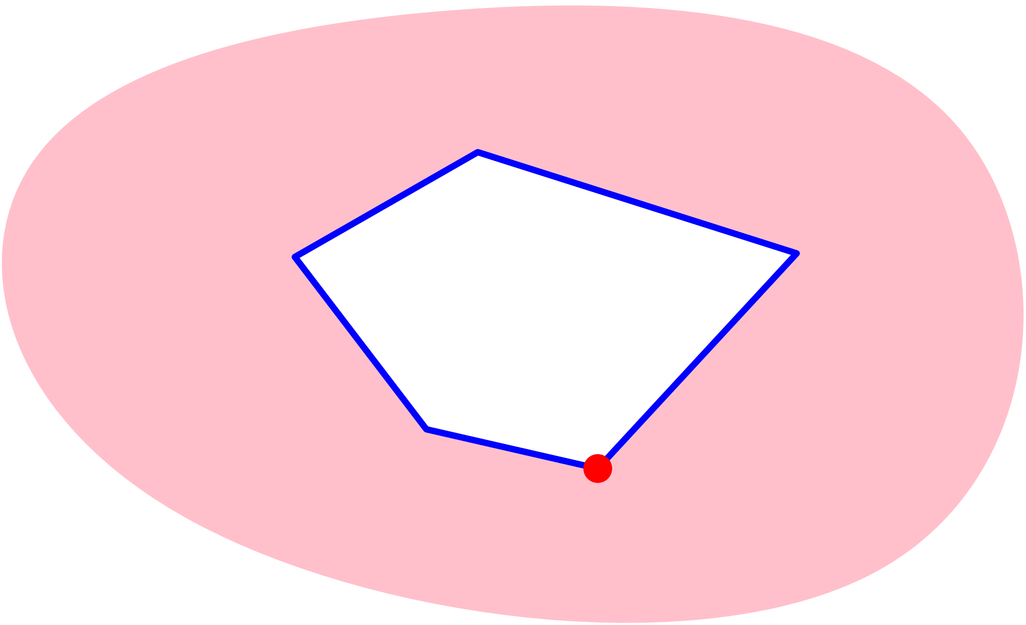
Two dimensions

Linear programming

A simple algorithm [Seidel]

Two dimensions

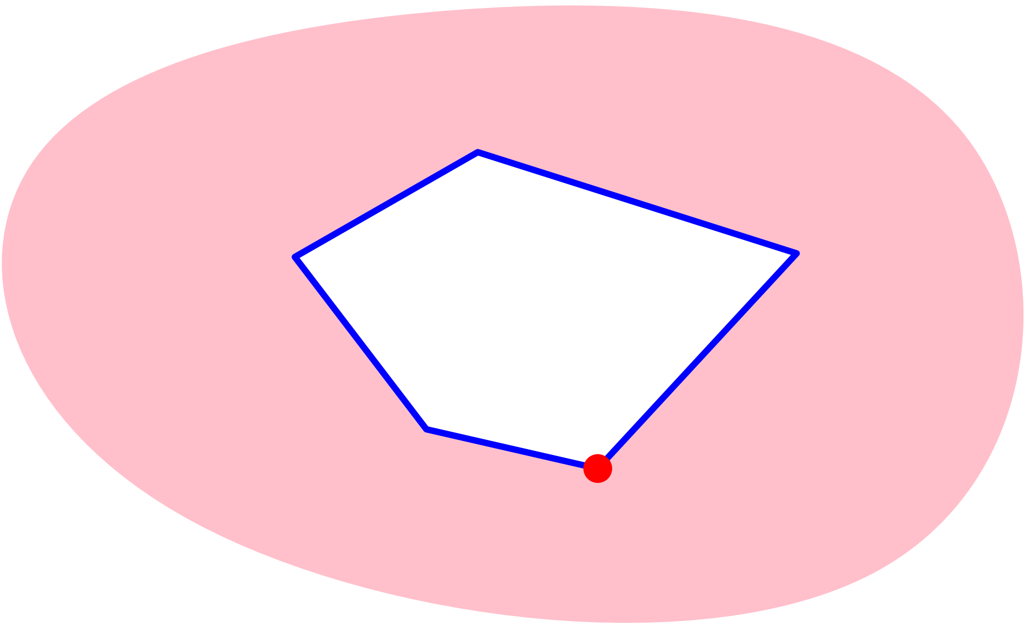
Incremental



Linear programming

A simple algorithm [Seidel]

Two dimensions



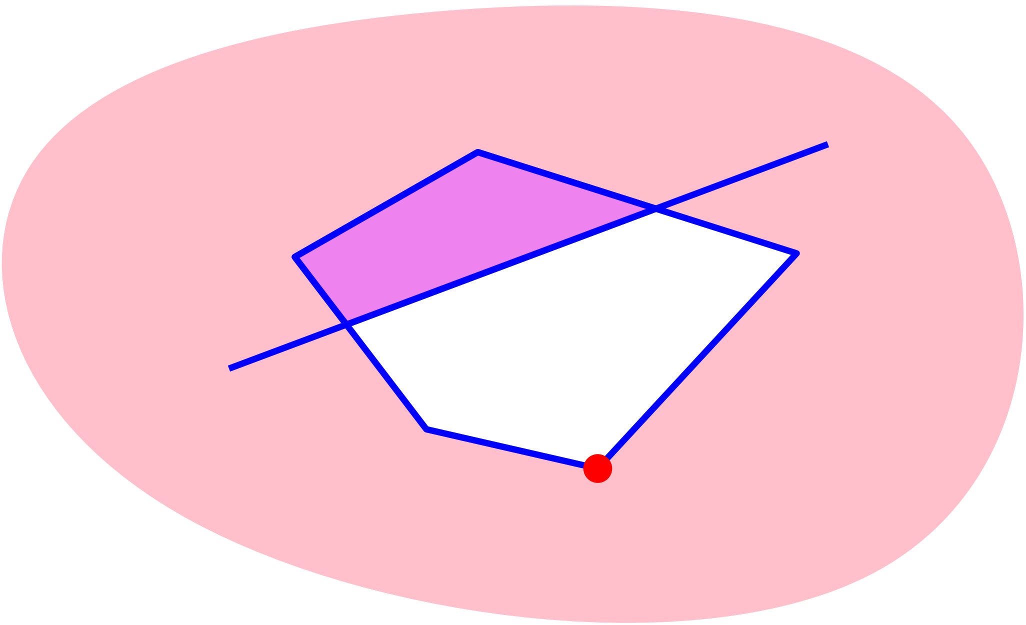
Incremental

Check solution with respect to new constraint

Linear programming

A simple algorithm [Seidel]

Two dimensions



Incremental

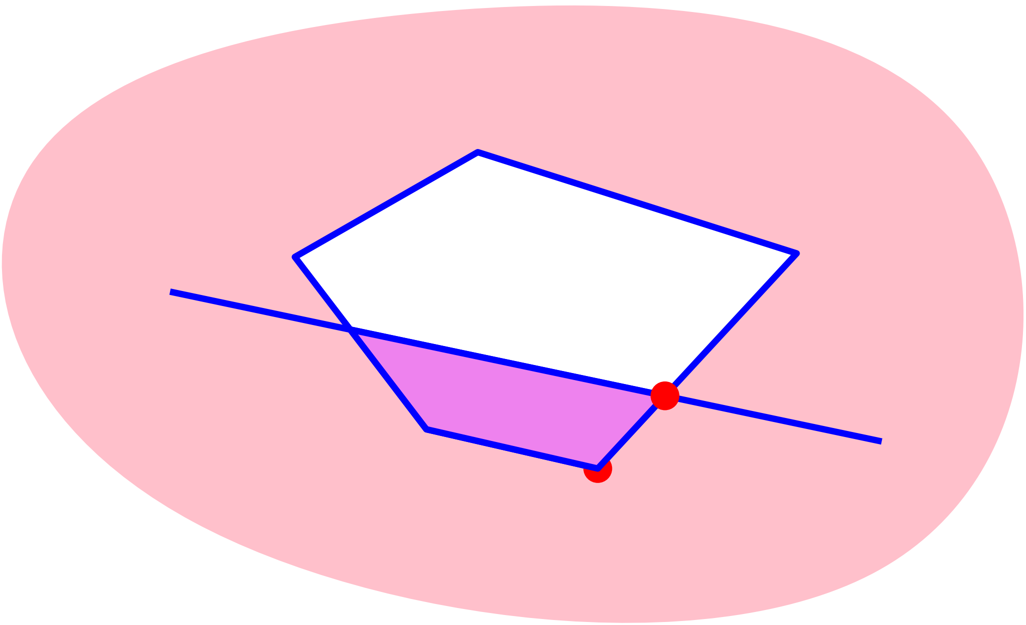
Check solution with respect to new constraint

OK ⟶ do nothing (next constraint)

Linear programming

A simple algorithm [Seidel]

Two dimensions



Incremental

Check solution with respect to new constraint

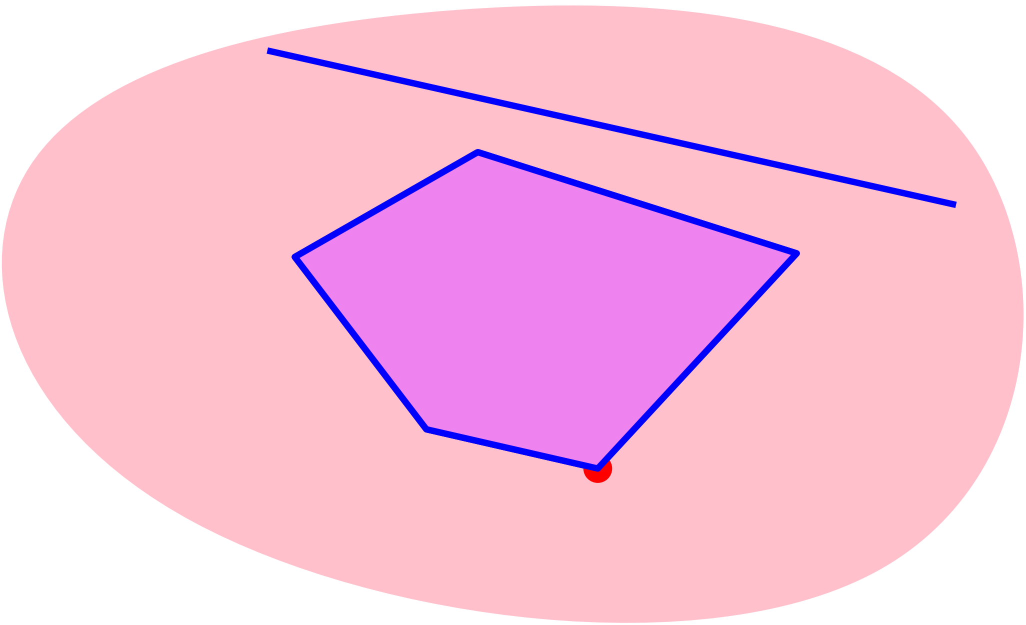
OK ⟶ do nothing (next constraint)

otherwise ⟶ solve in 1D on new constraint

Linear programming

A simple algorithm [Seidel]

Two dimensions



Incremental

Check solution with respect to new constraint

OK ⟶ do nothing (next constraint)

otherwise ⟶ solve in 1D on new constraint

Linear programming

A simple algorithm [Seidel]

Complexity

Quadratic in worst case

Linear programming

A simple algorithm [Seidel]

Complexity

Quadratic in worst case

Random order

$$f_2(n) = f_2(n-1) + \frac{n-2}{n} \cdot 1 + \frac{2}{n} f_1(n-1) = O(n)$$

Linear programming

A simple algorithm [Seidel]

Complexity

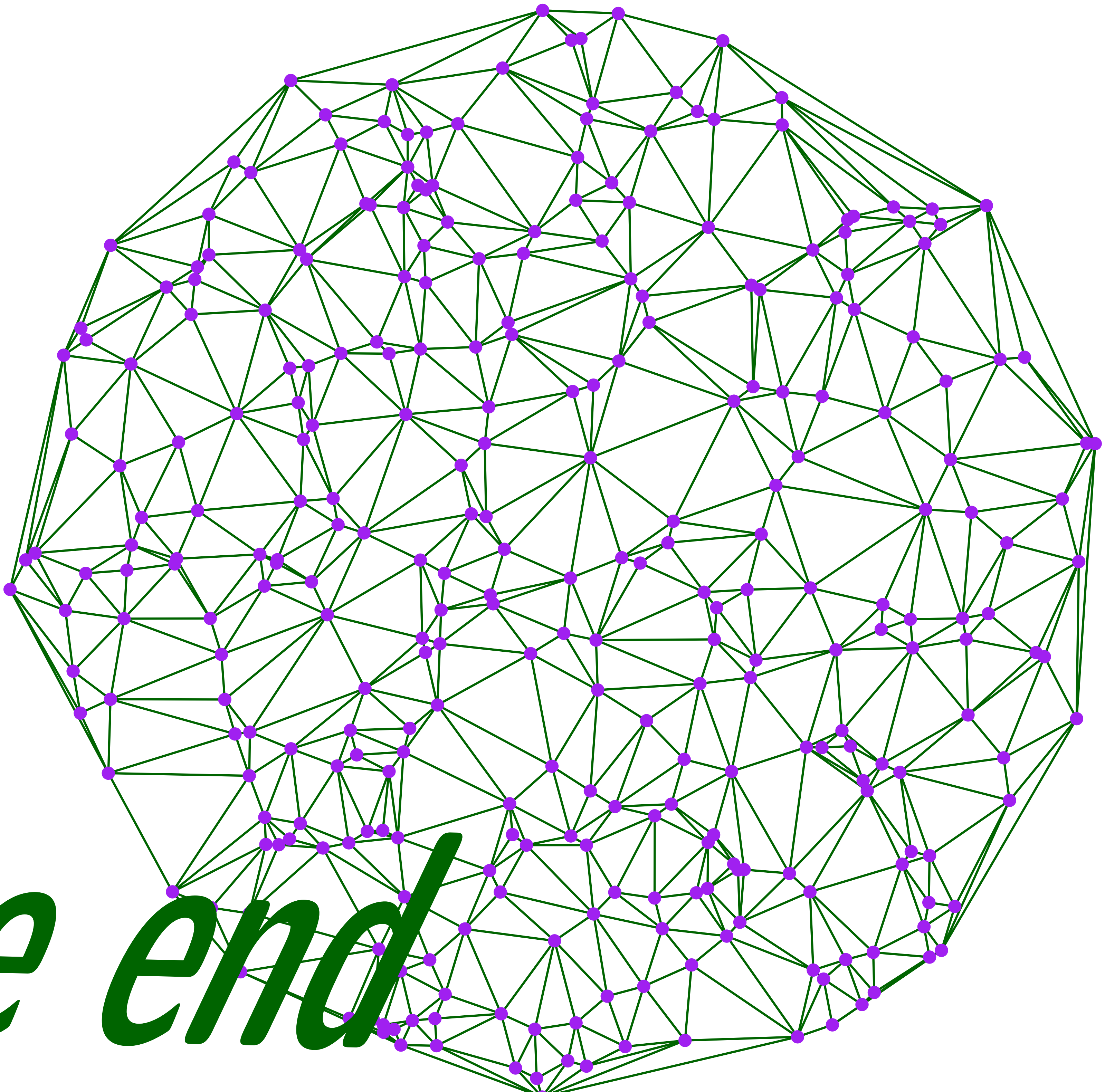
Quadratic in worst case

Random order

$$f_2(n) = f_2(n-1) + \frac{n-2}{n} \cdot 1 + \frac{2}{n} f_1(n-1) = O(n)$$

Higher dimension

$$f_d(n) = f_d(n-1) + \frac{n-2}{n} \cdot 1 + \frac{2}{n} f_{d-1}(n-1) = O(n)$$



The end