Computational Geometry and Topology
Géométrie et topologie algorithmiques

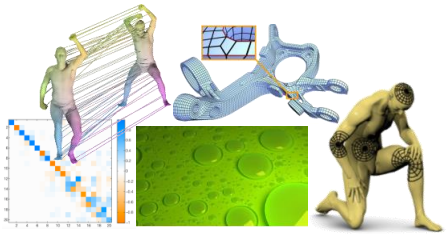
Steve OUDOT *Pooran MEMARI*

Acknowledgments of Involved Colleagues:

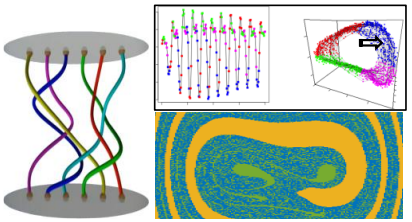
Jean Daniel Boissonnat, Frederic Chazal, Marc Glisse,
As well as Olivier Devillers and Luca Castelli

GeomeriX Research Team

Geometry-driven Numerics



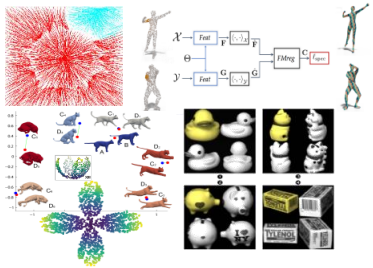
Geometry for Shape Processing



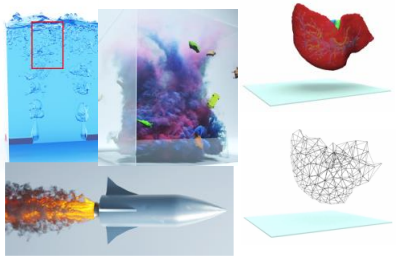
Geometry for Dynamical Systems

Geometry

Geometry for Data Science

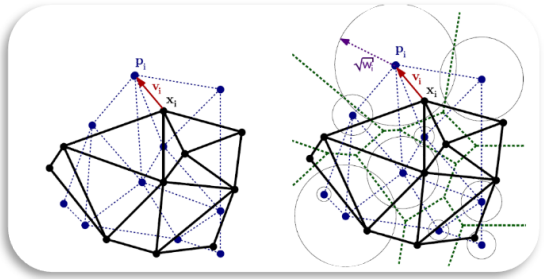


Geometry for Simulation



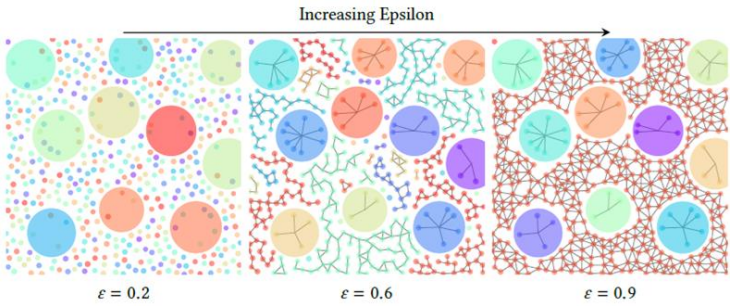
2025-2026: 8 lectures of 3h, Monday 8:45-11:45.

- [22/9] Warm up: 2D convex geometry [PM]
- [29/9] Comparing objects, polytopes [PM]
- [6/10] Nearest neighbor search [SO]
- [13/10] Voronoi, Delaunay [PM]
- [20/10] Reconstruction in higher dimensions [PM]
- [3/11] Clustering [SO]
- [10/11] Homology and persistent homology [SO]
- [17/11] Stability of persistent homology - homology inference [SO]
- [1/12] Written exam



Type: *Theoretical aspect of geometry and algorithms*

Pre-requisite: not allergic to theory, interested by applications



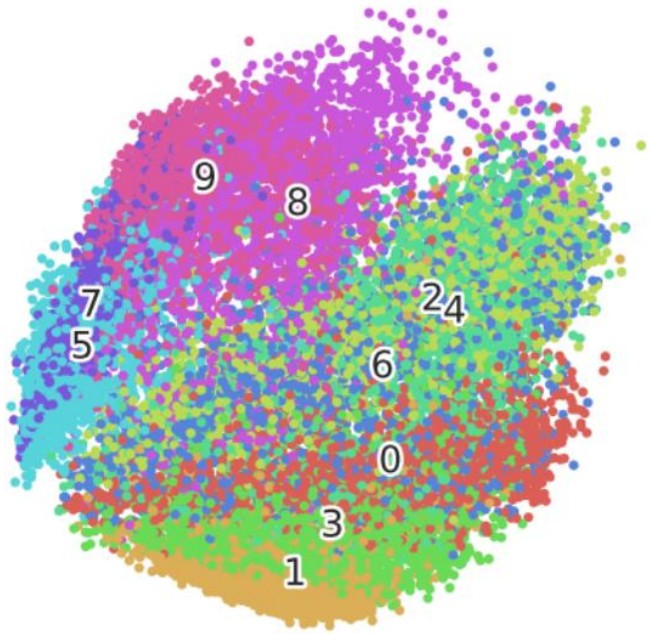
Questions:
- *Steve Oudot (steve.oudot@inria.fr)*
- *Pooran Memari (memari@lix.polytechnique.fr)*

Outline

- **Warm-up & Reminders**
- **Part I — Convex Geometry (Introduction)**
  - Convex hull algorithms
  - Duality
- **Part II — Applications** *(with break)*
- **Part III — Fundamental Results in Convex Geometry**
  - Radon's Theorem
  - Helly's Theorem
  - Centerpoint Theorem

Geometric RepresentationS

- Geometric Intuition
- Geometric Representation
- Geometric Algorithms



High dimensional Geometry Challenges

- Dimensionality severely restricts our intuition and ability to visualize data

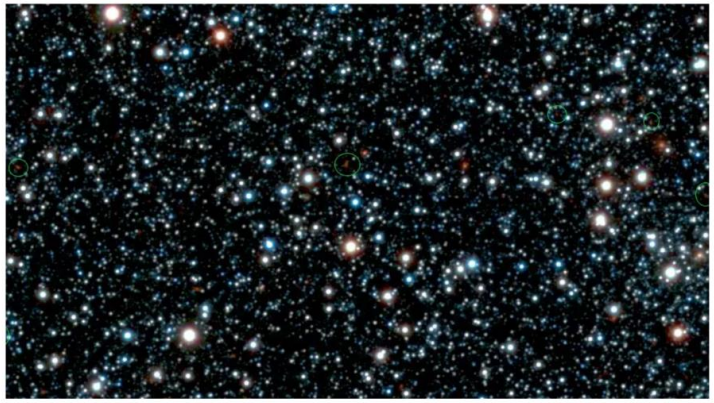
=> need for automated and provably correct methods S

- Complexity of data structures and algorithms rapidly grow as the dimensionality increases

=> no subdivision of the ambient space is affordable

=> data structures and algorithms should be sensitive to the intrinsic dimension (usually unknown) of the data

- Inherent defects : sparsity, noise, outliers



A section of the Small Magellanic Cloud as seen by the VISTA telescope with distant galaxies circled in green. (Image credit: ESO/VISTA Magellanic Clouds Survey)

Complexity Analysis: some reminders

- **Upper Bound**

  $g(n) = O(f(n))$

  There exist C, N such that for all $n \geq N$:

  $g(n) \leq C\, f(n)$
- **Lower Bound**

  $g(n) = \Omega(f(n))$

  There exist C_0, N such that for all $n \geq N$:

  $g(n) \geq C_0\, f(n)$
- **Tight Bound**

  $g(n) = \Theta(f(n))$

  There exist C, C_0, N such that for all $n \geq N$:

  $C_0 f(n) \leq g(n) \leq C f(n)$

Example

Sorting n numbers by comparisons:

$T(n) = \Theta(n \log n)$

Computing the Convex Hull of n Points in the Plane

- **Input:** a set P of n points in $\mathbb{R}^2$
- **Output:** the ordered list (counterclockwise) of the vertices of $\mathrm{conv}(P)$
- **Lower Bound:**

  $\Omega(n \log n)$

  Example reduction:

  $p_i = (x_i, x_i^2)$

  The convex hull of $\{p_i\}$ ⇒ sorting of $\{x_i\}$
- **Naive Upper Bound:**

  $O(n^3)$

Part I

Convex geometry: short introduction

Barycentric coordinates

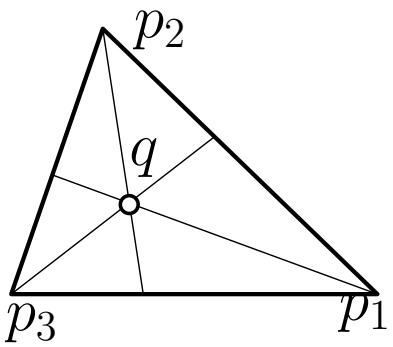
$$q = \sum_i^n \alpha_i p_i \text{ (where } \sum_i \alpha_i = 1)$$

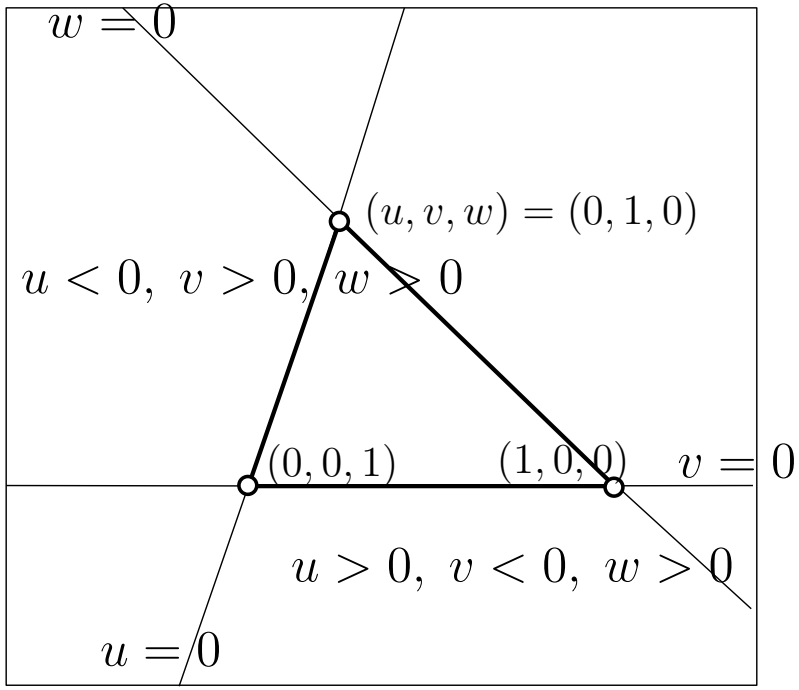
the coefficients $(\alpha_1, \ldots, \alpha_n)$ are called *barycentric coordinates* of point q with respect to $p_1, \ldots, p_n$

Barycentric coordinates

$$q = \sum_i^n \alpha_i p_i \text{ (where } \sum_i \alpha_i = 1)$$

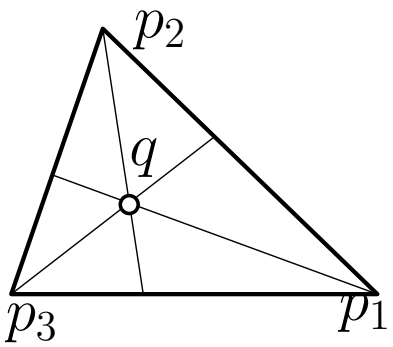
the coefficients $(\alpha_1, \ldots, \alpha_n)$ are called *barycentric coordinates* of point q with respect to $p_1, \ldots, p_n$

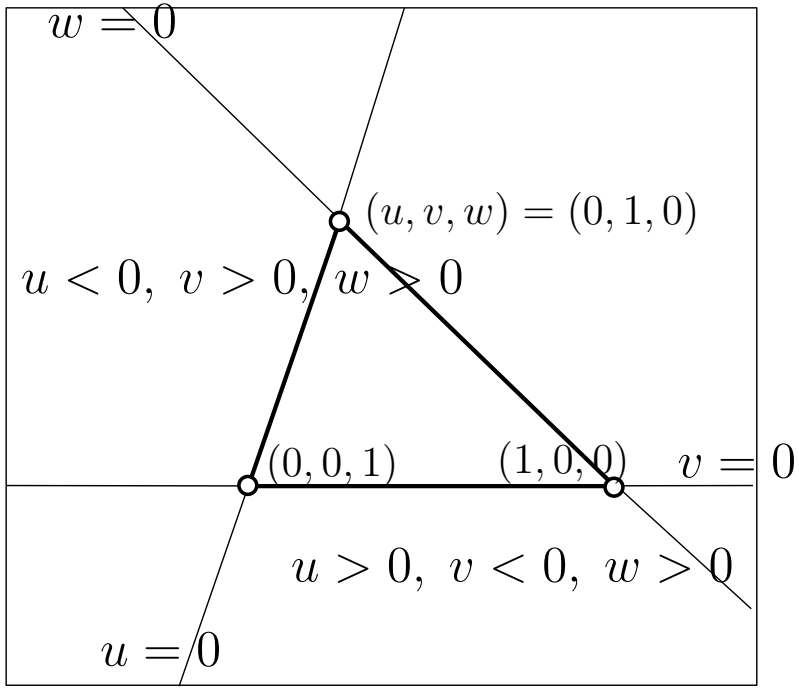




Barycentric coordinates

$$\mathbf{q} = u\,\mathbf{p_1} + v\,\mathbf{p_2} + w\,\mathbf{p_3} = \frac{A(q,p_2,p_3)}{A(p_1,p_2,p_3)}\mathbf{p_1} + \frac{A(p_1,q,p_3)}{A(p_1,p_2,p_3)}\mathbf{p_2} + \frac{A(p_1,p_2,q)}{A(p_1,p_2,p_3)}\mathbf{p_3}$$





Convex hulls

affine combination

$$aff(\mathcal{S}) = \{ \sum_{i=1}^{n} \alpha_i p_i \mid \sum_i \alpha_i = 1 \} \qquad \mathcal{S} := \{p_1, \ldots, p_n\}$$

$$convex(\mathcal{S}) = \{ \sum_{i=1}^{n} \alpha_i p_i \mid \alpha_i \geq 0, \sum_i \alpha_i = 1 \}$$

convex combination

Convex hulls

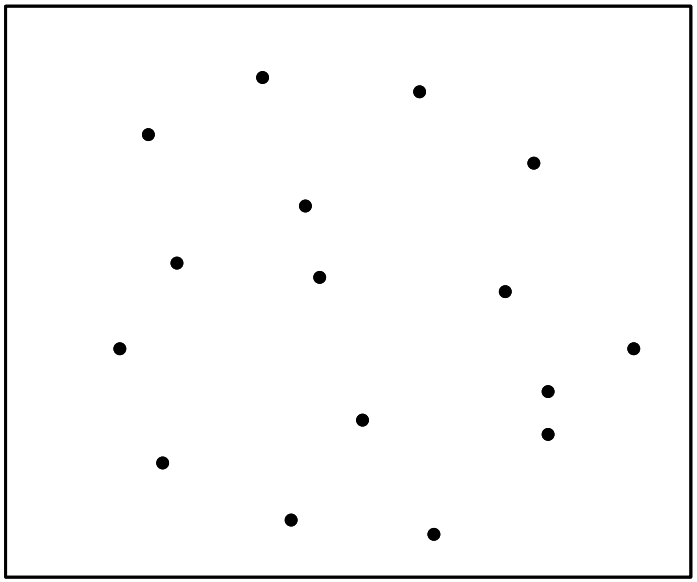
$\mathcal{S} := \{p_1, \ldots, p_n\}$

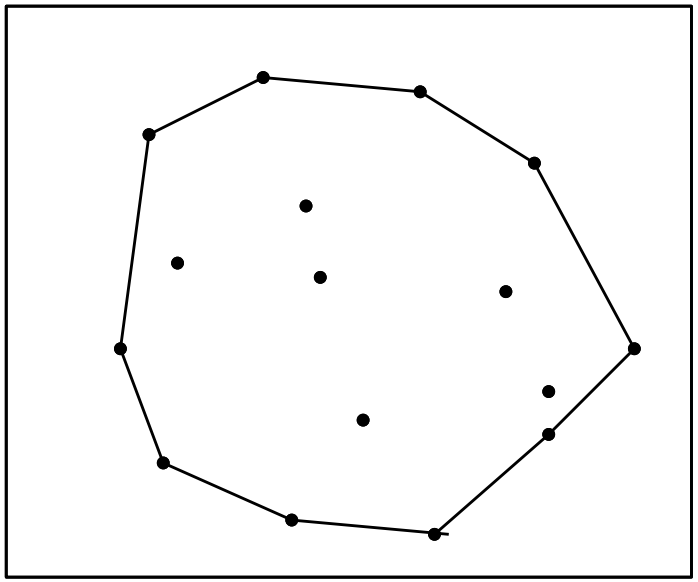
$aff(\mathcal{S}) = \{ \sum_{i=1}^{n} \alpha_i p_i \mid \sum_i \alpha_i = 1 \}$

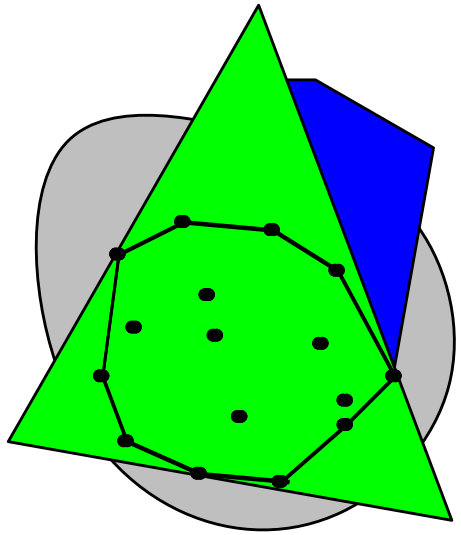
affine combination

$convex(\mathcal{S}) = \{ \sum_{i=1}^{n} \alpha_i p_i \mid \alpha_i \geq 0, \sum_i \alpha_i = 1 \}$

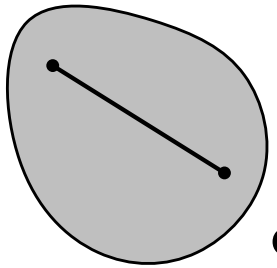
convex combination





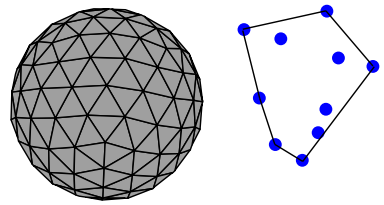


intersection of all convex sets containing $\mathcal{S}$



convex set

Polytopes



Def (*polytope*): the convex hull of a finite set of points in $\mathbb{E}^d$

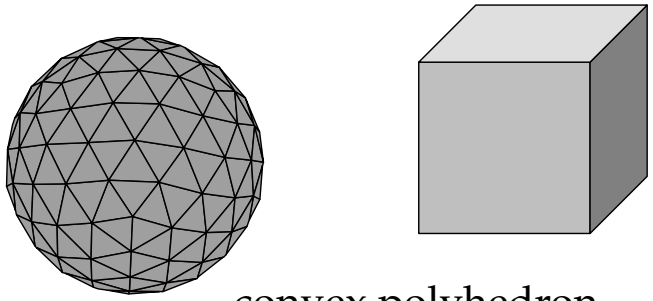
Upper bound theorem

A polytope with n vertices in $\mathbb{R}^d$ has at most $O(n^{\lfloor d/2 \rfloor})$ faces in overall.

This bound is achived for the *cyclic polytopes*: convex hulls of n points on the moment curve $(t, t^2, t^3 \ldots, t^d)$.

in 3D

Every polytope with n vertices has at most $O(n)$ faces and edges.

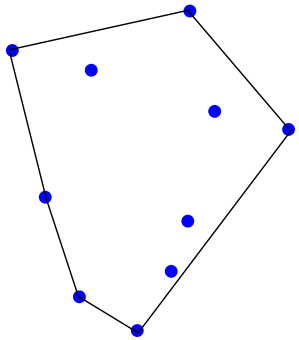


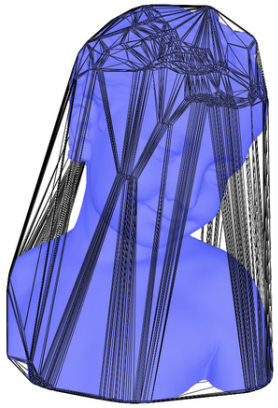
convex polyhedron

Convex hull computation: complexity

Lower bound

How much time do we need to compute the convex hull of n points?





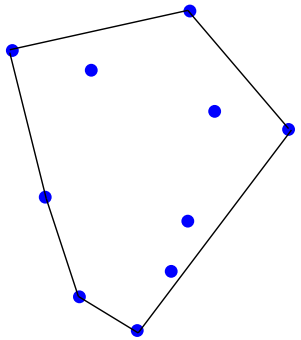
(image from CGAL user manual)

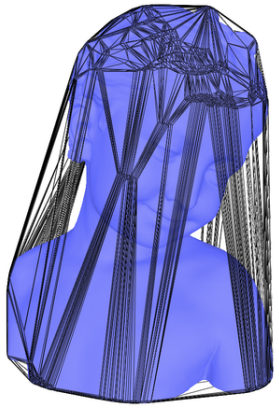
What kind of (memory) representation should we use?

Convex hull computation: complexity

Lower bound

How much time do we need to compute the convex hull of n points?

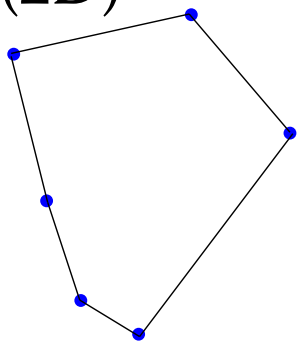




(image from CGAL user manual)

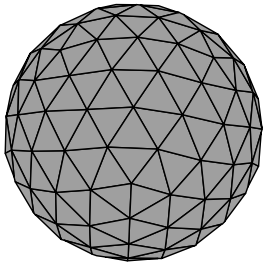
Convex hull representation

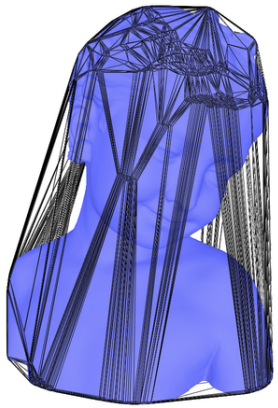
(2D)



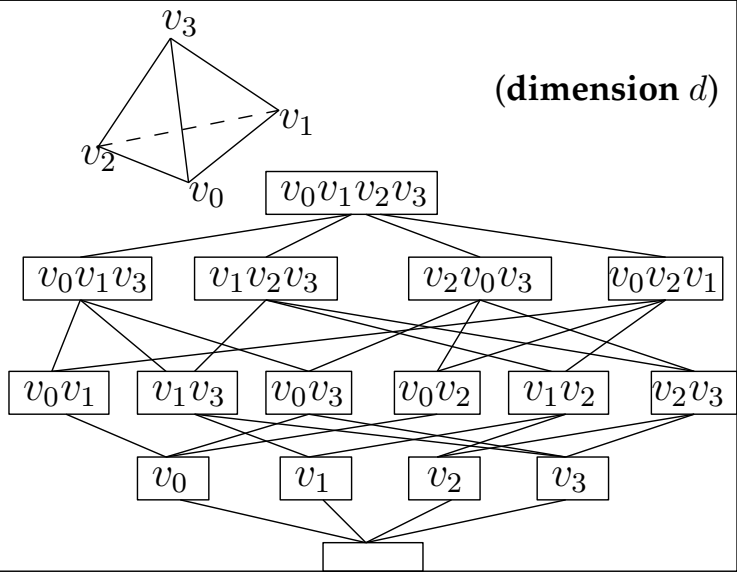
(circular doubly linked list)

(3D)





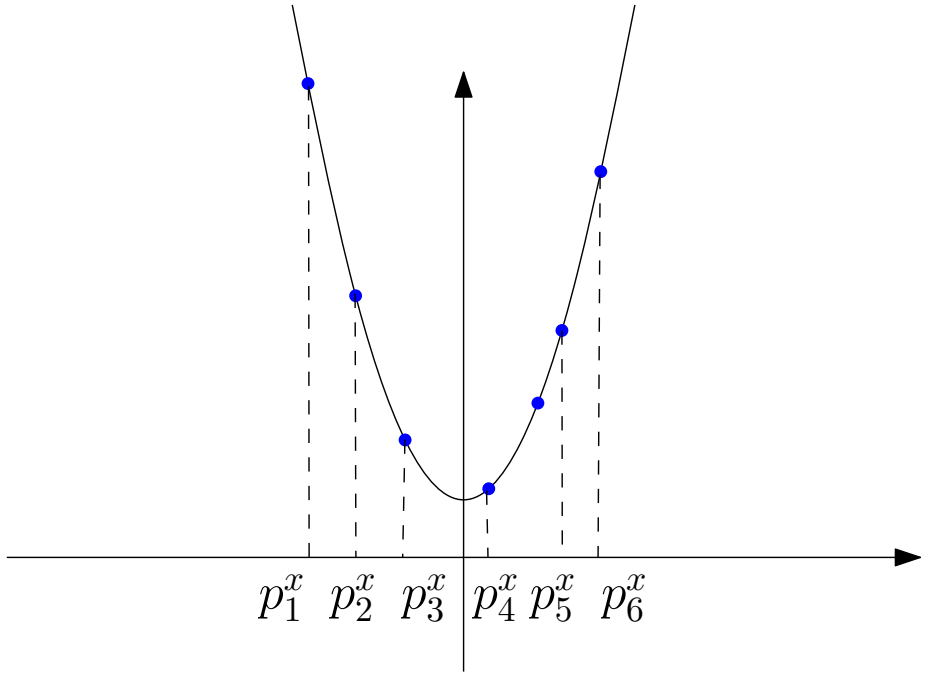
(dimension d)



Convex hull computation: complexity

Lower bound (2D)

The computation of the 2D convex hull of n points requires $\Omega(n \log n)$ time



Lower bound **(dimension d)**

The computation of the dD convex hull of n points requires time

$$O(n \log n + n^{\lfloor d/2 \rfloor})$$

Computing the Convex Hull of n points

In 2D

- **Complexity**:
  - Worst case: $O(n \log n)$.
  - Optimal, because sorting is a lower bound (convex hull encodes sorted order).
  - If the points are already sorted (say, by x-coordinate), linear-time $O(n)$ algorithms exist.
- **Classic algorithms**:
  - **Graham scan** ($O(n \log n)$)
  - **Jarvis march (Gift wrapping)** ($O(nh)$, where h is the number of hull vertices)
  - **Chan's algorithm** ($O(n \log h)$, output-sensitive)
  - **QuickHull** (average $O(n \log n)$, worst $O(n^2)$)

Computing the Convex Hull of n points

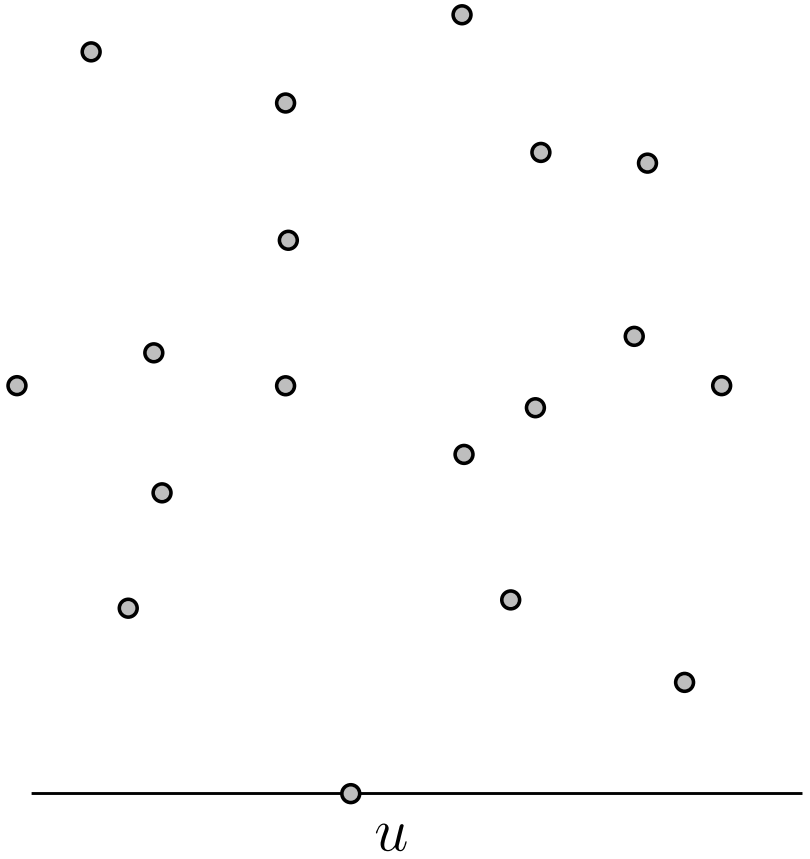
In higher dimensions ($d \geq 3$)

- **Complexity**:
  - Worst case: $O(n^{\lfloor d/2 \rfloor})$ (because the convex hull can have that many facets/ vertices).
  - For fixed d, the convex hull can be computed in

$$O(n \log n + n^{\lfloor d/2 \rfloor})$$

    which is optimal.
- **Classic algorithms**:
  - **Incremental algorithms** (Clarkson–Shor, Seidel, etc.)
  - **Divide-and-conquer** (Preparata–Hong)
  - **QuickHull** (generalized to higher d)
  - **Beneath–Beyond algorithm** (used in practice, e.g. in *Qhull*)

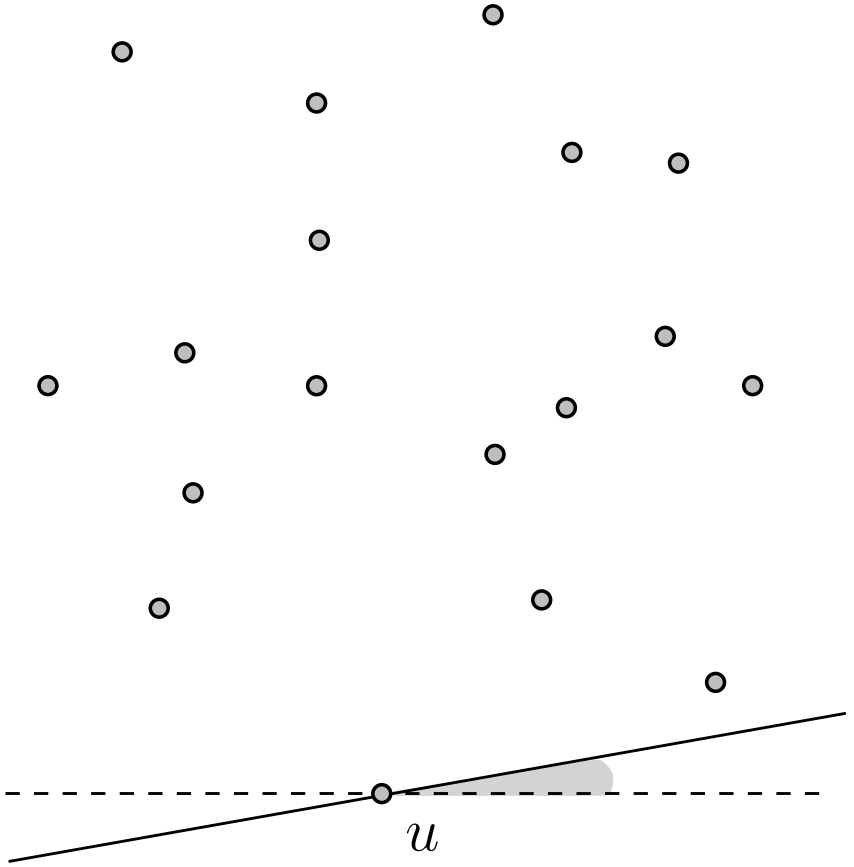
Convex hull computation: first algorithm (Jarvis)



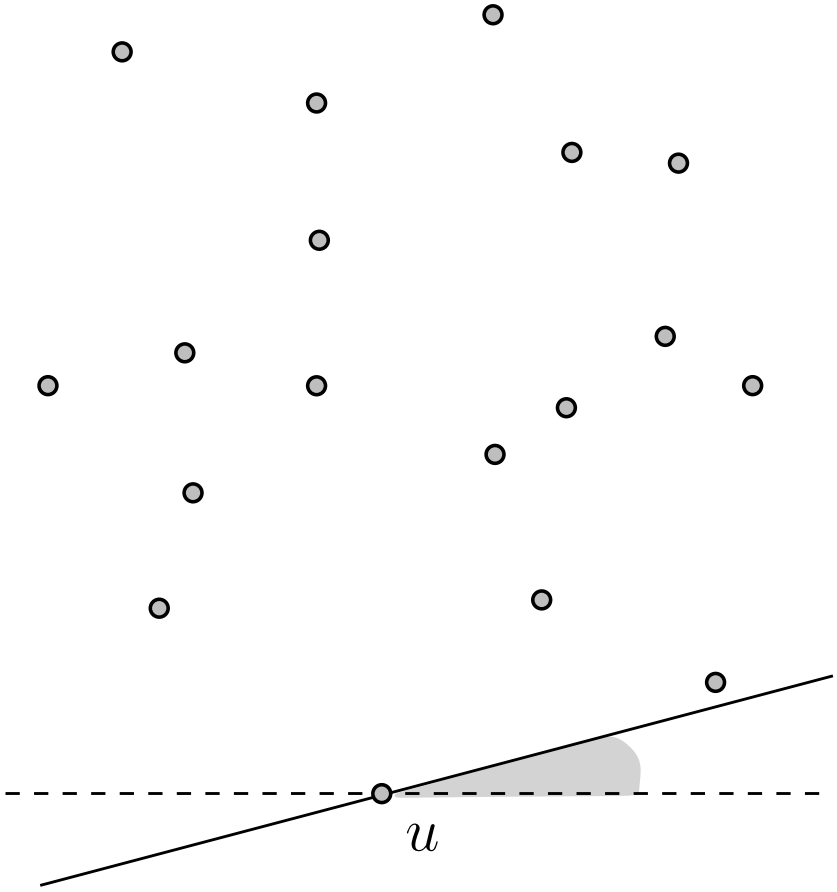
Remark

The lowest point (minimal y-coordinate) always belongs to the boundary of $conv(\mathcal{S})$

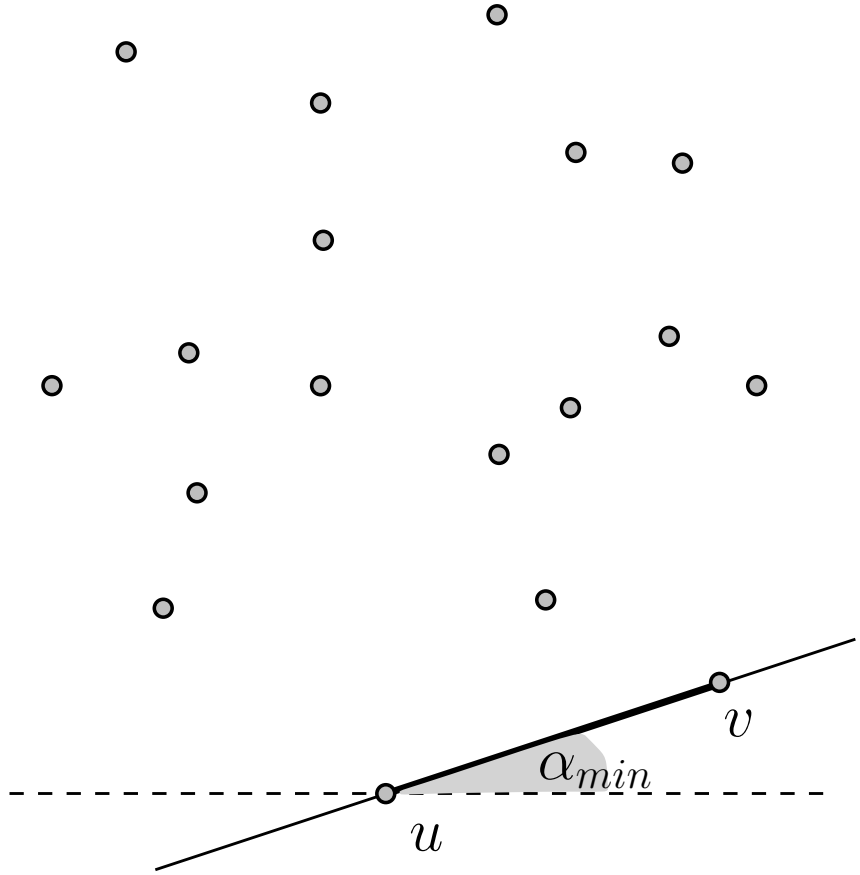
Convex hull computation: first algorithm (Jarvis)



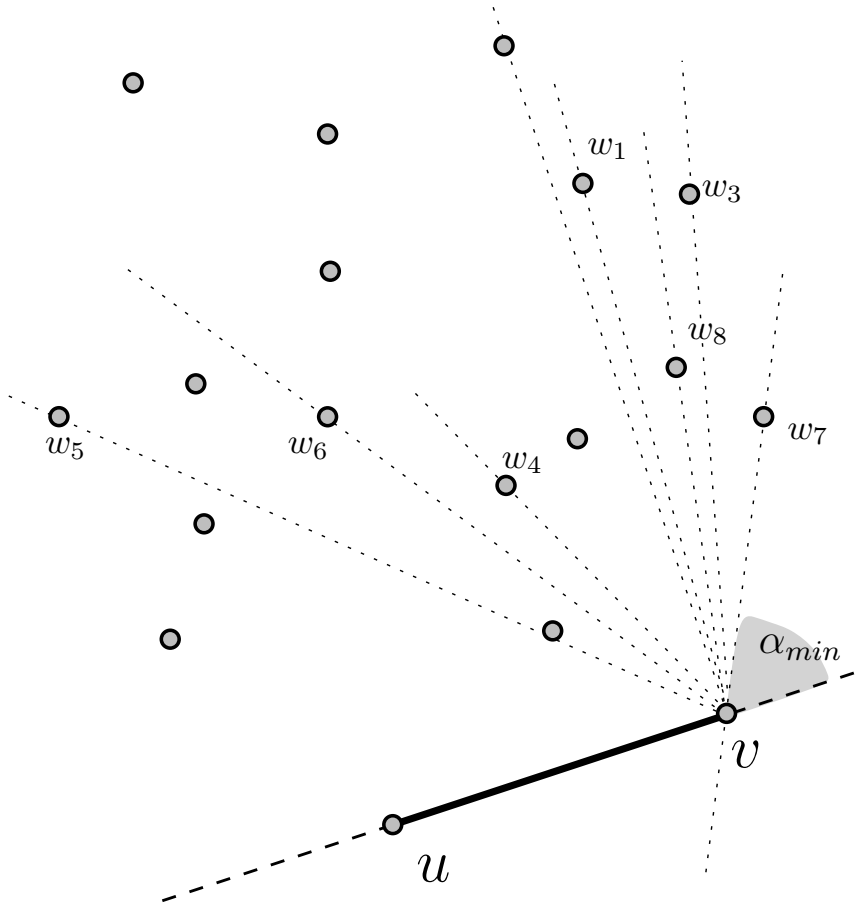
Convex hull computation: first algorithm (Jarvis)



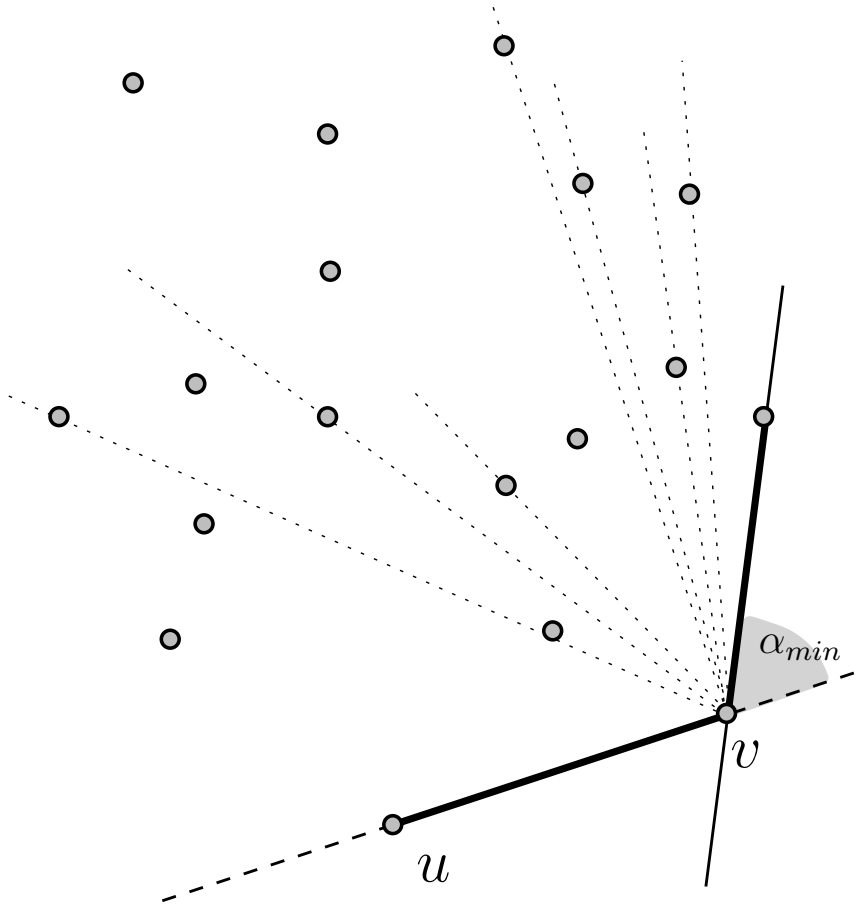
Convex hull computation: first algorithm (Jarvis)



Convex hull computation: first algorithm (Jarvis)

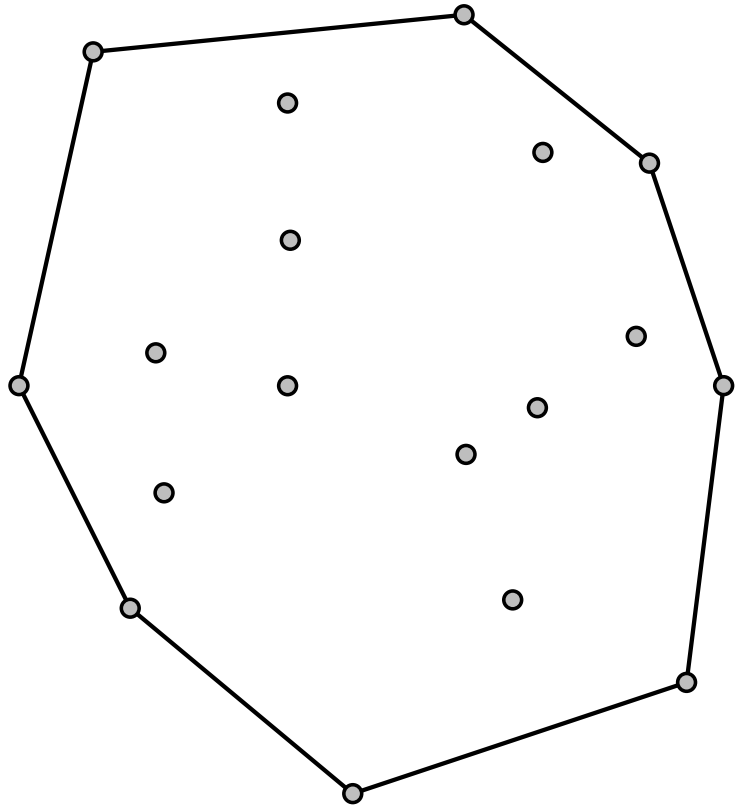


Convex hull computation: first algorithm (Jarvis)



Convex hull computation: first algorithm (Jarvis)

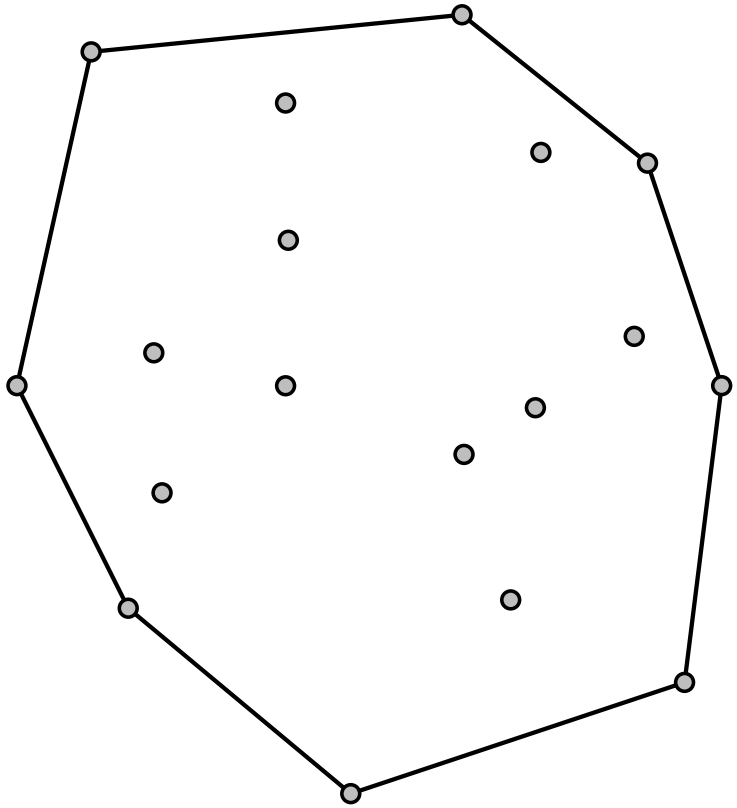
procedure : CH($\mathcal{S}$)
$u =$ lowest point of $\mathcal{S}$;
$min = \infty$
for each $w \in S \setminus \{u\}$
    if $angle(ux, uw) < min$ then $min = angle(ux, uw)$; $v = w$;
$u.suivant = v$;
do
    $S = S \setminus \{v\}$
    for each $w \in S$
        $min = \infty$
        if $angle(v.pred\ v, vw) < min$ then
            $min = angle(v.pred\ v, vw)$; $v.suivant = w$;
    $v = v.suivant$;
while $v \neq u$



Convex hull computation: complexity (Jarvis)

procedure : CH($\mathcal{S}$)
$u =$ lowest point of $\mathcal{S}$;
$min = \infty$
for each $w \in \mathcal{S} \setminus \{u\}$ $O(n)$
    if $angle(ux, uw) < min$ then $min = angle(ux, uw)$; $v = w$;
$u.suivant = v$;
do $O(n)$
    $\mathcal{S} = \mathcal{S} \setminus \{v\}$
    for each $w \in \mathcal{S}$ $O(n)$
        $min = \infty$
        if $angle(v.pred\ v, vw) < min$ then
            $min = angle(v.pred\ v, vw)$; $v.suivant = w$;
    $v = v.suivant$;
while $v \neq u$

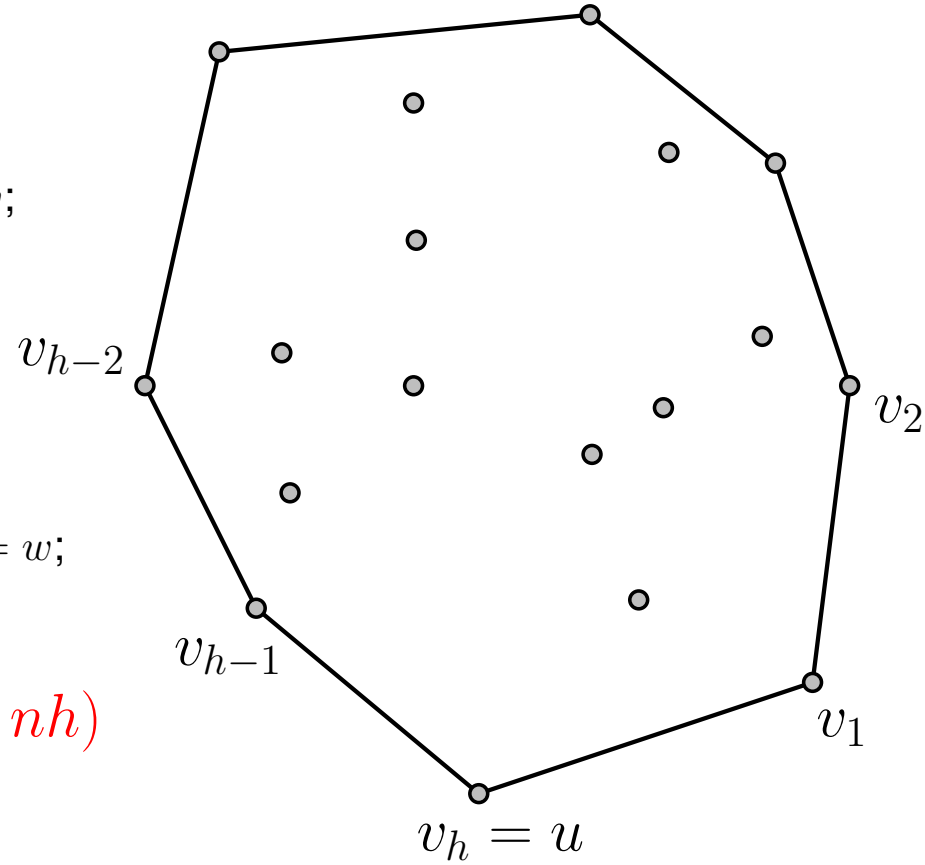
$$T(n) = O(n + n^2)$$



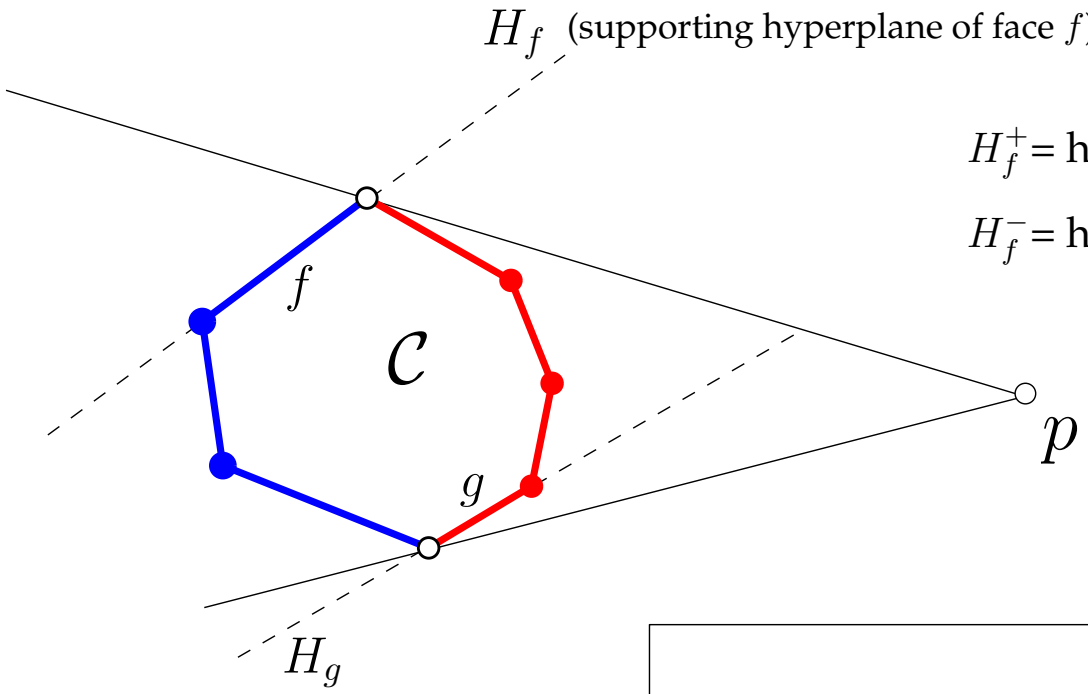
Convex hull computation: complexity (Jarvis)

procedure : CH($\mathcal{S}$)
$u =$ lowest point of $\mathcal{S}$;
$min = \infty$
for each $w \in S \setminus \{u\}$ $O(n)$
    if $angle(ux, uw) < min$ then $min = angle(ux, uw)$; $v = w$;
$u.suivant = v$;
do $O(h)$
    $S = S \setminus \{v\}$
    for each $w \in S$ $O(n)$
        $min = \infty$
        if $angle(v.pred\ v, vw) < min$ then
            $min = angle(v.pred\ v, vw)$; $v.suivant = w$;
    $v = v.suivant$;
while $v \neq u$

$$T(n) = O(n + nh)$$



Geometric properties: visibility

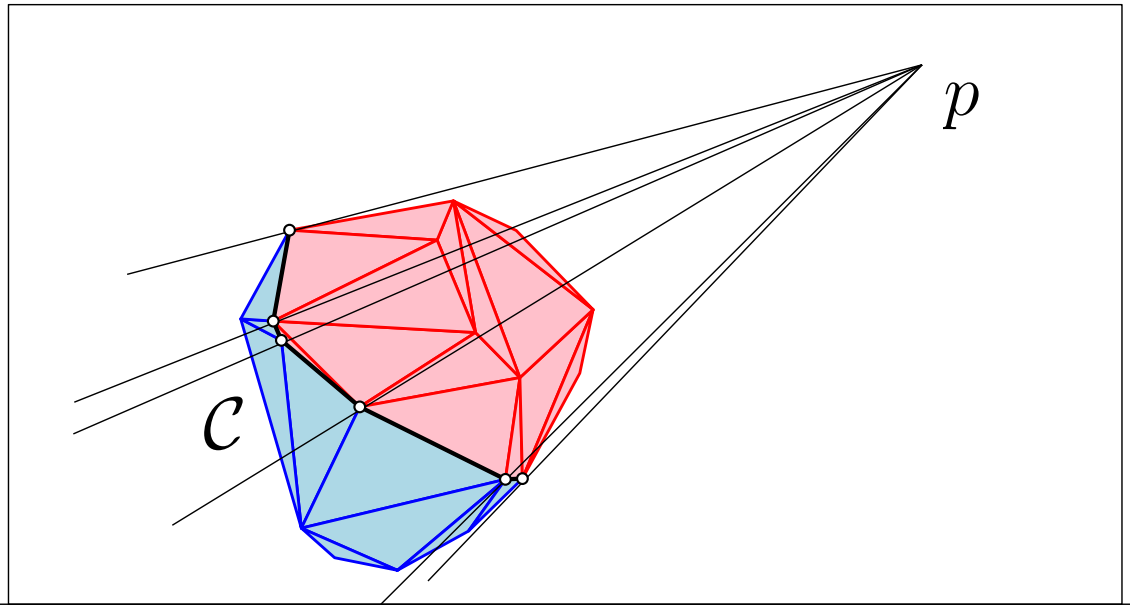


H_f^+ = half-space containing $\mathcal{C}$ and bounded by H_f

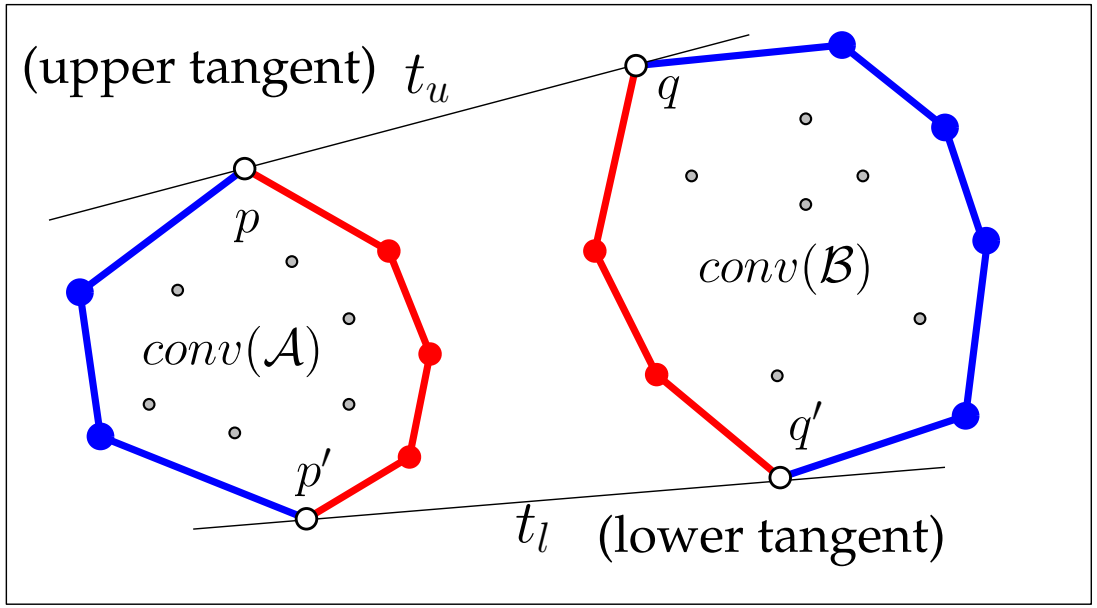
H_f^- = half-space not containing $\mathcal{C}$ and bounded by H_f

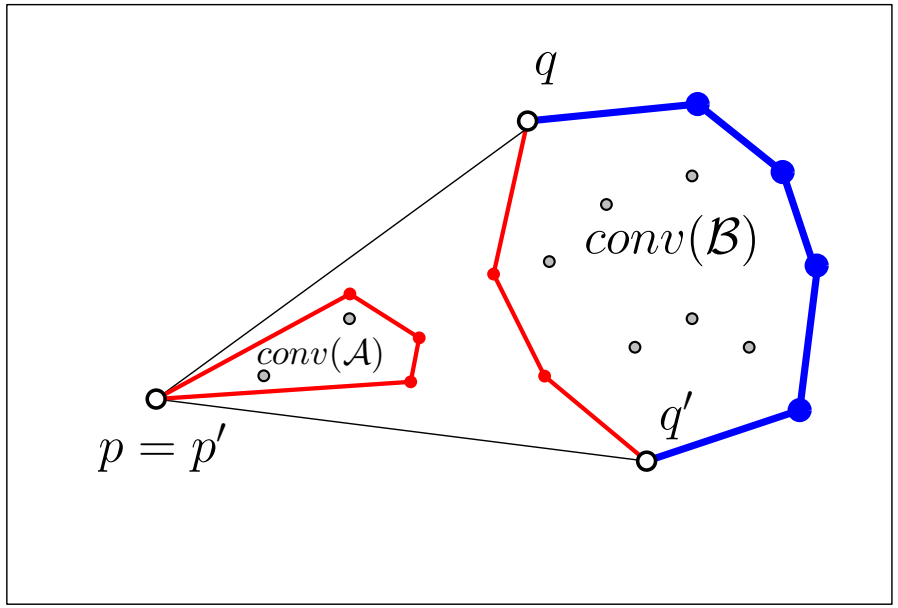
f is not visible from p ($p \in H_f^+$)

g is visible from p ($p \in H_f^-$)



Geometric properties: common tangents





Remarks

The set of red vertices is not empty

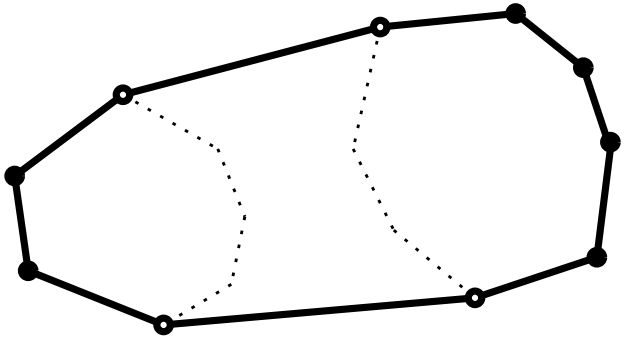
the set of red edges defines a (non-empty) chain

the set of blue edges defines a (possibly empty) chain

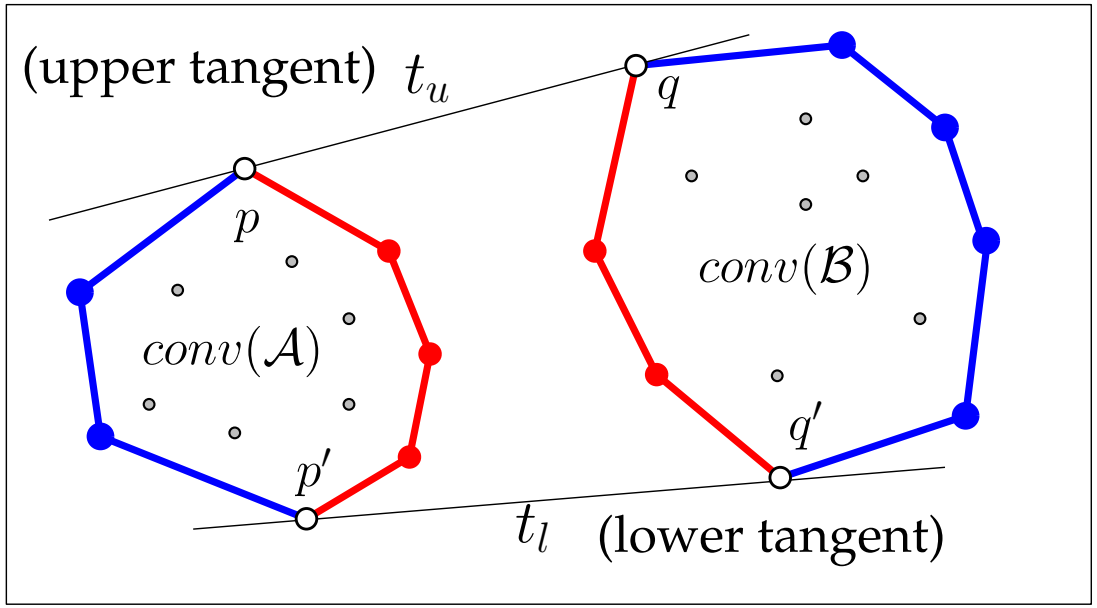
there are at most 4 tangent points

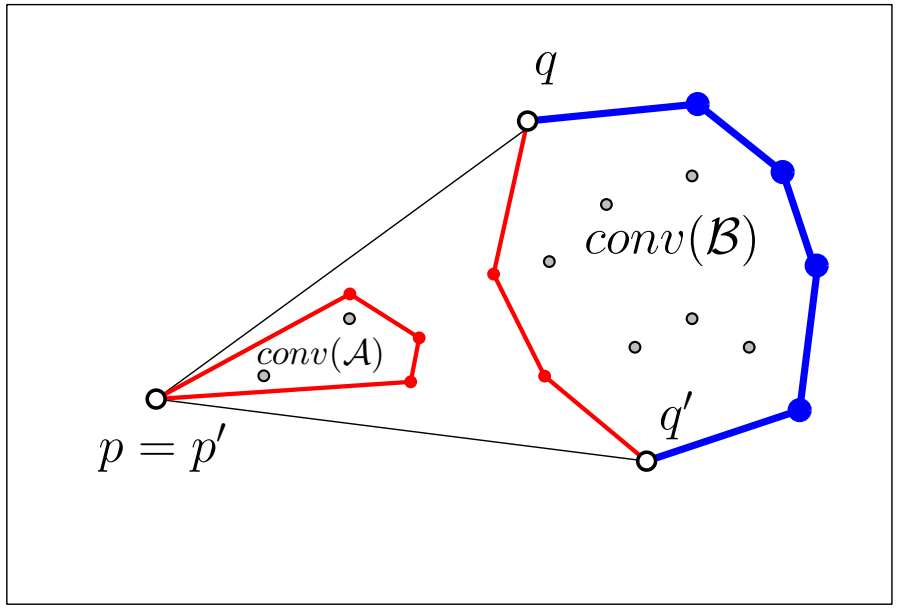
$conv(\mathcal{A} \cup \mathcal{B})$ contains all blue edges, plus the two common (upper and lower) tangents

$conv(\mathcal{A} \cup \mathcal{B})$



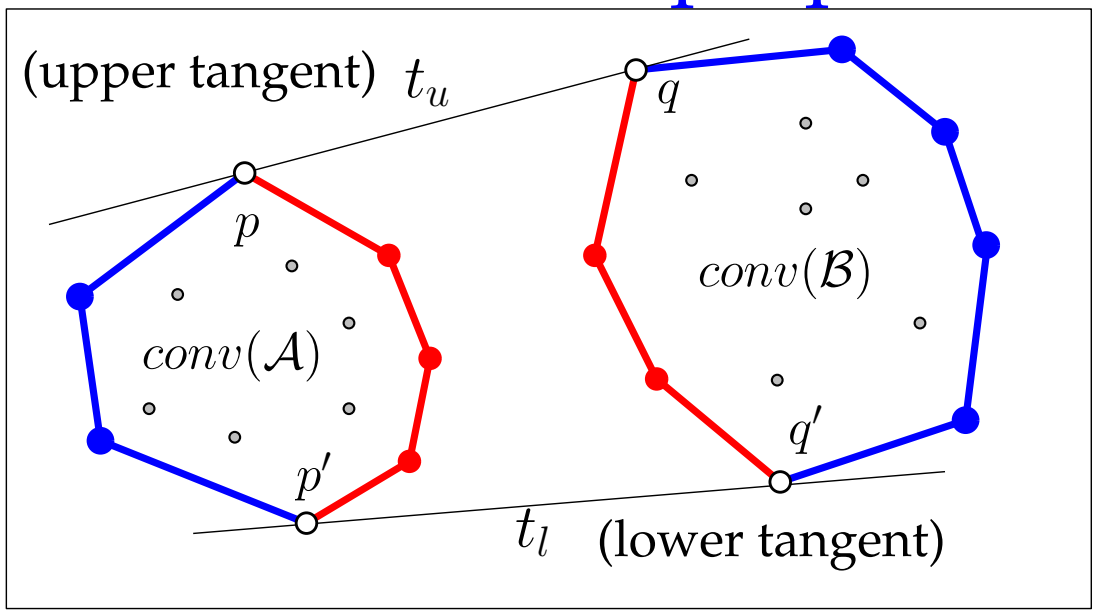
Geometric properties: common tangents

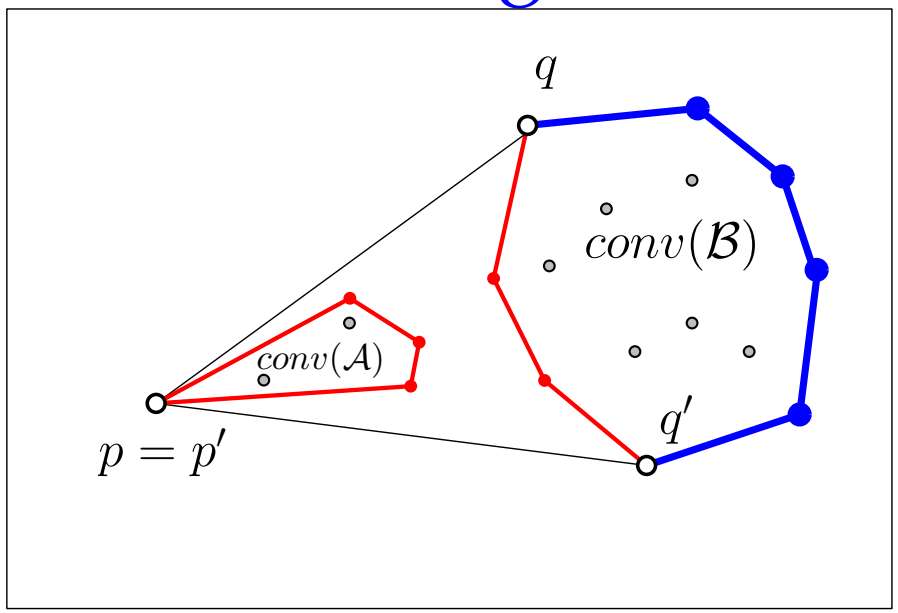




How can we check whether a vertex (or an edge) is visible or not?

Geometric properties: common tangents

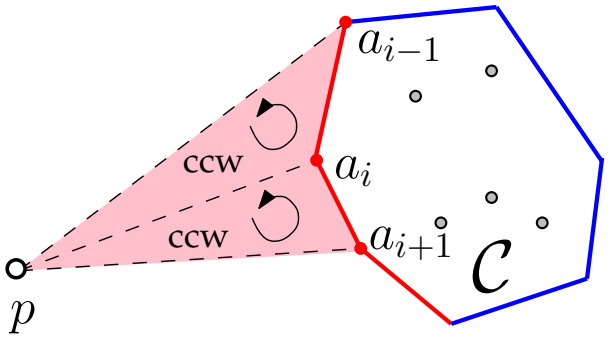




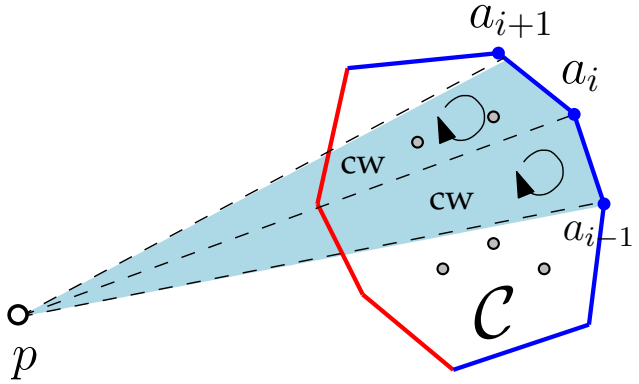
the segment $\overline{pa_i}$ does not intersect $Int(\mathcal{C})$

(p, a_{i+1}, a_i) and (p, a_i, a_{i-1}) are ccw oriented

$$or(p, a_i, a_{i-1}) = sign\left[det\begin{pmatrix} x_i - x_p & x_{i-1} - x_p \\ y_i - y_p & y_{i-1} - y_p \end{pmatrix}\right]$$

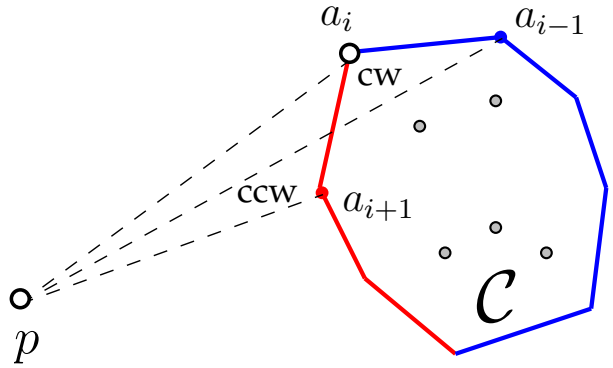


the segment $\overline{pa_i}$ intersects $Int(\mathcal{C})$



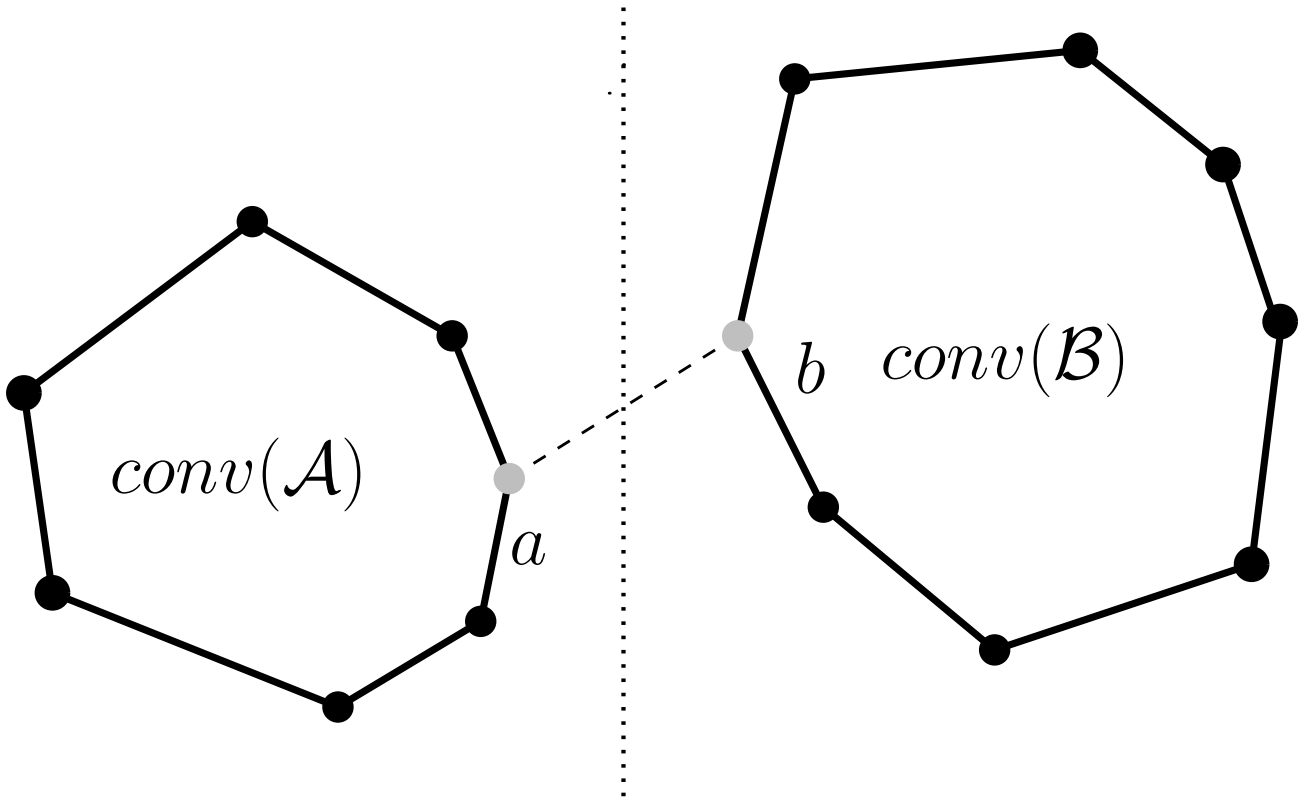
the segment $\overline{pa_i}$ defines the upper tangent

the triangles (p, a_{i+1}, a_i) and (p, a_i, a_{i-1}) have opposite orientations



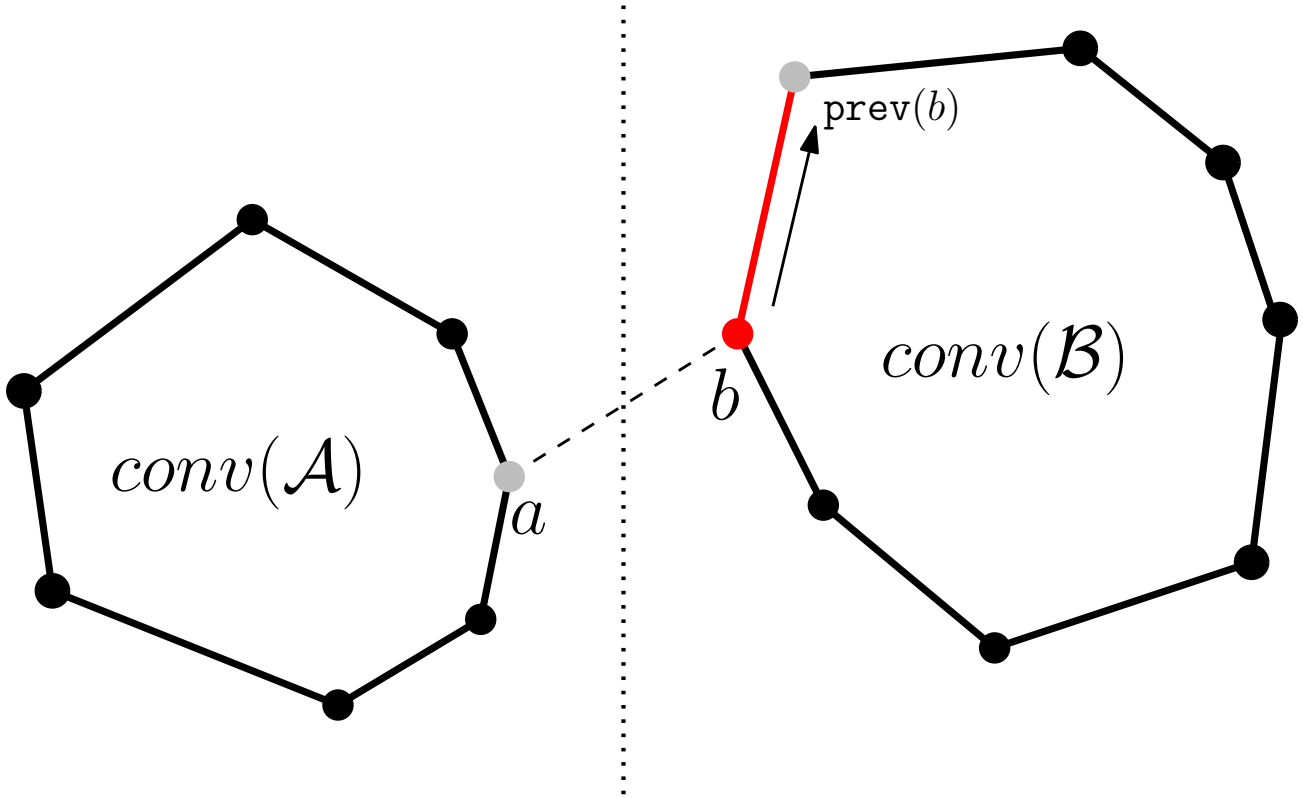
Upper tangent computation

procedure UPPERTANGENT($conv(\mathcal{A}), conv(\mathcal{B})$)
$a \leftarrow$ righmost vertex of $conv(\mathcal{A})$
$b \leftarrow$ leftmost vertex of $conv(\mathcal{B})$
while $(a \in H^+_{(\texttt{prev}(b),b)}$ **or** $b \in H^+_{(a,\texttt{next}(a))})$ — while (a, b) is not the upper tangent
$\begin{cases} \textbf{if } a \in H^+_{(\texttt{prev}(b),b)} \textbf{ then } b \leftarrow \texttt{prev}(b) \\ \textbf{else } a \leftarrow \texttt{next}(a) \end{cases}$
return (a, b)



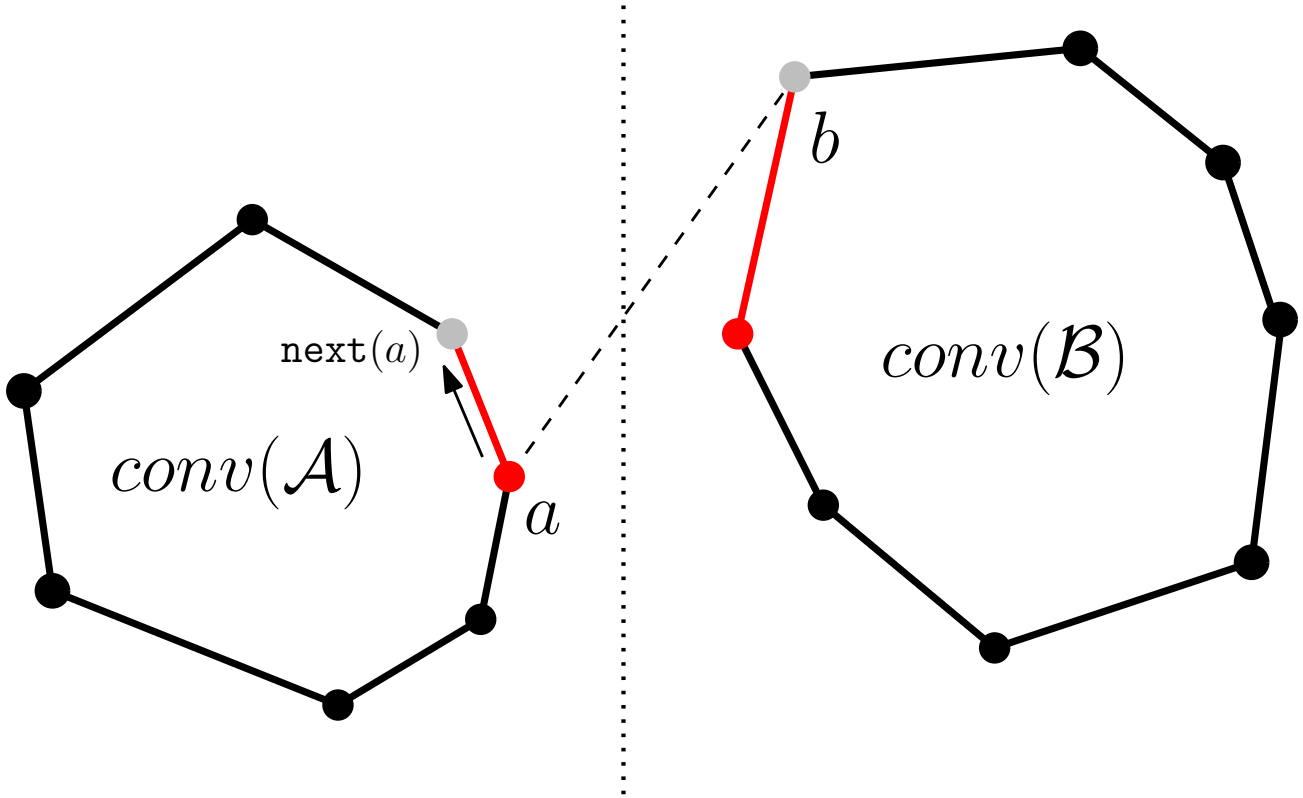
Upper tangent computation

procedure UPPERTANGENT($conv(\mathcal{A}), conv(\mathcal{B})$)
$a \leftarrow$ righmost vertex of $conv(\mathcal{A})$
$b \leftarrow$ leftmost vertex of $conv(\mathcal{B})$
while $(a \in H^+_{(\texttt{prev}(b),b)}$ **or** $b \in H^+_{(a,\texttt{next}(a))})$ — while (a, b) is not the upper tangent
$\begin{cases} \textbf{if } a \in H^+_{(\texttt{prev}(b),b)} \textbf{ then } b \leftarrow \texttt{prev}(b) \\ \textbf{else } a \leftarrow \texttt{next}(a) \end{cases}$
return (a, b)



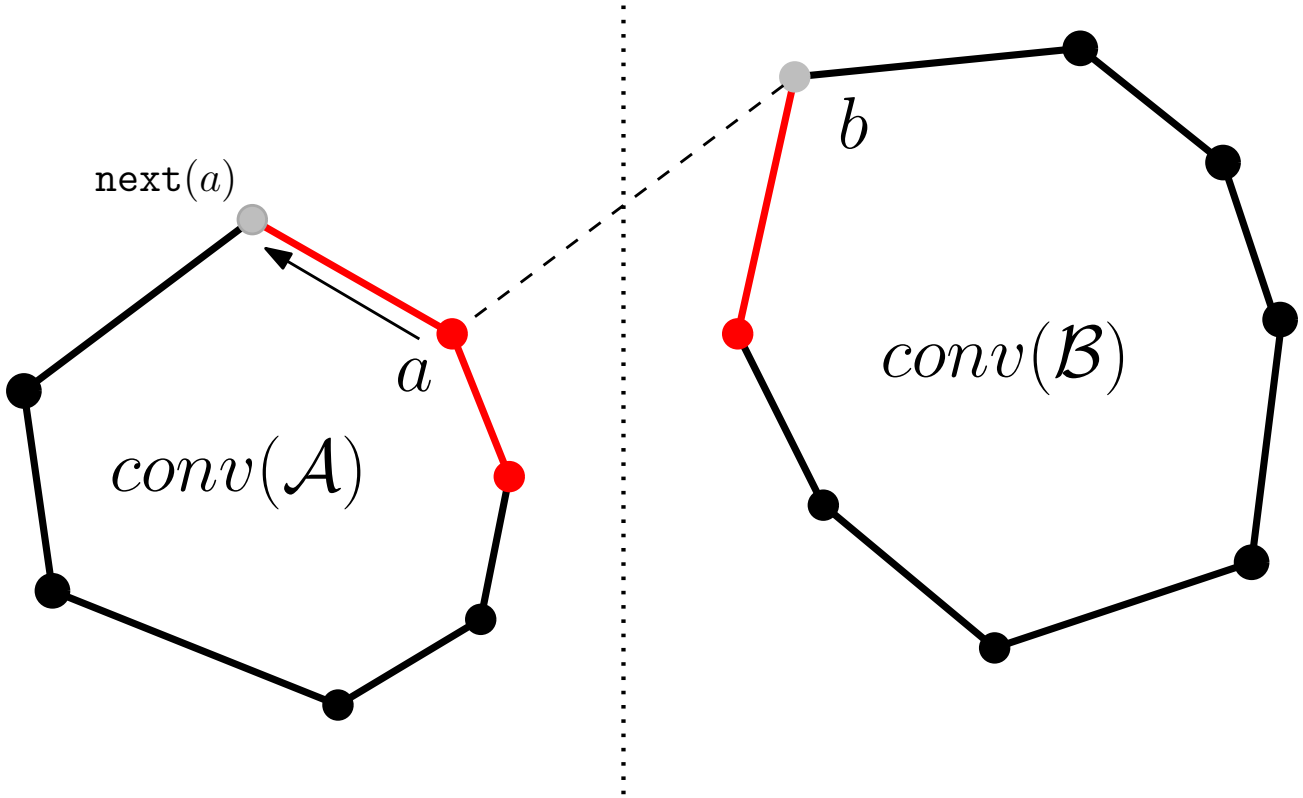
Upper tangent computation

procedure UPPERTANGENT($conv(\mathcal{A}), conv(\mathcal{B})$)
$a \leftarrow$ righmost vertex of $conv(\mathcal{A})$
$b \leftarrow$ leftmost vertex of $conv(\mathcal{B})$
while $(a \in H^+_{(\texttt{prev}(b),b)}$ **or** $b \in H^+_{(a,\texttt{next}(a))})$ — while (a, b) is not the upper tangent
$\begin{cases} \textbf{if } a \in H^+_{(\texttt{prev}(b),b)} \textbf{ then } b \leftarrow \texttt{prev}(b) \\ \textbf{else } a \leftarrow \texttt{next}(a) \end{cases}$
return (a, b)



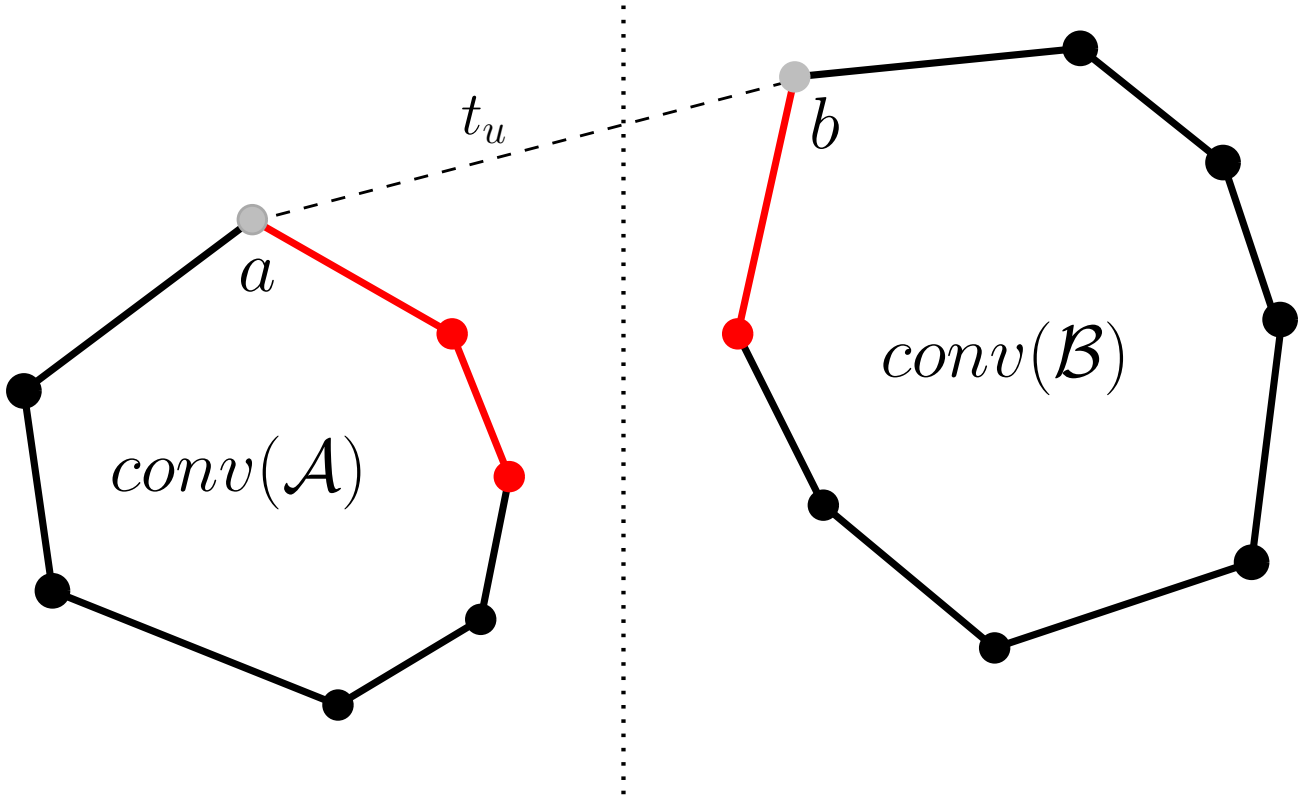
Upper tangent computation

procedure UPPERTANGENT($conv(\mathcal{A}), conv(\mathcal{B})$)
$a \leftarrow$ righmost vertex of $conv(\mathcal{A})$
$b \leftarrow$ leftmost vertex of $conv(\mathcal{B})$
while $(a \in H^+_{(\texttt{prev}(b),b)}$ **or** $b \in H^+_{(a,\texttt{next}(a))})$ — while (a, b) is not the upper tangent
$\begin{cases} \textbf{if } a \in H^+_{(\texttt{prev}(b),b)} \textbf{ then } b \leftarrow \texttt{prev}(b) \\ \textbf{else } a \leftarrow \texttt{next}(a) \end{cases}$
return (a, b)

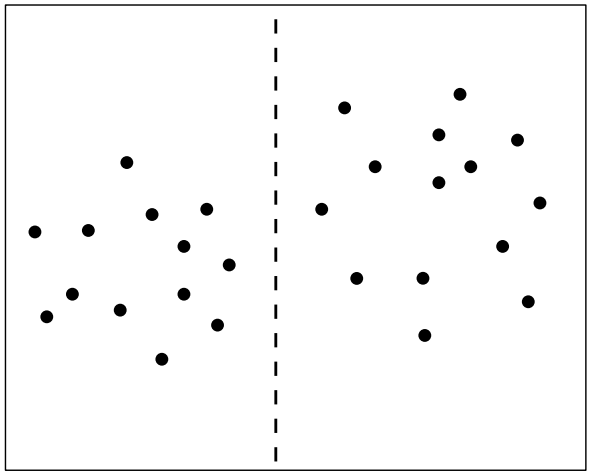


Upper tangent computation

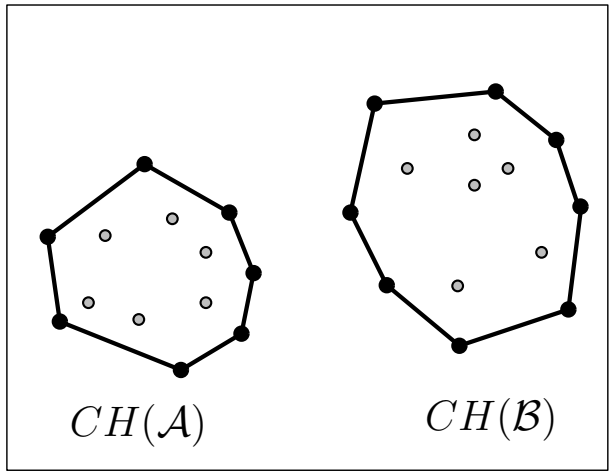
procedure UPPERTANGENT($conv(\mathcal{A}), conv(\mathcal{B})$)
$a \leftarrow$ righmost vertex of $conv(\mathcal{A})$
$b \leftarrow$ leftmost vertex of $conv(\mathcal{B})$
while $(a \in H^+_{(\texttt{prev}(b),b)}$ **or** $b \in H^+_{(a,\texttt{next}(a))})$ — while (a, b) is not the upper tangent
  if $a \in H^+_{(\texttt{prev}(b),b)}$ **then** $b \leftarrow \texttt{prev}(b)$
  else $a \leftarrow \texttt{next}(a)$
return (a, b)

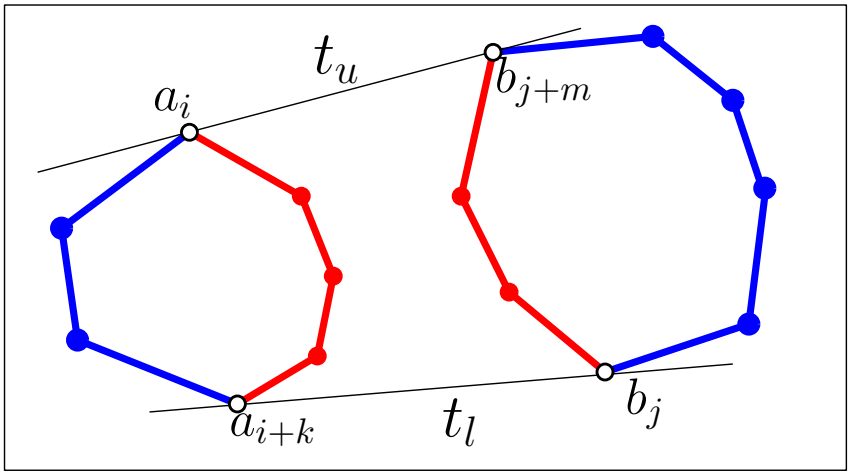


2D Convex hull: optimal *divide and conquer* algorithm



$\texttt{split}_\texttt{x}(\mathcal{S})$



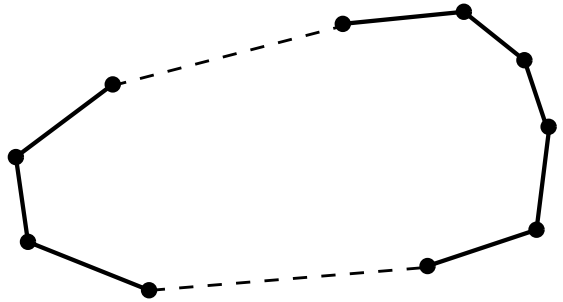


procedure CH(S)
if $n \leq 3$ **then return** $conv\{p_1, \ldots, p_n\}$ — using brute force
else
let $(\mathcal{A}, \mathcal{B}) = verticalSplit(S)$ — where $|\mathcal{A}| = \lceil n/2 \rceil$ and $|\mathcal{B}| = \lfloor n/2 \rfloor$
return $merge(CH(\mathcal{A}), CH(\mathcal{B}))$

$$l_A = \texttt{split}(conv(\mathcal{A}), a_i, a_{i+k})$$

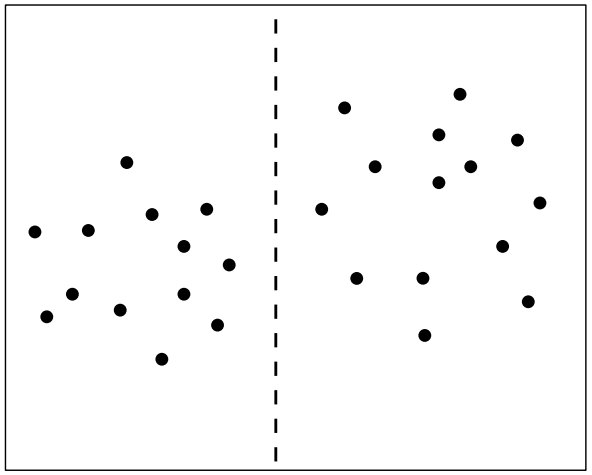
$$l_B = \texttt{split}(conv(\mathcal{B}), b_j, b_{j+m})$$

$$CH(\mathcal{A} \bigcup \mathcal{B}) = \texttt{merge}(l_A, t_l, l_b, t_u)$$

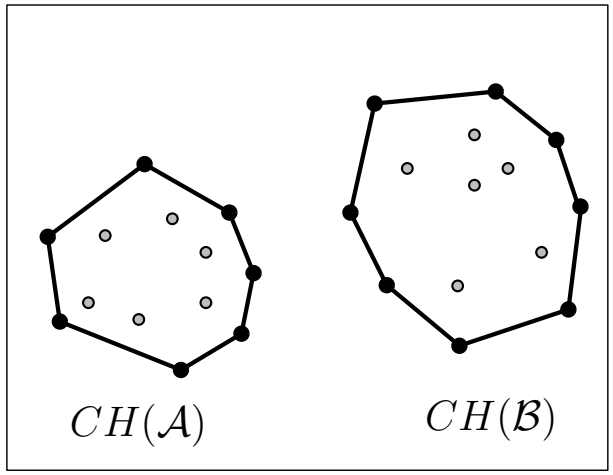


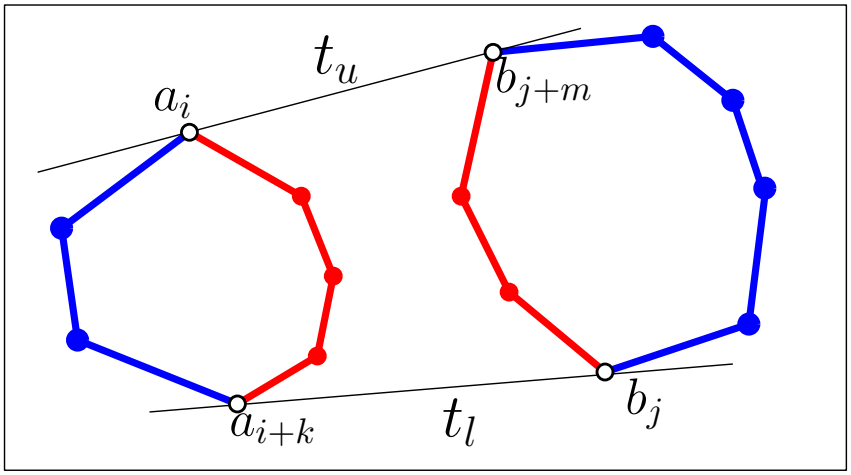
2D Convex hull: optimal *divide and conquer* algorithm

$O(n \log n) + O(1)$ $T(n/2)$ $T(n/2)$ $O(n)$



$\texttt{split}_x(\mathcal{S})$





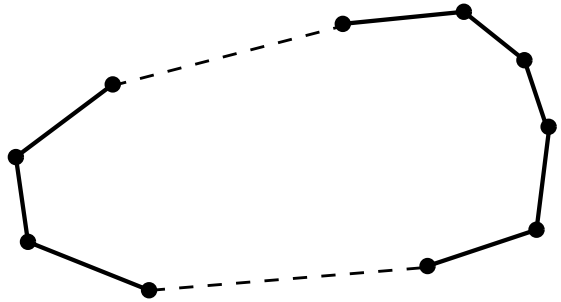
procedure CH(S)
if $n \leq 3$ **then return** $conv\{p_1, \ldots, p_n\}$ — using brute force
else
let $(\mathcal{A}, \mathcal{B}) = verticalSplit(S)$ — where $|\mathcal{A}| = \lceil n/2 \rceil$ and $|\mathcal{B}| = \lfloor n/2 \rfloor$
return $merge(CH(\mathcal{A}), CH(\mathcal{B}))$

$O(1)$ $l_A = \texttt{split}(conv(\mathcal{A}), a_i, a_{i+k})$

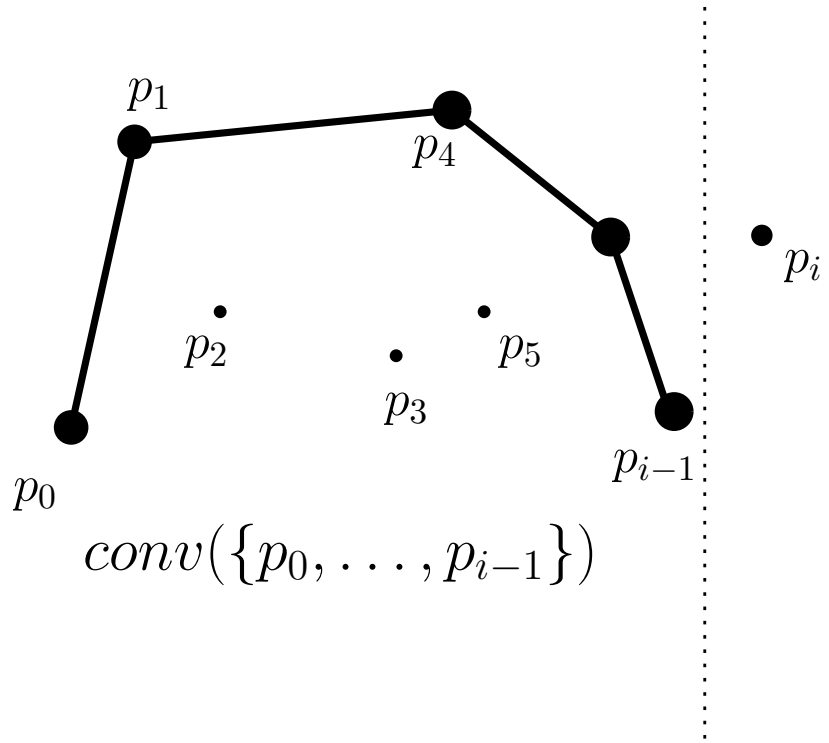
$O(1)$ $l_B = \texttt{split}(conv(\mathcal{B}), b_j, b_{j+m})$

$O(1)$ $CH(\mathcal{A} \bigcup \mathcal{B}) = \texttt{merge}(l_A, t_l, l_b, t_u)$

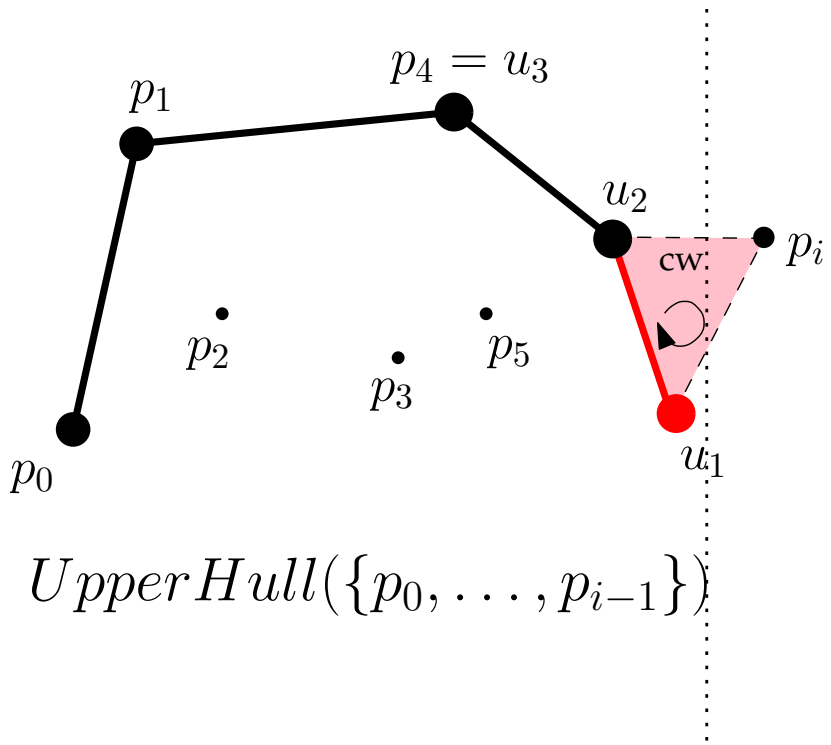
$$T(n) = \begin{cases} O(1) & n \leq 3 \\ O(n) + 2T(\frac{n}{2}) & n > 3 \end{cases} \longrightarrow T(n) = O(n \log n)$$



2D Convex hull: Graham-Andrew scan

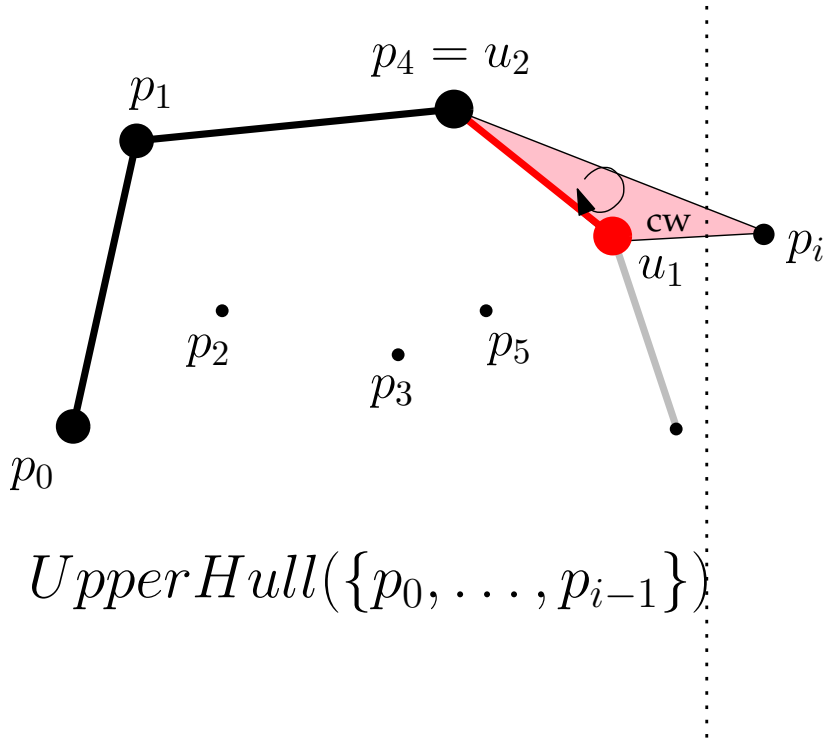


2D Convex hull: Graham-Andrew scan



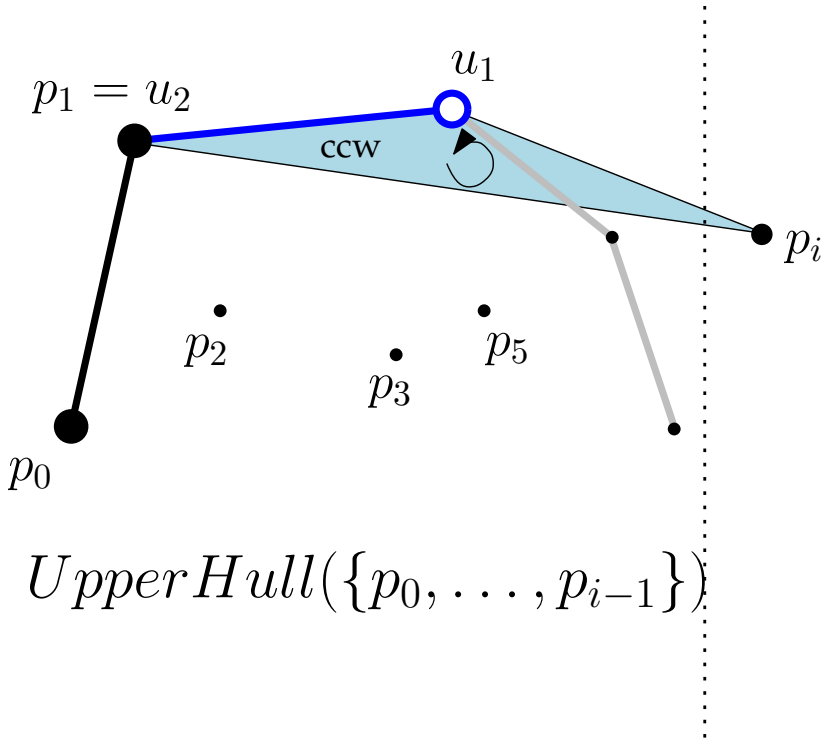
$UpperHull(\{p_0, \ldots, p_{i-1}\})$

2D Convex hull: Graham-Andrew scan

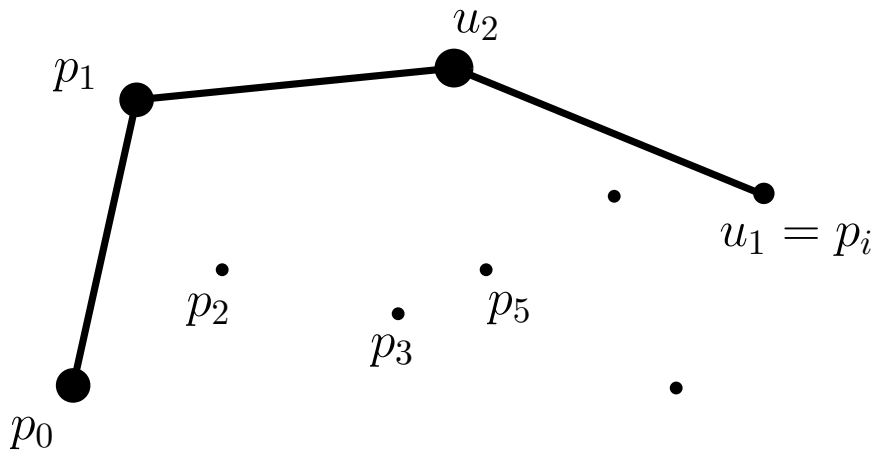


$UpperHull(\{p_0, \ldots, p_{i-1}\})$

2D Convex hull: Graham-Andrew scan

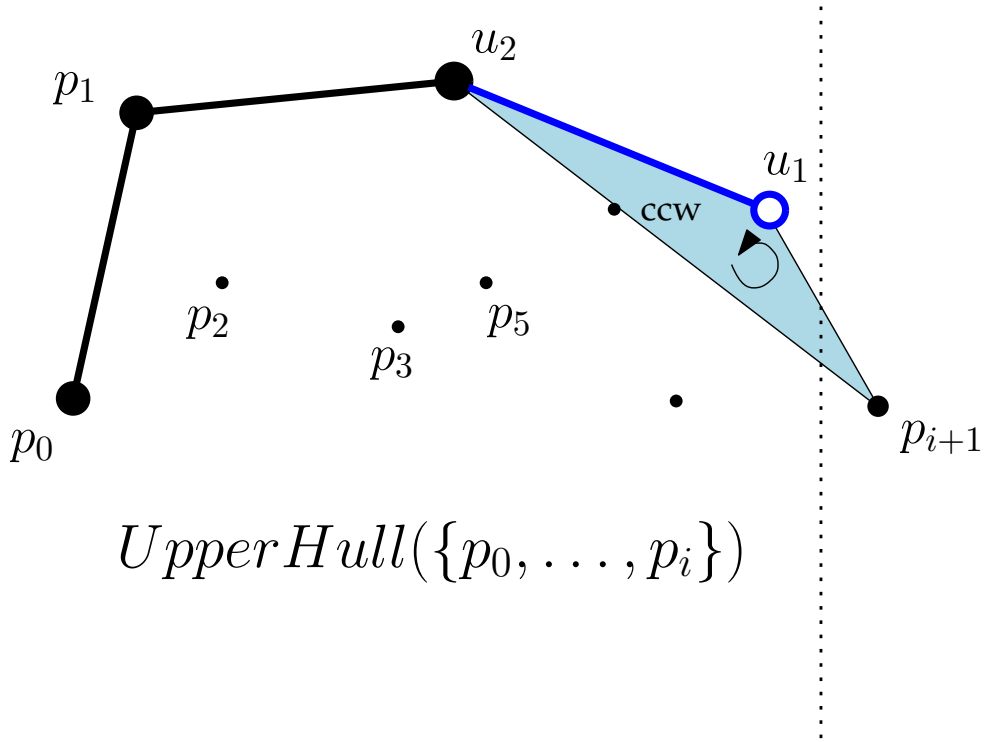


2D Convex hull: Graham-Andrew scan



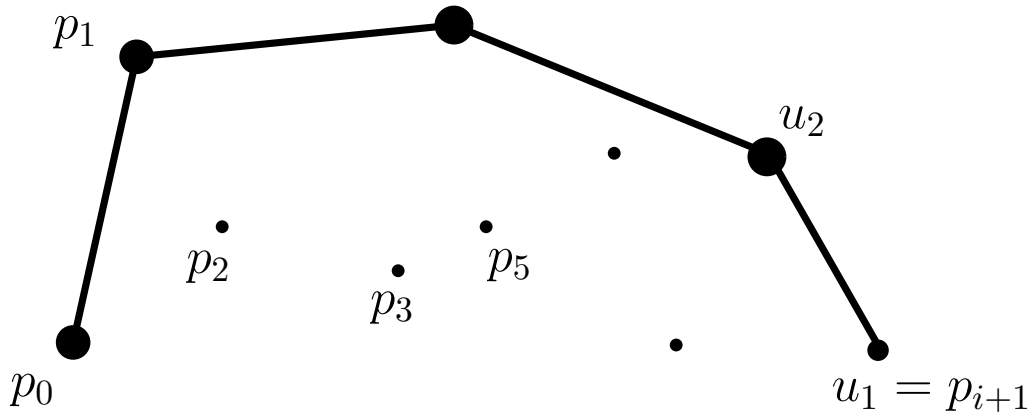
$UpperHull(\{p_0, \ldots, p_i\})$

2D Convex hull: Graham-Andrew scan



$UpperHull(\{p_0, \ldots, p_i\})$

2D Convex hull: Graham-Andrew scan

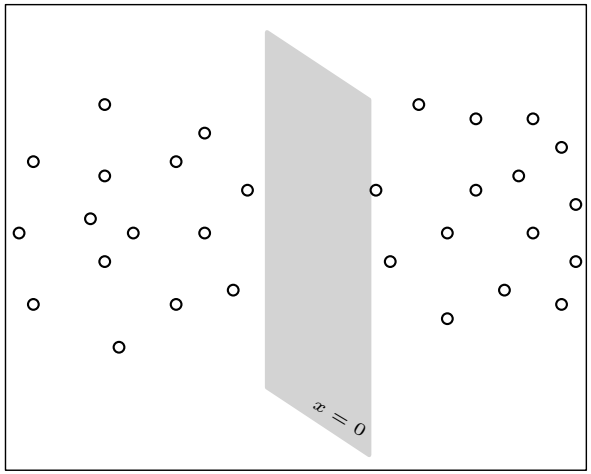


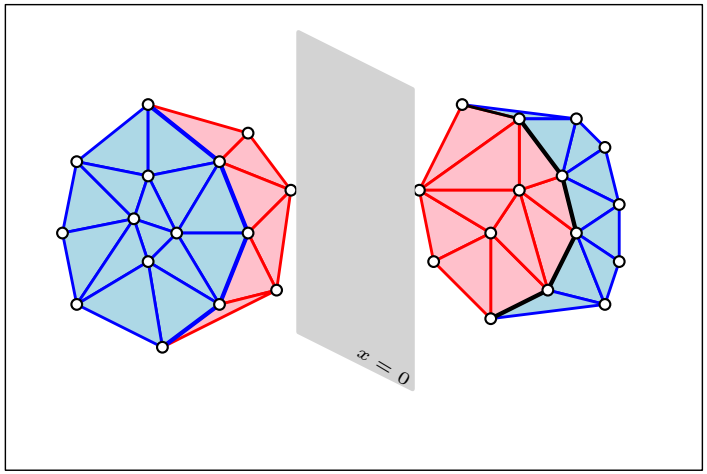
$UpperHull(\{p_0, \ldots, p_{i+1}\})$

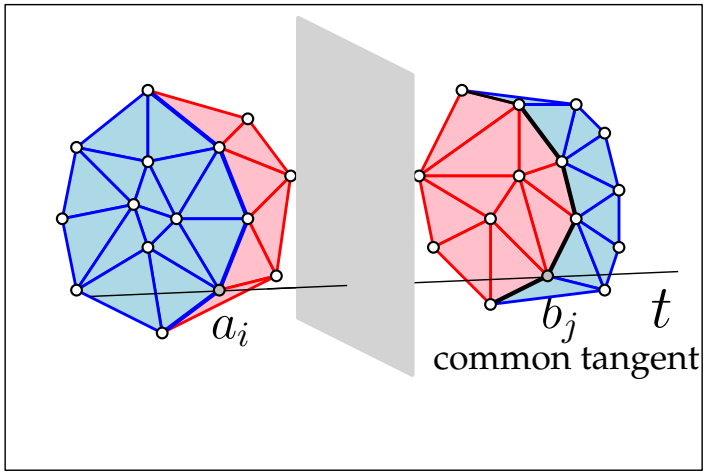
Entrée : S un ensemble de n points.
trier les points de S selon les coordonnées x croissantes;
initialiser $U = [p_1, p_2]$; *// pile représentant l'enveloppe convexe*
pour tout point $p \in \{p_3, \ldots, p_n\}$ faire $O(n)$
    soient u_1 et u_2 les deux premiers sommets de U;
    tant que (p, u_1, u_2) est orienté cw $O(n)$?
        dépiler u_{first} de U
    empiler p dans U
retourner U

3D Convex hull: optimal *divide and conquer* algorithm

$O(n \log n) + O(1)$ $T(n/2)$ $T(n/2)$ $O(n)$

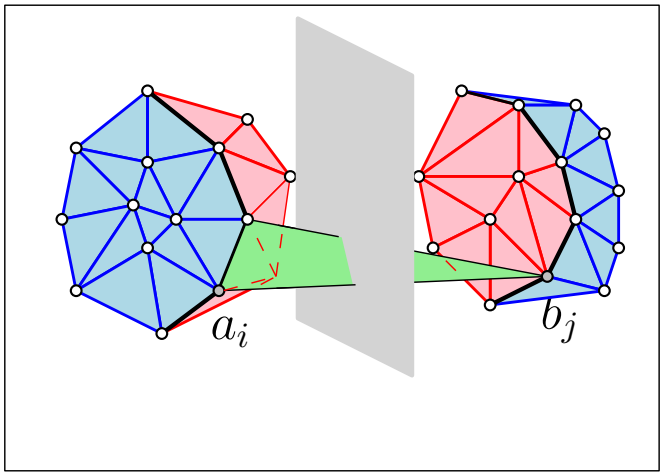


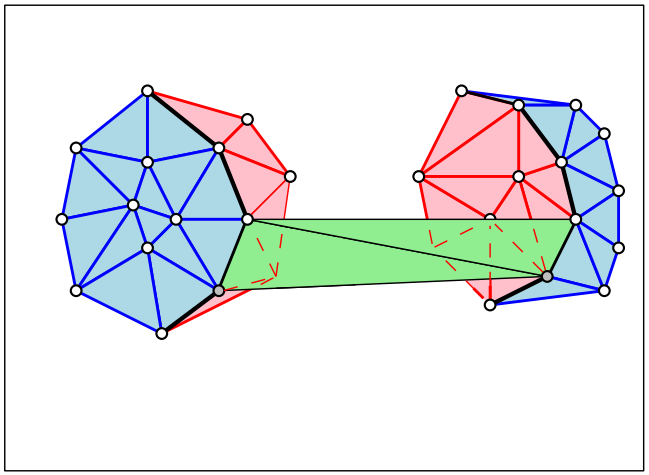


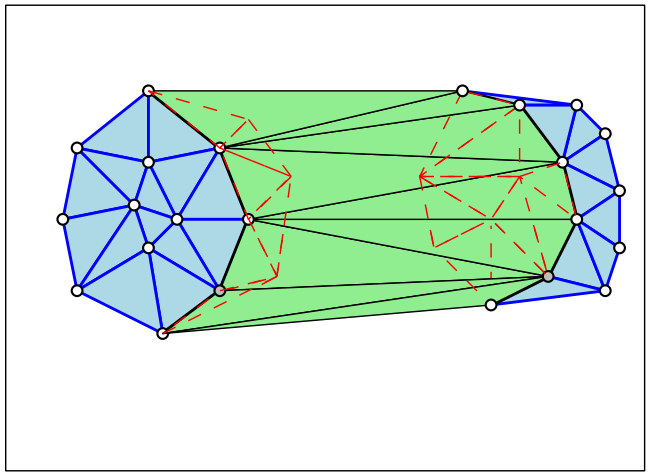


$\texttt{split}_\texttt{x}(\mathcal{S})$

$O(n)$

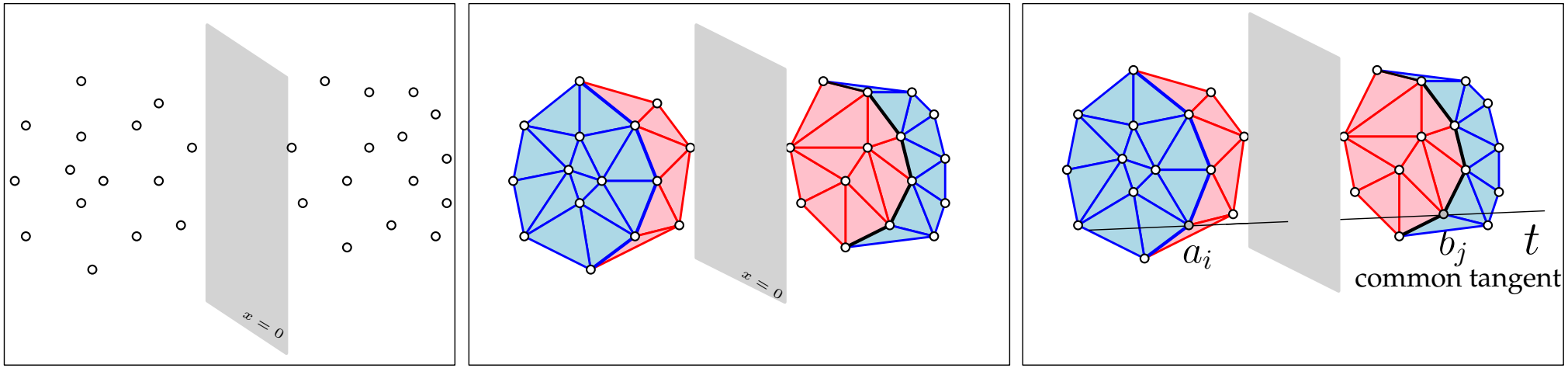




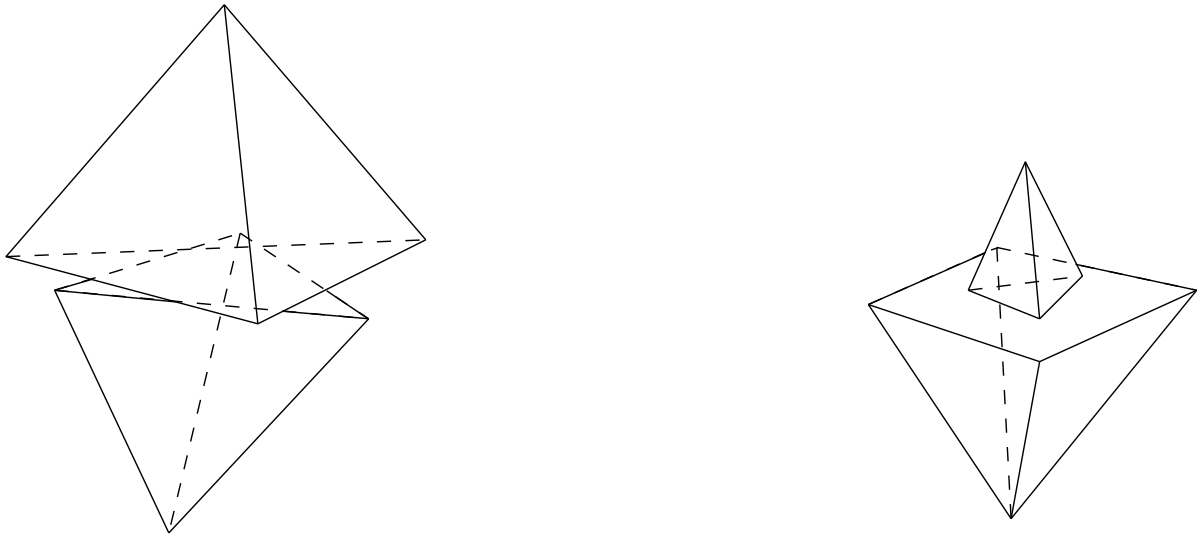


3D Convex hull: optimal *divide and conquer* algorithm

$O(n \log n) + O(1)$ $T(n/2)$ $T(n/2)$ $O(n)$

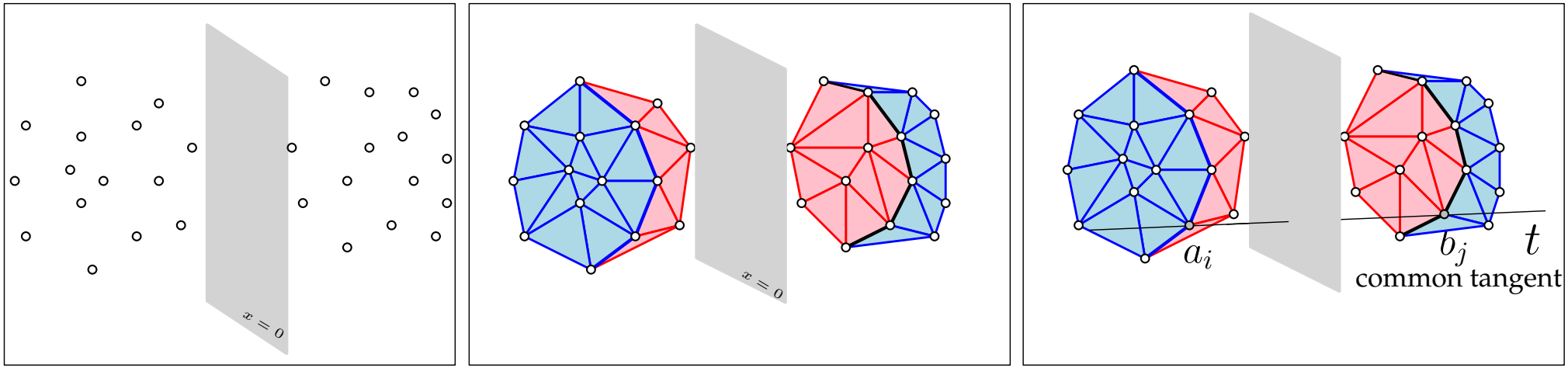


things are more complicated in the 3D world



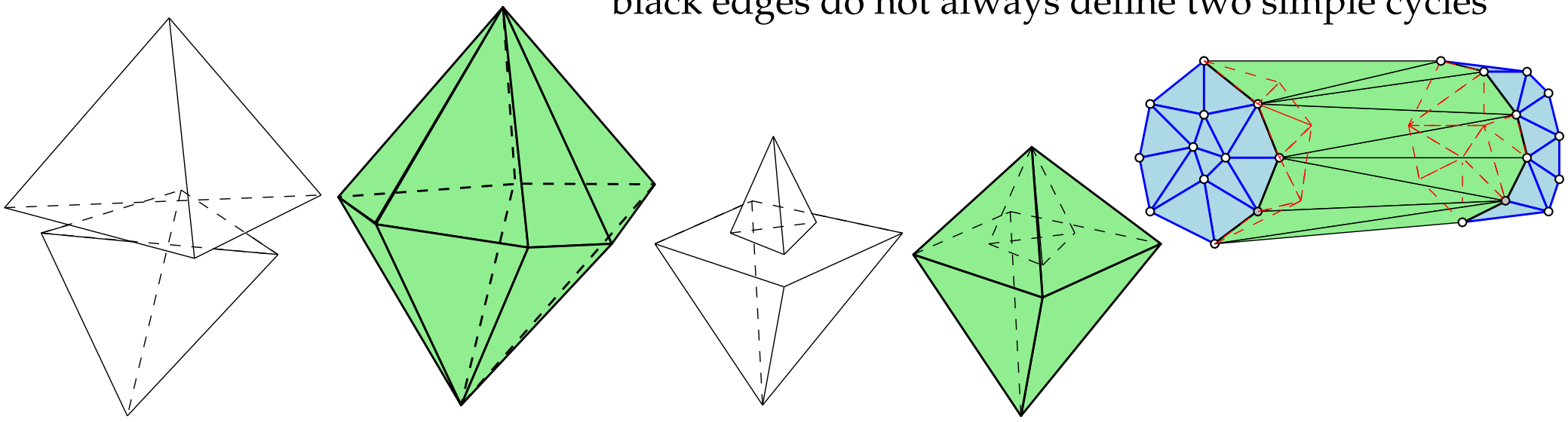
3D Convex hull: optimal *divide and conquer* algorithm

$O(n \log n) + O(1)$ $T(n/2)$ $T(n/2)$ $O(n)$

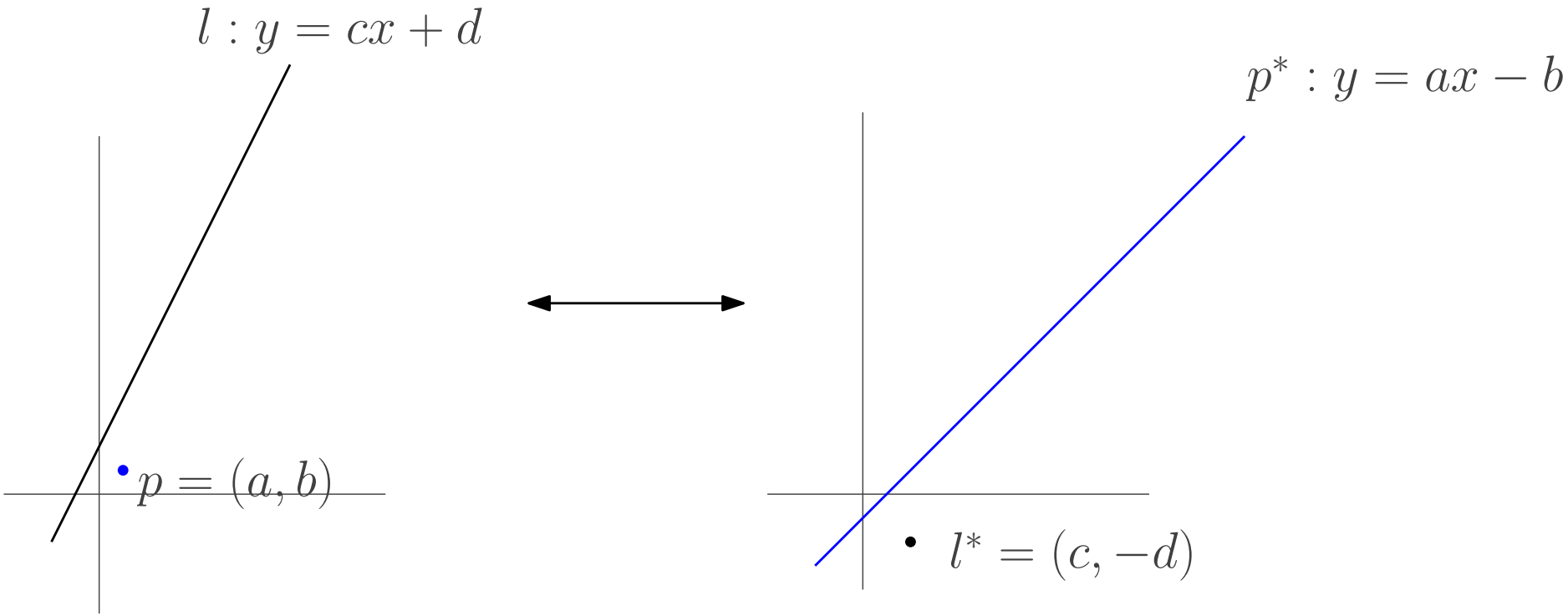


things are more complicated in the 3D world

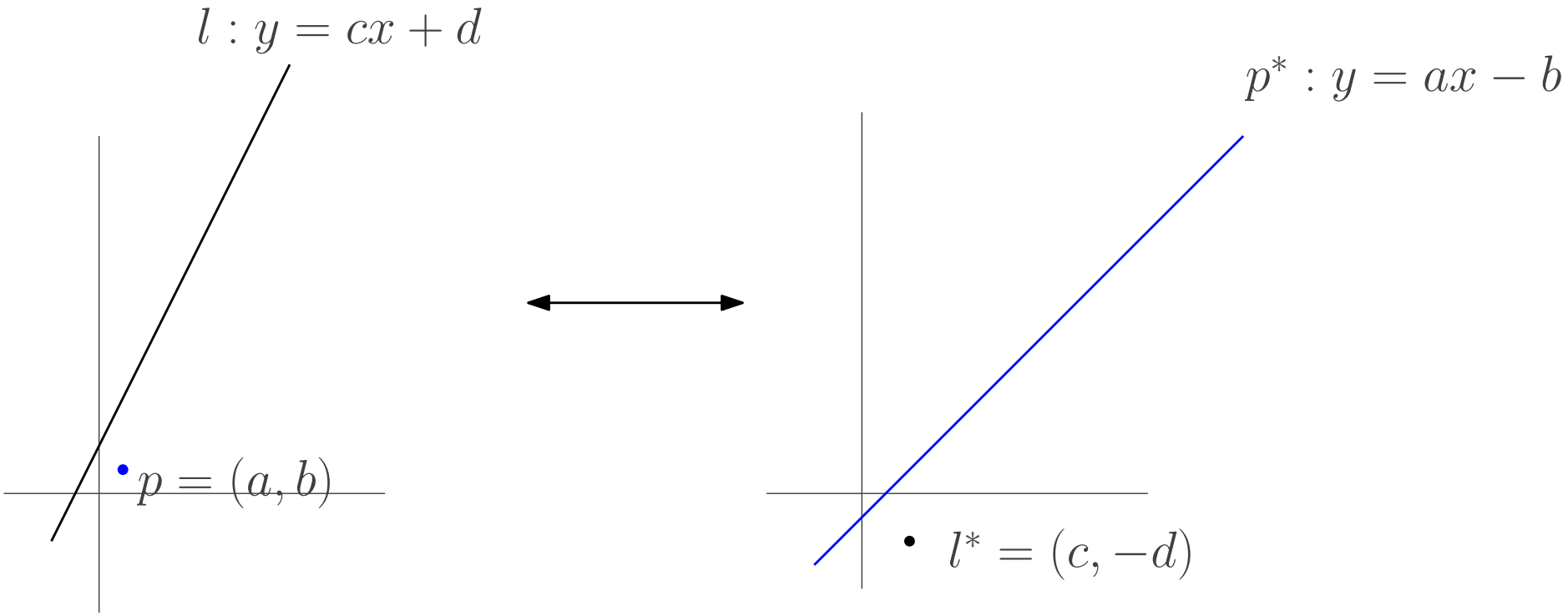
black edges do not always define two simple cycles



Duality in 2d: lines and points

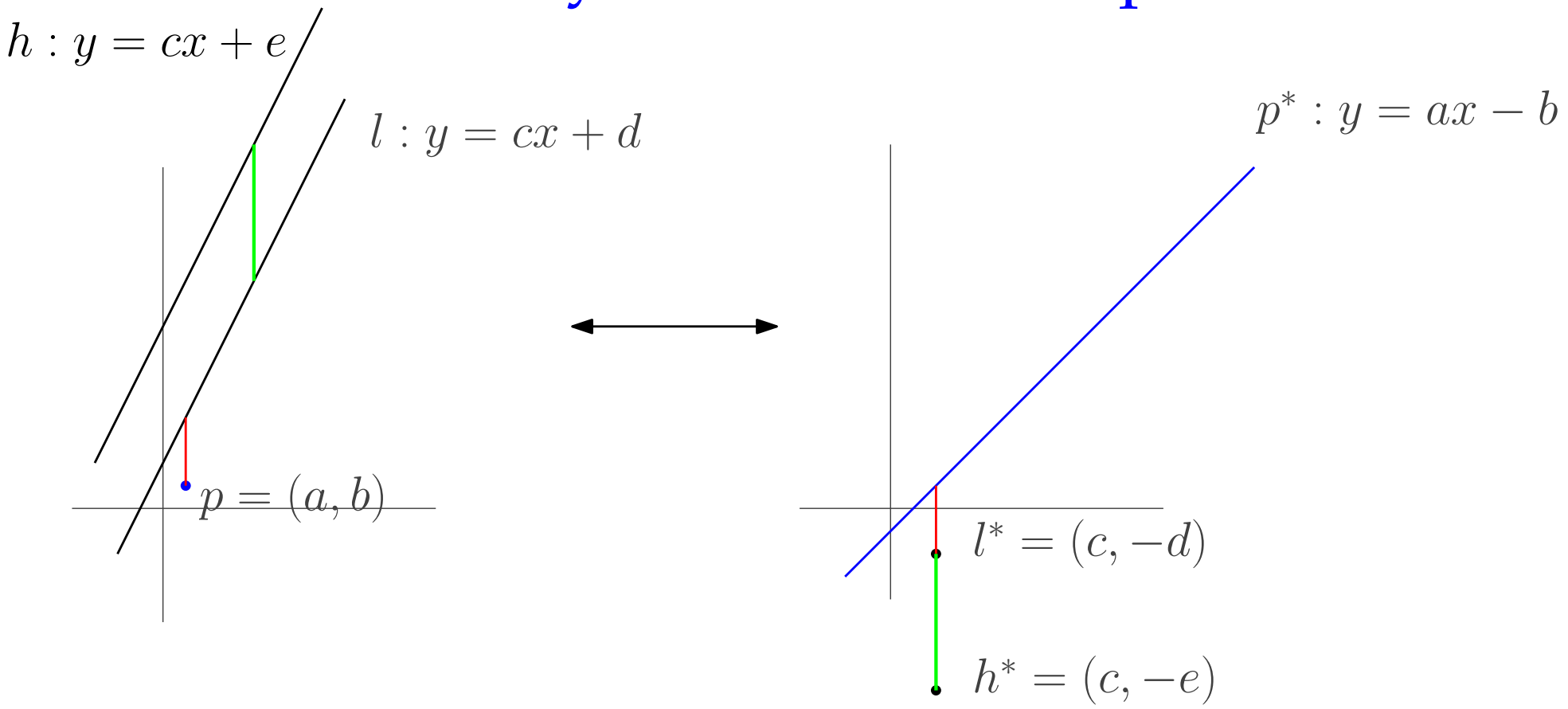


Duality in 2d: lines and points



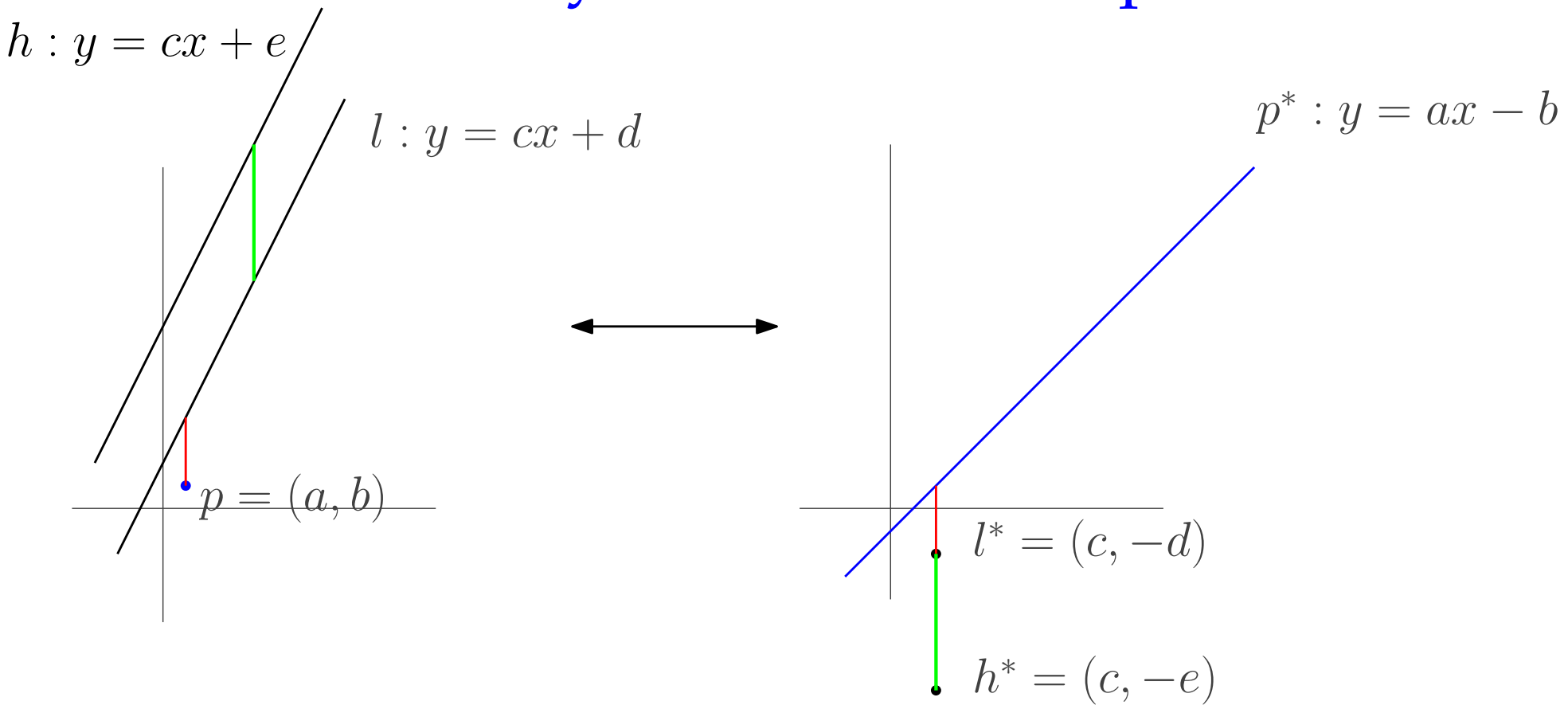
- $(p^*)^* = p$ $\qquad\qquad$ $(p^*)^* := (a, -(-b)) = p$
- p is below l if and only if l^* is below p^*

Duality in 2d: lines and points



- the vertical distance between p and l is equal to the vertical distance between l^* and p^*
- the vertical distance between l and h is equal to length of (l^*h^*)

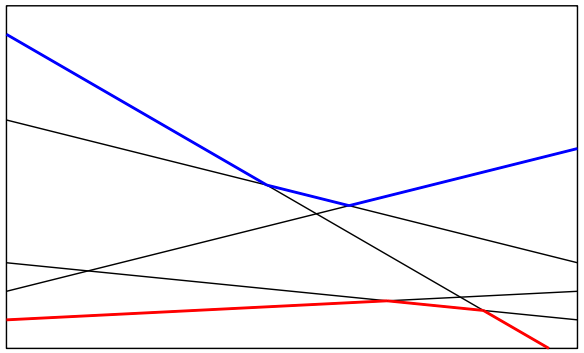
Duality in 2d: lines and points

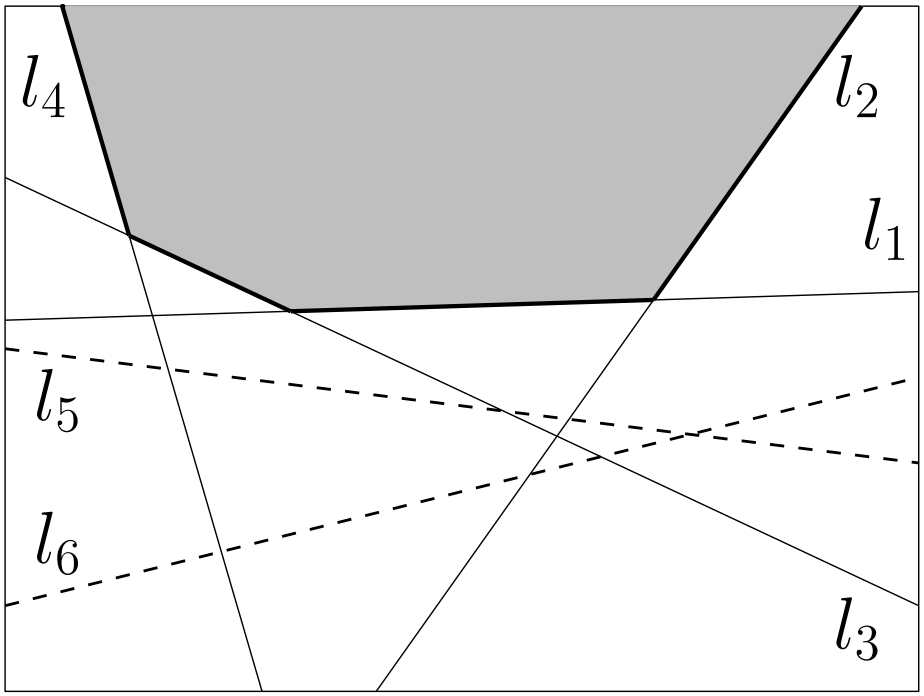


- the vertical distance between p and l is equal to the vertical distance between l^* and p^*
- the vertical distance between l and h is equal to length of (l^*h^*)

Duality in 2d: lower and upper envelopes

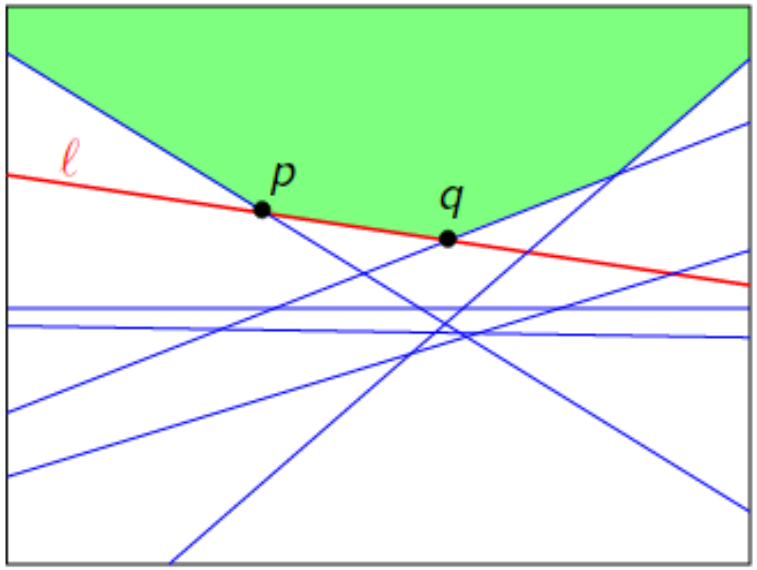
$\mathcal{L} = \{l_1, l_2, \ldots\}$



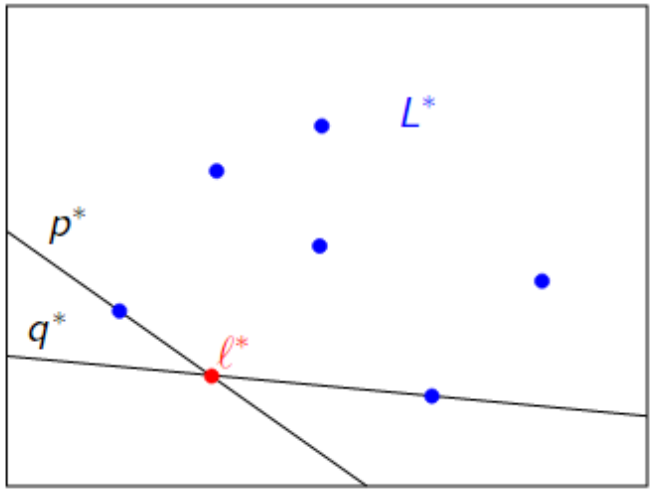


Duality in 2d: lower and upper envelopes

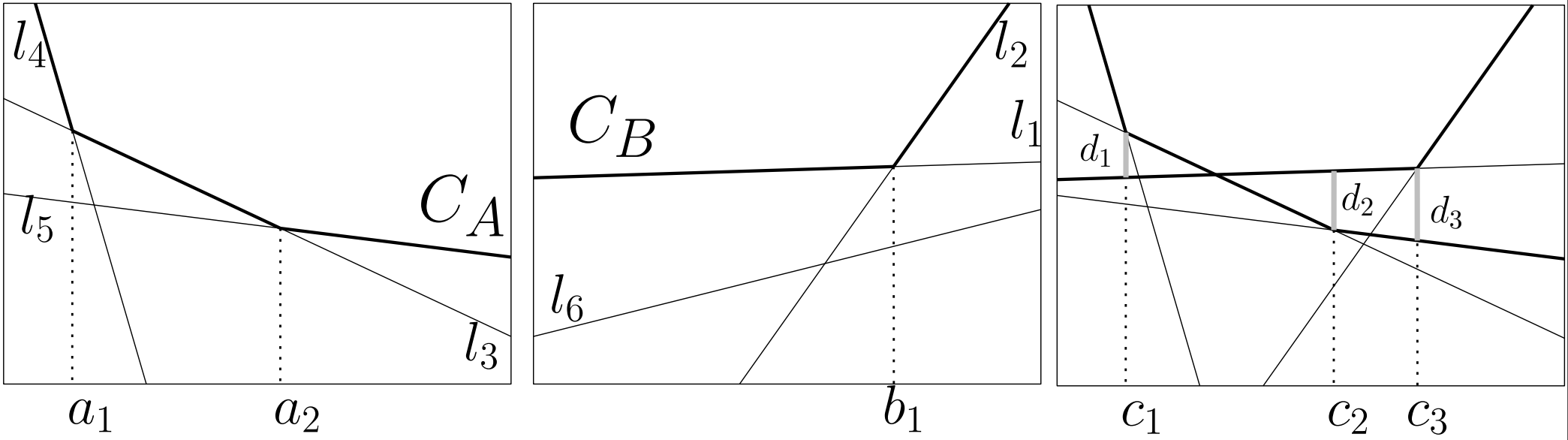
p and q are above all lines



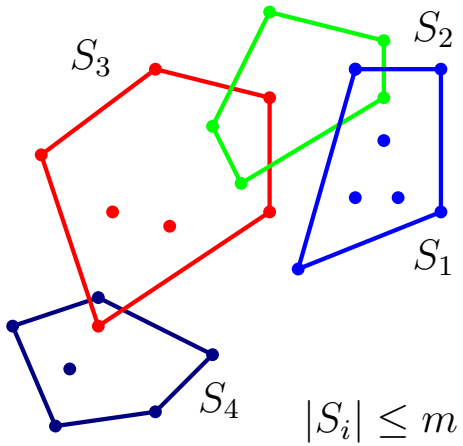
p and q are below all points



Divide and conquer: lower and upper envelopes

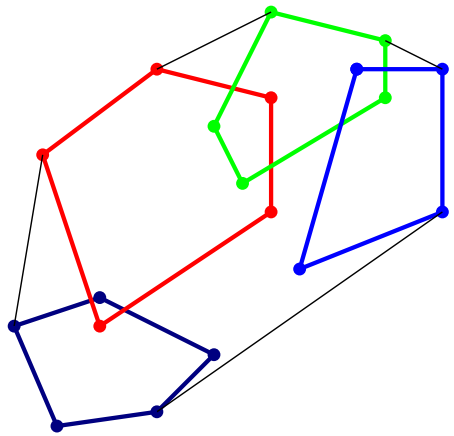


2D Convex hull: output sensitive algorithm



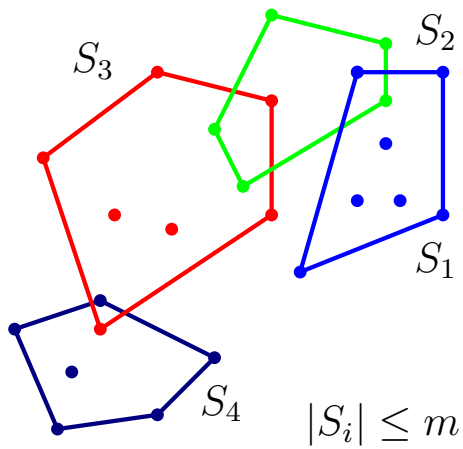
$|S_i| \leq m$

compute each $conv(S_i)$

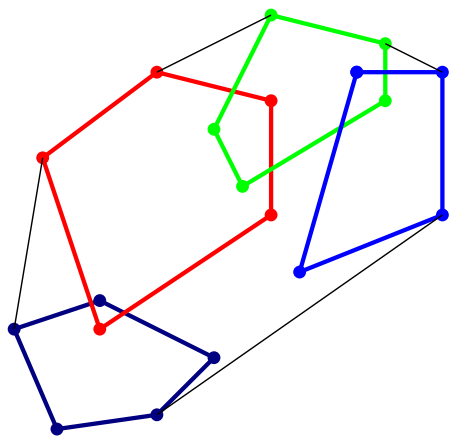


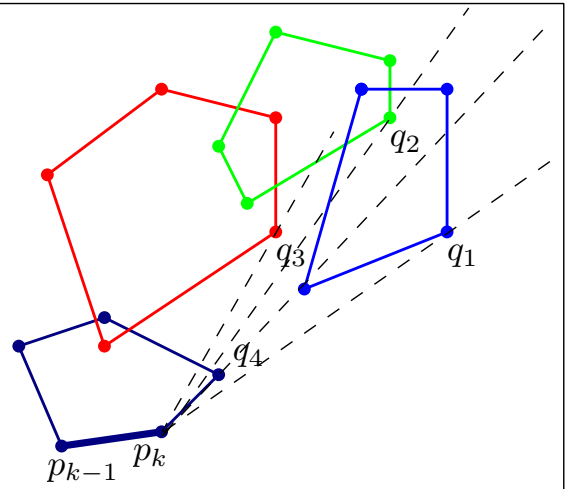
How can we compute common tangents?

2D Convex hull: output sensitive algorithm



compute each $conv(S_i)$





compute common tangents (via Jarvis march)

Let us assume we know the value of m

`PartialHull`$(\mathcal{S}, m)$

$p_0 := (-\infty, 0)$, $p_1 :=$ lowest point of $\mathcal{S}$

for $k = 1$ to m do

for $i = 1$ to r do

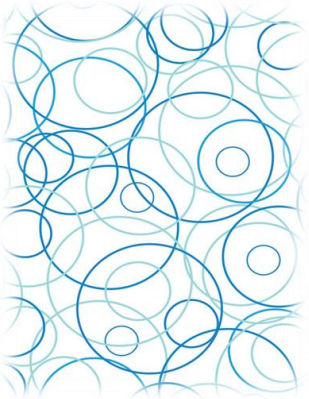
compute $q_i \in S_i$ maximizing the angle $(p_{k-1}p_kq_i)$ // tangent computation

let $q \in \{q_1, \ldots, q_r\}$the point maximizing angle $(p_{k-1}p_kq_i)$

$p_{k+1} := q$

if $p_{k+1} = p_1$ return $(p_1, \ldots, p_k)$

return `null`



Questions?