Introduction, Convex Hulls

CS 4235, Lecture 1
8 January 2004

Antoine Vigneron

`antoine@comp.nus.edu.sg`

National University of Singapore

Outline

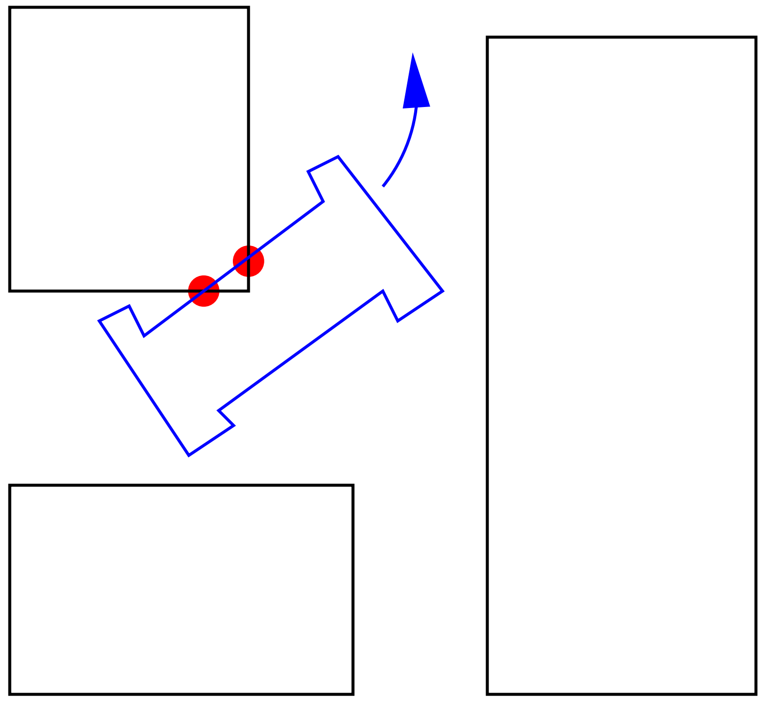
- Introduction
- Example: computing a convex hull
  - Geometry of the problem
  - A first algorithm
  - An optimal algorithm

Introduction

- Computational Geometry $=$
  Computer Science $\cap$ Geometry
- Algorithms, Data Structures, Geometry
- Applications:
  - Computer Graphics
  - Robotics
  - Spatial Databases, Geographic Information Systems
  - Computer Aided Design

Example 1: intersection detection

- Motion planning $\Rightarrow$ collision detection



Example 2: linear programming

- maximize the *objective function*

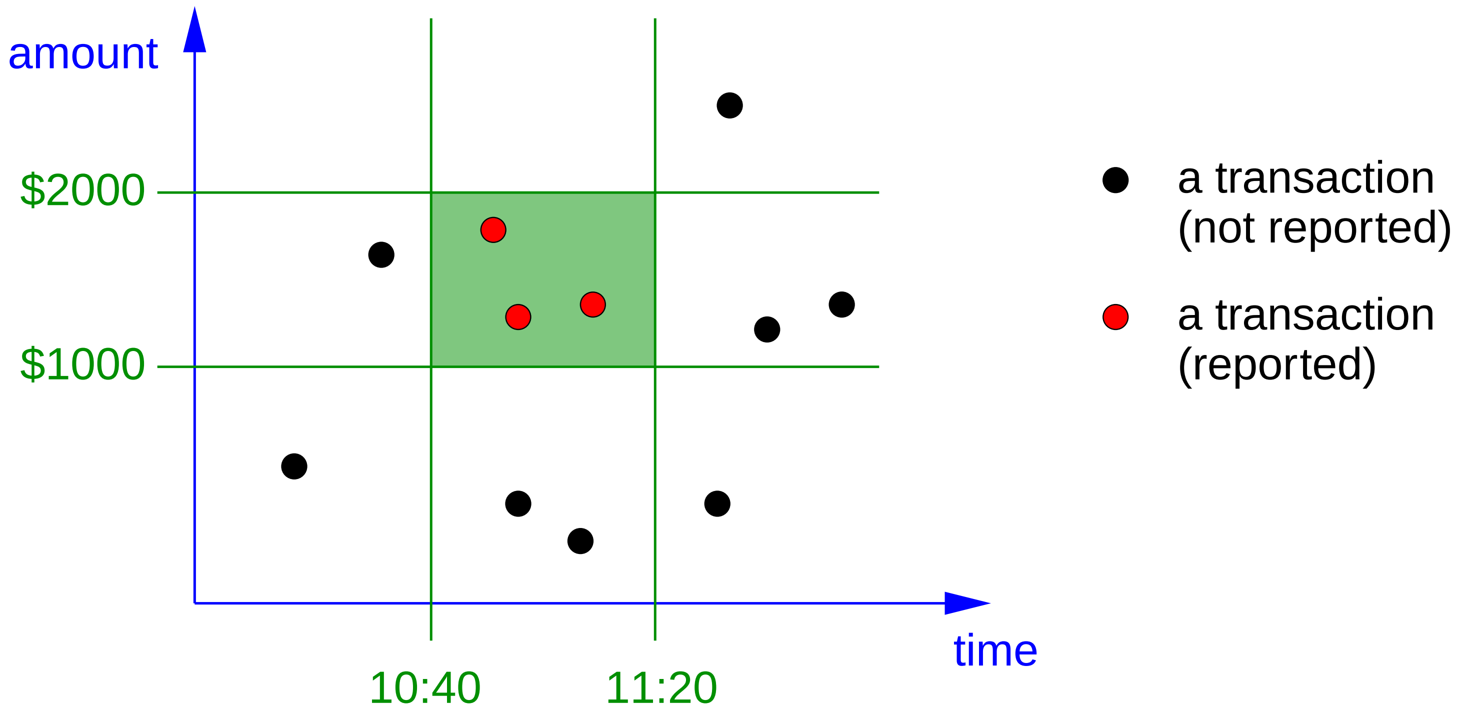
$$f(x_1, x_2 \ldots x_d) = c_1 x_1 + c_2 x_2 + \ldots + c_d x_d$$

- under the *constraints*

$$\begin{array}{ccccc} a_{1,1}x_1 & + \ldots + & a_{1,d}x_d & \leq & b_1 \\ a_{2,1}x_1 & + \ldots + & a_{2,d}x_d & \leq & b_2 \\ \vdots & & \vdots & & \vdots \\ a_{n,1}x_1 & + \ldots + & a_{n,d}x_d & \leq & b_n \end{array}$$

Example 3: range searching

- a database in a bank records transactions
- a query: find all the transactions such that
  - the amount is between \$ 1000 and \$ 2000
  - it happened between 10:40am and 11:20am
- geometric interpretation



CS 4235

- mainly algorithms in 2D
  $\Longleftarrow$ well established techniques: optimal, simple.
- no implementation issue

Students are expected to

- learn basic Computational Geometry techniques (lectures)
- apply them to design and analyze algorithms (tutorials)
- this includes writing simple proofs

Why should you take CS 4235?

- you think CS 3230 is useful/interesting
- you think geometry is useful/interesting
- applications
- work in software industry
  ($\Longleftarrow$ algorithm design)
- graduate studies

Organization

- Lecture/tutorial
  - 2 hours tutorial time slots
  - two midterms (at tutorial time)
  - in fact tutorials ≤ 90 minutes
- all exams are open book
- grading
  - final: 40%
  - midterm 1: 20%
  - midterm 2: 30%
  - participation: 10%

Schedule

- tutorials: Tuesday 12-2pm
  - two midterms at tutorial time slots
  - midterm 1: week 6
  - midterm 2: week 11

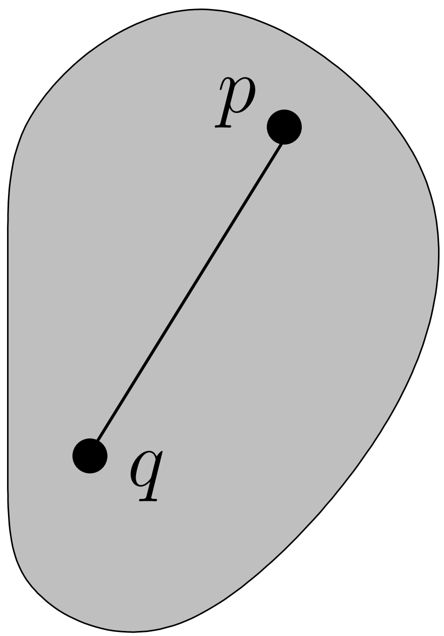
Resources

- IVLE website
- slides
- lecture notes by Dave Mount
- to learn more
  - textbook by de Berg et al.
  - discrete geometry book by J. Matousek
  - randomized algorithms book by K. Mulmuley

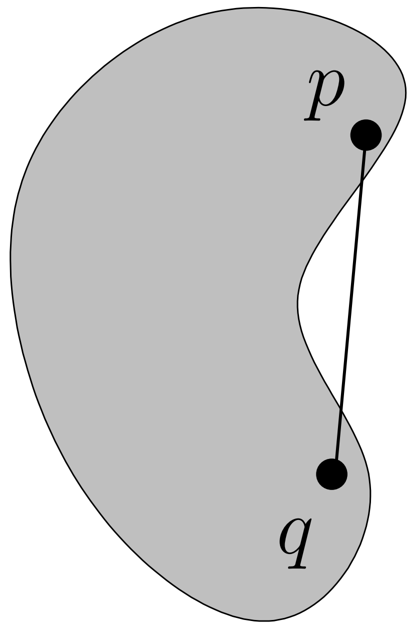
Convex Hulls

Convexity

A set $\mathcal{C} \subset \mathbb{R}^d$ is *convex* iff $\forall (p, q) \in \mathcal{C}^2$ the line segment $\overline{pq}$ is contained in $\mathcal{C}$.



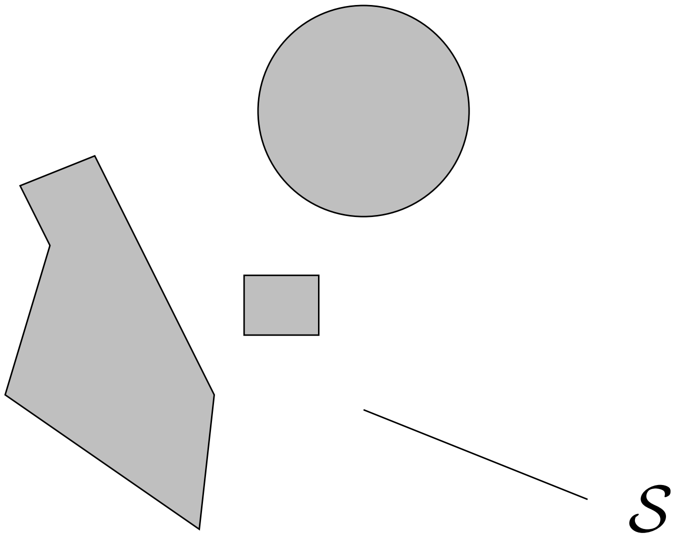
Convex



Non convex

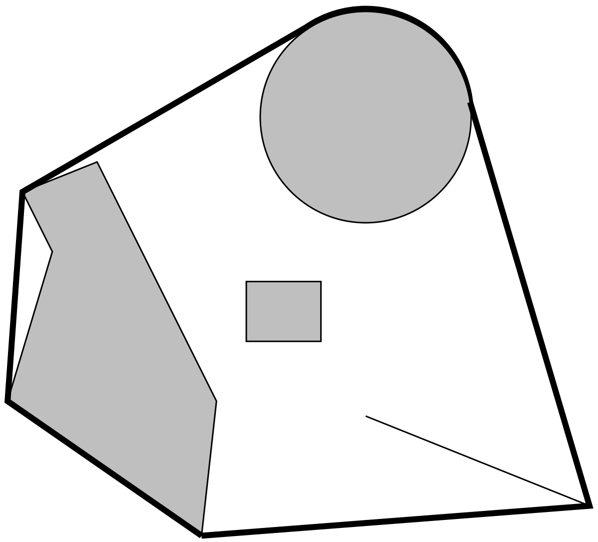
Convex hull

- The intersection of an arbitrary family of convex sets is convex (proof?).
- Let $\mathcal{S} \subset \mathbb{R}^d$. Its *convex hull* $\mathcal{CH}(\mathcal{S})$ is the intersection of all the convex sets that contain $\mathcal{S}$.
- $\mathcal{CH}(\mathcal{S})$ is the smallest convex set containing $\mathcal{S}$.



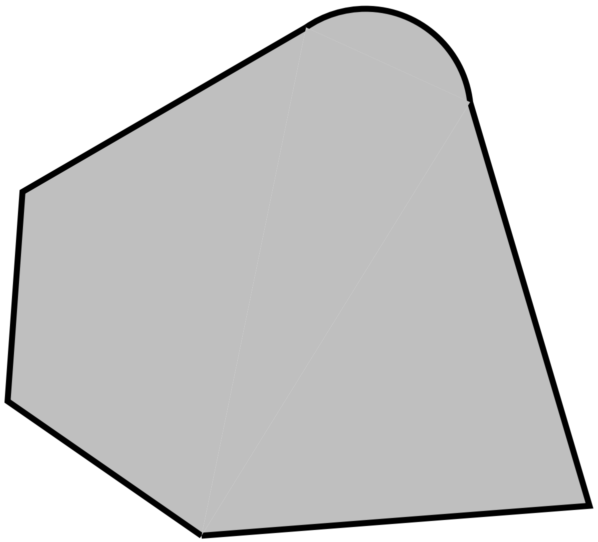
Convex hull

- The intersection of an arbitrary family of convex sets is convex (proof?).
- Let $\mathcal{S} \subset \mathbb{R}^d$. Its *convex hull* $\mathcal{CH}(\mathcal{S})$ is the intersection of all the convex sets that contain $\mathcal{S}$.
- $\mathcal{CH}(\mathcal{S})$ is the smallest convex set containing $\mathcal{S}$



Convex hull

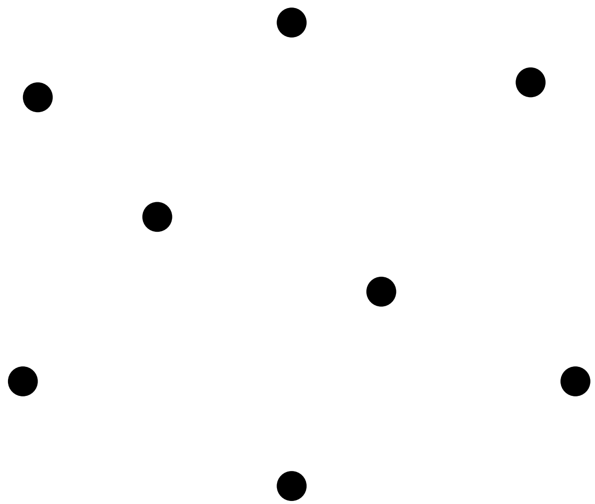
- The intersection of an arbitrary family of convex sets is convex (proof?).
- Let $\mathcal{S} \subset \mathbb{R}^d$. Its *convex hull* $\mathcal{CH}(\mathcal{S})$ is the intersection of all the convex sets that contain $\mathcal{S}$.
- $\mathcal{CH}(\mathcal{S})$ is the smallest convex set containing $\mathcal{S}$.



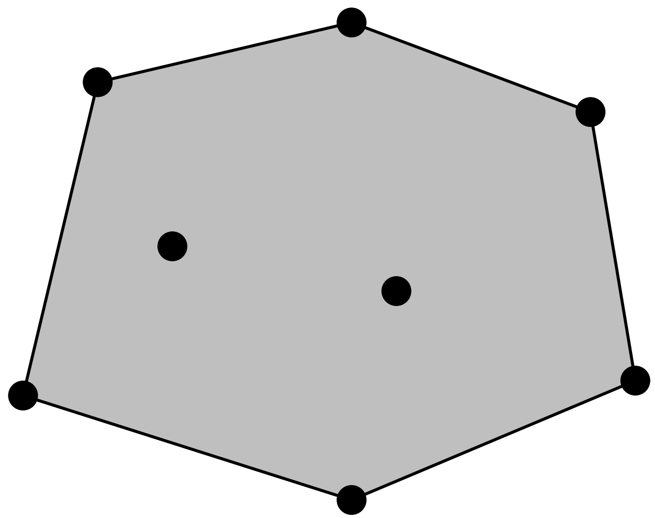
$\mathcal{CH}(\mathcal{S})$

Points in the plane

- Let $P = \{p_1, p_2, \ldots p_n\} \subset \mathbb{R}^2$.
- $\mathcal{CH}(P)$ is a convex polygon.



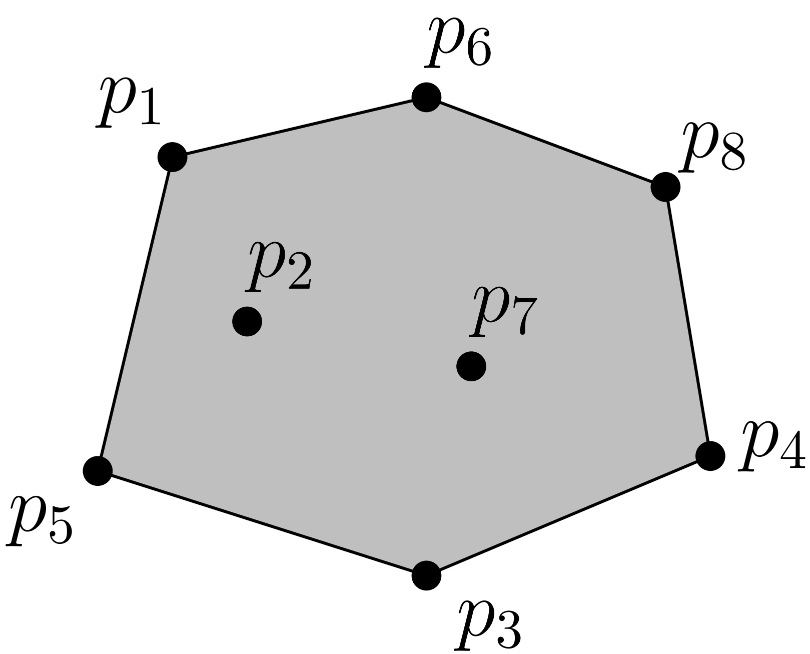
P



$\mathcal{CH}(P)$

Computing a convex hull

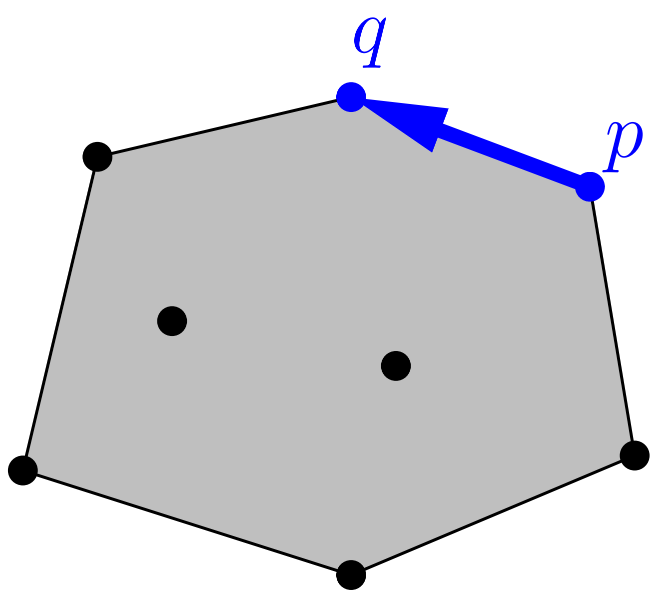
- Input: the set $P = \{p_1, p_2, \ldots p_n\} \subset \mathbb{R}^2$
- Output: a sequence $\mathcal{L} = (c_1, c_2, \ldots c_h)$ of vertices of $\mathcal{CH}(P)$ in counterclockwise order
- Example: $\mathcal{L} = (p_3, p_4, p_8, p_6, p_1, p_5)$



$\mathcal{CH}(P)$

Characterization

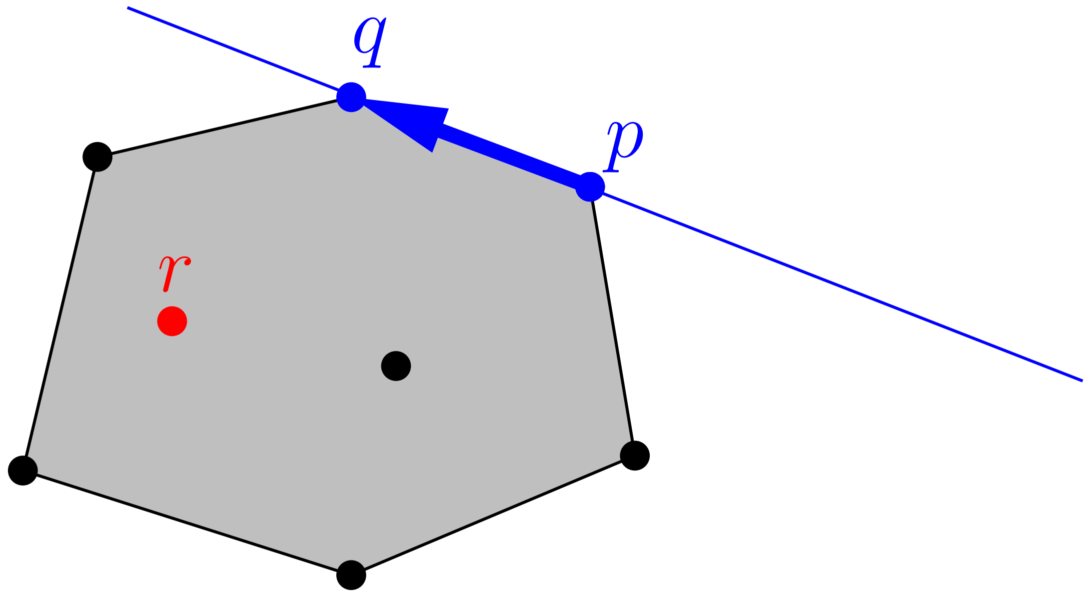
The directed edge (p, q) is an edge of $\mathcal{CH}(P)$ iff



$\mathcal{CH}(P)$

Characterization

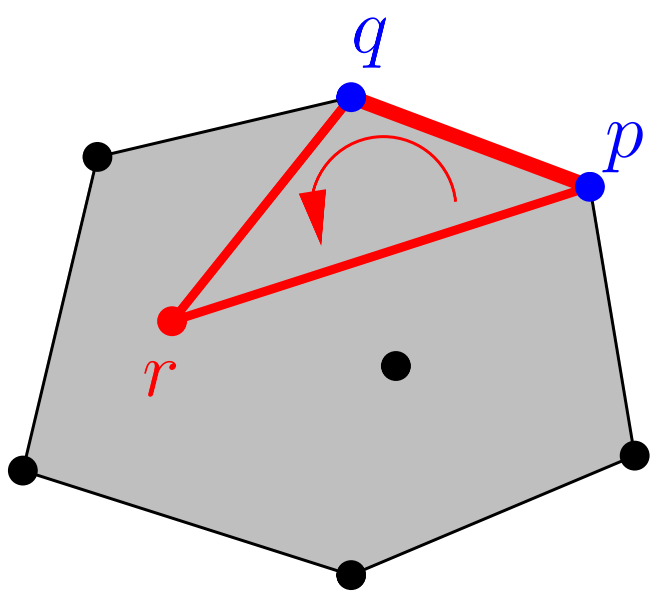
The directed edge (p, q) is an edge of $\mathcal{CH}(P)$ iff



all $r \in P \setminus \{p, q\}$ lies to the left of line pq (oriented by $\overrightarrow{pq}$).

Characterization

The directed edge (p, q) is an edge of $\mathcal{CH}(P)$ iff



$\forall r \in P \setminus \{p, q\}$ the triangle (p, q, r) is oriented counterclockwise.

Orientation test

- We denote

$$CCW(p,q,r) = \begin{vmatrix} x_p & x_q & x_r \\ y_p & y_q & y_r \\ 1 & 1 & 1 \end{vmatrix}$$

$$= (x_q - x_p)(y_r - y_p) - (x_r - x_p)(y_q - y_p)$$

- Triangle (p,q,r) is counterclockwise iff $CCW(p,q,r) > 0$.
- How fast can we perform this test?
  - 2 multiplications and 5 subtractions
  - takes $O(1)$ time

Precision issue

- previous slide seems to imply that we use floating point numbers
- so the result is not exact
- how to find the sign of $CCW(p, q, r)$ exactly?
- assume input coordinates are 32 bits integers
- then $(x_p - x_q)$ has 33 bits
- $(x_q - x_p)(y_r - y_p)$ has 66 bits
- $CCW(p, q, r)$ has 67 bits
- so it can be computed exactly in $O(67) = O(1)$ time

First algorithm

Algorithm *SlowConvexHull*(P)
Input: a set P of points in $\mathbb{R}^2$
Output: $\mathcal{CH}(P)$

1. $E \leftarrow P^2$
2. **for** all $(p, q, r) \in P^3$ such that $r \notin \{p, q\}$
3. **if** $CCW(p, q, r) \leq 0$
4. **then** remove (p, q) from E
5. Write the remaining edges of E into $\mathcal{L}$ in counterclockwise order
6. Return $\mathcal{L}$

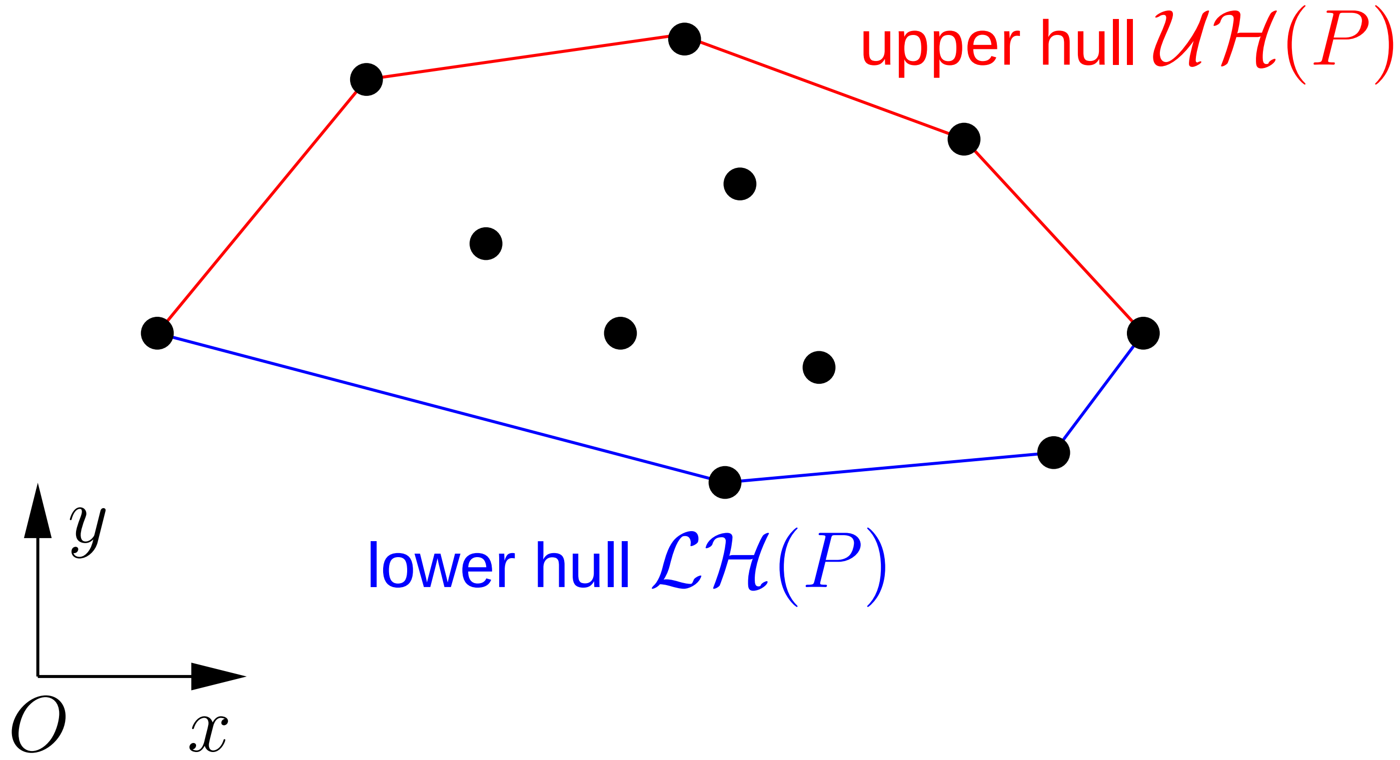
Comments

- Line 1: find all directed edges between two points of P
  $\longrightarrow O(n^2)$ time
- Lines 2-4: discard the edges that are not in the convex hull
  $\longrightarrow O(n^3)$ time
- line 5: how fast can you do it, and how?
  $\longrightarrow$ easy to do in $O(n^3)$ time

Conclusion: this algorithm runs in $O(n^3)$ time

Actually, $\Theta(n^3)$ time.

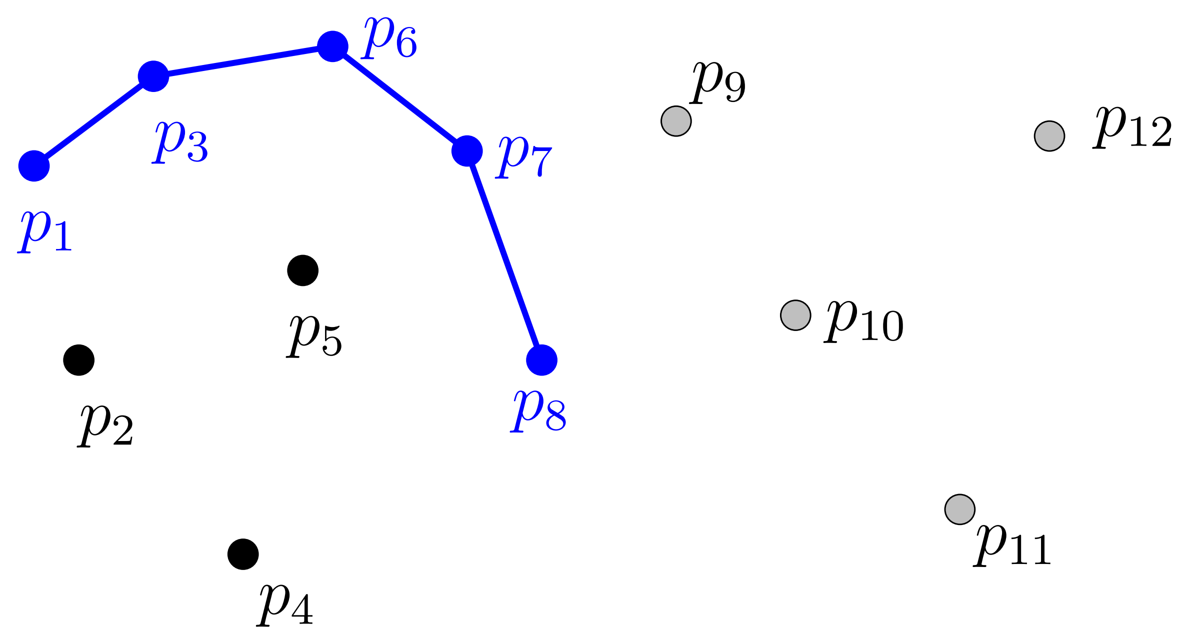
Upper hull and lower hull



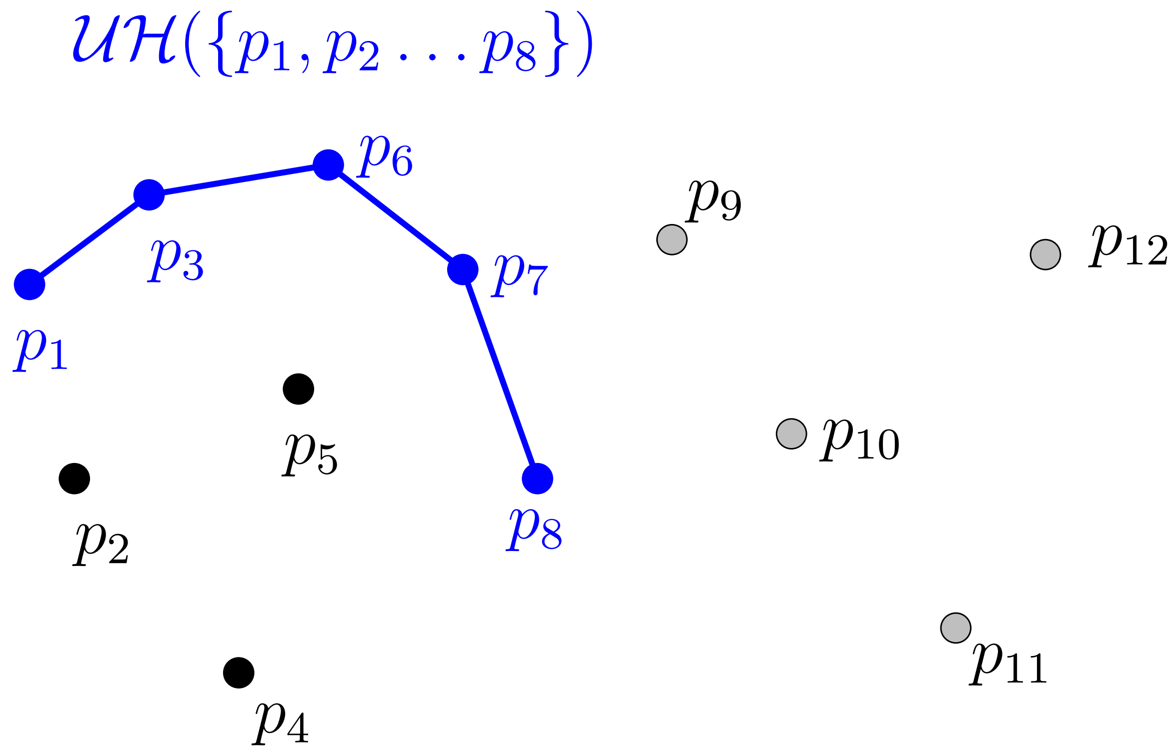
Computing $\mathcal{UH}(P)$

- Sort P according to x–coordinates
- Compute $\mathcal{UH}(P)$ from left to right

$\mathcal{UH}(\{p_1, p_2 \ldots p_8\})$

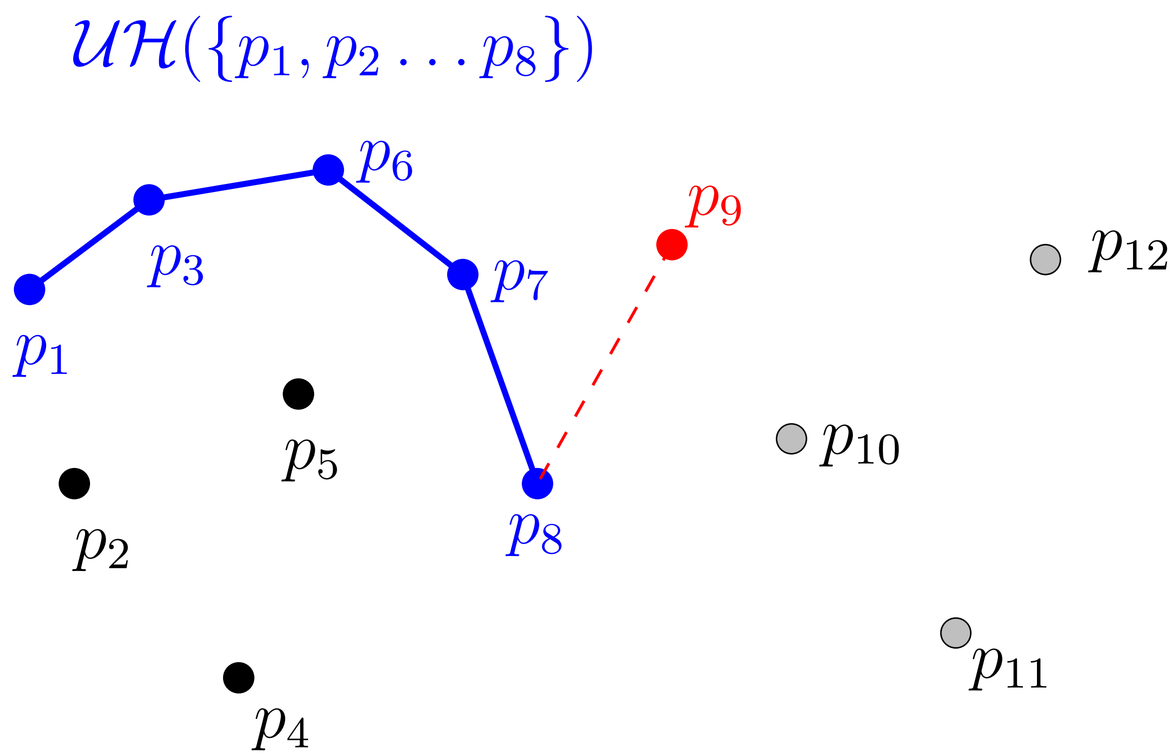


Inserting p_9



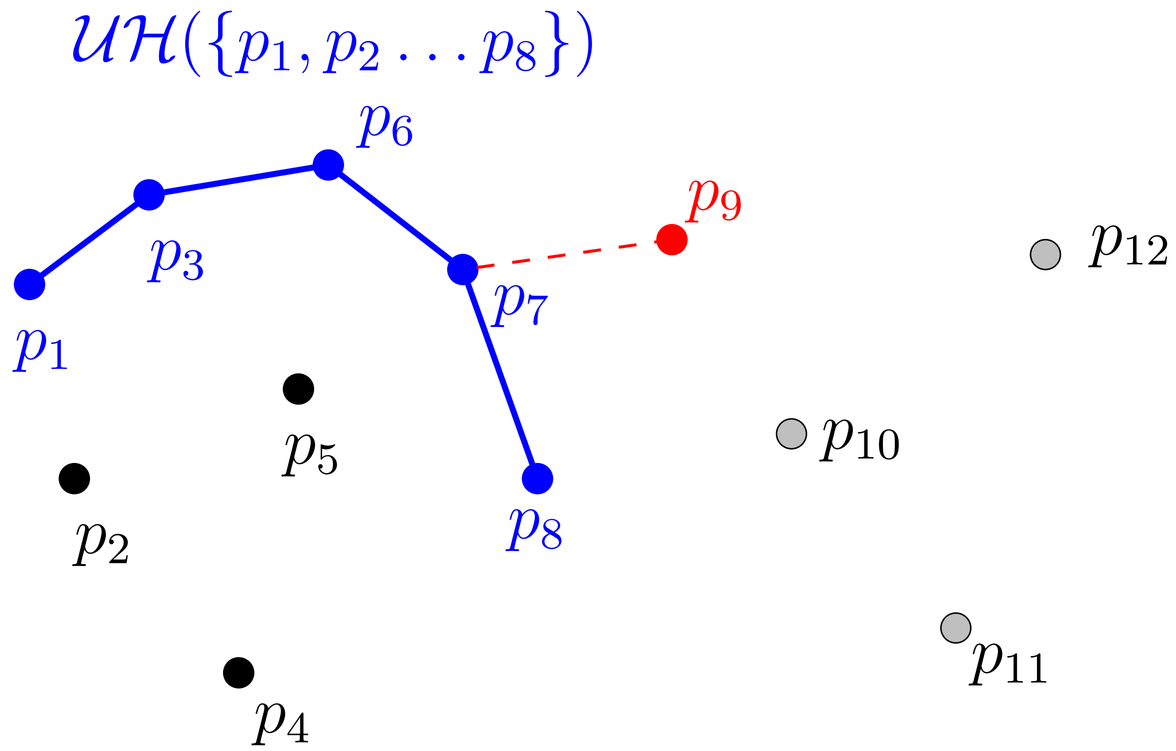
The upper hull of $p_1, p_2 \ldots p_8$ has just been computed. We now insert p_9.

Inserting p_9



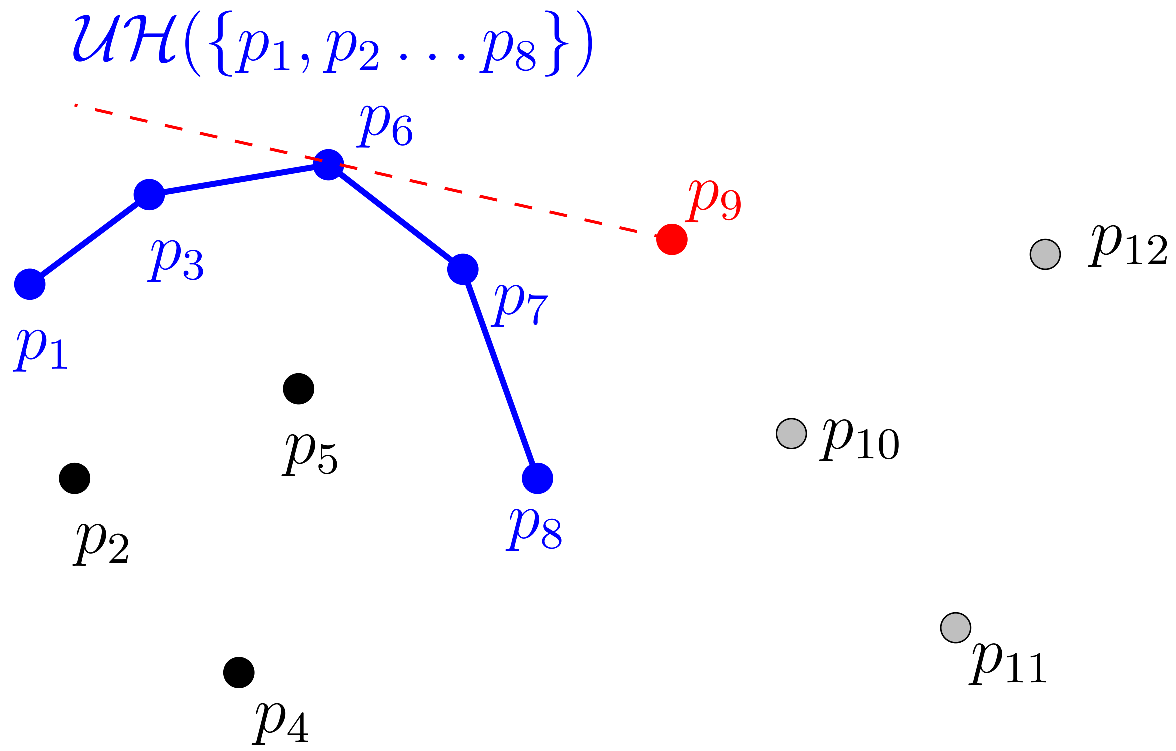
p_8 is not on the upper hull

Inserting p_9



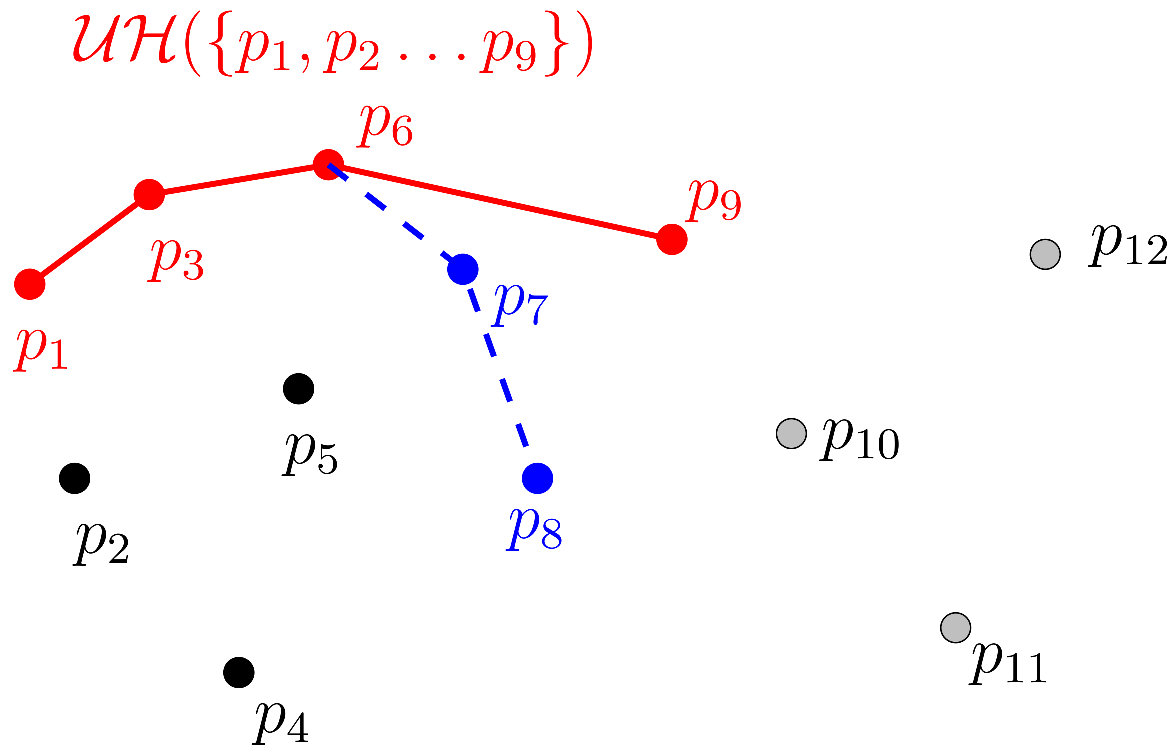
move leftward along $\mathcal{UH}(\{p_1, p_2, \ldots p_8\})$

Inserting p_9



$p_6 p_9$ is tangent to $\mathcal{UH}(\{p_1, p_2, \dots p_8\})$

Inserting p_9



Remove the left chain, and connect the right chain to p_9.

Algorithm

- sort P according to x coordinates
- initialization: the upper hull is (p_1, p_2)
- for $i = 3$ to n, insert p_i, update the upper hull
- similar algorithm to compute the lower hull
- form the boundary of the convex hull as the union of the upper hull and the lower hull

Analysis

- initial sorting takes $O(n \log n)$ time
- inserting p_i takes $\theta(n)$ time
  $\Longrightarrow$ computing $\mathcal{UH}(P)$ takes $n.O(n) = O(n^2)$ time
- in fact, this algorithm runs in $O(n)$ time
  $\Longleftarrow$ even though inserting a particular point may take linear time, the overall complexity is still linear.

Amortized analysis

- n_i denotes the number of points discarded from the upper hull when we insert p_i
- inserting p_i takes time $O(n_i)$
- note that $n_1 + n_2 \ldots + n_n = n - h < n$ (h is the number of points on $\mathcal{CH}(P)$).

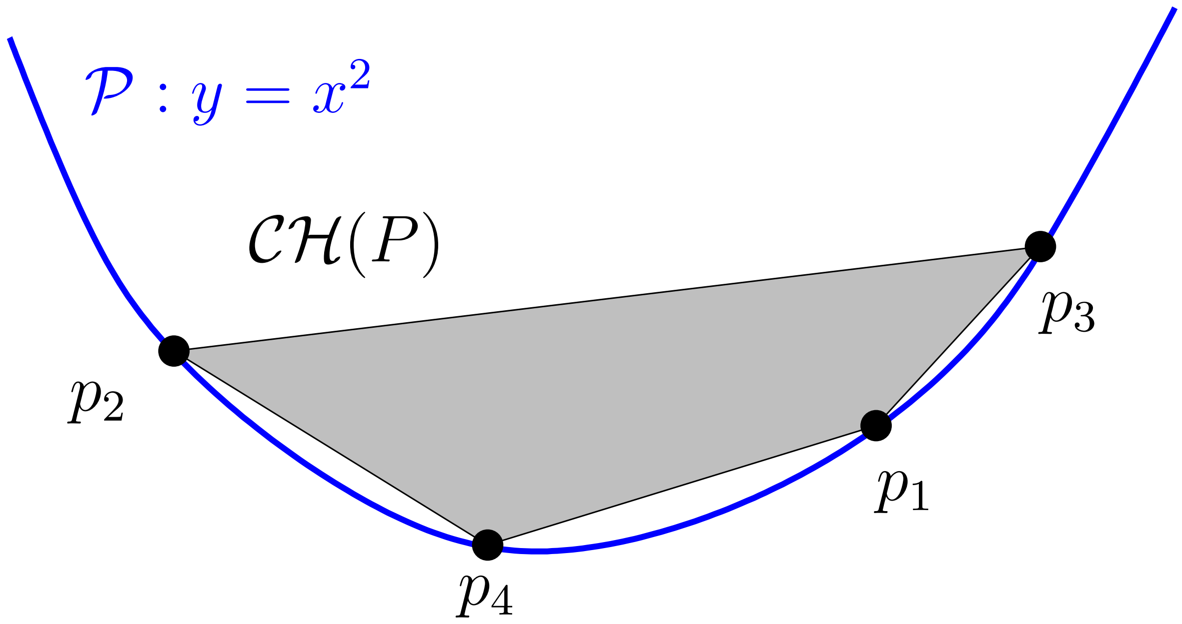
Running time =
$O(n \log n)$ (initial sorting)
+ $O(n)$ (inserting $p_3, p_4 \ldots p_n$ in upper hull)
+ $O(n)$ (lower hull)
+ $O(n)$ (forming the convex hull)

=$O(n \log n)$

Lower bound

Our algorithm is optimal (within a constant factor), here is a proof by reduction from sorting.

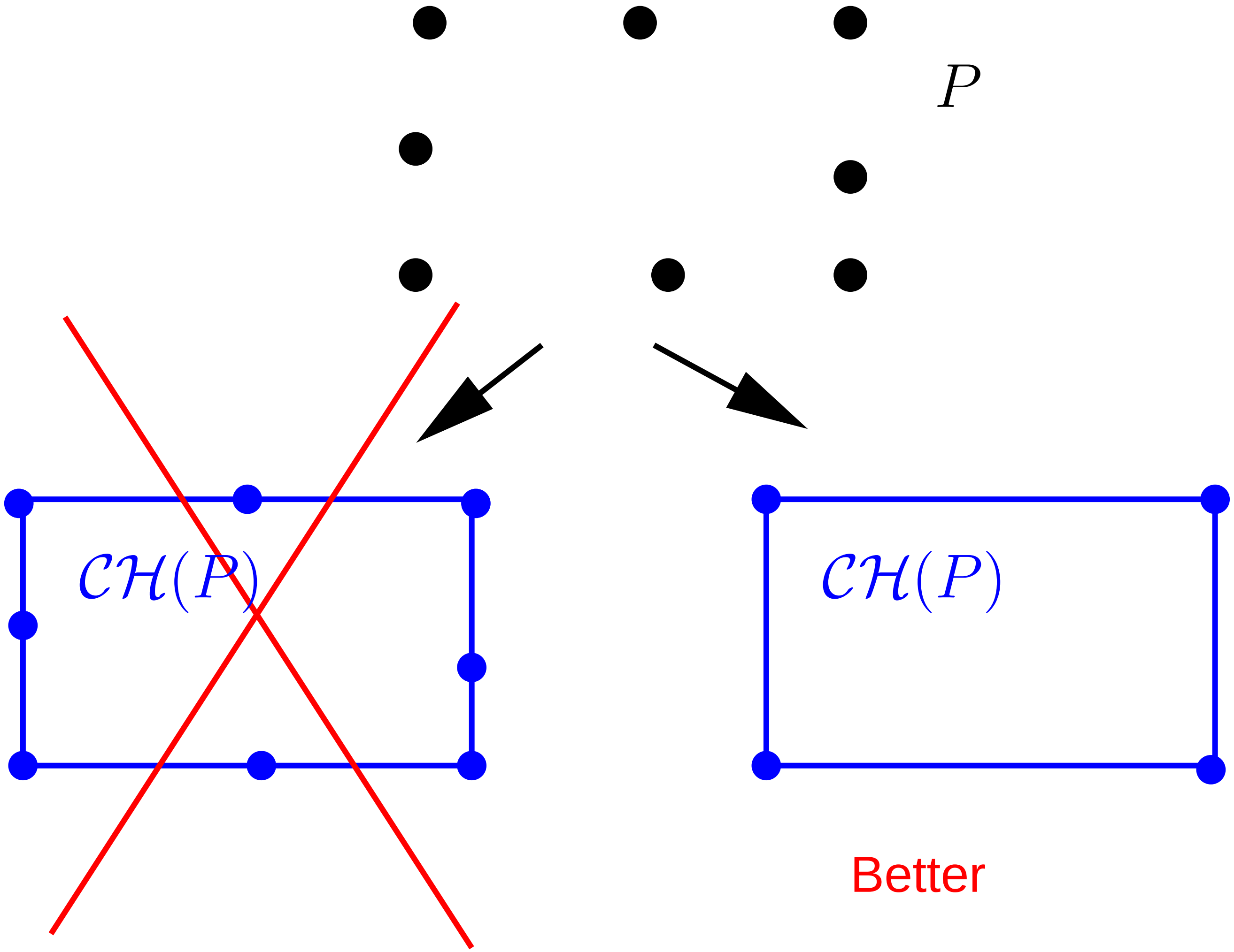
- let $N = (x_1, x_2, \ldots x_n) \subset \mathbb{R}$
- for all i let $p_i = (x_i, x_i^2)$
- compute $\mathcal{CH}(P)$



Lower bound

- find the leftmost point p in $\mathcal{CH}(P)$
- starting from p, walk from left to right along $\mathcal{LH}(P)$
- the x coordinates of these points give N in sorted order
- overall, it takes time $O(n)$ + time for computing $\mathcal{CH}(P)$
- lower bound for sorting n real numbers in general: $\Omega(n \log n)$ time
  $\Longrightarrow$ computing a convex hull takes $\Omega(n \log n)$ time

Degeneracies



Solution

- first algorithm OK
- upper hull: if several points have same x–coordinate, keep the highest

Algorithm design method:

- first assume *general position*: (here, no two points have same x–coordinate)
  $\Longleftarrow$ focus on a simpler, but very general instance
- if necessary, handle degeneracies by
  - ad–hoc methods
  - general (mainly theoretical) methods

General position assumptions

Typically

- no two points have same x–coordinates
- no three points are collinear, that is,
  $\forall(p, q, r) \in P, CCW(p, q, r) \neq 0$
- no four points are cocircular
- all of the above

Reasons

- if the points of P are drawn uniformly at random from a square, it happens with probability 1
- for any degenerate P, there is a set P' in general position that is arbitrarily close to P.