CSE520: Computational Geometry
Lecture 2
Convex Hulls

Antoine Vigneron

Ulsan National Institute of Science and Technology

June 15, 2020

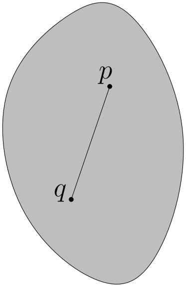
Introduction

- Reference: textbook Lecture 1.

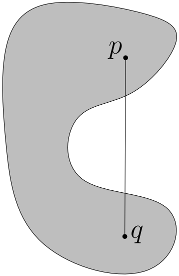
Convexity

Definition

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

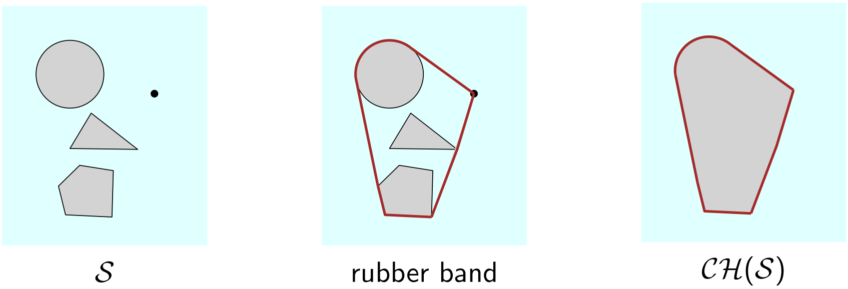
Convex Hull

The intersection of an arbitrary family of convex sets is convex (proof?).

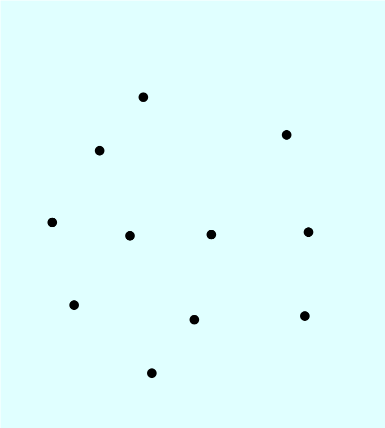
Definition (convex hull)

The *convex hull* of a set $\mathcal{S} \subset \mathbb{R}^d$ is the intersection of all the convex sets that contain $\mathcal{S}$.

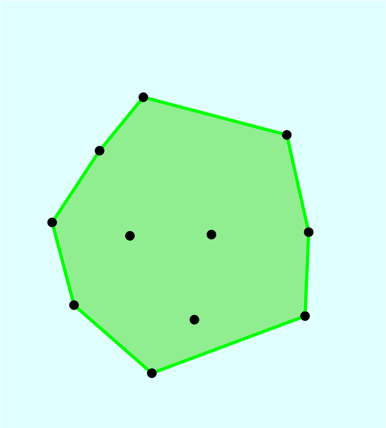
$\mathcal{CH}(\mathcal{S})$ is the smallest convex set containing $\mathcal{S}$.



Points in the Plane



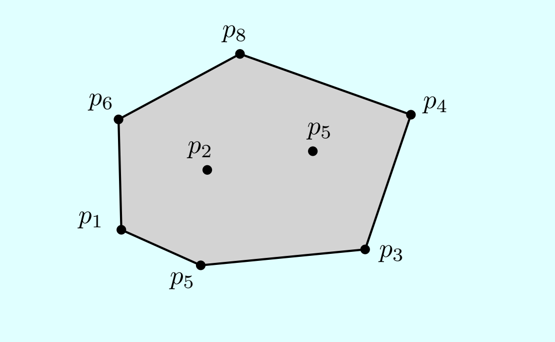
$P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$.



$\mathcal{CH}(P)$ is a convex polygon.

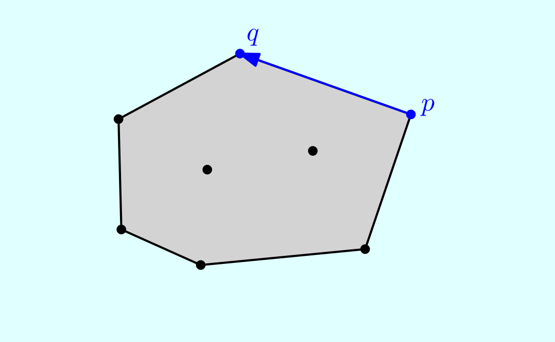
Computing a Convex Hull

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



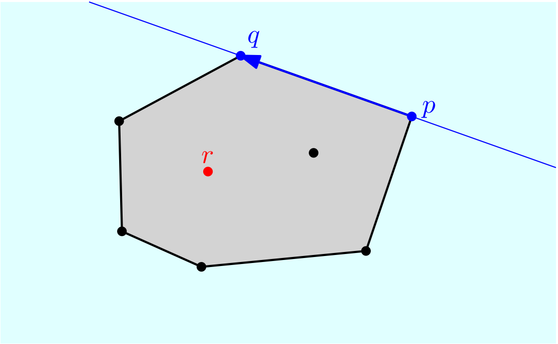
Characterization

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



Characterization

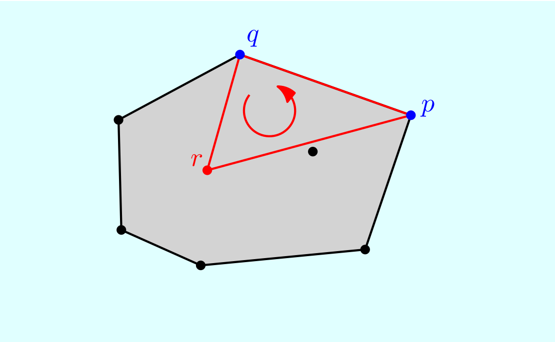
The directed edge (p, q) is an edge of $\mathcal{CH}(P)$ iff



each $r \in P \setminus \{p, q\}$ 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

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

- 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

First Algorithm

Naive convex hull 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 $(p, q, r) \in P^3$
3.   **if** $CCW(p, q, r) < 0$
4.   **then** remove (p, q) from E
5. Write the remaining edges of E into $\mathcal{L}$ in counterclockwise order
6. **return** $\mathcal{L}$

- In the algorithm above, we assume that no 3 points are collinear.
- How to fix it?
  - In line 3, the condition becomes:

$$CCW(p, q, r) \leqslant 0 \text{ and } r \notin \overline{pq}.$$

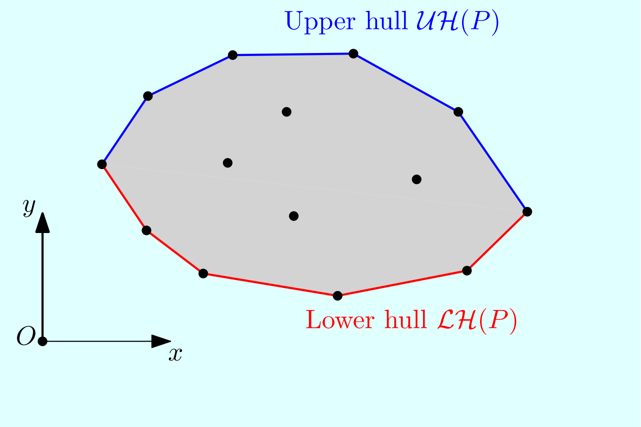
Analysis

- 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 $O(n^2)$ time.

Proposition

The naive algorithm runs in $\Theta(n^3)$ time.

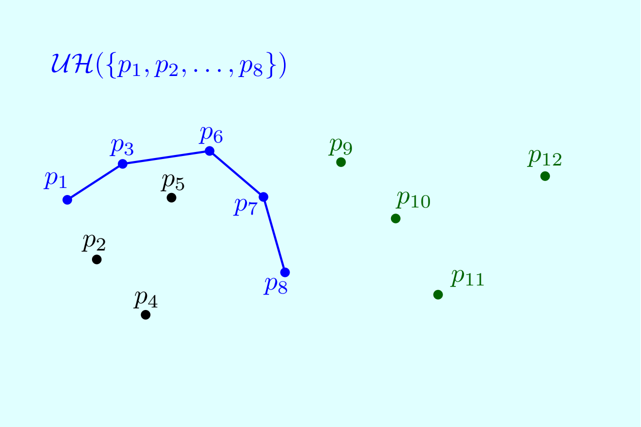
Upper Hull and Lower Hull



The upper hull is the part of the boundary of the convex hull that is above it.

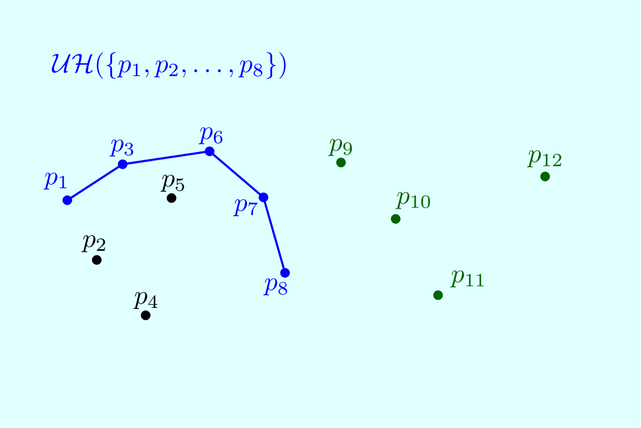
Computing $\mathcal{UH}(P)$

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



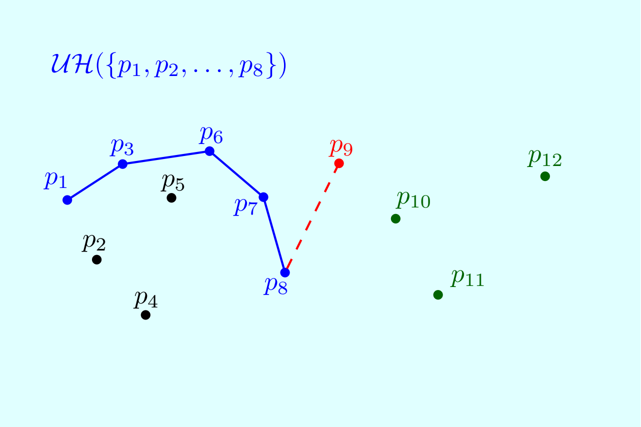
Computing $\mathcal{UH}(P)$: Inserting p_9

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



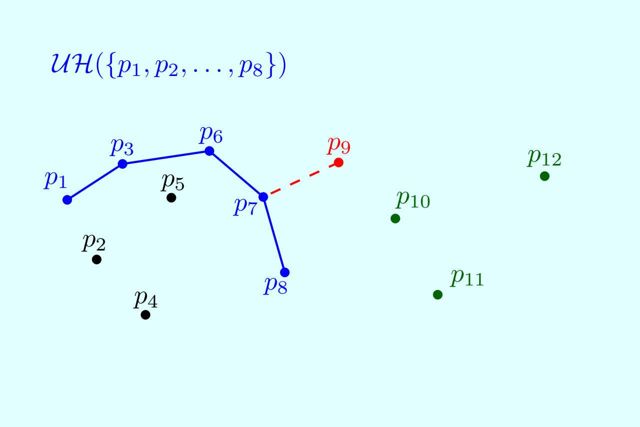
We will now insert p_9.

Computing $\mathcal{UH}(P)$: Inserting p_9



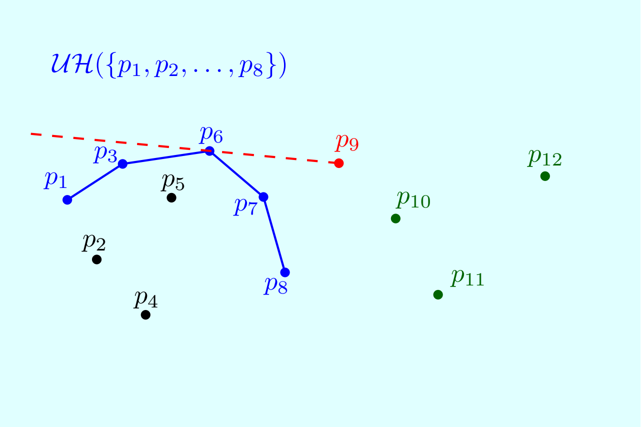
p_8 is not on the upper hull.

Computing $\mathcal{UH}(P)$: Inserting p_9



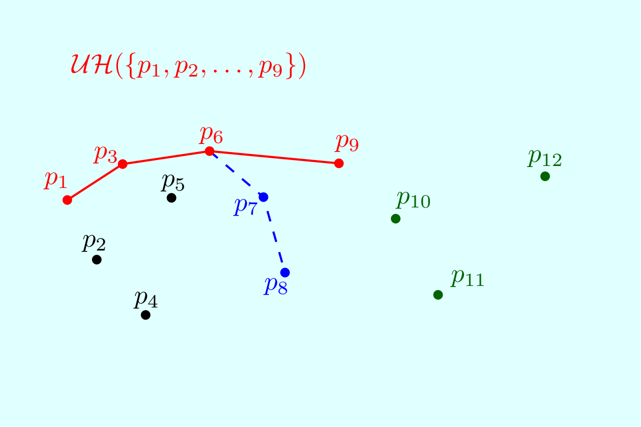
Move leftward along $\mathcal{UH}(\{p_1, p_2, \ldots p_8\})$.

Computing $\mathcal{UH}(P)$: Inserting p_9



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

Computing $\mathcal{UH}(P)$: Inserting p_9



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

Pseudocode

Efficient upper hull algorithm

Algorithm *FastUpperHull*(P)
Input: A set P of at least two points in $\mathbb{R}^2$
Output: $\mathcal{UH}(P)$

1. Sort P by increasing x-coordinates
2. $U[\cdot] \leftarrow [p_1, p_2]$, $k \leftarrow 2$
3. **for** $i \leftarrow 3, n$
4.   **while** $k > 1$ and $CCW(U[k-1], U[k], p_i) \geqslant 0$
5.   **do** $k \leftarrow k - 1$
6. $k \leftarrow k + 1$, $U[k] \leftarrow p_i$
7. **return** $U[1 \dots k]$

- 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 at line 1 takes $O(n \log n)$ time.
- Inserting p_i at lines 4–6 takes $\Theta(n)$ time.
  - Computing $\mathcal{UH}(P)$ takes $n.O(n) = O(n^2)$ time.
- But in fact, this algorithm runs in $O(n)$ time.
  - Even though inserting a particular point may take linear time, the overall complexity is still linear.

Amortized Analysis

- Let m_i denote the number of points discarded from the upper hull when we insert p_i.
- Inserting p_i takes time $O(m_i + 1)$.
- Observe that $m_3 + m_4 \cdots + m_n = n - h < n$ (h is the number of points on $\mathcal{UH}(P)$).

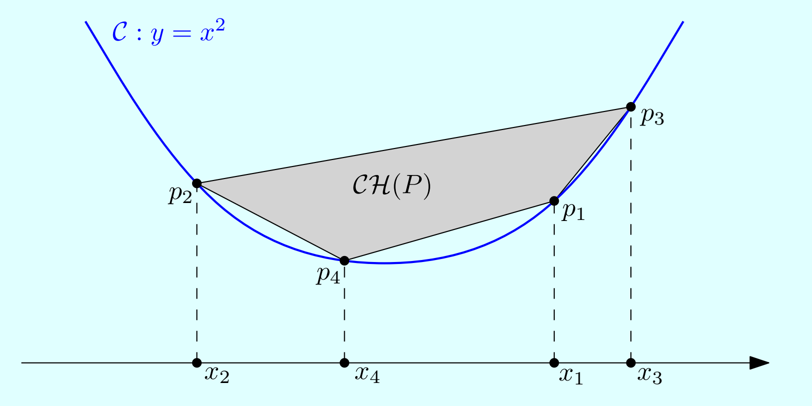
Running time

$O(n \log n)$ (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)
Total: $O(n \log n)$

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

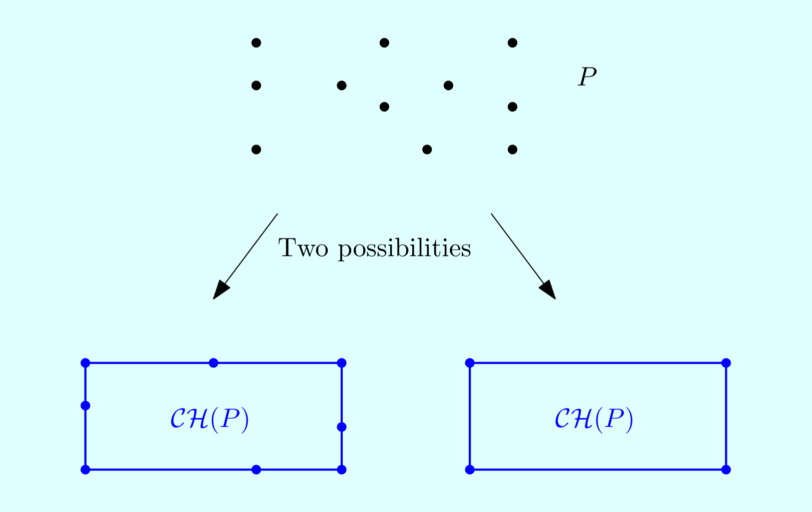
From previous slide, a convex hull algorithm allows us to sort a set of reals as follows:

- 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

- Computing a convex hull takes $\Omega(n \log n)$ time.

Degeneracy



Solution

- First algorithm OK.
- Upper hull: If several points have same x-coordinate, keep the highest.

Algorithm design approach:

- First assume *general position*:
  - Here, it means that no two points have same x-coordinate.
  - Purpose: Focus on a simpler, but still very general instance.
- If necessary, handle degeneracy by:
  - Ad-hoc methods,
  - or 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.