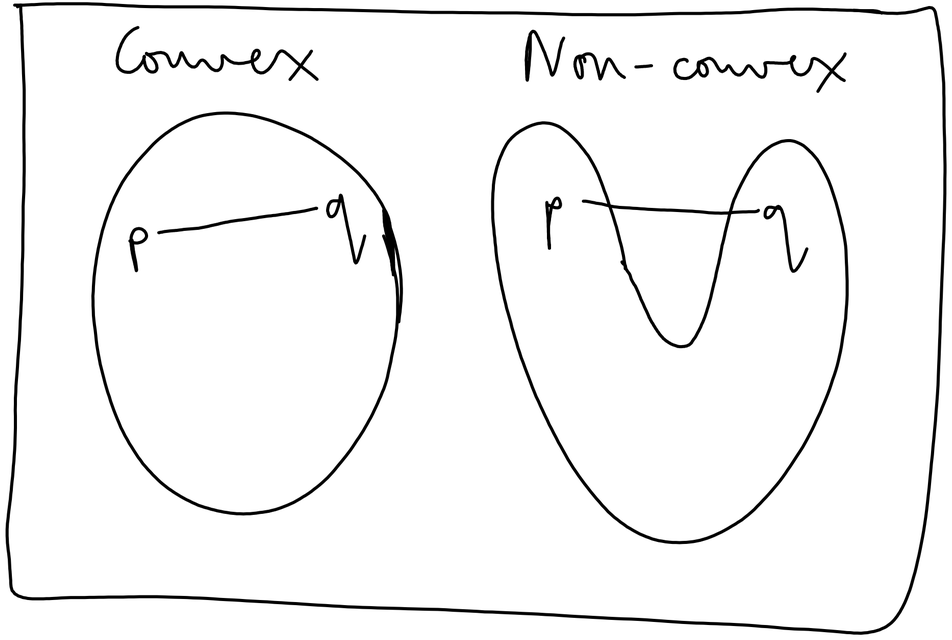
John Bourke bourkej@math.muni.cz

Geometric algorithms

- Each week : 1 algorithm
- Interconnected (eg. wk 2 - sweepline algorithm)
- E-learning course
- Uploading notes weekly to IS
- Book: Computational geometry by deBerg...
- Exam at end

Convex hull in the plane

$K \subseteq \mathbb{R}^2$ (plane) is convex if for all $p, q \in K$, the line segment $\overline{pq}$ is also in K.



$\overline{pq}$ consists of points of form

$$r = p + \lambda(q-p) \text{ for } \lambda \in [0,1].$$

$\lambda = 0 \mapsto r = p$, $\lambda = 1 \mapsto r = q$

Equivalently,

$$r = (1-\lambda)p + \lambda q \text{ or}$$

$$r = \lambda p + (1-\lambda)q$$

Convex hull

$CH(P) :=$ smallest convex set containing P

$$= \bigcap_{\substack{K \text{ convex} \\ P \subseteq K}} K$$

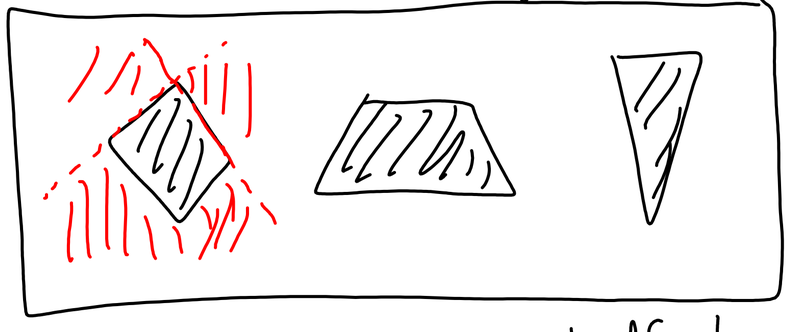
- Not computationally useful - infinitely many convex sets containing P.

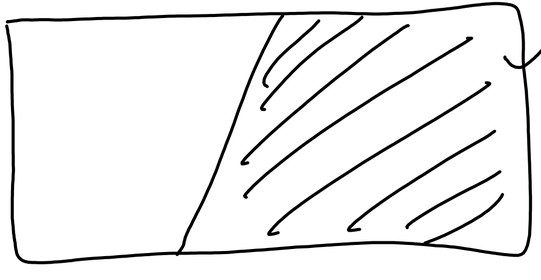
Goal: compute $CH(P)$ for P

a finite set of points.

P finite $\Rightarrow$ $CH(P)$ convex polygon :
bounded by finitely many straight-line segments.

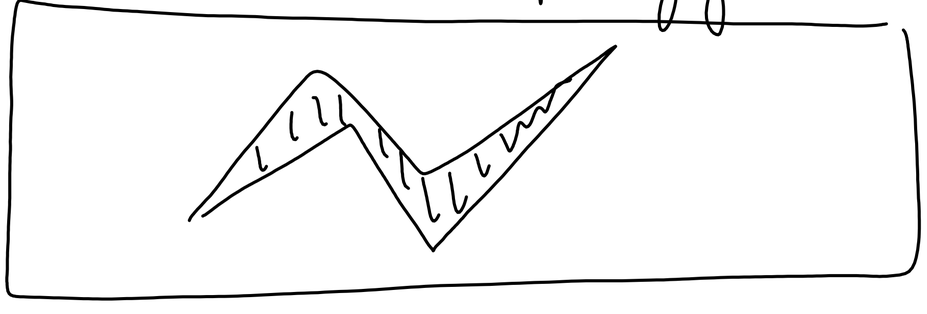
Convex polygons





half-plane : points on one side of a straight-line in plane

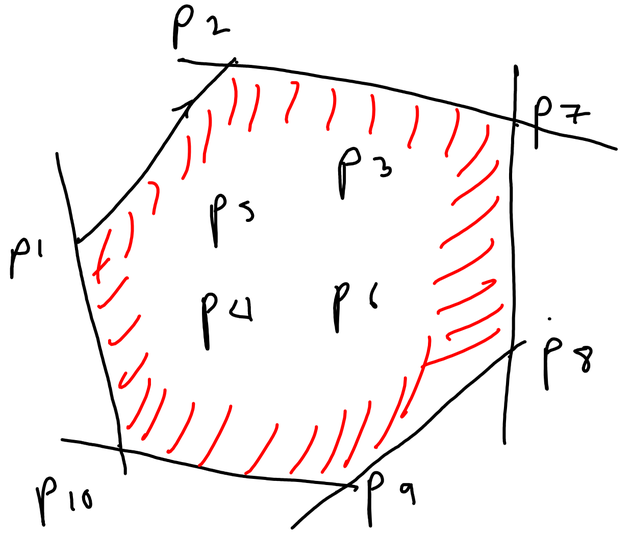
Non-convex polygon



For P finite,

$$CH(P) = \bigcap_{P \subseteq H} H$$

H a half-plane containing 2 points of P

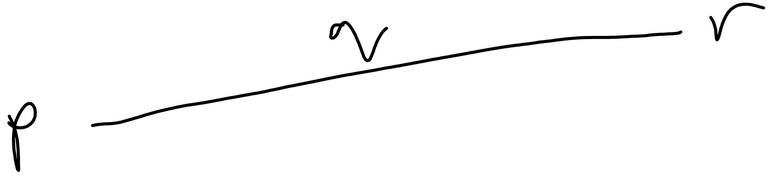


Idea: simple algorithm

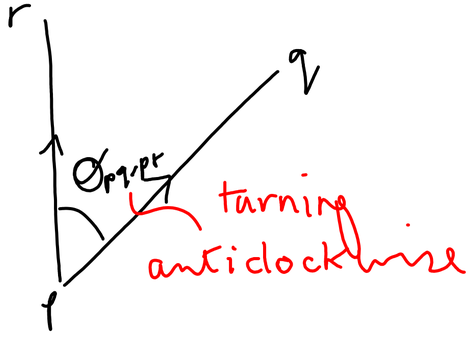
- search for directed line segments $\overrightarrow{pq}$ on the convexhull in clockwise order

eg. $\overrightarrow{p_1p_2}$ / $\overrightarrow{p_2p_7}$ ...

- A directed segment $\overrightarrow{pq}$ will lie on CH(P) if no points of P lies to its left.



When does a point r lie to the left of $\overrightarrow{pq}$?

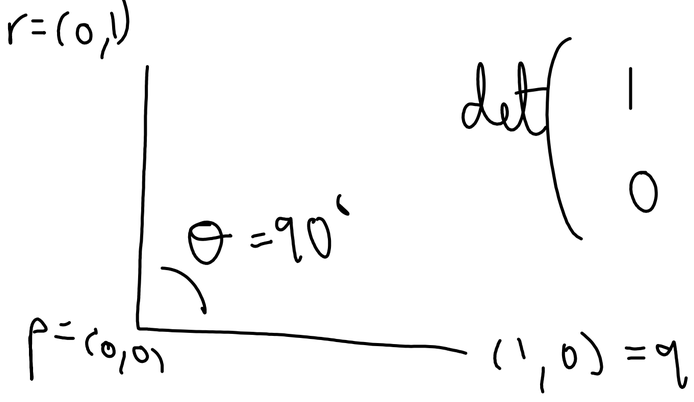


r lies to left of $\overrightarrow{pq}$ $\iff$

$0 < \theta_{pq,pr} < 180°$.

This happens $\iff$

$$\det\begin{pmatrix} q_x - p_x & r_x - p_x \\ q_y - p_y & r_y - p_y \end{pmatrix} > 0$$

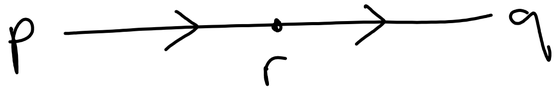


$$\det\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = 1$$

Constant time operation to check if a pt r lies to left of $\overrightarrow{pq}$.

Algorithm: SlowConvexHull(P)

- For each $p, q \in P$, test if no other point of P lies to left of $\overrightarrow{pq}$.
- If so, add $\overrightarrow{pq}$ to our list $\mathcal{L}$.



- Sort clockwise (can be done in time $O(n \log n)$ where n is no. of pts.)

Complexity

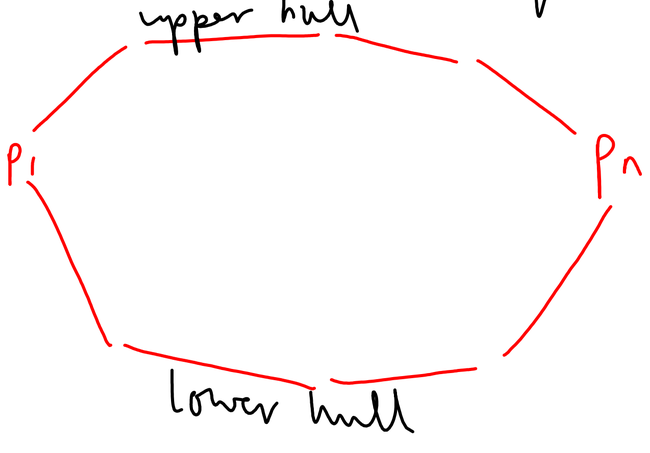
- $n(n-1)$ distinct pairs of points.
- For each such pair, check $n-2$ points (if they lie to left)
- Complexity $O(n(n-1)(n-2) + n\log n) = O(n(n-1)(n-2)) = O(n^3)$.

Faster algorithm: Graham's scan $O(n \log n)$

- Order points in P lexicographically: (left to right, top to bottom)

$$p < q \iff p_x < q_x \text{ or } (p_x = q_x \ \& \ p_y > q_y)$$

- Obtain ordered sequence $p_1 < p_2 < \dots < p_n$.

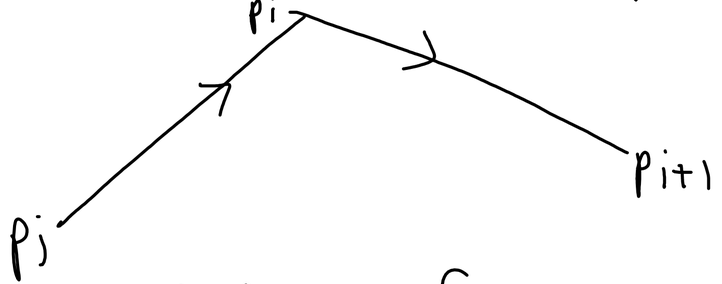


- p_1, p_n lie on the convex hull
- Convex hull splits into two parts, an upper part & a lower part.

Goal: search for upper hull & then lower hull.

Let $\mathcal{L}_i$ = upper hull for $\{p_1, p_2, \ldots, p_i\}$.

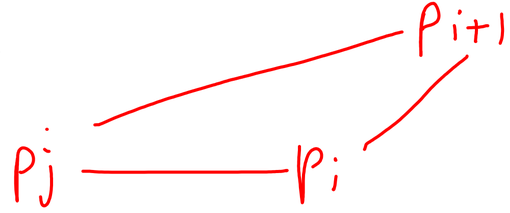
- $\mathcal{L}_2 = \{p_1, p_2\}$
- Given $\mathcal{L}_i$ construct $\mathcal{L}_{i+1}$: add p_{i+1} which must belong to $\mathcal{L}_{i+1}$.
- Consider last 3 points in $\mathcal{L}_{i+1}$: (p_j, p_i, p_{i+1}).



Say they form a <u>right turn</u> if p_{i+1} lies to the right of $\overrightarrow{p_j p_i}$ (i.e. $\det(\,) < 0$)

- If they form a right turn, we stop.

- If they do not form a right turn, delete the middle of 3 points p_i from $\mathcal{L}_{i+1}$. eg.



- Then look at last 3 points in $\mathcal{L}_{i+1}$ & repeat this step until:
  - last 3 points in $\mathcal{L}_{i+1}$ form a right turn or
  - only two points remain in $\mathcal{L}_{i+1}$.

- In this way, we obtain $\mathcal{L}_{upper} = \mathcal{L}_n$.

- Vertices of lower hull calculated similarly:

  i.e. calc lower hulls of $\{p_{n-1}, p_n\}$, $\{p_{n-2}, p_{n-1}, p_n\}$, $\ldots \{p_1, \ldots, p_n\}$

- Finally append L_{lower} to L_{upper} (firstly delete p_1, p_n from L_{lower})

Time complexity: $O(n \log n)$

- Order n pts lexicographically. Takes time $O(n \log n)$ — eg. mergesort
- On upper hull, a point is added & removed at most once — at most $2n$ actions
- On lower hull, $2n$.
- Append lists - constant

$$O(4n + n \log n) = O(n \log n).$$

Other algorithm : Gift wrapping

(output sensitive algorithm)

Complexity $O(nk)$

n — no. of points

k — no. of points on convex hull

Useful if k is small relative n as

$$O(kn) << O(n \log n).$$