Geometric Algorithms

- Lecturer : John Bourke
  bourkej@math.muni.cz
- Each week : 1 algorithm
  (+ exercise on board together at end, if possible)
- Interconnected : eg sweepline algorithm in W2 is used in many later algorithms
- E-learning course (see learning materials for link)
- Upload notes weekly to IS
- Book : Computational geometry by de Berg
- Exam at end.

List of algorithms

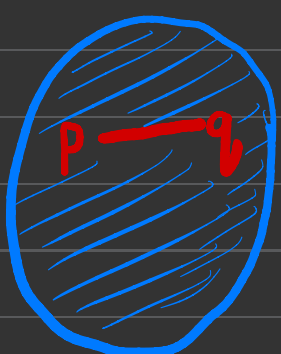
- Convex hull
- Line intersection (sweepline)
- Map overlay
- Polygon triangulation
- Half-plane intersection
- Linear programming
- Orthogonal range searching
- Point location
- Voronoi diagrams (Post-office problem)
- Delaunay triangulation

Lecture 1 - Convex hull in the plane

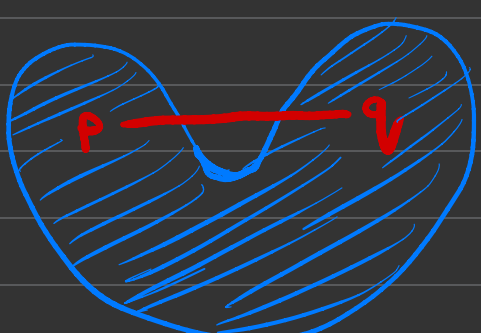
- $K \subseteq \mathbb{R}^2$ (the plane) is **convex** if for all $p, q \in K$, the line segment $\overrightarrow{pq}$ is also in K.

(Note $\overrightarrow{pq}$ contains points of form $r = p + \lambda(q-p)$ for $\lambda \in [0,1]$.)

Convex :



Non-convex :



Convex Hull

Let $P \subseteq \mathbb{R}^2$.

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

convex hull

We have obvious formula

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

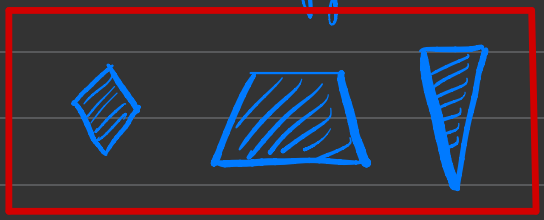
– ie intersection of all convex sets containing P

but not computationally useful since can be infinitely many convex sets containing P.

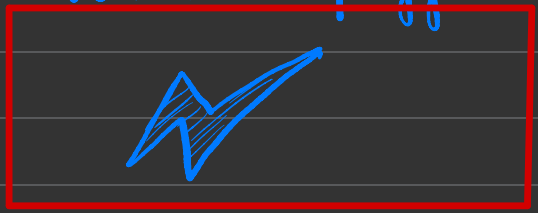
Goal : compute CH(P) for P Finite

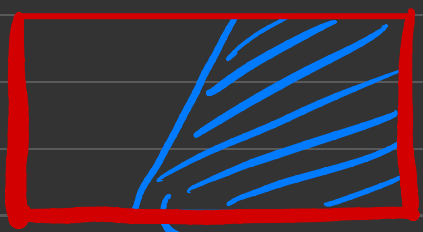
P Finite $\Rightarrow$ CH(P) a convex polygon :
ie. bounded by finitely many straight-line segments.

Convex Polygons



Non-convex polygons





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

For P Finite,
$CH(P) = \bigcap H$

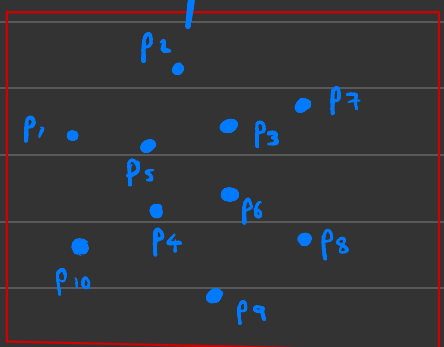
$P \subseteq H$ a half-plane containing 2 points of P

Goal: compute CH(P) for P finite

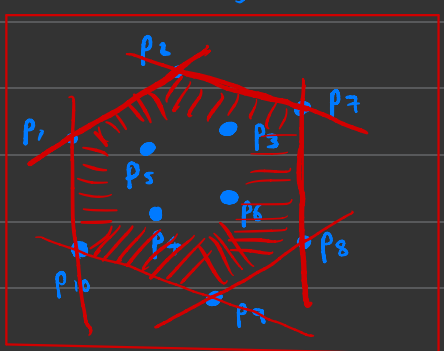
For P finite, $CH(P) = \bigcap H$

$P \subseteq H$ a half-plane containing 2 points of P

Example: P



CH(P)

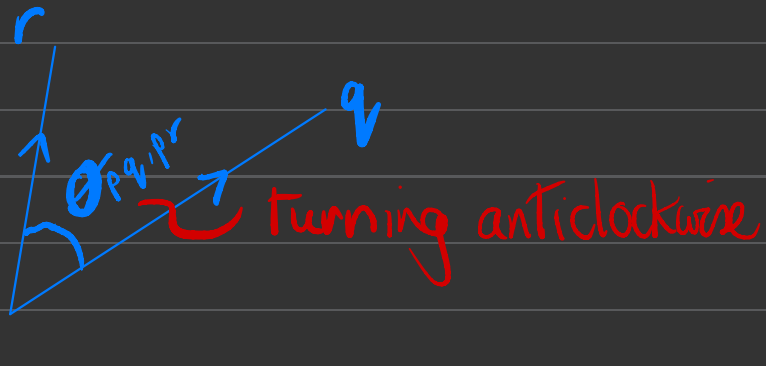


Idea for simple algorithm:

- search for directed line segments $\overrightarrow{pq}$ on convex hull in clockwise order - eg. $\overrightarrow{p_1p_2}$, $\overrightarrow{p_2p_7}$, ...
- a directed line segment will lie on CH(P) if no point of P lies to its left.

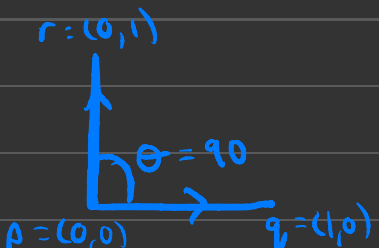
When does point r lie to left of $\overrightarrow{pq}$?

Consider



- Then r lies to left of $\overrightarrow{pq}$ $\Leftrightarrow$ $0 < \theta_{\overrightarrow{pq},\overrightarrow{pr}} < 180°$.

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

- E.g. 

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

So constant time operation to check if 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 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 $n-2$ pts (if they lie to left)

So complexity

$O(n(n-1)(n-2) + n\log n) = \underline{O(n^3)}$.

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

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

  i.e. $p < q \iff p_x < q_x$ or $(p_x = q_x \ \& \ p_y > q_y)$

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

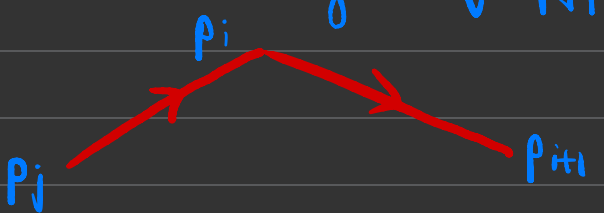
- p_1, p_n lie on convex hull.
- Convex hull splits in 2 parts : upper hull & lower hull.

Goal : search for upper hull, then lower hull.

Finding upper hull

- U_i := upper hull for $\{p_1, p_2, \ldots, p_i\}$.
- $U_2 = \{p_1, p_2\}$.
- Given U_i construct U_{i+1}
  - add p_{i+1} which must belong to U_{i+1}.
  - consider last 3 pts (p_j, p_i, p_{i+1}) in U_{i+1}.

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



- If they form a right turn, we stop.

Finding upper hull ctd

- If not, delete middle of 3 pts p_i from L_{i+1} (eg. p_j — p_i — p_{i+1})

- Then look at last 3 pts in L_{i+1} & repeat this step until:

- last 3 pts form a right turn or only 2 pts remain.

- In this way, obtain

$$L_{upper} = L_n .$$

End of algorithm

- $\mathcal{L}_{lower}$ is 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 $\mathcal{L}_{lower}$ to $\mathcal{L}_{upper}$ (firstly delete p_1, p_n from $\mathcal{L}_{lower}$)

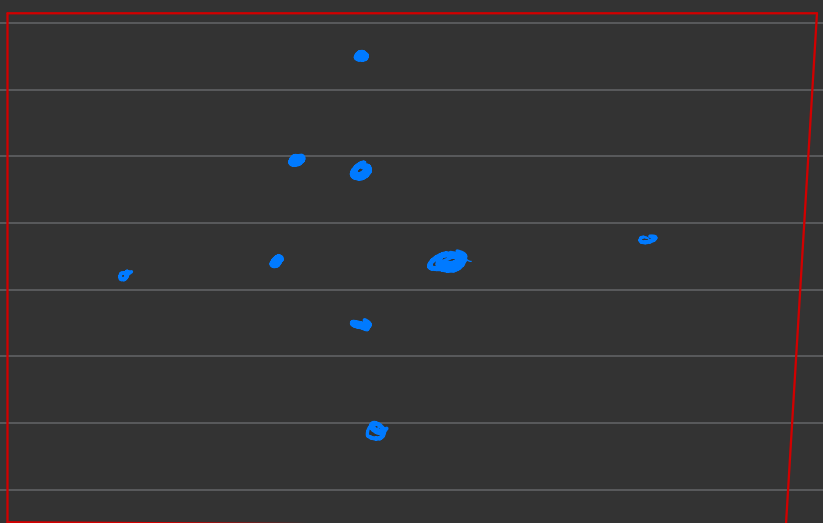
Time Complexity $O(n \log n)$

- Order n pts lexicographically - takes $O(n \log n)$ eg. mergesort
- On upper hull, constant time to add / remove a pt.
- Each pt added & removed at most once - at most $2n$ actions
- On lower hull, also $2n$.
- Append lists: n

$O(5n + n \log n) = O(n \log n)$

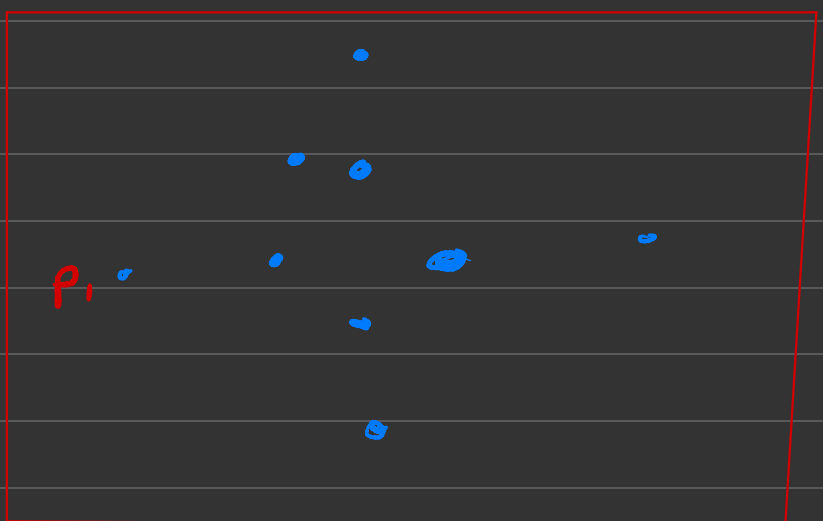
Another algorithm : Gift wrapping

- Useful if know in advance, no. of pts on convex hull $\leq k$, where k is small compared to n.



Another algorithm : Gift wrapping

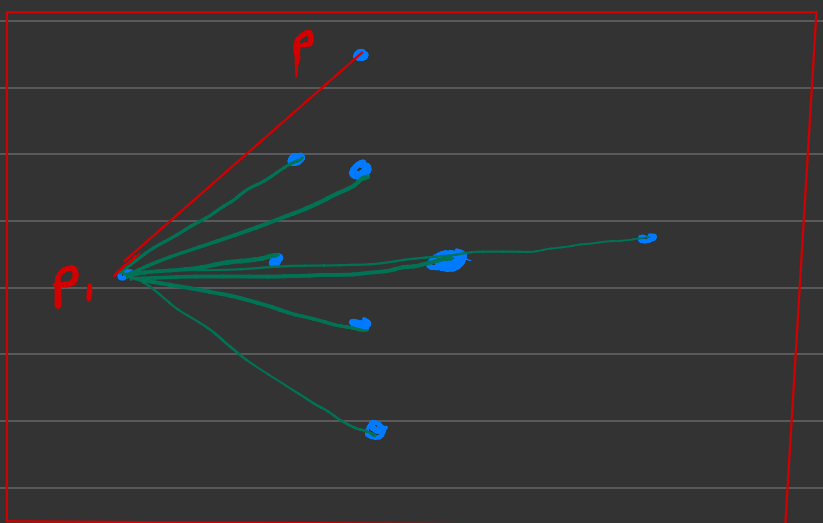
- Useful if know in advance, no. of pts on convex hull $\leq k$, where k is small compared to n.



- Find leftmost point p_1.

Another algorithm : Gift wrapping

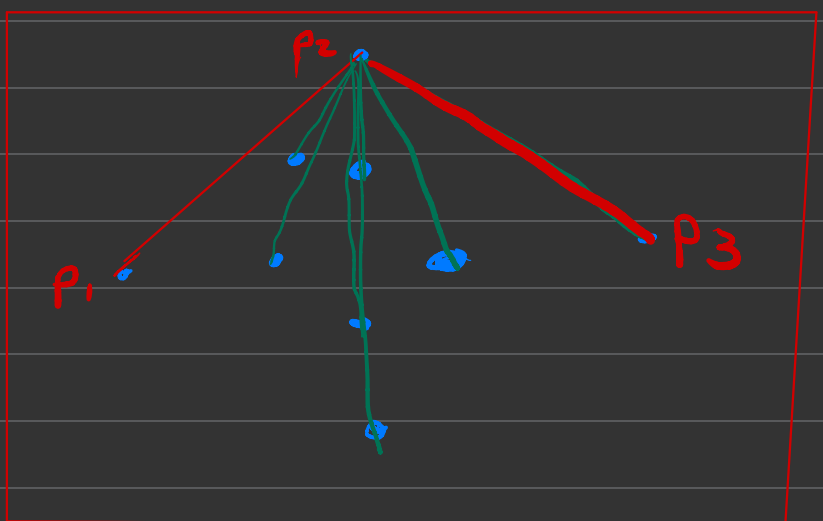
- Useful if know in advance, no. of pts on convex hull $\leq k$, where k is small compared to n.



- Find leftmost point p_1.
- Draw lines to other pts, & choose one of greatest slope. Endpoint p_2.

Another algorithm : Gift wrapping

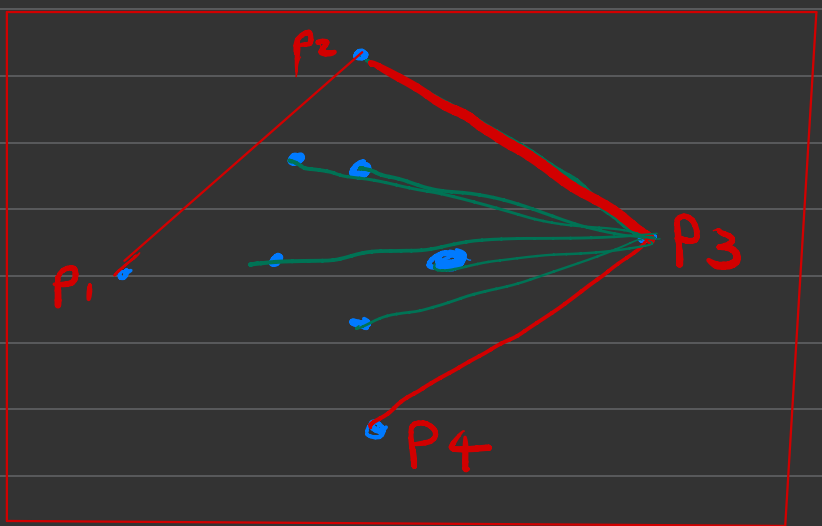
- Useful if know in advance, no. of pts on convex hull $\leq k$, where k is small compared to n.



- Find leftmost point p_1.
- Draw lines to other pts, & choose one of greatest slope. Endpoint p_2.
- Draw lines to remaining pts & choose p_3 so $\angle p_1p_2p_3$ is largest.

Another algorithm : Gift wrapping

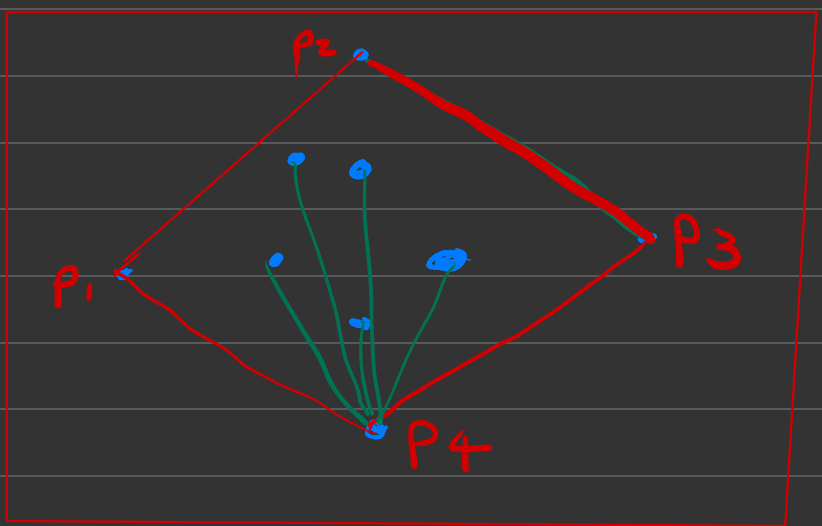
- Useful if know in advance, no. of pts on convex hull $\leq k$, where k is small compared to n.



- Find leftmost point p_1.
- Draw lines to other pts, & choose one of greatest slope. Endpoint p_2.
- Draw lines to remaining pts & choose p_3 so $\angle p_1p_2p_3$ is largest.
- Repeat until return to p_1.

Another algorithm : Gift wrapping

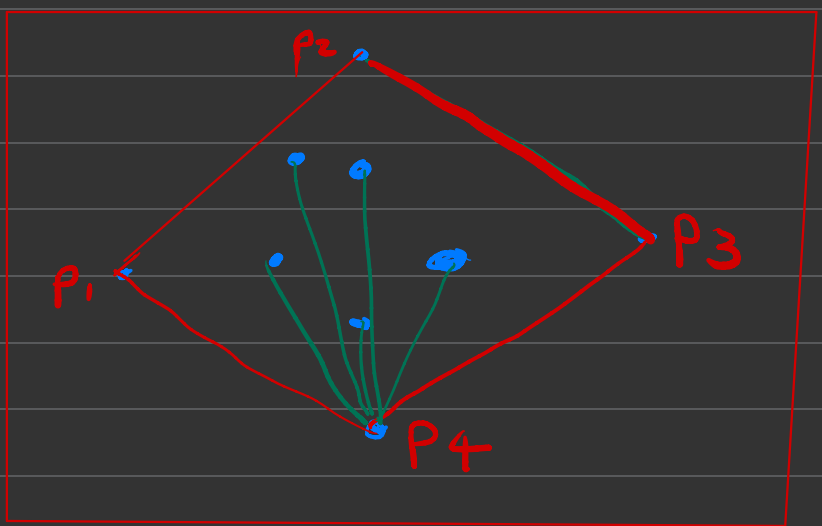
- Useful if know in advance, no. of pts on convex hull $\leq k$, where k is small compared to n.



- Find leftmost point p_1.
- Draw lines to other pts, & choose one of greatest slope. Endpoint p_2.
- Draw lines to remaining pts & choose p_3 so $\angle p_1p_2p_3$ is largest.
- Repeat until return to p_1.

Another algorithm : Gift wrapping

- Useful if know in advance, no. of pts on convex hull $\leq k$, where k is small compared to n.



- Find leftmost point p_1.
- Draw lines to other pts, & choose one of greatest slope. Endpoint p_2.
- Draw lines to remaining pts & choose p_3 so $\angle p_1p_2p_3$ is largest.
- Repeat until return to p_1.
- Complexity $O(kn)$

Exercise

- Apply Graham's scan to calc. convex hull of