Convex hulls in $\mathbb{R}^2$

Sándor Kisfaludi-Bak

Computaional Geometry
Summer semester 2020

max planck institut
informatik

Overview

Overview

- Problem definition

Overview

- Problem definition
- Computaitonal models, input and output

Overview

- Problem definition
- Computaitonal models, input and output
- Naive algorithm

Overview

- Problem definition
- Computaitonal models, input and output
- Naive algorithm
- Graham's scan

Overview

- Problem definition
- Computaitonal models, input and output
- Naive algorithm
- Graham's scan
- Chan's algorithm

Convex hull

Notations, definitions

$\mathbb{R}^d$ is d-dimensional Euclidean space

$P = \{p_1, \ldots, p_n\}$ set of n points

$X \subseteq \mathbb{R}^d$ is *convex* if for any $p, q \in X$ we have $pq \subseteq X$

Convex hull

Notations, definitions

$\mathbb{R}^d$ is d-dimensional Euclidean space

$P = \{p_1, \ldots, p_n\}$ set of n points

$X \subseteq \mathbb{R}^d$ is *convex* if for any $p, q \in X$ we have $pq \subseteq X$

Convex hull:

$$\mathrm{conv}(P) = \begin{cases} \text{minimum convex set containing } P \\ \text{intersection of convex sets containing } P \\ \{\alpha_1 p_1 + \cdots + \alpha_n p_n \mid \alpha_i \geq 0 \text{ and } \sum_{i=1}^{n} \alpha_i = 1\} \end{cases}$$

Convex hull

Notations, definitions

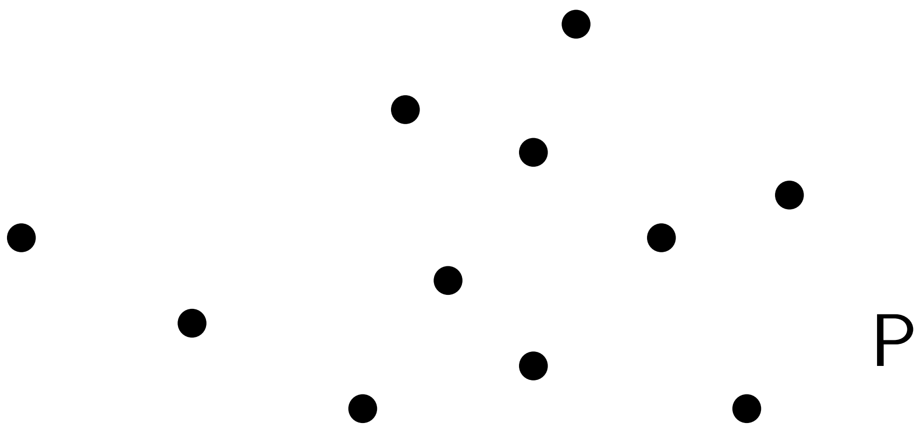
$\mathbb{R}^d$ is d-dimensional Euclidean space

$P = \{p_1, \ldots, p_n\}$ set of n points

$X \subseteq \mathbb{R}^d$ is *convex* if for any $p, q \in X$ we have $pq \subseteq X$

Convex hull:

$$\mathrm{conv}(P) = \begin{cases} \text{minimum convex set containing } P \\ \text{intersection of convex sets containing } P \\ \{\alpha_1 p_1 + \cdots + \alpha_n p_n \mid \alpha_i \geq 0 \text{ and } \sum_{i=1}^n \alpha_i = 1\} \end{cases}$$



Convex hull

Notations, definitions

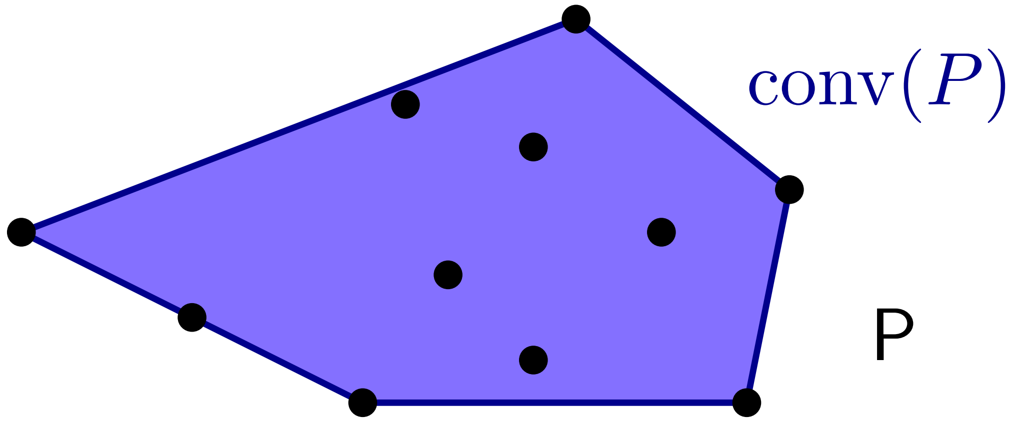
$\mathbb{R}^d$ is d-dimensional Euclidean space

$P = \{p_1, \dots, p_n\}$ set of n points

$X \subseteq \mathbb{R}^d$ is *convex* if for any $p, q \in X$ we have $pq \subseteq X$

Convex hull:

$$\text{conv}(P) = \begin{cases} \text{minimum convex set containing } P \\ \text{intersection of convex sets containing } P \\ \{\alpha_1 p_1 + \dots + \alpha_n p_n \mid \alpha_i \geq 0 \text{ and } \sum_{i=1}^n \alpha_i = 1\} \end{cases}$$



Real RAM vs. Word RAM

Real RAM

Word RAM

Real RAM vs. Word RAM

Real RAM

arbitrary real numbers

Word RAM

words of size $\Theta(\log n)$

Real RAM vs. Word RAM

Real RAM	Word RAM
arbitrary real numbers	words of size $\Theta(\log n)$
no rounding/floor, no modulo	realistic*operations (shifts, etc)

Real RAM vs. Word RAM

Real RAM

arbitrary real numbers

no rounding/floor, no modulo

Real inputs and outputs,
can extend with $\sqrt{.}, \ln(.)$

Word RAM

words of size $\Theta(\log n)$

realistic* operations (shifts, etc)

Exact arithmetic for
rational inputs with $+ - * /$

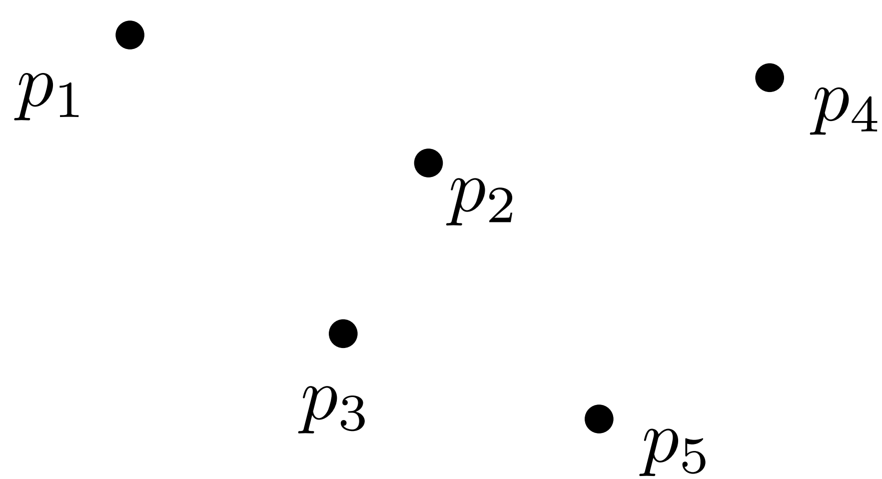
Real RAM vs. Word RAM

Real RAM	Word RAM
arbitrary real numbers	words of size $\Theta(\log n)$
no rounding/floor, no modulo	realistic* operations (shifts, etc)
Real inputs and outputs, can extend with $\sqrt{.}, \ln(.)$	Exact arithmetic for rational inputs with $+ - * /$
Unrealistic power	Too restrictive?

Convex hull: input and output

Input: Points with coordianate pairs $(x, y) \in \mathbb{R}^2$

$$(e, \pi), (3, 3), (2.95, 2.9), (\sqrt{11}, 3.05), (\pi, e)$$



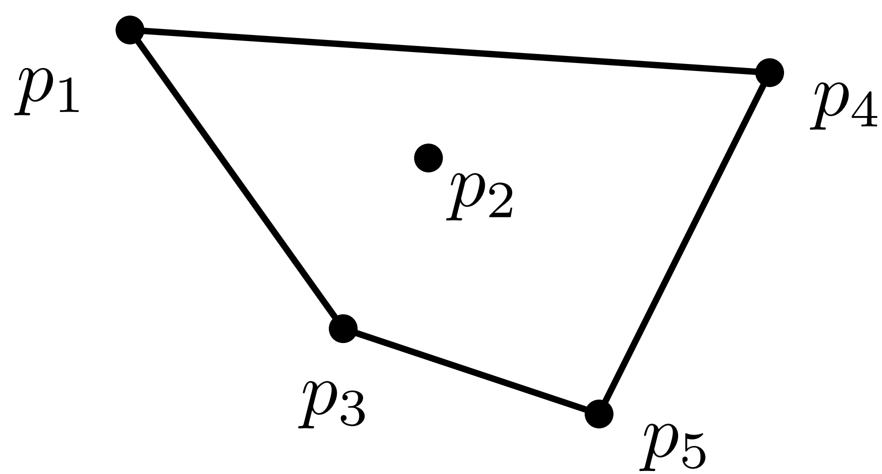
Convex hull: input and output

Input: Points with coordianate pairs $(x, y) \in \mathbb{R}^2$

$$(e, \pi), (3, 3), (2.95, 2.9), (\sqrt{11}, 3.05), (\pi, e)$$

Output: "corners" in clockwise order
smallest $Q \subseteq P$ s.t. $\mathrm{conv}(Q) = \mathrm{conv}(P)$

$$p_1, p_4, p_5, p_3$$



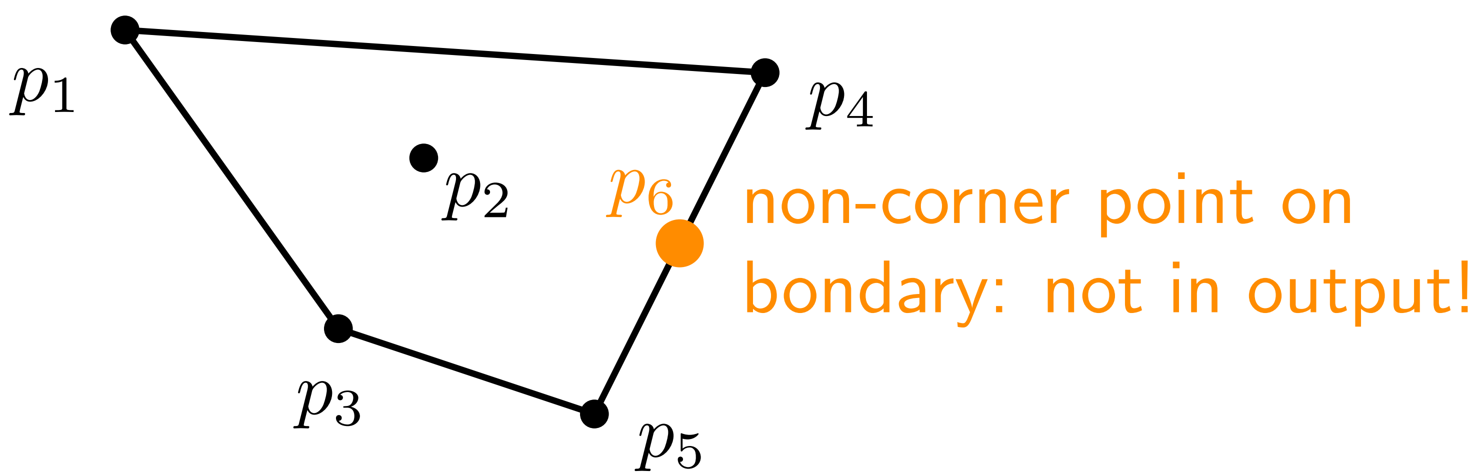
Convex hull: input and output

Input: Points with coordianate pairs $(x, y) \in \mathbb{R}^2$

$$(e, \pi), (3, 3), (2.95, 2.9), (\sqrt{11}, 3.05), (\pi, e)$$

Output: "corners" in clockwise order
smallest $Q \subseteq P$ s.t. $\mathrm{conv}(Q) = \mathrm{conv}(P)$

p_1, p_4, p_5, p_3



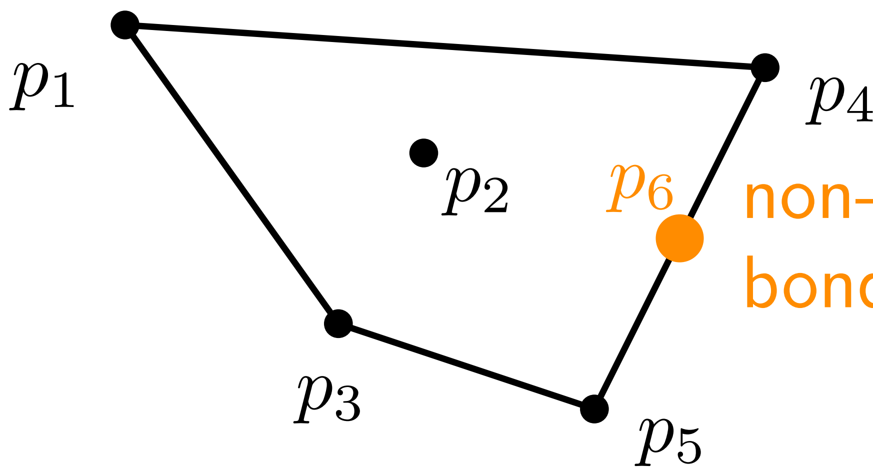
Convex hull: input and output

Input: Points with coordianate pairs $(x, y) \in \mathbb{R}^2$

$$(e, \pi), (3, 3), (2.95, 2.9), (\sqrt{11}, 3.05), (\pi, e)$$

Output: "corners" in clockwise order
smallest $Q \subseteq P$ s.t. $\mathrm{conv}(Q) = \mathrm{conv}(P)$

p_1, p_4, p_5, p_3



non-corner point on bondary: not in output!

Everything works with rational inputs on Word RAM!

Naive algorithm

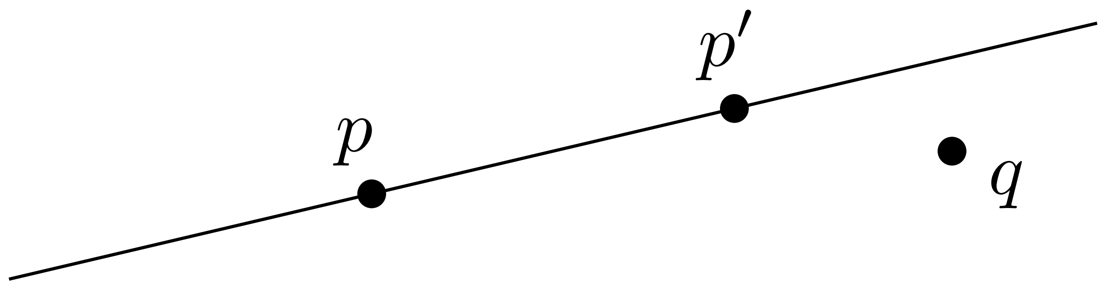
Naive Algorithm

Suppose no 3 points on one line. (no *collinear triples*)

Naive Algorithm

Suppose no 3 points on one line. (no *collinear triples*)

In $O(1)$ time, decide if q is on left or right side of line pp'

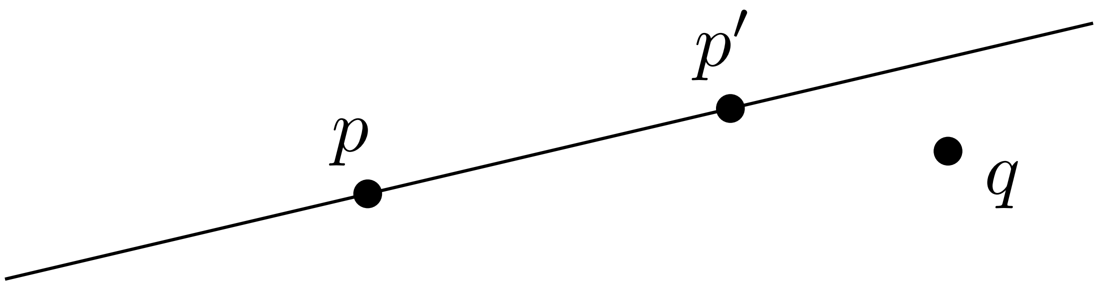


$$\operatorname{sgn}\left(\begin{vmatrix} p_x & p_y & 1 \\ p'_x & p'_y & 1 \\ q_x & q_y & 1 \end{vmatrix}\right)$$

Naive Algorithm

Suppose no 3 points on one line. (no *collinear triples*)

In $O(1)$ time, decide if q is on left or right side of line pp'



$$\operatorname{sgn}\left(\begin{vmatrix} p_x & p_y & 1 \\ p'_x & p'_y & 1 \\ q_x & q_y & 1 \end{vmatrix}\right)$$

Naive Convex Hull in $\mathbb{R}^2$

For each $p, p' \in P$,

check if all $q \in P \setminus \{p, p'\}$ is on the left of line pp'.

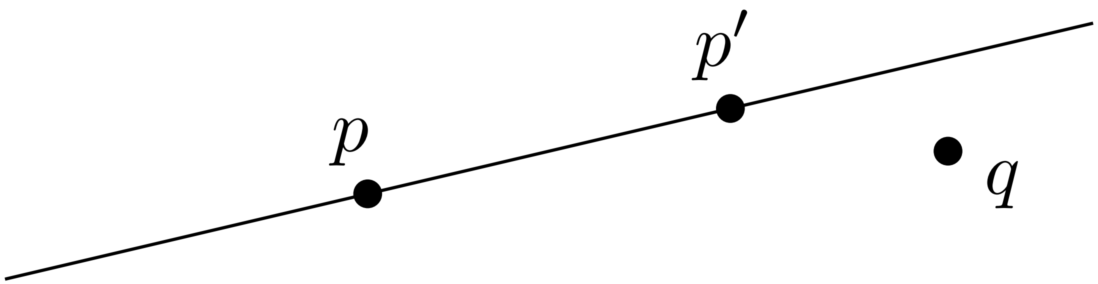
If yes, then p' follows p in $\operatorname{conv}(P)$.

Assemble and output the hull

Naive Algorithm

Suppose no 3 points on one line. (no *collinear triples*)

In $O(1)$ time, decide if q is on left or right side of line pp'



$$\operatorname{sgn}\left(\begin{vmatrix} p_x & p_y & 1 \\ p'_x & p'_y & 1 \\ q_x & q_y & 1 \end{vmatrix}\right)$$

Naive Convex Hull in $\mathbb{R}^2$

For each $p, p' \in P$,

check if all $q \in P \setminus \{p, p'\}$ is on the left of line pp'.

If yes, then p' follows p in $\mathrm{conv}(P)$.

Assemble and output the hull

Running time: $\binom{n}{2} \cdot (n-2) \cdot O(1) = O(n^3)$

Graham's scan (1972)

Graham's Scan idea

Suppose points have distinct x-coordinates.

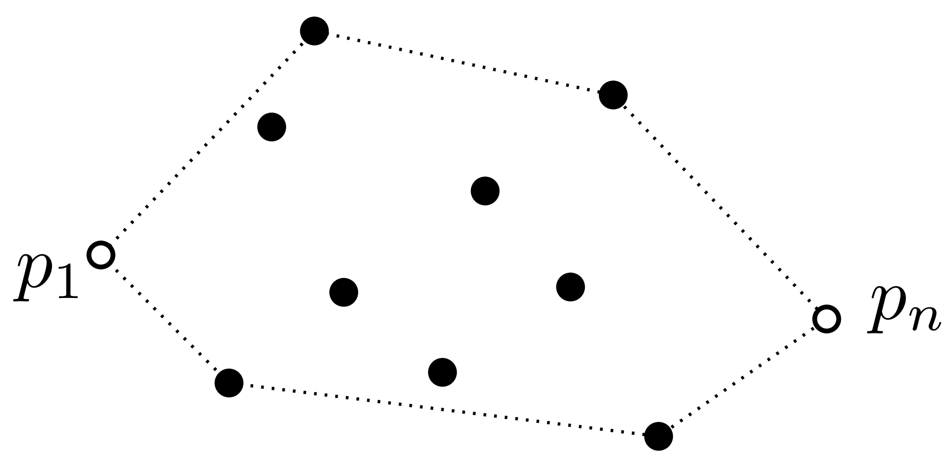
Let $p_1, \ldots, p_n$: points sorted with increasing x-coordinates.

Graham's Scan idea

Suppose points have distinct x-coordinates.

Let $p_1, \ldots, p_n$: points sorted with increasing x-coordinates.

$\twoheadrightarrow$ p_1, p_n are on convex hull



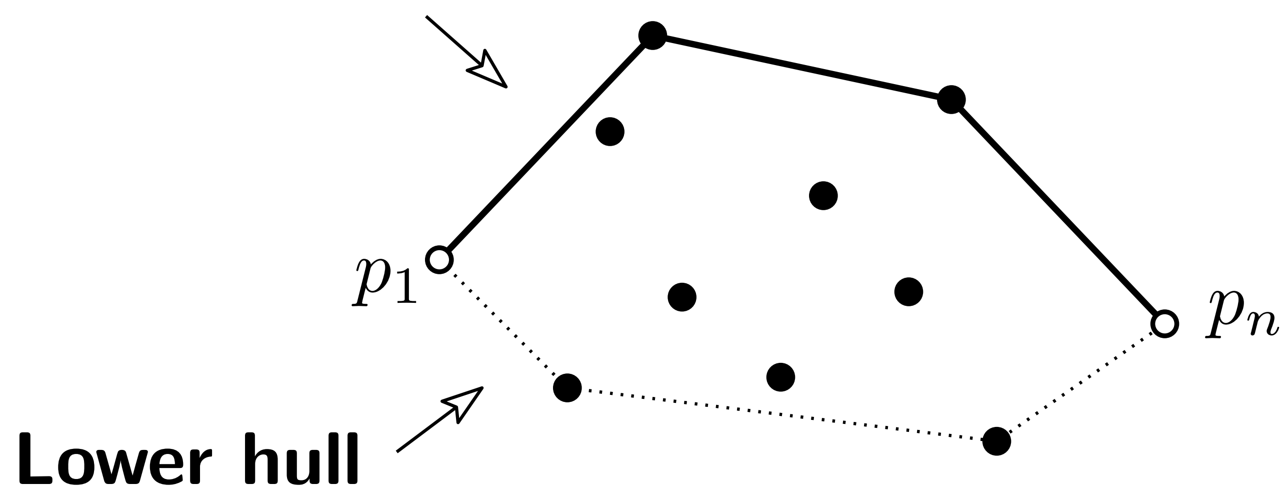
Graham's Scan idea

Suppose points have distinct x-coordinates.

Let $p_1, \ldots, p_n$: points sorted with increasing x-coordinates.

$\twoheadrightarrow$ p_1, p_n are on convex hull

Upper hull
part of the hull after p_1 and before p_n in clockwise order



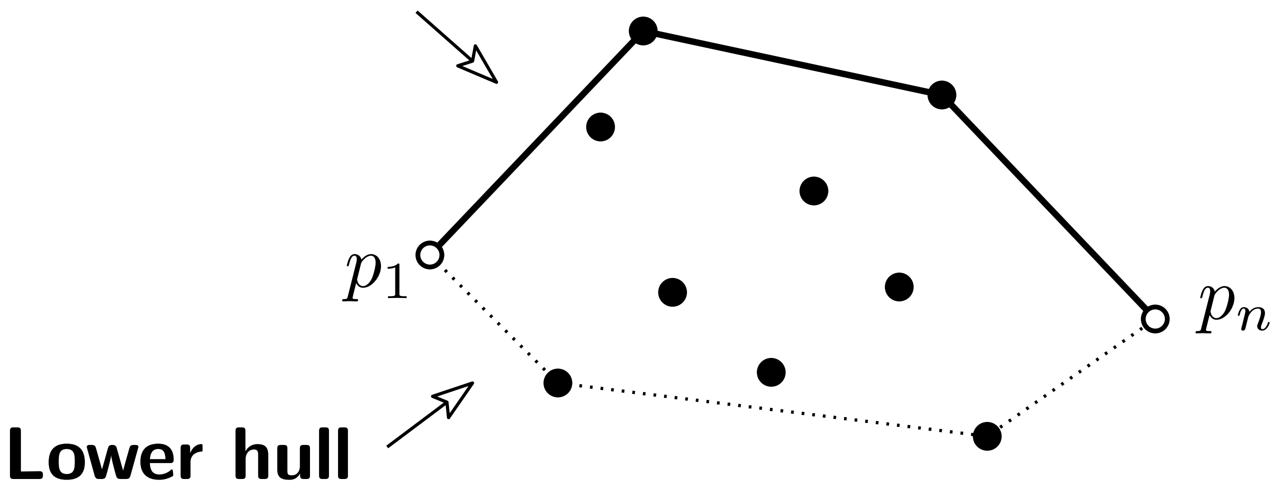
Graham's Scan idea

Suppose points have distinct x-coordinates.

Let $p_1, \ldots, p_n$: points sorted with increasing x-coordinates.

$\rightarrow$ p_1, p_n are on convex hull

Upper hull
part of the hull after p_1 and before p_n in clockwise order

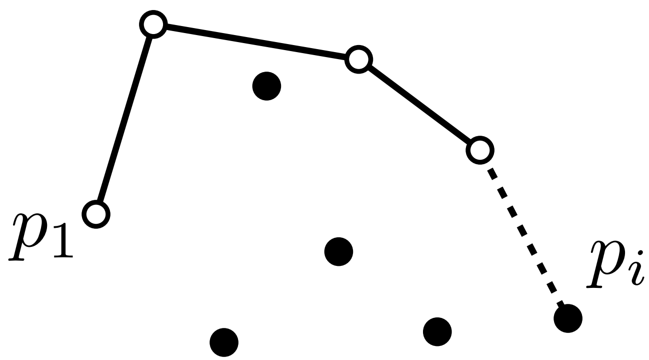


Idea:
Add points left to right, update upper hull after each addition

Graham's Scan: update

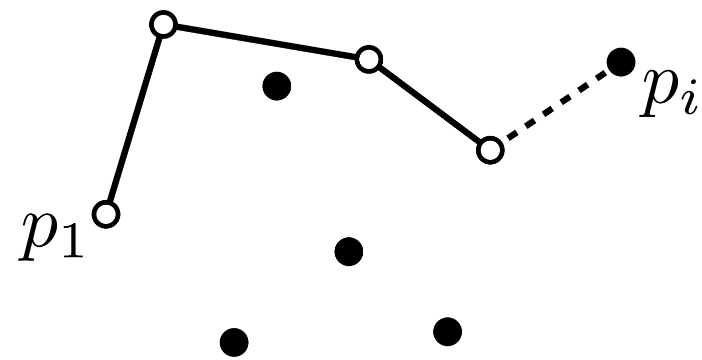
Right turn

(p_i is below last hull segment)



Left turn

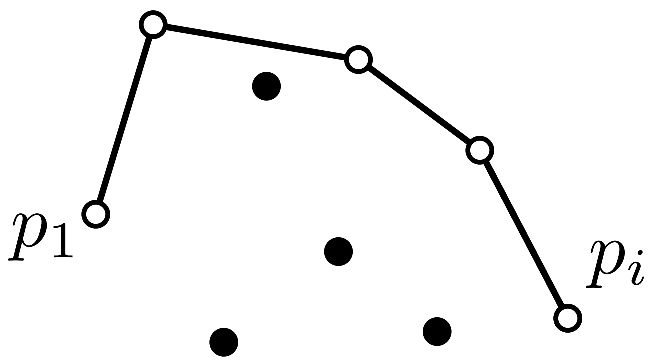
(p_i is above last hull segment)



Graham's Scan: update

Right turn

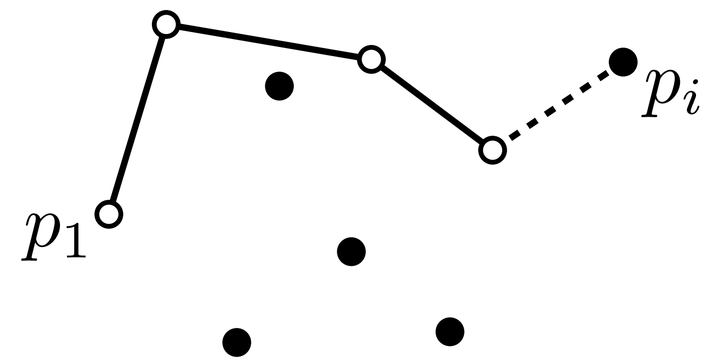
(p_i is below last hull segment)



Add p_i to the upper hull

Left turn

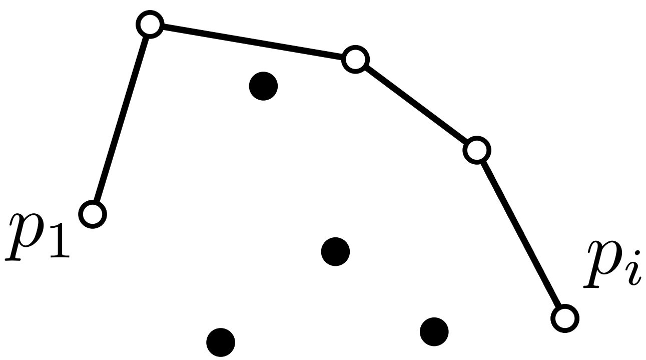
(p_i is above last hull segment)



Graham's Scan: update

Right turn

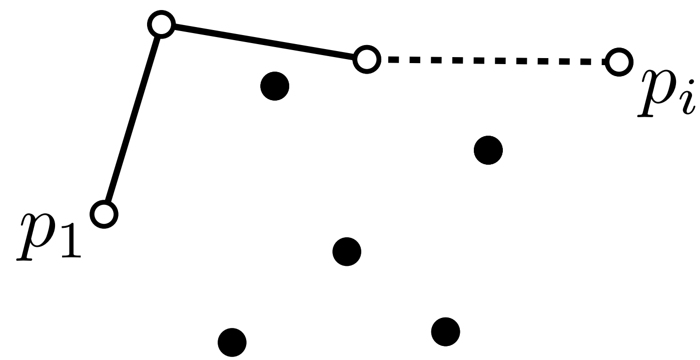
(p_i is below last hull segment)



Add p_i to the upper hull

Left turn

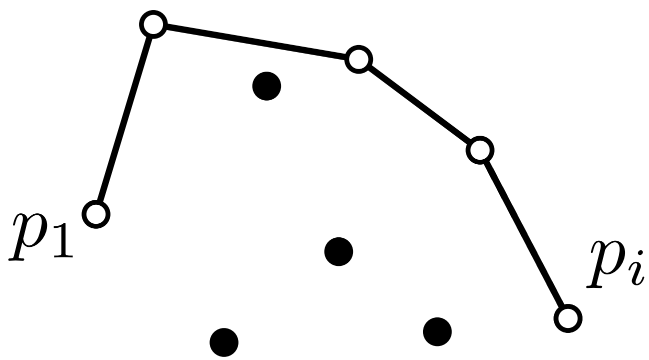
(p_i is above last hull segment)



Graham's Scan: update

Right turn

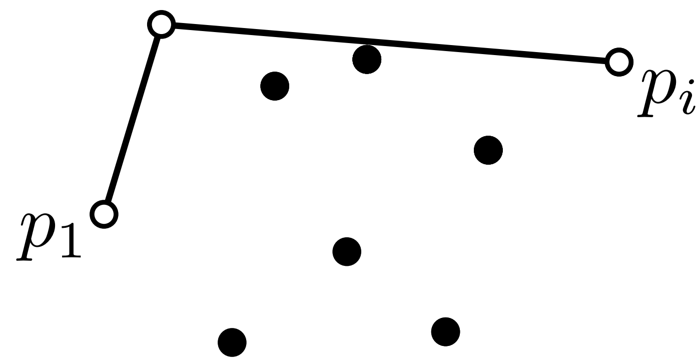
(p_i is below last hull segment)



Add p_i to the upper hull

Left turn

(p_i is above last hull segment)

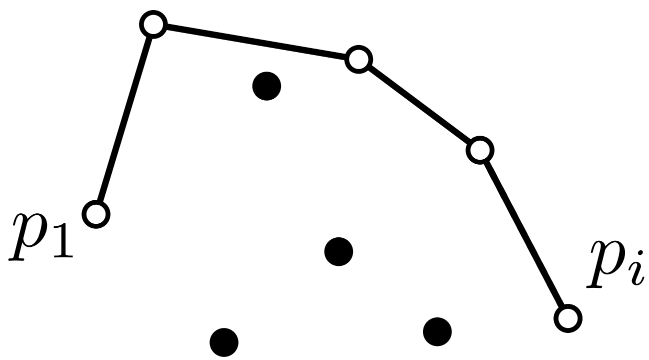


Add p_i but remove previous hull point until left turn disappears

Graham's Scan: update

Right turn

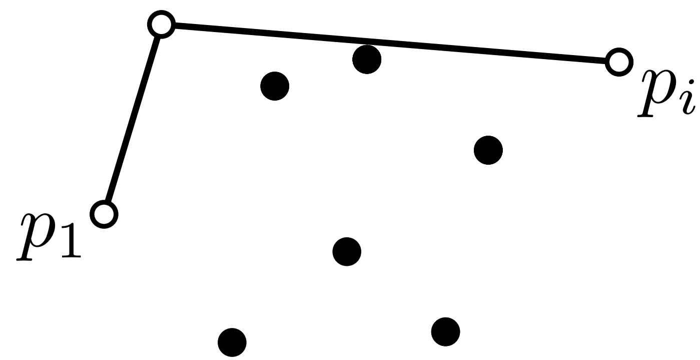
(p_i is below last hull segment)



Add p_i to the upper hull

Left turn

(p_i is above last hull segment)



Add p_i but remove previous hull point until left turn disappears

Similalrly for lower hull, after adding p_i:

while last three points of lower hull q, q', p_i are a right turn:

remove the middle point q'

Graham's Scan: pseudocode + runtime

```
Sort P by increasing x-coordinates
Add p_1, p_2 to U and L
for i = 3 to n do
    Add p_i to U and L
    while last three pts of U form left turn do
        Remove pt preceding p_i from U
    while last three pts of L form right turn do
        Remove pt preceding p_i from L
return L and reverse of U
```

Graham's Scan: pseudocode + runtime

```
Sort P by increasing x-coordinates
Add p_1, p_2 to U and L
for i = 3 to n do
    Add p_i to U and L
    while last three pts of U form left turn do
        Remove pt preceding p_i from U
    while last three pts of L form right turn do
        Remove pt preceding p_i from L
return L and reverse of U
```

Running time:

Sorting $\longrightarrow O(n \log n)$

Graham's Scan: pseudocode + runtime

```
Sort P by increasing x-coordinates
Add p_1, p_2 to U and L
for i = 3 to n do
    Add p_i to U and L
    while last three pts of U form left turn do
        Remove pt preceding p_i from U
    while last three pts of L form right turn do
        Remove pt preceding p_i from L
return L and reverse of U
```

Running time:

Sorting $\longrightarrow O(n \log n)$

Each $p \in P$ is:

- added once to U (same for L) $\longrightarrow O(n)$
- removed at most once from U (same for L) $\longrightarrow O(n)$

Triplets checked in While loop heads $\longrightarrow O(n)$

Graham's Scan: pseudocode + runtime

```
Sort P by increasing x-coordinates
Add p_1, p_2 to U and L
for i = 3 to n do
    Add p_i to U and L
    while last three pts of U form left turn do
        Remove pt preceding p_i from U
    while last three pts of L form right turn do
        Remove pt preceding p_i from L
return L and reverse of U
```

Running time: $O(n \log n)$

Graham's Scan: correctness

Claim

After each iteration of main loop, U is upper hull of $p_1, \ldots, p_i$.

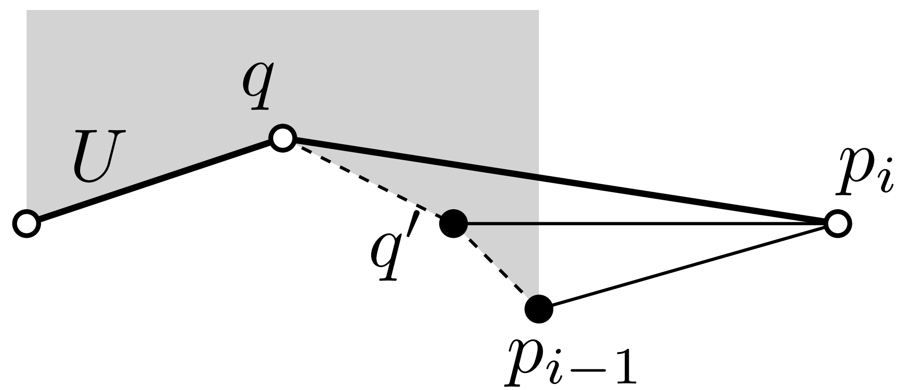
Graham's Scan: correctness

Claim
After each iteration of main loop, U is upper hull of $p_1, \ldots, p_i$.

Induction on i. Works for $i \leq 2$.
Suppose U is the upper hull of $p_1, \ldots, p_{i-1}$

$\Rightarrow$ Gray is empty

p_i is added to U ✓



Graham's Scan: correctness

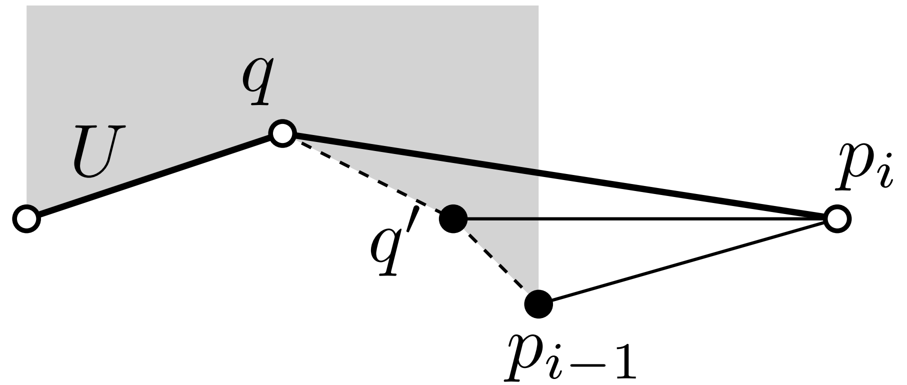
Claim
After each iteration of main loop, U is upper hull of $p_1, \ldots, p_i$.

Induction on i. Works for $i \leq 2$.
Suppose U is the upper hull of $p_1, \ldots, p_{i-1}$

$\Rightarrow$ Gray is empty

p_i is added to U ✓

q', p_{i-1}, p_i "left turn" $\Leftrightarrow$ p_{i-1} is below $q'p_i$.
Similarly, q' is below qp_i
$\Rightarrow$ All deleted vertices are below the new U.



Graham's Scan: correctness

Claim

After each iteration of main loop, U is upper hull of $p_1, \ldots, p_i$.

Induction on i. Works for $i \leq 2$.

Suppose U is the upper hull of $p_1, \ldots, p_{i-1}$

$\Rightarrow$ Gray is empty

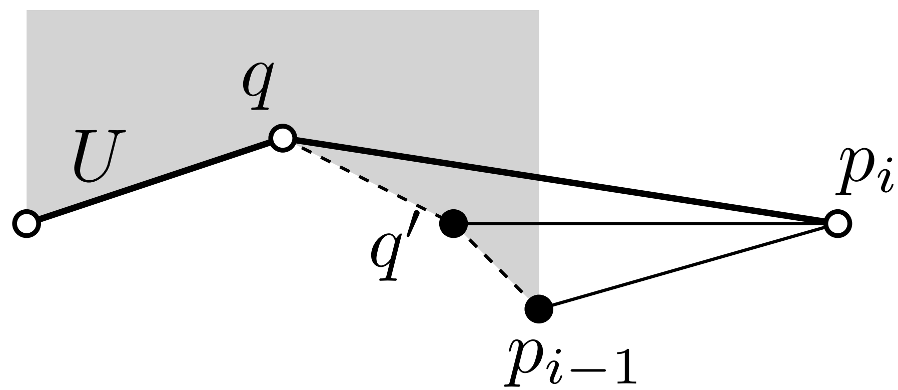
p_i is added to U ✓

q', p_{i-1}, p_i "left turn" $\Leftrightarrow$ p_{i-1} is below $q'p_i$.

Similarly, q' is below qp_i

$\Rightarrow$ All deleted vertices are below the new U.

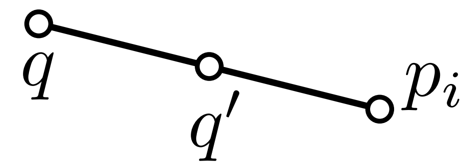
$\Rightarrow$ old U and all of $p_1, \ldots p_{i-1}$ are on or below new U □



Graham's Scan: non-general position

- collinear triple:

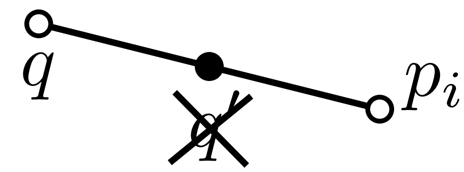
Instead of deleting for left turn in U:
Delete if straight or left turn



Graham's Scan: non-general position

- collinear triple:

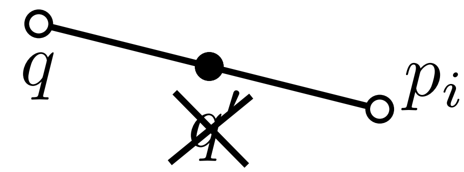
Instead of deleting for left turn in U:
Delete if straight or left turn



Graham's Scan: non-general position

- collinear triple:

Instead of deleting for left turn in U:
Delete if straight or left turn

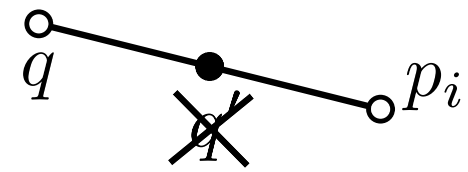


- equal x-coordiantes:

Graham's Scan: non-general position

- collinear triple:

Instead of deleting for left turn in U:
Delete if straight or left turn



- equal x-coordiantes:

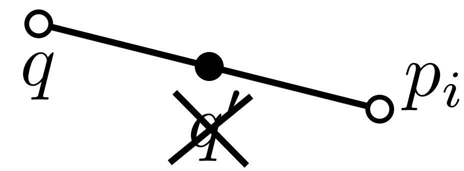
Use *lexicographic* order for initial sort.

$$(x, y) <_{lex} (x', y') \text{ iff } x < x' \vee (x = x' \wedge y < y')$$

Graham's Scan: non-general position

- collinear triple:

Instead of deleting for left turn in U:
Delete if straight or left turn



- equal x-coordiantes:

Use *lexicographic* order for initial sort.

$$(x, y) <_{lex} (x', y') \text{ iff } x < x' \vee (x = x' \wedge y < y')$$

p_1, p_n are still on the hull.
Upper hull U: part of hull after p_1 and before p_n in cw order

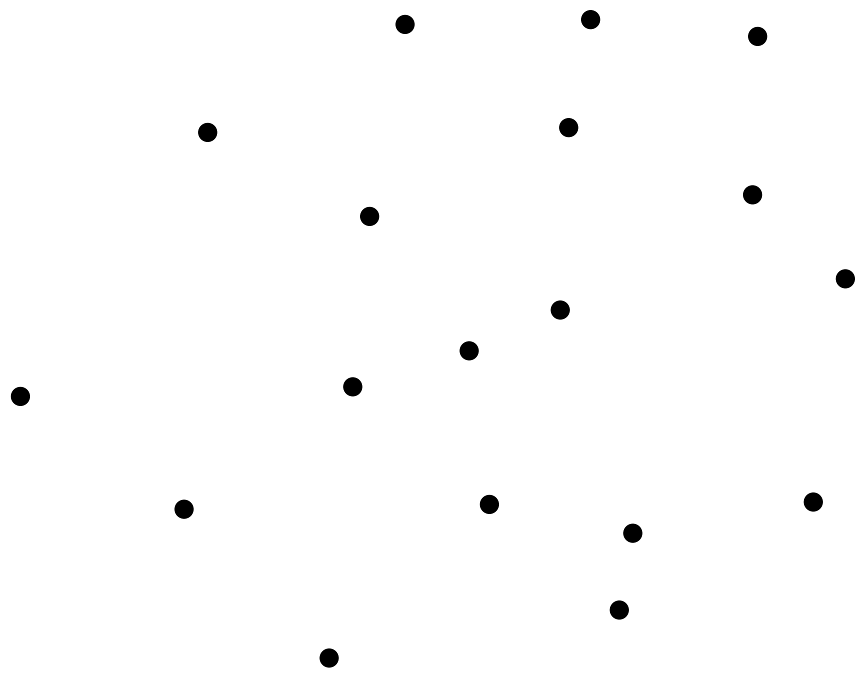
Practicalities

Collinear triples are common (grids!)

Practicalities

Collinear triples are common (grids!)

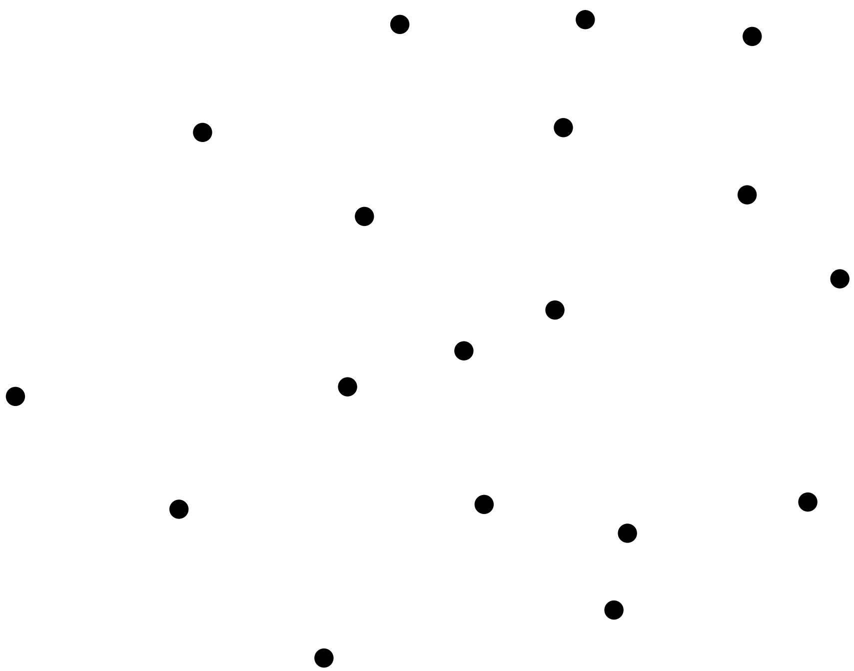
Almost collinear triples are also common!



Practicalities

Collinear triples are common (grids!)

Almost collinear triples are also common!



P uniform random in $[0,1]^2$:

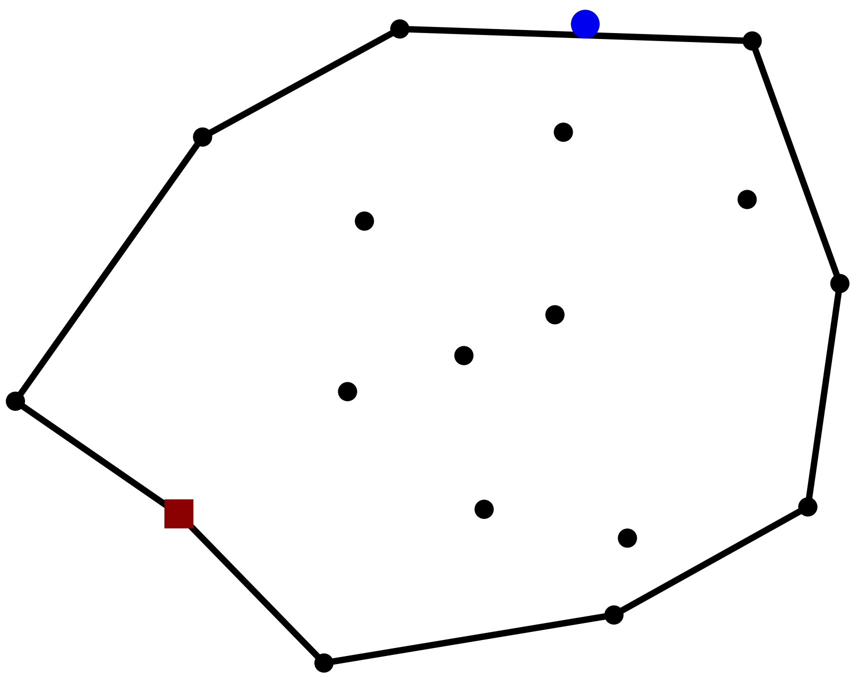
$$\mathbb{E}(\max_{p,q,r \in P} \sphericalangle pqr) = \pi - O(1/n^3)$$

not peer-reviewed source

Practicalities

Collinear triples are common (grids!)

Almost collinear triples are also common!



P uniform random in $[0,1]^2$:

$$\mathbb{E}(\max_{p,q,r \in P} \sphericalangle pqr) = \pi - O(1/n^3)$$

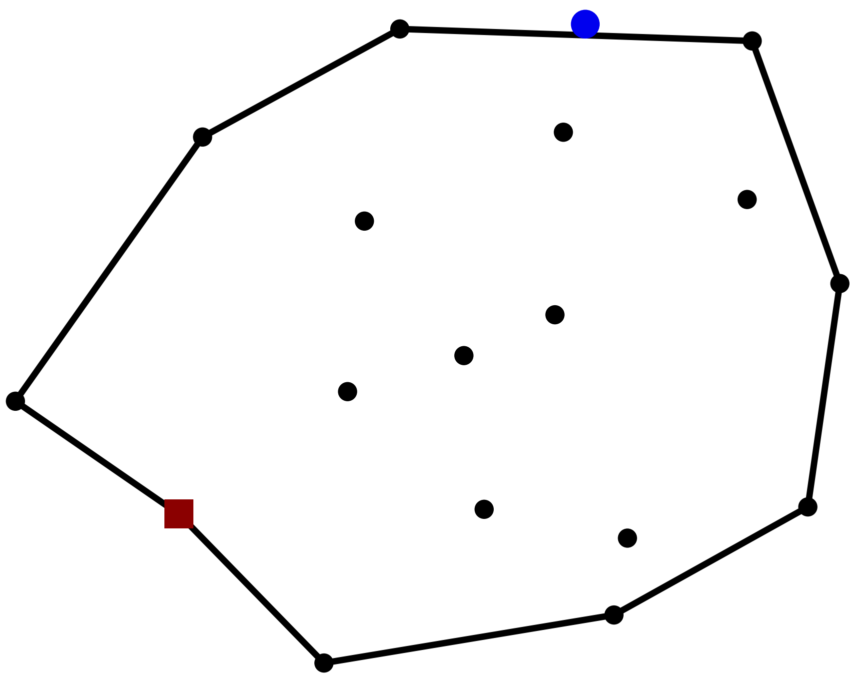
not peer-reviewed source

Naive floating point implementation of Real RAM:
false positives and false negatives

Practicalities

Collinear triples are common (grids!)

Almost collinear triples are also common!



P uniform random in $[0,1]^2$:

$$\mathbb{E}(\max_{p,q,r \in P} \sphericalangle pqr) = \pi - O(1/n^3)$$

↑ not peer-reviewed source

Naive floating point implementation of Real RAM:
false positives and false negatives

Good software libraries (e.g. CGAL) can protect you.

Chan's algorithm (1996)

Output-sensitive algorithm

Input size: natural lower bound for most problems

Output size: same! (but useless for decsion problems)

Output-sensitive algorithm

Input size: natural lower bound for most problems

Output size: same! (but useless for decsion problems)

Output-sensitive algorithm:

$\rightarrow$ An alg. that is faster if the output is small.

$\rightarrow$ running time expressed as function of input **and output** size

Output-sensitive algorithm

Input size: natural lower bound for most problems

Output size: same! (but useless for decsion problems)

Output-sensitive algorithm:
$\rightarrow$ An alg. that is faster if the output is small.
$\rightarrow$ running time expressed as function of input **and output** size

Example: enumerating spanning trees of a graph in

$$O(n^{n-2} \cdot n) \text{ vs. } O(\#\text{of trees} + n + |E|) \text{ time}$$

Output-sensitive algorithm

Input size: natural lower bound for most problems

Output size: same! (but useless for decsion problems)

Output-sensitive algorithm:
$\rightarrow$ An alg. that is faster if the output is small.
$\rightarrow$ running time expressed as function of input **and output** size

Example: enumerating spanning trees of a graph in

$$O(n^{n-2} \cdot n) \text{ vs. } O(\#\text{of trees} + n + |E|) \text{ time}$$

Here: h is $\#$ of convex hull vertices.

Can we get $O(n + h)$ as running time?

Output-sensitive algorithm

Input size: natural lower bound for most problems

Output size: same! (but useless for decsion problems)

Output-sensitive algorithm:
$\rightarrow$ An alg. that is faster if the output is small.
$\rightarrow$ running time expressed as function of input **and output** size

Example: enumerating spanning trees of a graph in

$$O(n^{n-2} \cdot n) \text{ vs. } O(\#\text{of trees} + n + |E|) \text{ time}$$

Here: h is $\#$ of convex hull vertices.

Can we get $O(n + h)$ as running time?

What about $O(n + h \log h)$?

Output-sensitive convex hull algorithms

- Kirkpatrick–Seidel ('86): $O(n \log h)$. Proven optimal!

Output-sensitive convex hull algorithms

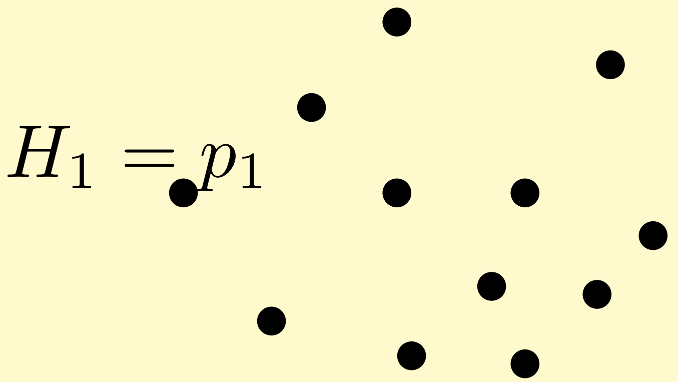
- Kirkpatrick–Seidel ('86): $O(n \log h)$. Proven optimal!
- Chan ('96): $O(n \log h)$, and also "works" in 3d.

Output-sensitive convex hull algorithms

- Kirkpatrick–Seidel ('86): $O(n \log h)$. Proven optimal!
- Chan ('96): $O(n \log h)$, and also "works" in 3d.

Based on the *Gift wrapping* algorithm:
start from p_1, and go around finding next hull vertex.

- Find p_1: $\to O(n)$

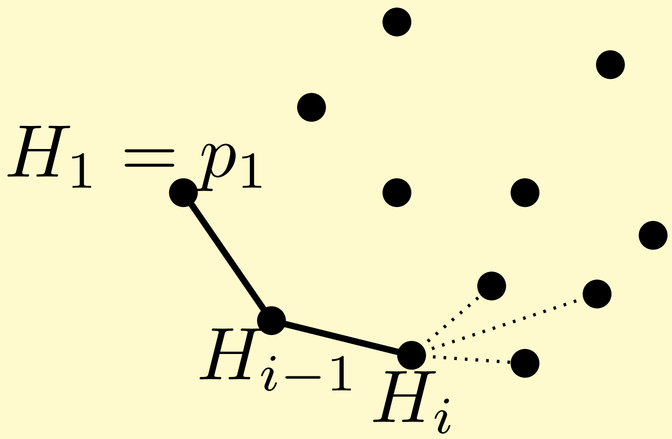


Output-sensitive convex hull algorithms

- Kirkpatrick–Seidel ('86): $O(n \log h)$. Proven optimal!
- Chan ('96): $O(n \log h)$, and also "works" in 3d.

Based on the *Gift wrapping* algorithm:
start from p_1, and go around finding next hull vertex.

- Find p_1: $\to O(n)$
- Given $H_1, \ldots, H_i$, find H_{i+1} s.t. $\sphericalangle H_{i-1} H_i H_{i+1}$ is maximized $\to O(n)$

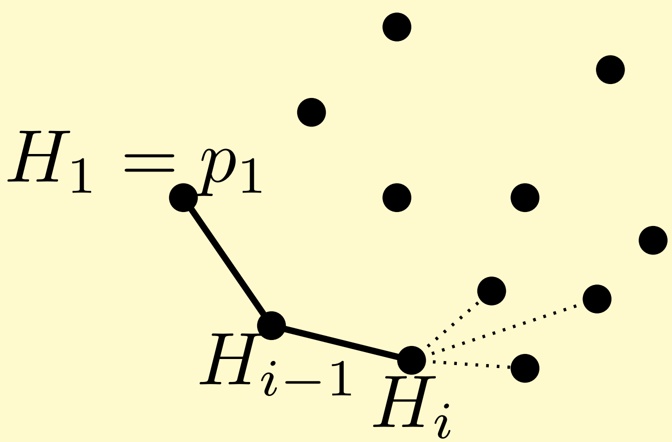


Output-sensitive convex hull algorithms

- Kirkpatrick–Seidel ('86): $O(n \log h)$. Proven optimal!
- Chan ('96): $O(n \log h)$, and also "works" in 3d.

Based on the *Gift wrapping* algorithm:
start from p_1, and go around finding next hull vertex.

- Find p_1: $\rightarrow O(n)$
- Given $H_1, \ldots, H_i$, find H_{i+1} s.t. $\sphericalangle H_{i-1} H_i H_{i+1}$ is maximized $\rightarrow O(n)$
- Repeated h times $\rightarrow O(nh)$ time in total



Chan's algorithm

Let $m \in \{1, \ldots, n\}$ be a parameter.

Group P into groups of size m

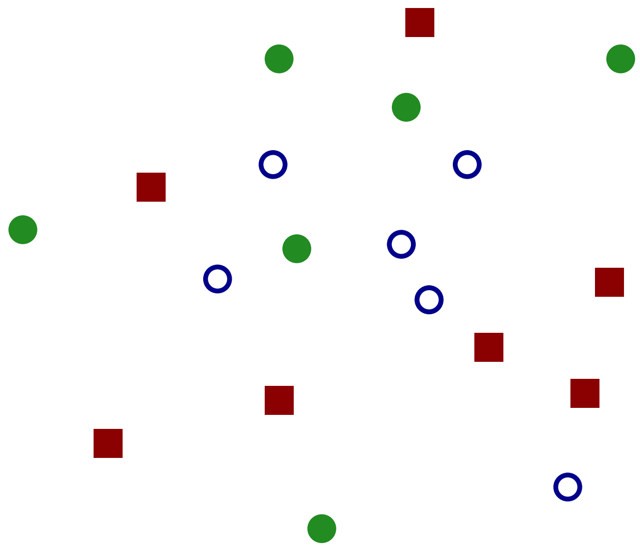
$$\Rightarrow \lceil n/m \rceil \text{ groups: } P_1, P_2, \ldots$$

Chan's algorithm

Let $m \in \{1, \ldots, n\}$ be a parameter.

Group P into groups of size m

$$\Rightarrow \lceil n/m \rceil \text{ groups: } P_1, P_2, \ldots$$



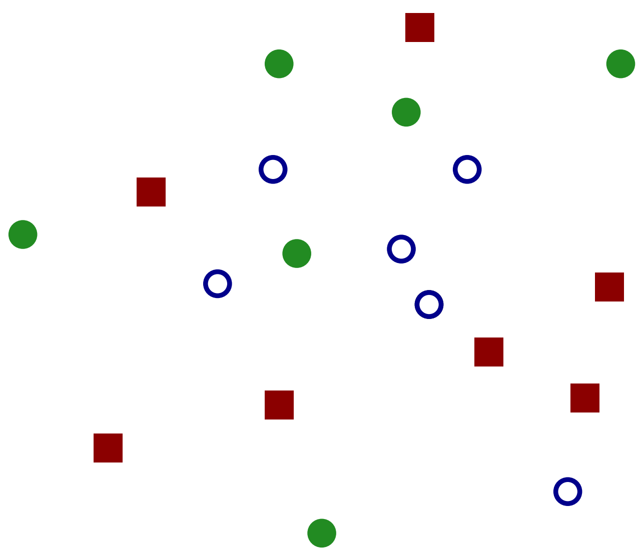
Chan's algorithm

Let $m \in \{1, \ldots, n\}$ be a parameter.

Group P into groups of size m

$$\Rightarrow \lceil n/m \rceil \text{ groups: } P_1, P_2, \ldots$$

Compute convex hull of each P_j with Graham's scan.



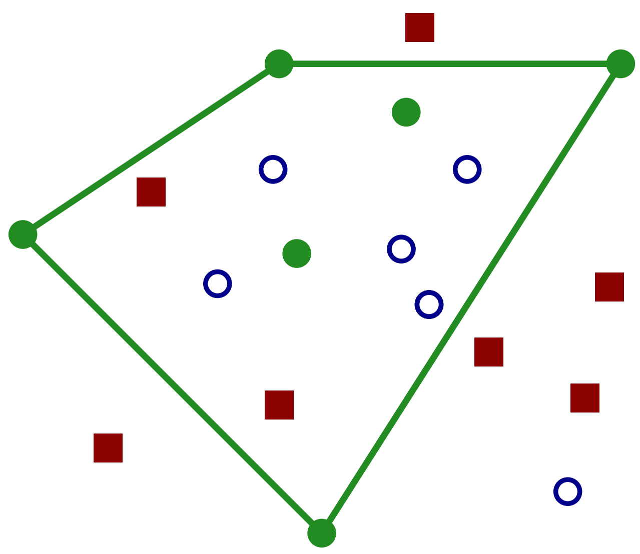
Chan's algorithm

Let $m \in \{1, \ldots, n\}$ be a parameter.

Group P into groups of size m

$$\Rightarrow \lceil n/m \rceil \text{ groups: } P_1, P_2, \ldots$$

Compute convex hull of each P_j with Graham's scan.



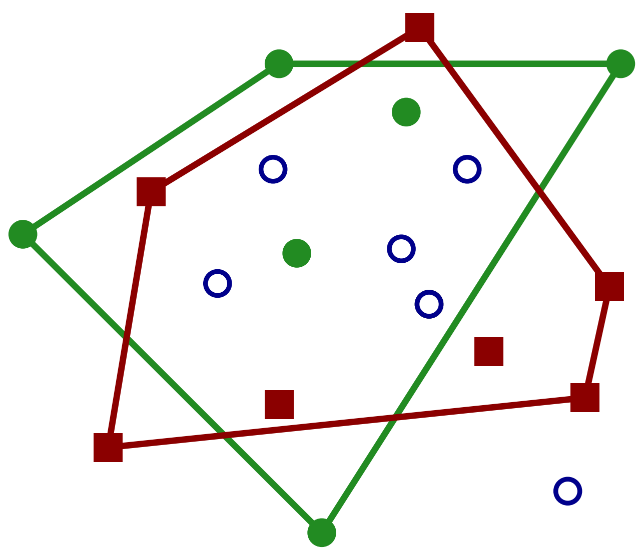
Chan's algorithm

Let $m \in \{1, \dots, n\}$ be a parameter.

Group P into groups of size m

$$\Rightarrow \lceil n/m \rceil \text{ groups: } P_1, P_2, \dots$$

Compute convex hull of each P_j with Graham's scan.



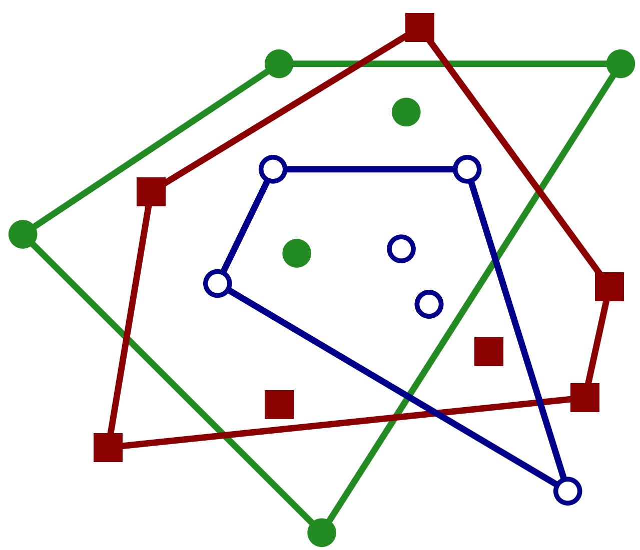
Chan's algorithm

Let $m \in \{1, \ldots, n\}$ be a parameter.

Group P into groups of size m

$$\Rightarrow \lceil n/m \rceil \text{ groups: } P_1, P_2, \ldots$$

Compute convex hull of each P_j with Graham's scan.



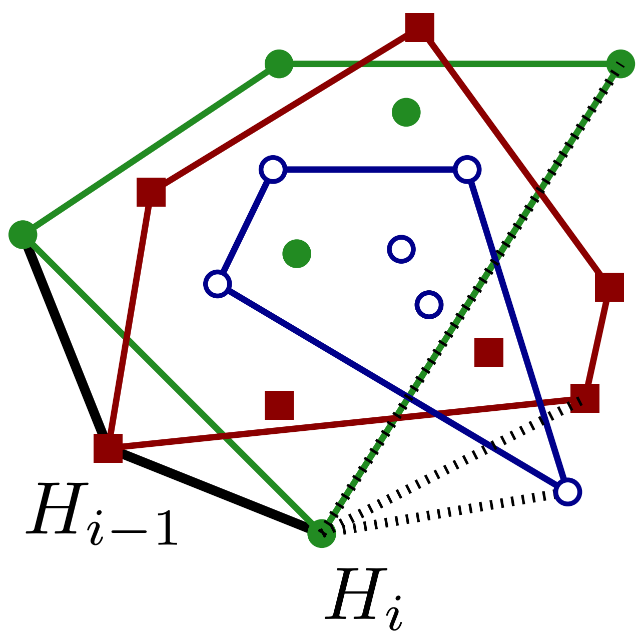
Chan's algorithm

Let $m \in \{1, \ldots, n\}$ be a parameter.

Group P into groups of size m

$$\Rightarrow \lceil n/m \rceil \text{ groups: } P_1, P_2, \ldots$$

Compute convex hull of each P_j with Graham's scan.



Gift wrapping:
Find largest angle $H_{i-1}H_iq^j$ with $q^j \in \mathrm{conv}(P_j)$ for each j, pick best

Chan's algorithm

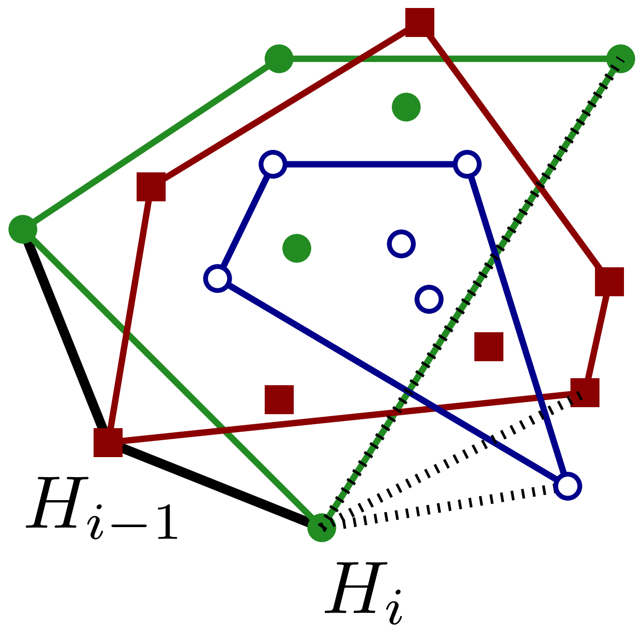
Let $m \in \{1, \ldots, n\}$ be a parameter.

Group P into groups of size m

$\Rightarrow \lceil n/m \rceil$ groups: $P_1, P_2, \ldots$

Compute convex hull of each P_j with Graham's scan.

$\lceil n/m \rceil \cdot O(m \log m)$



Gift wrapping:
Find largest angle $H_{i-1}H_iq^j$ with $q^j \in \mathrm{conv}(P_j)$ for each j, pick best

Chan's algorithm

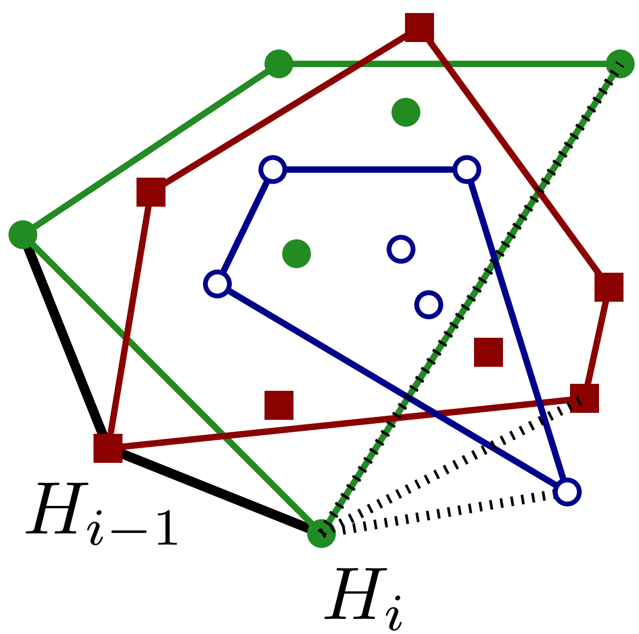
Let $m \in \{1, \ldots, n\}$ be a parameter.

Group P into groups of size m

$$\Rightarrow \lceil n/m \rceil \text{ groups: } P_1, P_2, \ldots$$

Compute convex hull of each P_j with Graham's scan.

$$\lceil n/m \rceil \cdot O(m \log m)$$



Gift wrapping:
Find largest angle $H_{i-1}H_iq^j$ with $q^j \in \mathrm{conv}(P_j)$ for each j, pick best

$$h\lceil n/m \rceil \cdot O(\log m)$$

Chan's algorithm

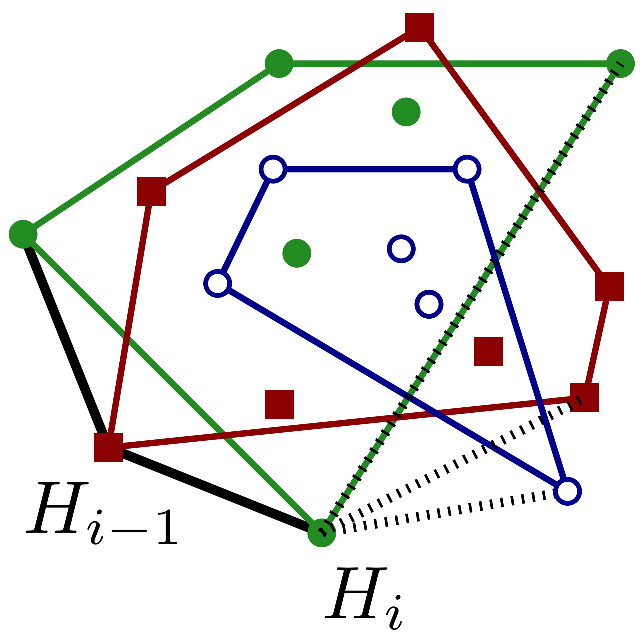
Let $m \in \{1, \ldots, n\}$ be a parameter.

Group P into groups of size m

$$\Rightarrow \lceil n/m \rceil \text{ groups: } P_1, P_2, \ldots$$

Compute convex hull of each P_j with Graham's scan.

$$\lceil n/m \rceil \cdot O(m \log m)$$



Gift wrapping:

Find largest angle $H_{i-1}H_iq^j$ with $q^j \in \text{conv}(P_j)$ for each j, pick best

$$h\lceil n/m \rceil \cdot \boxed{O(\log m)}$$

Time to find tangent of $\text{conv}(P_j)$

Chan's algorithm: running time and tangent finding

$$\lceil n/m \rceil \cdot O(m \log m) + h\lceil n/m \rceil \cdot O(\log m)$$

Chan's algorithm: running time and tangent finding

$$\lceil n/m \rceil \cdot O(m \log m) + h\lceil n/m \rceil \cdot O(\log m)$$

$$= O(n \log m + (h/m) n \log m)$$

Chan's algorithm: running time and tangent finding

$$\lceil n/m \rceil \cdot O(m \log m) + h\lceil n/m \rceil \cdot O(\log m)$$

$$= O(n \log m + (h/m) n \log m)$$

setting $m = h$:

$$= O(n \log h)$$

Chan's algorithm: running time and tangent finding

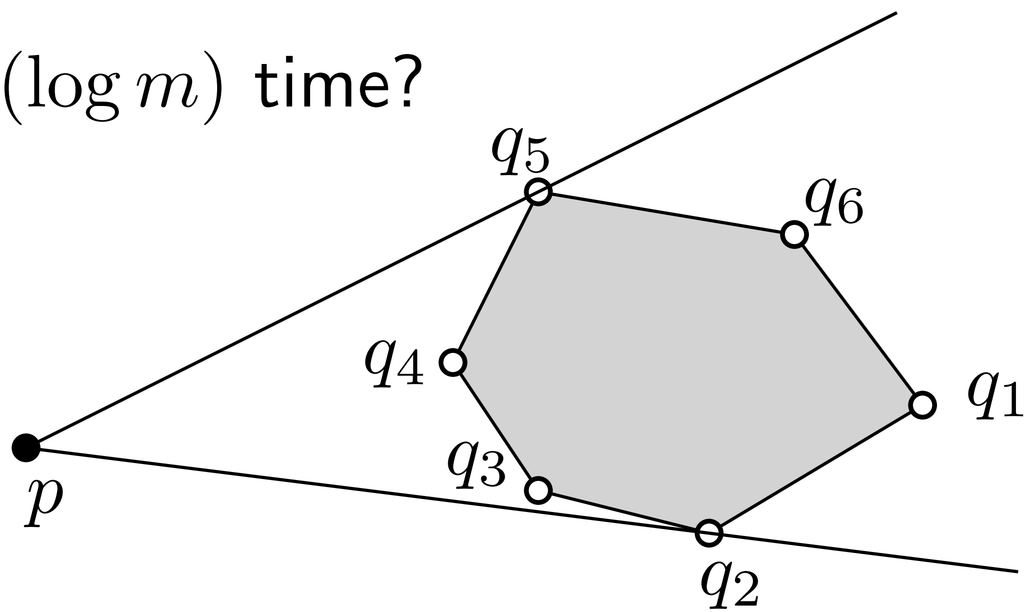
$$\lceil n/m \rceil \cdot O(m \log m) + h\lceil n/m \rceil \cdot O(\log m)$$

$$= O(n \log m + (h/m) n \log m)$$

setting $m = h$:

$$= O(n \log h)$$

Tangent finding in $O(\log m)$ time?



1. Set $m = h$

1. Set $m = h$
2. Let $P_j = \{p_{(j-1)m+1}, \ldots, p_{jm}\}$.

How to set $m = h$?

1. Set $m = \boxed{h}$?
2. Let $P_j = \{p_{(j-1)m+1}, \ldots, p_{jm}\}$.

How to set $m = h$?

1. Set $m = \boxed{h}$?
2. Let $P_j = \{p_{(j-1)m+1}, \ldots, p_{jm}\}$.

Idea 1: Let $m = 1, 2, \ldots, n$, run giftwrapping for m steps
Stop when gift is wrapped. (Then $m \geq h$ holds.)

$$\sum_{m=1}^{h} cn \log m = \Omega(hn)...$$

How to set $m = h$?

1. Set $m = \boxed{h}$?
2. Let $P_j = \{p_{(j-1)m+1}, \ldots, p_{jm}\}$.

Idea 1: Let $m = 1, 2, \ldots, n$, run giftwrapping for m steps

Too slow!

How to set $m = h$?

1. Set $m = \boxed{h}$?
2. Let $P_j = \{p_{(j-1)m+1}, \ldots, p_{jm}\}$.

Idea 1: Let $m = 1, 2, \ldots, n$, run giftwrapping for m steps

Idea 2: Let $m = 2^1, 2^2, 2^3, \ldots, 2^{\log n}$, run wrapping for m steps

$$\sum_{i=1}^{\lceil \log h \rceil} cn \log 2^i = \sum_{i=1}^{\lceil \log h \rceil} cni = \Theta(n \log^2 h) \ldots$$

How to set $m = h$?

1. Set $m = \boxed{h}$?
2. Let $P_j = \{p_{(j-1)m+1}, \ldots, p_{jm}\}$.

Idea 1: Let $m = 1, 2, \ldots, n$, run giftwrapping for m steps

Idea 2: Let $m = 2^1, 2^2, 2^3, \ldots, 2^{\log n}$, run wrapping for m steps

Too slow!

How to set $m = h$?

1. Set $m = \boxed{h}$?
2. Let $P_j = \{p_{(j-1)m+1}, \ldots, p_{jm}\}$.

Idea 1: Let $m = 1, 2, \ldots, n$, run giftwrapping for m steps

Idea 2: Let $m = 2^1, 2^2, 2^3, \ldots, 2^{\log n}$, run wrapping for m steps

Idea 3: Let $m = 2^{2^0}, 2^{2^1}, 2^{2^2}, \ldots, 2^{2^{\lceil \log \log n \rceil}}$, do m wrap-steps

$$\sum_{i=0}^{\lceil \log \log h \rceil} cn \log 2^{2^i} = \sum_{i=0}^{\lceil \log \log h \rceil} cn 2^i$$

$$= cn(2^{\lceil \log \log h \rceil + 1} - 1) = O(n \log h)$$

Chan's algorithm: Recap

for $i = 1$ to $\lceil \log \log n \rceil$ **do**
  $m = 2^{2^i}$
  for $j = 1$ to $\lceil n/m \rceil$ **do**
    Create group P_j
    $(q_1^j, q_2^j, \ldots,) = Graham(P_j)$
  H_1 =leftmost point in P
  for $s = 2$ to m **do**
    for $j = 1$ to $\lceil n/m \rceil$ **do**
      $q^j = TangentFind(H_{s-1}, (q_1^j, q_2^j, \ldots))$
    $H_s =$ point q^j maximizing $\sphericalangle H_{s-2} H_{s-1} q^j$
    if $H_s = H_1$ **then return** $(H_1, \ldots, H_{s-1})$