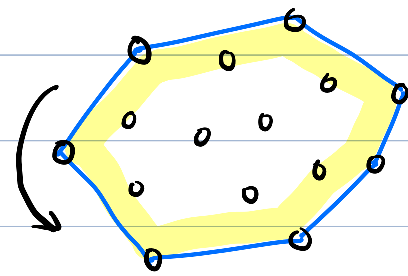
CMSC 754 - Computational Geometry
Lecture 3: Convex Hulls (Continued)

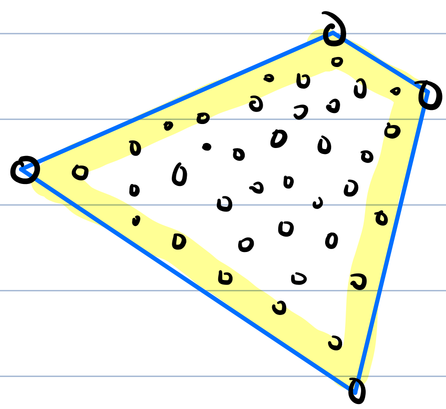
Recap:

- Given a pt. set $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^2$ compute conv(P) - smallest convex set containing P.
- Graham's Scan - $O(n \log n)$ time
- Output: Cyclic sequence of hull vertices



This Lecture:

- Can we beat $O(n \log n)$ time?
  - → No. $\Omega(n \log n)$ lower bound
- What if very few hull vertices? $h \ll n$
  - Jarvis March - $O(n \cdot h)$
  - Chan's Algorithm - $O(n \log h)$
  - Output sensitive algorithm



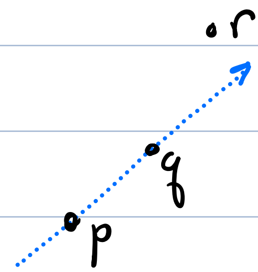
Lower bound for convex hulls:

Conv: Given a set P of n pts in $\mathbb{R}^2$, compute the vertices of $\text{conv}(P)$ in cyclic order.

Def: An algorithm is comparison-based if its decisions are based on the sign of a fixed-degree polynomial function of inputs. (Algebraic decision tree model)

Almost all geometric primitives satisfy:

E.g. if($\langle p,q,r \rangle$ form a left-hand turn)



$$\equiv \text{if}\left(\det \begin{pmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{pmatrix} > 0 \right)$$

$$\equiv \text{if}(f(p_x, p_y, q_x, q_y, r_x, r_y) > 0)$$

where:

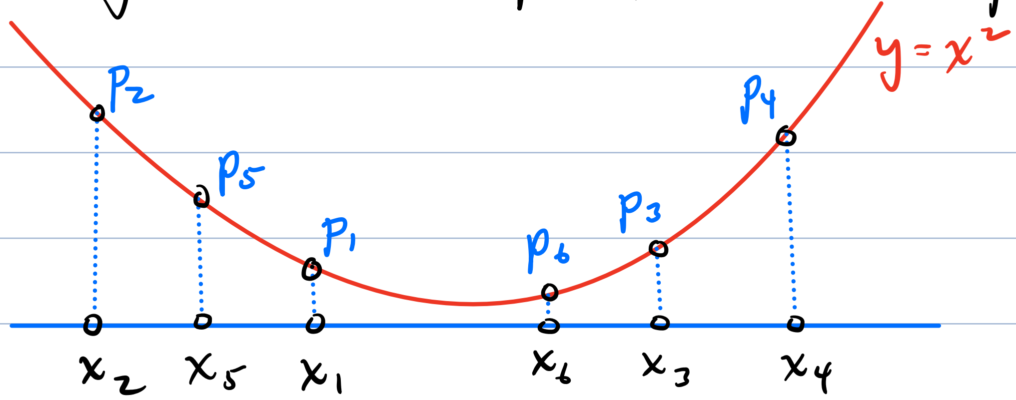
$$f(\ldots) = (q_x r_y - q_y r_x) - (p_x r_y - p_y r_x) + (p_x q_y - p_y q_x)$$

A polynomial of degree 2

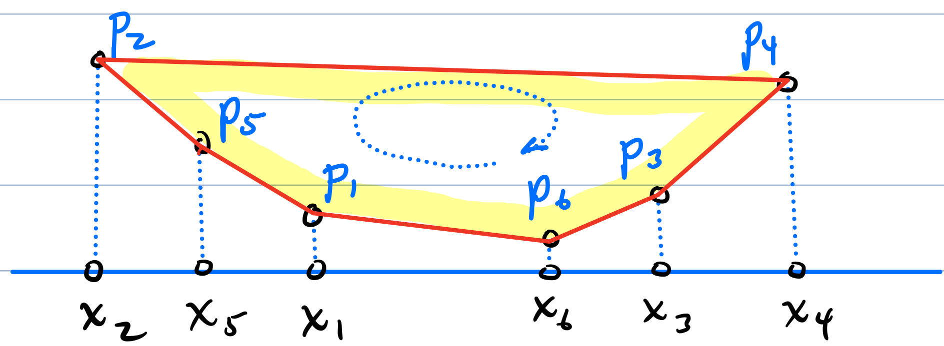
Theorem: Assuming a comparison-based algorithm, conv has a worst-case lower bound of $\Omega(n \log n)$

Proof: We will use the well-know fact that any comparison-based alg. for sorting reqs. $\Omega(n \log n)$ time in worst case.

We'll reduce sorting to conv. Given set $X = \{x_1, \dots, x_n\}$ to be sorted in $O(n)$ time we generate $P = \{p_1, \dots, p_n\}$ where $p_i = (x_i, x_i^2)$



If we compute conv(P), the vertices appear in sorted order of X, up to reversal and adjusting starting point $\leftarrow O(n)$ time



Letting $T(n)$ denote the time to compute $\text{conv}(P)$, up to constant factors, we can sort X in time

$$n + T(n) + n, \text{ which must be } \geq c \cdot n \log n$$

compute P from X ↗ ↑ reorient output

$$\Rightarrow T(n) \geq c \cdot n \log n - 2n \quad \Rightarrow T(n) = \Omega(n \log n)$$

$\square$

Obs: This exploits the fact that output is sorted cyclically. What if not?

Theorem: $\boxed{\text{Assuming a comparison-based algorithm determining whether } \text{conv}(P) \text{ has } h \text{ distinct vertices requires } \Omega(n \log h) \text{ time.}}$

$\Rightarrow$ Just counting vertices reqs. log factor.

(See latex lecture notes for proof)

Output Sensitivity: Algorithm's running time depends on output size

→ Is $O(n \log h)$ possible?

We'll do this in two steps...

Jarvis March: An $O(nh)$ algorithm

Idea: Compute **any** one vertex of hull $\rightarrow v_1$

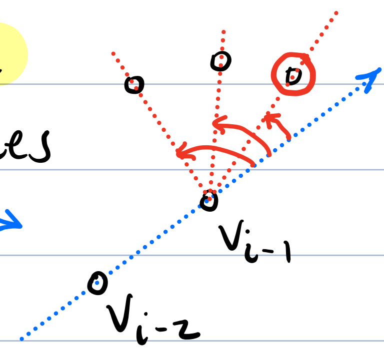
for $i = 2, 3, \ldots$

compute **next** vertex v_i on hull

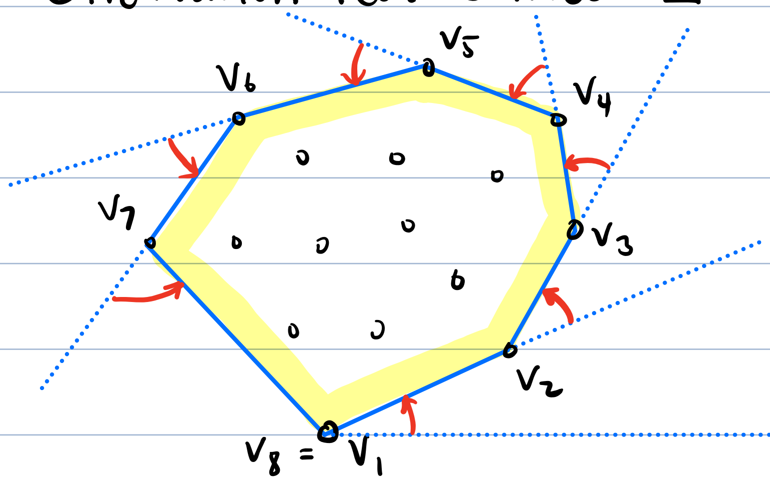
if $(v_i == v_1)$ return $\langle v_1, \ldots, v_{i-1} \rangle$

v_1? Point of P with min y-coordinate

next vertex? The point of P that minimizes **turn angle** w.r.t. prior two vertices



[This doesn't require trig. Orientation test suffices]



Correctness: Easy

Running time: Compute v_1 - $O(n)$

Compute v_i - $O(n) \leftarrow$ Repeat h times

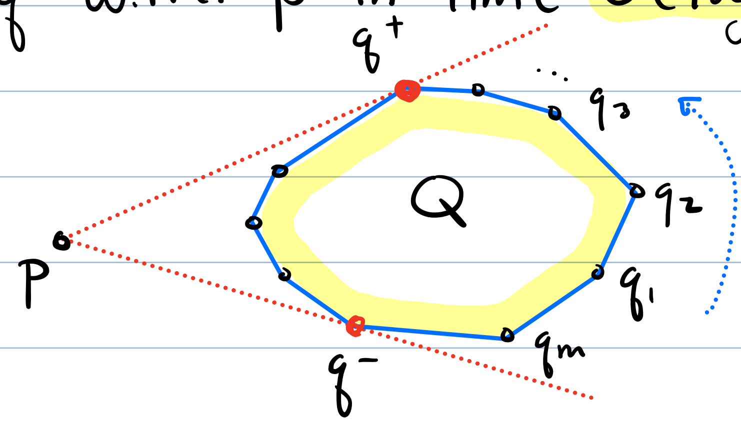
Total: $O((h+1)n) = O(h \cdot n)$

Chan's Algorithm: An $O(n \log h)$ algorithm

- Optimal w.r.t. input size n + output size h
- Combines two slow algorithms (Graham + Jarvis) to make faster algorithm
- Chicken + Egg: Algorithm needs to know value of h - How is this possible?

Utility Function: (used later)

Given a convex polygon Q given as a cyclic sequence of m vertices $\langle q_1, \dots, q_m \rangle$ and $p \notin Q$, can compute tangent vertices q^- + q^+ w.r.t. p in time $O(\log m)$



How? Exercise

Hint: Variant of binary search

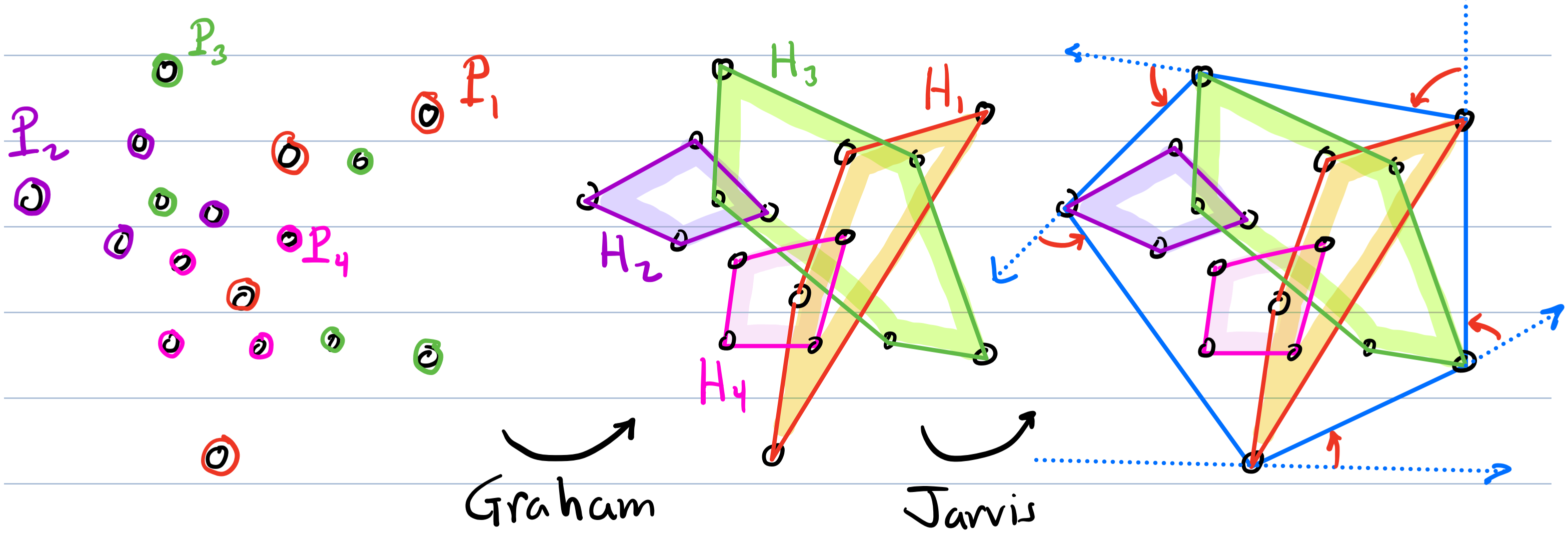
How to achieve $O(n \log h)$?

- Can't sort any set of size $\gg h$
- Guess the hull size – h^*
- Partition P into $\lceil n/h^* \rceil$ groups, each of size $\leq h^*$
  $\rightarrow P_1, \dots, P_k$, $k = O(n/h^*)$ $\longrightarrow O(n)$
- Run Graham on each group forming k "mini-hulls" $H_1, \dots, H_k$ $\longrightarrow O(k \cdot h^* \log h^*)$
  $= O(n \log h^*)$
  - If we guess right $(h^* = h) \rightarrow O(n \log h)$

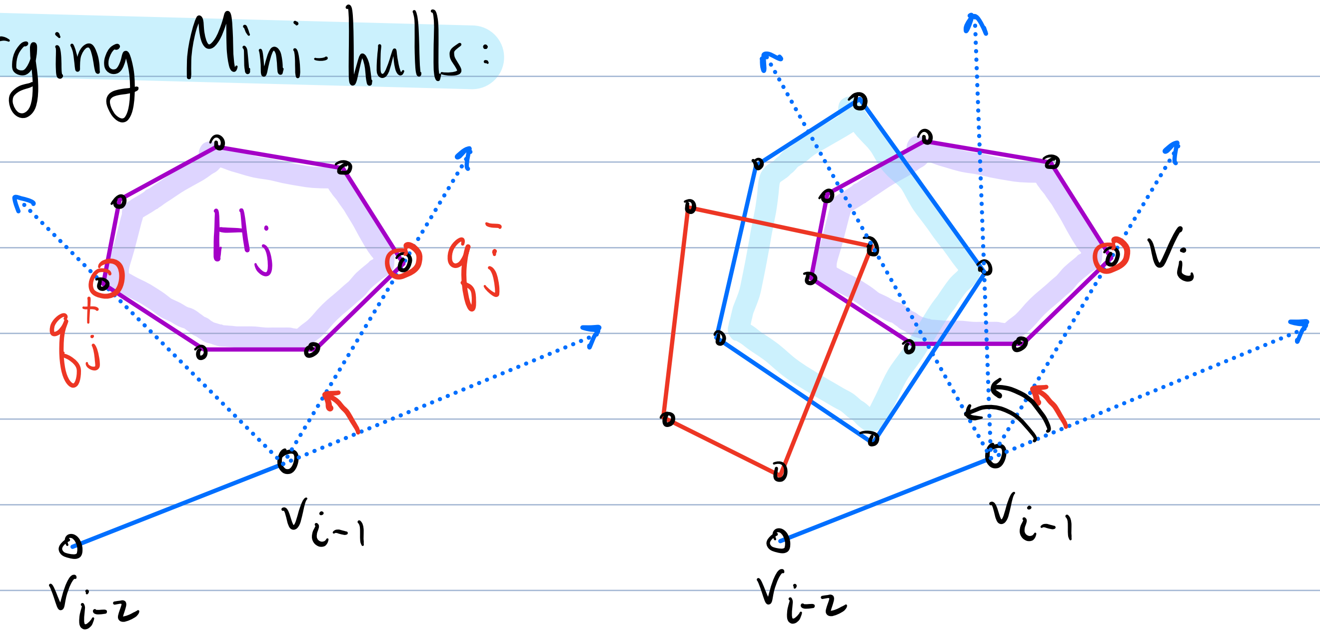
- Run Jarvis, but treat each mini-hull as a "fat point"
  - use the utility function to compute turning angles

Example: Suppose $k = 5$

Partition Mini-Hulls Merge



Merging Mini-hulls:



- By utility function, compute tangents q_j^- & q_j^+ for each H_j in time $O(\log h^*)$
- Compute all tangents in time $O(k \cdot \log h^*)$
- $v_i \leftarrow$ tangent with smallest turning angle
- Terminates after h iterations

$\Rightarrow$ Total merge time: $O(h \cdot k \cdot \log h^*)$

$\rightarrow$ If we guess right ($h^* = h$) then

$$O\left(h^*\left(\frac{n}{h^*}\right)\log h^*\right) = O(n \log h^*)$$
$$= O(n \log h)$$

Summary: If we guess correctly ($h^* = h$) this computes conv(P) in time $O(n \log h)$.

How to guess h?

Mini-hull Phase: $O(n \log h^*)$

Merge Phase: $O(n \frac{h}{h^*} \log h^*)$

If $h^* > h$ $\Rightarrow$ Mini-hull phase is too slow

Note: Can tolerate a polynomial error. E.g. if $h \leq h^* \leq h^2$

$$\Rightarrow O(n \log h^*) = O(n \log(h^2)) = O(2 \cdot n \log h) = O(n \log h) \text{ ok.}$$

If $h^* < h$ $\Rightarrow$ Merge phase too slow

- If Jarvis finds more than h^* hull pts - stop + return fail status

$\Rightarrow O(n \log h^*)$ time

Strategy:

Start small and increase until success

Arithmetic: $h^* = 3, 4, 5, \ldots$ way too slow $\rightarrow O(n \cdot h \cdot \log h)$

Exponential: $h^* = 4, 8, 16, \ldots, 2^i$ better $\rightarrow O(n \log^2 h)$

Double Exponential: $h^* = 4, 16, 256, \ldots 2^{2^i}$ best!

Note: $h_i^* = 2^{2^i}$ $\quad h_i^* \leftarrow (h_{i-1}^*)^2$

Final Algorithm:

Chan Hull(P):

$h^* = 2$

repeat

  $h^* \leftarrow (h^*)^2$

  $(status, V) \leftarrow conditionalHull(P, h^*)$

until (status == success)

return V

Correctness: Already explained

Time:

- Running time per iteration $O(n \log h^*)$
- $h^*_i = 2^{2^i}$
- Stops when $h^* \geq h$

$$2^{2^i} \geq h \Rightarrow i = \lceil \lg \lg h \rceil \text{ iterations}$$

- **Total time:** [up to constants]

$$\sum_{i=1}^{\lg \lg h} n \cdot \lg(2^{2^i}) = n \sum_{i=1}^{\lg \lg h} 2^i$$

$$\leq 2n \, 2^{\lg \lg h} \quad \left[\begin{array}{c}\text{Geom} \\ \text{series}\end{array}\right]$$

$$= 2n \lg h$$

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