Recap:
- Given a pt. set $P = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^2$ compute $\text{conv}(P)$ - 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?
  - $\rightarrow$ No. $\Omega(n \log n)$ lower bound
- What if very few hull vertices? $h \ll n$
  - **Jarvis March** - $O(nh)$
  - **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

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

$$\iff f(p_x, p_y, q_x, q_y, r_x, r_y) > 0$$

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, \( \text{conv} \) has a worst-case lower bound of \( \Omega(n \log n) \).

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

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

If we compute \( \text{conv}(P) \), the vertices appear in sorted order of \( X \), up to reversal and adjusting starting point \(- O(n) \) time.
Letting $T(n)$ denote the time to compute $\text{conv}(P)$, up to constant factors, we can sort $X$ in time $n + T(n) + n$, which must be $\geq c \cdot n \log n$ reorient output

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

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

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

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

(See latex lecture notes for proof)

**Output Sensitivity:** Algorithm’s running time depends on output size $\Rightarrow O(n \log h)$ possible?

We’ll do this in two steps...
**Jarvis March**: An $O(nh)$ algorithm

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

for $i = 2, 3, ...$

compute next vertex $v_i$ on hull

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

$v_i$? 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 $\langle q_1, \ldots, q_m \rangle$ and $p \not\in Q$, 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 $\leq h^*$
  \[ \rightarrow P_1, \ldots, P_k, k = O(n/h^*) \rightarrow O(n) \]
- Run Graham on each group forming $k$ mini-hulls $H_1, \ldots, H_k$
  \[ \rightarrow O(k \cdot h^* \log h^*) = O(n \log h^*) \]
- 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 $k = 5$
Merging Mini-hulls:

- By utility function, compute tangents \( q^+ + q^- \) 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.

\[ \Rightarrow \text{Total merge time: } O(h \cdot k \cdot \log h^*) \]

- If we guess right \( (h^* = h) \) then
  \[ O(h^*(\frac{n}{h^*}) \log h^*) = O(n \log h^*) = O(n \log h) \]

Summary: If we guess correctly \( (h^* = h) \), this computes \( \text{conv}(P) \) in time \( O(n \log h) \).
How to guess $h$?

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

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

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

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

$\Rightarrow O(n \log h^*) = O(n \log(h^2))$

$= O(2 \cdot n \log h)$

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

**If** $h^* < h$ ⇒ **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: $h^*_i = 2^i \quad h^*_i \leftarrow (h^*_i)^2$
Final Algorithm: Chan Hull (P):

\[ h^* = 2 \]

repeat

\[ h^* \leftarrow (h^*)^2 \]

\( (\text{status, } V) \leftarrow \text{conditional Hull (P, } h^* \) \)

until (status == success)

return \( V \)

Correctness: Already explained

Time:

- Running time per iteration \( O(n \log h^*) \)
- \( h^*_i = 2^{2^i} \)
- Stops when \( h^*_i \geq h \)
  \[ 2^{2^i} \geq h \Rightarrow i = \left\lceil \log \log h \right\rceil \text{ iterations} \]
- Total time: \([\text{up to constants}]\)

\[ \sum_{i=1}^{\lceil \log \log h \rceil} n \cdot \log \left( 2^{2^i} \right) = n \sum_{i=1}^{\lceil \log \log h \rceil} 2^i \leq 2n \cdot 2^{\log \log h} \text{ [Geom series]} \]

\[ = 2n \cdot 2^{\log \log h} \]

\[ = 2n \log h \]

\[ = O(n \log h) \] ☻