Area of ** \( A \)** is given a determinant and the left/right turn is given by sign 

\[ \theta, \quad \theta_1 > \theta_2 \]

Graham scan takes sorting + \( O(n) \) line Jarvis march ~ \( O(n \times h) \) steps 

\( h \) = # boundary vertices

The best possible runtime for output can be achieved as \( O(n \times \log h) \)

"Output sensitive" algorithms
Can we do better than $O(n \log n)$?

Consider sorting a given set $S$ of $n$ numbers $x_1, x_2, \ldots, x_n$.

$S \rightarrow S' \quad (x_1, x_2^2) \quad (x_2, x_4^3) \ldots \quad (x_n, x_n^2)$

$CH(S')$

All points in $S'$ will be on the boundary.

By projecting back to the $x$ axis, we can deduce the sorted set of points in $S$.

Total time for sorting $S$: $O(n) + T(n)$

$S$ can be sorted in $O(n) + T(n)$ time.

$\text{Sort}(n) = \Omega(n \log n)$
Theorem: \( O(n) + T(n) \) must be \( \Omega(n \log n) \)

\[ \Rightarrow T(n) \text{ must be } \Omega(n \log n) \]

Lower bound for sorting is \( \Omega(n \log n) \)

In general, for problems \( \pi_1 \) and \( \pi_2 \)

we say \( \pi_1 \) is reducible to \( \pi_2 \)

\[ \pi_1 \leq \pi_2 \text{ if } \pi_1 \preceq \pi_2 \]

If any algorithm for \( \pi_2 \) can be used to also solve \( \pi_1 \),

Observation 1: An upper bound for \( \pi_2 \) implies an upper bound for \( \pi_1 \)

2. Any lower bound for \( \pi_1 \) implies a similar lower bound for \( \pi_2 \)

We showed sorting \( \leq \) convex hull

The \# boundary faces in \( d \)-dim convex hull can be \( n^{\left\lfloor d/2 \right\rfloor} \)

(upper bound - known)
Binary search in the $x$ direction will lead us to the vertical strip containing the query location $q = (x^*, y^*)$.

By doing a binary search within the vertical strip, we can locate the query.

Total query time: $O(\log n)$

What about the data structure?
The total size of the data structure for the second phase is

\[ \eta_1 + \eta_2 + \eta_3 - \eta_k \]

where \( \eta_i \) is the \# of segment in slab \( i \).

This can be \( O(\eta_i) \) for worst case inputs.

---

**PLANAR POINT LOCATION**

---

A planar map can be thought of as a planar graph.

A graph that can be drawn on the plane with edge crossings.

\[ a \rightarrow d \rightarrow c \rightarrow a \rightarrow b \rightarrow c \rightarrow b \rightarrow a \rightarrow d \rightarrow c \]
\[ e : \text{edges} \quad v : \text{vertices} \quad f : \text{faces} \]

\[ v - e + f = 2 \quad \text{Euler's result} \]

The maximum number of faces in a planar graph is given by the Delta theorem of a planar graph.

In a Delta planar region

\[ 2e = 3f \quad \Rightarrow \quad f = \frac{2}{3}e \]

Substitute into \( (1) \)

\[ v - e + \frac{2}{3}e = 2 \]

\[ e \leq 3(v - 2) \leq 3v \]

\[ \boxed{e = 3(v - 2) < 3v} \]
(Wlog, let us consider a planar subdivision where every face (including the outer face) is a triangle.

Step 1. Eliminate "many" non-adjacent vertices.

Step 2. Re-triangulate some of these regions.

Step 3. Keep track of how the new $\Delta$s intersect with the previous (eliminated) $\Delta$s.

\[ T^0_1, T^0_2, T^0_3, T^0_4 \]

\[ T^h_1, T^h_2, T^h_3, T^h_4 \]
Repeat steps 1 to 3 until # colors is less than 25

For point location queries we first find the location with respect to the 25 point graph, perhaps by brute force.

We refine the search over the different levels we have created until we know the location on the original graph.