# Algorithms and data structures

Amin Coja-Oghlan

LFCS

# The convex hull

## Definition

- A set $C \subset \mathbf{R}^2$ is convex if for all $p, q \in C$ the line segment $\overline{pq}$ lies in $C$.
- The convex hull of a set $S \subset \mathbf{R}^2$ is the smallest convex set $C$ that contains $S$.

# The convex hull

## Definition

- A set $C \subset \mathbf{R}^2$ is convex if for all $p, q \in C$ the line segment $\overline{pq}$ lies in $C$.
- The convex hull of a set $S \subset \mathbf{R}^2$ is the smallest convex set $C$ that contains $S$.

## Observation

The convex hull of a *finite* set $S$ is a convex *polygon* whose vertices (ie "corners") are elements of $S$.

# The convex hull (ctd.)

## Computational problem

- Input: a finite set $S \subset \mathbf{R}^2$.
- Output: the vertices of the convex hull of $S$ in counterclockwise order.

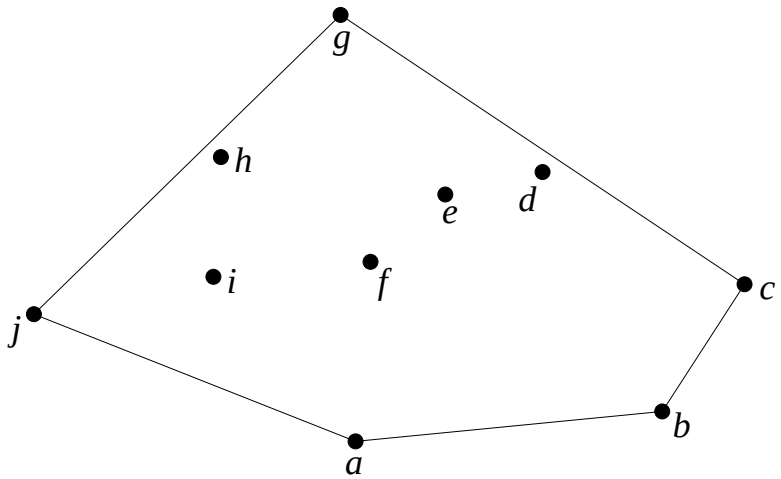
# The convex hull (ctd.)

## Computational problem

- Input: a finite set $S \subset \mathbf{R}^2$.
- Output: the vertices of the convex hull of $S$ in counterclockwise order.

## Example

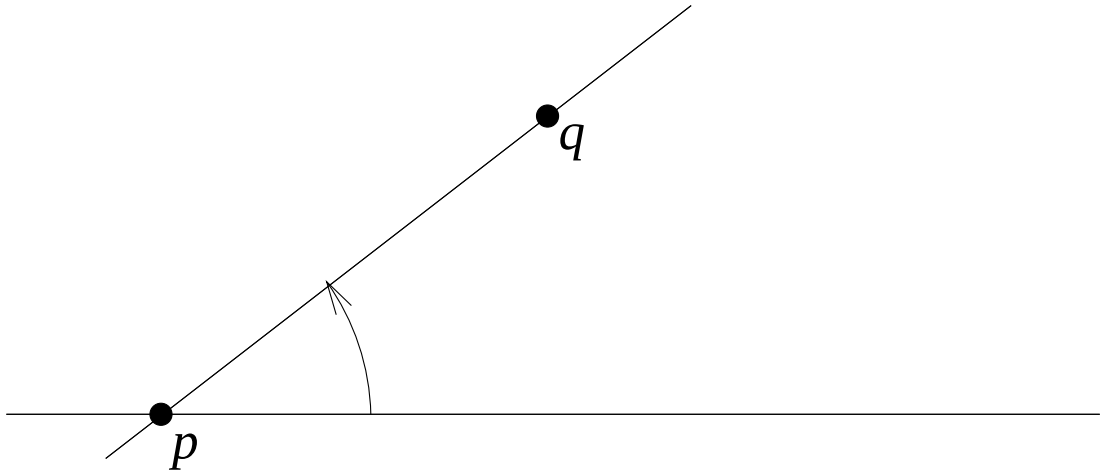
Desired output: $a, b, c, g, j$.



# Polar angles

## Definition

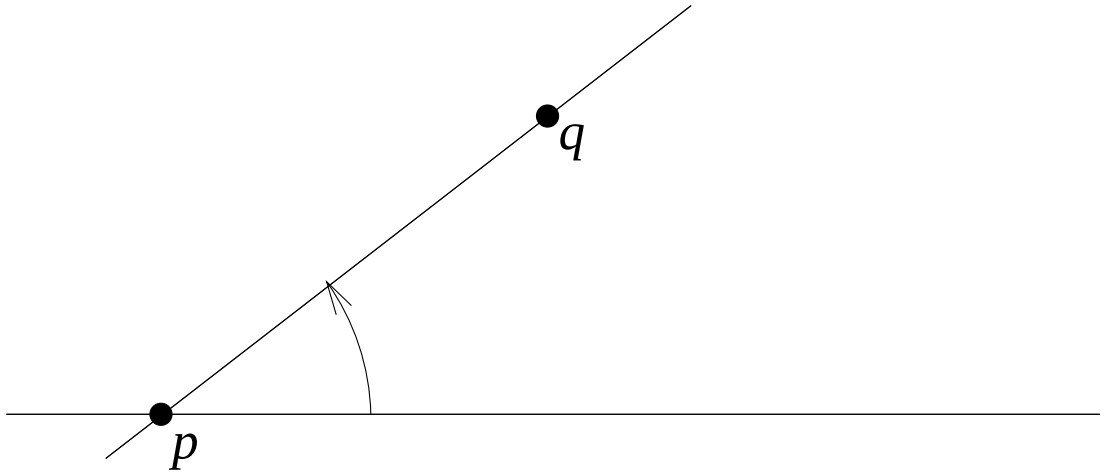
The polar angle of $q \in \mathbf{R}^2$ w.r.t. $p \in \mathbf{R}^2$ is the angle between the horizontal line through $p$ and the line through $p$ and $q$.



# Polar angles

## Definition

The polar angle of $q \in \mathbf{R}^2$ w.r.t. $p \in \mathbf{R}^2$ is the angle between the horizontal line through $p$ and the line through $p$ and $q$.




## Lemma

There is an algorithm $\mathcal{A}$ that sorts given points $p_0, p_1 \ldots, p_n$ by non-decreasing polar angle w.r.t. $p_0$ in time $O(n \log n)$.

# Graham's scan

## Idea

- Let $p_0$ bethe bottom-most point.
- Start walking around the points in the order of increasing polar angles.
- As long as you turn left, keep going.

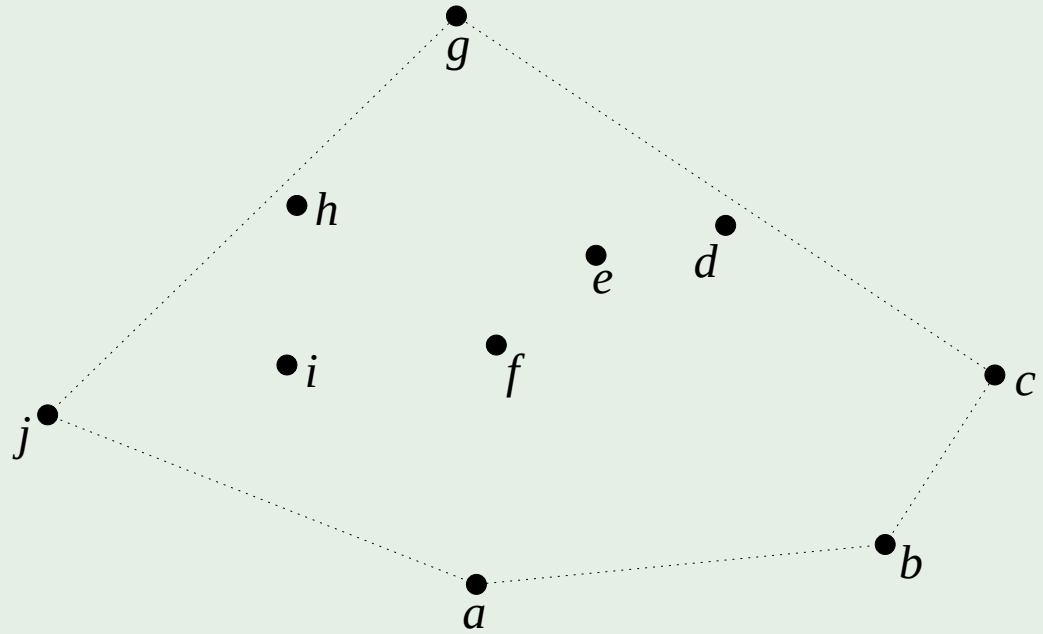
# Graham's scan

## Idea

- Let $p_0$ bethe bottom-most point.
- Start walking around the points in the order of increasing polar angles.
- As long as you turn left, keep going.
- If you have to turn right, discard the current point and step back to the previous one.
- Repeat until you can turn left to the next point.
- The points that remain are the vertices of the convex hull.

# Graham's scan (ctd.)

## Example



# Graham's scan (ctd.)

## Algorithm `Graham`($Q$)

*Input:* a set $Q$ of points. *Output:* the vertices of the convex hull.

1. Let $p_0$ be the point in $Q$ with minimum $y$ coordinate. [In case of a tie, take the leftmost point.]
2. Sort the points in $Q \setminus \{p_0\}$ by non-decreasing polar angles with respect to $p_0$. [If several points have the same polar angle, discard all but the one farthest away from $p_0$.]
3. Let $\langle p_1, \ldots, p_m \rangle$ be the resulting list
4. If $m \leq 1$ then return $\langle p_0, \ldots, p_m \rangle$
5. Initialise a stack $S$.
6. $S$.`push`($p_0$), $S$.`push`($p_1$), $S$.`push`($p_2$).

# Graham's scan (ctd.)

## Algorithm `Graham`($Q$)

*Input:* a set $Q$ of points. *Output:* the vertices of the convex hull.

7 For $i = 3, \ldots, m$ do
8  While the angle formed by the topmost two elements of $S$ and $p_i$ does not make a left turn do
9  $S$.pop.
10  $S$.`push`($p_i$).
11 Return $S$

# Graham's scan (ctd.)

## Running time

- Letting $n = |Q|$, we have $m \leq n$.
- Lines 3–6 and 11 require time $O(1)$.
- Line 1 requires $O(n)$.
- Line 2 needs $O(n \log n)$.
- The for-loop gets iterates $m - 2 \leq n$ times.
- Every element enters the stack at most once. Therefore, the inner loop takes time $O(n)$.
- Hence, the total running time is $O(n \log n)$.

# Graham's scan (ctd.)

## Proof of correctness

- The vertices of the convex hull are clearly among $p_0, \ldots, p_m$.
- If $m \leq 1$, there is nothing to show. Thus, assume $m \geq 2$.
- Let $C_i$ be the convex hull of $p_0, \ldots, p_i$.
- After line 6 the vertices on $S$ are the vertices of $C_2$ in clockwise order.

# Graham's scan (ctd.)

## Proof of correctness

- The vertices of the convex hull are clearly among $p_0, \ldots, p_m$.
- If $m \leq 1$, there is nothing to show. Thus, assume $m \geq 2$.
- Let $C_i$ be the convex hull of $p_0, \ldots, p_i$.
- After line 6 the vertices on $S$ are the vertices of $C_2$ in clockwise order.

## Claim

After the $i$'th iteration of the "for" loop, the points on $S$ are the vertices of $C_i$ in clockwise order.

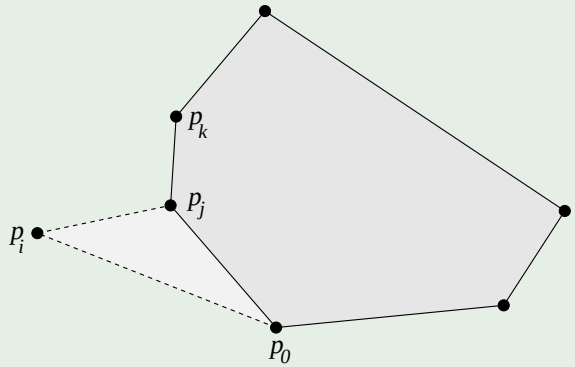
# Graham's scan (ctd.)

## Claim

After the $i$'th iteration of the "for" loop, the points on $S$ are the vertices of $C_i$ in clockwise order ($i \geq 3$).

## Proof (ctd.)

- Suppose that the claim holds for $i - 1$.
- The polar angle of $p_i$ is bigger than that of $p_{i-1}$. Therefore, $p_0, p_{i-1}, p_i$ for a triangle that is not contained in $C_{i-1}$.




- In particular, $p_i$ is a vertex of $C_i$.

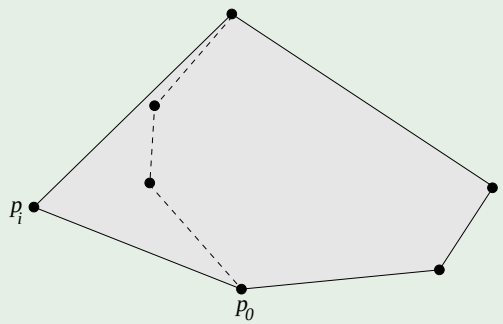
# Graham's scan (ctd.)

## Claim

After the $i$'th iteration of the "for" loop, the points on $S$ are the vertices of $C_i$ in clockwise order ($i \geq 3$).

## Proof (ctd.)

- Let $p_j$ and $p_k$ be the two topmost elements of $S$.
- If the angle formed by $p_k, p_j, p_i$ makes a right turn, then the triangle $p_k, p_j, p_i$ is contained in the triangle $p_k, p_0, p_i$.
- Therefore, $p_j$ is not a vertex of $C_i$.
- Thus, it is correct to remove $p_j$ from $S$ in line 9.



# Graham's scan (ctd.)

## Claim

After the $i$'th iteration of the "for" loop, the points on $S$ are the vertices of $C_i$ in clockwise order ($i \geq 3$).

## Proof (ctd.)

- By contrast, if the angle formed by $p_k, p_j, p_i$ makes a left turn, then the triangle $p_k, p_j, p_i$ is *not* contained in the triangle $p_k, p_0, p_i$.
- Thus, $p_j$ is a vertex of $C_i$.
- Moreover, all vertices of $C_i$ between $p_0$ and $p_j$ in the counterclockwise order remain vertices of $C_j$.
- Hence, after Step 11 $S$ contains precisely the vertices of $C_i$, as claimed.

# Graham's scan (ctd.)

## Is `GrahamScan` optimal?

- The best current method has running time $O(n \log h)$, where $h$ is the *number of vertices* of the convex hull.
- On the other hand, the worst-case running time for finding the vertices of the convex hull is $\Omega(n \log n)$.
- The proof of the lower bound is by a reduction to *sorting*.

# Reading assignment

## Why not take a look at. . .

- [CLRS] pages 947–957.
- Wikipedia.