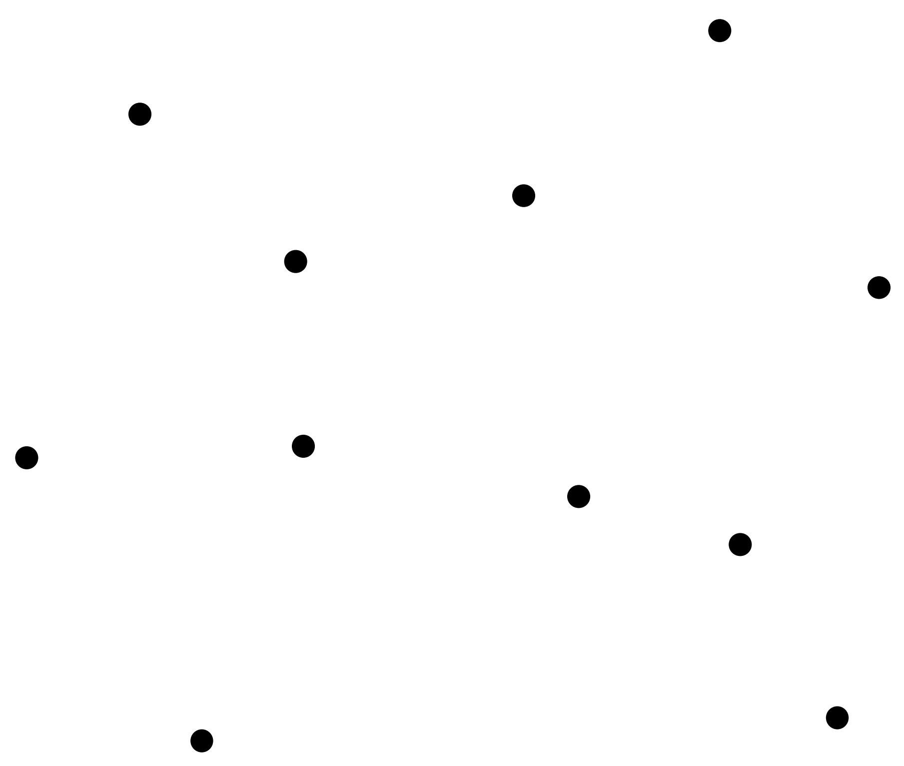
Convex hull computation – Graham's scan

Gabriel Nivasch

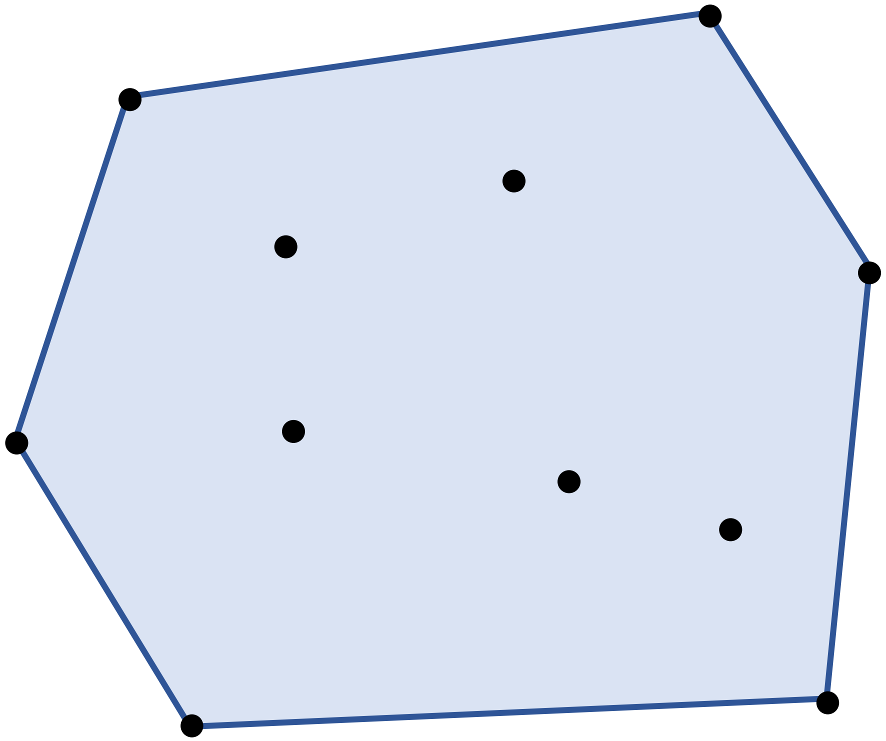
Convex hull



Let S be a set of n points in the plane

The ***convex hull*** of S is a convex polygon P, with vertices in S, that contains all the points of S

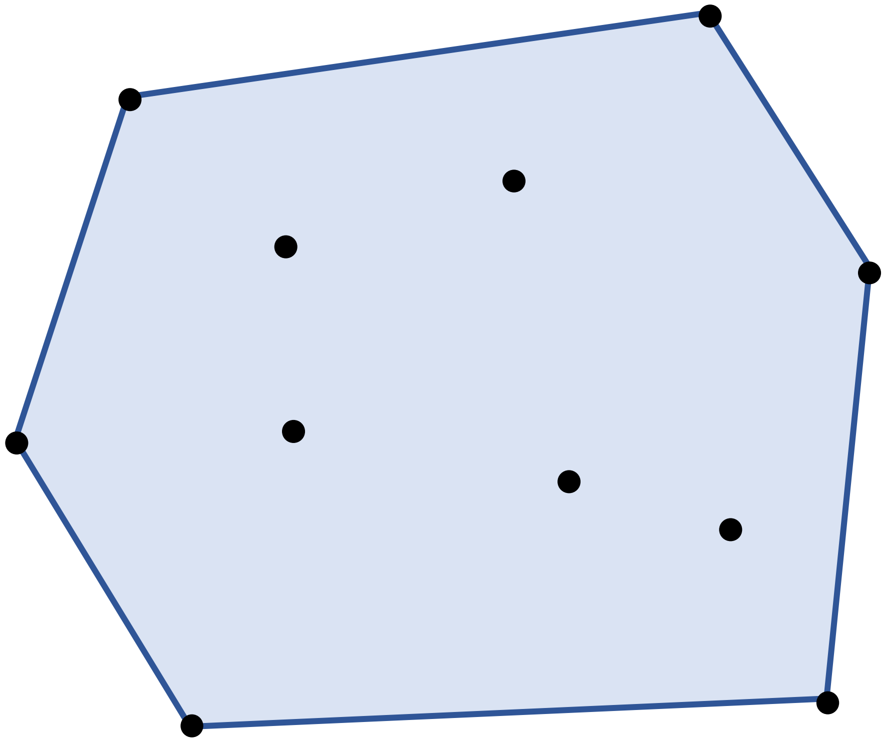
Convex hull



Let S be a set of n points in the plane

The ***convex hull*** of S is a convex polygon P, with vertices in S, that contains all the points of S

Convex hull



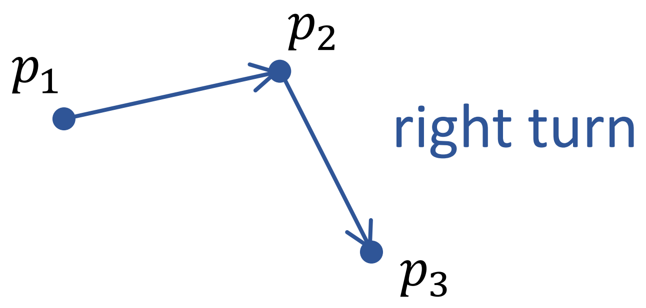
Let S be a set of n points in the plane

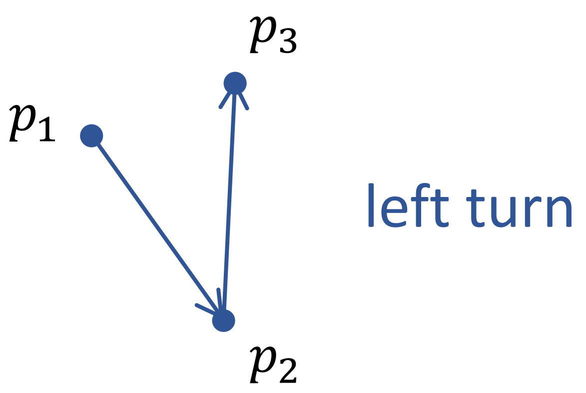
The ***convex hull*** of S is a convex polygon P, with vertices in S, that contains all the points of S

How do we compute the convex hull efficiently?

Basics: Orientation of 3 points

Given points p_1, p_2, p_3, not on the same line,

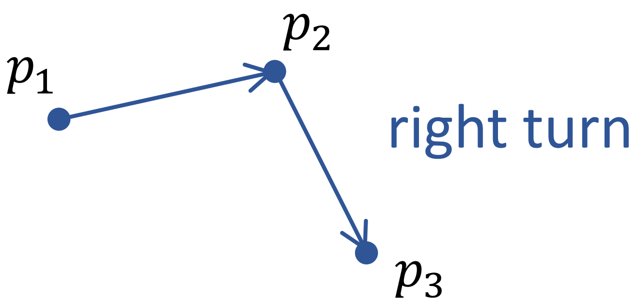


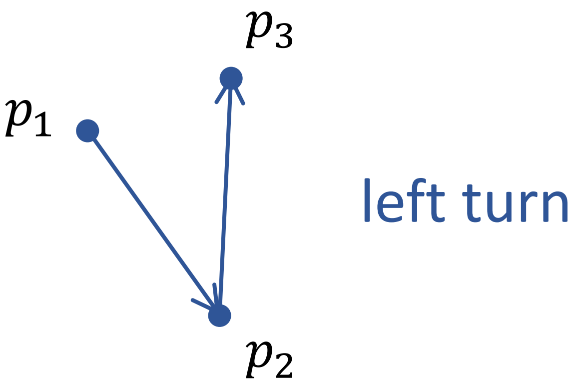


The path $p_1 \rightarrow p_2 \rightarrow p_3$ can make either a ***left turn*** or a ***right turn*** at p_2

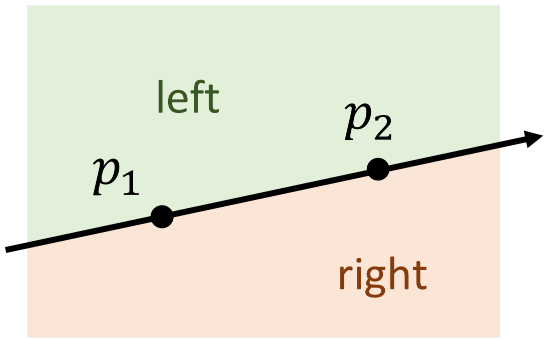
Basics: Orientation of 3 points

Given points p_1, p_2, p_3, not on the same line,





The path $p_1 \to p_2 \to p_3$ can make either a ***left turn*** or a ***right turn*** at p_2



How do we calculate whether p_1, p_2, p_3 make a left or a right turn?

How do we calculate whether p_1, p_2, p_3 make a left or a right turn?

Claim: Look at the sign of $\det \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix}$

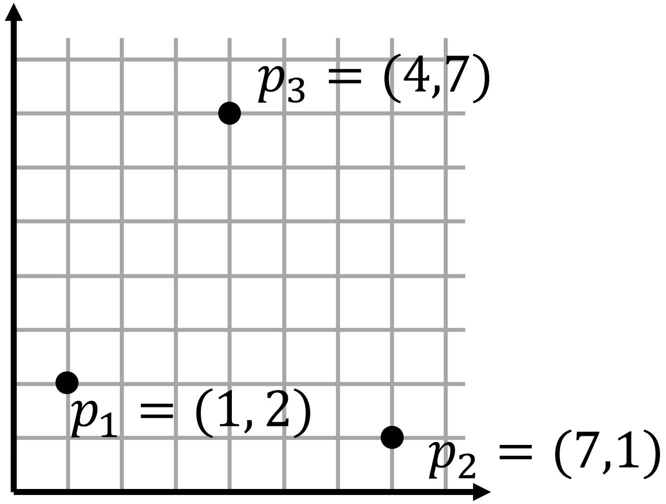
Positive: Left turn
Negative: Right turn
Zero: Points are collinear

How do we calculate whether p_1, p_2, p_3 make a left or a right turn?

Claim: Look at the sign of $\det \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix}$

Positive: Left turn
Negative: Right turn
Zero: Points are collinear

Example:



$$\det \begin{bmatrix} 1 & 1 & 1 \\ 1 & 7 & 4 \\ 2 & 1 & 7 \end{bmatrix} = 33$$

Left turn

Proof: We first prove the 0 case

Proof: We first prove the 0 case

Recall: A ***line*** is a set of the form $L = \{(x, y): ax + by = c\}$ where a, b are not both 0

Example: $3x + 4y = 5$

Proof: We first prove the 0 case

Recall: A ***line*** is a set of the form $L = \{(x, y): ax + by = c\}$ where a, b are not both 0

Example: $3x + 4y = 5$

Given p_1, p_2, p_3 let $L = \{(x, y): ax + by = c\}$ be the line through p_1, p_2

Is p_3 also in L?

Proof: We first prove the 0 case

Recall: A ***line*** is a set of the form $L = \{(x, y): ax + by = c\}$ where a, b are not both 0

Example: $3x + 4y = 5$

Given p_1, p_2, p_3 let $L = \{(x, y): ax + by = c\}$ be the line through p_1, p_2

Is p_3 also in L?

Say $a \neq 0$. Suppose w.l.o.g. $a = 1$

Proof: We first prove the 0 case

Recall: A ***line*** is a set of the form $L = \{(x, y): ax + by = c\}$ where a, b are not both 0

Example: $3x + 4y = 5$

Given p_1, p_2, p_3 let $L = \{(x, y): ax + by = c\}$ be the line through p_1, p_2

Is p_3 also in L?

Say $a \neq 0$. Suppose w.l.o.g. $a = 1$

Take the matrix $\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix}$ and do the row operation $r_2 \leftarrow r_2 + br_3 - cr_1$

Proof: We first prove the 0 case

Recall: A ***line*** is a set of the form $L = \{(x, y): ax + by = c\}$ where a, b are not both 0

Example: $3x + 4y = 5$

Given p_1, p_2, p_3 let $L = \{(x, y): ax + by = c\}$ be the line through p_1, p_2

Is p_3 also in L?

Say $a \neq 0$. Suppose w.l.o.g. $a = 1$

Take the matrix $\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix}$ and do the row operation $r_2 \leftarrow r_2 + br_3 - cr_1$

Determinant is unchanged

Proof: We first prove the 0 case

Recall: A ***line*** is a set of the form $L = \{(x, y): ax + by = c\}$ where a, b are not both 0

Example: $3x + 4y = 5$

Given p_1, p_2, p_3 let $L = \{(x, y): ax + by = c\}$ be the line through p_1, p_2

Is p_3 also in L?

Say $a \neq 0$. Suppose w.l.o.g. $a = 1$

Take the matrix $\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix}$ and do the row operation $r_2 \leftarrow r_2 + br_3 - cr_1$

Determinant is unchanged

Row r_2 becomes 0 if and only if $p_3 \in L$

Proof: We first prove the 0 case

Recall: A ***line*** is a set of the form $L = \{(x, y): ax + by = c\}$ where a, b are not both 0

Example: $3x + 4y = 5$

Given p_1, p_2, p_3 let $L = \{(x, y): ax + by = c\}$ be the line through p_1, p_2

Is p_3 also in L?

Say $a \neq 0$. Suppose w.l.o.g. $a = 1$

Take the matrix $\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix}$ and do the row operation $r_2 \leftarrow r_2 + br_3 - cr_1$

Determinant is unchanged

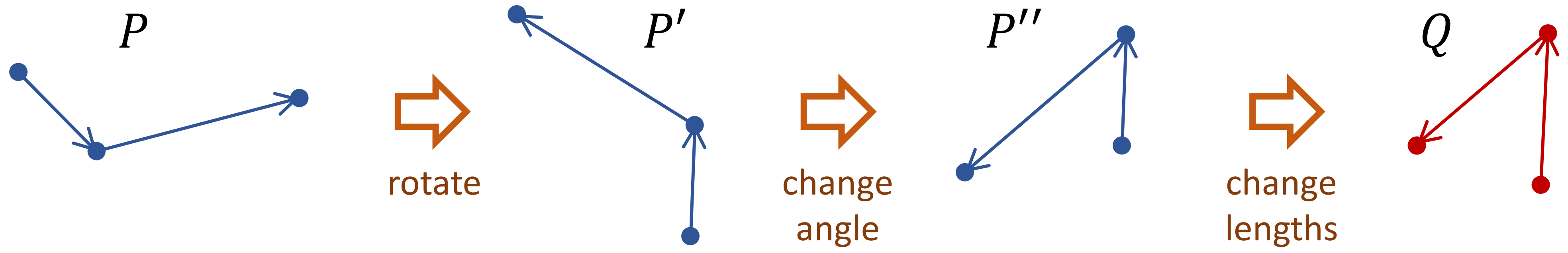
Row r_2 becomes 0 if and only if $p_3 \in L$

(Also $y_1 \neq y_2$)

Proof (cont.):

Suppose $P = (p_1, p_2, p_3)$ make a left turn, and $Q = (q_1, q_2, q_3)$ also

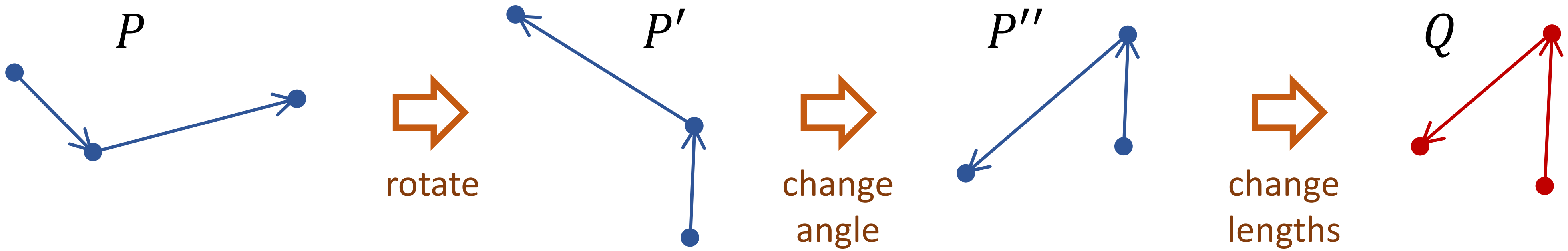
We can transform P into Q continuously, without the points ever becoming collinear



Proof (cont.):

Suppose $P = (p_1, p_2, p_3)$ make a left turn, and $Q = (q_1, q_2, q_3)$ also

We can transform P into Q continuously, without the points ever becoming collinear

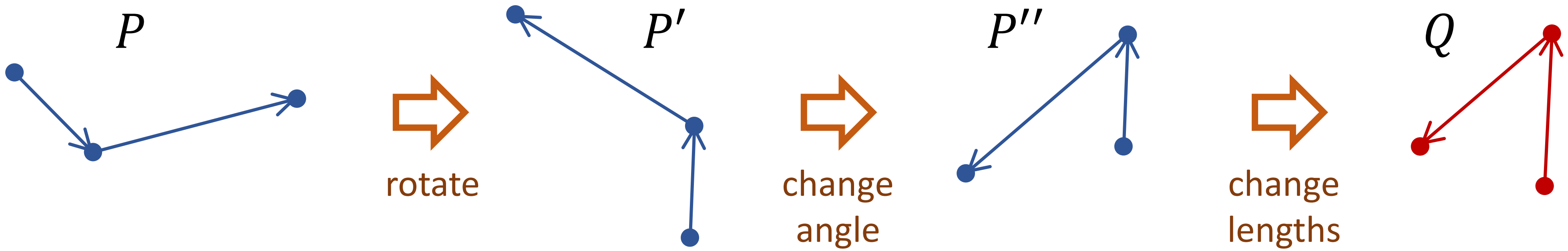


Determinant varies continuously, and never becomes 0

Proof (cont.):

Suppose $P = (p_1, p_2, p_3)$ make a left turn, and $Q = (q_1, q_2, q_3)$ also

We can transform P into Q continuously, without the points ever becoming collinear

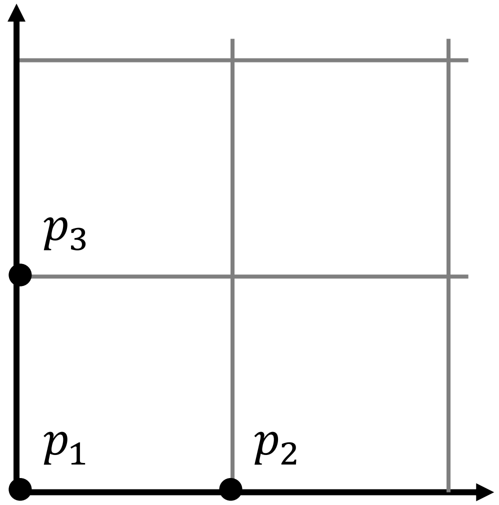


Determinant varies continuously, and never becomes 0

➔ P and Q have the same determinant sign

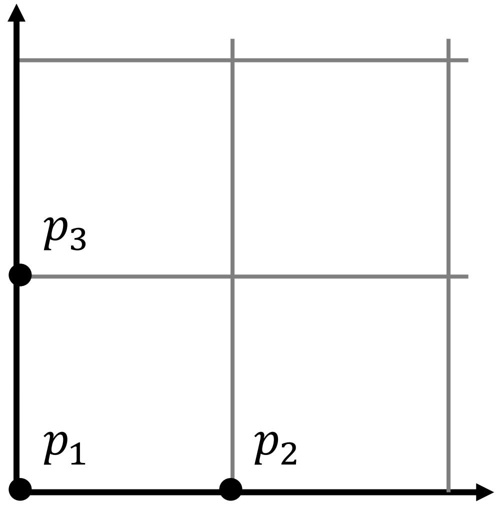
Proof (cont.):

Verify an easy “left turn” case: $p_1 = (0,0),\ p_2 = (1,0),\ p_3 = (0,1)$



Proof (cont.):

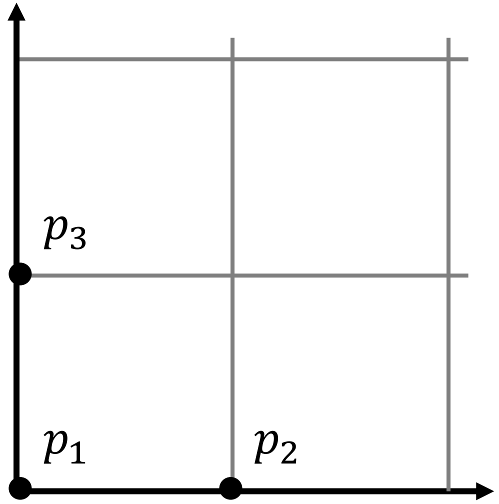
Verify an easy "left turn" case: $p_1 = (0,0),\ p_2 = (1,0),\ p_3 = (0,1)$



Determinant is positive

Proof (cont.):

Verify an easy "left turn" case: $p_1 = (0,0),\ p_2 = (1,0),\ p_3 = (0,1)$

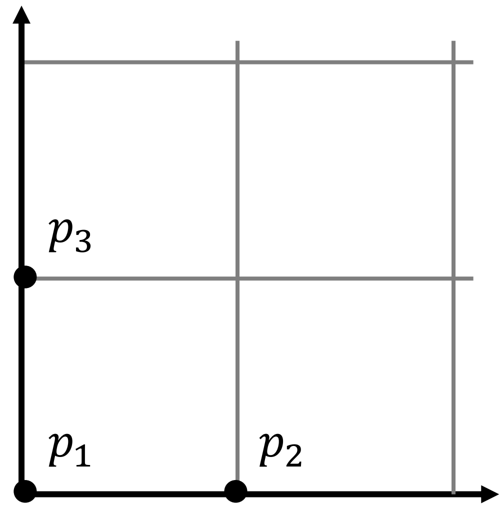


Determinant is positive

➔ Every left turn has positive determinant

Proof (cont.):

Verify an easy "left turn" case: $p_1 = (0,0),\ p_2 = (1,0),\ p_3 = (0,1)$



Determinant is positive

➔ Every left turn has positive determinant

"Right turn": Similarly **QED**

Graham's scan

Input: Sequence of n points $p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n)$

Output: Vertices of the CH, in clockwise order

Graham’s scan

Input: Sequence of n points $p_1 = (x_1, y_1), \dots, p_n = (x_n, y_n)$

Output: Vertices of the CH, in clockwise order

(Assume general position: No 3 points collinear)

Graham’s scan

Input: Sequence of n points $p_1 = (x_1, y_1), \dots, p_n = (x_n, y_n)$

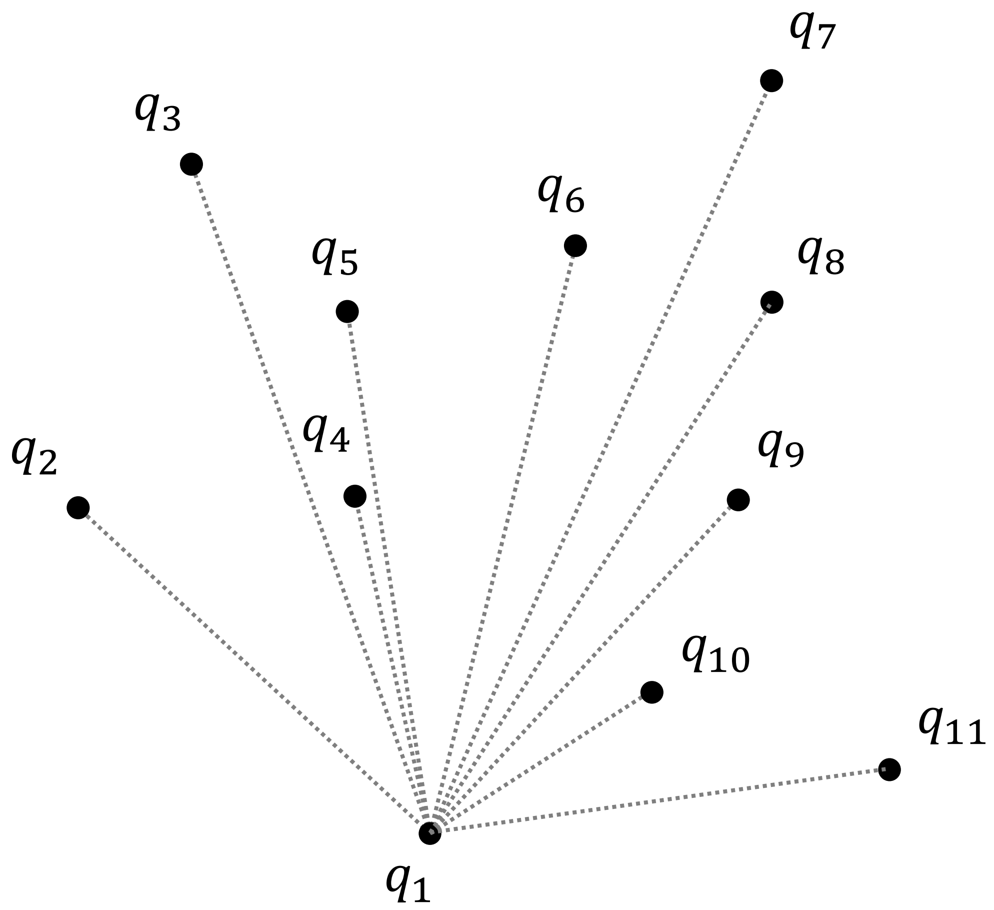
Output: Vertices of the CH, in clockwise order

(Assume general position: No 3 points collinear)

Step 1: Find a point q_1 that is certainly a CH vertex, say point with lowest y-coordinate

Time: $O(n)$

Step 2: Sort the remaining points clockwise w.r.t. q_1

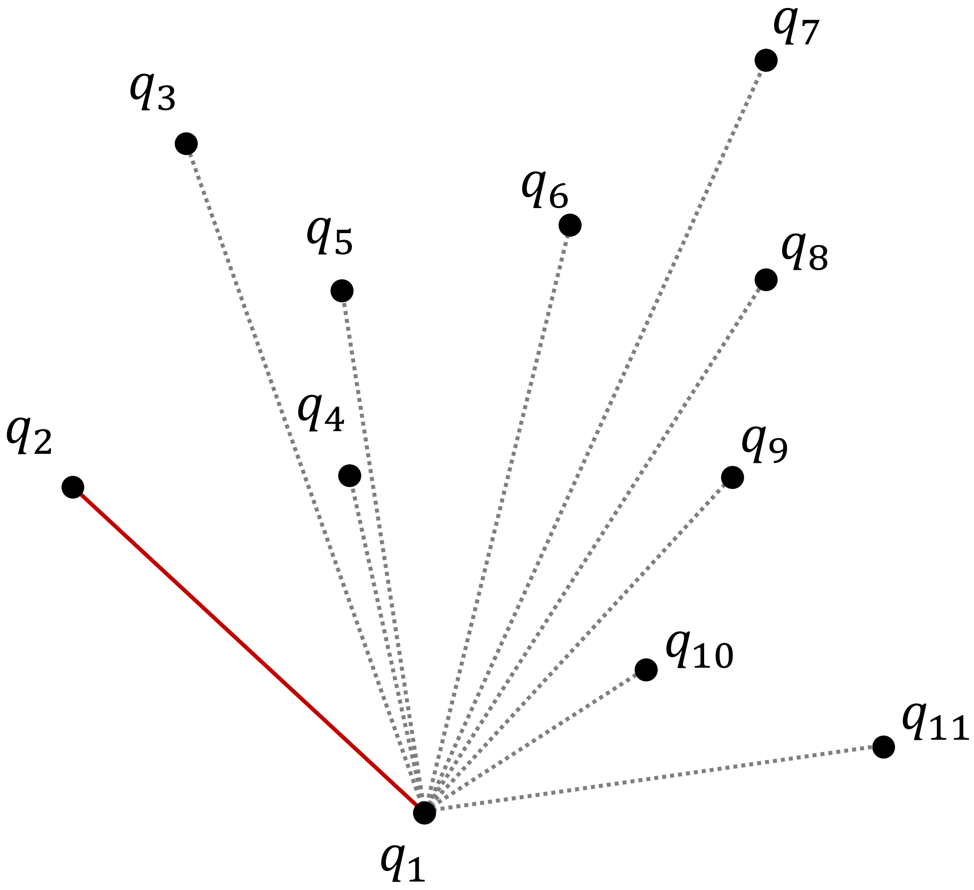


Whenever the sorting algorithm asks "is $r < s$?" we check whether r, q_1, s make a left turn

Time: $O(n \log n)$

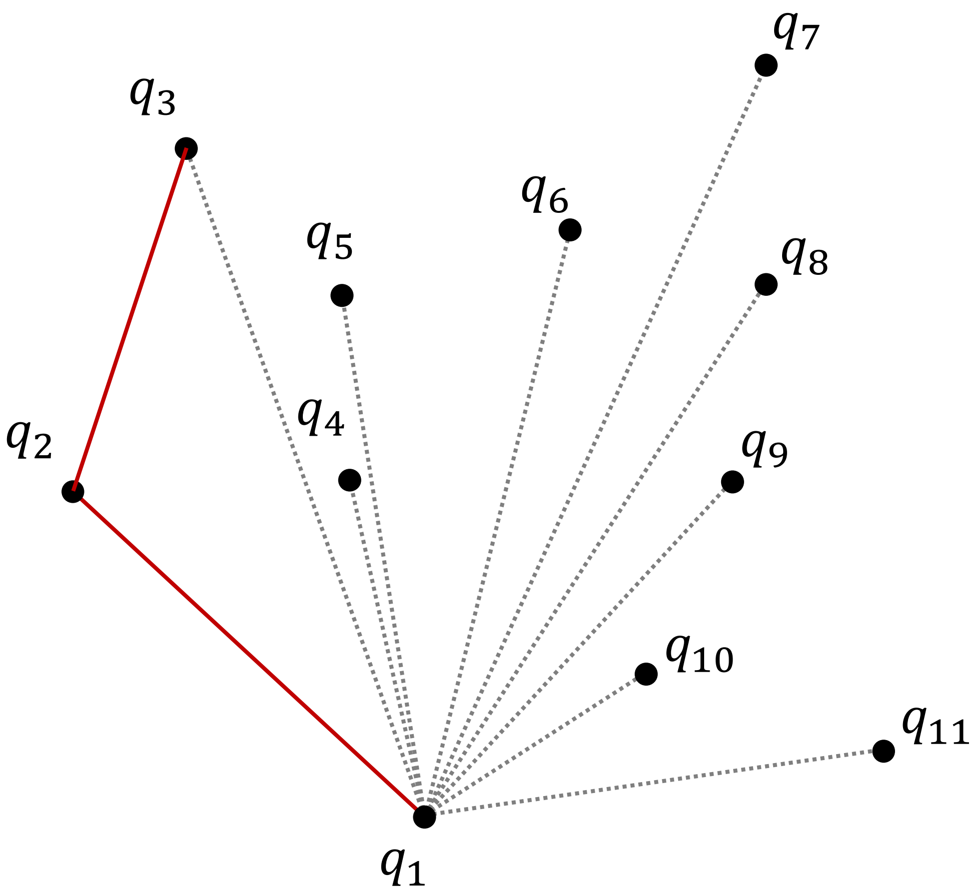
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



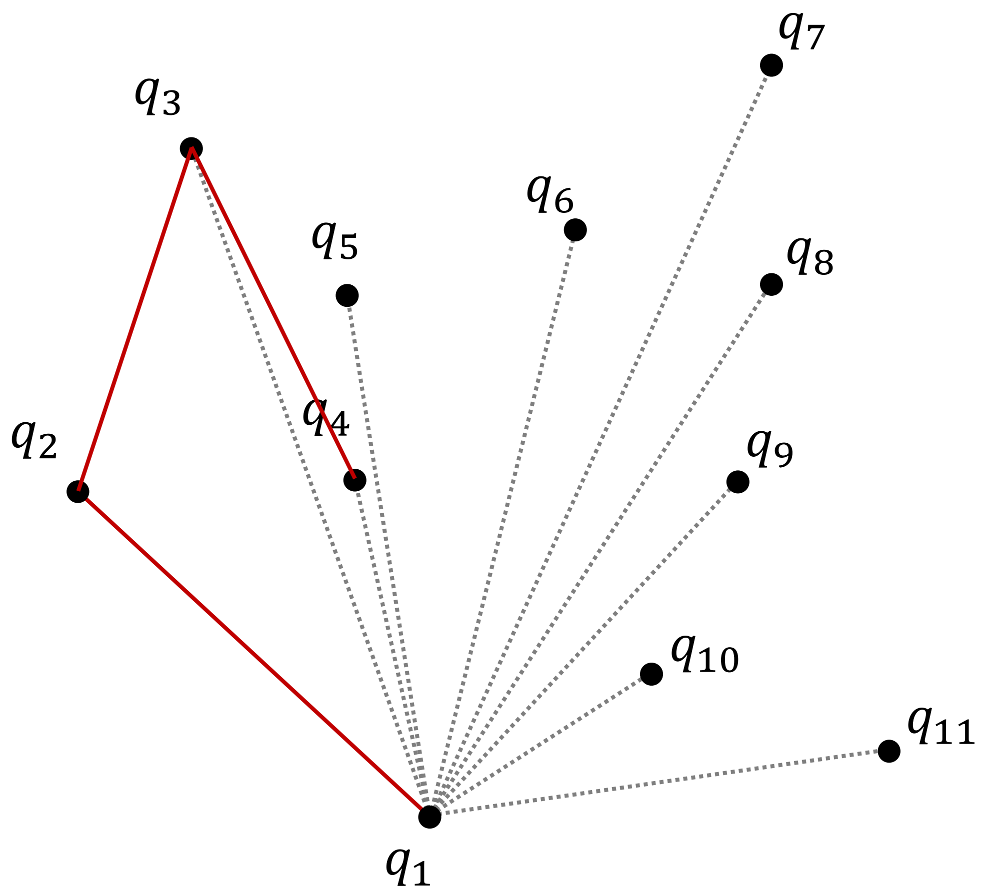
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



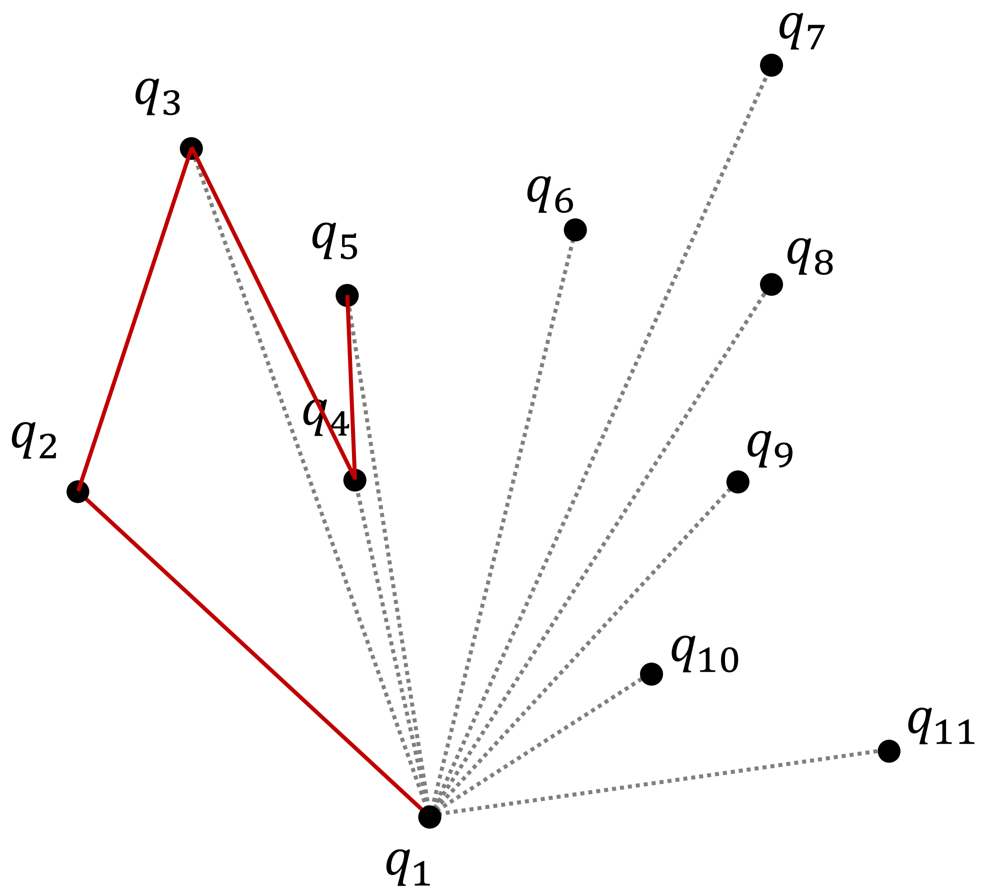
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



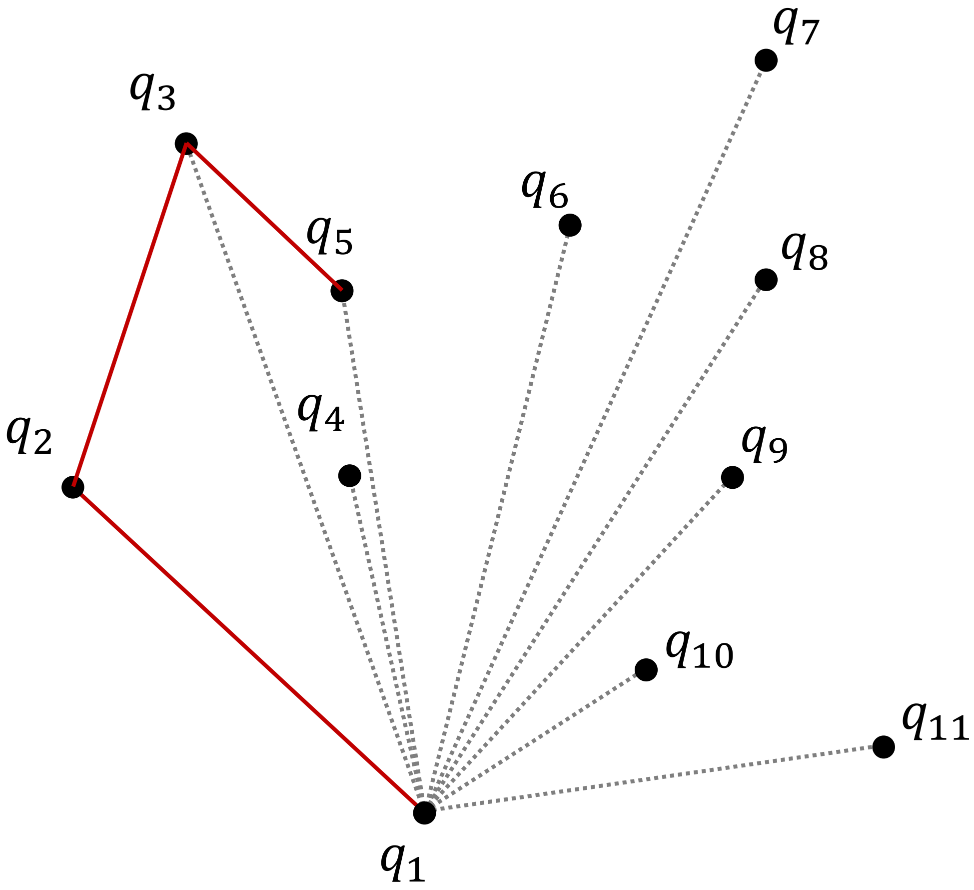
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



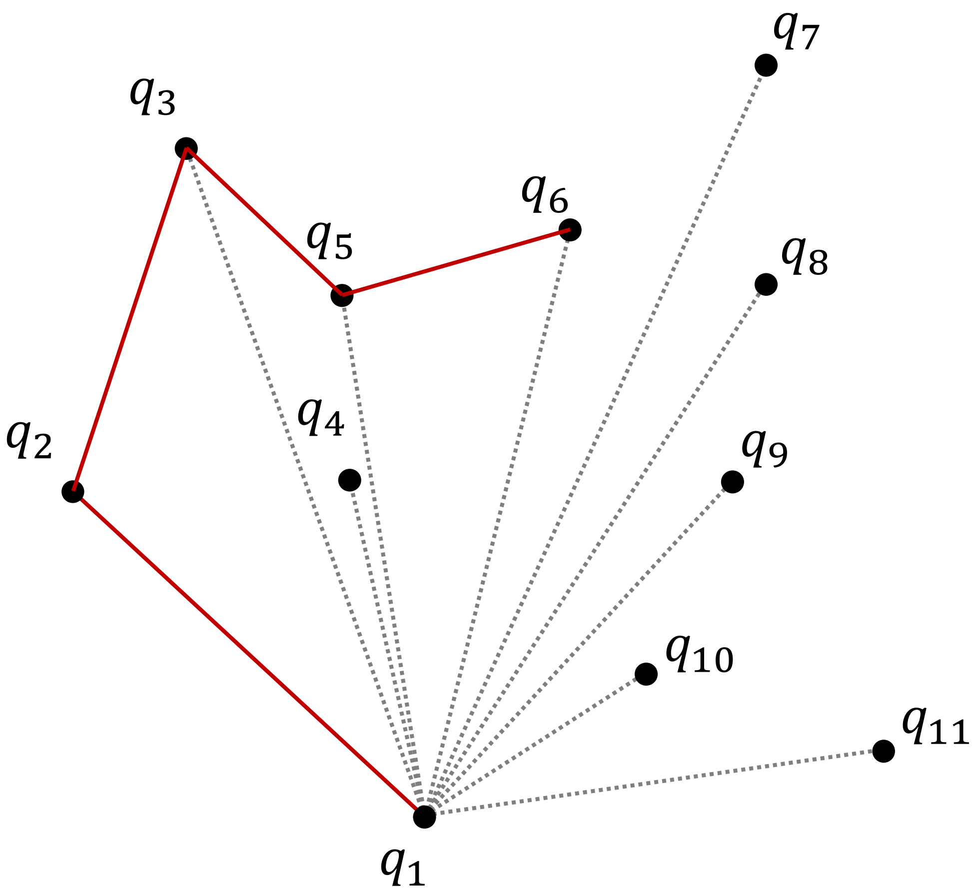
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



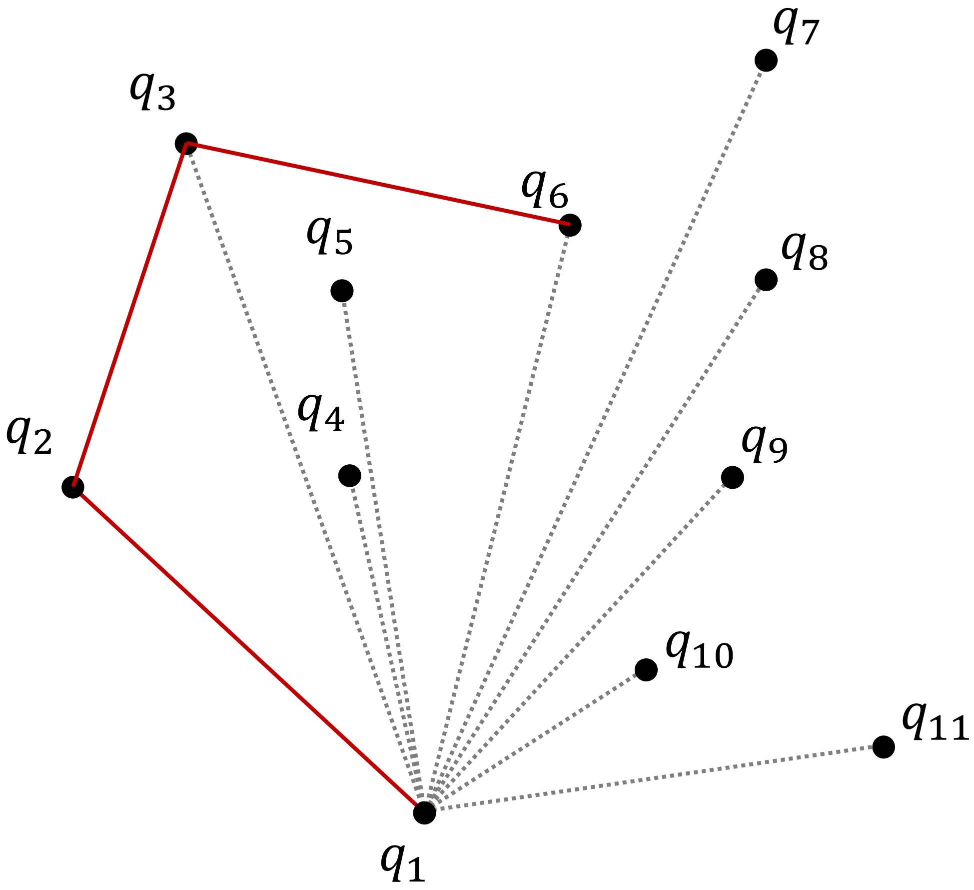
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



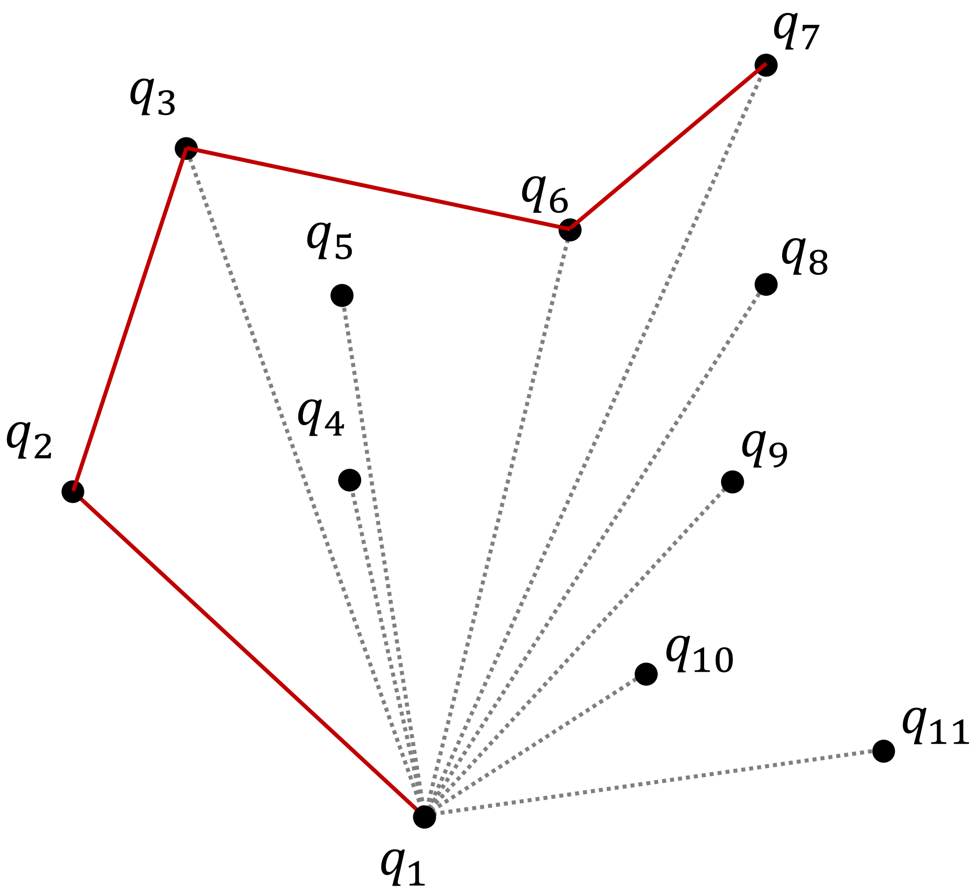
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



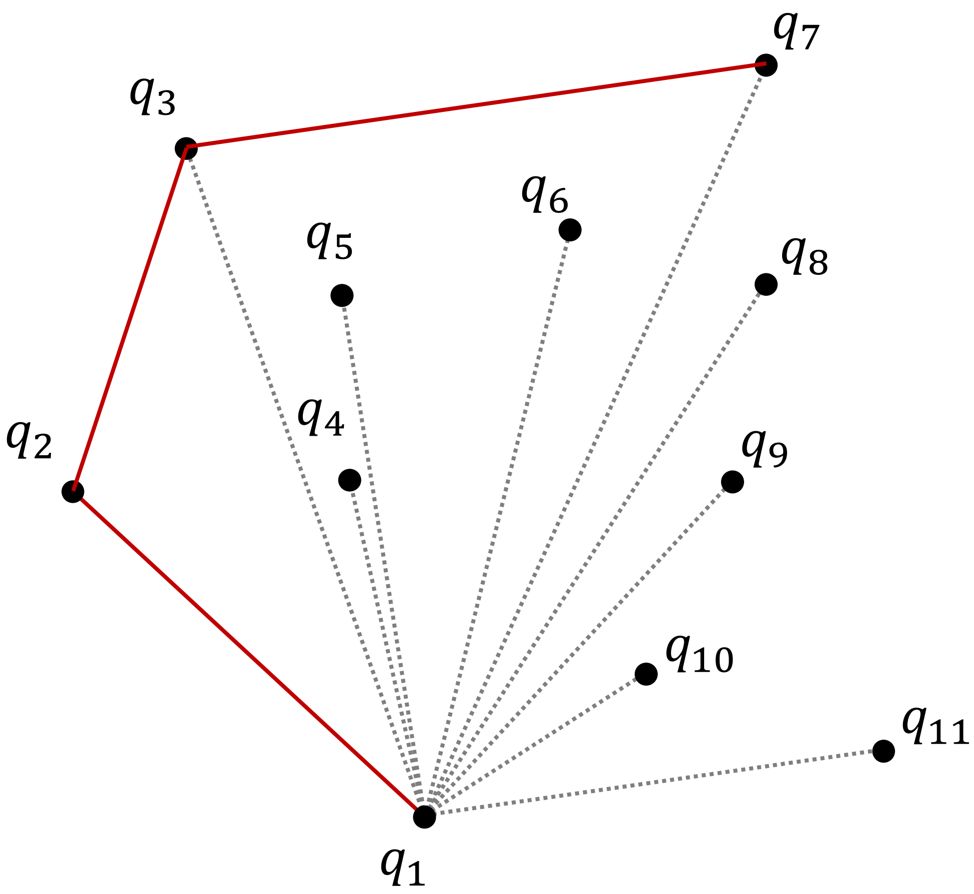
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



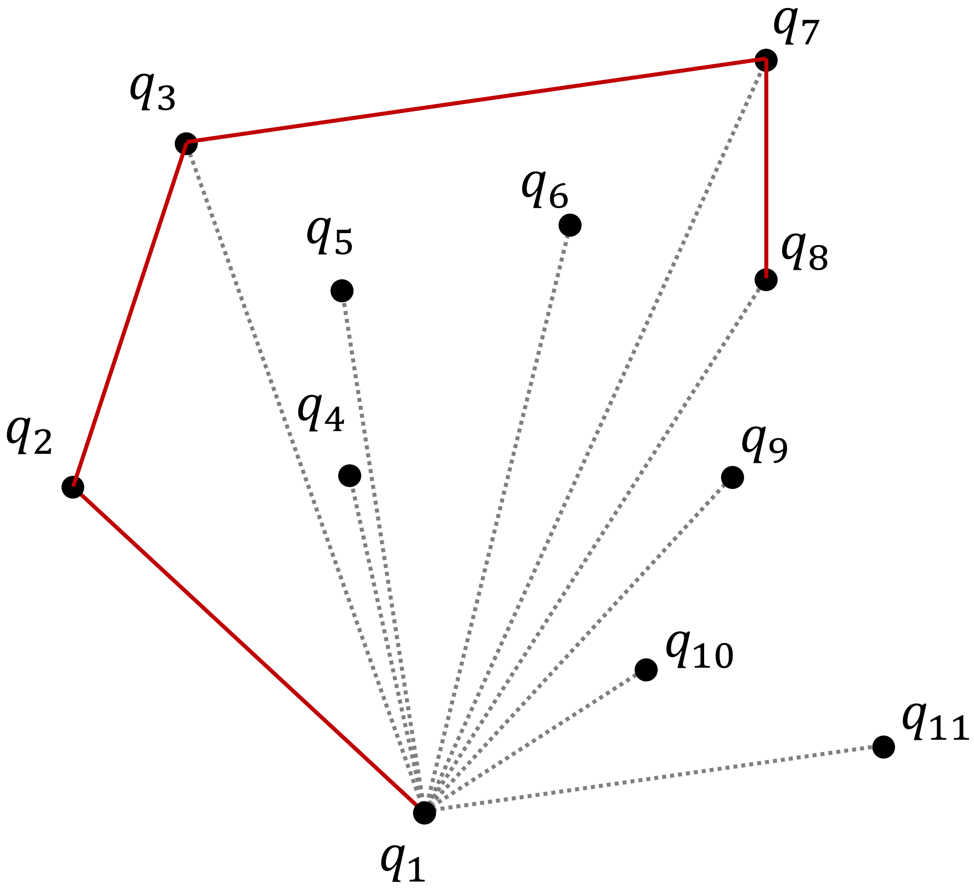
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



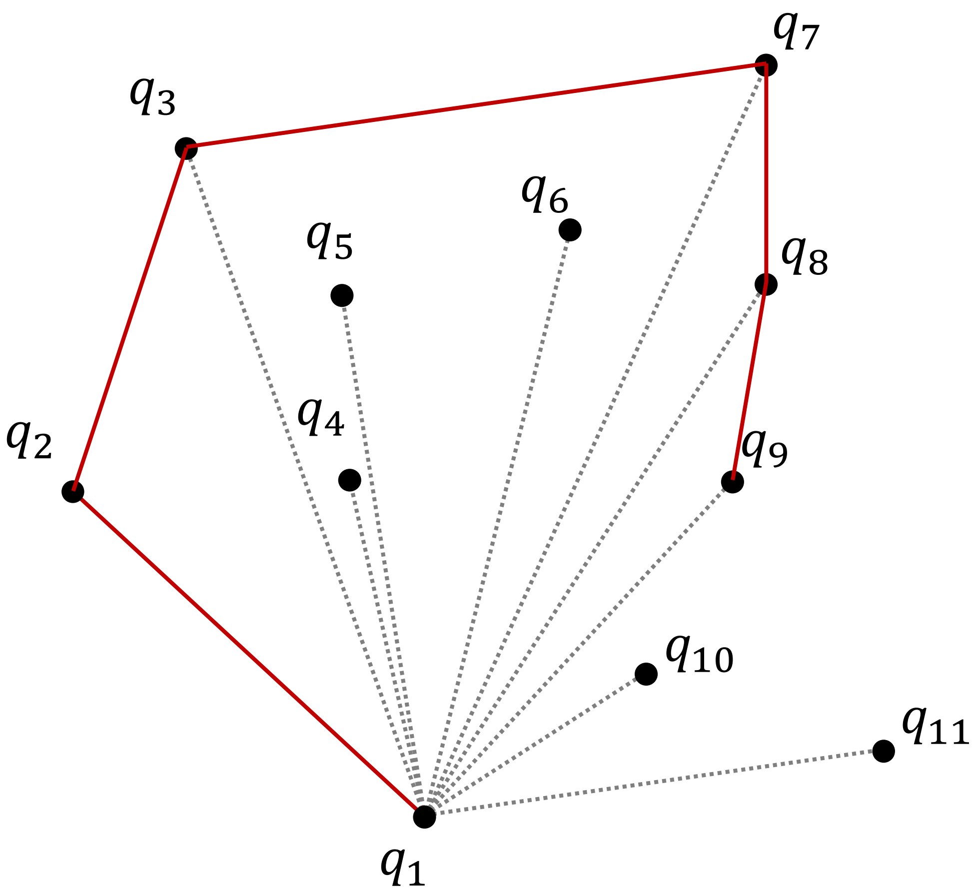
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



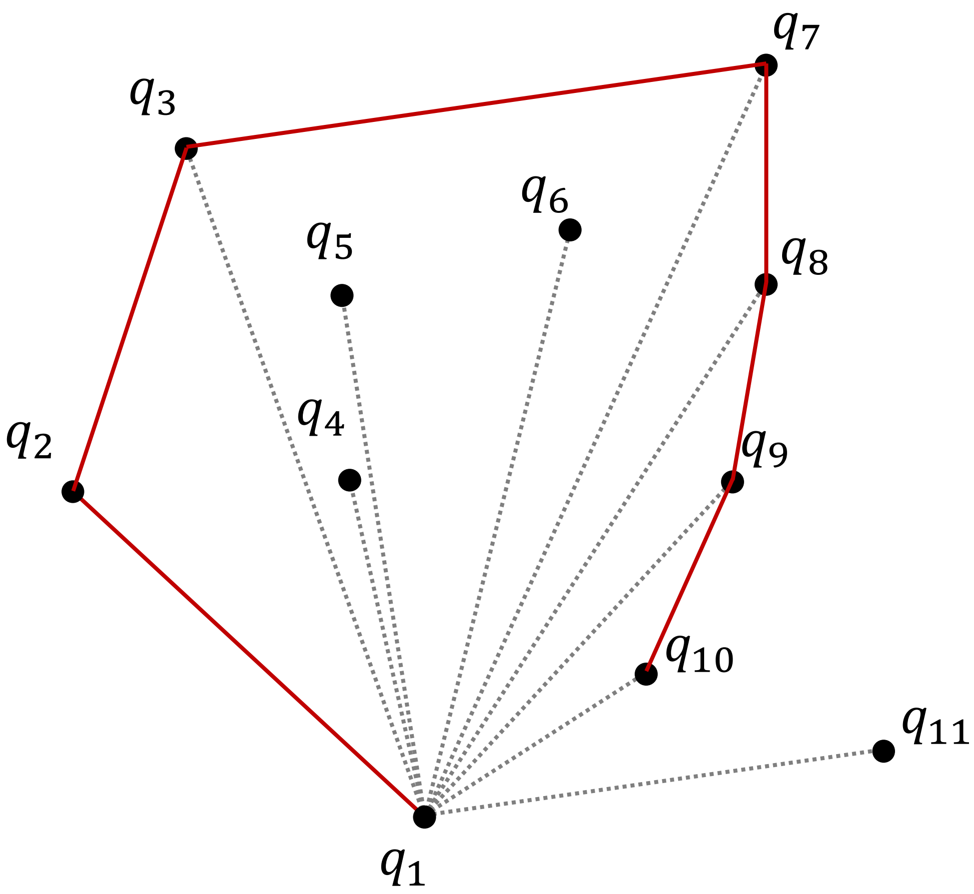
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



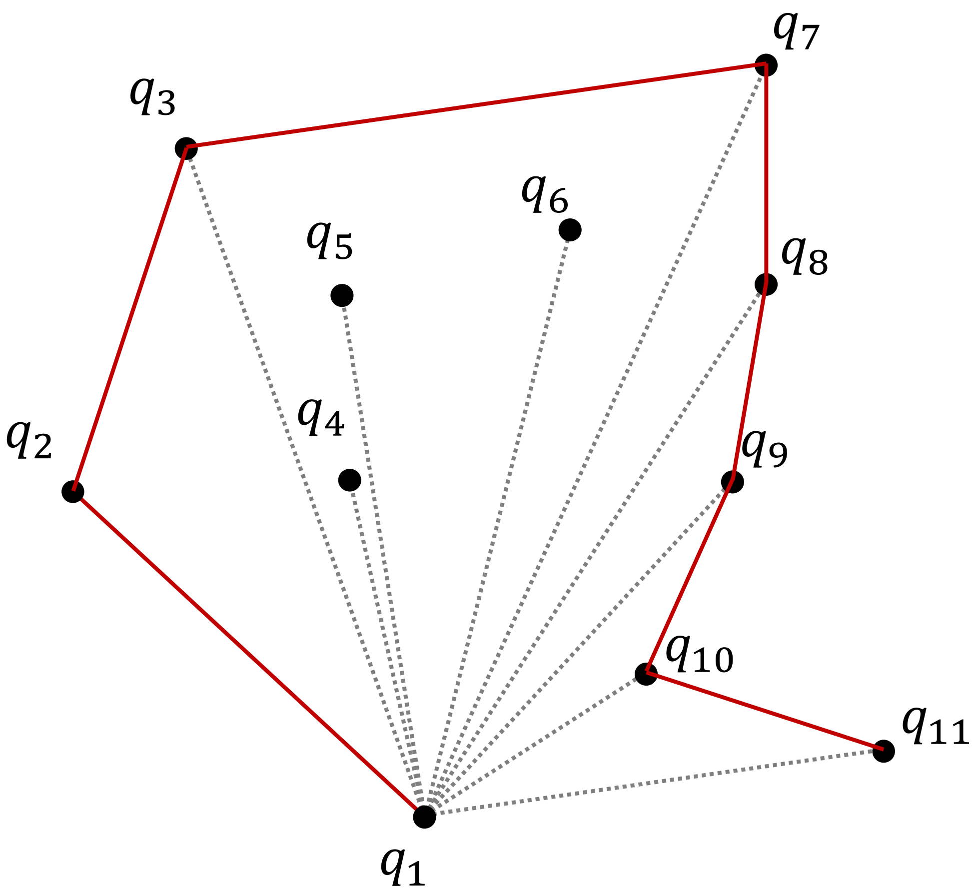
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



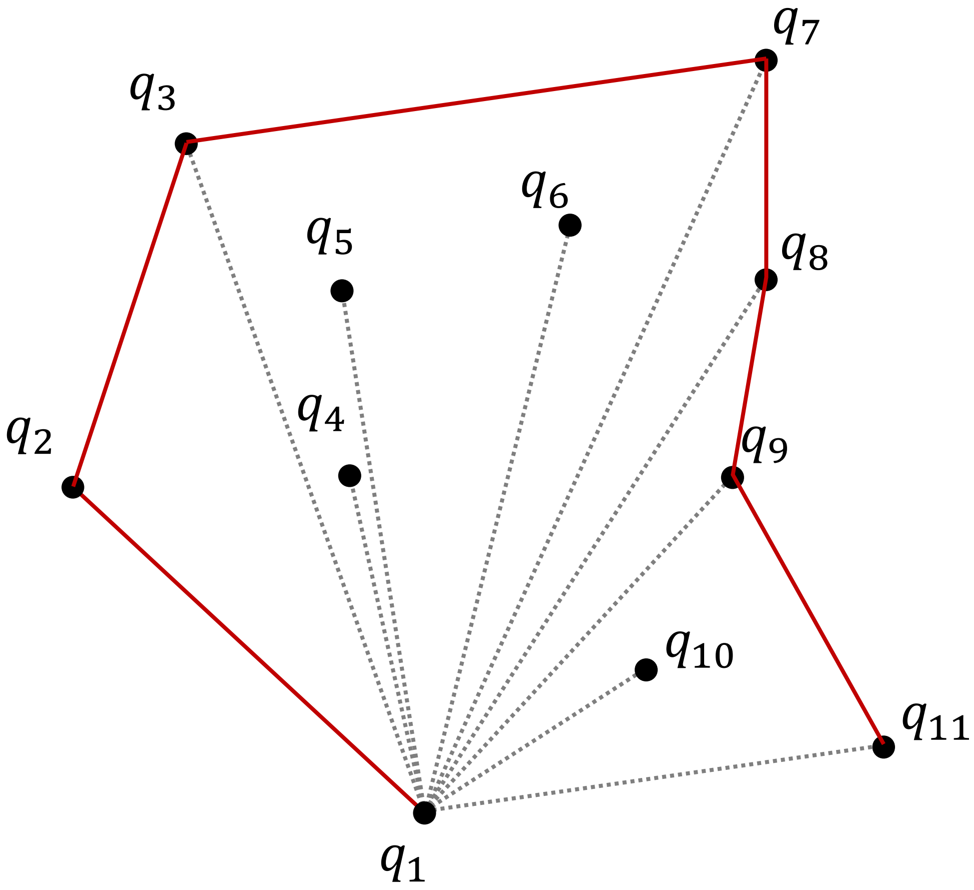
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



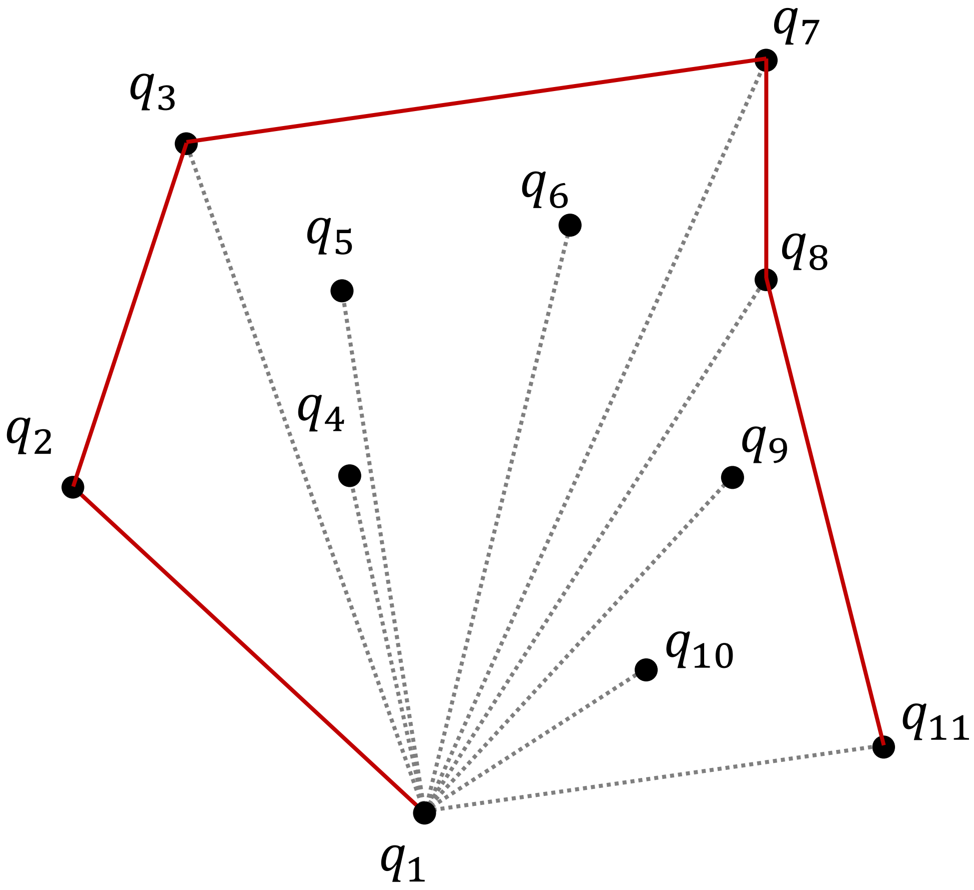
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



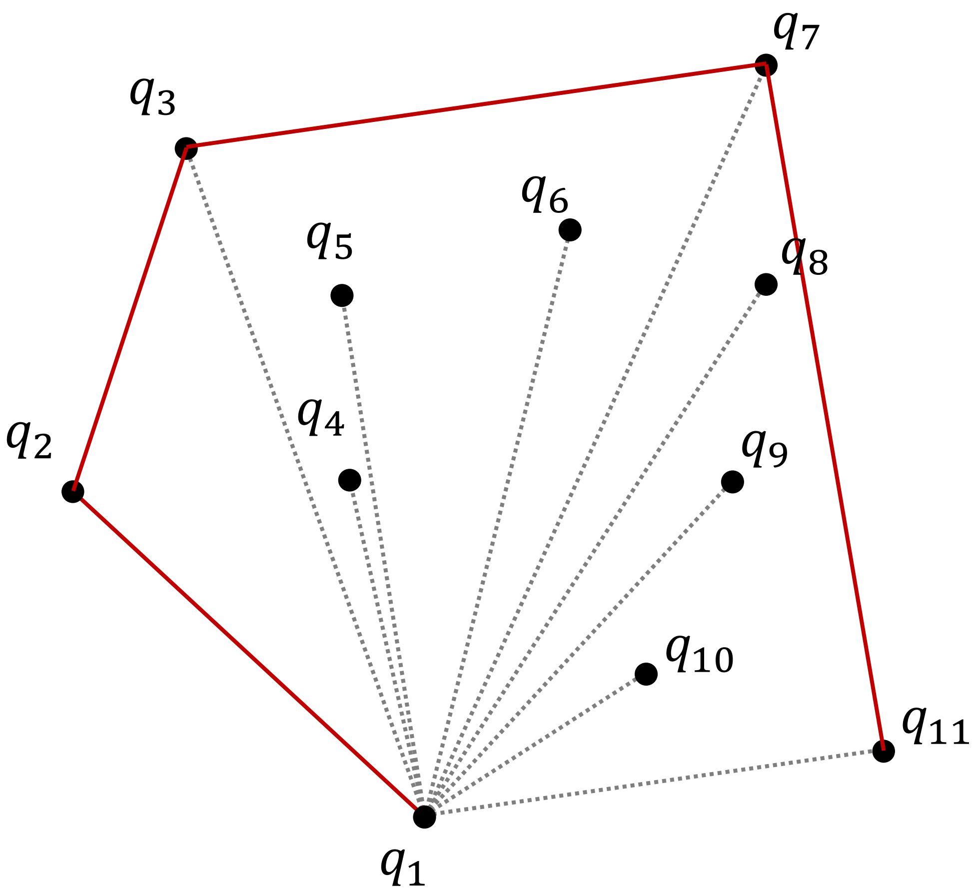
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



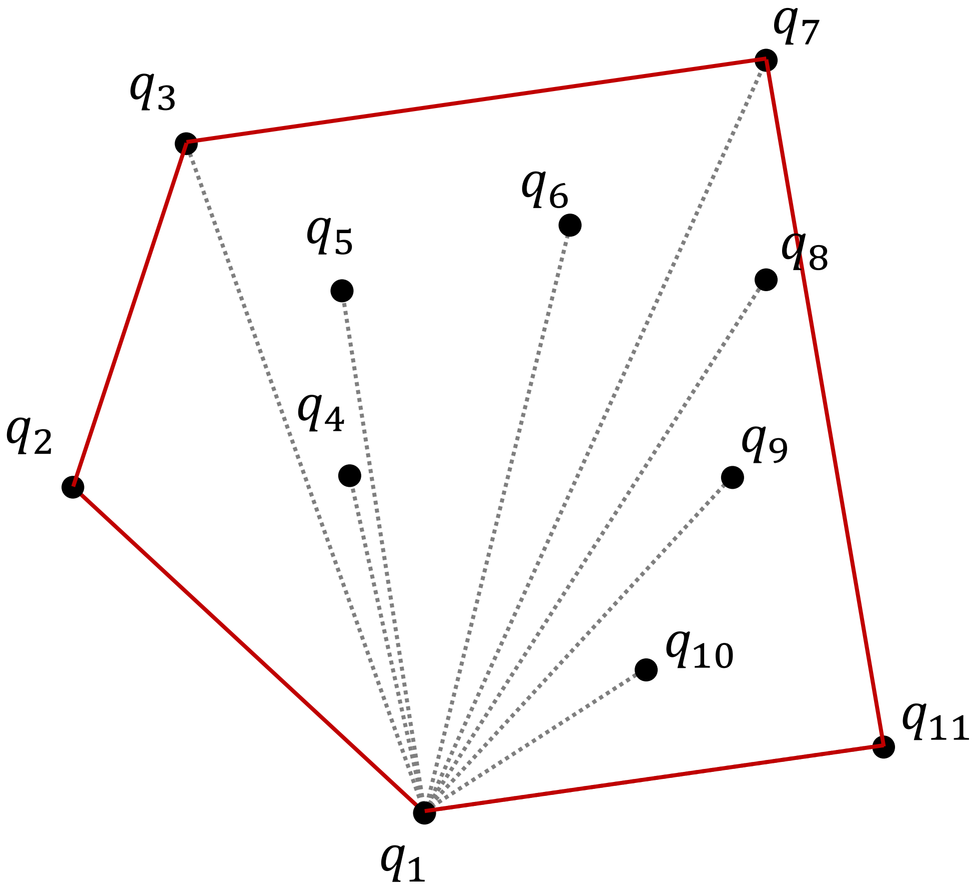
Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



Step 3:

- Initialize a stack to $S = (q_1, q_2)$
- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$
- Return the points in S



Running time of step 3?

- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$

Running time of step 3?

- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$

$O(n)$ worst case

Running time of step 3?

- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$

$\}$ $O(n)$ worst case

Worst-case time $O(n^2)$??

Running time of step 3?

- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$

$O(n)$ worst case

Worst-case time $O(n^2)$??

No! Each point is pushed once, and popped at most once

Running time of step 3?

- For $i = 3$ to n:
  - While $S[-2], S[-1], q_i$ make a left turn
    - $S.\mathrm{pop}()$
  - $S.\mathrm{push}(q_i)$

$O(n)$ worst case

Worst-case time $O(n^2)$??

No! Each point is pushed once, and popped at most once

Step 3 running time: $O(n)$

This is called ***amortized analysis***

Time to handle each point q_i in step 3:

- **Worst case:** $O(n)$
- **Amortized:** $O(1)$

Meaning, handling all n points together takes time $O(n)$, so we divide that by n

This is called ***amortized analysis***

Time to handle each point q_i in step 3:

- **Worst case:** $O(n)$
- **Amortized:** $O(1)$

Meaning, handling all n points together takes time $O(n)$, so we divide that by n

$$\boxed{\textbf{Overall running time of Graham's scan: } O(n \log n)}$$