Algorithms for Programming Contests - Week 11

Tobias Meggendorfer, Philipp Meyer,
Christian Müller, Gregor Schwarz
`conpra@in.tum.de`

16.01.2019

Geometry

Geometry

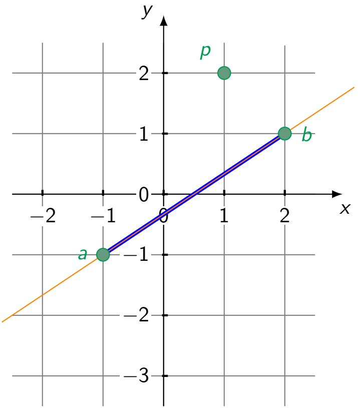
- Euclidean geometry
- 2-dimensional space
- Cartesian coordinate system

Basic elements

- Point $p = (p_x, p_y) \in \mathbb{R}^2$
- Line (a, b) given by two points a and b with $a \neq b$
- Line segment between a and b

Points $p = (1, 2)$, $a = (-1, -1)$, $b = (2, 1)$

Line (segment) given by a and b



Points and lines

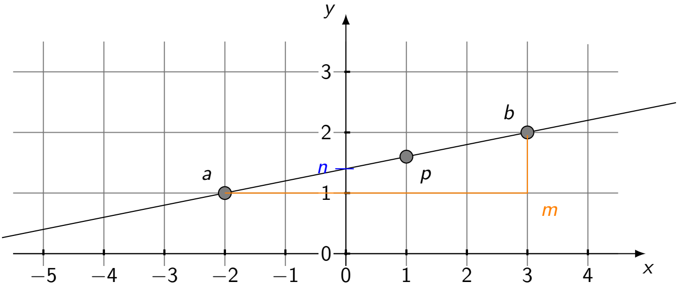
Point $p = (p_x, p_y)$ *lies on* a line given by $a = (a_x, a_y)$ and $b = (b_x, b_y)$ if

$$(p_y - a_y)(b_x - a_x) = (p_x - a_x)(b_y - a_y)$$

If $a_x \neq b_x$, we get the *slope-intercept form*:

$$p_y = m \cdot p_x + n \qquad \text{where } m := \frac{b_y - a_y}{b_x - a_x} \text{ and } n := a_y - m \cdot a_x$$

If $a_x = b_x$ and $a_y \neq b_y$ (line parallel to y-axis), we get $p_x = a_x$.



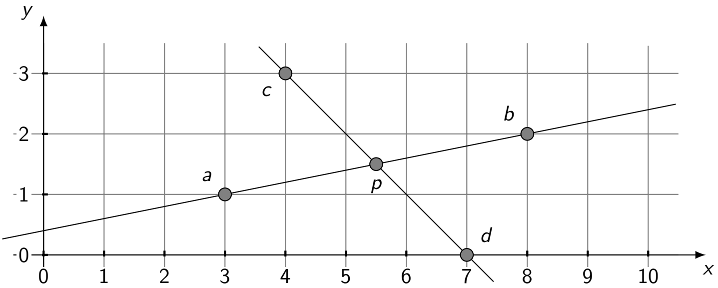
Intersection of lines

A point p is on the intersection of two lines (a, b) and (c, d) if:

$$(p_y - a_y)(b_x - a_x) = (p_x - a_x)(b_y - a_y)$$
$$(p_y - c_y)(d_x - c_x) = (p_x - c_x)(d_y - c_y)$$

There is a unique solution if the lines are not parallel, i.e. if:

$$(b_y - a_y)(d_x - c_x) \neq (b_x - a_x)(d_y - c_y)$$

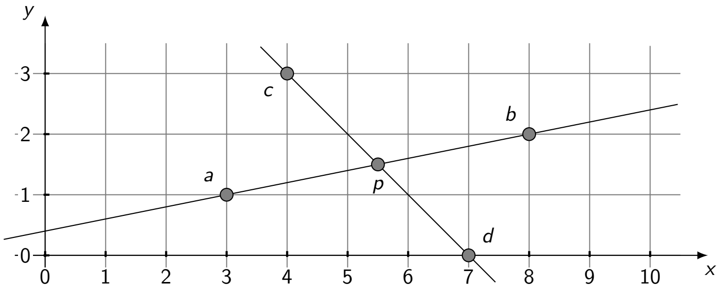


Intersection of lines

This unique solution is given by the closed form:

$$p_x = \frac{(b_x - a_x)(c_x d_y - d_x c_y) - (d_x - c_x)(a_x b_y - b_x a_y)}{(b_x - a_x)(d_y - c_y) - (b_y - a_y)(d_x - c_x)}$$

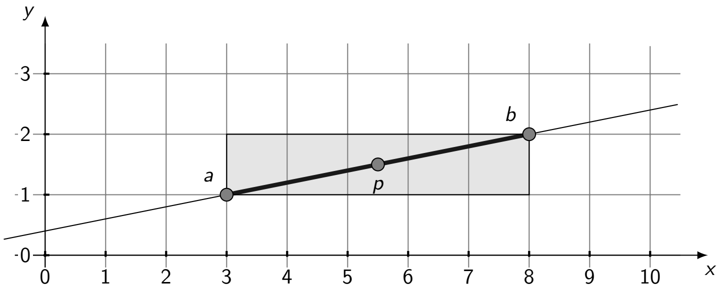
$$p_y = \frac{(b_y - a_y)(c_x d_y - d_x c_y) - (d_y - c_y)(a_x b_y - b_x a_y)}{(b_x - a_x)(d_y - c_y) - (b_y - a_y)(d_x - c_x)}$$



Point on line segment

If p is on the line (a, b), then it is on the *line segment* between a and b if

$$((a_x \leq p_x \leq b_x) \vee (b_x \leq p_x \leq a_x)) \text{ and } ((a_y \leq p_y \leq b_y) \vee (b_y \leq p_y \leq a_y))$$



Take care of numerical stability when differences are small!

Point on side of a line

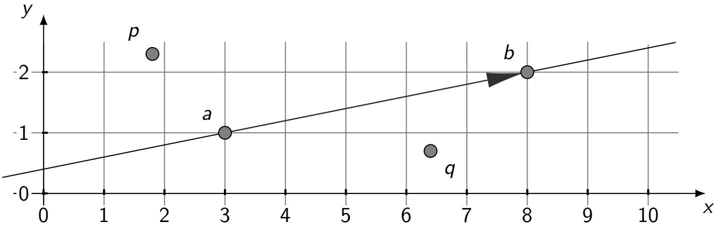
Point on side of a line problem

Given a point p and a line (a, b), determine if the point

- on the line,
- to the left of the line, or
- to the right of the line,

when considering the direction from a to b.

p is to the left and q is to the right of the directed line (a, b)

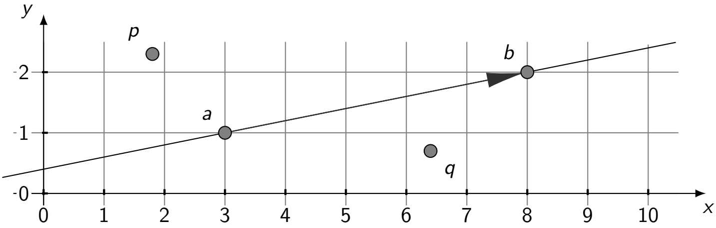


Point on side of a line

Define the following sets:

$$M_o = \{p \mid p \text{ is on } (a, b)\}$$
$$M_l = \{p \mid p \text{ is to the left of } (a, b)\}$$
$$M_r = \{p \mid p \text{ is to the right of } (a, b)\}$$

p is to the left and q is to the right of the directed line (a, b)

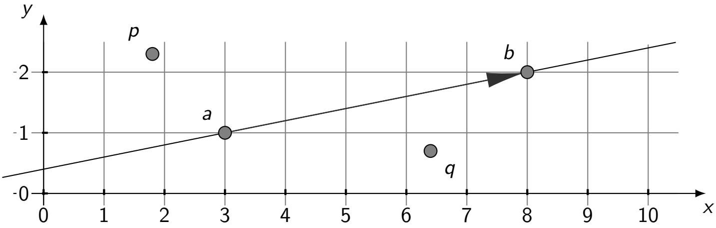


Point on side of a line

If $a_x \neq b_x$, with the slope-intercept form $p_y = mp_x + n$, we get:

$$M_o = \{p \mid p_y = mp_x + n\}$$
$$M_l = \{p \mid p_y > mp_x + n\}$$
$$M_r = \{p \mid p_y < mp_x + n\}$$

p is to the left and q is to the right of the directed line (a, b)

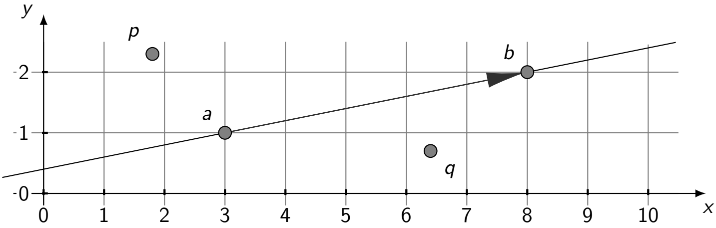


Point on side of a line

If $a_x = b_x$ and (w.l.o.g) $b_y < a_y$, we get:

$$M_o = \{p \mid p_x = a_x\}$$
$$M_l = \{p \mid p_x > a_x\}$$
$$M_r = \{p \mid p_x < a_x\}$$

p is to the left and q is to the right of the directed line (a, b)



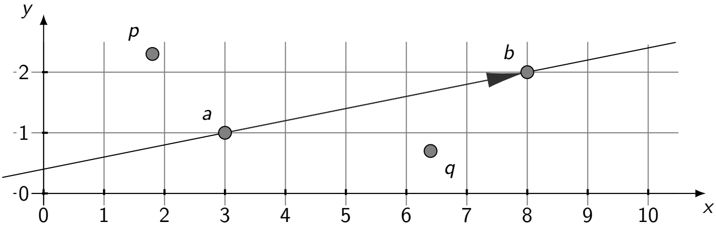
Point on side of a line

With $\mathrm{CCW}(a, b, p) := (p_y - a_y)(b_x - a_x) - (p_x - a_x)(b_y - a_y)$, we obtain

$$M_o = \{p \mid \mathrm{CCW}(a, b, p) = 0\}$$
$$M_l = \{p \mid \mathrm{CCW}(a, b, p) > 0\}$$
$$M_r = \{p \mid \mathrm{CCW}(a, b, p) < 0\}$$

for both cases.

p is to the left and q is to the right of the directed line (a, b)

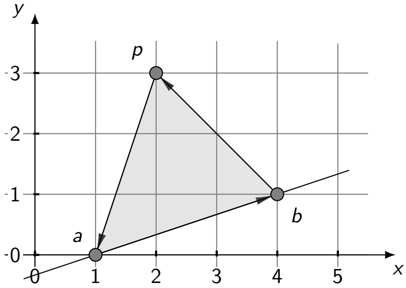


Counter-clockwise function

- CCW is called the counter-clockwise function, as it is positive if the path from a to b to p constitutes a counter-clockwise turn.
- CCW can be also be defined by $\mathrm{CCW}(a, b, p) = \det \begin{pmatrix} a_x & b_x & p_x \\ a_y & b_y & p_y \\ 1 & 1 & 1 \end{pmatrix}$
- Value of $\frac{1}{2}\left|\mathrm{CCW}(a, b, p)\right|$ is the area of the triangle (a, b, p).

Example:

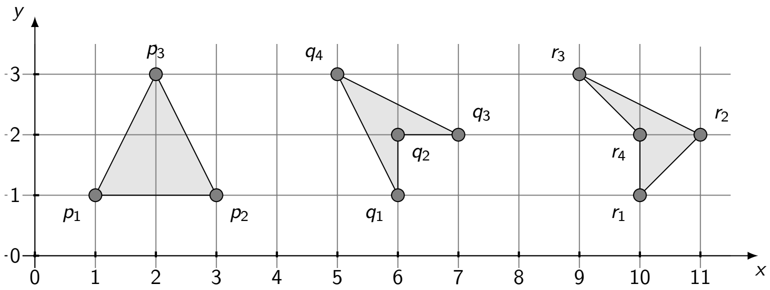
$$\begin{aligned} a &= (1, 0) \\ b &= (4, 1) \\ p &= (2, 3) \\ \mathrm{CCW}(a, b, p) &= (3-0)(4-1) \\ &\quad + (2-1)(1-0) = 8 \\ \text{area of } (a, b, p) &= 4 \end{aligned}$$



Polygons

Definition: Polygon

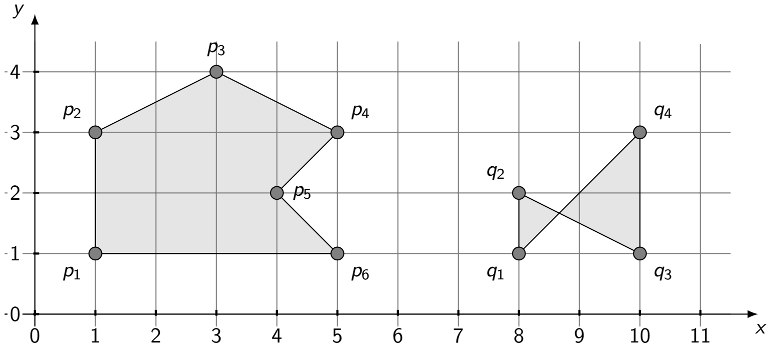
- Plane shape bounded by a finite closed chain of line segments.
- Given by sequence of points $(p_1, p_2, \ldots, p_n)$.
- A point p_i is a *vertex* of the polygon.
- A line segment between p_i and p_{i+1} or p_n and p_1 is an *edge*.



Polygons

- A polygon is *simple* if its edges do not intersect except in the corresponding vertices.
- We will only consider simple polygons.

Simple polygon $(p_1, p_2, p_3, p_4, p_5, p_6)$ **Non-simple polygon** (q_1, q_2, q_3, q_4)



Point-in-polygon problem

Point-in-polygon problem

Given a polygon $(p_1, \ldots, p_n)$ and a point q, determine whether q is inside the polygon.

Point-in-polygon problem

Point-in-polygon problem

Given a polygon $(p_1, \ldots, p_n)$ and a point q, determine whether q is inside the polygon.

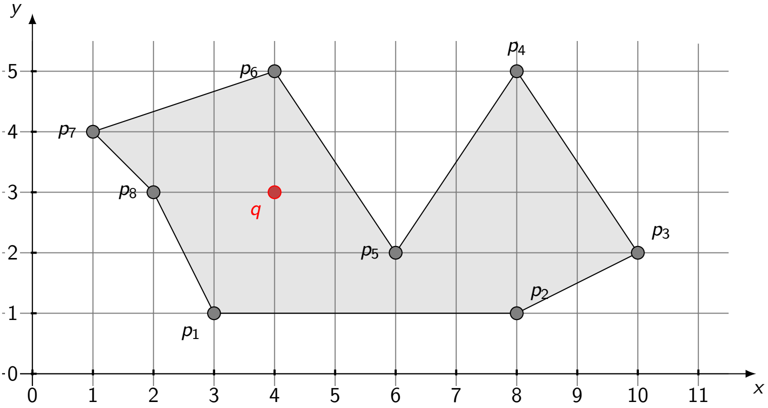
Ray casting algorithm

- Starting from q, cast a ray (a half-line) in a random direction.
- Count number of intersections of ray with edges of polygon.
- If number of intersections is

  odd $\Rightarrow$ point is inside polygon.
  even $\Rightarrow$ point is outside polygon.

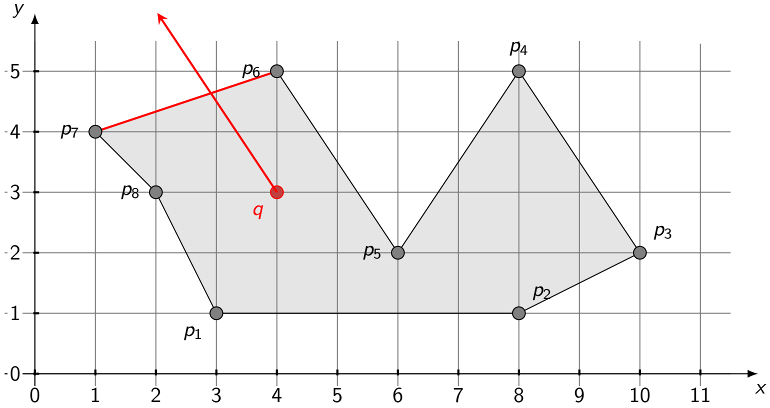
Ray casting algorithm (example)

- Point inside polygon if number of intersections is odd.



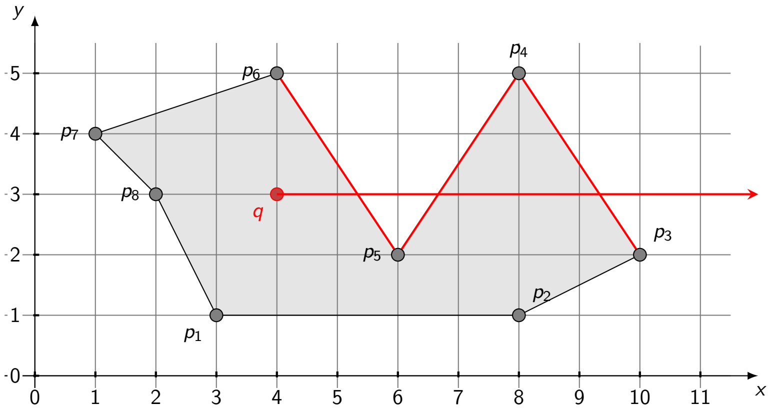
Ray casting algorithm (example)

- Point inside polygon if number of intersections is odd.



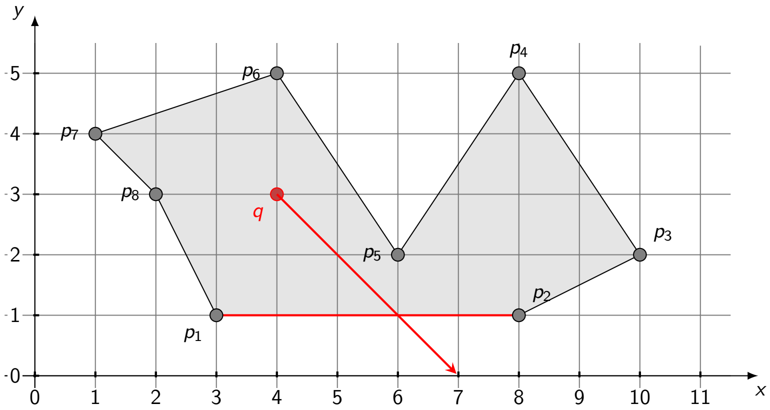
Ray casting algorithm (example)

- Point inside polygon if number of intersections is odd.



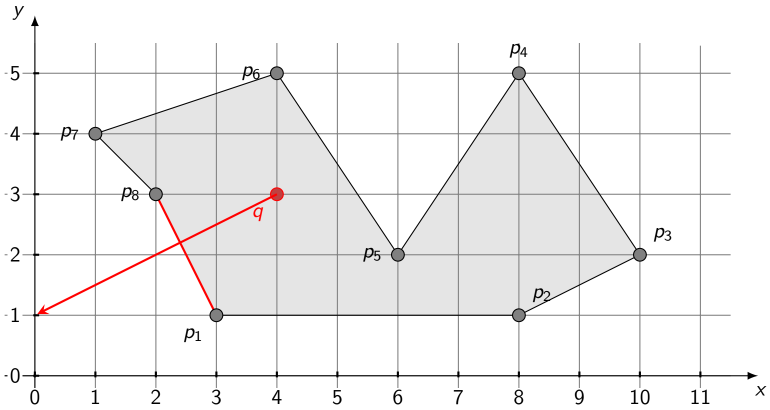
Ray casting algorithm (example)

- Point inside polygon if number of intersections is odd.



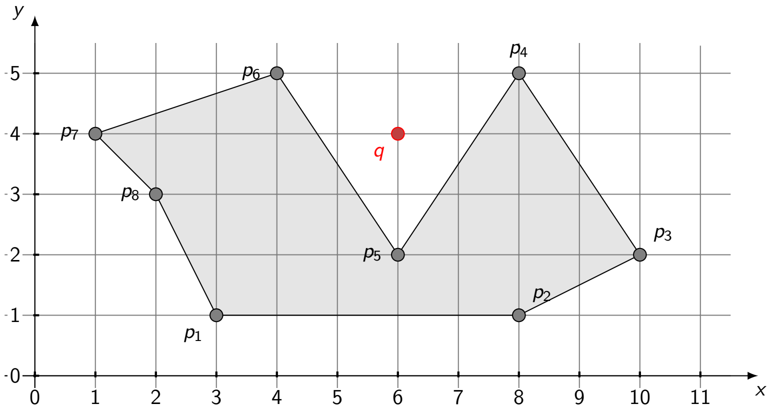
Ray casting algorithm (example)

- Point inside polygon if number of intersections is odd.



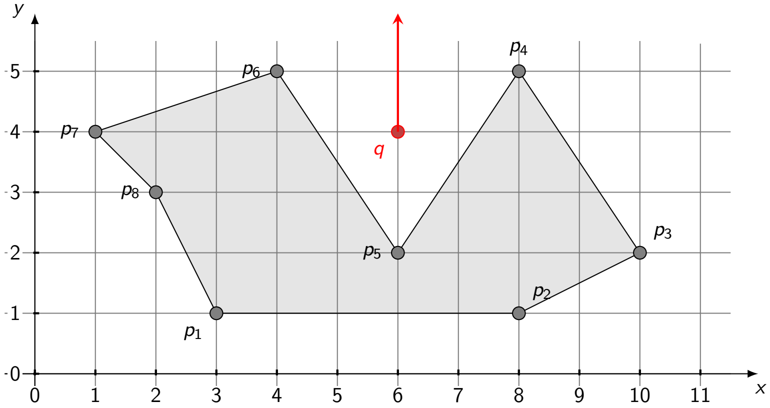
Ray casting algorithm (example)

- Point outside polygon if number of intersections is even.



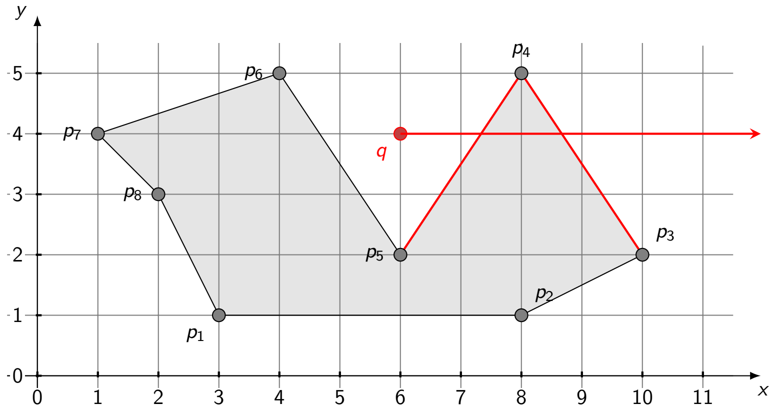
Ray casting algorithm (example)

- Point outside polygon if number of intersections is even.



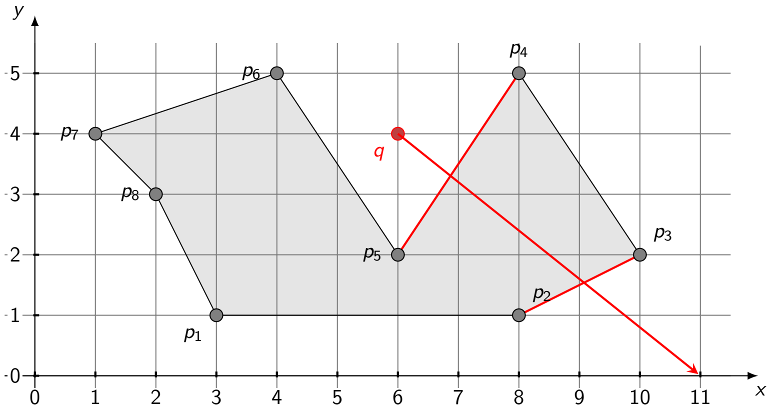
Ray casting algorithm (example)

- Point outside polygon if number of intersections is even.



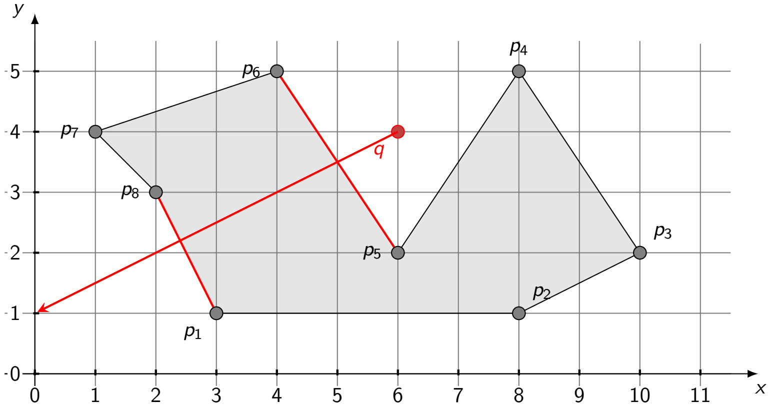
Ray casting algorithm (example)

- Point outside polygon if number of intersections is even.



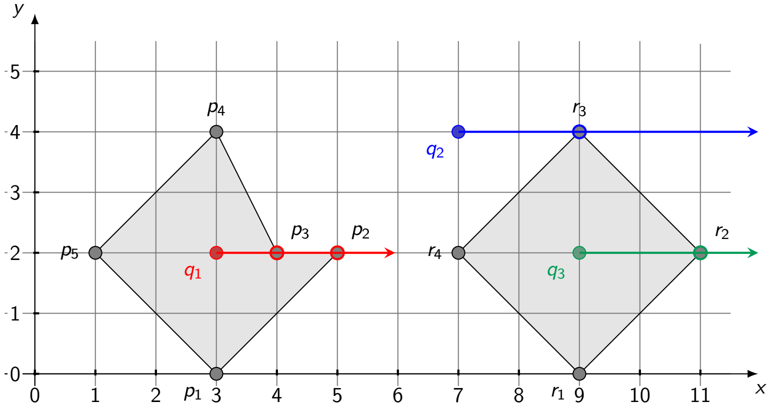
Ray casting algorithm (example)

- Point outside polygon if number of intersections is even.



Ray casting algorithm: special cases

- What to do when hitting a vertex? Hard to handle different cases ⇒ simply repeat with another direction.
- Take care of floating point precision as q may be close to an edge.



Convex sets and hull

Convex set

A set of points P is *convex* if for any two points $x, y \in P$, the line segment between x and y is fully contained in P.

Convex sets and hull

Convex set

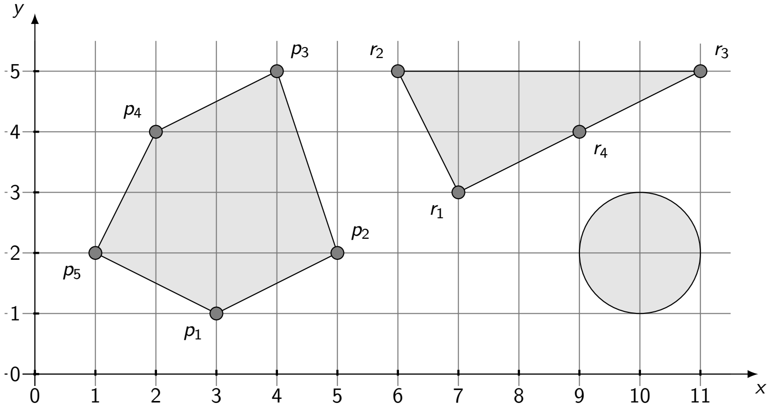
A set of points P is *convex* if for any two points $x, y \in P$, the line segment between x and y is fully contained in P.

Convex hull

Given a set of points P, the *convex hull* of P is the smallest set H such that H is convex and $P \subseteq H$.

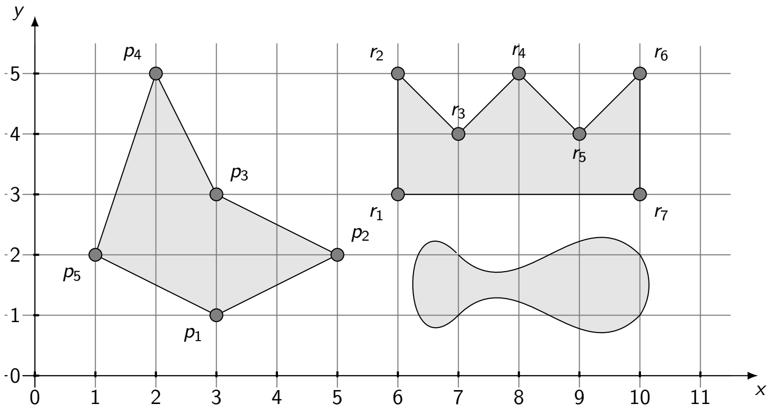
Convex sets and hull (example)

Convex sets



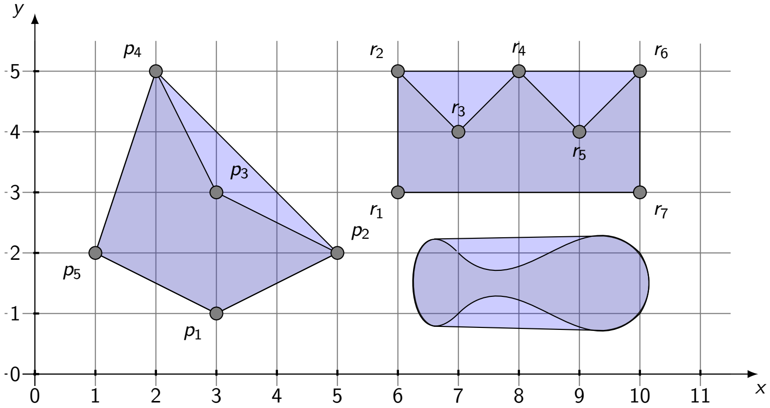
Convex sets and hull (example)

Non-convex sets



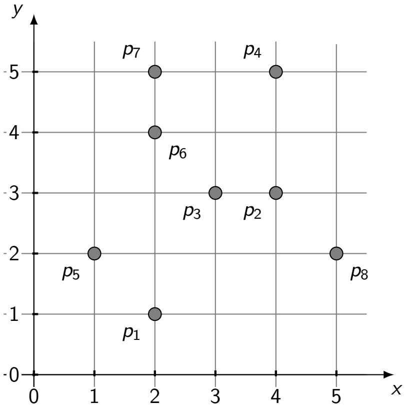
Convex sets and hull (example)

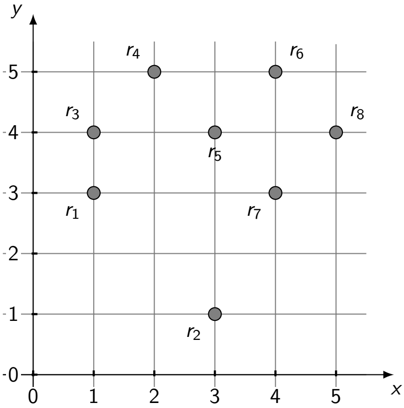
Convex hulls of the non-convex sets



Convex sets and hull (example)

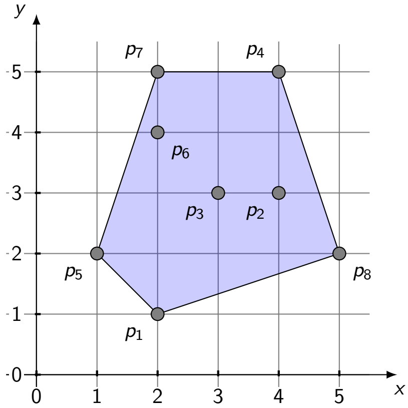
Convex hulls of sets of points

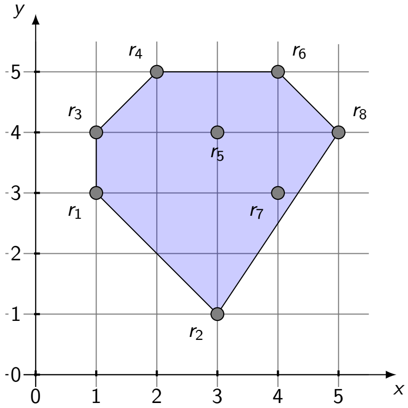




Convex sets and hull (example)

Convex hulls of sets of points





Convex hull and polygons

- Convex hull of a finite set of points is a polygon.
- Convex hull of a polygon is again a polygon.
- Vertices of the convex hull are vertices of the original polygon.
- Convex hull of the polygon is the same as convex hull of its vertices.
- A polygon is convex if and only if its interior angles are at most 180°.
- Given a convex polygon $(p_1, \ldots, p_n)$, a point q is inside the polygon if and only if q is on the same side of all edges $(p_i, p_{(i+1) \bmod n+1})$.

Finding the convex hull

- For finite set of points, we can represent the convex hull by the vertices of a convex polygon.

Finding the convex hull

- For finite set of points, we can represent the convex hull by the vertices of a convex polygon.
- Define the lexicographic order $\prec$ on points such that

$$p \prec q \Leftrightarrow (p_x < q_x) \vee (p_x = q_x \wedge p_y < q_y)$$

Finding the convex hull

- For finite set of points, we can represent the convex hull by the vertices of a convex polygon.
- Define the lexicographic order $\prec$ on points such that

$$p \prec q \Leftrightarrow (p_x < q_x) \lor (p_x = q_x \land p_y < q_y)$$

- Given a finite set of points P, the smallest point p of P regarding $\prec$ is always a vertex of the convex hull of M.

Finding the convex hull

- For finite set of points, we can represent the convex hull by the vertices of a convex polygon.
- Define the lexicographic order $\prec$ on points such that

$$p \prec q \Leftrightarrow (p_x < q_x) \vee (p_x = q_x \wedge p_y < q_y)$$

- Given a finite set of points P, the smallest point p of P regarding $\prec$ is always a vertex of the convex hull of M.
- Given a finite set of points P, a vertex p of its convex hull and a point q of P, (p, q) is an edge of the convex hull if and only if all other points of P lie on the same side of the line (a, b).

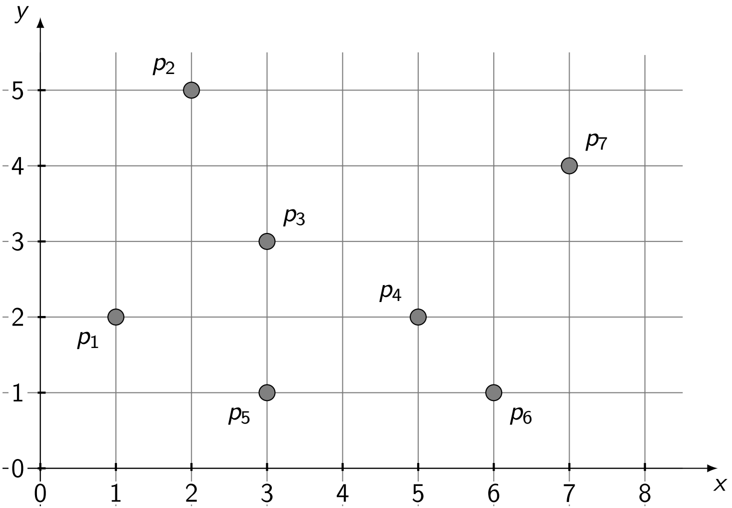
Gift-wrapping algorithm

Algorithm 1 Gift-wrapping algorithm

Input: Set of points $P = \{p_1, \ldots, p_n\}$
Output: Convex hull H of P

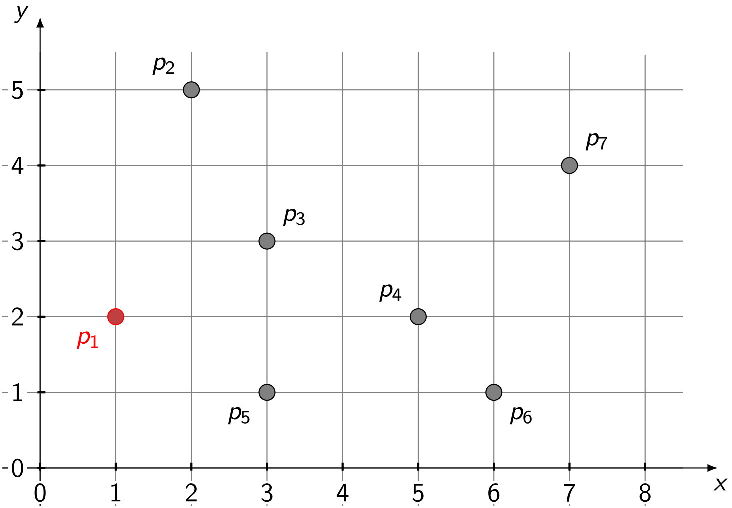
```
  ▷ p is first vertex of convex hull
  p ← min_≺ P
  ▷ H is a list with h elements
  H(1) ← p; h ← 1
  while there is q ∈ P \ H such that all r ∈ P \ {p, q} lie on the same
  side of the line (p, q) do
      ▷ q is a vertex of the convex hull
      H(h + 1) ← q; h ← h + 1
      p ← q
  end while
  return (H(1), ..., H(h))
```

Gift-wrapping algorithm (example)



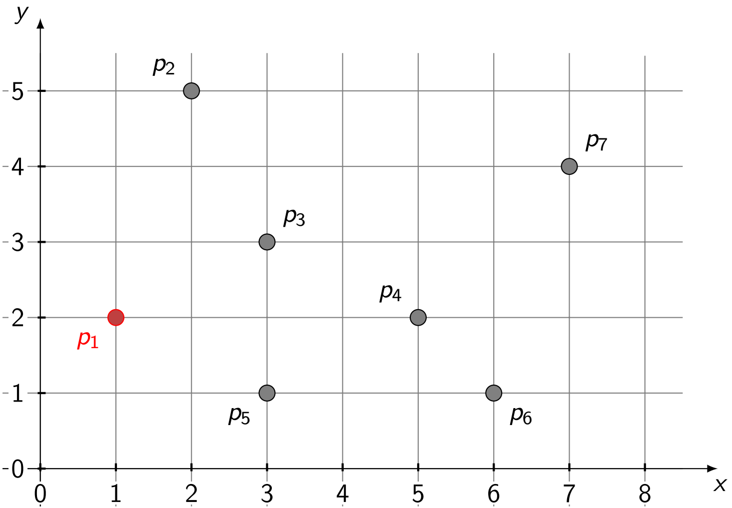
Gift-wrapping algorithm (example)

$p = \min_{\prec} P = p_1$



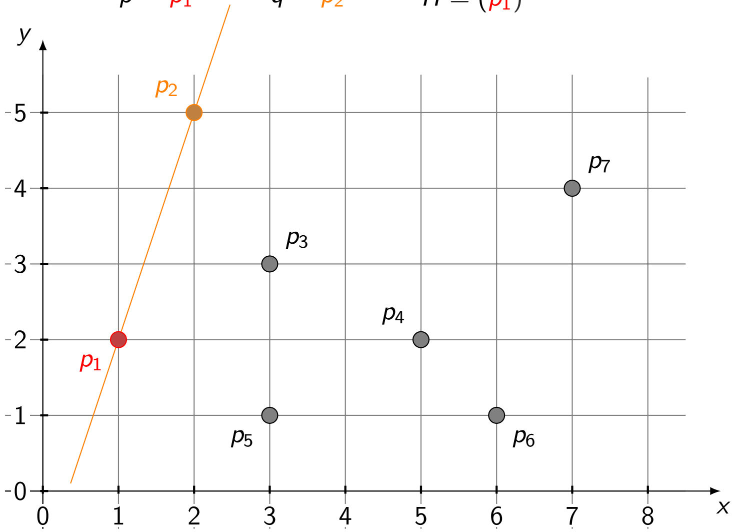
Gift-wrapping algorithm (example)

$p = p_1$ $\qquad$ $H = (p_1)$



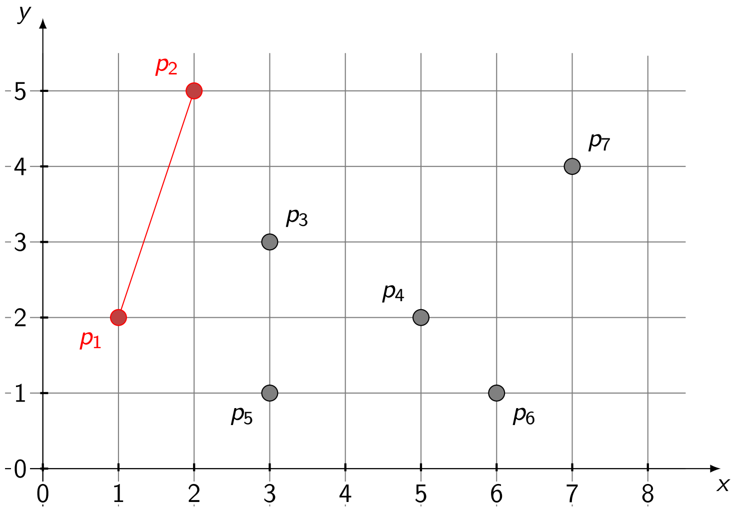
Gift-wrapping algorithm (example)

$p = p_1$ $q = p_2$ $H = (p_1)$



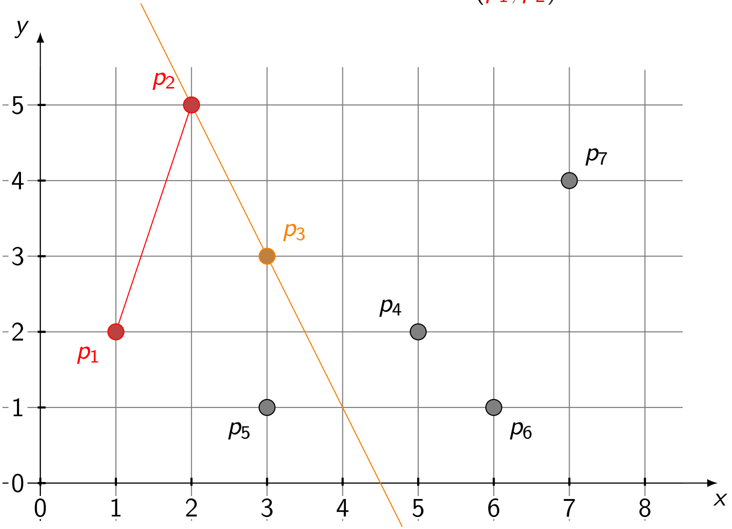
Gift-wrapping algorithm (example)

$p = p_2$ $\qquad$ $H = (p_1, p_2)$



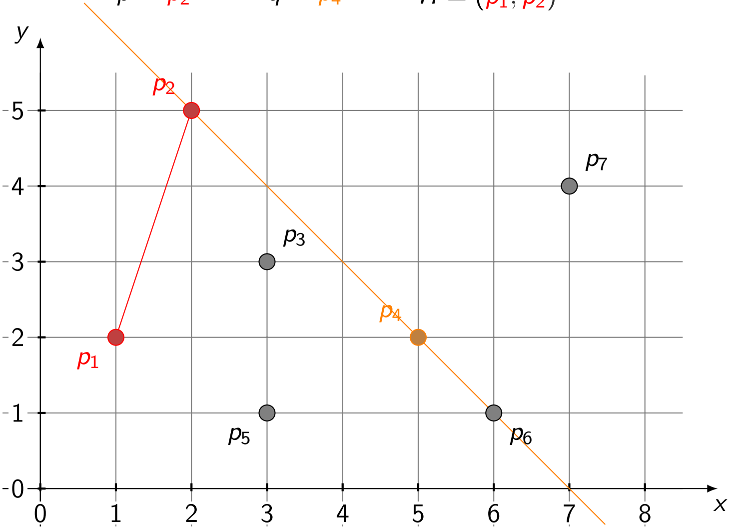
Gift-wrapping algorithm (example)

$p = p_2 \qquad q = p_3 \qquad H = (p_1, p_2)$



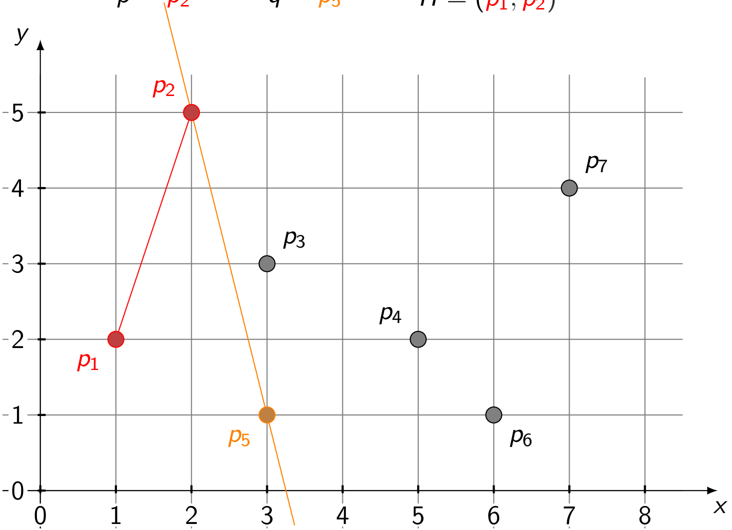
Gift-wrapping algorithm (example)

$p = p_2 \qquad q = p_4 \qquad H = (p_1, p_2)$



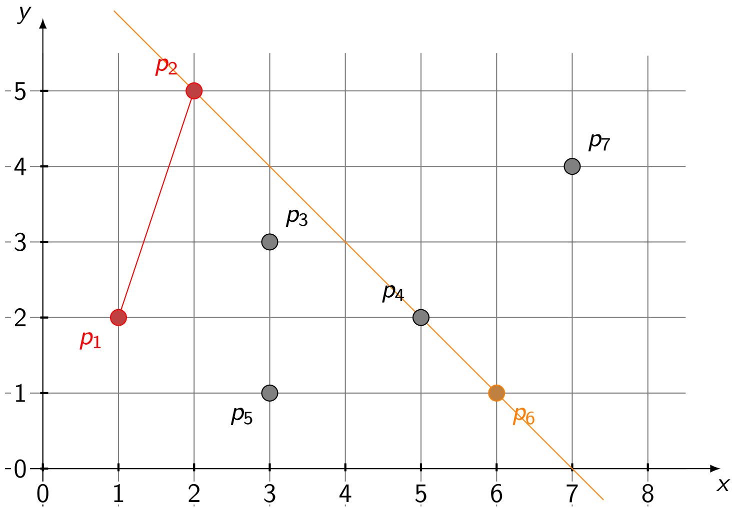
Gift-wrapping algorithm (example)

$p = p_2 \qquad q = p_5 \qquad H = (p_1, p_2)$



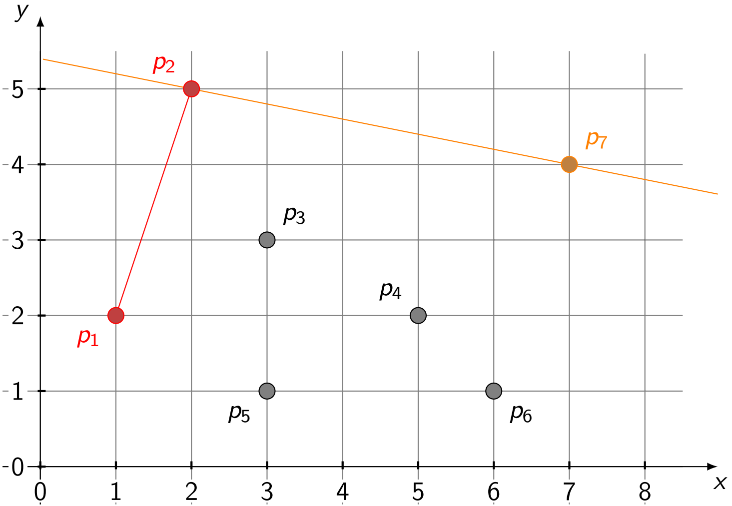
Gift-wrapping algorithm (example)

$p = p_2 \qquad q = p_6 \qquad H = (p_1, p_2)$



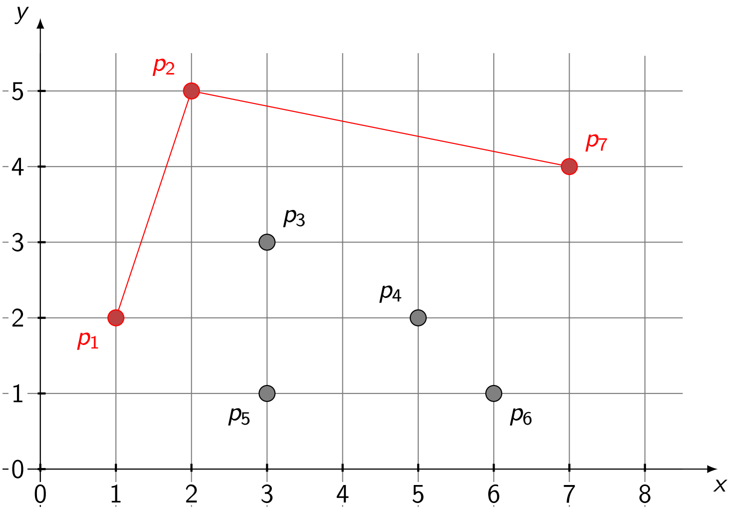
Gift-wrapping algorithm (example)

$p = p_2$ $\quad q = p_7$ $\quad H = (p_1, p_2)$



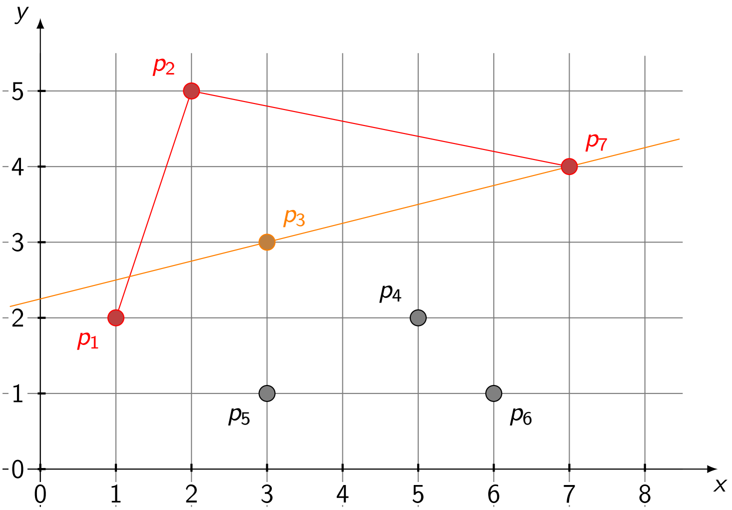
Gift-wrapping algorithm (example)

$p = p_7$ $H = (p_1, p_2, p_7)$



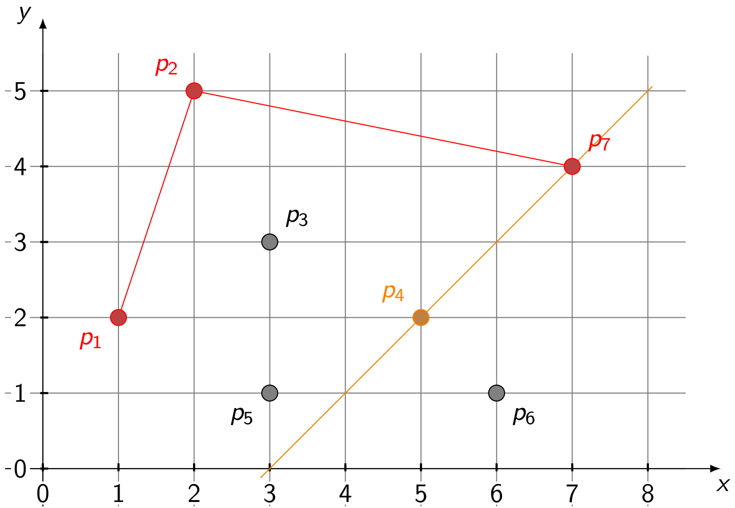
Gift-wrapping algorithm (example)

$p = p_7 \qquad q = p_3 \qquad H = (p_1, p_2, p_7)$



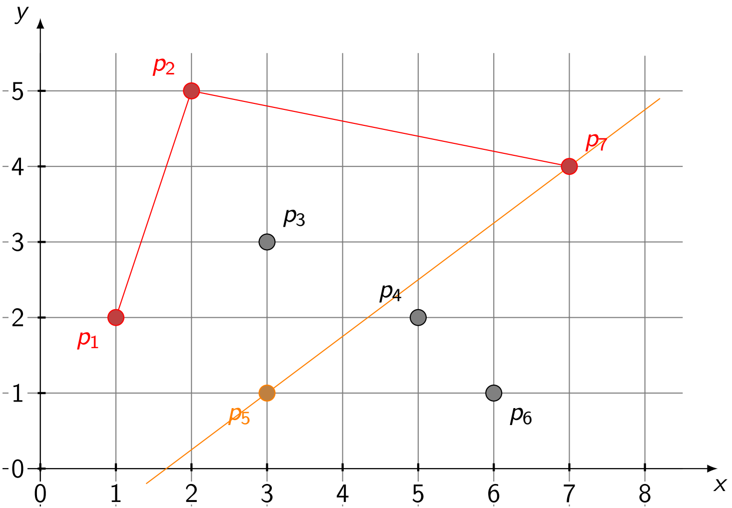
Gift-wrapping algorithm (example)

$p = p_7 \qquad q = p_4 \qquad H = (p_1, p_2, p_7)$



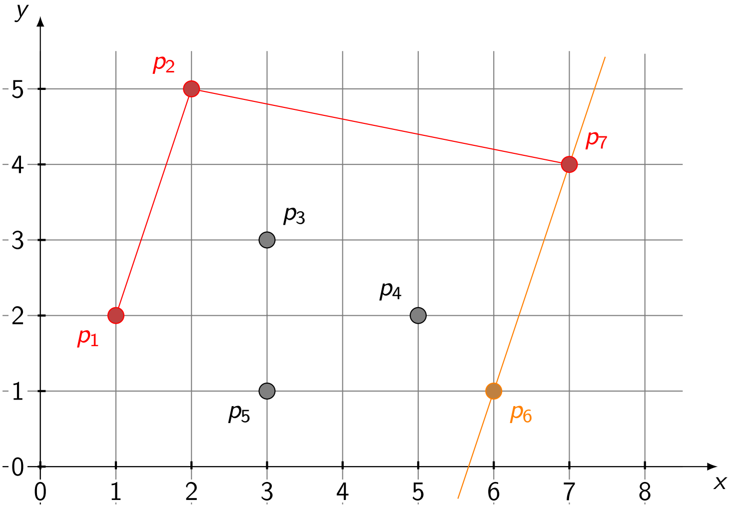
Gift-wrapping algorithm (example)

$p = p_7 \qquad q = p_5 \qquad H = (p_1, p_2, p_7)$



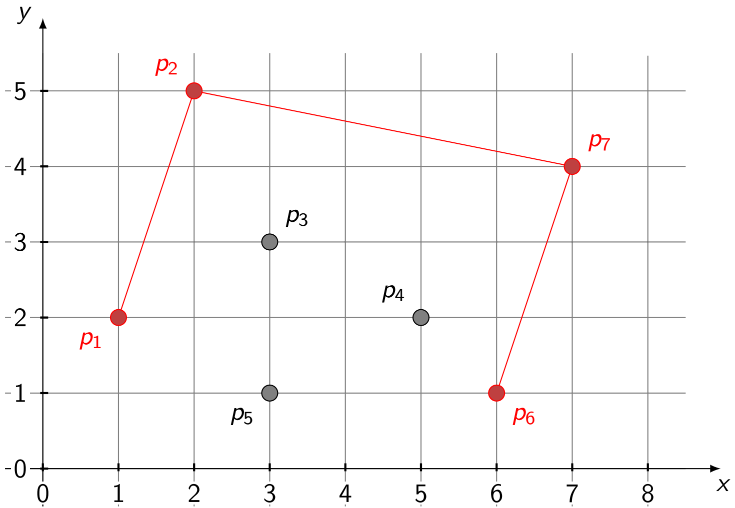
Gift-wrapping algorithm (example)

$p = p_7 \qquad q = p_6 \qquad H = (p_1, p_2, p_7)$



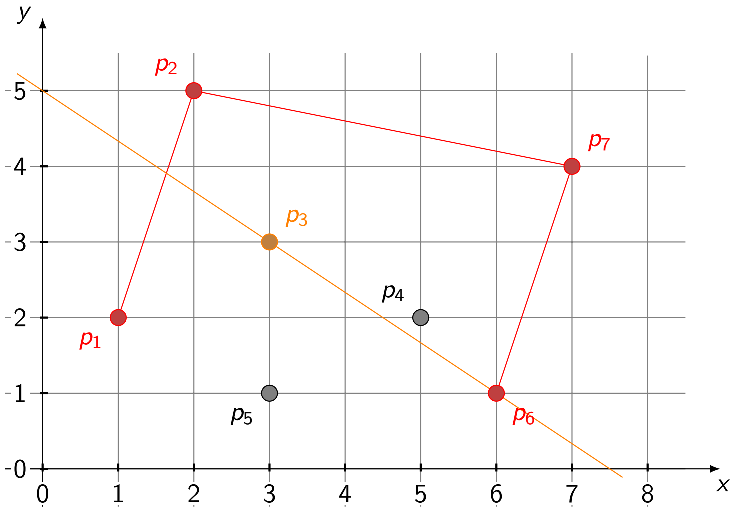
Gift-wrapping algorithm (example)

$p = p_6$ $\qquad$ $H = (p_1, p_2, p_7, p_6)$



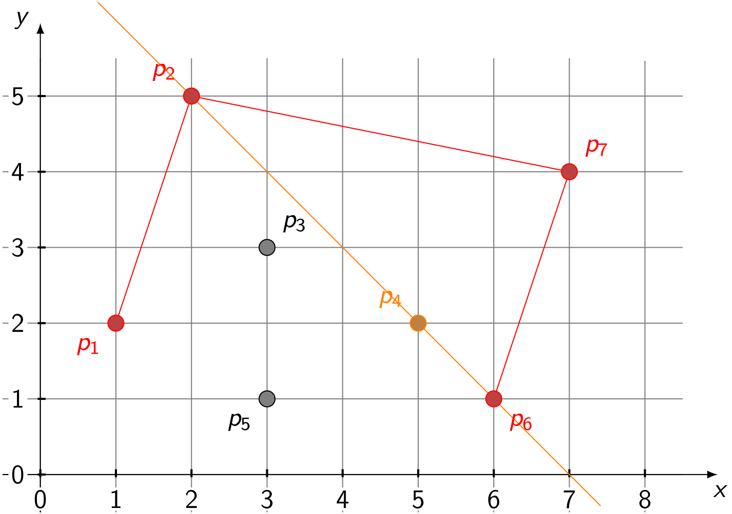
Gift-wrapping algorithm (example)

$p = p_6$ $q = p_3$ $H = (p_1, p_2, p_7, p_6)$



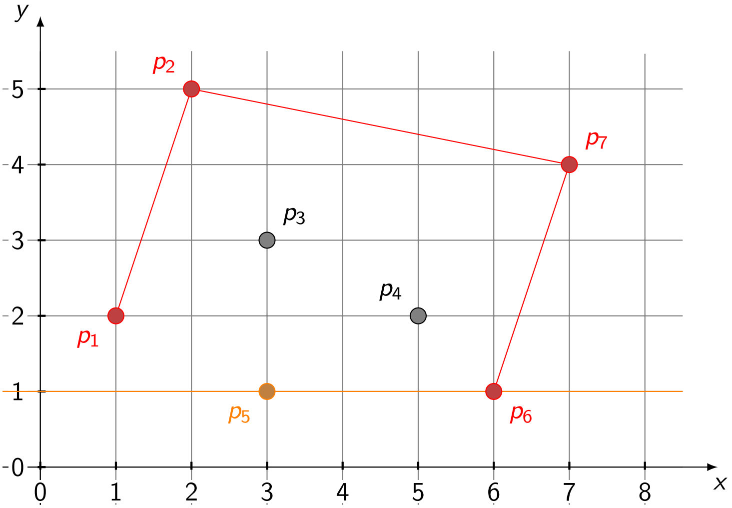
Gift-wrapping algorithm (example)

$p = p_6 \qquad q = p_4 \qquad H = (p_1, p_2, p_7, p_6)$



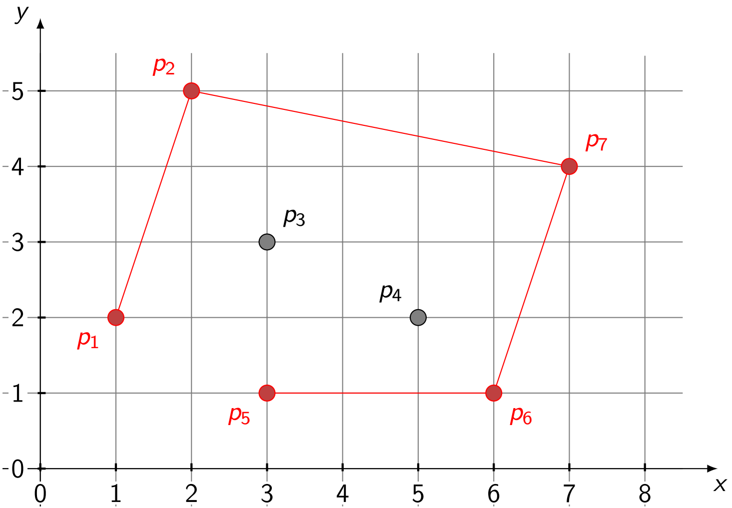
Gift-wrapping algorithm (example)

$p = p_6 \qquad q = p_5 \qquad H = (p_1, p_2, p_7, p_6)$

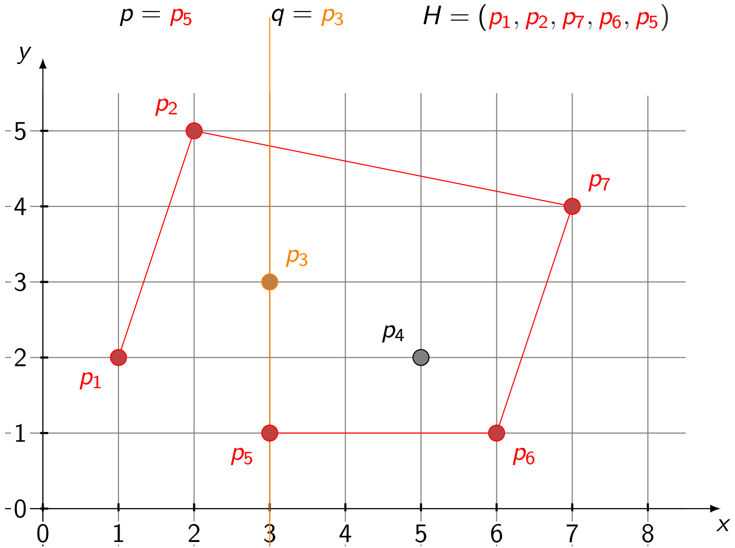


Gift-wrapping algorithm (example)

$p = p_5$ $\qquad$ $H = (p_1, p_2, p_7, p_6, p_5)$

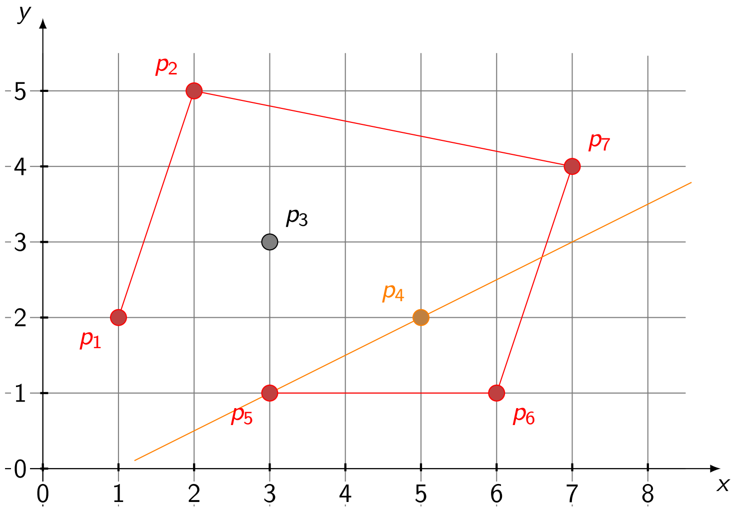


Gift-wrapping algorithm (example)



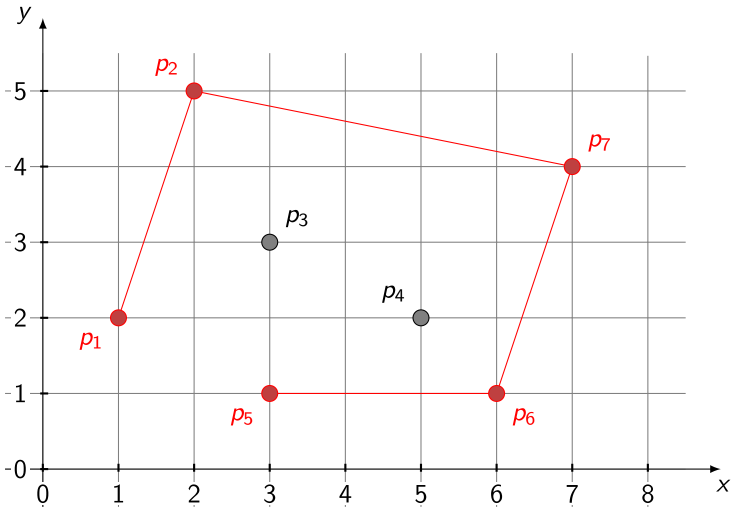
Gift-wrapping algorithm (example)

$p = p_5 \qquad q = p_4 \qquad H = (p_1, p_2, p_7, p_6, p_5)$



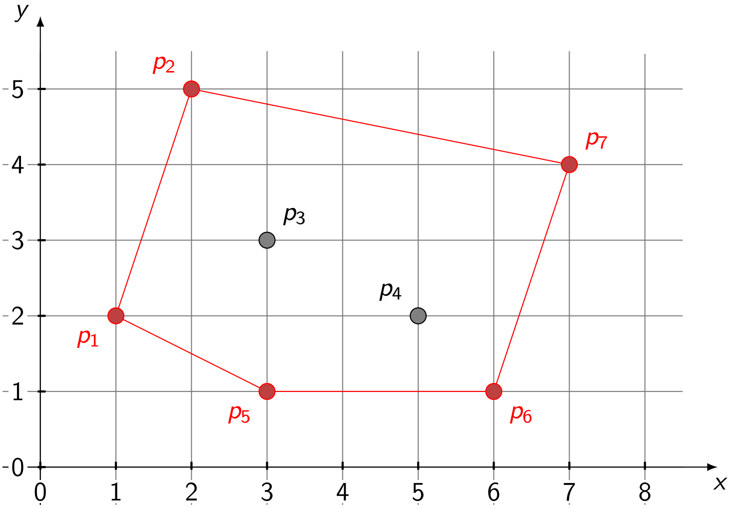
Gift-wrapping algorithm (example)

$p = p_5$ $\qquad$ $H = (p_1, p_2, p_7, p_6, p_5)$



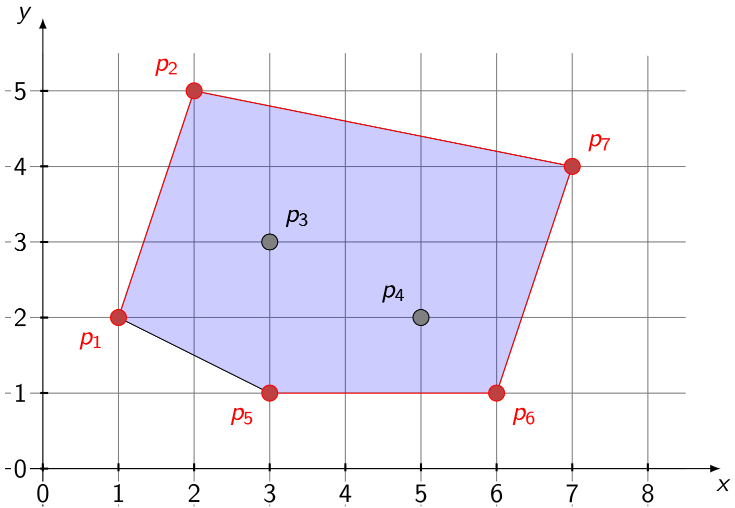
Gift-wrapping algorithm (example)

$p = p_5$ $H = (p_1, p_2, p_7, p_6, p_5)$



Gift-wrapping algorithm (example)

$$H = (p_1, p_2, p_7, p_6, p_5)$$



Gift-wrapping algorithm

- There may be multiple possibilities to choose q if points are colinear $\Rightarrow$ choose e.g. q with shortest distance.

Gift-wrapping algorithm

- There may be multiple possibilities to choose q if points are colinear $\Rightarrow$ choose e.g. q with shortest distance.

Complexity

- $\mathcal{O}(n^3)$ with naive implementation.

Gift-wrapping algorithm

- There may be multiple possibilities to choose q if points are colinear $\Rightarrow$ choose e.g. q with shortest distance.

Complexity

- $\mathcal{O}(n^3)$ with naive implementation.
- $\mathcal{O}(n^2)$: choose point to left/right of (p, q) as new candidate for q.

Gift-wrapping algorithm

- There may be multiple possibilities to choose q if points are colinear $\Rightarrow$ choose e.g. q with shortest distance.

Complexity

- $\mathcal{O}(n^3)$ with naive implementation.
- $\mathcal{O}(n^2)$: choose point to left/right of (p, q) as new candidate for q.
- Total runtime $\mathcal{O}(nh)$ with h the number of points in H.

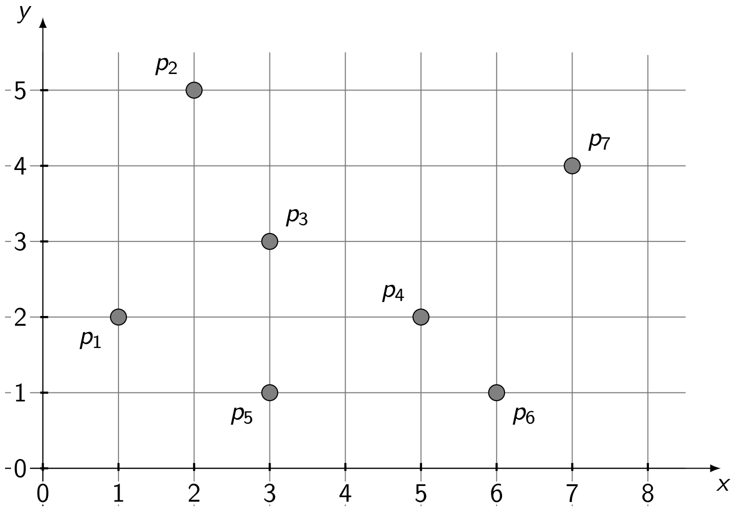
Graham's scan

Algorithm 2 Graham's scan

Input: Set of points $P = \{p_1, \dots, p_n\}$
Output: Convex hull H of P

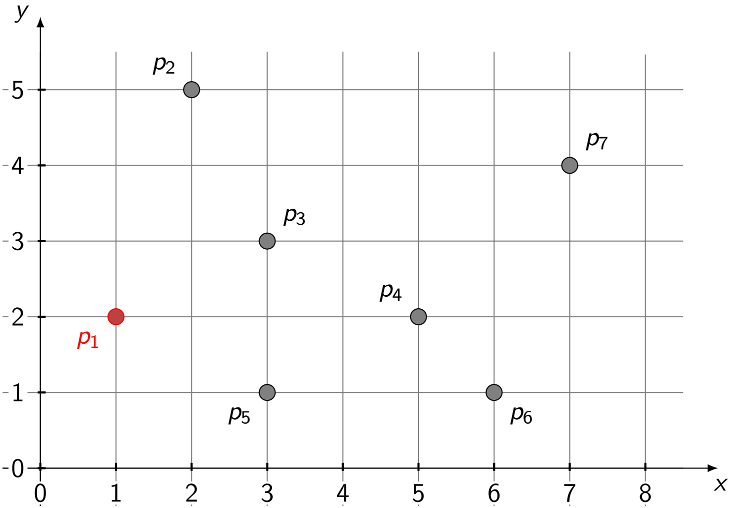
```
p ← min_≺ P                                    ▷ p is first vertex of convex hull
▷ Sort remaining points q ≠ p by angle between line (p, q) and y-axis
Q ← (q_1, q_2, ..., q_n) ordered list of points in P where q_1 = p and
        ∡(Y, p, q_i) ≤ ∡(Y, p, q_j) for 2 ≤ i < j ≤ n with Y = (p_x, p_y + 1)
▷ H is a stack with h elements
H(1) ← q_1; H(2) ← q_2; h ← 2
for i = 3, 4, ..., n do
    while CCW(H(h − 1), H(h), q_i) ≥ 0 do
        ▷ left turn or collinear, H(h) not a vertex of convex hull
        h ← h − 1
    end while
    H(h + 1) ← q_i; h ← h + 1
end for
return (H(1), ..., H(h))
```

Graham's scan (example)



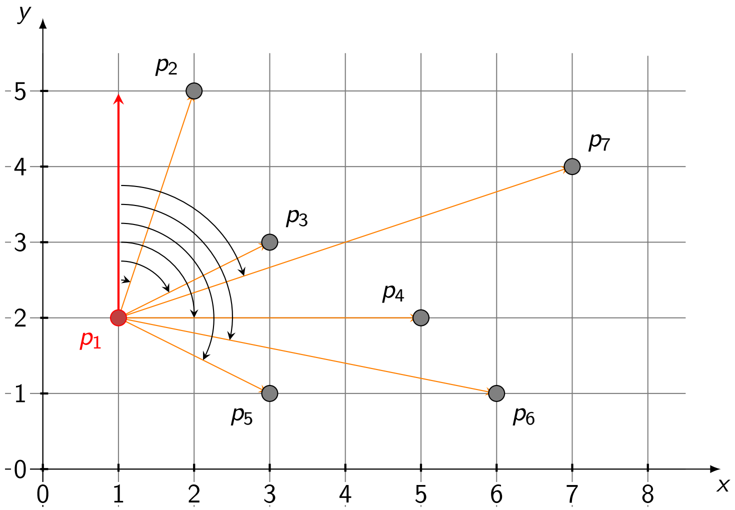
Graham's scan (example)

$$p = \min_{\prec} P = p_1$$



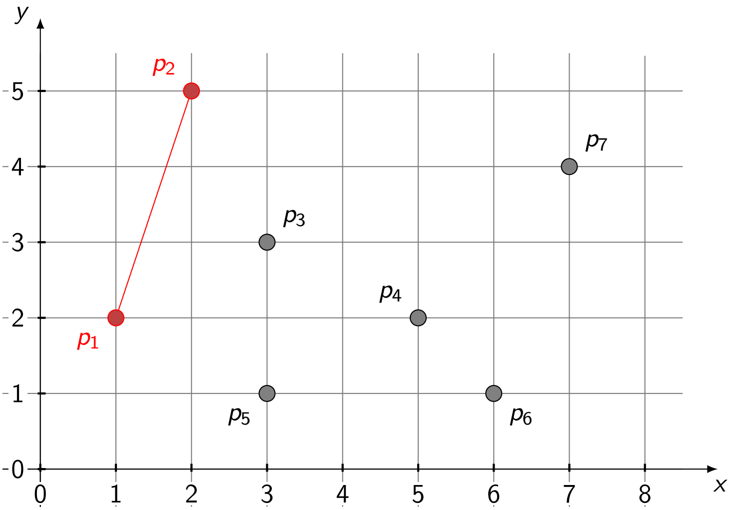
Graham's scan (example)

$$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5)$$



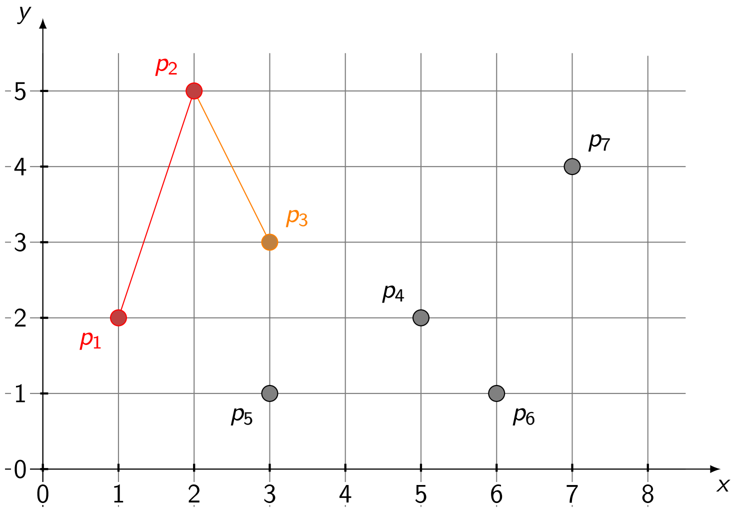
Graham's scan (example)

$$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5) \quad H = (p_1, p_2)$$



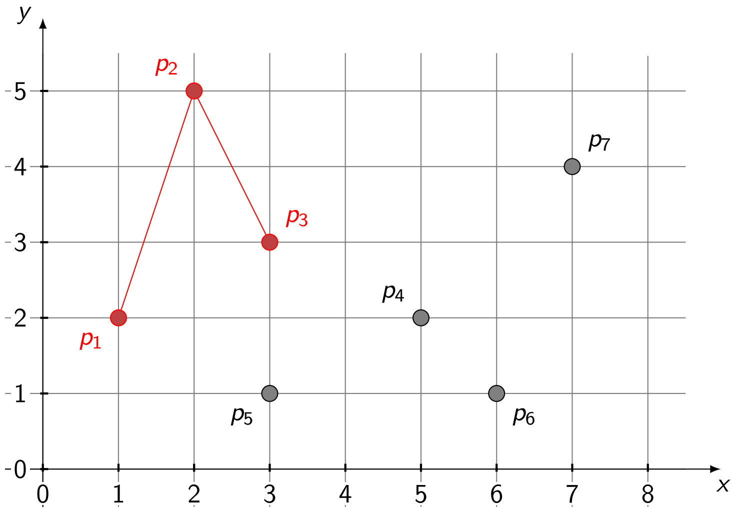
Graham's scan (example)

$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5) \quad H = (p_1, p_2)$



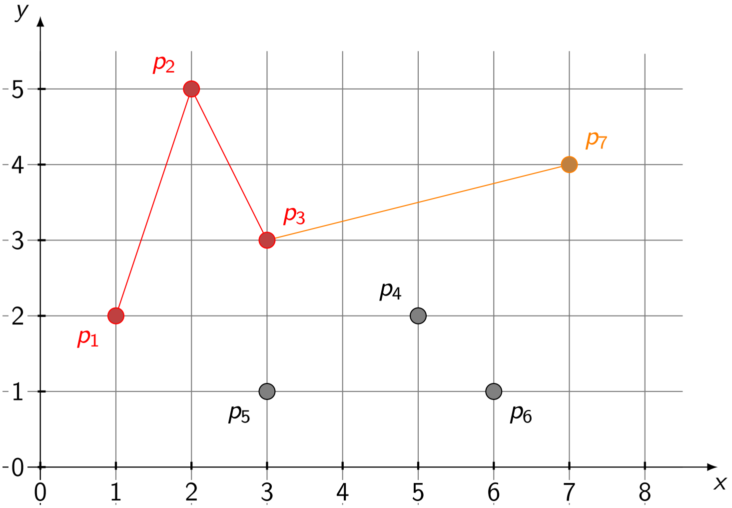
Graham's scan (example)

$$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5) \quad H = (p_1, p_2, p_3)$$



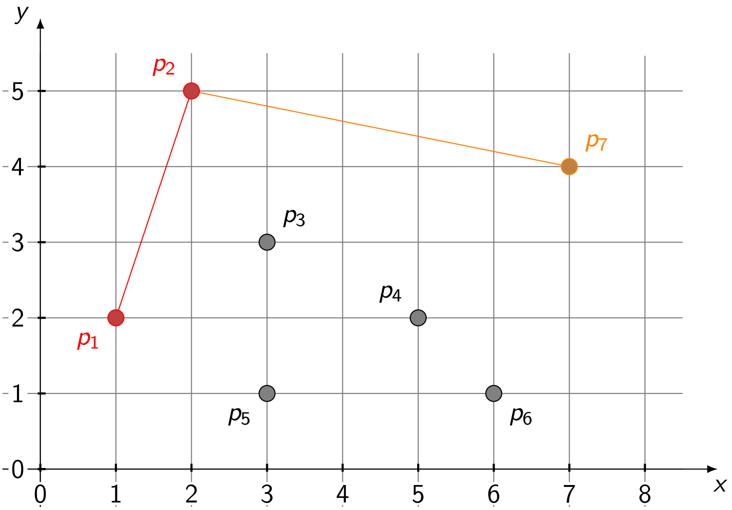
Graham's scan (example)

$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5)$ $\quad H = (p_1, p_2, p_3)$



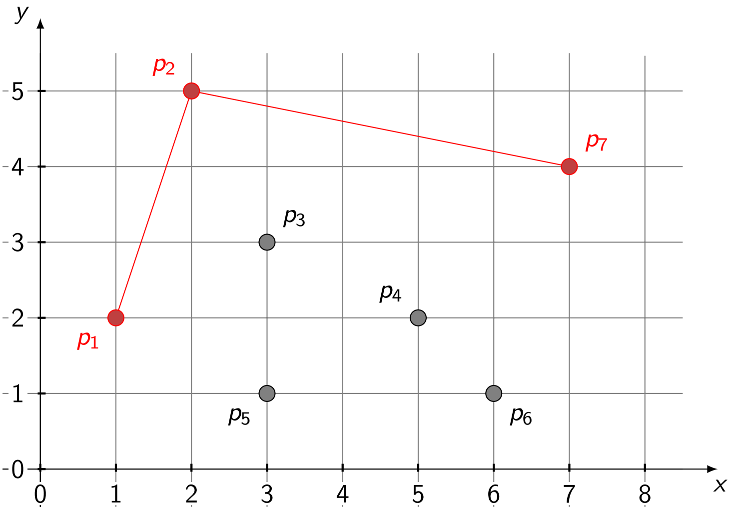
Graham's scan (example)

$$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5) \quad H = (p_1, p_2)$$



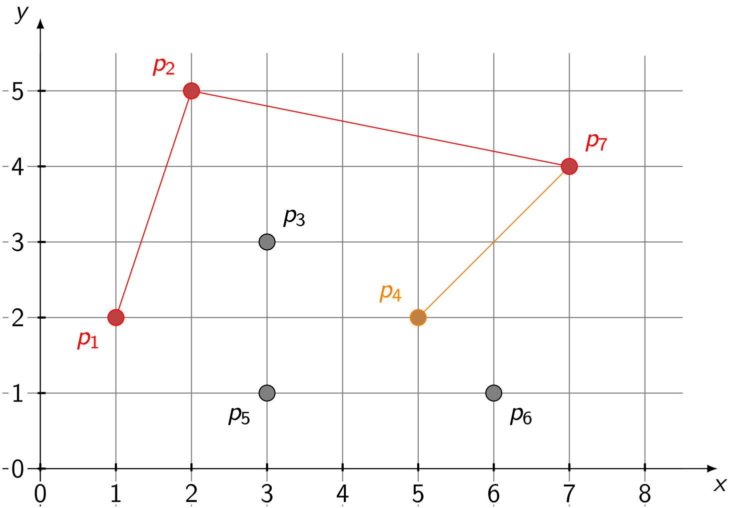
Graham's scan (example)

$$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5) \quad H = (p_1, p_2, p_7)$$



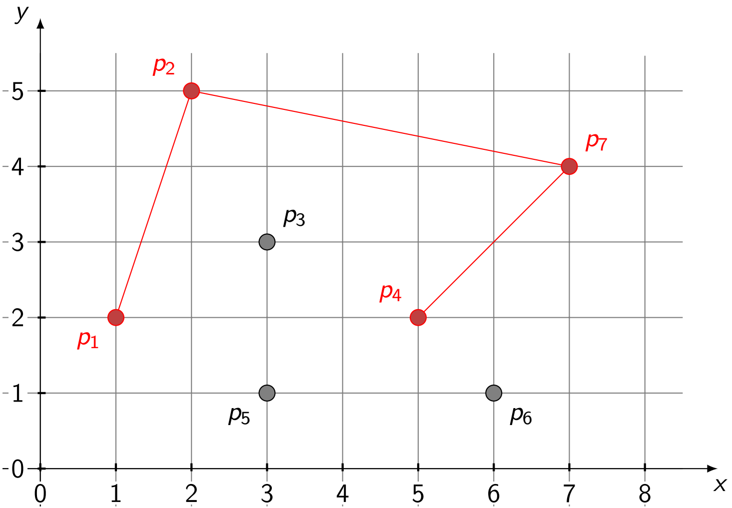
Graham's scan (example)

$$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5) \quad H = (p_1, p_2, p_7)$$



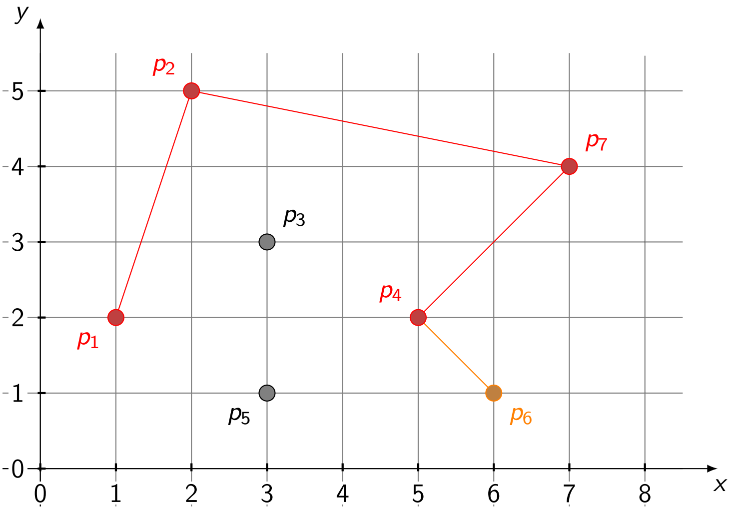
Graham's scan (example)

$$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5) \quad H = (p_1, p_2, p_7, p_4)$$



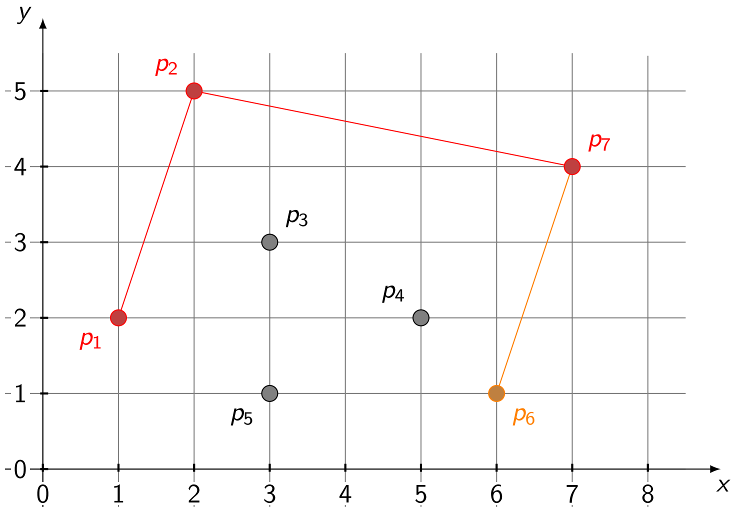
Graham's scan (example)

$$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5) \quad H = (p_1, p_2, p_7, p_4)$$



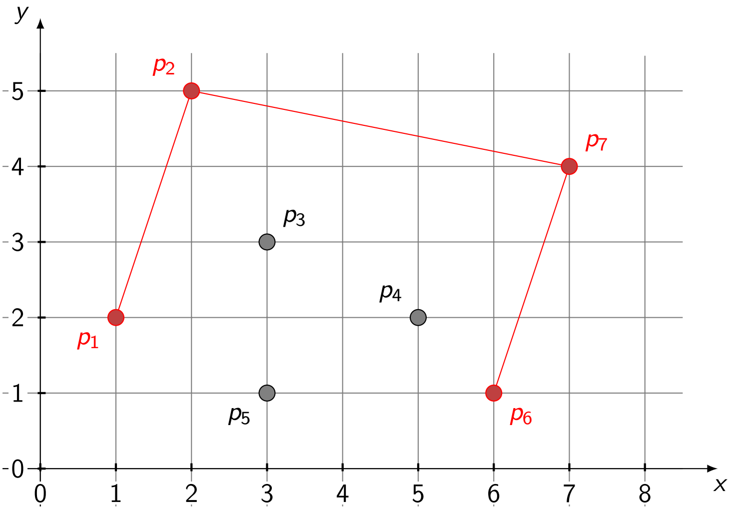
Graham's scan (example)

$$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5) \quad H = (p_1, p_2, p_7)$$



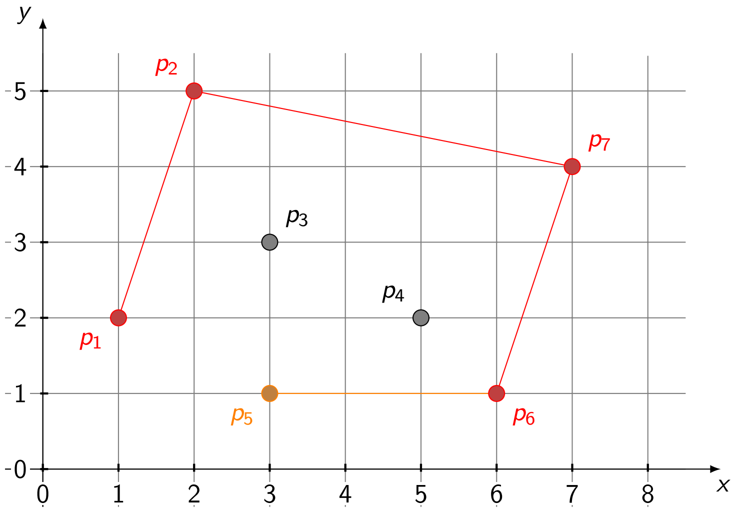
Graham's scan (example)

$$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5) \quad H = (p_1, p_2, p_7, p_6)$$



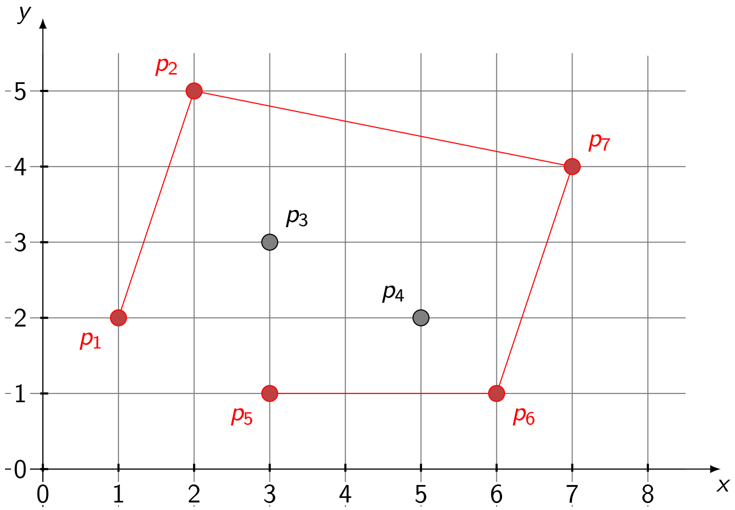
Graham's scan (example)

$$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5) \quad H = (p_1, p_2, p_7, p_6)$$



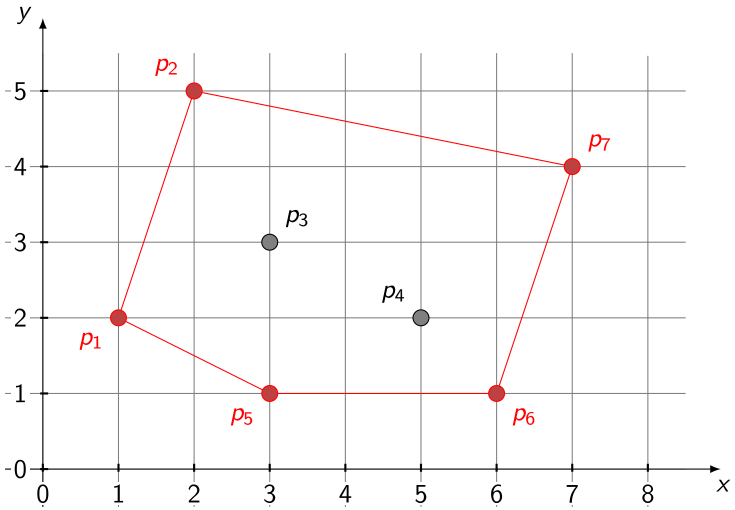
Graham's scan (example)

$$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5) \quad H = (p_1, p_2, p_7, p_6, p_5)$$



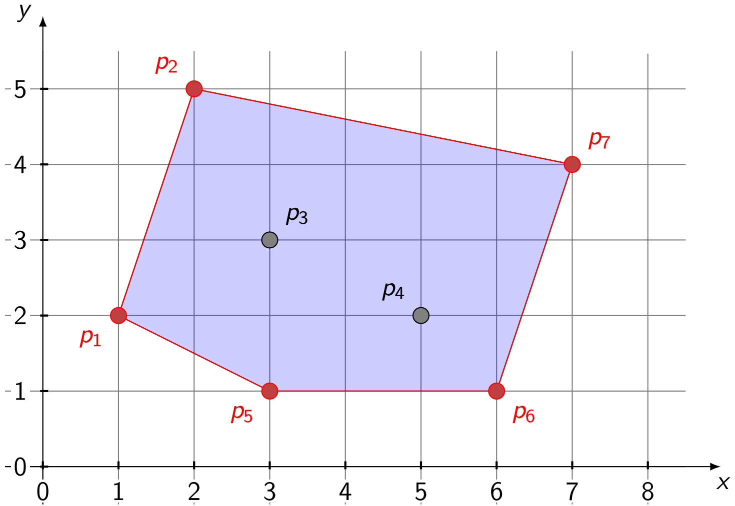
Graham's scan (example)

$$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5) \quad H = (p_1, p_2, p_7, p_6, p_5)$$



Graham's scan (example)

$$Q = (p_1, p_2, p_3, p_7, p_4, p_6, p_5) \quad H = (p_1, p_2, p_7, p_6, p_5)$$



Graham's scan

- When comparing points during sorting, relative orientation by CCW can be used instead of computing the actual angle.

Graham's scan

- When comparing points during sorting, relative orientation by CCW can be used instead of computing the actual angle.

Complexity

- Sorting needs $\mathcal{O}(n \log n)$ time.

Graham's scan

- When comparing points during sorting, relative orientation by CCW can be used instead of computing the actual angle.

Complexity

- Sorting needs $\mathcal{O}(n \log n)$ time.
- Each point is added at most once to H and removed at most once $\Rightarrow$ loop needs $\mathcal{O}(n)$ time

Graham's scan

- When comparing points during sorting, relative orientation by CCW can be used instead of computing the actual angle.

Complexity

- Sorting needs $\mathcal{O}(n \log n)$ time.
- Each point is added at most once to H and removed at most once $\Rightarrow$ loop needs $\mathcal{O}(n)$ time
- Total complexity $\mathcal{O}(n \log n)$.

More algorithms for the convex hull

- Lower bound for output-sensitive convex hull algorithms: $\mathcal{O}(n \log h)$.
- Quickhull: expected time $\mathcal{O}(n \log n)$, but worst case time $\mathcal{O}(nh)$.
- Andrew's Monotone Chain Algorithm: similiar to Graham's scan, but points only need to be sorted by coordinates. Faster in practice.
- Kirkpatrick–Seidel algorithm (1986): $\mathcal{O}(n \log h)$, optimal.
- Chan's algorithm (1996): $\mathcal{O}(n \log h)$, also optimal, but simpler.