Geometric Algorithms

1. Definitions
2. Line-Segment properties
3. Inner Points of polygons
4. Simple polygons
5. Convex hulls
6. Closest pair of points
7. Diameter of a point set

1. Objects

- 2-dimensional plane
- System of coordinates $X - 0 - Y$
- Points $\{p_i\}$, where $p_i = (x_i, y_i)$, $i = 1, 2, \ldots, n$
- Segments $\overline{p_i p_j}$, $1 \le i < j \le n$
- Lines $p_i - p_j$, $1 \le i < j \le n$

If $p_1 = (0, 0)$ then we treat a segment p_1, p_2 as a vector p_2.

We call two segments $\overline{p_1 p_2}$ and $\overline{q_1 q_2}$ intersecting if they have at least one common point.

We define a chain as a set of points $\{p_1, \ldots, p_n\}$ and segments $\{\overline{p_1 p_2}, \ldots, \overline{p_{n-1} p_n}\}$.

A polygon is a chain with $n \geq 3$ and $p_1 = p_n$. A polygon is called simple if no two its non-neighboring segments intersect.

A polygon divides the plane into the inner and the outer parts. A polygon P is called convex if for any two its inner points p and q, every point of the segment $\overline{pq}$ is an inner point of P.

Specific features of Computational Geometry problems:

- A very large number of points (over 10^6).
- A large number of special cases
- Arithmetic operations have different costs:

  Inexpensive: addition, subtraction, comparison, boolean

  Moderate: multiplication

  Expensive: division, powers, arithmetic roots

2. Segment intersection test

The problem:
Instance: Segments $\overline{p_1p_2}$ and $\overline{p_3p_4}$ of positive lengths
Question: Do the segments have a common point?

We apply a two-step method:

1. Quick rejection.
   Let $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$. Denote

$$\hat{x}_1 = \min\{x_1, x_2\}, \qquad \hat{x}_2 = \max\{x_1, x_2\}$$
$$\hat{y}_1 = \min\{y_1, y_2\}, \qquad \hat{y}_2 = \max\{y_1, y_2\}.$$

   Consider the rectangle $P = (\hat{p}_1, \hat{p}_2)$, where $\hat{p}_1 = (\hat{x}_1, \hat{y}_1)$ and $\hat{p}_2 = (\hat{x}_2, \hat{y}_2)$. Similarly, construct a rectangle $Q = (\hat{p}_3, \hat{p}_4)$.
   One has: $P \cap Q = \emptyset \Rightarrow \overline{p_1p_2} \cap \overline{p_3p_4} = \emptyset$. Now, $P \cap Q \neq \emptyset$ iff $(\hat{x_1} \geq \hat{x_3}) \wedge (\hat{x_4} \geq \hat{x_1}) \wedge (\hat{y_1} \geq \hat{y_3}) \wedge (\hat{y_4} \geq \hat{y_1})$.

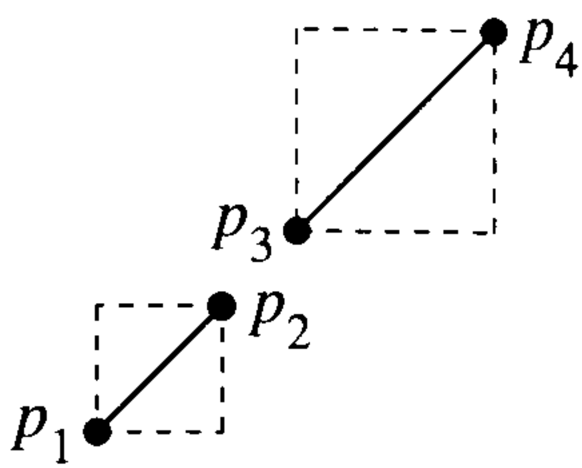

Figure 1: Quick elimination of non-intersecting segments

2. For $q_1 = (x', y')$ and $q_2 = (x'', y'')$ denote

$$q_1 \times q_2 = x'y'' - x''y'$$

One has:

- $\text{sign}((p_3 - p_1) \times (p_2 - p_1)) \neq \text{sign}((p_4 - p_1) \times (p_2 - p_1))$ $\Rightarrow \overline{p_1 p_2}$ and $\overline{p_3 p_4}$ do intersect
- $\text{sign}((p_3 - p_1) \times (p_2 - p_1)) = \text{sign}((p_4 - p_1) \times (p_2 - p_1))$ $\Rightarrow \overline{p_1 p_2}$ and $\overline{p_3 p_4}$ do not intersect

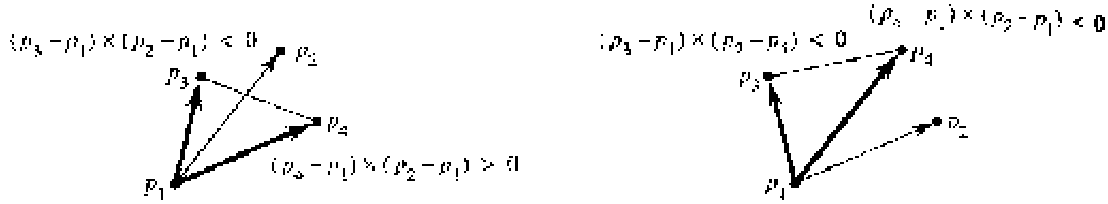

Figure 2: Proper intersection test

- $(p_3 - p_1) \times (p_2 - p_1) \neq 0$ and $(p_4 - p_1) \times (p_2 - p_1) = 0$ $\Rightarrow p_4 \in \overline{p_1 p_2}$
- $(p_3 - p_1) \times (p_2 - p_1) = 0$ and $(p_4 - p_1) \times (p_2 - p_1) = 0$ $\Rightarrow p_3 \in \overline{p_1 p_2}$ and $p_2 \in \overline{p_3 p_4}$

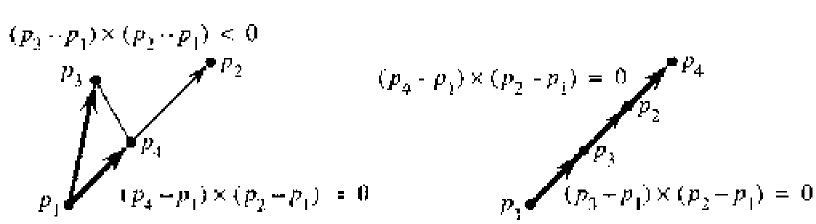

Figure 3: Improper intersection test

3. Inner points of polygons

The problem:
Instance: Polygon P and a point $q = (x, y)$.
Question: Determine whether q is an inner point of P.

Idea: consider an external point s of P and compute how much times does the segment $\overline{qs}$ intersect the boundary of P. If this number is even, then q is an external point, otherwise it is an inner one.

If P consists of points $\{p_0, \ldots, p_{n-1}\}$ and segments $\{e_0, \ldots, e_{n-1}\}$, with $e_i = \overline{p_i p_{(i+1) \bmod n}}$, we consider the vertical line $q - s$.

Algorithm 1 POINT-IN-POLYGON(P, q);
Let s be an external point of P and let $L = q - s$
$c = 0$
for $i = 0$ **to** $n - 1$
  if $e_i \cap L$ is just 1 point /* BE CAREFUL !!! */
    $c = c + 1$
if c is odd
  Inside = TRUE
else
  Inside = FALSE
return c

This method will have a problem if $e_i \cap L \in \overline{\{p_i, p_{(i+1) \bmod n}\}}$, but it is easy to fix it.

The running time of the algorithm is $O(n)$.

4. Simple Polygons

Instance: Set of points $\{p_1, \ldots, p_n\}$.
Problem: Construct a simple polygon with nodes in these points.

Let $n \geq 3$ and assume p_1, p_2, p_3 are not collinear. We set the origin 0 in an inner point of the triangle p_1, p_2, p_3 (i.e. in its center of gravity).

Furthermore, define the order of points $\mathcal{L}$ as follows:
$p_i \leq_{\mathcal{L}} p_j \iff$

(i) $\angle(p_1, 0, p_i) < \angle(p_1, 0, p_j)$ ($\angle$ is the angle), or

(ii) if $\angle(p_1, 0, p_i) = \angle(p_1, 0, p_j)$, then dist$(0, p_i) <$ dist$(0, p_j)$.

Algorithm 2 POLYGON$(p_1, \ldots, p_n)$;

1. Sort the points $p_1, \ldots, p_n$ w.r.t the order $\mathcal{L}$.
2. **for** $i = 1$ to $n - 1$
   Connect the point number i in Order $\mathcal{L}$
   with point number $i + 1$ mod n in $\mathcal{L}$
3. **return** Polygon

The running time of POLYGON is determined by sorting and is $O(n \log n)$.

The Convex Polygon Inclusion Problem: (CPI)
Given a convex polygon P and a point z. Determine whether z is an inner point of P.

Proposition 1 *The CPI-Problem is solvable in $O(\log n)$ with a $O(n)$ preprocessing.*

Proof. Let q be an inner point of the polygon $\{p_1, \ldots, p_n\}$. The half-infinite lines $q - p_i$ split the plane in n sectors.

We introduce the polar coordinate system (r, φ) originated in q and compute the angles $\angle(p_i)$, $i = 1, \ldots, n$.

Algorithm 3

1. Find i with $\angle(p_i) \leq \angle(z) < \angle(p_{i+1})$.
2. If the segments $\overline{q, z}$ and $\overline{p_i, p_{i+1}}$ intersect properly,
   then z is an external point.
   Otherwise z is an inner point.

A polygon $P = \{p_1, \ldots, p_n\}$ is called star polygon, if there exists an inner point q of P, such that each point of the segment $\overline{q, p_i}$ is an inner point of P.

Corollary 1 *The CPI problem for star polygons is solvable in time $O(\log n)$ with $O(n)$ preprocessing time.*

5. Convex Hull

Definition 1 *The convex hull $CH(Q)$ for a set of points Q is the minimum convex polygon that contains all points of Q.*

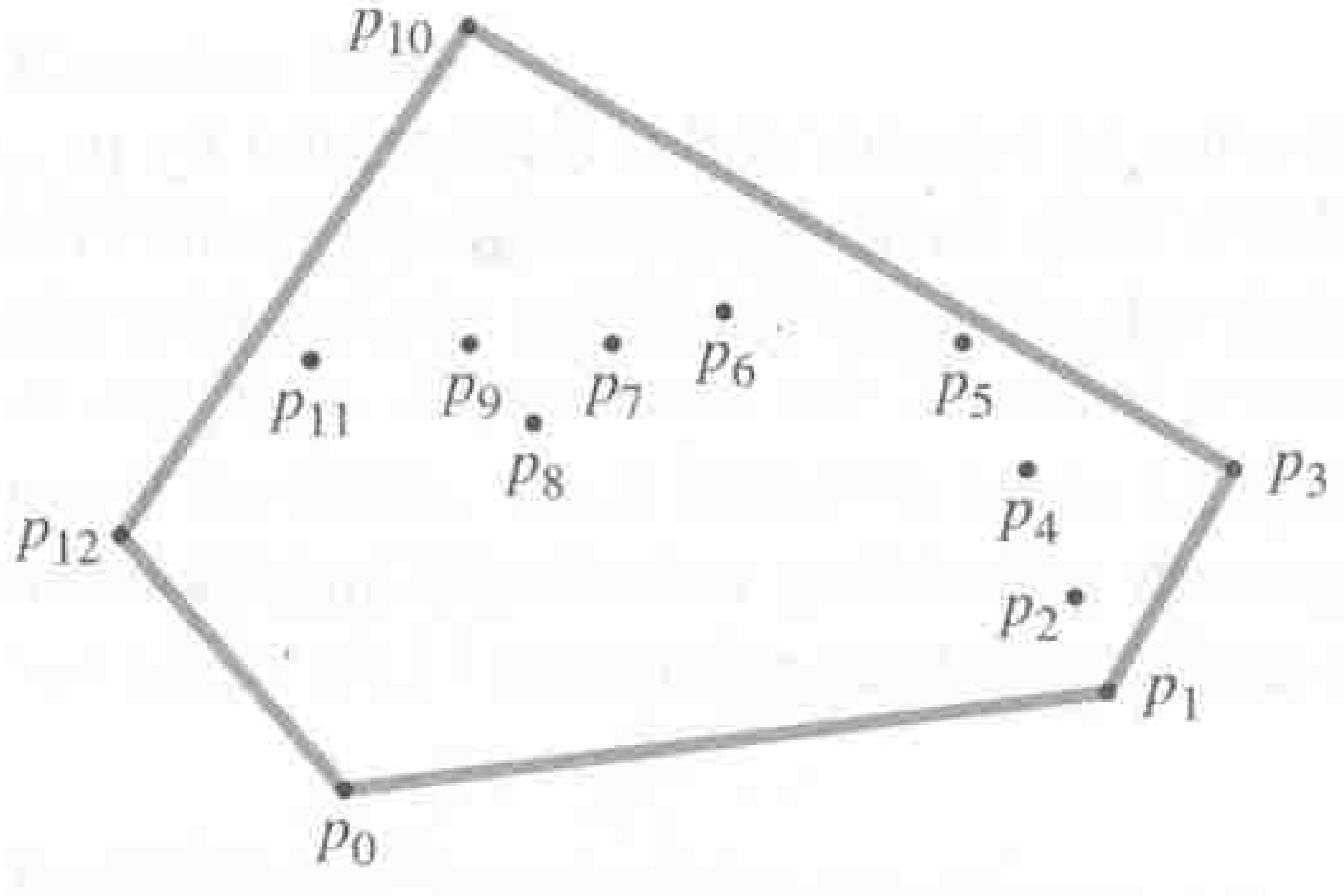

Figure 4: Example of a convex hull

Trivial solution: $O(n^2)$ operations. We design an algorithm with the running time $O(n \log n)$. The fastest known method requires just $O(n \log(|CH(Q)|))$ operations.

Remark 1 *Let p_0, p_1, p_2 be points. One has:*

$$(p_1 - p_0) \times (p_2 - p_0) : \begin{cases} > 0 & \Leftrightarrow \overrightarrow{p_0p_2} \text{ is counterclockwise to } \overrightarrow{p_0p_1} \\ = 0 & \Leftrightarrow \overrightarrow{p_0p_1} \text{ and } \overrightarrow{p_0p_2} \text{ are collinear} \\ < 0 & \Leftrightarrow \overrightarrow{p_0p_2} \text{ is clockwise to } \overrightarrow{p_0p_1} \end{cases}$$

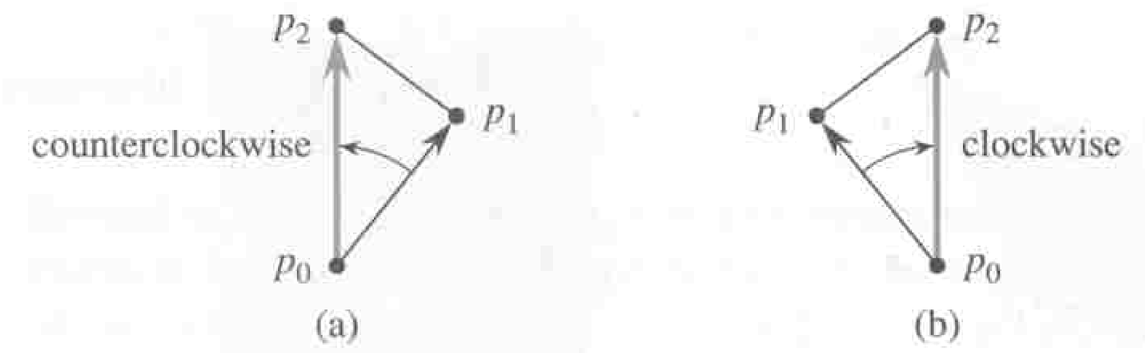

Figure 5: Checking for clockwise position

Let $Q = \{p_0, \ldots, p_n\}$ ($n \geq 2$) and let p_0 be the point with the minimum y-coordinate. If there exist several such points, take as p_0 the one that has the minimum x-coordinate.

We assume that the points $\{p_1, \ldots, p_n\}$ are sorted according to the angles of vectors $\overrightarrow{p_0 p_1}$ in the counterclockwise order. If more than one point has the same angle with p_0 we leave in Q only the one with the maximum distance from p_0.

We use a stack S. The procedure $\textsc{Top}(S)$ returns the top element of the stack without modifying it. Similarly, the procedure $\textsc{Next-to-Top}(S)$ returns the second top element of S.

Algorithm 4 $\textsc{Graham-Scan}(Q)$;

```
PUSH(p_0, S)
PUSH(p_1, S)
PUSH(p_2, S)
 for i = 3  to m                // here m ≤ n
     while p_i is to the right of NEXT-TO-TOP(S),TOP(S)
        POP(S)
    PUSH(p_i, S)
 return S
```

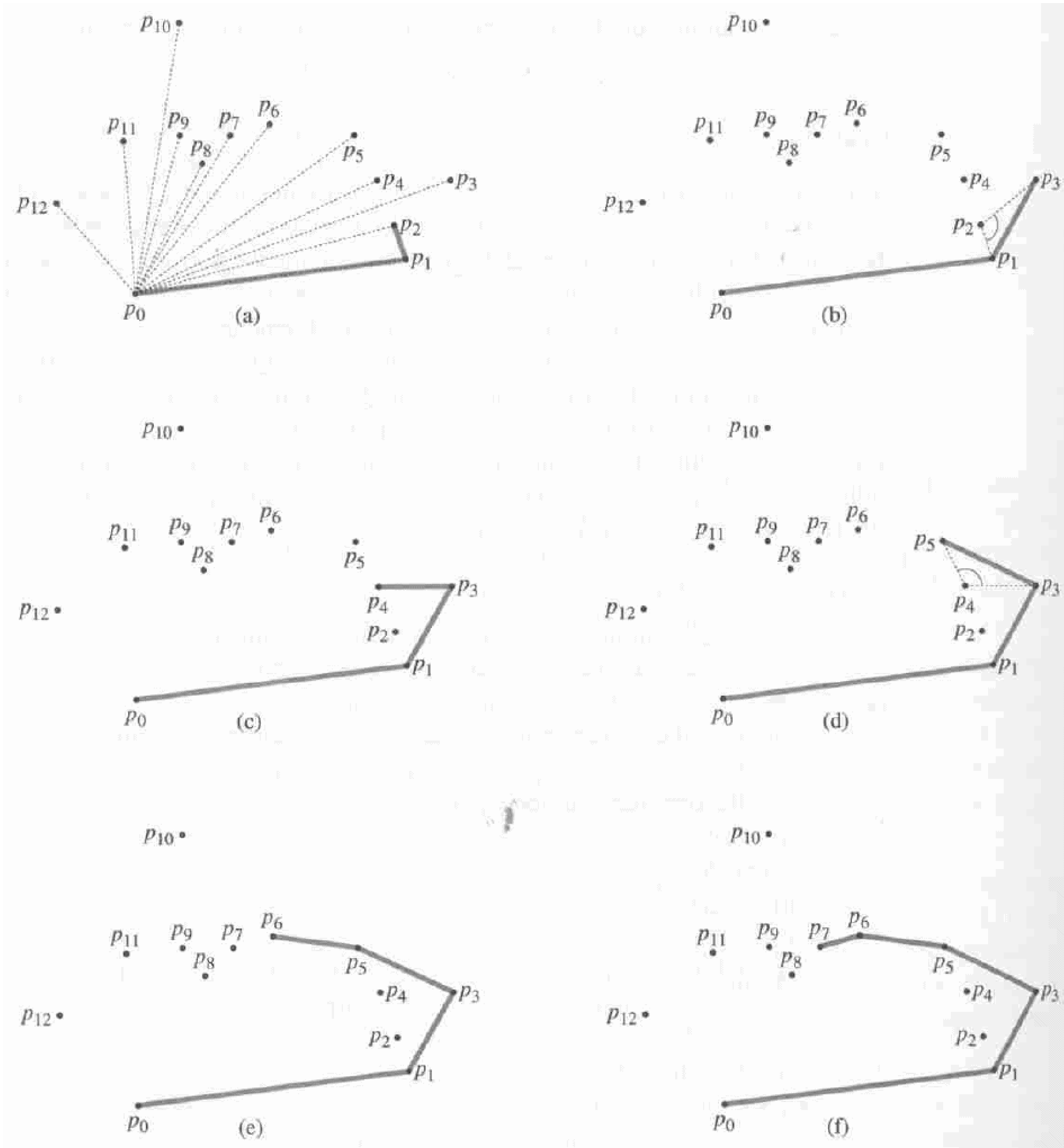

Figure 6: The Graham scan algorithm

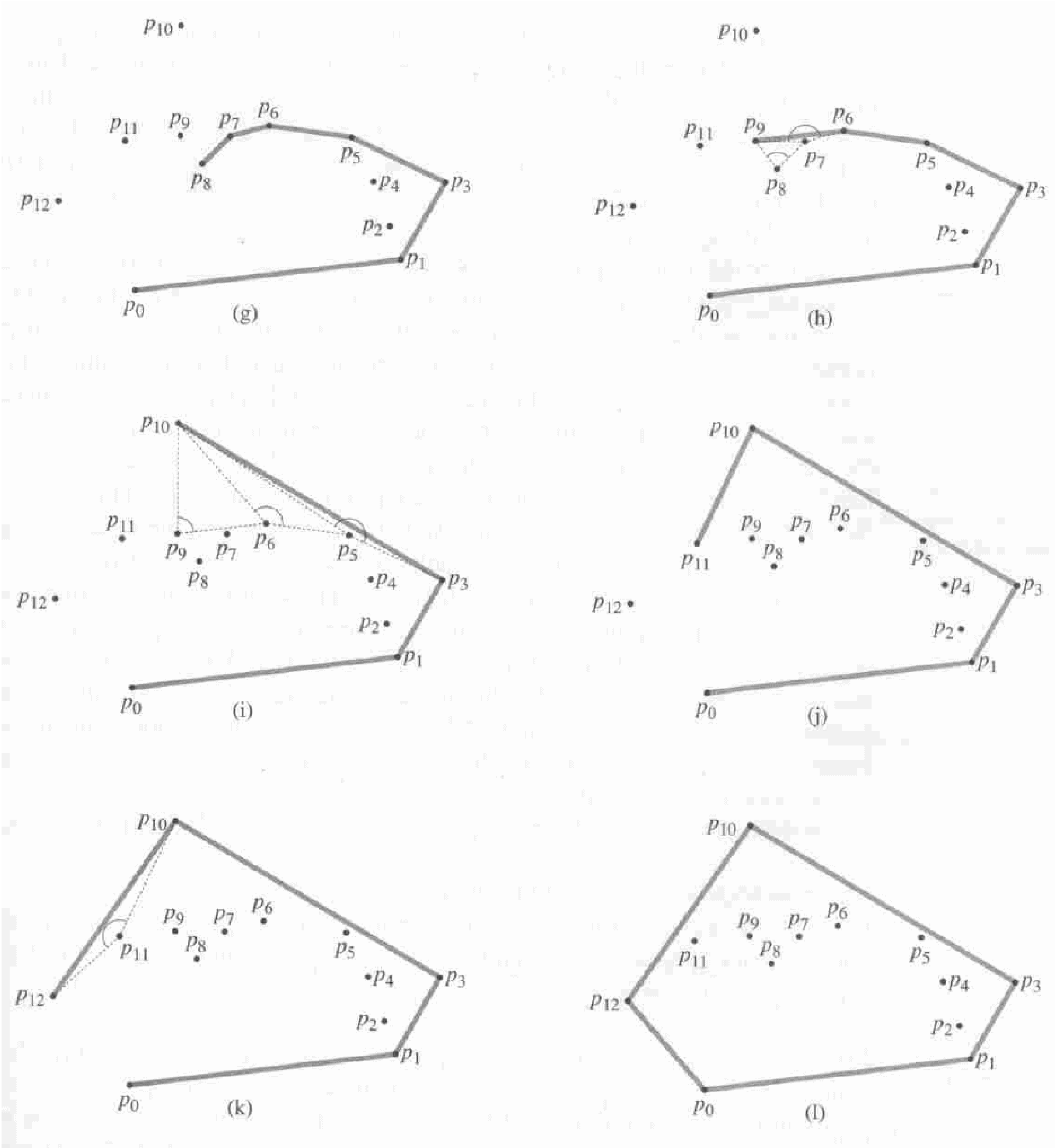

Figure 7: The Graham scan algorithm (continued)

Theorem 1 *A point p is in stack S of the* GRAHAM-SCAN *algorithm iff $p \in CH(Q)$.*

Proof.

We show $p_t \notin S \Rightarrow p_t \notin CH(Q)$.

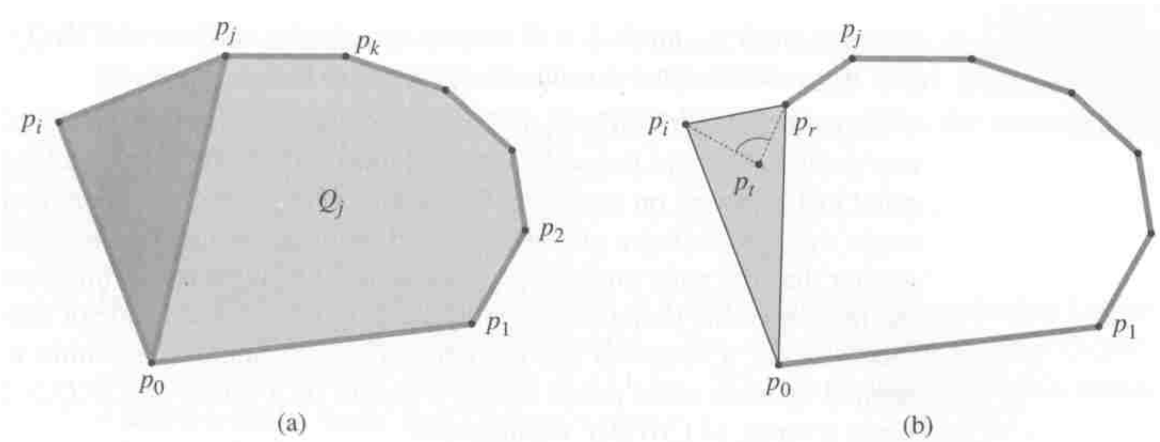

Figure 8: Correctness of the Graham scan

Indeed, if $p_t \notin S$ then $p_t \in \Delta(p_0, p_i, p_r)$, where $p_t = \text{TOP}(S)$ and $p_r = \text{NEXT-TO-TOP}(S)$ at time i. Hence, $p_t \notin CH(Q)$.

$$\Rightarrow \qquad CH(Q) \subseteq S$$

To prove the converse, we show by induction on i that on the i-th iteration of the **for** -loop S consists of the points of $CH(Q_i)$ only.

$$\Rightarrow \qquad S \subseteq CH(Q)$$

<u>The running time:</u> Sorting takes $O(n \log n)$ time.
Both POP and PUSH take $O(1)$ time.
The entire **while** -loop is executed $O(n)$ times, since each point is put on stack exactly once (and popped out of stack at most once).

$\Rightarrow$ Total running time is $O(n \log n)$.

6. Finding a closest pair of points

Instance: A set of points $P = \{p_1, \ldots, p_n\}$.
Problem: Find a pair of points with minimum distance.

Trivial solution: check all $\binom{n}{2}$ pairs of points. This leads to a $\Theta(n^2)$ algorithm.

We apply the Divide and Conquer method to solve the problem in time $O(n \log n)$.

The algorithm consists of the following steps:

Divide: Split the set P by a vertical line ℓ into two parts P_L and P_R ($|P_L| = \lfloor n/2 \rfloor$, $|P_R| = \lceil n/2 \rceil$). If some points are on ℓ assign them arbitrarily to P_L or P_R.

Conquer: Find closest neighbors in P_L and P_R (let δ_L and δ_R be the shortest distances and $\delta = \min\{\delta_L, \delta_R\}$). Sort the points in P_L and P_R w.r.t. x- and y-coordinates and create the sorted arrays X_L, X_R, Y_L and Y_R.

Combine: The nearest neighbors x, y of P are either on distance δ (i.e. $x, y \in P_L$ or $x, y \in P_R$) or $x \in P_L$, $y \in P_R$. We can consider only those points that are on distance at most δ from ℓ (the Y'-zone).

To find a closest pair (x, y) with $x \in P_L$, $y \in P_R$ we do the following:

1. Sort the points of Y' w.r.t. the y-coordinate.
2. For each $p \in Y'$ construct all points of Y' within the distance δ from p and find the shortest distance δ_p between them.
3. Return $\min\{\delta, \min_p \delta_p\}$.

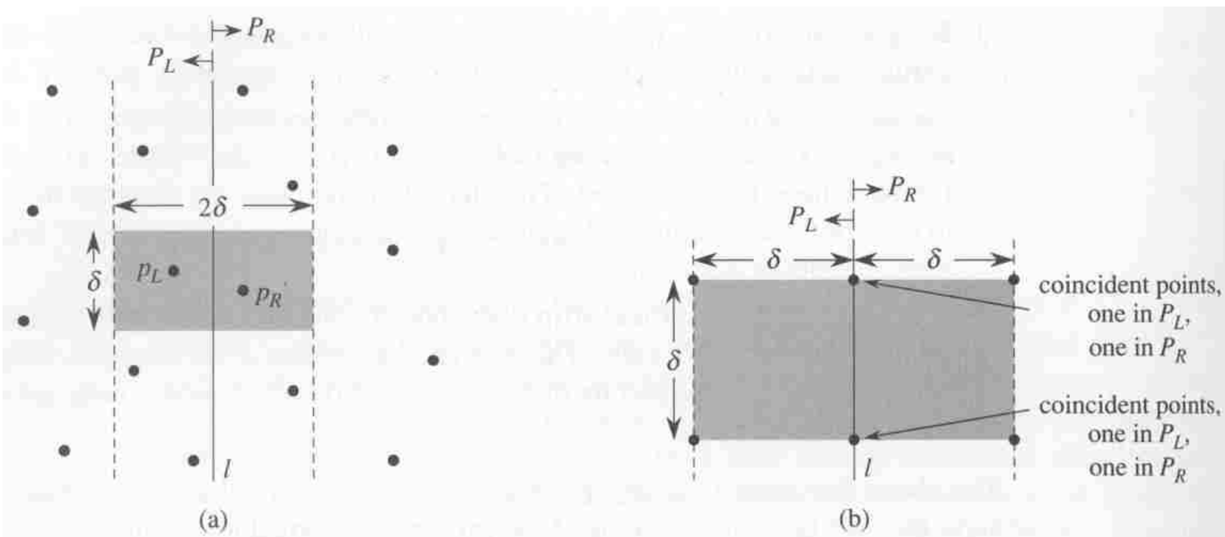

Figure 9: Finding a closest pair of points

If the points of the Y'-zone are sorted according to the y-coordinate, one needs to consider for every point $p \in Y'$ only up to 5 points of Y'.

Implementation details

We can efficiently construct the sorted arrays Y_L, Y_R (needed for the recursive calls) from Y (sorted array of all points y-coordinates) as follows:

Algorithm 5

1. Initialize empty arrays Y_L and Y_R.
2. **for** $i = 1$ to $|Y|$
3. **if** $(Y[i] \in P_L)$
4. add $Y[i]$ to Y_L
5. **else**
6. add $Y[i]$ to Y_R

This procedure runs in linear time. Similarly sort arrays X_L and X_R can be constructed from X (needed for finding the line ℓ).

If the array Y of all y-coordinates is sorted (an $O(n \log n)$ preprocessing), the sorted array Y' can be constructed in $O(n)$ time instead of $O(n \log n)$ (**important!**).

Therefore, for the running time $T(n)$ one has:

$$T(n) = 2 \cdot T(n/2) + O(n),$$

which implies $T(n) = O(n \log n)$.

Hence, the total running time ($T(n) + O(n \log n)$ preprocessing) is $O(n \log n)$.

7. Diameter of a point set

Consider the two following problems:

DIAMETER
Given n points in the plane, find a pair with maximum distance.

DISJOINTNESS
Given two sets A, B of positive numbers, determine if $A \cap B \neq \emptyset$.

Theorem 2 (Ben-Or, 1983)
To solve the DISJOINTNESS *problem,* $\Omega(n \log n)$ *comparisons are necessary.*

We transform DISJOINTNESS into DIAMETER:

Let A, B be an instance for DISJOINTNESS.
Denote by C the disk of radius 1 centered in $(0, 0)$.
For $a_i \in A$ let a'_i be the intersection of C and the line $y = a_i x$ for $x > 0$.
For $b_j \in B$ let b'_j be the intersection of C and the line $y = b_j x$ for $x < 0$.
Consider the set $\{a'_i\} \cup \{b'_j\}$ as an instance for DIAMETER.
One has: Diam$(\{a'_i\} \cup \{b'_j\}) = 2 \Leftrightarrow A \cap B \neq \emptyset$.

This implies:

Theorem 3 *To find the diameter of a set of* n *points,* $\Omega(n \log n)$ *operations are necessary.*

Theorem 4 (Hocking-Young, 1961)
The diameter of a set of points equals the diameter of its convex hull.

Theorem 5 (Yaglom-Boltyanskii, 1961)
The diameter of a convex polygon is the maximum distance between its parallel tangent lines.

We call two points of a convex polygon antipodal, if there exist two parallel tangent lines passing through these points.

Our goal is, therefore, to find all pairs of antipodal points.

Let P be a simple polygon and let its points $p_0, \ldots, p_{n-1}$ be numbered in the counterclockwise order. Starting in some point $p_i \in P$, we visit the points of P in the cyclic order to find the first point $q_R \in P$ with a maximum distance from the line (p_{i-1}, p_i) (the point indices are considered modulo n).

After that, keeping going in the same direction, we find a point q_L with maximum distance from the line (p_i, p_{i+1}).

The set of points between q_R and q_L (inclusive) determine all points that are antipodal to p_i (see the next page).

In the following pseudocode, the method $\textsc{Next}(p_i)$ returns the point next to p_i in the cyclic order (i.e. the point $p_{i+1 \bmod n}$).

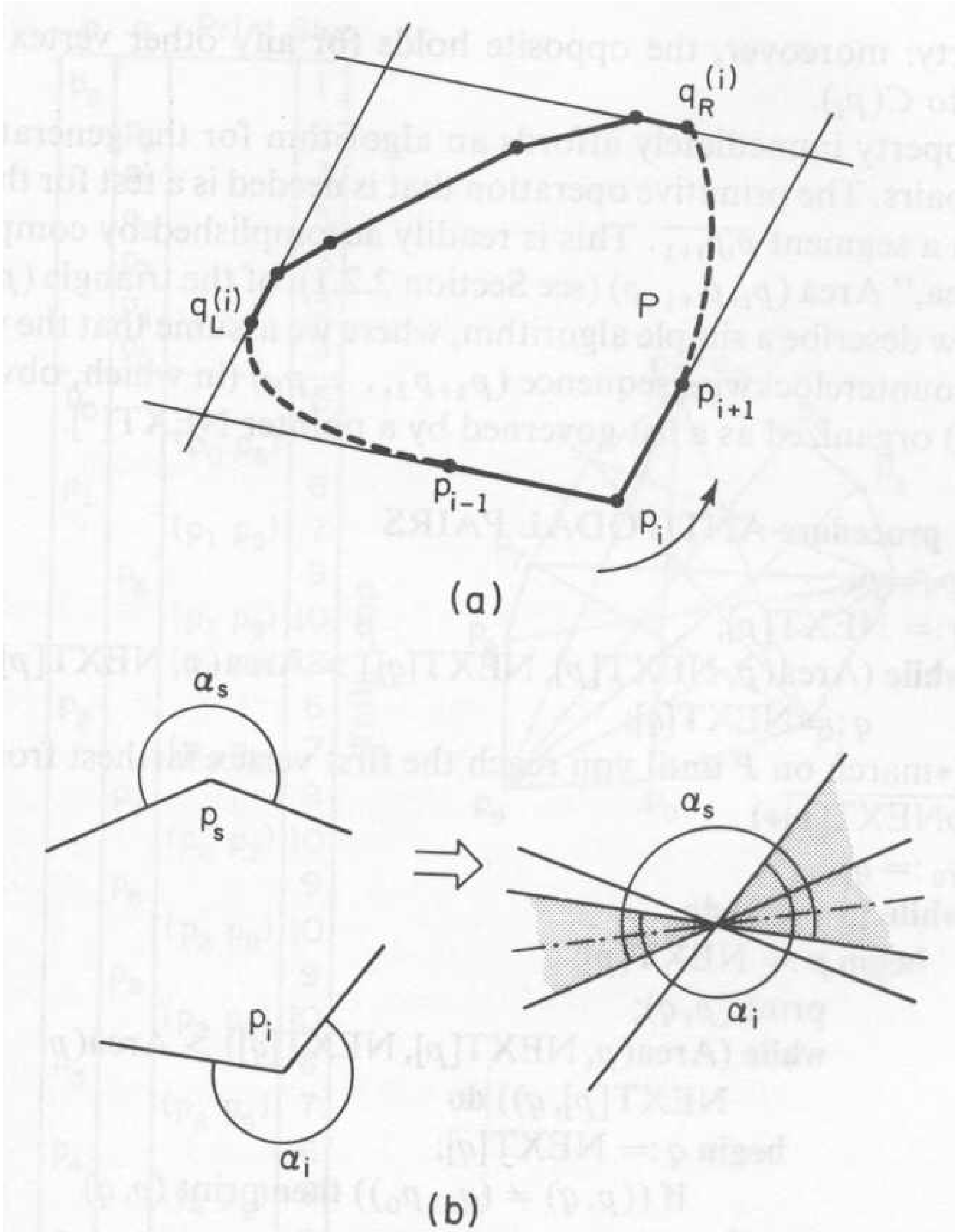

Figure 10: Seach for all points antipodal to p_i (points between q_R and q_L)

p	q	Print	Step
p_9			1
	p_0		2
	p_1		3
	p_2		3
	p_3		3
	p_4		3
	p_5		3
p_0			6
		$(p_0\ p_5)$	7
p_1			6
		$(p_1\ p_5)$	7
	p_6		9
		$(p_1\ p_6)$	10
		$(p_1\ p_7)$	12
p_2			6
		$(p_2\ p_6)$	7
	p_7		9
		$(p_2\ p_7)$	10
	p_8		9
		$(p_2\ p_8)$	10
	p_9		9
		$(p_2\ p_9)$	10
p_3			6
		$(p_3\ p_9)$	7
p_4			6
		$(p_4\ p_9)$	7
p_5			6
		$(p_5\ p_9)$	7
	p_0		9

While Loop

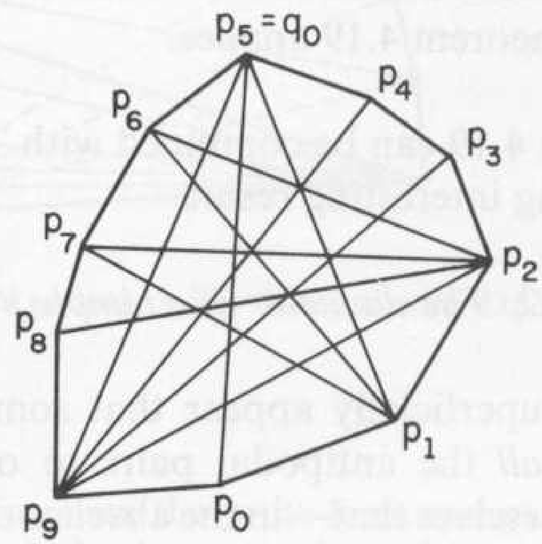

Figure 11: Computing the diameter of a polygon (here $\overline{p_1 p_2} \parallel \overline{p_6 p_7}$)

Algorithm 6 $\textsc{AntiPairs}$;

1. $i := n - 1$
2. $q := p_0$
3. **while** $\mathsf{Dist}(p_i, p_{i+1}; \textsc{Next}(q)) > \mathsf{Dist}(p_i, p_{i+1}; q)$
   $q := \textsc{Next}(q)$
4. $q_0 := q$

5. **while** $q \neq p_0$ **do**
6. $\quad i := i + 1$ (mod n)
7. $\quad$ Output(p_i, q)
8. $\quad$ **while** $\mathsf{Dist}(p_i, p_{i+1}; \textsc{Next}(q)) \geq \mathsf{Dist}(p_i, p_{i+1}; q)$
9. $\qquad q := \textsc{Next}(q)$
10. $\qquad$ **if** $(p_i, q) \neq (q_0, p_0)$ Output(p_i, q)
11. $\qquad$ **else** break

Since the number of pairs of parallel segments of the polygon it at most $\lfloor n/2 \rfloor$, the number of antipodal pairs is at most $3n/2$. So:

Theorem 6 *The diameter of a convex polygon can be found in $O(n)$ time.*

Corollary 2 *The diameter of a set of n points can be found in $O(n \log n)$ time.*

Corollary 3 *The diameter of a simple polygon can be found in a linear time.*