Computational Geometry

Computational Geometry – Introduction

- Branch of computer science that studies algorithms for solving geometric problems
- Modern engineering and mathematics, computational geometry has applications in such diverse fields as computer graphics, robotics, VLSI design, computer-aided design, molecular modeling, metallurgy, manufacturing, textile layout, forestry, and statistics

Computational Geometry – Introduction

- Input – Description of a set of geometric objects, such as a set of points, a set of line segments, or the vertices of a polygon in counterclock–wise order
- Output – a response to a query about the objects, such as whether any of the lines intersect, or perhaps a new geometric object, such as the convex hull (smallest enclosing convex polygon) of the set of points.

Computational Geometry – Introduction

- We look at a few computational–geometry algorithms in two dimensions, that is, plane
- We represent each input object by a set of points $\{p_1, p_2, p_3, \ldots\}$, where each $p_i = (x_i, y_i)$ and $x_i, y_i \in R$
- For example, we represent an n–vertex polygon P by a sequence $\langle p_0, p_1, p_2, \ldots, p_{n-1} \rangle$ of its vertices in order of their appearance on the boundary of P

Computational Geometry – Introduction

- Computational geometry can also apply to three dimensions, and even higher-dimensional spaces, but such problems and their solutions can be very difficult to visualize

Line–segment properties

- A convex combination of two distinct points $p_1 = (x_1 , y_1)$ and $p_2 = (x_2, y_2)$ is any point $p_3 = (x_3 , y_3)$ such that for some α in the range $0 \leq \alpha \leq 1$, we have $x_3 = \alpha x_1 + (1 - \alpha) x_2$ and $y_3 = \alpha y_1 + (1 - \alpha) y_2$
- We also write that $p_3 = \alpha p_1 + (1 - \alpha) p_2$
- Intuitively, p_3 is any point that is on the line passing through p_1 and p_2 and is on or between p_1 and p_2 on the line

Line–segment properties

- Given two distinct points p_1 and p_2 , the line segment $\overline{p_1 p_2}$ is th e set of convex combinations of p_1 and p_2
- We call p_1 and p_2 the endpoints of segment $\overline{p_1 p_2}$
- Sometimes the ordering of p_1 and p_2 matters, and we speak of the directed segment $\overrightarrow{p_1 p_2}$.
- If p_1 is the origin (0, 0), then we can treat the directed segment $\overrightarrow{p_1 p_2}$ as the vector p 2

Line–segment Questions

- Given two directed segments, $\overrightarrow{p_0 p_1}$ and $\overrightarrow{p_0 p_2}$, is $\overrightarrow{p_0 p_1}$ clockwise from $\overrightarrow{p_0 p_2}$ with respect to their common endpoint p_0 ?
- Given two line segments $\overline{p_0 p_1}$ and $\overline{p_1 p_2}$, if we traverse $\overline{p_0 p_1}$ and then $\overline{p_1 p_2}$, do we make a left turn at point p_1 ?
- Do line segments $\overline{p_1 p_2}$ and $\overline{p_3 p_4}$ intersect?

Line–segment Questions

- Can answer each question in O(1) time
- No surprise since the input size of each question is O(1)
- Our methods use only additions, subtractions, multiplications, and comparisons
- Need neither division nor trigonometric functions, both of which can be computationally expensive and prone to problems with round–off error

Line–segment Questions

- For example, the “straightforward” method of determining whether two segments intersect—compute the line equation of the form $y = mx + b$ for each segment (m is the slope and b is the y–intercept),
- Find point of intersection of lines, and check whether this point is on both segments—uses division to find the point of intersection

Line–segment Questions

- When segments are nearly parallel, this method is very sensitive to precision of division operation on real computers
- The method in this section, which avoids division, is much more accurate

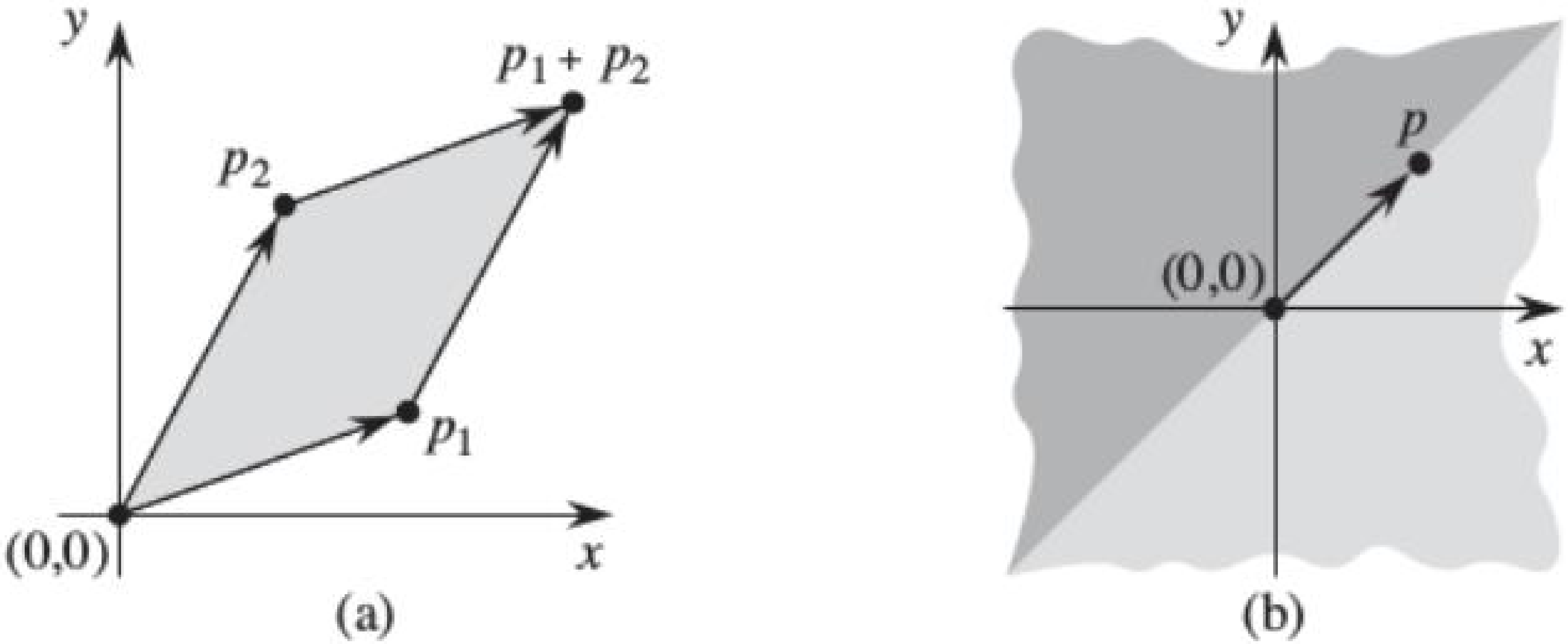
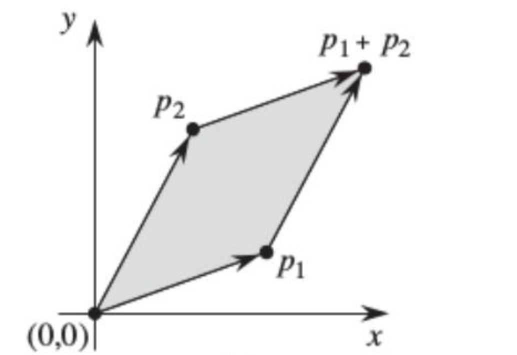


Figure 33.1 **(a)** The cross product of vectors p_1 and p_2 is the signed area of the parallelogram. **(b)** The lightly shaded region contains vectors that are clockwise from p. The darkly shaded region contains vectors that are counterclockwise from p.

Cross products

- Computing cross products lies at the heart of our line–segment methods
- Consider vectors p_1 and p_2, shown in Figure



(a)

- We can interpret the cross product p_1 x p_2 as the signed area of the parallelogram formed by the points (0, 0), p_1 , p_2, and $p_1 + p_2 = (x_1 + x_2 , y_1 + y_2)$

Cross products

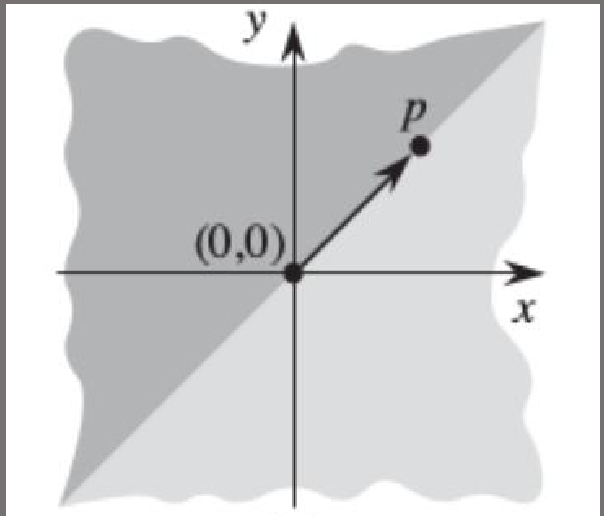
- An equivalent, but more useful, definition gives the cross product as the determinant of a matrix:

$$\begin{aligned} p_1 \times p_2 &= \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} \\ &= x_1 y_2 - x_2 y_1 \\ &= -p_2 \times p_1 . \end{aligned}$$

- If p_1 X p_2 is positive, then p_1 is clockwise from p_2 with respect to the origin (0, 0), if this cross product is negative, then p_1 is counterclockwise from p_2

Cross products

- Figure 33.1(b) shows the clockwise and counterclockwise regions relative to a vector p
- A boundary condition arises if the cross product is 0; in this case, the vectors are colinear, pointing in either the same or opposite directions



(b)

Cross products

- To determine whether a directed segment $\overrightarrow{p_0 p_1}$ is closer to a directed segment $\overrightarrow{p_0 p_2}$ in a clockwise direction or in a counterclockwise direction with respect to their common endpoint p_0, we simply translate to use p_0 as the origin.

Cross products

- That is, we let $p_1 - p_0$ denote the vector $p'_1 = (x'_1, y'_1)$, where $x'_1 = x_1 - x_0$ and $y'_1 = y_1 - y_0$, and we define $p_2 - p_0$ similarly
- We then compute the cross product
- $(p_1 - p_0) \times (p_2 - p_0) = (x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0)$
- If this cross product is positive, then $\overrightarrow{p_0 p_1}$ is clockwise from $\overrightarrow{p_0 p_2}$; if negative, it is counterclockwise

Determining whether consecutive segments turn left or right

- Whether two consecutive line segments p_0p_1 and p_1p_2 turn left or right at point p_1
- Equivalently, we want a method to determine which way a given angle $\angle p_0 p_1 p_2$ turns
- Cross products allow us to answer this question without computing the angle.

Determining whether consecutive segments turn left or right

- As Figure 33.2 shows, we simply check whether directed segment $\overrightarrow{p_0 p_2}$ is clockwise or counterclockwise relative to directed segment $\overrightarrow{p_0 p_1}$
- To do so, we compute the cross product $(p_2 - p_0) \times (p_1 - p_0)$
- If the sign of this cross product is negative, then $\overrightarrow{p_0 p_2}$ is counterclockwise with respect to $\overrightarrow{p_0 p_1}$, and thus we make a left turn at p_1

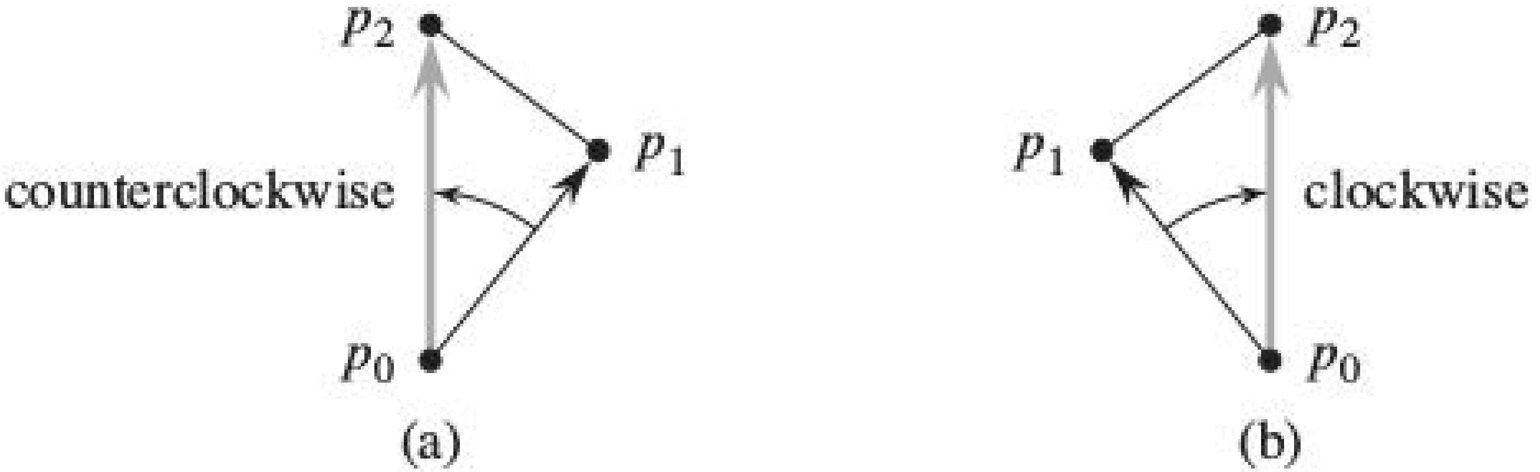


Figure 33.2 Using the cross product to determine how consecutive line segments $\overline{p_0 p_1}$ and $\overline{p_1 p_2}$ turn at point p_1. We check whether the directed segment $\overrightarrow{p_0 p_2}$ is clockwise or counterclockwise relative to the directed segment $\overrightarrow{p_0 p_1}$. **(a)** If counterclockwise, the points make a left turn. **(b)** If clockwise, they make a right turn.

Determining whether consecutive segments turn left or right

- A positive cross product indicates a clockwise orientation and a right turn
- A cross product of 0 means that points p_0, p_1, and p_2 are colinear

Determining whether two line segments intersect

- To determine whether two line segments intersect, we check whether each segment straddles the line containing the other
- A segment $\overline{p_1 p_2}$ straddles a line if point p_1 lies on one side of the line and point p_2 lies on the other side
- Two line segments intersect if and only if either (or both) of the following conditions holds:

Determining whether two line segments intersect

1. Each segment straddles the line containing the other.
2. An endpoint of one segment lies on the other segment. (This condition comes from the boundary case.)

Determining whether two line segments intersect

- The following procedures implement this idea. SEGMENTS–INTERSECT returns TRUE if segments $\overline{p_1 p_2}$ and $\overline{p_3 p_4}$ intersect and FALSE if they do not
- It calls the subroutines DIRECTION, which computes relative orientations using the cross–product method above, and ON–SEGMENT , which determines whether a point known to be colinear with a segment lies on that segment.

```
SEGMENTS-INTERSECT(p1, p2, p3, p4)
d1 = DIRECTION(p3, p4, p1)
d2 = DIRECTION(p3, p4, p2)
d3 = DIRECTION(p1, p2, p3)
d4 = DIRECTION(p1, p2, p4)
if ((d1 > 0 and d2 < 0) or (d1 < 0 and d2 > 0)) and
        ((d3 > 0 and d4 < 0) or (d3 < 0 and d4 > 0))
    return TRUE
elseif d1 == 0 and ON-SEGMENT(p3, p4, p1)
    return TRUE
elseif d2 == 0 and ON-SEGMENT(p3, p4, p2)
    return TRUE
elseif d3 == 0 and ON-SEGMENT(p1, p2, p3)
    return TRUE
elseif d4 == 0 and ON-SEGMENT(p1, p2, p4)
    return TRUE
else return FALSE
```

```
DIRECTION(p_i, p_j, p_k)
    return (p_k − p_i) × (p_j − p_i)
```

```
ON-SEGMENT(p_i, p_j, p_k)
    if min(x_i, x_j) ≤ x_k ≤ max(x_i, x_j) and min(y_i, y_j) ≤ y_k ≤ max(y_i, y_j)
            return TRUE
    else return FALSE
```

Determining whether two line segments intersect

- S EGMENTS –I NTERSECT works as follows. Lines 1 – 4 compute the relative orientation di of each endpoint pi with respect to the other segment.
- If all the relative orientations are nonzero, then we can easily determine whether segments $\overline{p_1 p_2}$ and $\overline{p_3 p_4}$ intersect, as follows.

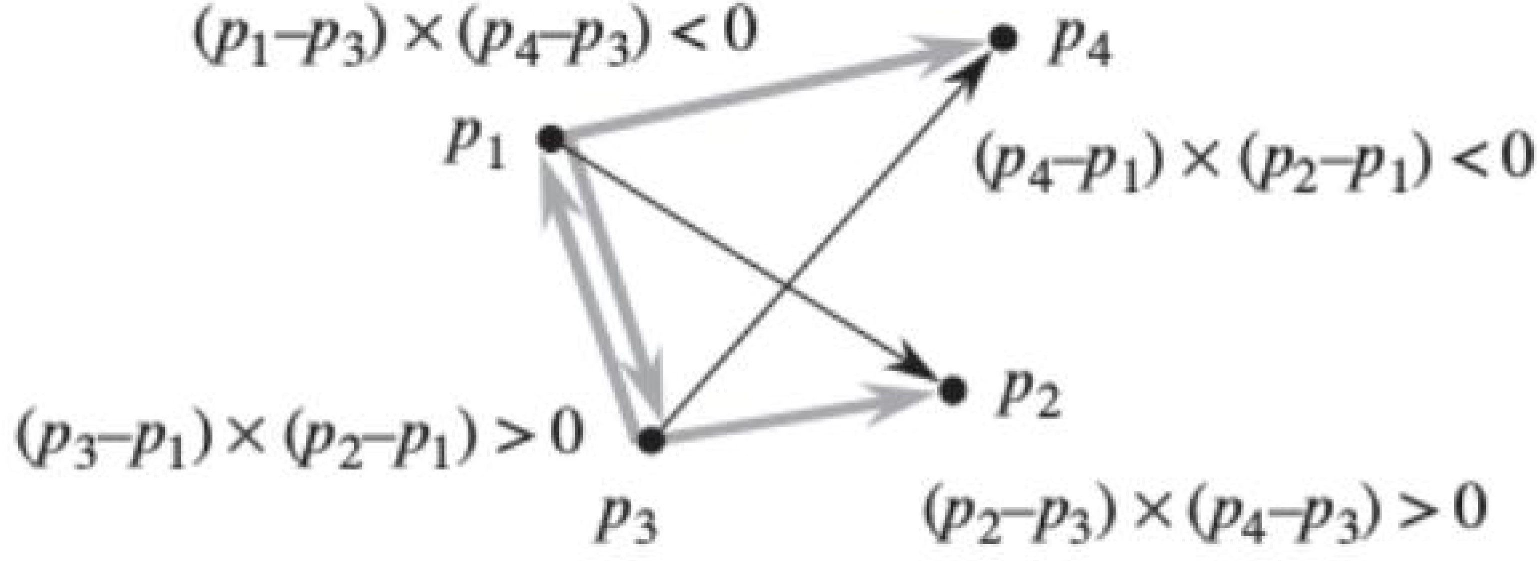
Determining whether two line segments intersect

- Segment $\overline{p_1 p_2}$ straddles the line containing segment $\overline{p_3 p_4}$ if if directed segments $\overrightarrow{p_3 p_1}$ and $\overrightarrow{p_3 p_2}$ have opposite orientations relative to $\overrightarrow{p_3 p_4}$
- In this case, the signs of d_1 and d_2 differ
- Similarly, $\overline{p_3 p_4}$ straddles the line containing $\overline{p_1 p_2}$ if the signs of d_3 and d_4 differ

Determining whether two line segments intersect

- If the test of line 5 is true, then the segments straddle each other, and S EGMENTS –I NTERSECT returns TRUE .
- Figure 33.3(a) shows this case

ntersect



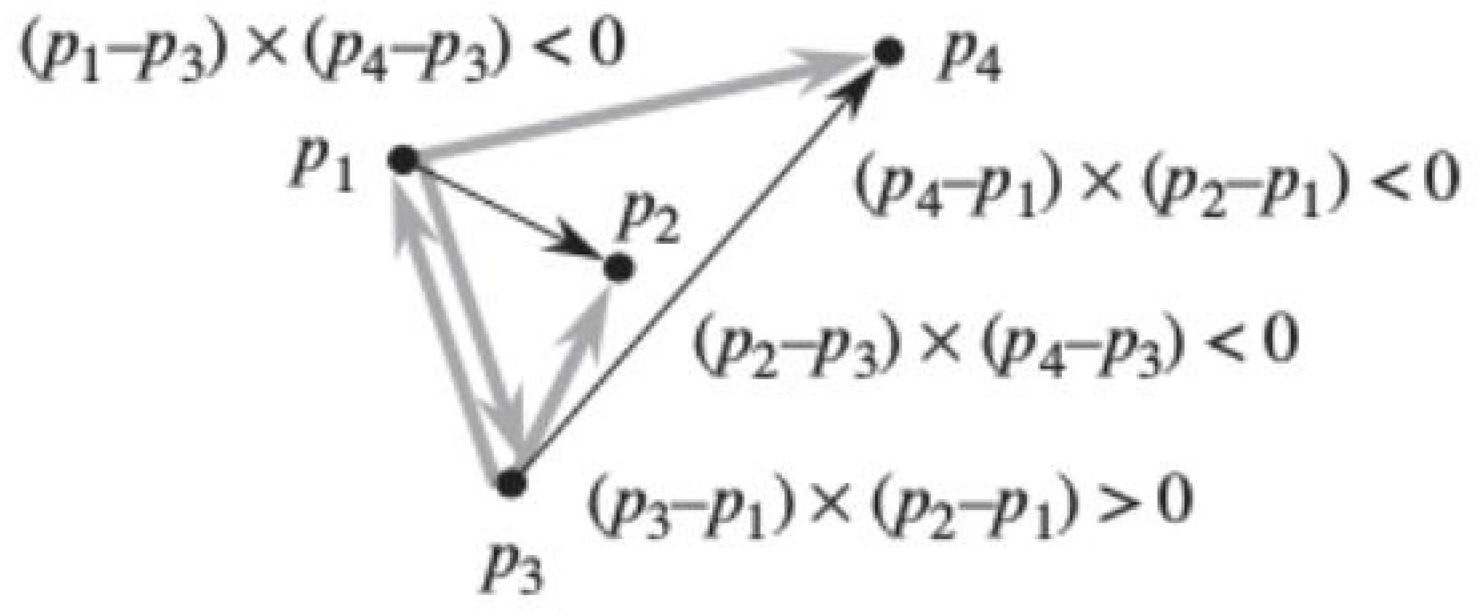
(a)

- The segments p_1p_2 and p_3 p_4 straddle each other's lines. Because p_3p_4 straddles the line containing p_1p_2, the signs of the cross products $(p_3 - p_1)$ x $(p_2 - p_1)$ and $(p_4 - p_1)$ x $(p_2 - p_1\ )$ differ. Because p_1p_2 straddles the line containing p_3 p_4 , the signs of the cross products $(p_1 - p_3\ )$ x $(p_4 - p_3\ )$ and $(p_2 - p_3)$ x $(p_4 - p_3)$ differ.

Determining whether two line segments intersect

- Otherwise, the segments do not straddle each other's lines, although a boundary case may apply
- If all the relative orientations are nonzero, no boundary case applies.
- All the tests against 0 in lines 7 – 13 then fail, and SEGMENTS –I NTERSECT returns FALSE in line 15

ts intersect



(b)

- Segment p_3p_4 straddles the line containing p_1p_2 , but p_1 p_2 does not straddle the line containing p_3p_4 . The signs of the cross products $(p_1 - p_3)$ x $(p_4 - p_3)$ and $(p_2 - p_3)$ x $(p_4 - p_3)$ are the same.

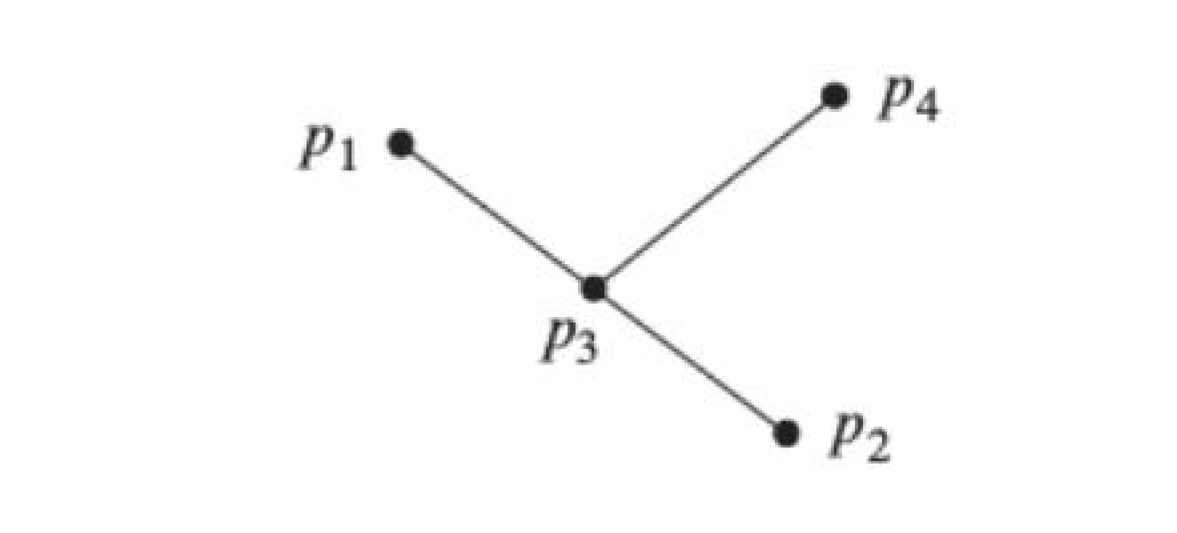
Determining whether two line segments intersect

- A boundary case occurs if any relative orientation d_k is 0.
- Here, we know that pk is colinear with the other segment.
- It is directly on the other segment if and only if it is between the endpoints of the other segment
- The procedure ON –S EGMENT returns whether p_k is between the endpoints of segment $\overline{p_i p_j}$

Determining whether two line segments intersect

- which will be the other segment when called in lines 7 – 13; the procedure assumes that p_k is colinear with segment $\overline{p_i p_j}$
- Figures 33.3(c) and (d) show cases with colinear points
- In Figure 33.3(c), p 3 is on $\overline{p_1 p_2}$,
- and so S EGMENTS –I NTERSECT returns TRUE in line 12.
- No endpoints are on other segments in Figure 33.3(d), and so SEGMENTS –INTERSECT returns FALSE in line 15.

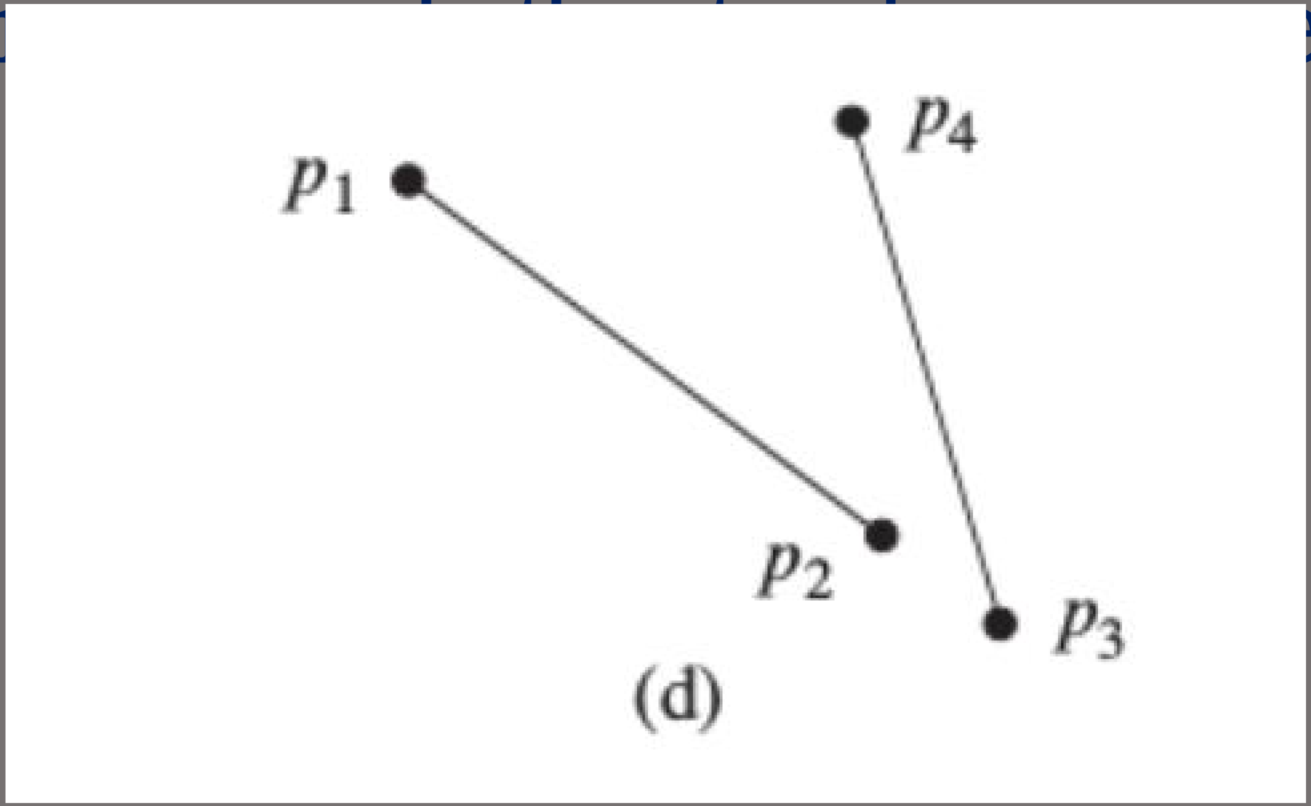
Determining whether two line segments intersect



(c)

- Point p 3 is colinear with p 1 p 2 and is between p 1 and p 2

Dete… …ents intersect



- Point p3 is colinear with p1p2 , but it is not between 1 and p 2 . The segments do not intersect

Other applications of cross products

- Later sections of this chapter introduce additional uses for cross products.
- In Section 33.3, we shall need to sort a set of points according to their polar angles with respect to a given origin.
- As Exercise 33.1–3 asks you to show, we can use cross products to perform the comparisons in the sorting procedure.

Other applications of cross products

- In Section 33.2, we shall use red–black trees to maintain the vertical ordering of a set of line segments.
- Rather than keeping explicit key values which we compare to each other in the red–black tree code, we shall compute a cross–product to determine which of two segments that intersect a given vertical line is above the other.