33.1 Line-segment properties

Several of the computational-geometry algorithms in this chapter will require answers to questions about the properties of line segments. 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 $0 \le \alpha \le 1$, 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. Given two distinct points p_1 and p_2, the ***line segment*** $\overline{p_1 p_2}$ is the 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.

In this section, we shall explore the following questions:

1. 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?
2. 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?
3. Do line segments $\overline{p_1 p_2}$ and $\overline{p_3 p_4}$ intersect?

There are no restrictions on the given points.

We can answer each question in $O(1)$ time, which should come as no surprise since the input size of each question is $O(1)$. Moreover, our methods will use only additions, subtractions, multiplications, and comparisons. We need neither division nor trigonometric functions, both of which can be computationally expensive and prone to problems with round-off error. 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 the point of intersection of the lines, and check whether this point is on both segments−uses division to find the point of intersection. When the segments are nearly parallel, this method is very sensitive to the precision of the division operation on real computers. The method in this section, which avoids division, is much more accurate.

Cross products

Computing cross products is at the heart of our line-segment methods. Consider vectors p_1 and p_2, shown in Figure 33.1(a). The ***cross product*** $p_1 \times p_2$ can be interpreted as the signed area of the parallelogram formed by the points (0, 0), p_1, p_2, and $p_1 + p_2 = (x_1 + x_2, y_1 + y_2)$. An equivalent, but more useful, definition gives the cross product as the determinant of a matrix:[1]

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

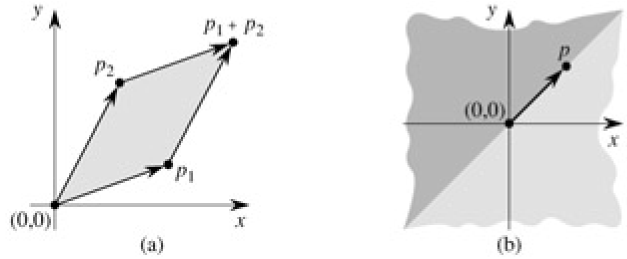

Figure 33.1: *(a)* The cross product of vectors p_1 and p_2 is the signed area of the parallelogram. *(b)* The lightly shaded region contains vectors that are clockwise from p. The darkly shaded region contains vectors that are counterclockwise from p.

If $p_1 \times p_2$ 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. (See Exercise 33.1-1.) 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 ***collinear***, pointing in either the same or opposite directions.

To determine whether a directed segment $\overrightarrow{p_0p_1}$ is clockwise from a directed segment $\overrightarrow{p_0p_2}$ with respect to their common endpoint p_0, we simply translate to use p_0 as the origin. 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 $\overrightarrow{p_0p_1}$ is clockwise from $\overrightarrow{p_0p_2}$; if negative, it is counterclockwise.

Determining whether consecutive segments turn left or right

Our next question is whether two consecutive line segments $\overline{p_0p_1}$ and $\overline{p_1p_2}$ turn left or right at point p_1. Equivalently, we want a method to determine which way a given angle $\angle p_0p_1p_2$ turns. Cross products allow us to answer this question without computing the angle. As shown in Figure 33.2, we simply check whether directed segment $\overrightarrow{p_0p_2}$ is clockwise or counterclockwise relative to directed segment $\overrightarrow{p_0p_1}$. To do this, we compute the cross product $(p_2 - p_0) \times (p_1 - p_0)$. If the sign of this cross product is negative, then $\overrightarrow{p_0p_2}$ is counterclockwise with respect to $\overrightarrow{p_0p_1}$, and thus we make a left turn at p_1. 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 collinear.

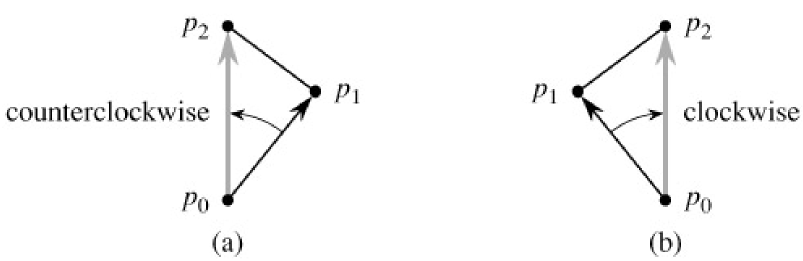

Figure 33.2: Using the cross product to determine how consecutive line segments $\overline{p_0p_1}$ and $\overline{p_1p_2}$ turn at point p_1. We check whether the directed segment $\overrightarrow{p_0p_2}$ is clockwise or counterclockwise relative to the directed segment $\overrightarrow{p_0p_1}$. *(a)* If counterclockwise, the points make a left turn. *(b)* If clockwise, they make a right turn.

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 $\overline{p_1p_2}$ ***straddles*** a line if point p_1 lies on one side of the line and point p_2 lies on the other side. A boundary case arises if p_1 or p_2 lies directly on the line. Two line segments intersect if and only if either (or both) of the following conditions holds:

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.)

The following procedures implement this idea. SEGMENTS-INTERSECT returns TRUE if segments $\overline{p_1p_2}$ and $\overline{p_3p_4}$ 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 collinear with a segment lies on that segment.

```
SEGMENTS-INTERSECT(p1, p2, p3, p4)
 1  d1 ← DIRECTION(p3, p4, p1)
 2  d2 ← DIRECTION(p3, p4, p2)
 3  d3 ← DIRECTION(p1, p2, p3)
 4  d4 ← DIRECTION(p1, p2, p4)
 5  if ((d1 > 0 and d2 < 0) or (d1 < 0 and d2 > 0)) and
          ((d3 > 0 and d4 < 0) or (d3 < 0 and d4 > 0))
 6    then return TRUE
 7  elseif d1 = 0 and ON-SEGMENT(p3, p4, p1)
 8    then return TRUE
 9  elseif d2 = 0 and ON-SEGMENT(p3, p4, p2)
10    then return TRUE
11  elseif d3 = 0 and ON-SEGMENT(p1, p2, p3)
12    then return TRUE
13  elseif d4 = 0 and ON-SEGMENT(p1, p2, p4)
14    then return TRUE
15  else return FALSE
```

```
DIRECTION(pi, pj, pk)
1  return (pk - pi) × (pj - pi)
```

```
ON-SEGMENT(pi, pj, pk)
1  if min(xi, xj) ≤ xk ≤ max(xi, xj) and min(yi, yj) ≤ yk ≤ max(yi, yj)
2    then return TRUE
3    else return FALSE
```

SEGMENTS-INTERSECT works as follows. Lines 1−4 compute the relative orientation d_i of each endpoint p_i with respect to the other segment. If all the relative orientations are nonzero, then we can easily determine whether segments $\overline{p_1p_2}$ and $\overline{p_3p_4}$ intersect, as follows. Segment $\overline{p_1p_2}$ straddles the line containing segment $\overline{p_3p_4}$ if directed segments $\overrightarrow{p_3p_1}$ and $\overrightarrow{p_3p_2}$ have opposite orientations relative to $\overrightarrow{p_3p_4}$. In this case, the signs of d_1 and d_2 differ. Similarly, $\overline{p_3p_4}$ straddles the line containing $\overline{p_1p_2}$ if the signs of d_3 and d_4 differ. If the test of line 5 is true, then the segments straddle each other, and SEGMENTS-INTERSECT returns TRUE. Figure 33.3 (a) shows this case. 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-INTERSECT returns FALSE in line 15. Figure 33.3(b) shows this case.

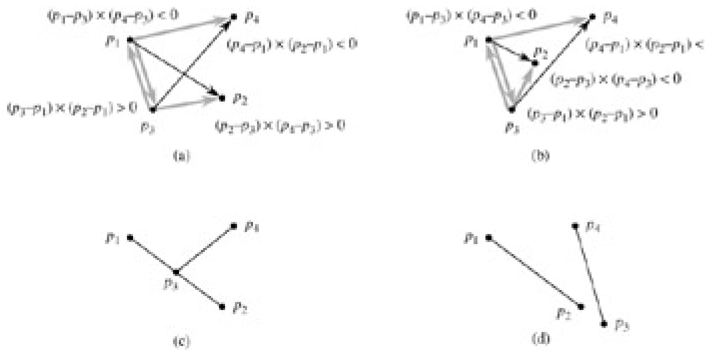

Figure 33.3: Cases in the procedure SEGMENTS-INTERSECT. *(a)* The segments $\overline{p_1p_2}$ and $\overline{p_3p_4}$ straddle each other's lines. Because $\overline{p_3p_4}$ straddles the line containing $\overline{p_1p_2}$, the signs of the cross products $(p_3 - p_1) \times (p_2 - p_1)$ and $(p_4 - p_1) \times (p_2 - p_1)$ differ. Because $\overline{p_1p_2}$ straddles the line containing $\overline{p_3p_4}$, the signs of the cross products $(p_1 - p_3) \times (p_4 - p_3)$ and $(p_2 - p_3) \times (p_4 - p_3)$ differ. *(b)* Segment $\overline{p_3p_4}$ straddles the line containing $\overline{p_1p_2}$, but $\overline{p_1p_2}$ does not straddle the line containing $\overline{p_3p_4}$. The signs of the cross products $(p_1 - p_3) \times (p_4 - p_3)$ and $(p_2 - p_3) \times (p_4 - p_3)$ are the same. *(c)* Point p_3 is collinear with $\overline{p_1p_2}$ and is between p_1 and p_2. *(d)* Point p_3 is collinear with $\overline{p_1p_2}$, but it is not between p_1 and p_2. The segments do not intersect.

A boundary case occurs if any relative orientation d_k is 0. Here, we know that p_k is collinear 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-SEGMENT returns whether p_k is between the endpoints of segment $\overline{p_ip_j}$, which will be the other segment when called in lines 7–13; the procedure assumes that p_k is collinear with segment $\overline{p_ip_j}$. Figures 33.3(c) and (d) show cases with collinear points. In Figure 33.3(c), p_3 is on $\overline{p_1p_2}$, and so SEGMENTS-INTERSECT 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.

Other applications of cross products

Later sections of this chapter will 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, cross products can be used to perform the comparisons in the sorting procedure. 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, we shall replace each key comparison in the red-black tree code by a cross-product calculation to determine which of two segments that intersect a given vertical line is above the other.

Exercises 33.1-1

Prove that if $p_1 \times p_2$ is positive, then vector p_1 is clockwise from vector p_2 with respect to the origin (0, 0) and that if this cross product is negative, then p_1 is counterclockwise from p_2.

Exercises 33.1-2

Professor Powell proposes that only the *x*-dimension needs to be tested in line 1 of ON-SEGMENT. Show why the professor is wrong.

Exercises 33.1-3

The ***polar angle*** of a point p_1 with respect to an origin point p_0 is the angle of the vector $p_1 - p_0$ in the usual polar coordinate system. For example, the polar angle of (3, 5) with respect to (2, 4) is the angle of the vector (1, 1), which is 45 degrees or $\pi/4$ radians. The polar angle of (3, 3) with respect to (2, 4) is the angle of the vector (1, -1), which is 315 degrees or $7\pi/4$ radians. Write pseudocode to sort a sequence $\langle p_1, p_2, ..., p_n \rangle$ of n points according to their polar angles with respect to a given origin point p_0. Your procedure should take $O(n \lg n)$ time and use cross products to compare angles.

Exercises 33.1-4

Show how to determine in $O(n^2 \lg n)$ time whether any three points in a set of n points are collinear.

Exercises 33.1-5

A ***polygon*** is a piecewise-linear, closed curve in the plane. That is, it is a curve ending on itself that is formed by a sequence of straight-line segments, called the ***sides*** of the polygon. A point joining two consecutive sides is called a ***vertex*** of the polygon. If the polygon is ***simple***, as we shall generally assume, it does not cross itself. The set of points in the plane enclosed by a simple polygon forms the ***interior*** of the polygon, the set of points on the polygon itself forms its ***boundary***, and the set of points surrounding the polygon forms its ***exterior***. A simple polygon is ***convex*** if, given any two points on its boundary or in its interior, all points on the line segment drawn between them are contained in the polygon's boundary or interior.

Professor Amundsen proposes the following method to determine whether a sequence $\langle p_0, p_1, ..., p_{n-1} \rangle$ of n points forms the consecutive vertices of a convex polygon. Output "yes" if the set $\{\angle p_i p_{i+1} p_{i+2} : i = 0, 1, ..., n - 1\}$, where subscript addition is performed modulo n, does not contain both left turns and right turns; otherwise, output "no." Show that although this method runs in linear time, it does not always produce the correct answer. Modify the professor's method so that it always produces the correct answer in linear time.

Exercises 33.1-6

Given a point $p_0 = (x_0, y_0)$, the ***right horizontal ray*** from p_0 is the set of points $\{p_i = (x_i, y_i) : x_i \geq x_0 \text{ and } y_i = y_0\}$, that is, it is the set of points due right of p_0 along with p_0 itself. Show how to determine whether a given right horizontal ray from p_0 intersects a line segment $\overline{p_1 p_2}$ in $O(1)$ time by reducing the problem to that of determining whether two line segments intersect.

Exercises 33.1-7

One way to determine whether a point p_0 is in the interior of a simple, but not necessarily convex, polygon P is to look at any ray from p_0 and check that the ray intersects the boundary of P an odd number of times but that p_0 itself is not on the boundary of P. Show how to compute in $\Theta(n)$ time whether a point p_0 is in the interior of an n-vertex polygon P. (*Hint:* Use Exercise 33.1-6. Make sure your algorithm is correct when the ray intersects the polygon boundary at a vertex and when the ray overlaps a side of the polygon.)

Exercises 33.1-8

Show how to compute the area of an n-vertex simple, but not necessarily convex, polygon in $\Theta(n)$ time. (See Exercise 33.1-5 for definitions pertaining to polygons.)

[1]Actually, the cross product is a three-dimensional concept. It is a vector that is perpendicular to both p_1 and p_2 according to the "right-hand rule" and whose magnitude is $|x_1 y_2 - x_2 y_1|$. In this chapter, however, it will prove convenient to treat the cross product simply as the value $x_1 y_2 - x_2 y_1$.

PREV < Day Day Up > NEXT