Algorithms and Data Structures: Computational Geometry I and II

8th and 12th Nov, 2013

Computational Geometry

In general, we will be considering 2-dimensional geometric problems (problems in the real plane).

Notation and basic definitions

- **Points** are pairs \((x, y)\) with \(x, y \in \mathbb{R}\).
- A **convex combination** of two points \(p_1 = (x_1, y_1)\) and \(p_2 = (x_2, y_2)\) is a point \(p = (x, y)\) such that
  \[
  x = \alpha x_1 + (1 - \alpha) x_2 \\
  y = \alpha y_1 + (1 - \alpha) y_2
  \]
  for some \(0 \leq \alpha \leq 1\).

Abbreviate to \(p = \alpha p_1 + (1 - \alpha) p_2\).

Intuitively, a point \(p\) is a convex combination of \(p_1\) and \(p_2\) if it is on the line segment from \(p_1\) to \(p_2\).

Undirected line segment \(p_1p_2\) (set of all convex combinations of \(p_1\) and \(p_2\))

Directed line segment \(\overrightarrow{p_1p_2}\):

When \(p_1 = (0, 0)\), the origin, treat \(\overrightarrow{p_1p_2}\) as the vector \(p_2\).
Clockwise and Counterclockwise from a Vector

Given \( p = (x_p, y_p) \), \( q = (x_q, y_q) \). Define cross product by:

\[
\mathbf{p} \times \mathbf{q} = \det \begin{pmatrix} x_p & x_q \\ y_p & y_q \end{pmatrix} = x_p y_q - x_q y_p.
\]

Signed area of parallelogram spanned by vectors \( \mathbf{p}, \mathbf{q} \):

\[
\mathbf{p} \times \mathbf{q}.
\]

Basic Problems

1. Given \( \overrightarrow{p_0p_1} \) and \( \overrightarrow{p_0p_2} \) is \( \overrightarrow{p_0p_1} \) collinear with, clockwise or counterclockwise from \( \overrightarrow{p_0p_2} \) w.r.t. \( p_0 \)?
2. Given \( \overrightarrow{p_1p_2} \) and \( \overrightarrow{p_2p_3} \), if we traverse \( \overrightarrow{p_1p_2} \) and then \( \overrightarrow{p_2p_3} \) do we make a left, a right, or no turn at \( p_2 \)?
3. Do \( \overrightarrow{p_1p_2} \) and \( \overrightarrow{p_3p_4} \) intersect?

Design aim: use only +, −, × and comparisons. Avoid division and trigonometric functions.

Straightforward Solutions

Use division and/or trigonometric functions. Not our approach

- For Problem (1) (special case with \( p_0 = (0, 0) \), \( p_2 = (x_2, 0) \)):

  \[ 0 < \angle(p_1, p_2) < \pi \quad \iff \quad \sin(\angle(p_1, p_2)) > 0. \]

  (It turns out, however, that we can compute the sign of \( \sin(\angle(p_1, p_2)) \) precisely without using either division or trigonometric functions.)

  In measuring the angle from vector \( p_1 \) round to vector \( p_2 \), we measure anti-clockwise from \( p_1 \). It is a convention.

- For Problem (3):

  - Compute intersection point \( p \) of lines through \( p_1, p_2 \) and through \( p_3, p_4 \) (if no such point exists, then the segments \( \overrightarrow{p_1p_2} \) and \( \overrightarrow{p_3p_4} \) do not intersect).
  - Then check if \( p \) is on both segments.

Cross product

Given \( p = (x_p, y_p) \), \( q = (x_q, y_q) \). Define cross product by:

\[
\mathbf{p} \times \mathbf{q} = \det \begin{pmatrix} x_p & x_q \\ y_p & y_q \end{pmatrix} = x_p y_q - x_q y_p.
\]

Intuitively: Signed area of parallelogram spanned by vectors \( p, q \):
Properties of the Cross Product

Lemma 1

\( p = (x_p, y_p), q = (x_q, y_q) \) points in the plane. Then

1. \( p \times q = -q \times p \)
2. – If \( p \times q > 0 \), then vector \( p \) is clockwise from \( q \).
   – If \( p \times q = 0 \), then vectors \( p \) and \( q \) are collinear.
   – If \( p \times q < 0 \), then vector \( p \) is counterclockwise from \( q \).

**Proof:** (1) is immediate from the definition. (2) is elementary analytical geometry. For homework, first compute the line through \((0,0)\) and \(q\). Then check where \(p\) should lie in relation to this line - there are 2 cases, \(x_q \geq 0\) and \(x_q < 0\).

Solution to Problem (1)

**Problem**

Given \( \overrightarrow{p_0p_1} \) and \( \overrightarrow{p_0p_2} \), is \( \overrightarrow{p_0p_1} \) collinear with, clockwise or anti-clockwise from \( (1,1) \) (which is anti-clockwise of the vector).

**Solution**

\[ (p_1 - p_0) \times (p_2 - p_0) = (x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0) \]

**Tip**

Can do a test: eg check vector \((0,0)(2,0)\) against the point \((1,1)\) (which is anti-clockwise of the vector).

Solution to Problem (2)

**Problem**

Given \( \overrightarrow{p_0p_1} \) and \( \overrightarrow{p_1p_2} \), if we traverse \( \overrightarrow{p_0p_1} \) and then \( \overrightarrow{p_1p_2} \) do we make a left, a right, or no turn at \( p_1 \)?

**Solution**

\[ (p_1 - p_0) \times (p_2 - p_0) = 0: \text{collinear segments — no turn.} \]
\[ (p_1 - p_0) \times (p_2 - p_0) < 0: \text{right turn at } p_1. \]
\[ (p_1 - p_0) \times (p_2 - p_0) > 0: \text{left turn at } p_1. \]

Solution to Problem (3)

**Problem**

\( \overrightarrow{p_1p_2} \) and \( \overrightarrow{p_3p_4} \) intersect?

**Solution**

\( \overrightarrow{p_1p_2} \) straddles \( \overrightarrow{p_3p_4} \) if \( p_1 \) and \( p_2 \) lie on different sides of the line through \( p_3, p_4 \).
Then \( \overrightarrow{p_1p_2} \) and \( \overrightarrow{p_3p_4} \) intersect if, and only if, one of the following conditions holds:

- \( \overrightarrow{p_1p_2} \) straddles \( \overrightarrow{p_3p_4} \) and \( \overrightarrow{p_3p_4} \) straddles \( \overrightarrow{p_1p_2} \).
- An endpoint of one segment lies on the other.
4 cases for the Intersection Question

L intersects with L₃ (one point of L₃ lies on L) and with L₄ (both “straddle tests” succeed).
L does not intersect L₂ (only one of the “straddle tests” succeeds) or L₁.

Point on Segment

p₃ is on segment p₁p₂ if

\[(p₃ - p₁) \times (p₂ - p₁) = 0\]

and

\[\min(x₁, x₂) \leq x₃ \leq \max(x₁, x₂)\]

and

\[\min(y₁, y₂) \leq y₃ \leq \max(y₁, y₂)\]

The last two conditions simply say that p is in the rectangle with (diagonally opposite) corner points p₁, p₂.

Solution of Problem (3) Completed

Algorithm \textsc{Segments-Intersect}(p₁, p₂, p₃, p₄)
1. \(d_{12,3} \leftarrow (p₃ - p₁) \times (p₂ - p₁)\)
2. \(d_{12,4} \leftarrow (p₄ - p₁) \times (p₂ - p₁)\)
3. \(d_{34,1} \leftarrow (p₁ - p₃) \times (p₄ - p₃)\)
4. \(d_{34,2} \leftarrow (p₂ - p₃) \times (p₄ - p₃)\)
5. \textbf{if} \(d_{12,3}d_{12,4} < 0 \text{ and } d_{34,1}d_{34,2} < 0\) \textbf{then return} TRUE
6. \textbf{else if} \(d_{12,3} = 0 \text{ and } \textsc{In-Box}(p₁, p₂, p₃)\) \textbf{then return} TRUE
7. \textbf{else if} \(d_{12,4} = 0 \text{ and } \textsc{In-Box}(p₁, p₂, p₄)\) \textbf{then return} TRUE
8. \textbf{else if} \(d_{34,1} = 0 \text{ and } \textsc{In-Box}(p₃, p₄, p₁)\) \textbf{then return} TRUE
9. \textbf{else if} \(d_{34,2} = 0 \text{ and } \textsc{In-Box}(p₃, p₄, p₂)\) \textbf{then return} TRUE
10. \textbf{else return} FALSE

Algorithm \textsc{In-Box}(p₁, p₂, p₃)
1. \textbf{return} \(\min(x₁, x₂) \leq x₃ \leq \max(x₁, x₂)\)
   \textbf{and} \(\min(y₁, y₂) \leq y₃ \leq \max(y₁, y₂)\)


Straddle Test

p₁p₂ straddles p₃p₄ if, and only if,
\[\left(\left(p₁ - p₃\right) \times \left(p₄ - p₃\right)\right)\left(\left(p₂ - p₃\right) \times \left(p₄ - p₃\right)\right) < 0.\]
Detecting Intersection of Line Segments

Input: A number of line segments (n in total).
Problem: Determine whether at least one pair of segments intersect.

Note: Obvious method takes $\Omega(n^2)$ time in worst-case.

Simplifying assumptions:
- No segment is vertical.
- No three segments intersect at the same point.

The Sweep Line Algorithm

IDEA
- Move vertical sweep line across the line segments and record, at each position, which line segments intersect the sweep line.
- This enables us to determine intersections efficiently, because it suffices to check for intersections between neighboured line segments.

Example

Moving the sweep-line

Manage two sets of data:

- **Sweep-line status**: Gives relationship among objects intersected by sweep-line at a position. (Defines a total order at each position $x$ of sweep-line.)
- **Event-point schedule**: Sequence of $x$-coordinates in increasing order. Each point marks a change in the sweep-line status: call these event points.
Data Structure for the Sweep-Line Status

▶ Line segments are represented by as pairs \((p_1, p_2)\), where we always assume that \(x_1 < x_2\) (remember we assume that there are no vertical segments).

▶ We use an ordered set data structure to store the sweep line status, i.e., the set of lines currently intersecting the sweep line.

▶ Note that the order of the segments on the sweep line does not change unless there is an intersection. Thus we only need to put lines entering the sweep line in the right place (and test for intersections as lines leave the sweep-line).

Ordered-Set Data Structures

Stores are collection of elements which can be compared by some function \(\text{LESS-THAN}(e_1, e_2)\).

Support the following operations:

- **INSERT(e)**: Insert element \(e\) into set.
- **DELETE(e)**: Remove element \(e\) from set.
- **BELOW(e)**: Return the largest element below \(e\) in the order determined by \(\text{LESS-THAN}\), or \(\text{NIL}\) if no such element exists.
- **ABOVE(e)**: Return the smallest element above \(e\) in the order determined by \(\text{LESS-THAN}\), or \(\text{NIL}\) if no such element exists.

Can be implemented by balanced search trees such as AVL trees (cf. Inf2B).

All operations require time \(O(\lg n)\).
Implementation

Algorithm Any-Segments-Intersect(S)
1. $T \leftarrow 0$
2. sort endpoints of segments in $S$ lexicographically
3. for each endpoint $p$ do
4.    if $p$ is left endpoint of segment $s$ then
5.        $T$.INSERT($s$)
6.    if $s$ intersects $T$.ABOVE($s$) or $T$.BELOW($s$) then
7.        return TRUE
8.    if $p$ is right endpoint of segment $s$ then
9.        if $T$.ABOVE($s$) and $T$.BELOW($s$) intersect then
10.       return TRUE
11.    $T$.DELETE($s$)
12. return FALSE

Proof of Correctness
⇒: If Any-Segments-Intersect($S$) returns TRUE, then there is an intersection.
Proof: This is the simpler part of the Proof. We only need to note that in both places where the Algorithm may return TRUE (lines 7. and 10.), this return is conditional on a test-of-intersection succeeding immediately beforehand.

⇐: Now we must prove that if there is at least one pair of line segments $s_1, s_2 \in S$ that intersect, then Any-Segments-Intersect($S$) is guaranteed to return TRUE.
Proof: Let $s_1, s_2$ be the pair of segments with the leftmost intersection point. (Break ties by taking the bottom-most intersection point.) Assume without loss of generality that $s_2$ enters the sweep line before $s_1$. There are two sub cases.

case (i): If $s_1$ is a neighbour of $s_2$ on the sweep line immediately after $s_2$ entered the sweep line, the intersection will be detected in line 6. Hence the Algorithm will return TRUE on line 7.

Proof of Correctness cont’d

case (ii): Alternatively, suppose that that $s_1$ is not a neighbour of $s_2$ on the sweep line immediately after $s_2$ has entered. Let $s_3, \ldots, s_k$ denote the segments between $s_2$ and $s_1$ on the sweep line at this stage.

- None of the segments $s_3, \ldots, s_k$ intersect $s_1$ or $s_2$, because the intersection of $s_1$ and $s_2$ is the leftmost intersection.
- Thus $s_3, \ldots, s_k$ must depart from the sweep line before the intersection point between $s_1$ and $s_2$. Let $s_l$ be the last of these segments to leave.
- Then the intersection of $s_1$ and $s_2$ is detected in line 9 immediately before $s_l$ is deleted, and TRUE is returned by line 10. immediately after.

Analysis
Let $n$ be the number of line segments in $S$.
- Line 2 requires time $\Theta(n \log n)$.
- Lines 5, 6, 9, 11 require time $\Theta(\log n)$
- All other lines require time $\Theta(1)$.
- Loop in lines 3–11 is iterated $n$ times. Thus the execution of the loop requires time $\Theta(n \log n)$.

Overall Running Time:
$\Theta(n \log n)$. 


Reading Assignment
Sections 33.1-2, pages 933-947, of [CLRS].

Problems

1. Exercises 33.1-5 and 33.1-6 of [CLRS].
2. Exercises 33.2-1 and 33.2-4 of [CLRS].