

Algorithms and Data Structures: Computational Geometry

18th & 22rd Nov, 2011

ADS: lects 15 & 16 – slide 1 – 18th & 22rd Nov, 2011

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

$$x = \alpha x_1 + (1 - \alpha)x_2$$

$$y = \alpha y_1 + (1 - \alpha)y_2$$

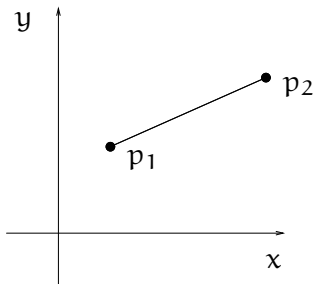
for some $0 \leq \alpha \leq 1$.

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

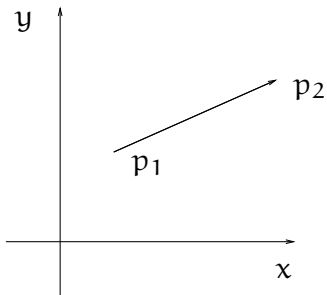
Line Segments

Undirected line segment $\overline{p_1 p_2}$ (set of all convex combinations of p_1 and p_2)



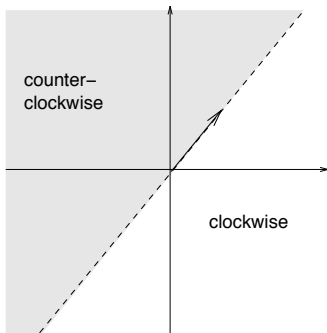
Directed Line Segments

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



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 p_0, p_1 and p_2 , if we traverse $\overline{p_0p_1}$ and then $\overline{p_1p_2}$ (ie, walk from p_0 to p_1 to p_2) do we make a left, a right, or no turn at p_1 ?
3. Do $\overline{p_1p_2}$ and $\overline{p_3p_4}$ intersect?

Design aim: use only $+$, $-$, \times and comparisons.

Avoid division and trigonometric functions.

Straightforward Solutions

IF WE USE division and/or trigonometry.

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

vector p_1 is clockwise from vector p_2

$$\iff 0 < \angle(p_1, p_2) < \pi \iff \sin(\angle(p_1, p_2)) > 0.$$

(However, slide 9 shows how to compute the *sign* of $\sin(\angle(p_1, p_2))$ **without using division or trigonometric functions (*hurrah!*)**.)

PS: In measuring the angle from vector p_1 round to vector p_2 , we measure anti-clockwise from p_1 . It is a convention (which I don't like).

- ▶ 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 $\overline{p_1 p_2}$ and $\overline{p_3 p_4}$ do not intersect).
 - ▶ Then check if p is on both segments.

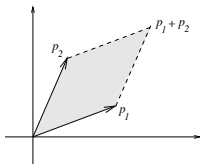
See slides 12-15 for the **no trigonometry, no division solution!!**

Cross product

Given $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$. Define *cross product* (for this application) by:

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

Intuitively: *Signed area of parallelogram spanned by vectors p_1 , p_2 :*



Note: *Official definition of cross product is a vector; here, we just look at the vector's length.*

Properties of the Cross Product

Lemma 1

p_1, p_2 points in the plane. Then

1. $p_1 \times p_2 = -p_2 \times p_1$

2.

- If $p_1 \times p_2 > 0$, then vector p_1 is clockwise from p_2 .
- If $p_1 \times p_2 = 0$, then vectors p_1 and p_2 are *collinear*.
- If $p_1 \times p_2 < 0$, then vector p_1 is counterclockwise from p_2 .

Proof: (1) is immediate from the definition, (2) is elementary analytical geometry (Let $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$). For homework, first compute the line through $(0, 0)$ and p_2 . Then check where p_1 should lie in relation to this line - there are 2 cases, $x_2 \geq 0$ and $x_2 < 0$).

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 $\overrightarrow{p_0p_2}$ w.r.t. p_0 ?

Solution

Shift common endpoint to origin and use cross product.
Just examine sign of:

$$(p_1 - p_0) \times (p_2 - p_0) = (x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0).$$

Tip

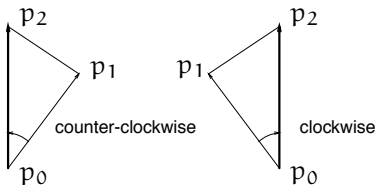
Can be hard to remember which sign (+ or -) matches clockwise etc. Can do a test: eg check vector $\overrightarrow{(0,0)(2,0)}$ and the point (1,1) (which is anti-clockwise)

Solution to Problem (2)

Problem

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

Solution



$(p_2 - p_0) \times (p_1 - p_0) = 0$: collinear segments — no turn.

$(p_2 - p_0) \times (p_1 - p_0) < 0$: left turn at p_1 .

$(p_2 - p_0) \times (p_1 - p_0) > 0$: right turn at p_1 .

Solution to Problem (3)

Problem

$\overline{p_1p_2}$ and $\overline{p_3p_4}$ intersect?

Solution

$\overline{p_1p_2}$ straddles $\overline{p_3p_4}$ if p_1 and p_2 lie on different sides of the line through p_3, p_4 .

Then $\overline{p_1p_2}$ and $\overline{p_3p_4}$ intersect if, and only if, one of the following conditions holds:

- ▶ $\overline{p_1p_2}$ straddles $\overline{p_3p_4}$ and $\overline{p_3p_4}$ straddles $\overline{p_1p_2}$.
- ▶ An endpoint of one segment lies on the other.

Straddle Test

$\overline{p_1 p_2}$ straddles $\overline{p_3 p_4}$ if, and only if,

$$((p_1 - p_3) \times (p_4 - p_3))((p_2 - p_3) \times (p_4 - p_3)) < 0.$$

Point on Segment

p_3 is on segment $\overline{p_1p_2}$ if

$$(p_3 - p_1) \times (p_2 - p_1) = 0$$

and

$$\min(x_1, x_2) \leq x_3 \leq \max(x_1, x_2)$$

and

$$\min(y_1, y_2) \leq y_3 \leq \max(y_1, y_2)$$

The last two conditions simply say that p is in the rectangle with (diagonally opposite) corner points p_1, p_2

Solution of Problem (3) Completed

Algorithm SEGMENTS-INTERSECT(p_1, p_2, p_3, p_4)

1. $d_{12,3} \leftarrow (p_3 - p_1) \times (p_2 - p_1)$
2. $d_{12,4} \leftarrow (p_4 - p_1) \times (p_2 - p_1)$
3. $d_{34,1} \leftarrow (p_1 - p_3) \times (p_4 - p_3)$
4. $d_{34,2} \leftarrow (p_2 - p_3) \times (p_4 - p_3)$
5. **if** $d_{12,3}d_{12,4} < 0$ **and** $d_{34,1}d_{34,2} < 0$ **then return** TRUE
6. **else if** $d_{12,3} = 0$ **and** IN-BOX(p_1, p_2, p_3) **then return** TRUE
7. **else if** $d_{12,4} = 0$ **and** IN-BOX(p_1, p_2, p_4) **then return** TRUE
8. **else if** $d_{34,1} = 0$ **and** IN-BOX(p_3, p_4, p_1) **then return** TRUE
9. **else if** $d_{34,2} = 0$ **and** IN-BOX(p_3, p_4, p_2) **then return** TRUE
10. **else return** FALSE

Algorithm IN-BOX(p_1, p_2, p_3)

1. **return** $\min(x_1, x_2) \leq x_3 \leq \max(x_1, x_2)$
and $\min(y_1, y_2) \leq y_3 \leq \max(y_1, y_2)$

The Convex Hull

Definition 2

1. A set C of points is *convex* if for all $p, q \in C$ the whole line segment \overline{pq} is contained in C .
2. The convex hull of a set S of points is the smallest convex set C that contains S .

Observation 3

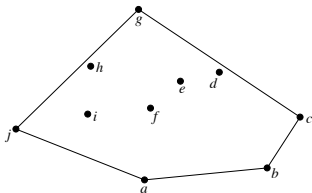
The convex hull of a finite set S of points is a convex polygon whose vertices (corner points) are elements of S .

The Convex Hull Problem

Input: *A finite set S of points in the plane*

Output: *The vertices of the convex hull of S in counterclockwise order.*

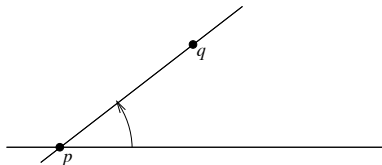
Example 4



Output of a convex-hull algorithm: a, b, c, g, j

Polar Angles

The *polar angle* of a point q with respect to a point p is the angle between a horizontal line and the line through p and q .



Lemma

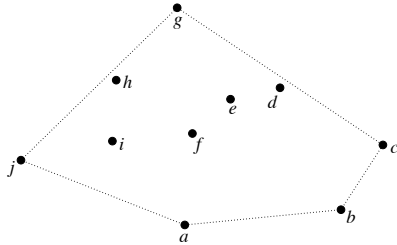
There is an algorithm that, given points p_0, p_1, \dots, p_n , sorts p_1, \dots, p_n by non-decreasing polar angle with respect to p_0 in $O(n \lg n)$ time (How? - nice homework exercise).

Graham's Scan

IDEA

- ▶ Let p_0 be a bottom-most point. Start walking around the points in the order of increasing polar angles.
- ▶ As long as you turn left, keep on walking.
- ▶ If you have to turn right to reach the next point, discard the current point and step back to the previous point. Repeat this until you can turn left to the next point.
- ▶ The points that remain are the vertices of the convex hull.

Example



Implementation

Algorithm GRAHAM-SCAN(Q)

1. Let p_0 be the point in Q with minimum y coordinate.
In case of a tie, take the leftmost point.
2. Sort the points in $Q \setminus \{p_0\}$ by non-decreasing polar angles with respect to p_0 .
If several points have the same polar angle, discard all but the one farthest away from p_0 .
Let $\langle p_1, \dots, p_m \rangle$ be the resulting list
3. **if** $m \leq 1$ **then return** $\langle p_0, \dots, p_m \rangle$
4. Initialise stack S
5. $S.PUSH(p_0)$
6. $S.PUSH(p_1)$
7. $S.PUSH(p_2)$
8. **for** $i \leftarrow 3$ **to** m **do**
9. **while** the angle formed by the topmost two elements of S and p_i does not make a left turn
 do
10. $S.POP$
11. $S.PUSH(p_i)$
12. **return** S

Analysis

Let $n = |Q|$, then $m \leq n$.

- ▶ Lines 3–7, 12 require time $\Theta(1)$.
- ▶ Line 1 requires time $\Theta(n)$ in the worst case.
- ▶ Line 2 requires time $\Theta(n \lg n)$
- ▶ The outer loop in lines 8–11 is iterated $m - 2$ times. Thus, disregarding the time needed by the inner loop, the loop requires time $\Theta(m) = O(n)$.
- ▶ The inner loop in lines 9–10 is executed at most once for each element, because every element enters the stack at most once and thus can only be popped once. Thus overall the inner loop requires time $O(n)$.

Thus the overall worst-case running time is

$$\Theta(n \lg n).$$

Proof of Correctness

Let p_0, p_1, \dots, p_m be the list of points obtained after executing line 2.

1. The vertices of the convex hull are among p_0, p_1, \dots, p_m .

Proof: If a point q is discarded because it has the same polar angle as some other point p_i , then q is contained in $\overline{p_0 p_i}$, because p_i is farther from p_0 than q is.

Thus q is contained in the convex hull of p_0, p_1, \dots, p_m .

2. If $m \leq 1$ (at most two points), the algorithm obviously works correctly by returning all points.

3. Otherwise, $m \geq 2$. For $2 \leq i \leq m$, let C_i denote the convex hull of p_0, \dots, p_i .

We will prove the following: *After the execution of the main outer loop with point p_i , the points on S are the vertices of C_i in clockwise order.* (*)

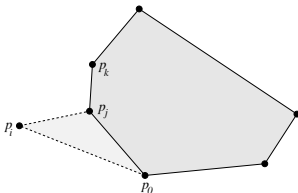
“Base case”: ($i = 2$): The convex hull is p_0, p_1, p_2 . These are the points we add to S , in that order. So claim holds (note we don't actually execute the loop for $i = 2$).

Induction step: Now let $i \geq 3, i \leq m$.

- ▶ We know the points on S **BEFORE** the ' p_i '-execution of the outer loop are the vertices of C_{i-1} in clockwise order. This holds by the “Base case” (if $i = 3$) or by the induction hypothesis (if $i > 3$).

Proof of Correctness cont'd

- ▶ Since the polar angle of p_i is greater than the polar angle of p_{i-1} , $p_0 p_{i-1} p_i$ form a triangle not contained in C_{i-1} .



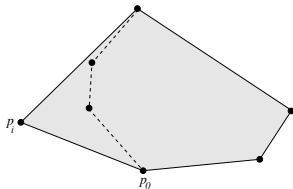
In particular, p_i is not contained in C_{i-1} and thus is a vertex of C_i .

- ▶ Let p_j and p_k be the two topmost elements of the stack S . If the angle formed by $p_k p_j p_i$ makes a right turn, then the triangle $p_k p_j p_i$ is contained in the triangle $p_k p_0 p_i$. Thus p_j is contained in the convex hull of p_0, p_k, p_i and thus not a vertex of C_i .

Therefore, it is correct to remove p_j from S in line 10.

Proof of Correctness cont'd

- ▶ On the other hand, if the angle formed by $p_k p_j p_i$ makes a left turn, then the triangle $p_k p_j p_i$ is not contained in the triangle $p_k p_0 p_i$. Thus p_j is a vertex of C_i .



- ▶ Moreover, it can be easily seen that all vertices of C_{i-1} between p_0 and p_j in the counterclockwise order remain vertices of C_j . So after the execution of line 11, S contains all vertices of C_i . Thus (\star) is proved.
5. Statement (\star) for $i = m$ proves the correctness of the algorithm.

Remark: The correctness “proof” given above is still not a full mathematical proof, because some intuitive facts about convex polygons were just ‘claimed’. But all the missing details can be filled in.

ADS: lects 15 & 16 – slide 25 – 18th & 22rd Nov, 2011

Optimality

- ▶ The best-known algorithm for finding the convex hull has a running time of $O(n \lg h)$, where h is the number of vertices of the convex hull.
- ▶ On the other hand, it can be proved that every algorithm for finding the convex hull has a worst-case running time of

$$\Omega(n \lg n).$$

The *proof* of this lower bound is due to the fact that we can implement sorting using Convex Hull (Q5 of week 10 tutorial sheet).

Reading Assignment

Section 33.3, pages 947-957, of [CLRS]. *This is Section 35.3, pages 898-908, of [CLR].*

Problems

1. Exercise 33.3-5, page 957, of [CLRS]. *Ex 35.3-5, page 907, of [CLR].*
2. Prove that the problem of finding the Convex Hull of n points has a lower bound of $\Omega(n \lg n)$. For this, think about using a reduction from sorting to Convex Hull (that is, think about how to use a Convex Hull algorithm to sort a list of numbers).