Computational Geometry

Yongkoo Kang, Ryan Jian

November 15, 2013

1 Introduction

Computational Geometry really sucks and no one likes it, but it shows up in contests and is something you should know. Most of it revolves around somewhat routine uses of the dot product and the cross product which is not completely intuitive at first but does make sense. However, one of the biggest issues in most computational geometry problems is the plethora of edge-like cases which break for reasons ranging from concurrency to double errors.

2 Vectors

So. What is a vector? In this case, the vector we are referring to is a quantity with both a direction and a magnitude. Vectors are commonly used to represent directed line segments. Representation of a vector is usually in its componentized form, where its projections along the major axes are used as quantities by which it is defined. Usually we will be working in three dimensions so

$$\mathbf{v} = v_x\mathbf{i} + v_y\mathbf{j} + v_z\mathbf{k}$$

Adding two vectors and multiplying a vector by a scalar is done component-wise.

$$\mathbf{u} + \mathbf{v} = (u_x + v_x)\mathbf{i} + (u_y + v_y)\mathbf{j} + (u_z + v_z)\mathbf{k}$$

$$k\mathbf{v} = kv_x\mathbf{i} + kv_y\mathbf{j} + kv_z\mathbf{k}$$

2.1 Dot Product

The dot product is a quantity which is the length of one vector times the length of the amount of another vector parallel to that vector. Given two vectors u and v, the dot product of $\mathbf{u}$ and $\mathbf{v}$, denoted by $\mathbf{u} \cdot \mathbf{v} = |\mathbf{u}||\mathbf{v}|\cos(\theta)$. From this, you can see that if two vectors are perpendicular, their dot product will be 0. Given two vectors $\mathbf{v} = v_x i + v_y j + v_z k$ and $\mathbf{w} = w_x i + w_y j + w_z k$, $\mathbf{v} \cdot \mathbf{w} = v_x * w_x + v_y * w_y + v_z * w_z$.

2.2 Cross Product

The cross product in three dimensions is given by

$$\mathbf{u} \times \mathbf{v} = (u_y v_z - u_z v_y)\mathbf{i} + (u_z v_x - u_x v_z)\mathbf{j} + (u_x v_y - u_y v_x)\mathbf{k}$$

which can be remembered as the determinant of the following matrix:

$$\begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ u_x & u_y & u_z \\ v_x & v_y & v_z \end{vmatrix}$$

This formula gives the cross product a number of very useful properties. The cross product takes two vectors and produces a vector that is perpendicular to both vectors. Having this property means that here are two opposing directions possible for the resulting vector to take, however the formula clearly only gives us one of these. The direction of the vector computed by the cross product is given by the right hand rule. Point your thumb up. Orient your index finger in the direction of $\mathbf{u}$ and your middle finger in the direction of $\mathbf{v}$. Your thumb now points in the direction of $\mathbf{u} \times \mathbf{v}$

The cross product is also not commutative, instead $u \times v = -v \times u$. It will also turn out that $|\mathbf{u} \times \mathbf{v}| = |\mathbf{u}||\mathbf{v}|\sin(\theta)$ where theta is the angle between the two vectors. Geometrically, this means that the magnitude of the cross product gives the area of the parallelogram that is formed by $\mathbf{u}$ and $\mathbf{v}$. Notice that this means the cross product of two parallel vectors is always zero.

2.3 Subtle Usages

Distance from a Point to a Line: Given a line defined by the line segment AB and a point P which are given as position vectors, the distance from P to AB is

$$\frac{|(P-A) \times (B-A)|}{|B-A|}$$

Distance from a Point to a Line Segment Check if $\triangle PAB$ is acute. If so, use the point to line distance formula above. Otherwise take the minimum of the lengths of PA and PB.

Same Side of a Line In two dimensions, two points, C and D, are on the same side of a line, AB, if the z components of $(B-A) \times (C-A)$ and $(B-A) \times (D-A)$ have the same sign.

Line Segment Intersection Two line segments AB and CD intersect if A and B are on different sides of CD and if C and D are on different sides of AB.

Point in Triangle By extension of the previous observation, to see if a point is in a triangle, choose a known point, such as the average of the three vertices, and see if both points are on the same side of all of the sides. (This also works for convex polygons in general).

Coplanar Points Given four points A, B, C, and D, D is coplanar with A, B, and C if $((B-A) \times (C-A)) \cdot (D-A)$ is sufficiently close to 0.

Convexity of a Polygon Go clockwise around the polygon and for every triplet of vertices, calculate the cross product $(B-A) \times (C-A)$. If the z component of every single one of these products is positive, the polygon is convex.

Area of a Simple Polygon Given the points of vertices (x_i, y_i) numbered from $i = 1$ to n of the polygon sorted in counterclockwise/clockwise order.

$$A = \frac{1}{2}\left|\sum_{i=1}^{n-1}(x_i y_{i+1}) + x_n y_1 - \sum_{i=1}^{n-1}(x_{i+1} y_i) - x_1 y_n\right|$$

This is known as the Shoelace formula because writing out the points in a two column matrix and drawing lines between pairs of elements that are multiplied makes the matrix resemble a shoe with its laces done.

Point inside a Polygon To see whether or not a point is inside a polygon, draw a ray that does not intersect one of the polygon's vertices from the point out (either choose a ray with a large slope or randomly choose a slope and repeat if the ray intersects any vertices) If this ray intersects the polygon an odd number of times, the point is inside otherwise, it is outside.

3 Line Sweep

Line sweep is a general method to solve computational geometry problems by sweep through the input points in order of increasing x coordinates. The key to line sweep is to have the algorithm maintain information about the points that were processed previously. Usually we want to use an efficient data structure to maintain this information, such as a balanced binary search tree or a heap. Take a look at problems 5 and 6.

4 Implementation

Working with floating point values is annoying due to precision errors. Therefore it is a good idea to avoid division as much as possible. For example when sorting lines by slope we can manipulate $\frac{x_i}{y_i} < \frac{x_j}{y_j}$ into $x_i * y_j < x_j * y_i$ and avoid explicitly computing the slopes.

If you have to work with floating point numbers remember to never compare them for equality directly. Instead check if $|a - b| < \epsilon$, where $\epsilon = 10^{-7}$.

5 Problems

1. Given $N(1 \leq N \leq 100,000)$ points, find the line through the origin that intersects the most number of points.
2. (USACO MAR11 rotsym) Given $N(1 \leq N \leq 2000)$ points, count the number of parallelograms formed by the points.
3. Given $N(1 \leq N \leq 100,000)$ lines in the plane,

   nd the x-coordinate of the leftmost intersection.
4. (USACO DEC08 fence) Given $N(1 \leq N \leq 250)$ posts in the plane, what is the greatest number of posts Farmer John can include in a convex fence?
5. (USACO NOV13 crowded) FJ's $N(1 \leq N \leq 50,000)$ cows are grazing along a one-dimensional fence. Cow i is standing at location $x(i)$ and has height $h(i)(1 \leq x(i), h(i) \leq 1,000,000,000)$. A cow feels "crowded" if there is another cow at least twice her height within distance D on her left, and also another cow at least twice her height within distance D on her right $(1 \leq D \leq 1,000,000,000)$. Count the number of such cows.
6. (USACO FEB12 planting) Farmer John's cows eat grass in rectangular areas. Given a list of these rectangular areas, compute the total area of grass that cows have eaten from in $O(N \log(N))$. Note that parts of areas where the cows eat may intersect other cows' grazing areas.
7. (Codeforces Round #152 D) Given $N(1 \leq N \leq 100,000)$ points in the second quadrant and two angles, Donkey starts from the origin and can move to any point that lies between two rays shooting out from the origin at the given angles with respect to the x axis. He then repeats the process, except now he can only go to points lying between two rays starting from the new point he is at. What is the most number of points Donkey can visit in this manner?

Formulas taken mostly from Jonathan Wang