2D Geometry Introduction

The equation of a line

The equation of a line in two dimensions is typically taught in beginning algebra as

$$y = mx + b$$

where m is the slope of the line and b is the y-intercept. For any two points (x_1, y_1), (x_2, y_2) on the line, $m = \frac{y_2 - y_1}{x_2 - x_1}$. b can be found by setting $y_1 = m \cdot x_1 + b$ and solving for b. If there are two lines, then there are two equations in two variables, which can be solved for the x and y of the intersection point.

This form of the line equation is called the *implicit* equation. Given any x, you can find a matching y; however, it does not tell you how to generate (x, y) coordinate pairs from scratch. The implicit form has some pitfalls when used in code: for example, consider the vertical line $x = 5$. The slope m for this line is infinity, and b is undefined as the line never intersects the y axis!

An alternate form of the line equation is actually two equations: one for x and one for y. If we have two points on the line $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, then the *parametric* equation of the line containing P_1 and P_2 is

$$x(t) = x_1 + t(x_2 - x_1)$$
$$y(t) = y_1 + t(y_2 - y_1)$$

We've introduced a parameter variable t; we now have two functions $x(t)$ and $y(t)$ that generate points on the line. Often, two variables $\Delta x = x_2 - x_1$ and $\Delta y = y_2 - y_1$ are defined, giving the form

$$x(t) = x_1 + t\Delta x$$
$$y(t) = y_1 + t\Delta y$$

This form has several nice properties. It has no fractions; horizontal and vertical lines can be represented without a special case. If we are actually representing a line *segment* with endpoints P_1 and P_2, then $t = 0$ gives P_1 and $t = 1$ gives P_2. Thus we can see if a point on the line is in the segment just by checking $0 \le t \le 1$.

If we have two lines and we wants to find the intersection point, we have to solve for two parametric variables t_1 and t_2. We set

$$x_{1_1} + t_1\Delta x_1 = x_{2_1} + t_2\Delta x_2$$
$$y_{1_1} + t_1\Delta y_1 = y_{2_1} + t_2\Delta y_2$$

rearranging to two equations in terms of t_1 and t_2:

$$\Delta x_1 t_1 - \Delta x_2 t_2 = x_{2_1} - x_{1_1}$$
$$\Delta y_1 t_1 - \Delta y_2 t_2 = y_{2_1} - y_{1_1}$$

Once we solve for both t's, we can get the actual intersection point by plugging one of them back into the original parametric equation for its line. Using Cramer's rule (see end of handout if you aren't familar with it), we can immediately write

$$t_1 = \frac{(y_{2_1} - y_{1_1})\Delta x_2 - (x_{2_1} - x_{1_1})\Delta y_2}{\Delta x_2 \Delta y_1 - \Delta x_1 \Delta y_2}$$
$$t_2 = \frac{(y_{2_1} - y_{1_1})\Delta x_1 - (x_{2_1} - x_{1_1})\Delta y_1}{\Delta y_1 \Delta y_1 - \Delta x_1 \Delta y_2}$$

Note that the denominator for both solutions is a determinant solely involving the components of the slopes of both lines. If the denominator is zero, then the lines are parallel. If the lines are disjoint, then there is no intersection point; otherwise, they are the same line and every point on the line is a solution.

Example

The learn set problem *Birdman of Waikiki*, stripped of its story, asks us to determine if two line segments intersect. We are given the endpoints of both segments.

We can use the above method to solve for t on each line segment. If $0 \le t_1, t_2 \le 1$ then the lines intersect within the range of the given segments. If you coded this up, your code would give correct answers for the sample data and fail on submission.

As with most geometry problems, there are a lot of special cases. In this problem, we need to take special care when the denominator is zero. We first need to determine if the lines are disjoint or the same line. We can do this by testing if the endpoint from one segment lies on the other segment's line. If the lines are disjoint, there is no intersection.

If the lines are the same, we still aren't done. We are dealing with line *segments*, not lines, so we need to figure out if the segments on the same line intersect. We can do this by looking at either the x or y values of the endpoints of each segment and seeing if those ranges intersect. We again have to be careful of the cases where the line is horizontal or vertical.

The moral here is that, even more so than other contest problems, geometry problems tend to have a lot of edge cases. Proper representation of data can eliminate some of them, but thinking through and creating good tests cases is critical.

2D Geometry Introduction

Vectors

We can now introduce the concept of a *vector*. A point (x, y) is an ordered pair representing a position in the 2D plane. In the parametric form of the line, we used two values Δx and Δy to represent the distance between two points. If we put these values together as an ordered pair, the result is a 2D vector, written as $\vec{v} = [\Delta x, \Delta y]$. We can also write $\vec{v} = [v_x, v_y]$ to name the vector's components. Vectors are ubiquitous in both geometry and linear algebra.

Two vectors are added or subtracted by adding or subtracting each of their components. For example

$$\vec{p} + \vec{q} = [p_x, p_y] + [q_x, q_y] = [p_x + q_x, p_y + q_y]$$

Adding two vectors gives a new vector pointing in a new direction with a new length. Adding a vector to a point gives a point, moved in the plane by the direction and length of the vector (this operation is known as *translation*. Subtracting two points A and B gives a vector $\vec{v} = B - A$ that points from A to B. It is important to keep the distinction between a point and a vector in mind; they are represented in a similar way but are unique concepts.

Multiplication is a bit more complex; there are two ways to multiply two vectors that we'll cover in a bit. However, multiplying a vector by a number is easy. In this case, the number is called a *scalar* (which just means it's a single value, not a vector), and thus this operation is called scalar multiplication (also referred to as *scaling* the vector). It just means multiplying each component of the vector by the scalar:

$$s \cdot \vec{v} = s{\cdot}[v_x, v_y] = [s{\cdot}v_x, s{\cdot}v_y]$$

Scaling a vector by s has the effect of multiplying the vector's length by s: $|s\vec{v}| = s|\vec{v}|$.

Using these operations, we can write the parametric equation of a line more succinctly. If we have two points P_0 and P_1, and $\vec{d} = P_1 - P_0$, then the equation for the line containing them is

$$P = P_0 + t\vec{d}$$

The *magnitude* of a vector $\vec{v}$ is defined as

$$|\vec{v}| = \sqrt{v_x^2 + v_y^2}$$

If the vector is the difference between two points $\vec{v} = P_0 - P_1$, then $|\vec{v}|$ is the distance between those two points.

A *unit* vector (also called a *normalized* vector) is a vector of length 1. It is computed by multiplying a vector by the inverse of its length: $unit(\vec{v}) = \frac{1}{|\vec{v}|}\vec{v}$. Some formulas only work with unit vectors, and some with non-normalized vectors, and it's important to remember which is which.

The dot and cross products

The first way of multiplying two vectors is called the *dot product*. The dot product is simply the sum of the product of matching components in each vector:

$$\vec{p} \cdot \vec{q} = p_x q_x + p_y q_y$$

While we are only concerned with two dimensions here, the dot product is defined for vectors of any length. Note that the dot product of two vectors is not another vector, but a scalar. The dot product provides a handy way of finding the angle between two vectors due to this identity:

$$\vec{p} \cdot \vec{q} = |\vec{p}||\vec{q}| \cos(\theta)$$

where θ is the angle between the two vectors. If the two vectors are unit vectors, the equation simplifies to

$$\cos(\theta) = \vec{p} \cdot \vec{q}, \theta = \cos^{-1}(\vec{p} \cdot \vec{q})$$

In many geometry problems, we actually are looking for the cosine rather than the actual angle, and can just use the dot product directly.

Example

What is the angle between the two vectors in figure 1?

We have vector $\vec{a} = [5, 1]$ and $\vec{b} = [2, 3]$.

$$|\vec{a}| = \sqrt{5 \cdot 5 + 1 \cdot 1} \approx 5.1$$

$$|\vec{b}| = \sqrt{2 \cdot 2 + 3 \cdot 3} \approx 3.6$$

The normalized vectors are therefore

$$\vec{a}_{unit} = [\frac{5}{5.1}, \frac{1}{5.1}] \approx [0.98, 0.20]$$
$$\vec{b}_{unit} = [\frac{2}{3.6}, \frac{3}{3.6}] \approx [0.56, 0.83]$$

and

$$\cos(\theta) = \vec{a}_{unit} \cdot \vec{b}_{unit} = 0.98 \cdot 0.56 + 0.20 \cdot 0.83 = 0.71$$
$$\theta = \cos^{-1}(0.71) \approx 0.77(radians) \approx 44(degrees)$$

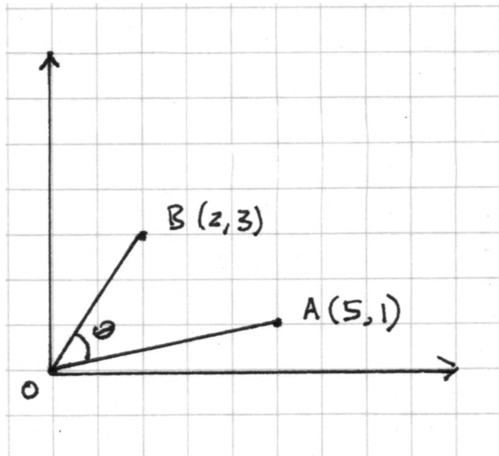

Figure 1: Dot product is cosine

The second way of multiplying two vectors is the *cross product*, which yields another vector:

$$\vec{p} \times \vec{q} = \vec{p}_x \vec{q}_y - \vec{p}_y \vec{q}_x$$

If you're paying attention, you noticed that I said the result was a vector but I gave a formula that yields a scalar. The cross product is really defined on 3D vectors; we're treating our two 2D vectors as 3D vectors with $z = 0$. The cross product of these two vectors will be of the form $[0, 0, z]$; by convention, when talking about the cross product in 2D, we just use its z element as the result.

Note that unlike the dot product, the cross product is not commutative; in fact $\vec{p} \times \vec{q} = -\vec{q} \times \vec{p}$.

The most important geometric interpretation of the cross product is that it is the signed area of the parallelogram having two vectors as adjacent sides. By extension, it is also twice the signed area of a triangle having the participating vectors as two sides. We say the area is signed because the cross product can be negative; the sign will indicate the order of the vertices. If the cross product is positive, then the vertices are given counter-clockwise; otherwise, they are given clockwise.

Example

What is the area of the triangle OAB in figure 2, and in what direction are its vertices wound?

We will take the two vectors $\vec{a} = A - O = [5, 0]$ and $\vec{b} = B - O = [0, 5]$. The area of the triangle is $\frac{1}{2} \vec{a} \times \vec{b} = \frac{1}{2} |5{\cdot}5 - 0{\cdot}0| = 12.5$. Since the cross product is positive, the triangle is counter-clockwise.

In this simple case, we can easily double-check our answer with the formula $area = \frac{1}{2}base{\cdot}height = \frac{1}{2}5{\cdot}5 = 12.5$.

If the three vertices of a triangle are colinear (all on the same line), then the triangle is called *degenerate* and its area will be zero. We can use this fact to create a test to see if three points are on the same line: simply treat the points as a triangle as above, compute the two side vectors, and take the cross product. If the cross product is zero, then the points are colinear.

Note that we can use the dot product for the same test. Then the dot product of the **normalized** side vectors will be 1 or -1 if the points are colinear. Also, if the dot product is zero, then the two vectors are perpendicular. The cross product test for colinearity, however, will be faster because it doesn't need the vectors to be normalized, which requires taking a square root.

There is one other very useful property of the cross product. The dot product allows us to compute the cosine of the angle between two vectors; the cross product allows us to compute the sine. If we have two vectors $\vec{p}$ and $\vec{q}$, and θ is the angle between them,

$$\vec{p} \times \vec{q} = |\vec{p}||\vec{q}| \sin\theta$$

We have to be a bit careful here due to the cross product not being commutative. If $\vec{p} \times \vec{q} < 0$, then we will end up with the angle "outside" the two vectors; i.e. $2\pi - \theta$ (look at the graph of sine to see why). We can avoid this by taking the absolute value of the cross product.

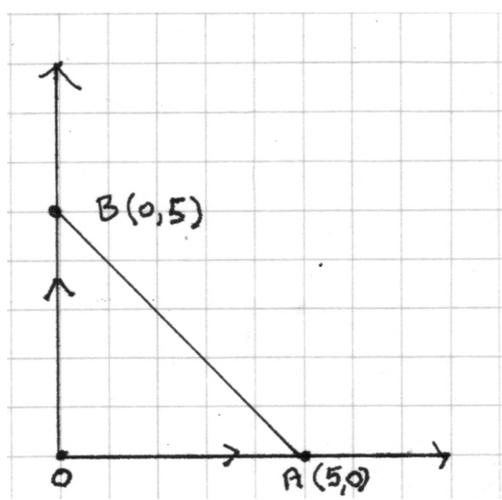

Figure 2: Cross product is area

Point-line distance

The problem of finding the shortest distance from a point to a line comes up fairly often. From a point P to a line L, the shortest distance is along another line, containing P, perpendicular to L. We can use this fact to set up a right triangle problem (see figure 3). From the diagram we can see that

$$\sin(\theta) = \frac{d}{|\vec{v}|}, d = |\vec{v}| \sin(\theta)$$

If we plug in $\vec{v} \times \vec{l} = |\vec{v}||\vec{l}| \sin\theta$, we get

$$d = \frac{\vec{v} \times \vec{l}}{|\vec{l}|}$$

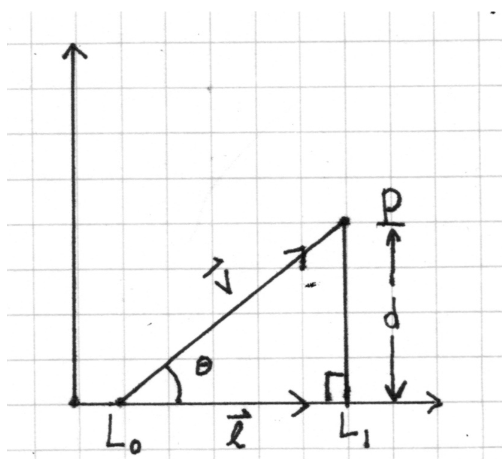

Figure 3: Point-line distance

Area of a polygon

We find the area of a polygon by decomposing the polygon into triangles, and then adding up the areas of all the triangles (see figure 4). The area is

$$a = \triangle OP_0P_1 + \triangle OP_1P_2 + \triangle OP_2P_3 + \triangle OP_3P_0$$

This may seem counterintuitive; after all, we're adding in the area of triangles outside the polygon! This formula depends on the concept of signed area aluded to earlier. The first two triangles are wound counterclockwise, so their signed area is positive; this includes the area inside the polygon plus the area between the polygon and the origin. The other two triangles are wound clockwise, so their area is negative, and they only cover the area outside the triangle and the origin. Effectively those triangles cancel out the extra area added with the first two triangles. It can be proved that this works for any polygon, even concave ones.

We've already seen the formula for the signed area of a triangle: half the cross product of the side vectors. All of our vectors are from the origin to points on the polygon. The final formula for an n-gon with vertices $(x_i, y_i), i \in 0..n-1$ is

$$a = \frac{1}{2}\sum_{i=0}^{n-1} x_i y_{(i+1)\%n} - x_{(i+1)\%n} y_i$$

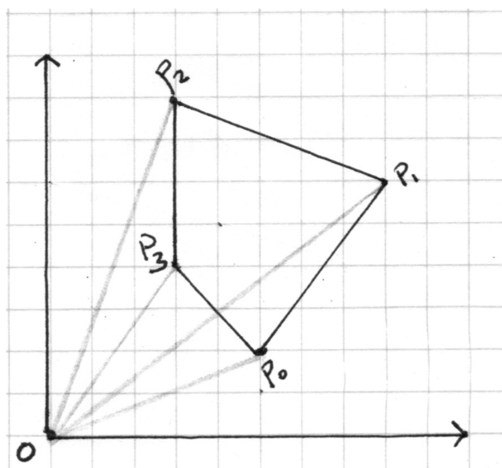

Figure 4: Area of a polygon

General notes

As noted above, computational geometry is rife with special cases. Some of the most common to watch out for are:

1. Horizontal and vertical lines. Many algorithms divide by Δx or Δy; horizontal or vertical lines will cause these quantities to be zero. The special case always has a geometric interpretation; figure out what it is and make sure your code responds appropriately.

2. Degenerate objects. Above, we discussed degenerate triangles as a test for colinearity. Degenerate triangles can also crop up in test data (unless the problem spec promises they won't.) Another case is a line segment for which both endpoints are the same.

```
void wrong(double a, double b) {
  double c = (a / b) * b - a;
  if (c == 0.0) {
    printf("yes!"\n);
  }
}

const double epsilon = 1e-9;
void right(double a, double b) {
  double c = (a / b) * b - a;
  if (fabs(c) < epsilon) {
    printf("yes!"\n);
  }
}
```

Figure 5: Floating point error

3. Sign of results. If we're careful, this can help, as in the polygon area algorithm. However, carelessness with signs will cause wrong answers. If an operation can result in a signed value, think about what it means.

4. There are always two angles between any two non-colinear vectors: one acute and one obtuse. Similar to signs, you should know you're getting the proper angle.

Another thing to watch out for is real numbers. Talking about imprecision and error accumulation in floating point operations could take an entire lecture; the important thing to remember is that floats and doubles are, in most cases, only approximations to the real values they represent. Figure 5 shows two ways of comparing against zero. In this simple example, the compiler will probably optimize away the entire computation of `c`, but in more complex code, `c` stands a good chance of being a very small non-zero value. We can't even predict if it will be positive or negative. The proper way of comparing against zero is shown in function `right`. The value of `epsilon` is dependent on the data, but 1×10^{-6} for floats and 1×10^{-9} for doubles are common. For testing the equality of two doubles, use the form `if (fabs(a-b) < epsilon) {...}`.

In some cases, a problem will specify that all coordinates are given as integers. With these problems it is often possible to sidestep the entire issue of floating point imprecision by sticking to integer math. We can keep intermediate results explicitly as fractions. If we have to add two fractions, we have to get them to the same denominator; however, to just compare we can cross multiply:

$$\frac{a}{b} = \frac{c}{d} \iff ad = bc$$

For testing a range (such as seeing if $0 \leq t \leq 1$ for a line segment), we can multiply the entire equation by the denominator:

$$0 \leq \frac{a}{b} \leq 1, b \geq 0 \iff 0 \leq a \leq b$$

Cramer's Rule

Cramer's rule is a formulaic method of solving n equations in n variables. It is based on the matrix concept of the *determinant*. The determinant of any 2 by 2 matrix is given as

$$det \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

Although we are only dealing with two dimensions here, the determinant is defined for any square matrix, and Cramer's rule can be used to solve systems of any size. If we have a linear system of two variables

$$a_1 x + b_1 y = c_1$$

$$a_2 x + b_2 y = c_2$$

We can write this in matrix form as

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

With a lot of algebra, the following solutions for x and y can be derived

$$x = \frac{det \begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix}}{det \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}}, y = \frac{det \begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}}{det \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}}$$

The denominator is the determinant of the matrix containing all of the coefficients. In each numerator, we replace the variable we're solving for's coefficients with the system's right hand side constants.

If the denominator is zero, then there is not a single solution to the system. There are either no solutions, or an infinite number of solutions.

Example

Assume we have these equations we would like to solve:

$$2x + 3y = 6$$

$$5x - 6y = -3$$

The three determinants of interest are

$$d_{den} = det \begin{vmatrix} 2 & 3 \\ 5 & -6 \end{vmatrix} = -27, d_x = det \begin{vmatrix} 6 & 3 \\ -3 & -6 \end{vmatrix} = -27, d_y = det \begin{vmatrix} 2 & 6 \\ 5 & -3 \end{vmatrix} = -36$$

and we have

$$x = \frac{d_x}{d_{den}} = \frac{-27}{-27} = 1, y = \frac{d_y}{d_{den}} = \frac{-36}{-27} = \frac{4}{3}$$

which we can verify as correct by substitution back into the original equations.