Code Jam 2012 - World Finals

Analysis: Twirling Towards Freedom

The Small Input

On most Google Code Jam problems, the small input can be solved without worrying about the running time. That was not the case here though. Even if N = 10 and M = 10, there are still 10^{10} different rotation patterns you could try. That is a lot!

There are various ways you could try to bring the running time down to a more manageable amount, but here is one fact that makes it easy:

- The 1st star you rotate around, the 5th star you rotate around, and the 9th star you rotate around should all be the same.
- The 2nd star you rotate around, the 6th star you rotate around, and the 10th star you rotate around should all be the same.
- The 3rd star you rotate around and the 7th star you rotate around should be the same.
- The 4th star you rotate around and the 8th star you rotate around should be the same.

If you realize this, there are only 10^4 possibilities to try. But why is this fact true? Read on and you'll find out!

Understanding Rotations

One of the biggest challenges here is just wrapping your head around the problem. How can you intuitively understand what it means to rotate something 100,000 times? The best way to do this is to roll up your sleeves and write down a couple formulas.

When dealing with rotations and translations of points on a plane, *complex numbers* provide an excellent notation:

- A point (x, y) can be represented as x + iy.
- Complex numbers can be added and subtracted just like vectors.
- Complex numbers can be rotated clockwise 90 degrees about the origin simply by multiplying them by -i.

It is this last property that makes them so clean to work with, and it is what we will use for the analysis. If you prefer, you could also imagine replacing everything with matrices.

We know that rotating a point P_0 by 90 degrees about the origin will send it to -i * P_0. However, what happens if we rotate it about a different point Q_0? There is a standard formula for this situation that you may have already seen. The resulting point P_1 satisfies the following:

$$P_1 - Q_0 = -i * (P_0 - Q_0)$$

This is our previous formula applied to the fact that rotating $P_0 - Q_0$ by 90 degrees about the origin must give you $P_1 - Q_0$. (Do you see why that is true? It's really just a coordinate change.) In our case, it will be helpful to group things a little differently in the formula:

$$P_1 = -iP_0 + Q_0 * (1 - i)$$

Now, suppose we rotate P_1 by 90 degrees about another point Q_1, then by 90 degrees about another point Q_2, and so on. What would happen to this formula? Let's write out a few examples:

- $P_2 = -iP_1 + Q_1 * (1 - i) = -P_0 + Q_1 * (1 - i) + Q_0 * (-1 - i)$.
- $P_3 = -iP_2 + Q_2 * (1 - i) = iP_0 + Q_2 * (1 - i) + Q_1 * (-1 - i) + Q_0 * (-1 + i)$.
- $P_4 = -iP_3 + Q_3 * (1 - i) = P_0 + Q_3 * (1 - i) + Q_2 * (-1 - i) + Q_1 * (-1 + i) + Q_0 * (1 + i)$.
- $P_5 = -iP_4 + Q_4 * (1 - i) = -iP_0 + (Q_4 + Q_0) * (1 - i) + Q_3 * (-1 - i) + Q_2 * (-1 + i) + Q_1 * (1 + i)$.
- etc.

From here, it's not too hard to guess the general formula. If the origin is rotated by 90 degrees about Q_0, then Q_1, and so on, all the way through Q_{m-1}, then the final resulting point P_m is given by:

$$(1 - i) * (Q_{m-1} + Q_{m-5} + Q_{m-9} + \dots) + (-1 - i) * (Q_{m-2} + Q_{m-6} + Q_{m-10} + \dots) + (-1 + i) * (Q_{m-3} + Q_{m-7} + Q_{m-11} + \dots) + (1 + i) * (Q_{m-4} + Q_{m-8} + Q_{m-12} + \dots)$$

And once the formula is written down, it's not too hard to check that it always works. So anyway, is this simpler than the original problem formulation? It may look pretty complicated, but for the most part, you're now just adding points together here, and addition is easier than rotation!

Pick a Direction

We want to choose stars $Q_0, Q_1, \dots, Q_{m-1}$ in such a way as to make the following point as far away from the origin as possible:

$$(1 - i) * (Q_{m-1} + Q_{m-5} + Q_{m-9} + \dots) + (-1 - i) * (Q_{m-2} + Q_{m-6} + Q_{m-10} + \dots) + (-1 + i) * (Q_{m-3} + Q_{m-7} + Q_{m-11} + \dots) + (1 + i) * (Q_{m-4} + Q_{m-8} + Q_{m-12} + \dots)$$

There are different ways of thinking about things from here, but the key is always convex hulls.

Let X be the point furthest from the origin that we can attain, and more generally, let X_v be the point that is furthest in the direction of v that we can attain. Certainly, $X = X_v$ for some v. (This is because X is surely the point that is furthest in the direction of X.) Therefore, it suffices to calculate X_v for all v, and we can then measure which of these is furthest from the origin to get our final answer.

So how do we calculate X_v for some given v? We want $(1 - i) * Q_{m-1}$ to be as far as possible in the v direction, or equivalently, we want Q_{m-1} to be the star that is furthest in the $(1 + i)*v$ direction. Of course, the same thing is true for Q_{m-5}, Q_{m-9}, etc. We should choose the same star for all of them. (This is the fact that we used for the small input solution!) Similarly, for Q_{m-2}, Q_{m-6}, etc., we want to choose the star is furthest in the $(-1 + i)*v$ direction. In general, we want to choose stars from the original set that are as far as possible in the following directions: $(1 + i)*v$, $(-1 + i)*v$, $(-1 - i)*v$, $(1 - i)*v$.

We are now almost done (at least conceptually). First we find the convex hull of the stars and solve for one particular v. Now what happens when we rotate v? For a while, nothing will change, but eventually, one of the four directions we are trying to optimize will be perpendicular to an edge of the convex hull, and as a result, the optimal star will switch to the next point on the convex hull. This simulation can be done in constant time at each step, and there will be only O(N) steps since each star choice will rotate once around the convex hull.

At this point, we are done! We have found every X_v, and we can manually check each of them to see which one is best. The implementation here is tricky, but it is an example of a general technique called rotating calipers. You can find plenty more examples of it online. Or you can always download solutions from our finalists!

Bonus Question!

When we generated test data for this problem, we were surprised to find that we could not come up with a case to break solutions which misread "clockwise" as "counter-clockwise" in the problem statement. In fact, there is no such case! Can you see why?