15295 Fall 2017 #11 -- Problem Discussion

November 8, 2017

This is where we collectively describe algorithms for these problems. To see the problem statements follow this link. To see the scoreboard, go to this page and select this contest.

A. Candy division

B. Triangle in a Triangle

You can show that the only points you need to consider for the corners of the biggest triangle are the (up to) six points that are extreme on the three sides of the triangle. This is just a consequence of the fact that only points on the convex hull need be considered to find the largest area triangle among a set of points. So you find these six points and try all subsets of three and take the largest triangle that can be made. --DS

One way to see why only the extreme points on each side matter: say you fix two points of your triangle. What's the best point to pick for your third point in order to maximize area? Consider the line segment connecting your two fixed points, and view this as the base of your triangle. Recalling that area = base * height / 2, you want your third point to maximize the height of your triangle, that is, to maximize the perpendicular distance away from the base. Draw some pictures and convince yourself that such a point must be the extreme point of a line. Now you know that at least one point of the triangle should be taken to be an extreme point. Repeat this argument to see that all points of the triangle should be extreme points. -- Tom

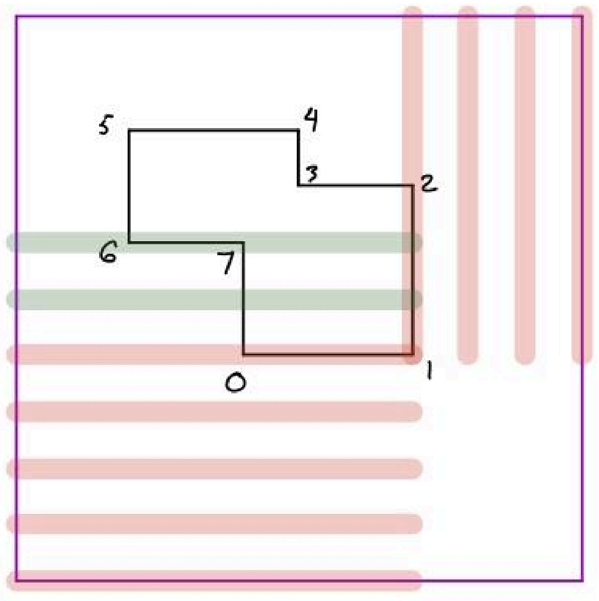
C. Chess

D. Effective network

E. Compass

Extend each horizontal segment which occurs in the input polygon in both directions from 0 to 1000. Do the same thing for all the vertical segments. Now you have N^2 intersections among these. If there's a solution there is one at one of these intersections. This leads to an algorithm.

For each such vertical and horizontal segment we're going to maintain a range along it which might still be a valid solution. Initially these ranges are [0, 1000]. We walk around the polygon processing each edge and each external corner, as shown in the figure below. For each one we intersect the valid regions with new intervals, thus shrinking the valid regions to a possibly smaller region.



The figure illustrates the two types of restrictions that are generated. We're walking around the polygon in counter clockwise order. When we process edge (1,2) there are two integral (y coordinate) rows shown in green for which this edge implies a restriction on where the magnet could be. Namely the magnet must be (if it is in either of these rows) in the green part of the rows. For the corner 1 red lines illustrate regions on those 4 vertical columns and 5 horizontal rows which are get additional restrictions.

When we process segment (5,6) the restriction is to the right instead of the left. And (6,7) is up from that edge, etc. So there are four functions for edges and four for corners. And I wrote them all out instead of doing it abstractly and elegantly.

There are a total of 2N columns and rows. Each edge can touch N of them and each external corner can touch 2N of them. So the total work to apply these restrictions is $O(N^2)$.

Finally after we've computed the valid intervals in each column and each row, we need to determine if there is anything valid left. To do this we check each column and each row and see if their intersection is still valid in that column and row. If such a point is found we have a solution, otherwise there is none. This process is also $O(N^2)$. Since $N \le 1000$ this is fast enough. --DS

F. Affine

An affine transformation is equivalent to doing a linear transformation (i.e. multiplying by a matrix) followed by a translation.

The hypercube is convex, so an affine map of it must be convex too. It's also symmetric about its center point, so the affine map of it is too. Thus if it's a non-degenerate polygon, it must be convex, and have an even number of sides and opposite sides are parallel and the same length. And in this case the dimension of the hypercube is n/2. So this is all easy to test using standard primitives like line-side-test.

But it turns out they wanted to be evil. So the polygon could be a point (hypercube dimension 0), a line segment (hypercube dimension 1).   The latter, strictly speaking, violates the requirement that the polygon boundary not intersect itself. At least this case is in the sample data.

And it turns out that they made the problem harder by dividing the sides up into several parallel pieces, just to be gratuitously annoying. Of course they also included the 1 point case (test case number 117). --DS