Code Jam 2019 - World Finals

Analysis: Juggle Struggle: Part 1

Test Set 1 (Visible)

We are given a set of 2**N** points that we are to pair into **N** line segments such that they all intersect each other. That makes the set of line segments magnificent.

We can extend any line segment S into a full line L that divides the plane in two. In a magnificent set of line segments, since S intersects all other segments, all those other segments have one endpoint on each side of L (no other point can be on L or it would be collinear with the endpoints of S). This means that if we pick a point P in the input, it must be paired with another point Q such that the line that goes through both leaves exactly the same number of other points on each side. This hints at an algorithm: choose a point P, find a Q according to the definition above, pair P and Q, and repeat. However, what if there is more than one such Q?

One way to deal with the problem is by choosing P intelligently. For example, if we choose a leftmost point as P, all candidates for Q end up in the right half of the plane cut by a vertical line through P (with at most one other point possibly on the line). Consider a line rotating clockwise centered on P, going from vertical to vertical again (rotating 180 degrees). At the starting point, there are no points on the left side of the line, because P is leftmost. As it rotates, it will pass over all points, repeatedly adding one point to one side and removing one from the other. Since there are no collinear triples of points, this shift is always by 1. This means there is exactly one such line that has P and one more point on it, with **N**-1 points on each side. Moreover, if we sort all non-P points by the angle they form with P (choosing 0 to coincide with the vertical line), the point that ends up on the line is exactly the median of the sorted list.

Choosing P as a leftmost point gives a unique choice of another point Q to pair P with. We can then remove both and continue. The algorithm requires **N** iterations, and in each of them we must find a leftmost point, and sort O(**N**) points by angle. Calculating the cosine of the angles to compare without loss precision takes constant time. Picking the median and removing the pair of points all can be done in O(**N**) time or better. Therefore, the overall algorithm requires O($\mathbf{N}^2$ log **N**) time, and it is enough to solve Test Set 1. If one uses a linear algorithm to find the median instead of sorting, it would improve the time complexity to O($\mathbf{N}^2$). However, sorting to find the median is likely to be faster in practice due to constant factors.

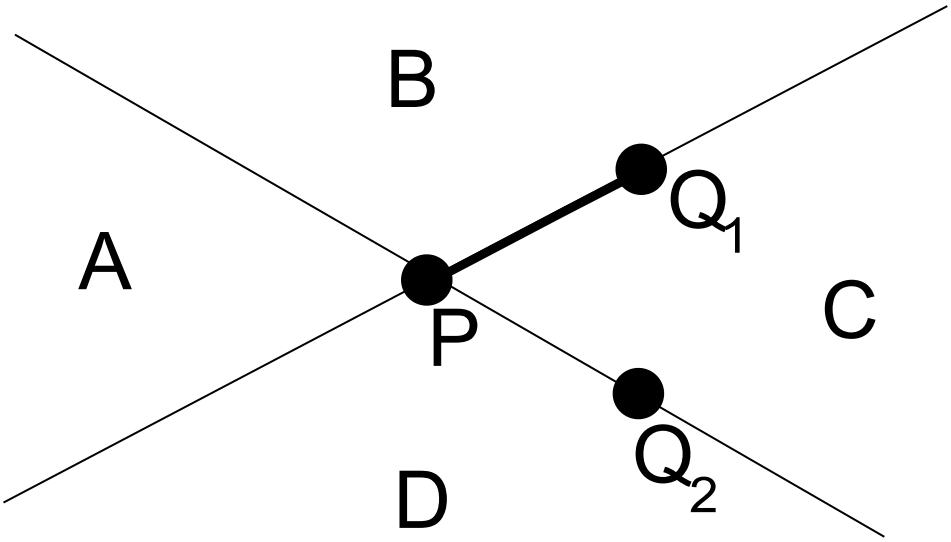
A consequence of the reasoning above is that the solution is actually unique. Another observation is that any point in the convex hull of the points has the same property as a leftmost point, because the definition is invariant through rotations and any point in the convex hull can be made a leftmost point through a rotation. But a leftmost is one of the easiest to find.

Test Set 2 (Hidden)

There is another approach requiring an additional proof but leading to a slightly simpler algorithm for Test Set 1. Most importantly, it leads us one step closer to solving Test Set 2.

The additional key observation for this approach is that if a set of 2**N** points admits a magnificent pairing, then for *every* point P in the set (not just those in the convex hull) there is exactly one Q such that the line through P and Q leaves half the points on each side. The fact that there is at least one is immediate from the existence of a magnificent arrangement. The argument that there cannot be more than one requires more thought.

Assume there is a point P and two other points Q_1 and Q_2 such that both of the lines that go through P and each Q_i have half of all the other points on each side. Moreover, without loss of generality, assume Q_1 is the one that actually pairs with P in the magnificent pairing (we know it is unique). The following picture illustrates the situation.



In the picture, we have labeled the four areas into which the two lines divide the plane. Let A, B, C and D be the sets of input points contained in each area (not including P, Q_1 or Q_2). Notice that any point in A must be paired with a point in C in order for the produced segment to intersect PQ_1. Similarly, any point in D must be paired with a point in B. All points below the line that goes through P and Q_2 are in either A or D, which means there are $|A| + |D|$ of them. The number of points above, on the other hand, is $|B| + |C| + 1$. Since we showed $|A| \leq |C|$ and $|B| \leq |D|$, $|B| + |C| + 1 \geq |A| + |D| + 1 > |A| + |D|$. This contradicts the assumption that the line that goes through P and Q_2 has the same number of input points on each side.

This observation by itself only allows us to avoid the "find a leftmost point" step of the previous algorithm, which does not change its overall time complexity. The definite improvement is to more quickly shrink the set of points that we must consider, so that all steps within the iteration take less time. To do that, consider that after we have paired M pairs of points, the lines through each pair divide the plane into 2^M areas, with only the outside areas (the ones with unbounded area) possibly containing leftover points. Moreover, each point is to be paired with one in the area that is opposite in order to cross all the lines, which is a necessary condition to intersect all line segments. When we create a new pair, exactly two of the areas are split in half. The idea is then: instead of removing the new pair and continuing with the full set, continue recursively with two separate sets. If we consider sets X and Y coming from opposite areas, split after pairing into X_1 and X_2 and Y_1 and Y_2, respectively, then we need to solve recursively for X_1 and Y_1 separately from solving for X_2 and Y_2 (assuming the areas are labeled such that X_1, X_2, Y_1, Y_2 appear in that order when going through them in clockwise order).

The last thing to do is to make sure we split X and Y more or less evenly with the new line. Notice that if we restrict ourselves to start with a point P that is part of the convex hull of X or Y or $X \cup Y$, this might be impossible (i.e., all those pairs may split X and Y, leaving most remaining points on one side). Hence the need for the property with which we started this section. However, notice that if we sort all pairs to be made between points in X and points in Y by the slope of their produced segments, the first and last will split X and Y into an empty set and a set with $|X| - 1$ points (notice that $|X| = |Y|$). The second and next-to-last will split them into a set with 1 point and a set with $|X| - 2$ points. The i-th will split them into a set with $i - 1$ points and a set with $|X| - i$ points. Therefore, if we choose randomly, we end up with a recursion similar to quicksort's, in which the reduction on the size of the sets is expected to be close enough to always partitioning evenly. Since the time it takes for the non-recursive part of a set of size M is

O(M log M) — notice that calculating the partitions is only an additional linear time step — the overall expected time complexity of this recursive algorithm is ($\mathbf{N}$ $\log^2$ $\mathbf{N}$).