EGOI 2025 Editorial - A String Problem

Task Author: Yoav Linhart

July 18, 2025

Introduction

In this problem, we are given $2N$ points, evenly distributed on a circle. There are N straight lines that each connect a unique pair of points. In each step, one endpoint of one line can be changed to another endpoint. The goal is to output the shortest sequence of steps so that, afterward, all lines are parallel to each other, and each endpoint is connected to exactly one line.

Line Notation

From now on, we will also refer to the line that connects the points i and j as the line (i, j).

Subtask 1: Line i is attached to the points i and $i + 1$

In this subtask, every line connects points that are next to each other on the circle. If N is odd, then initially no two lines are parallel, and we have to change all lines except for one (which we can choose arbitrarily). Therefore, the minimum number of steps is $N - 1$. If N is even, then for each line, there is exactly one other line that is parallel to it. Therefore, we have to change all lines except for two, which takes at least $N - 2$ steps.

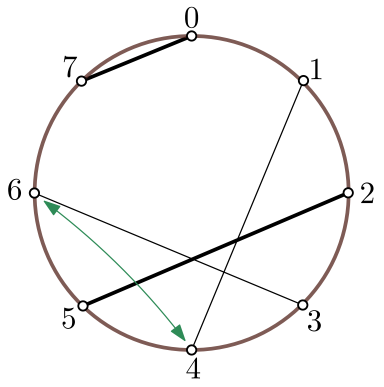
We can construct a sequence of steps that uses this number of moves (and thus is optimal). We can fix an arbitrary line that we do not change, and make all other lines parallel to it. We choose this line to be the line $(0, 1)$. Therefore, for all $2 \leq i \leq 2N - 1$, the point i should be connected to the point $2N - i + 1$. Now we can look at pairs of lines that we need to fix. There are $\lfloor \frac{N-1}{2} \rfloor$ such pairs, and the ith pair ($1 \leq i \leq \lfloor \frac{N-1}{2} \rfloor$) contains the lines $(2i, 2i + 1)$ and $(2N - 2i, 2N - 2i + 1)$. We should change this pair of lines to the pair $(2i, 2N - 2i + 1)$ and $(2i + 1, 2N - 2i)$. It is easy to do this using two steps. So in total, we use $2 \cdot \lfloor \frac{N-1}{2} \rfloor$ steps, which is optimal.

Distinguishing between rotations

We can check if two lines are parallel by considering their endpoints. In particular, a line (a, b) and a line (c, d) are parallel if and only if $a + b \equiv c + d \bmod (2N)$. In other words, the expression $a + b \bmod (2N)$ specifies the *rotation* of the line. We have $2N$ possible line rotations overall, and we want all lines to have the same rotation.

Subtask 2: At most two steps are needed

First, we observe that it is impossible that a single line is not parallel to the rest. Thus, either all lines are parallel to each other (in which case the number of steps is 0) or all lines are parallel to each other except for two (in which case the answer is 2). We can distinguish the cases by computing the rotation of every line, and by counting the number of times each rotation appears. In the case that two lines are not parallel to the rest, these two lines can be fixed by swapping an endpoint of one line with an endpoint of the other line. This can be seen in the figure below.



Since there are only four possibilities for this, we can simply try all of them and choose the one that makes all lines have the same rotation.

Note that similarly to subtask 1, the minimum number of steps is the number of lines (N), minus the number of times the most common rotation appears. We want to make this work for the general case.

Constructing a solution given a fixed rotation

Given a fixed rotation, we say that a line is *good* if its rotation is the fixed one, otherwise we call it *bad*. To make all lines parallel to the fixed rotation, just changing an arbitrary endpoint of a bad line to make it good does not suffice; we have to be careful that no two lines share an endpoint after all changes are done.

We can avoid this by doing as follows: We pick a bad line ℓ_1, and change one of its endpoints to make it good. This new endpoint is occupied by another line ℓ_2 that is currently bad. Again, we change this endpoint (occupied by the two lines ℓ_1 and ℓ_2) such that line ℓ_2 becomes good and does not share endpoints with ℓ_1. We repeat this process until the new endpoint is not occupied by any other line. After that, we pick a new bad line and start the process again. We do this until all lines are good. This can be done either iteratively or recursively (similar to a DFS), and takes $O(N)$ time. The number of steps we used is N minus the number of lines that initially had the fixed rotation.

Why does the construction always work?

While playing with examples might convince us that the previous construction always works (and this is enough to solve the problem), it is not trivial to understand why. In particular, the unclear part is why we will never reach a new endpoint that is occupied by another *good* line (which would cause us to get stuck).

One way to prove this is using graph theory. Suppose that we already chose the fixed rotation of the parallel lines. Let's model the problem with a bipartite graph – one side contains N nodes that represent the lines in the initial configuration, and the other side contains N nodes that represent the good lines in the final configuration. For each line ℓ in the initial configuration, we add two edges from the node that represents it to the two nodes that represent the good lines to which ℓ can become after changing one of its endpoints. Note that the two edges may connect the same pair of nodes, and this happens if and only if ℓ is initially good. So, we have a bipartite graph with N nodes in each side, and the degree of each node is 2. Our goal is to find a perfect matching in the graph.

We know that any graph that contains only nodes of degree 2 is a collection of disjoint cycles. Since the graph is bipartite, we can infer that all cycles are of an even length!

Now it becomes clear why the construction always works – each disjoint cycle consists of an even number of edges, so if we take every other edge of a cycle, each node in it appears exactly once. Thus, doing so for each cycle results in a perfect matching. Since each edge denotes an initial line and the final line it should become (using one step), considering only the edges in the perfect matching we found results in a correct optimal solution. Note that we have one special case – we do not change anything for cycles of length 2, as they correspond to a line that is initially good.

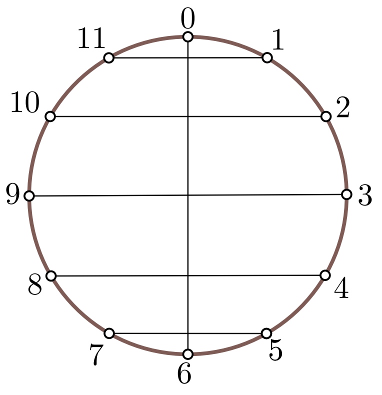
Subtask 3: It is guaranteed that there is a solution where a line connects the points 0 and 1

Since we know that every line should be parallel to the line that connects 0 and 1, we know our fixed rotation and we can directly apply the previous construction to find an optimal sequence of steps in $O(N)$ time.

Subtask 4: $N \leq 1000$

In this subtask, the desired final rotation is not known. We can iterate over all possible rotations and apply the previous algorithm, and take the shortest sequence we found. Since there are $2N$ rotations and an iteration takes $O(N)$ time, this algorithm takes $O(N^2)$ time.

However, there is something we haven't considered: not all rotations are possible – for some rotations, it is impossible to make all lines parallel to that rotation no matter the number of steps. In particular, this is the case if and only if the value of the rotation is even (recall that the value of the rotation of a line (a, b) is $a + b \bmod (2N)$). This is because a line with an even rotation means that the number of points between its two endpoints is odd. Such an example can be seen in the figure below, where it is impossible to make all lines horizontal (or vertical).



So we only check the N odd rotations, which gives us a correct algorithm with a running time of $O(N^2)$.

Full solution

For the full solution, we have to find the optimal rotation without simulating the construction every time. We have already observed that we change each line at most once to become good, and we only change the lines that are initially bad. So, for each fixed rotation, the number of steps is simply the number of bad lines in the initial configuration.

To minimize this, we need to find the largest group of lines that initially have the same rotation. This can be done by counting the number of lines of each rotation and choosing the most common one (the rotation values are from 0 to $2N - 1$ so we can simply use an array for this). Again, we have to be careful to consider just the odd rotations. If there is no line with an odd rotation, then all lines need to be changed (to an arbitrary rotation). Together with the construction part from subtask 3, we get a solution in $O(N)$ time.