However, there was an additional mistake that could be made. If $n \leq 2\ell - 2$, only overlaps $t \geq 2$ can come into play. This means that for such small values of n, the strings X and Y are actually equi-probable. In general, we need to ignore any overlap values that are smaller than $2\ell - n$ when constructing the overlap sequence.

Overlap sequences can be constructed in $O(\ell)$ time using KMP or hashing, leading to an $O(\ell s \log s)$ time algorithm.

Problem L: Visual Python++

Shortest judge solution: 1776 bytes. Shortest team solution (during contest): 1727 bytes.
Python solutions by the judges: only Pypy

This problem can be solved by a sweepline algorithm. Let us sweep from left to right, keeping a set of encountered but unmatched top left corners, ordered by r-coordinate. When we encounter a new corner:

- If it is an upper left corner, add it to the set of unmatched corners. If there was already a corner in the set with the same r-coordinate, we have encountered a syntax error – it will never be possible to create non-nesting matchings for these two top left corners.
- If it is a lower right corner with r-coordinate r_2, match it with the top left corner in our set with the largest r-coordinate r_1 that is $\leq r_2$. If there are no such top left corners in our set, we have encountered a syntax error.

Each event can be processed in $O(\log n)$ time so we have a time complexity of $O(n \log n)$.

However, we are not yet done. Even if this process finishes, the constructed rectangles may intersect. To check for this, we run essentially the same sweep again, but this time, since we know exactly how the rectangles look, we also check when adding and removing top left corners from our active set that adjacent pairs of rectangles in this set do not intersect.

Note that the solution, if it exists, is actually unique, but like with the Replicate problem, figuring this out is part of solving the problem.