MAJSTOR

We use the algorithm given in the problem statement to calculate Sven's score. For each round, compare Sven's symbol with the other symbols and add the appropriate number of points to the score.

The largest number of points Sven could have scored can be calculated by trying all three symbols in every round and using the one that scores best.

This is easily done if we write a helper function points(r, symbol) that takes two arguments, the round number s and the symbol Sven supposedly shows in that round. If A and B are the two numbers we want to calculate for a round (the actual score and the largest possible score), we can calculate them as:

A = A + points(r, Sven[r])

B = B + max{points(r, 'S'), points(r, 'P'), points(r, 'K')}

NIZOVI

Let fuzziness(S, E) be the fuzziness obtained by removing the first S and last E numbers from both sequences. A solution that for each pair (S, E) naively calculates fuzziness(S, E) is of complexity $O(N^3)$ which is too slow for large N and scores 30 points.

Note that fuzziness(S, E) can always be expressed as:

fuzziness(S, E) = A(S)·B(N−E) + B(S)·A(N−E) + fuzziness(S+1, E+1)

In other words, fuzziness can be calculated starting from the middle of the sequences and extending the interval of elements included in both directions. The complexity of this solution is $O(N^2)$ since there are 2·N−1 ways to choose the midpoint of the intervals and the most we can extend an interval from any starting point is N/2 times.

TABLICA

Directly implementing the algorithm given in the problem statement gives a solution of time complexity $O(K \cdot N^2)$ and space complexity $O(N^2)$, scoring 30 points.

A better solution exploits the fact that we don't need to know the positions of all numbers, just the K numbers that appear as X in the moves. We go about simulating the moves as before, but only move numbers whose positions will be significant sometime later. The time complexity of this solution is $O(K^2)$ and the space complexity O(K).

CVJETICI

Simulating the growth of plants by maintaining a two-dimensional table is of time and space complexity $O(N^2 + N \cdot M)$, where M is the maximum coordinate. This solution scores 30 points.

Let A(x) be the number of plants for which a flower may grow at coordinate x (but hasn't yet). The sequence A initially contains all zeros and changes when plant (L, R) grows:

- Flowers will grow at coordinates L and R, a total of A(L) + A(R) of them, so we output that number. After these flowers have grown, there are no more available plants at coordinates L and R, so we reset A(L) and A(R) to zero.
- The horizontal segment [L+1, R−1] is now available to grow flowers, so we increase A(x) by one for each of those coordinates.

Directly implementing the above algorithm yields a solution of complexity $O(N \cdot M)$, scoring 45 points.

For full score, we need a data structure that supports the following operations:

- Set A(x) = 0;
- Increase A(x) by one for each x in [L-1, R+1].

Some of the data structures that do this are interval trees and Fenwick trees for O(log N) per query, or splitting the sequence A into blocks of size about sqrt(N), for O(sqrt(N)) per query.

The official source code splits the sequence A into blocks of size 256. See task JAGODA (COCI 2009, round 5) for details on this structure.