Solution to task F/2006

Author: Wojciech Kazana

DESCRIPTION AND SOLUTION

Our solution consists of checking a few (actually all of them) possibilities amongst which the solution must exist.

Reading the task we find out that the optimal solution has to consist of two sequences of gears which share a common prefix (it's length can be equal to zero) and then they HAVE NO MORE COMMON GEARS. So our sequences can be described as AX and AY, where X and Y are different and every element of A, X and Y is a name of one gear or a concatenation of two such names. First step of our algorithm is to compute every possible way of describing N (the number of gears - up to 6) as a sum of numbers 1 and 2. From now on let's name such a sum a pattern (for example if N is 6 then 1+2+2+1 is a pattern). We can write it as A'X'Y'Z' where A', X', Y', Z' describe the patterns of A,X,Y,Z (Z are the gears that weren't used and A,X and Y are as shown in the beginning). Now for every permutation π of gears and for every pattern A'X'Y'Z' we claim that π is described as AXYZ. We consider all patterns and every single permutation, so the optimal solution (of course if it exists) must be somewhere amongst them.

For every case we first check whether AX actually describes the minute hand (that the combination of gears in AX allows us to obtain 24 turns a day in a clockwise direction) and AY the hour hand (2 turns a day). If not - we move to another case. If it does, we check if this particular solution is better than the best one so far and if it is, we swap the previously best solution with this one.

In the end we just print the solution onto the screen in a way described in the task. In particular there can be three gears on the last common shaft and there is no mistake in the procedure shown above - it just means that the last element of A, together with the first element of X and the first element of Y, are all situated on the same shaft.

COMPLEXITY ANALISYS

The limitations in the task are very little. Actually ($N \leq 6$) so the number of cases checked is also very small, even though we check not only every permutation and pattern, but also every splitting of a particular pattern into A,X,Y and Z described above. Counting it very roughly we get: at most 6! permutations, only 13 patterns in the worst case of 6 gears, at most $\binom{6}{3} = 20$ splits of a pattern (of length 6), so we have only 187200 cases to check. Each of them is being checked linearly in time of N (we have to count the speeds of every shaft in a pattern).

SOME TECHNICAL STUFF

In our implementation we use two different functions to check the cases:
when $Y = \epsilon$ we use function `probuj(vector<string>)`
and if it is not we use function `probuj(vector<string>, vector<string>, int)`.

To generate every permutation in C++ we use library function next_permutation. In Java we had to implement it ourselves.