Code Jam 2021 - Round 1A

Analysis: Append Sort

Test Set 1

First of all, note that it is never optimal to append anything to the first number. Now, with the low limits we can iterate through all the ways to append digits to the second and third numbers while achieving the problem goal. We should stop once we understand that we cannot do better, for example, when both numbers become longer than 4 digits.

Test Set 2

We can use a greedy approach to solve the problem: for every number starting with the second one, make it larger than the previous one but as small as possible. Note that this ensures we use as few digits as possible for the current number and at the same time makes it as easy as possible for the next number. This means the greedy choice is optimal.

So the problem now is how do we compute this efficiently. More formally: given integers A and B, find the minimum B' such that $B' > A$ and B is a prefix of B'. Let us denote the number of digits in an integer Z as $\text{len}(Z)$.

First of all, if $A < B$, we can set $B' = B$.

Now let's consider the case when $A \geq B$ and $\text{len}(A) = \text{len}(B)$. Note that appending any digit to B will make it larger than A. To make it as small as possible we can just append a zero.

The only case that is not considered now is when B has fewer digits than A. Let $k = \text{len}(A) - \text{len}(B)$.

First, we can try making B' equal to $B \times 10^k$, that is, append k zeroes to the right of B. If such a B' is larger than A, then this is the optimal solution.

Now, we check if it is possible that B' is the same length as A. If appending k nines to B doesn't result in something larger than A, then B' will need to have more digits than A. In such a case, we can just make B exactly 1 digit longer A and as small as possible by appending $k + 1$ zeroes to B.

If appending k zeroes to B is too small but appending k nines makes it larger than A, then B is actually a prefix of A. In such a case $B' = A + 1$ is the optimal answer.

All the checks performed above are linear in the length of A and B and each number in the input is processed at most once as A and at most once as B. However, numbers may become longer after each operation, but only by 1 digit. Therefore, the complexity of the overall algorithm is quadratic in the total number of digits in the input, and linear in the number of digits of the output. As an example, an input of $\mathbf{N}$ strictly decreasing integers of the same length yields an output where the i-th integer consists of exactly one more digit than the previous, which needs $(\mathbf{N} \cdot (\mathbf{N} - 1))/2$ operations in total.