Kick Start 2022 - Round A

Analysis: Challenge Nine

For simplicity, let us use L to represent the number of digits in $\mathbf{N}$, note the scale of L is actually $O(log_{10}\mathbf{N})$. And let us use $\{a_{L-1}, a_{L-2}, \ldots, a_1, a_0\}$ to represent the digits of $\mathbf{N}$, where a_{L-1} is the most significant digit and a_0 is the least significant digit ($0 \leq a_i \leq 9$ for all $i < L-1$, and $1 \leq a_{L-1} \leq 9$). We can read $\mathbf{N}$ as a string, where the characters in the string $\{s[0], s[1], \ldots, s[L-2], s[L-1]\}$ represent $\{a_{L-1}, a_{L-2}, \ldots, a_1, a_0\}$. Or, we can read $\mathbf{N}$ as an integer, whose value equals $a_{L-1} \times 10^{L-1} + a_{L-2} \times 10^{L-2} + \cdots + a_1 \times 10 + a_0$. Inserting a new digit d in the $k-$th position in $\mathbf{N}$ means inserting it after the first $L-k$ digits (on the left side) and before the last k digits (on the right side) of $\mathbf{N}$. If $\mathbf{N}$ is string type, then the new number can be built by slicing and concatenating the string as $s[0 \ldots L-k] + string(d) + s[L-k \ldots L]$ in linear time, where $s[i \ldots j]$ means characters of string s from index i to index j. On the other hand, if $\mathbf{N}$ is integer type, then the new number can be calculated as $(\mathbf{N} - \mathbf{N}\%10^k) \times 10 + d \times 10^k + (\mathbf{N}\%10^k)$ in constant time.

Test Set 1

Since $\mathbf{N} \leq 10^5$, the number of digits of $\mathbf{N}$ is at most 6, which means there are at most 7 positions to insert a new digit. As for the new digit, there are at most 10 options ($0 \ldots 9$), therefore the total number of results is at most 70. We can enumerate all these 70 results, eliminate those ones which are not multiples of 9 or have leading zeros, then find the smallest one from them.

In this solution, there are $O(L)$ positions to insert a new digit, and a constant number of choices for the new digit ($0 \ldots 9$). If $\mathbf{N}$ is read as an integer, then we need constant time to insert a new digit, decide if the inserted result is a multiple of 9 and compare the candidates, so the time complexity of this solution is $O(L)$. If $\mathbf{N}$ is read as a string, then every operation such as the insertion of a new digit or value comparison among candidates will take $O(L)$ time, so the time complexity will be $O(L^2)$.

Test Set 2

Now that $\mathbf{N}$ is at most 10^{123456}, we cannot read $\mathbf{N}$ as 32-bit/64-bit integer. Instead, we should read $\mathbf{N}$ as string where each character represents a digit of $\mathbf{N}$. Since L is at most $123456 + 1$, the brute force enumeration method with time complexity $O(L^2)$ is unacceptable. We need a more efficient algorithm.

First, let us decide which digit to insert. In fact, a number is a multiple of 9 if and only if the sum of its all digits is a multiple of 9. Therefore, we can add up all the L digits of the given $\mathbf{N}$, and use $9 - (sum \ mod \ 9)$ to get the new digit we want. Wherever we insert it, the sum of all digits of the new number and the new number itself will be a multiple of 9. Note that when $sum \ mod \ 9 = 0$, adding either a new 0 or a new 9 can make the result be a multiple of 9, but 0 is always more preferable than 9 because we are looking for the smallest answer.

Secondly, let us decide where to insert the new digit. Let us use d to represent the new digit we are going to insert. We can start from the most significant digit a_{L-1}, then a_{L-2}, and then use this order to visit all the digits in $\mathbf{N}$, to find the first digit in $\mathbf{N}$ that is larger than d. Then we should insert d right before this digit. In other words, say a_k is the first digit in $\mathbf{N}$ that is larger than d (i.e. $a_i \leq d$ for all $k+1 \leq i \leq L-1$), then the new number we are going to make is $\{a_{L-1}, a_{L-2}, \ldots, a_{k+1}, d, a_k, \ldots, a_0\}$, whose value equals to $N_g = a_{L-1} \times 10^L + a_{L-2} \times 10^{L-1} + \cdots + a_{k+1} \times 10^{k+2} + d \times 10^{k+1} + a_k \times 10^k + \cdots + a_0$.

Is this always the best choice? Yes, let us prove it. If we insert d to a more significant position, i.e. between a_{q+1} and a_q where $q > k$, then the new number we are going to make is $\{a_{L-1}, a_{L-2}, \ldots, a_{q+1}, d, a_q, \ldots, a_k, \ldots, a_0\}$ with value $N_o = a_{L-1} \times 10^L + a_{L-2} \times 10^{L-1} + \cdots + a_{q+1} \times 10^{q+2} + d \times 10^{q+1} + a_q \times 10^q + \cdots a_k \times 10^k + \cdots + a_0$. The difference between this solution and the solution we claimed is $N_o - N_g = d \times 10^{q+1} - 9 \times (a_q \times 10^q + a_{q-1} \times 10^{q-1} + \cdots + a_{k+2} \times 10^{k+2} + a_{k+1} \times 10^{k+1}) - d \times 10^{k+1}$. Since we have that a_q is always no greater than d, we can make sure that $9 \times (a_q \times 10^q + a_{q-1} \times 10^{q-1} + \cdots + a_{k+2} \times 10^{k+2} + a_{k+1} \times 10^{k+1}) + d \times 10^{k+1}$ is always no greater than $d \times 10^{q+1}$. Therefore, no matter what other position q we choose to insert d, the result is always no less than inserting at k.

On the other hand, if we insert d to a less significant position, i.e. between a_{p+1} and a_p where $p < k$, then the new number we are going to make is $\{a_{L-1}, a_{L-2}, \ldots, a_k, \ldots, a_{p+1}, d, a_p, \ldots, a_0\}$ with value $N_o = a_{L-1} \times 10^L + a_{L-2} \times 10^{L-1} + \cdots + a_k \times 10^{k+1} + \cdots + a_{p+1} \times 10^{p+2} + d \times 10^{p+1} + a_p \times 10^p + \cdots + a_0$. The difference between this solution and the solution we claimed is $N_o - N_g = -d \times 10^{k+1} + 9 \times (a_k \times 10^k + \cdots + a_{p+1} \times 10^{p+1}) + d \times 10^{p+1}$. Since we have that $a_k > d$, we can make sure that $9 \times (a_k \times 10^k + \cdots + a_{p+1} \times 10^{p+1}) + d \times 10^{p+1}$ is always greater than $d \times 10^{k+1}$. Therefore, no matter what other position p we choose to insert d, the result is always greater than inserting at k.

Note that there are some edge cases. For example, all digits in $\mathbf{N}$ are less than d. In this case, we should append d to the end of $\mathbf{N}$. And if d is 0, we will find that the first digit in $\mathbf{N}$ that is larger than d is the leading digit. Be careful that we should not put d before it because the result cannot have leading zeros. In this case, we should put 0 right after the leading digit of $\mathbf{N}$ to get the smallest result.

The time complexity for finding out which d to insert is $O(L)$ because we need to add up all digits of $\mathbf{N}$ to get the remainder of $\mathbf{N}$ divided by 9. Then to find where to insert d, we also need $O(L)$ time to visit all the digits to find the first digit that is larger than d (or find out all the digits are no larger than d). Therefore, the total time complexity of this solution is $O(L + L) = O(L)$.