CodeForces Educational Round 178 E

omeganot

August 2, 2025

As is typical of query problems, we will try to solve it for only one query. Intuitively, if we would like to know the minimum number of letters we must append before t is no longer a subsequence of s, we must first check if t is a subsequence of s. Furthermore, if t is a subsequence of s, we would like to know the minimum index i such that t is a subsequence of the substring of s from 1 to i.

i is helpful because, if t is a subsequence of s, then the minimum number of letters we must add to t is exactly the length of the smallest subsequence which is not present in the substring of s from $i + 1$ to n.

So, how do we find i, if it exists? Well, we would start at the first index of string s, and then find the nearest occurrence of letter t_1. Then, we would find the nearest occurrence of t_2 after that index. This would take $O(n)$ to compute as we would just iterate over s, and once we find the next character of t, start searching for the one after.

However, we could optimize it to $O(|t|)$ by precomputing the next occurrence of each of the k letters for each index in string s. Basically, we would compute $nxt_{i,j}$ for all $1 \leq i \leq n, 1 \leq j \leq k$, where $nxt_{i,j}$ is the smallest index x such that $s_x = j$ and $x \geq i$. Then, we only have to iterate over t and keep calculating the next value.

To compute $nxt_{i,j}$, we can iterate from the end of string s and set $nxt_i = nxt_{i+1}$, with the exception of nxt_{i,s_i}, which equals i. This is easily done in $O(nk)$.

This would allow us to find i for each query in $O(|t|)$ per query, which satisfies the constraints of the problem. Now, we need to precompute the length of the smallest subsequence not present in each suffix of s. If we can do that quickly, we can then answer all queries in $O(|t|)$ by retrieving the answer for suffix $i + 1$.

Let a_i be our answer for the suffix of s starting from index i. In other words, the length of the smallest subsequence not present in the suffix of s starting at index i. We know that, if $a_i \geq 1$, it must start with one of our k letters.

As such, we can iterate from $1 \leq j \leq k$ and assume j to be the next letter. If j is the next letter, then our answer would be $1 + a_{nxt_{i,j}+1}$. This is because, if j is our first letter, our answer would be the length of the smallest subsequence not present in the suffix of s beginning at the index $nxt_{i,j} + 1$, as we $nxt_{i,j}$ is the nearest occurrence of j. Of course, if $nxt_{i,j}$ doesn't exist, then $a_i = 1$. We will take the minimum over all j. Thus, if we iterate from n to 1, we can compute all a_i in $O(nk)$.

Putting it all together, we use our nxt array to find the minimum index i such that t is a subsequence of substring of s from 1 to i in $O(|t|)$. If t is not a subsequence of s, the answer is 0. Otherwise, our answer is a_{i+1}. Here, $a_{n+1} = 1$. Thus, we can solve the problem in $O(nk + q + \sum |t|)$.