IOI 2024 Solutions: Problem Hieroglyphs

Problem Information

Problem Author: Félix Moreno Peñarrubia (Spain)

Problem Preparation: Mohammadreza Maleki, Yiping Liu, Félix M. Peñarrubia

Editorial: Félix Moreno Peñarrubia

Initial Observations

We refer to the values of the elements in the sequences as "characters", even though the alphabet Σ is not necessarily small. We assume that the size of the alphabet is $|\Sigma| = O(n + m)$ by performing coordinate compression at the beginning if necessary. Note also that we can ignore characters that appear in only one of the sequences. We abbreviate "universal common subsequence" as UCS. Let $\mathrm{occ}_x(A)$ denote the number of occurrences of character x in sequence A. Let $A[i \dots j]$ denote the contiguous subsequence of A from indices i to j inclusive (0-indexed); we will use parenthesis () to denote exclusive ranges at either or both of the endpoints.

Note that, by considering common subsequences of the form $xx \dots x$, we see that if the UCS C of A and B exists, then $\mathrm{occ}_x(C) = \min(\mathrm{occ}_x(A), \mathrm{occ}_x(B))$. So the number of occurrences of each character in the UCS is fixed, and we have to determine the relative order between them.

We say that a character x is A-weak if $\mathrm{occ}_x(A) \leq \mathrm{occ}_x(B)$, and we say x is B-weak if $\mathrm{occ}_x(B) < \mathrm{occ}_x(A)$. Note that we can determine the relative order within A-weak characters and within B-weak characters: the subsequence of A-weak characters of A and the subsequence of B-weak characters of B must be subsequences of C. We denote the subsequence of A-weak characters of A by W_A and the subsequence of B-weak characters of B by W_B.

Subtask 1

Since all characters appear in the UCS, the UCS must have length N, and therefore must be equal to A and also equal to B. Therefore we return A if $A = B$, otherwise return $[-1]$.

Subtask 2

If a character appears only in one of the sequences, we can ignore it. Therefore, we have that $\min(\mathrm{occ}_x(A), \mathrm{occ}_x(B)) = 1$ for all non-ignored characters, and the UCS consists of all distinct characters.

First, we find a candidate for the UCS. We need to merge the sequences W_A and W_B preserving relative order. If x is the first character of W_A and y is the first character of W_B, we can decide which character goes first by checking whether xy or yx are common subsequences of A and B. If the UCS exists, exactly one of the two sequences should be a common subsequence. We can check it in constant time, and we pop the corresponding character from the begininning of the corresponding sequence (W_A or W_B), add it to the end of the UCS candidate, and repeat until we have merged both sequences.

Once we have the candidate, we have to check if it is indeed a UCS. First, we check if it is a common subsequence. If it is, we can see that the fact that each character appears only once implies that the only way it can fail to be a UCS is if there are two characters x and y so that both xy and yx are common subsequences of A and B. This can only happen if we have the pattern $x \dots y \dots x$ in one sequence and $y \dots x \dots y$ in the other. This can be checked in $O(n \log n)$ using Segment Tree or Fenwick Tree, or in $O(n)$ using stacks.

Subtask 3

When the alphabet is binary, we have a simple characterization of when there is a UCS:

A UCS exists if and only if one sequence is a subsequence of the other or, after removing common prefix and suffix, one of the subsequences only has one of the two characters.

It is clear that if the condition holds there is an UCS, and it is easy to find the UCS. Let us prove that otherwise there is no UCS. Let A and B be sequences without any common prefix or suffix (it is easy to see that after removing common prefix or suffix the property of having or not having UCS is preserved) and such that none is a subsequence of the other. Assume that a UCS exists: we will prove that one of the sequences consists of a repetition of one of the two characters. Note that if a UCS exists and no sequence is a subsequence of the other, then one of the characters must be A-weak and the other must be B-weak, and it cannot happen that there is the same number of occurrences of a character in the two sequences. WLOG assume 0 is A-weak and 1 is B-weak. Note that if A starts with 0 and B with 1, the first character of the UCS can not be uniquely determined, so the UCS does not exist. Same if A ends with 0 and B with 1. Therefore, A starts and ends with 1 and B starts and ends with 0. WLOG assume the UCS starts with 0 (the other case is analogous). Suppose the UCS also contains an 1. Then, since there are strictly more 0s in B than in the UCS, either there are more 0s to the left of the last 1 in B than 0s to the left of the last 1 in the UCS, or there are more 0s to the right of the first 1 in B than 0s to the right of the first 1 in the UCS. We then have that a sequence of the form $0 \dots 01$ or $10 \dots 0$ is a common subsequence of A and B that is not a subsequence of the UCS. Therefore, the UCS can not contain 1s, and therefore B only contains 0s, as desired.

Subtask 4

In this subtask, we have to find a *candidate* for the UCS.

We will merge the sequences W_A and W_B like we did to find a candidate for Subtask 2, but this time we need to handle multiple appearances of characters. If x and y are the characters from W_A and W_B respectively that we are currently comparing, instead of testing whether the sequences xy and yx are common subsequences of A and B, we have to check the sequences $x \dots xy \dots y$ and $y \dots yx \dots x$, where the number of x in the first sequence is 1 plus the number of xs that have already been added to the candidate, and the number of ys is the number of remaining y that have not been added yet to the candidate (and respectively for the second subsequence). We can make the checks in constant time if we precompute, for each character, a list of its positions in each of the subsequences.

Subtask 5

Assuming that we have a candidate for the UCS, we want to verify it in $O(nm)$ time. Note that we can obtain a candidate in $O(nm)$ time by computing the LCS, so it is not necessary to solve the previous subtask to solve this one.

We use dynamic programming. Let $\text{DP}[i][j]$ be equal to the minimum k such that all common subsequences of $A[0 \dots i)$ and $B[0 \dots j)$ are subsequences of $C[0 \dots k)$. $\text{DP}[i+1][j+1]$ is equal to

- $\max(\text{DP}[i+1][j], \text{DP}[i][j+1])$ if $A[i] \neq B[j]$,
- $\max(\text{DP}[i+1][j], \text{DP}[i][j+1], \text{nextocc}(C, A[i], \text{DP}[i][j]))$ if $A[i] = B[j]$,

where $\text{nextocc}(C, x, k))$ is the next occurrence of character x in C after index k. We can precompute $\text{nextocc}(C, x, k))$ for all characters x that appear in both sequences and all indices k in $O(nm)$ time (after coordinate compression), so that the transitions can be computed in constant time and DP can be computed in $O(nm)$ total time. If for some combination of indices i, j with $A[i] = B[j]$, $\text{nextocc}(C, A[i], \text{DP}[i][j]))$ does not exist, then C is not a valid UCS, since there is a common subsequence of $A[0 \dots i]$ and $B[0 \dots j]$ which is not a common subsequence of C.

Full Solution

Following Subtask 4, we already found a candidate C for the UCS. In particular, we assume C satisfies the following:

- C has W_A and W_B as subsequences.
- C is a common subsequence of A and B.

Now we will check whether there exists a common subsequence of A and B that is not a common subsequence of C in $O(n \log n + m \log m)$ time.

Let us define some notation. Given a sequence S and a subsequence T, $I_L^{S,T}$ is the smallest index so that T is a subsequence of $S[0 \dots I_L^{S,T}]$ and $I_R^{S,T}$ is the largest index so that T is a subsequence of $S[I_R^{S,T} \dots |S|)$. Informally, $I_L^{S,T}$ is the last index of subsequence T in S when "going from left to right" and $I_R^{S,T}$ is the first index of subsequence T in S when "going from right to left". Also, if C is a UCS candidate as above, let $w_L(i) = I_L^{A;C[0\dots i]}$ if $C[i]$ is an A-weak character, and $w_L(i) = I_L^{B;C[0\dots i]}$ if $C[i]$ is a B-weak character (similarly define $w_R(i)$).

The key observation lies in the following lemma:

Lemma: Let C be a UCS candidate as above, and suppose that there exists a common subsequence of A and B that is not a a subsequence of C. Then, there exists a common subsequence S of A and B which is not a subsequence of C that can be expressed as $S = S_1 + S_2$ satisfying one of the following:

- S_1 ends with an A-weak character, S_2 begins with a B-weak character, $I_L^{A,S_1} = w_L(I_L^{C,S_1})$ and $I_R^{B,S_2} = w_R(I_R^{C,S_2})$.
- S_1 ends with a B-weak character, S_2 begins with an A-weak character, $I_L^{B,S_1} = w_L(I_L^{C,S_1})$ and $I_R^{A,S_2} = w_R(I_R^{C,S_2})$.

Proof: Take S to be a common subsequence of A and B and not a subsequence of C with minimum number of alternations between A-weak characters and B-weak characters. WLOG suppose that S ends with an A-weak character (the other case is analogous). Let S_2 be the suffix of A-weak characters of S, and let S_1 be the remaining characters.

We now prove that this is the desired decomposition. We have that $I_R^{A,S_2} = w_R(I_R^{C,S_2})$ since S_2 consists only of A-weak characters. We have that $I_L^{B,S_1} \leq w_L(I_L^{C,S_1})$ since S_1 is a subsequence of $C[0 \dots I_L^{C,S_1}]$ and $C[0 \dots I_L^{C,S_1}]$ is a subsequence of $B[0 \dots w_L(I_L^{C,S_1})]$. Assume that $I_L^{B,S_1} < w_L(I_L^{C,S_1})$ and consider the sequence S' equal to S_1 concatenated with $C[I_L^{C,S_1}]$ repeated as many times as the character appears in $C[I_L^{C,S_1} \dots |C|)$. By construction, S' is a common subsequence of A and B (it is a subsequence of A because of the inequality $I_L^{A,S_1} < I_R^{A,S_2} = w_R(I_R^{C,S_2}) < w_R(I_L^{C,S_1})$) which is not a subsequence of C, and it has fewer alternations than S, contradiction. Therefore, $I_L^{B,S_1} = w_L(I_L^{C,S_1})$, as desired. $\square$

Now, we use the lemma to perform the check efficiently.

Given two sequences S and T, let $V_L^{S;T}$ be an array of size $|T|$ for which $V_L^{S;T}[i]$ is equal to the lowest index so that there is a subsequence S' of $S[0 \dots V_L^{S;T}[i]]$ for which $I_L^{T;S'} = i$. Similarly define $V_R^{S;T}[i]$ as the highest index so that there is a subsequence S' of $S[V_R^{S;T}[i] \dots |S|)$ for which $I_R^{T;S'} = i$. $V_L^{S;T}$ and $V_R^{S;T}$ can be computed in time $O(|T| \log |T| + |S|)$ using range minimum queries or binary search in cumulative minimum stack.

Compute the vectors $V_L^{A;C}$, $V_R^{A;C}$, $V_L^{B;C}$, $V_R^{B;C}$. By the lemma, if C is not a

UCS, then there exist indices $0 \le i < j < |C|$ so that either:

- $C[i]$ is A-weak, $C[j]$ is B-weak, $V_L^{A;C}[j] < w_R(i)$ and $V_R^{B;C}[i] > w_L(j)$.
- $C[i]$ is B-weak, $C[j]$ is A-weak, $V_L^{B;C}[j] < w_R(i)$ and $V_R^{A;C}[i] > w_L(j)$.

These conditions can be checked in $O(n \log n)$ time using Segment Tree or Fenwick Tree, or $O(n)$ time using stacks.