Journal of Mathematics and Informatics
Vol. 28, 2025, 59-72
ISSN: 2349-0632 (P), 2349-0640 (online)
Published 6 August 2025
www.researchmathsci.org
DOI:http://dx.doi.org/10.22457/jmi.v28a06259

Journal of **Mathematics and Informatics**

An Algorithm for the Constrained Longest Common Subsequence and Substring Problem for Multiple Strings*

Rao Li[1*], ***Richy Modugu***[2] **and** ***Brandon Weathers***[3]

[1]Department of Computer Science, Engineering, and Mathematics,
University of South Carolina Aiken, Aiken, SC 29801, USA,
Email: raol@usca.edu
[2]Department of Computer Science, Engineering, and Mathematics,
University of South Carolina Aiken, Aiken, SC 29801, USA,
Email: rmodugu@usca.edu
[3]Department of Computer Science, Engineering, and Mathematics,
University of South Carolina Aiken, Aiken, SC 29801, USA,
Email: brw12@usca.edu
*Corresponding author

Received 2 July 2025; accepted 5 August 2025

Abstract. Let ∑ be an alphabet. For multiple strings X, Y_1, Y_2, ..., Y_n, and a constrained string P over the alphabet ∑, we introduce the constrained longest common subsequence and substring problem for strings X, Y_1, Y_2, ..., Y_n with respect to P as to find a longest string Z which is a subsequence of X, a substring of Y_1, Y_2, ..., and Y_n, and has P as a subsequence. In this paper, we propose an algorithm for solving the above problem.

Keywords: Longest common subsequence, longest common subsequence and substring, constrained longest common subsequence and substring, constrained longest common subsequence and substring for multiple strings.

AMS Mathematics Subject Classification (2010): 68W32, 68W40

1. Introduction

Let ∑ be an alphabet and S a string over ∑. A subsequence of a string S over an alphabet ∑ is obtained by deleting zero or more letters of S. A substring of a string S is a subsequence of S consisting of consecutive letters in S. An empty string is a string that does not have any letters in it. An empty string is a subsequence and substring of any string. The number of letters in a string S, denoted |S|, is called the length, of the string S. The longest common subsequence problem (LCSSeq) for two strings is to find a longest string which is a

* This research was funded by the Summer Scholars Institute (2025) at University of South Carolina Aiken.

subsequence of both strings. The longest common substring (LCSStr) problem for two strings is to find a longest string which is a substring of both strings.

Both the longest common subsequence problem and the longest common substring problem have been well-investigated in the last several decades. More details on the studies for the LCSSeq problem can be found in [2], [3], [4], [5], [7], [9], [11], [12], [13], [16], [17], and[18] and the LCSStr problem can be found in [1], [8], [10], and [20].

Motivated by LCSSeq and LCSStr problems, Li, Deka, and Deka [14] introduced the longest common subsequence and substring (LCSSeqSStr) problem for two strings. For two strings X and Y, the longest common subsequence and substring problem for X and Y is to find a longest string which is a subsequence of X and a substring of Y. They also designed an O(|X||Y|) time algorithm for LCSSeqSStr problem for two strings X and Y in [14].

Motivated by LCSSeq problem, Tsai [19] extended the longest common subsequence problem for two strings to the constrained longest common subsequence (CLCSSeq) problem for two strings. For two strings X, Y, and a constrained string P, the constrained longest common subsequence problem for two strings X and Y with respect to P is to find a longest string Z such that Z is a common subsequence for X and Y and P is a subsequence of Z. Tsai designed an $O(|X|^2\ |Y|^2\ |P|)$ time algorithm for the CLCSSeq problem for two strings in [19]. Chin et al. [6] improved Tsai's algorithm and designed an O(|X| |Y| |P|) time algorithm for the CLCSSeq problem for two strings X and Y and a constrained string P.

Motivated by Li, Deka, and Deka's LCSSeqSStr problem and Tsai's CLCSSeq problem, Li, Deka, Deka, and Li [15] introduced the constrained longest common subsequence and substring problem for two strings with respect to a constrained string. For two strings X, Y, and a constrained string P, the constrained longest common subsequence and substring (CLCSSeqSStr) problem for two strings X and Y with respect to P is to find a longest string Z such that Z is a subsequence of X, a substring of Y, and has P as a subsequence. Clearly, the LCSSeqSStr problem is a special CLCSSeqSStr problem with an empty constrained string. Li, Deka, Deka, and Li [15] designed an O(|X| |Y| |P|) time algorithm for the CLCSSeqSStr problem for two strings and a constrained string.

In this paper, we further generalize the CLCSSeqSStr problem as follows. For multiple strings X, Y_1, Y_2, ..., Y_n, and a constrained string P over an alphabet ∑, we define the constrained longest common subsequence and substring (CLCSSeqSStrM) problem for strings X, Y_1, Y_2, ..., and Y_n with respect to P as to find a longest string Z which is a subsequence of X, a substring of Y_1, Y_2, ..., and Y_n, and has P as a subsequence. We will propose an algorithm to solve the CLCSSeqSStrM problem in this paper.

2. The Recursions in the algorithm

In order to present our algorithm, we need to establish some recursions to be used in our algorithm. Before doing that, we need some notations as follows. For a given string $S = s_1\ s_2\ ...\ s_l$ over an alphabet ∑, the ith prefix of S is defined as $S[i] = s_1\ s_2\ ...\ s_i$, where $1 \leq i \leq l$. Conventionally, S[0] is defined as an empty string. The l suffixes of S are the strings of $s_1\ s_2\ ...\ s_l$, $s_2\ s_3\ ...\ s_l$, ..., $s_{l-1} s_l$, and s_l. Let

$$X = x_1\ x_2\ ...\ x_m,$$
$$Y_1 = y[1,1]\ y[1,2]\ ...\ y[1,p_1],$$

$Y_2 = y[2, 1]\ y[2, 2]\ ...\ y[2, p_2]$,
......
$Y_n = y[n, 1]\ y[n, 2]\ ...\ y[n, p_n]$, and
$P = p_1\ p_2\ ...\ p_r$.

We define $Z[i, j_1, j_2, ..., j_n, k]$ as a string satisfying the following conditions:

(1.1) it is a subsequence of $X[i] = x_1\ x_2\ ...\ x_i$,
(2.1) it is a suffix of $Y_1[j_1] = y[1, 1]\ y[1, 2]\ ...\ y[1, j_1]$,
(2.2) it is a suffix of $Y_2[j_2] = y[2, 1]\ y[2, 2]\ ...\ y[2, j_2]$,
......
(2.n) it is a suffix of $Y_n[j_n] = y[n, 1]\ y[n, 2]\ ...\ y[n, j_n]$,
(3.1) it has P_k as a subsequence,
(4.1) under the conditions above, its length is maximum,

where $0 \le i \le m$, $0 \le j_1 \le p_1$, $0 \le j_2 \le p_2$, ..., $0 \le j_n \le p_n$, and $0 \le k \le r$.

Obviously, if ($i = 0$ and $k = 0$) or ($j_1 = 0$ and $k = 0$) or ($j_2 = 0$ and $k = 0$) or ... or ($j_n = 0$ and $k = 0$), then $Z[i, j_1, j_2, ..., j_n, k]$ is an empty string and $|Z[i, j_1, j_2, ..., j_n, k]| = 0$.

Also, if ($i = 0$ and $k \ge 1$) or ($j_1 = 0$ and $k \ge 1$) or ($j_2 = 0$ and $k \ge 1$) or ... or ($j_n = 0$ and $k \ge 1$), then $Z[i, j_1, j_2, ..., j_n, k]$ do not exist.

Next, we will prove the following claims on $Z[i, j_1, j_2, ..., j_n, k]$.

Claim 1. Assume $k = 0$, $i \ge 1$, $j_1 \ge 1$, $j_2 \ge 1$, ... , and $j_n \ge 1$. If $y[1, j_1]$, $y[2, j_2]$, ... , and $y[n, j_n]$ are not the same, then $Z[i, j_1, j_2, ..., j_n, k]$ is an empty string.

Proof of Claim 1. Suppose $Z[i, j_1, j_2, ..., j_n, k]$ is not empty. Then the last letter of it must be equal to $y[1, j_1]$, $y[2, j_2]$, ... , and $y[n, j_n]$. Thus $y[1, j_1] = y[2, j_2] = \ldots = y[n, j_n]$, a contradiction. Hence the proof of Claim 1 is complete.

Claim 2. Assume $k = 0$, $i \ge 1$, $j_1 \ge 1$, $j_2 \ge 1$, ... , and $j_n \ge 1$. If $y[1, j_1] = y[2, j_2] = \ldots = y[n, j_n] := \omega$, then

Case 2.1. if $x_i = \omega$, then $|Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]| + 1$.
Case 2.2. if $x_i \neq \omega$, then $|Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1, j_2, ..., j_n, k]|$.

Proof of Case 2.1 in Claim 2. In this case, it is clear that the string $Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]\ \omega$

(1.1) is a subsequence of $X[i]$,
(2.1) is a suffix of $Y_1[j_1]$,
(2.2) is a suffix of $Y_2[j_2]$,
......
(2.n) is a suffix of $Y_n[j_n]$,
(3.1) has P_k, which is empty, as a subsequence.

By the definition of $Z[i, j_1, j_2, ..., j_n, k]$, we have that $|Z[i, j_1, j_2, ..., j_n, k]| \geq |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]\ \omega| = |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]| + 1$.

Suppose $Z[i, j\_1, j\_2, ..., j\_n, k] = u_1 u_2 ... u_{a-1} u_a$. Then $u_a = y[1, j_1] = y[2, j_2] = \ldots = y[n, j_n] = \omega$. Thus $Z[i, j_1, j_2, ..., j_n, k] – \{u_a\} = u_1 u_2 ... u_{a-1}$ is

(1.1) is a subsequence of $X[i - 1]$,
(2.1) is a suffix of $Y_1[j_1 - 1]$,
(2.2) is a suffix of $Y_2[j_2 - 1]$,
......
(2.n) is a suffix of $Y_n[j_n - 1]$,
(3.1) has P_k, which is empty, as a subsequence.

By the definition of $Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]$, we have that $|Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]| \geq |Z[i, j_1, j_2, ..., j_n, k] - \{u_a\}| = |Z[i, j_1, j_2, ..., j_n, k]| - 1$.

Hence $|Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]| + 1$ and the proof of Case 2.1 in Claim 2 is complete.

Proof of Case 2.2 in Claim 2. In this case, it is clear that the string $Z[i - 1, j_1, j_2, ..., j_n, k]$ is

(1.1) is a subsequence of $X[i]$,
(2.1) is a suffix of $Y_1[j_1]$,
(2.2) is a suffix of $Y_2[j_2]$,
......
(2.n) is a suffix of $Y_n[j_n]$,
(3.1) has $P_k, which is empty, as a subsequence.

By the definition of $Z[i, j_1, j_2, ..., j_n, k]$, we have that $|Z[i, j_1, j_2, ..., j_n, k]| \geq |Z[i - 1, j_1, j_2, ..., j_n, k]|$.

Suppose $Z[i, j_1, j_2, ..., j_n, k] = u_1 u_2 ... u_{b-1} u_b$. Then $u_b = y[1, j\_1] = y[2, j\_2] = \ldots = y[n, j_n] = \omega \neq x_i$. Thus $Z[i, j_1, j_2, ..., j_n, k]$ is

(1.1) is a subsequence of $X[i - 1]$,
(2.1) is a suffix of $Y_1[j_1]$,
(2.2) is a suffix of $Y_2[j_2]$,
......
(2.n) is a suffix of $Y_n[j_n]$,
(3.1) has P_k, which is empty, as a subsequence.

By the definition of $Z[i - 1, j_1, j_2, ..., j_n, k]$, we have that $Z[i - 1, j_1, j_2, ..., j_n, k]| \geq |Z[i, j_1, j_2, ..., j_n, k]|$.

Hence $|Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1, j_2, ..., j_n, k]|$ and the proof of Case 2.2 in Claim 2 is complete.

Claim 3. Assume $k \geq 1$, $i \geq 1$, $j_1 \geq 1$, $j_2 \geq 1$, ... , and $j_n \geq 1$. If $y[1, j_1]$, $y[2, j_2]$, ... , and $y[n, j_n]$ are not the same, then $Z[i, j_1, j_2, ..., j_n, k]$ does not exist.

Proof of Claim 3. Suppose $Z[i, j_1, j_2, ..., j_n, k]$ exists. Notice that the condition of $k \geq 1$ implies that $Z[i, j_1, j_2, ..., j_n, k]$ is not empty. Thus the last letter of $Z[i, j_1, j_2, ..., j_n, k]$ must be equal to $y[1, j_1]$, $y[2, j_2]$, ... , and $y[n, j_n]$. Thus $y[1, j_1] = y[2, j_2] = \ldots = y[n, j_n]$, a contradiction. Hence the proof of Claim 3 is complete.

Claim 4. Assume $k \geq 1$, $i \geq 1$, $j_1 \geq 1$, $j_2 \geq 1$, ... , and $j_n \geq 1$. If $y[1, j_1] = y[2, j_2] = \ldots = y[n, j_n] := \omega$ and $Z[i, j_1, j_2, ..., j_n, k]$ exists, then we just have the following cases and the statement in each case is true.

Case 4.1. $x_i = \omega = p_k$, and $|Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k - 1]| + 1$ in this case.
Case 4.2. $x_i = \omega \neq p_k$, and $|Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]| + 1$ in this case.
Case 4.3. $x_i \neq \omega$, $x_i \neq p_k$, $\omega = p_k$, and $|Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1, j_2, ..., j_n, k]|$ in this case.
Case 4.4. $x_i \neq \omega$, $x_i \neq p_k$, $\omega \neq p_k$, and $|Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1, j_2, ..., j_n, k]|$ in this case.
Case 4.5. $x_i \neq \omega$, $x_i = p_k$, $\omega \neq p_k$, and this case cannot happen.

Proof of Claim 4. The five cases can be figured out in the following way. Firstly, we have two cases of $x_i = \omega$ or $x_i \neq \omega$. When $x_i = \omega$, we just can have two possible cases of $x_i = \omega = p_k$ or $x_i = \omega \neq p_k$. When $x_i \neq \omega$, we just can have three possible cases of $x_i \neq p_k$ and $\omega = p_k$, $x_i \neq p_k$ and $\omega \neq p_k$, or $x_i = p_k$ and $\omega \neq p_k$. Since $Z[i, j_1, j_2, ..., j_n, k]$ exists and $k \geq 1$, $Z[i, j_1, j_2, ..., j_n, k]$ is not empty. Next we will prove the statements in the five cases.

Case 4.1. $x_i = \omega = p_k$.
In this case, it is clear that $Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k - 1]\ \omega$

(1.1) is a subsequence of $X[i]$,
(2.1) is a suffix of $Y_1[j_1]$,
(2.2) is a suffix of $Y_2[j_2]$,
......
(2.n) is a suffix of $Y_n[j_n]$,
(3.1) has P_k as a subsequence.

By the definition of $Z[i, j_1, j_2, ..., j_n, k]$, we have that $|Z[i, j_1, j_2, ..., j_n, k]| \geq |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k - 1]\ \omega| = |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k - 1]| + 1$.

Suppose $Z[i, j_1, j_2, ..., j_n, k] = u_1 u_2 \ldots u_{c-1} u_c$. Then $u_c = y[1, j_1] = y[2, j_2] = \ldots = y[n, j_n] = \omega = x_i = p_k$. Thus $Z[i, j_1, j_2, ..., j_n, k] - \{u_c\} = u_1 u_2 \ldots u_{c-1}$

(1.1) is a subsequence of $X[i - 1]$,
(2.1) is a suffix of $Y_1[j_1 - 1]$,
(2.2) is a suffix of $Y_2[j_2 - 1]$,

......
(2.n) is a suffix of $Y_n[j_n - 1]$,
(3.1) has P_{k-1} as a subsequence.

By the definition of $Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k - 1]$, we have $|Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k - 1]| \geq |Z[i, j_1, j_2, ..., j_n, k] - \{u_c\}| = |Z[i, j_1, j_2, ..., j_n, k]| - 1$.

Hence $|Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k - 1]| + 1$ and the proof of Case 4.1 in Claim 4 is complete.

Case 4.2. $x_i = \omega \neq p_k$.
In this case, it is clear that $Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]\ \omega$

(1.1) is a subsequence of $X[i]$,
(2.1) is a suffix of $Y_1[j_1]$,
(2.2) is a suffix of $Y_2[j_2]$,
......
(2.n) is a suffix of $Y_n[j_n]$,
(3.1) has P_k as a subsequence.

By the definition of $Z[i, j_1, j_2, ..., j_n, k]$, we have that $|Z[i, j_1, j_2, ..., j_n, k]| \geq |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]\ \omega| = |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]| + 1$.

Suppose $Z[i, j_1, j_2, ..., j_n, k] = u_1 u_2 ... u_{d-1} u_d$. Then $u_d = y[1, j_1] = y[2, j_2] = … = y[n, j_n] = \omega = x_i \neq p_k$. Thus $Z[i, j_1, j_2, ..., j_n, k] - \{u_d\} = u_1 u_2 ... u_{d-1}$

(1.1) is a subsequence of $X[i - 1]$,
(2.1) is a suffix of $Y_1[j_1 - 1]$,
(2.2) is a suffix of $Y_2[j_2 - 1]$,
......
(2.n) is a suffix of $Y_n[j_n - 1]$,
(3.1) has P_k as a subsequence.

By the definition of $Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]$, we have $|Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]| \geq |Z[i, j_1, j_2, ..., j_n, k] - \{u_d\}| = |Z[i, j_1, j_2, ..., j_n, k]| - 1$.

Hence $|Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]| + 1$ and the proof of Case 4.2 in Claim 4 is complete.

Case 4.3. $x_i \neq \omega$, $x_i \neq p_k$, $\omega = p_k$.
In this case, it is clear that $Z[i - 1, j_1, j_2, ..., j_n, k]$

(1.1) is a subsequence of $X[i]$,
(2.1) is a suffix of $Y_1[j_1]$,
(2.2) is a suffix of $Y_2[j_2]$,
......
(2.n) is a suffix of $Yn[j_n]$,
(3.1) has P_k as a subsequence.

By the definition of $Z[i, j_1, j_2, ..., j_n, k]$, we have that $|Z[i, j_1, j_2, ..., j_n, k]| \geq |Z[i - 1, j_1, j_2, ..., j_n, k]|$.

Suppose $Z[i, j_1, j_2, ..., j_n, k] = u_1 u_2 ... u_{e-1} u_e$. Then $u_e = y[1, j_1] = y[2, j_2] = \ldots = y[n, jn] = \omega \neq x_i$. Thus $Z[i, j_1, j_2, ..., j_n, k]$

(1.1) is a subsequence of $X[i - 1]$,
(2.1) is a suffix of $Y_1[j_1]$,
(2.2) is a suffix of $Y_2[j_2]$,
......
(2.n) is a suffix of $Y_n[j_n]$,
(3.1) has P_k as a subsequence.

By the definition of $Z[i - 1, j_1, j_2, ..., j_n, k]$, we have $|Z[i - 1, j_1, j_2, ..., j_n, k]| \geq |Z[i, j_1, j_2, ..., j_n, k]|$.

Hence $|Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1, j_2, ..., j_n, k]|$ and the proof of Case 4.3 in Claim 4 is complete.

Case 4.4. $x_i \neq \omega$, $x_i \neq p_k$, $\omega \neq p_k$.
In this case, it is clear that $Z[i - 1, j_1, j_2, ..., j_n, k]$

(1.1) is a subsequence of $X[i]$,
(2.1) is a suffix of $Y_1[j_1]$,
(2.2) is a suffix of $Y_2[j_2]$,
......
(2.n) is a suffix of $Y_n[j_n]$,
(3.1) has P_k as a subsequence.

By the definition of $Z[i, j_1, j_2, ..., j_n, k]$, we have that $|Z[i, j_1, j_2, ..., j_n, k]| \geq |Z[i - 1, j_1, j_2, ..., j_n, k]|$.

Suppose $Z[i, j_1, j_2, ..., j_n, k] = u_1 u_2 ... u_{f-1} u_f$. Then $u_f = y[1, j_1] = y[2, j_2] = \ldots = y[n, j\_n] = \omega \neq x_i$. Thus $Z[i, j_1, j_2, ..., j_n, k]$

(1.1) is a subsequence of $X[i - 1]$,
(2.1) is a suffix of $Y_1[j_1]$,
(2.2) is a suffix of $Y_2[j_2]$,
......
(2.n) is a suffix of $Y_n[j_n]$,
(3.1) has P_k as a subsequence.

By the definition of $Z[i - 1, j_1, j_2, ..., j_n, k]$, we have $|Z[i - 1, j_1, j_2, ..., j_n, k]| \geq |Z[i, j_1, j_2, ..., j_n, k]|$.

Hence $|Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1, j_2, ..., j_n, k]|$ and the proof of Case 4.4 in Claim 4 is complete.

Case 4.5. $x_i \neq \omega$, $x_i = p_k$, and $\omega \neq p_k$.

Suppose $Z[i, j_1, j_2, ..., j_n, k] = u_1 u_2 ... u_{g-1} u_g$. Then $u_g = y[1, j_1] = y[2, j_2] = \ldots = y[n, j_n] = \omega \neq x_i$. Since $u_1 u_2 ... u_{g-1} u_g$ is a subsequence of X_i and $x_i \neq u_g$, we have that u_g appears before x_i on X_i. Since $p_1 p_2 ... p_k$ is a subsequence of $u_1 u_2 ... u_{g-1} u_g$, p_k appears in $u_1 u_2 ... u_{g-1} u_g$ which is a subsequence of X_i, contradicting to $p_k = x_i$. Thus this case cannot happen and the proof of Case 4.5 in Claim 4 is complete. Since this case does not happen, it is not necessary for us to deal with this case in our algorithm.

Claim 5. Assume $k \geq 1$, $i \geq 1$, $j_1 \geq 1$, $j_2 \geq 1$, ... , and $j_n \geq 1$. If $y[1, j_1] = y[2, j_2] = \ldots = y[n, j_n] := \omega$ and $Z[i, j_1, j_2, ..., j_n, k]$ does not exist, then

[1]. If $x_i = \omega = p_k$, then $Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k - 1]$ does not exist.

[2]. If $x_i = \omega \neq p_k$, then $Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]$ does not exist.

[3]. If $x_i \neq \omega$, $x_i \neq p_k$, $\omega = p_k$, then $Z[i - 1, j_1, j_2, ..., j_n, k]$ does not exist.

[4]. If $x_i \neq \omega$, $x_i \neq p_k$, $\omega \neq p_k$, then $Z[i - 1, j_1, j_2, ..., j_n, k]$ does not exist.

Proof of [1] in Claim 5. Suppose $Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k - 1]$ exists. Since $x_i = \omega = p_k$, $Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k - 1]\ \omega$

(1.1) is a subsequence of X[i],
(2.1) is a suffix of $Y_1[j_1]$,
(2.2) is a suffix of $Y_2[j_2]$,
......
(2.n) is a suffix of $Y_n[j_n]$,
(3.1) has P_k as a subsequence.

This implies that $Z[i, j_1, j_2, ..., j_n, k]$ exists, a contradiction. Thus the proof of [1] in Claim 5 is complete.

Proof of [2] in Claim 5. Suppose $Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]$ exists. Since $x_i = \omega \neq p_k$, $Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]\ \omega$

(1.1) is a subsequence of X[i],
(2.1) is a suffix of $Y_1[j_1]$,
(2.2) is a suffix of $Y_2[j_2]$,
......
(2.n) is a suffix of $Y_n[j_n]$,
(3.1) has P_k as a subsequence.

This implies that $Z[i, j_1, j_2, ..., j_n, k]$ exists, a contradiction. Thus the proof of [2] in Claim 5 is complete.

Proof of [3] in Claim 5. Suppose $Z[i - 1, j_1, j_2, ..., j_n, k]$ exists. Since $x_i \neq \omega$, $x_i \neq p_k$, $\omega = p_k$, $Z[i - 1, j_1, j_2, ..., j_n, k]$

(1.1) is a subsequence of X[i],
(2.1) is a suffix of $Y_1[j_1]$,
(2.2) is a suffix of $Y_2[j_2]$,
......
(2.n) is a suffix of $Y_n[j_n]$,
(3.1) has P_k as a subsequence.

This implies that $Z[i, j_1, j_2, ..., j_n, k]$ exists, a contradiction. Thus the proof of [3] in Claim 5 is complete.

Proof of [4] in Claim 5. Suppose $Z[i - 1, j_1, j_2, ..., j_n, k]$ exists. Since $x_i \neq \omega$, $x_i \neq p_k$, $\omega \neq p_k$, $Z[i - 1, j_1, j_2, ..., j_n, k]$

(1.1) is a subsequence of X[i],
(2.1) is a suffix of $Y_1[j_1]$,
(2.2) is a suffix of $Y_2[j_2]$
......
(2.n) is a suffix of $Y_n[j_n]$,
(3.1) has P_k as a subsequence.

This implies that $Z[i, j_1, j_2, ..., j_n, k]$ exists, a contradiction. Thus the proof of [4] in Claim 5 is complete.

Claim 6. Let $U^k = u_1^k u_2^k ... u_{h(k)}^k$, where $0 \leq k \leq r$, be a longest string which

(1.1) is a subsequence of X,
(2.1) is a substring of Y_1,
(2.2) is a substring of Y_2,
......
(2.n) is a substring of Y_n,
(3.1) has P_k as a subsequence.

Then $h(k) = \max\{|Z[i, j_1, j_2, ..., j_n, k]| : 1 \leq i \leq m, 1 \leq j_1 \leq p_1, 1 \leq j_2 \leq p_2, ..., 1 \leq j_n \leq p_n, 0 \leq k \leq r\}$.

Proof of Claim 6. For each i with $1 \leq i \leq m$, each j_1 with $1 \leq j_1 \leq p_1$, each j_2 with $1 \leq j_2 \leq p_2$, ..., each j_n with $1 \leq j_n \leq p_n$, and each k with $0 \leq k \leq r$. By the definition of $Z[i, j_1, j_2, ..., j_n, k]$, we have that

(1.1) is a subsequence of X,
(2.1) is a substring of Y_1,
(2.2) is a substring of Y_2,
......
(2.n) is a substring of Y_n,

(3.1) has P_k as a subsequence.

By the definition of U^k, we have that $|Z[i, j_1, j_2, ..., j_n, k]| \leq |U^k| = h(k)$. Thus $\max\{ |Z[i, j_1, j_2, ..., j_n, k]| : 1 \leq i \leq m, 1 \leq j_1 \leq p_1, 1 \leq j_2 \leq p_2, ..., 1 \leq j_n \leq p_n, 1 \leq k \leq r \} \leq h(k)$.

Since $U^k = u_1^k u_2^k ... u_{h(k)}^k$ is a longest string which

(1.1) is a subsequence of X,
(2.1) is a substring of Y_1,
(2.2) is a substring of Y_2,
......
(2.n) is a substring of Y_n,
(3.1) has P_k as a subsequence,

there are indices i, l_1, l_2, ..., and l_n with $u_{h(k)}^k = x_i$, $u_{h(k)}^k = y[1, l_1]$, $u_{h(k)}^k = y[2, l_2]$, ..., $u_{h(k}^k = y[n, l_n]$, and $U^k = u_1^k u_2^k ... u_{h(k)}^k$ has P_k as a subsequence, where $0 \leq k \leq r$ such that $U^k = u_1^k u_2^k ... u_{h(k)}^k$ is a string which

(1.1) is a subsequence of X[i],
(2.1) is a suffix of $Y_1[l_1]$,
(2.2) is a suffix of $Y_2[l_2]$,
......
(2.n) is a suffix of $Y_n[l_n]$,
(3.1) has P_k as a subsequence.

By the definition of $Z[i, j_1, j_2, ..., j_n, k]$, we have that $h(k) \leq |Z[i, l_1, l_2, ..., l_n, k]| \leq \max\{ |Z[i, j_1, j_2, ..., j_n, k]| : 1 \leq i \leq m, 1 \leq j_1 \leq p_1, 1 \leq j_2 \leq p_2, ..., 1 \leq j_n \leq p_n, 0 \leq k \leq r \}$.

Hence $h(k) = \max\{|Z[i, j_1, j_2, ..., j_n, k]| : 1 \leq i \leq m, 1 \leq j_1 \leq p_1, 1 \leq j_2 \leq p_2, ..., 1 \leq j_n \leq p_n, 0 \leq k \leq r\}$ and the proof of Claim 6 is complete.

3. The algorithm

Now we can present our algorithm. Let us recall

$X = x_1 x_2 ... x_m$,
$Y_1 = y[1,1] y[1, 2] ... y[1, p_1]$,
$Y_2 = y[2, 1] y[2, 2] ... y[2, p_2]$,
......
$Y_n = y[n, 1] y[n, 2] ... y[n, p_n]$, and
$P = p_1 p_2 ... p_r$.

Let M be a (n + 2)-dimensional array of size $(m + 1)(p_1 + 1)(p_2 + 1) … (p_n + 1)(r + 1)$.

For $0 \leq i \leq m$, $0 \leq j_1 \leq p_1$, $0 \leq j_2 \leq p_2$, ..., $0 \leq j_n \leq p_n$, $0 \leq k \leq r$, if Z[i, j_1, j_2, ..., j_n, k] exist, the cell M[i][j_1][j_2] …[j_n][k] = |Z[i, j_1, j_2, ..., j_n, k]|; if Z[i, j_1, j_2, ..., j_n, k] do not exist, the cell M[i][j_1][j_2] … [j_n][k] = $-\infty$, where ∞ is a larger number. For instance, ∞ can be $1000(m + 1)(p_1 + 1)(p_2 + 1) \dots (p_n + 1)(r + 1)$. Our algorithm consists of the following steps. Firstly, we will fill in the cells in array M.

Step 1. If (i = 0 and k = 0) or (j_1 = 0 and k = 0) or (j_2 = 0 and k = 0) or ... or (j_n = 0 and k = 0), then Z[i, j_1, j_2, ..., j_n, k] is an empty string and |Z[i, j_1, j_2, ..., j_n, k]| = 0. Thus M[i][j_1][j_2] … [j_n][k] = 0.

Step 2. If (i = 0 and $k \geq 1$) or (j_1 = 0 and $k \geq 1$) or (j_2 = 0 and $k \geq 1$) or ... or (j_n = 0 and $k \geq 1$), then Z[i, j_1, j_2, ..., j_n, k] do not exist. Thus M[i][j_1][j_2] … [j_n][k] = $-\infty$.

Step 3. If k = 0, $i \geq 1$, $j_1 \geq 1$, $j_2 \geq 1$, ..., and $j_n \geq 1$, y[1, j_1], y[2, j_2], ... , and y[n, j_n] are not the same, then Z[i, j_1, j_2, ..., j_n, k] is an empty string. Thus M[i][j_1][j_2] … [j_n][k] = 0.

Step 4. If k = 0, $i \geq 1$, $j_1 \geq 1$, $j_2 \geq 1$, ... , and $j_n \geq 1$, y[1, j_1] = y[2, j_2] = … = y[n, j_n] : = ω, and $x_i = \omega$, then |Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]| + 1. Thus M[i][j_1][j_2] … [j_n][k] = M[i - 1][j_1 - 1][j_2 - 1] … [j_n - 1][k] + 1.

Step 5. If k = 0, $i \geq 1$, $j_1 \geq 1$, $j_2 \geq 1$, ... , and $j_n \geq 1$, y[1, j_1] = y[2, j_2] = … = y[n, j_n] : = ω, and $x_i \neq \omega$, then |Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1, j_2, ..., j_n, k]|. Thus M[i][j_1][j_2] … [j_n][k] = M[i - 1][j_1][j_2] … [j_n][k].

Step 6. If $k \geq 1$, $i \geq 1$, $j_1 \geq 1$, $j_2 \geq 1$, ... , and $j_n \geq 1$, y[1, j_1], y[2, j_2], ..., and y[n, j_n] are not the same, then Z[i, j_1, j_2, ..., j_n, k] do not exist. Thus M[i][j_1][j_2] … [j_n][k] = $-\infty$.

Step 7. If $k \geq 1$, $i \geq 1$, $j_1 \geq 1$, $j_2 \geq 1$, ... , and $j_n \geq 1$, y[1, j_1] = y[2, j_2] = … = y[n, j_n] : = ω, and $x_i = \omega = p_k$, then |Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k -]| + 1. Thus M[i][j_1][j_2] … [j_n][k] = M[i - 1][j_1 - 1][j_2 - 1] … [j_n - 1][k - 1] + 1.

Step 8. If $k \geq 1$, $i \geq 1$, $j_1 \geq 1$, $j_2 \geq 1$, ..., and $j_n \geq 1$, y[1, j_1] = y[2, j_2] = … = y[n, j_n] : = ω, and $x_i = \omega \neq p_k$, then |Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1 - 1, j_2 - 1, ..., j_n - 1, k]| + 1. Thus M[i][j_1][j_2] … [j_n][k] = M[i - 1][j_1 - 1][j_2 - 1] … [j_n - 1][k] + 1.

Step 9. If $k \geq 1$, $i \geq 1$, $j_1 \geq 1$, $j_2 \geq 1$, ... , and $j_n \geq 1$, y[1, j_1] = y[2, j_2] = … = y[n, j_n] : = ω, and $x_i \neq \omega$, $x_i \neq p_k$, $\omega = p_k$, then |Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1, j_2, ..., j_n, k]|. Thus M[i][j_1][j_2] … [j_n][k] = M[i - 1][j_1][j_2] … [j_n][k].

Step 10. If $k \geq 1$, $i \geq 1$, $j_1 \geq 1$, $j_2 \geq 1$, ... , and $j_n \geq 1$, y[1, j_1] = y[2, j_2] = … = y[n, j_n] : = ω, and $x_i \neq \omega$, $x_i \neq p_k$, $\omega \neq p_k$, then |Z[i, j_1, j_2, ..., j_n, k]| = |Z[i - 1, j_1, j_2, ..., j_n, k]|. Thus M[i][j_1][j_2] … [j_n][k] = M[i - 1][j_1][j_2] … [j_n][k].

Notice that Claim 6 implies that if a longest string which

(1.1) is a subsequence of X,
(2.1) is a substring of Y_1,
(2.2) is a substring of Y_2,
......
(2.n) is a substring of Y_n,
(3.1) has Pr as a subsequence,

called a desired string, exists, then its length is equal to max { $M[i][j_1][j_2] \dots [j_n][k]$: $1 \le i \le m$, $1 \le j_1 \le p_1$, $1 \le j_2 \le p_2$, …, $1 \le j_n \le p_n$, $0 \le k \le r$ }. Hence a desired string can be found in the following steps.

Step 11. Define one variable called *maxLength* which eventually denotes the length of a desired string and its initial value is 0.

Step 12. Define another variable called *lastIndexOnY1* which eventually denotes the last index of the desired string on the string Y_1 and its initial value is p_1.

Step 13. Visit all the cells of $M[i][j_1][j_2] \dots [j_n][k]$, where $0 \le i \le m$, $0 \le j_1 \le p_1$, $0 \le j_2 \le p_2$, …, $0 \le j_n \le p_n$, and $k = r$, in array M by using loops embedded loops. During the visitation, if $M[i][j_1][j_2] \dots [j_n][k] >$ *maxLength*, then update and *lastIndexOnY1* and *maxLength* to and j_1 and $M[i][j_1][j_2] \dots [j_n][k]$ respectively.

Step 14. After finishing the visitation of all the cells of $M[i][j\_{1}][j\_{2}] \dots [j_n][k]$, where $0 \le i \le m$, $0 \le j_1 \le p_1$, $0 \le j_2 \le p_2$, …, $0 \le j_n \le p_n$, and $k = r$, we output the substring of Y_1 with starting index of *(lastIndexOnY1* **-** *maxLength)* and ending index of *lastIndexOnY1* and *maxLength*.

The combination of Claim 1, Claim 2, Claim 3, Claim 4, Claim 5, and Claim 6 in Section 2 ensures that the output string is a desired string and the output *maxLength* is the length of the desired string. It is clear that both time complexity and space complexity of the above algorithm are $O((m + 1)(p_1 + 1)(p_2 + 1) \dots (p_n + 1)(r + 1)) = O(m\, p_1\, p_2 \dots p_n\, r)$.

4. Conclusion

In this paper, we introduce a new problem called the constrained longest common subsequence and substring problem for multiple strings X, Y_1, $Y_{2, \dots,}$ Y_n and a constrained string P. We propose an algorithm with time complexity and space complexity of $O(|X||Y_1||Y_2| \dots |Y_n||P|)$ to solve the problem. In future, we will design new algorithms improving the time and space complexities and find the practical applications of our algorithm.

Acknowledgments. The authors would like to thank the referee for his or her comments on improving the initial manuscript.

Conflicts of interest. The authors declare no conflicts of interest.

Authors' contributions. All authors contributed equally to this work.

REFERENCES

1. A. Amir, P. Charalampopoulos, S. Pissis and J. Radoszewski, Dynamic and internal longest common substring, *Algorithmica*, 82 (2020) 3707-3743.
2. A. Apostolico, String editing and longest common subsequences, in: G. Rozenberg and A. Salomaa (Eds.), Linear Modeling: Background and Application, in: Handbook of Formal Languages, Vol. 2, Springer-Verlag, Berlin, 1997.
3. A. Apostolico, Chapter 13: General pattern matching, in: M. J. Atallah (Ed.), Handbook of Algorithms and Theory of Computation, CRC, Boca Raton, FL, 1998.
4. L. Bergroth, H. Hakonen and T. Raita, A survey of longest common subsequence algorithms, in: SPIRE, A Corua, Spain, 2000.
5. C. Blum, M. Djukanovic, A. Santini, H. Jiang, C. Li, F. Manyà, and G. R. Raidl, Solving longest common subsequence problems via a transformation to the maximum clique problem, *Computers and Operations Research*, 125 (2021) 105089.
6. F. Y. L. Chin, A. De Santis, A. L. Ferrara, N. L. Ho, and S. K. Kim, A simple algorithm for the constrained sequence problems, *Information Processing Letters*, 90 (2004) 175-179.
7. T. Cormen, C. Leiserson, and R. Rivest, Section 16.3: Longest common subsequence, Introduction to Algorithms, MIT Press, Cambridge, MA, 1990.
8. M. Crochemore, C. S. Iliopoulos, A. Langiu, and F. Mignosi, The longest common substring problem. Mathematical Structures in Computer Science, pp 1-19, Cambridge University Press 2015, doi:10.1017/S0960129515000110.
9. M. Djukanovic, G. Raidl, and C. Blum, Finding longest common subsequences: New anytime A* search results, *Applied Soft Computing*, 95 (2020) 106499.
10. D. Gusfield, II: Suffix Trees and Their Uses, Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology, Cambridge University Press, 1997.
11. D. Hirschberg, A linear space algorithm for computing maximal common subsequences, *Communications of the ACM*, 18 (1975) 341-343.
12. D. Hirschberg, Serial computations of Levenshtein distances, in: A. Apostolico and Z. Galil (Eds.), Pattern Matching Algorithms, Oxford University Press, Oxford, 1997.
13. J. Hunt and T. Szymanski, A fast algorithm for computing longest common subsequences, *Communications of the ACM*, 20 (1977) 350-353.
14. R. Li, J. Deka, and K. Deka, An algorithm for the longest common subsequence and substring problem, *Journal of Mathematics and Informatics*, 25 (2023) 77-81.
15. R. Li, J. Deka, K. Deka, and D. Li, An algorithm for the constrained longest common subsequence and substring problem, *Journal of Mathematics and Informatics*, 26 (2024) 41-48.
16. Y. Li, An efficient algorithm for LCS problem between two arbitrary sequences, Mathematical Problems in Engineering (2018), Article ID 4158071, https://doi.org/10.1155/2018/4158071.
17. S. R. Mousavi and F. Tabataba, An improved algorithm for the longest common subsequence problem, *Computers and Operations Research*, 39 (2012) 512-520.
18. C. Rick, New algorithms for the longest common subsequence problem, Research Report No. 85123-CS, University of Bonn, 1994.

19. Y. T. Tsai, The constrained longest common subsequence problem, *Information Processing Letters*, 88 (2003) 173-176.
20. P. Weiner, Linear pattern matching algorithms. In: 14th Annual Symposium on Switching and Automata Theory, Iowa City, Iowa, USA, October 15–17, 1973, pp. 1-11.