Journal of Mathematics and Informatics
Vol. 26, 2024, 41-48
ISSN: 2349-0632 (P), 2349-0640 (online)
Published 19 March 2024
www.researchmathsci.org
DOI:http://dx.doi.org/10.22457/jmi.v26a04240

Journal of
Mathematics and Informatics

An Algorithm for the Constrained Longest Common Subsequence and Substring Problem

***Rao Li*[1*], *Jyotishmoy Deka*[2], *Kaushik Deka*[3] and *Dorothy Li*[4]**

[1]Department of Computer Science, Engineering, and Mathematics
University of South Carolina Aiken, Aiken, SC 29801, USA
E-mail: raol@usca.edu
[2]Department of Electrical Engineering
Tezpur University, Tezpur, Assam 784028, India
E-mail: jyotishmoydeka62@gmail.com
[3]Department of Computer Science and Engineering
National Institute of Technology Silchar, Cachar, Assam 788010, India
E-mail: jagatdeka20@gmail.com
[4]12000 Market Street, Unit 63, Reston, VA 20190, USA
E-mail: dorothy.li1994@gmail.com
[*]Corresponding author

Received 25 January 2024; accepted 10 March 2024

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

Keywords: The longest common subsequence and substring, The constrained longest common subsequence and substring

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 consists of consecutive letters in S. The length of S, denoted |S|, is defined as the number of letters in S. The longest common subsequence (LCSSeq) problem for two strings is to find a longest string which is a 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-studied in the last several decades. More details on the studies for the first problem can be found in [1], [2], [4,6,7,8,9,11] and the second problem can be found in [3, 13].

Tsai [12] extended the longest common subsequence problem for two strings to the constrained longest common subsequence (CLCSSeq) problem for two strings and a constrained string. 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 string Z such that Z is a longest common subsequence for both X and Y and P is a subsequence of Z. Tsai [12] designed an $O(|X|^2 |Y|^2 |P|)$ time algorithm for the CLCSSeq problem for two strings X, Y, and a constrained string P. Chin et al. [5] improved Tsai's algorithm and designed an $O(|X| |Y| |P|)$ time algorithm for the CLCSSeq problem for two strings X, Y, and a constrained string P.

Motivated by LCSSeq and LCSStr problems, Li et al. [10] 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 [10].

Motivated by Tsai's extension of LCSSeq for two strings to CLCSSeq for two strings and a constrained string, we introduce the constrained longest common subsequence and substring problem for two strings and 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 string Z such that Z is a longest common 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. In this paper, we, using some ideas in [5], design an $O(|X| |Y| |P|)$ time algorithm for CLCSSeqSStr problem for two strings and a constrained string.

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 establishing the recursions, we need some notations as follows. For a given string $S = s_1 s_2 \ldots s_l$ over an alphabet $\sum$, the ith prefix of S is defined as $S_i = s_1 s_2 \ldots s_i$, where $1 \leq i \leq l$. Conventionally, S_0 is defined as an empty string. The l suffixes of S are the strings of $s_1 s_2 \ldots s_l$, $s_2 s_3 \ldots s_l$, ..., $s_{l-1} s_l$, and s_l. Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two strings and $P = p_1 p_2 \ldots p_r$ a constrained string. We define Z[i, j, k] as a string satisfying the following conditions, where $1 \leq i \leq m$, $1 \leq j \leq n$, and $1 \leq k \leq r$,

(1) it is a subsequence of X_i,
(2) it is a suffix of Y_j,
(3) it has P_k as a subsequence,
(4) under (1), (2) and (3), its length is as large as possible.

Claim 1. Suppose that $X_i = x_1 x_2 \ldots x_i$, $Y_j = y_1 y_2 \ldots y_j$, and $P = p_1 p_2 \ldots p_k$, where $1 \leq i \leq m$, $1 \leq j \leq n$, and $1 \leq k \leq r$. If $Z[i, j, k] = z_1 z_2 \ldots z_a$ is a string satisfying conditions (1), (2), (3), and (4) above. Then we have only the following possible cases and the statement in each case is true.

Case 1. $x_i = y_j = p_k$. We have $|Z[i, j, k]| = |Z[i - 1, j - 1, k - 1]| + 1$ in this case.

Case 2. $x_i = y_j \neq p_k$. We have $|Z[i, j, k]| = |Z[i - 1, j - 1, k]| + 1$ in this case.

Case 3. $x_i \neq y_j$, $x_i \neq p_k$, and $y_j = p_k$. We have $|Z[i, j, k]| = |Z[i - 1, j, k]|$ in this case.

Case 4. $x_i \neq y_j$, $x_i \neq p_k$, and $y_j \neq p_k$. We have $|Z[i, j, k]| = |Z[i - 1, j, k]|$ in this case.

Case 5. $x_i \neq y_j$, $x_i = p_k$, and $y_j \neq p_k$. This case does not happen.

Proof of Claim 1. The five cases can be figured out in the following way. Firstly, we have two cases of $x_i = y_j$ or $x_i \neq y_j$. When $x_i = y_j$, we just can have two possible cases of $x_i = y_j = p_k$ or $x_i = y_j \neq p_k$. When $x_i \neq y_j$, we just can have three possible cases of $x_i \neq p_k$ and $y_j = p_k$, $x_i \neq p_k$ and $y_j \neq p_k$, or $x_i = p_k$ and $y_j \neq p_k$. Next, we will prove the statements in the five cases.

Case 1. Since $Z[i, j, k] = z_1 z_2 \ldots z_a$ is a suffix of Y_j, we have that $z_a = y_j = x_i = p_k$. Let $W = w_1 w_2 \ldots w_b = Z[i - 1, j - 1, k - 1]$ be a string satisfying the following conditions,

(1) it is a subsequence of X_{i-1},
(2) it is a suffix of Y_{j-1},
(3) it has P_{k-1} as a subsequence,
(4) under (1), (2) and (3), its length is as large as possible.

Note that $z_1 z_2 \ldots z_{a-1}$ is a string which is a subsequence of X_{i-1}, a suffix of Y_{j-1}, and has P_{k-1} as a subsequence. By the definition of $W = w_1 w_2 \ldots w_b$, we have that $a - 1 \leq b$. Namely, $a \leq b + 1$.

Note that $w_1 w_2 \ldots w_b z_a$ is a string satisfying the following conditions,

- it is a subsequence of X_i,
- it is a suffix of Y_j,
- it has P_k as a subsequence.

By the definition of $Z[i, j, k] = z_1 z_2 \ldots z_a$, we have that $b + 1 \leq a$. Thus $a = b + 1$ and $|Z[i, j, k]| = |Z[i - 1, j - 1, k - 1]| + 1$.

Case 2. Since $Z[i, j, k] = z_1 z_2 \ldots z_a$ is a suffix of Y_j, we have that $z_a = y_j = x_i \neq p_k$. Let $U = u_1 u_2 \ldots u_c = Z[i - 1, j - 1, k]$ be a string satisfying the following conditions,

(1) it is a subsequence of X_{i-1},
(2) it is a suffix of Y_{j-1},
(3) it has P_k as a subsequence,
(4) under (1), (2) and (3), its length is as large as possible.

Note that $z_1 z_2 \ldots z_{a-1}$ is a string which is a subsequence of X_{i-1}, a suffix of Y_{j-1}, and has P_k as a subsequence. By the definition of $U = u_1 u_2 \ldots u_c = Z[i - 1, j - 1, k]$, we have that $a - 1 \leq c$. Namely, $a \leq c + 1$.

Note that $u_1 u_2 \ldots u_c$ is a string satisfying the following conditions,

- it is a subsequence of X_{i-1},
- it is a suffix of Y_{j-1},
- it has P_k as a subsequence.

Thus $u_1 u_2 \dots u_c y_j$ is a string which is a subsequence of X_i, a suffix of Y_j, and has P_k as a subsequence. By the definition of $Z[i, j, k] = z_1 z_2 \dots z_a$, we have that $c + 1 \le a$. Thus $a = c + 1$ and $|Z[i, j, k]| = |Z[i - 1, j - 1, k]| + 1$.

Case 3. Since $Z[i, j, k] = z_1 z_2 \dots z_a$ is a suffix of Y_j, we have that $z_a = y_j = p_k \neq x_i$. Let $V = v_1 v_2 \dots v_d = Z[i - 1, j, k]$ be a string satisfying the following conditions,

(1) it is a subsequence of X_{i-1},
(2) it is a suffix of Y_j,
(3) it has P_k as a subsequence,
(4) under (1), (2) and (3), its length is as large as possible.

Note that $z_1 z_2 \dots z_a$ is a string which is a subsequence of X_{i-1}, a suffix of Y_j, and has P_k as a subsequence. By the definition of $V = v_1 v_2 \dots v_d = Z[i - 1, j, k]$, we have that $a \le d$.

Note that $v_1 v_2 \dots v_d$ is a string satisfying conditions,

- it is a subsequence of X_{i-1},
- it is a suffix of Y_j,
- it has P_k as a subsequence.

Thus $v_1 v_2 \dots v_d$ is a string which is a subsequence of X_i, a suffix of Y_j, and has P_k as a subsequence. By the definition of $Z[i, j, k] = z_1 z_2 \dots z_a$, we have that $d \le a$. Thus $a = d$ and $|Z[i, j, k]| = |Z[i - 1, j, k]|$.

Case 4. Since $Z[i, j, k] = z_1 z_2 \dots z_a$ is a suffix of Y_j, we have that $z_a = y_j \neq p_k$, $z_a = y_j \neq x_i$, and $x_i \neq p_k$. Let $Q = q_1 q_2 \dots q_e = Z[i - 1, j, k]$ be a string satisfying the following conditions,

(1) it is a subsequence of X_{i-1},
(2) it is a suffix of Y_j,
(3) it has P_k as a subsequence,
(4) under (1), (2) and (3), its length is as large as possible.

Note that $z_1 z_2 \dots z_a$ is a string which is a subsequence of X_{i-1}, a suffix of Y_j, and has P_k as a subsequence. By the definition of $Q = q_1 q_2 \dots q_e = Z[i - 1, j, k]$, we have that $a \le e$.

Note that $q_1 q_2 \dots q_e$ is a string satisfying the following conditions,

- it is a subsequence of X_{i-1},
- it is a suffix of Y_j,
- it has P_k as a subsequence.

Thus $q_1 q_2 \dots q_e$ is a string which is a subsequence of X_i, a suffix of Y_j, and has P_k as a subsequence. By the definition of $Z[i, j, k] = z_1 z_2 \dots z_a$, we have that $e \le a$. Thus $a = e$ and $|Z[i, j, k]| = |Z[i - 1, j, k]|$.

Case 5. Since $Z[i, j, k] = z_1 z_2 \dots z_a$ is a suffix of Y_j, we have that $z_a = y_j \neq x_i = p_k$. Since $z_1 z_2 \dots z_a$ is a subsequence of X_i and $x_i \neq z_a$, we have that z_a appears before x_i on X_i. Since $x_i = p_k$ on X_i, $p_1 p_2 \dots p_k$ cannot be a subsequence of $z_1 z_2 \dots z_a$, a contradiction. Since this case does not happen, it is not necessary for us to deal with this case in our algorithm.

Therefore, the proof of Claim 1 is complete.

The following Claim 2 which will be used in our algorithm demonstrates the implications of the condition that there is not a string which is a subsequence of $X_i = x_1 x_2 \dots x_i$, a suffix of $Y_j = y_1 y_2 \dots y_j$, and has $P_k = p_1 p_2 \dots p_k$ as a subsequence.

Claim 2. Suppose there is not a string which is a subsequence of $X_i = x_1 x_2 \dots x_i$, a suffix of $Y_j = y_1 y_2 \dots y_j$, and has $P_k = p_1 p_2 \dots p_k$ as a subsequence.
[1]. If $x_i = y_j = p_k$, then there is not a string which is a subsequence of $X_{i-1} = x_1 x_2 \dots x_{i-1}$, a suffix of $Y_{j-1} = y_1 y_2 \dots y_{j-1}$, and has $P_{k-1} = p_1 p_2 \dots p_{k-1}$ as a subsequence.
[2]. If $x_i = y_j \neq p_k$, then there is not a string which is a subsequence of $X_{i-1} = x_1 x_2 \dots x_{i-1}$, a suffix of $Y_{j-1} = y_1 y_2 \dots y_{j-1}$, and has $P_k = p_1 p_2 \dots p_k$ as a subsequence.
[3]. If $x_i \neq y_j$, $x_i \neq p_k$, and $y_j = p_k$, then there is not a string which is a subsequence of $X_{i-1} = x_1 x_2 \dots x_{i-1}$, a suffix of $Y_j = y_1 y_2 \dots y_j$, and has $P_k = p_1 p_2 \dots p_k$ as a subsequence.
[4]. If $x_i \neq y_j$, $x_i \neq p_k$, and $y_j \neq p_k$, then there is not a string which is a subsequence for $X_{i-1} = x_1 x_2 \dots x_{i-1}$, a suffix of $Y_j = y_1 y_2 \dots y_j$, and has $P_k = p_1 p_2 \dots p_k$ as a subsequence.

Proof of Claim 2. We next will prove the statements in the four cases.
[1]. Now we have that $x_i = y_j = p_k$. Suppose, to the contrary, that there is a string W_1 which is a subsequence of $X_{i-1} = x_1 x_2 \dots x_{i-1}$, a suffix of $Y_{j-1} = y_1 y_2 \dots y_{j-1}$, and has $P_{k-1} = p_1 p_2 \dots p_{k-1}$ as a subsequence. Then $W_1 x_i$ is a string which is a subsequence of $X_i = x_1 x_2 \dots x_i$, a suffix of $Y_j = y_1 y_2 \dots y_j$, and has $P_k = p_1 p_2 \dots p_k$ as a subsequence, a contradiction.
[2]. Now we have that $x_i = y_j \neq p_k$. Suppose, to the contrary, that there is a string W_2 which is a subsequence of $X_{i-1} = x_1 x_2 \dots x_{i-1}$, a suffix of $Y_{j-1} = y_1 y_2 \dots y_{j-1}$, and has $P_k = p_1 p_2 \dots p_k$ as a subsequence. Then $W_2 x_i$ is a string which is a subsequence of $X_i = x_1 x_2 \dots x_i$, a suffix of $Y_j = y_1 y_2 \dots y_j$, and has $P_k = p_1 p_2 \dots p_k$ as a subsequence, a contradiction.
[3]. Now we have that $x_i \neq y_j$, $x_i \neq p_k$, and $y_j = p_k$. Suppose, to the contrary, that there is a string W_3 which is a subsequence for $X_{i-1} = x_1 x_2 \dots x_{i-1}$, a suffix of $Y_j = y_1 y_2 \dots y_j$, and has $P_k = p_1 p_2 \dots p_k$ as a subsequence. Then W_3 is a string which is a subsequence of $X_i = x_1 x_2 \dots x_i$, a suffix of $Y_j = y_1 y_2 \dots y_j$, and has $P_k = p_1 p_2 \dots p_k$ as a subsequence, a contradiction.
[4]. Now we have that $x_i \neq y_j$, $x_i \neq p_k$, and $y_j \neq p_k$. Suppose, to the contrary, that there is a string W_4 which is a subsequence of $X_{i-1} = x_1 x_2 \dots x_{i-1}$, a suffix of $Y_j = y_1 y_2 \dots y_j$, and has $P_k = p_1 p_2 \dots p_k$ as a subsequence. Then W_4 is a string which is a subsequence of $X_i = x_1 x_2 \dots x_i$, a suffix of $Y_j = y_1 y_2 \dots y_j$, and has $P_k = p_1 p_2 \dots p_k$ as a subsequence, a contradiction.

Therefore, the proof of Claim 2 is complete.

Our algorithm will use the following Claim 3 when we trace back to find the longest string which is a subsequence of X, a substring of Y, and has P as a subsequence.

Claim 3. Let $U^k = u_1^k u_2^k \dots u_{h(k)}^k$ be a longest string which is a subsequence of X, a substring of Y, and has P_k as a subsequence. Then $h(k) = \max\{|Z[i, j, k]| : 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq r\}$.
Proof of Claim 3. For each i with $1 \leq i \leq m$, each j with $1 \leq j \leq n$, and each k with $1 \leq k \leq r$, we, from the definition of Z[i, j, k], have that Z[i, j, k] is a subsequence of X, a substring of Y, and has P_k as a subsequence. By the definition of U^k, we have that $|Z[i, j, k]| \leq |U^k| = h(k)$. Thus $\max\{|Z[i, j, k]| : 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq r\} \leq h(k)$.

Since $U^k = u_1^k\, u_2^k \ldots u_{h(k)}^k$ is a string which is a subsequence of X, a substring of Y, and has P_k as a subsequence, there is an index s and an index t such that $u_{h(k)}^k = x_s$ and $u_{h(k)}^k = y_t$ such that $U^k = u_1^k\, u_2^k \ldots u_{h(k)}^k$ is a subsequence of X_s, a suffix of Y_t, and has P_k as a subsequence. From the definition of Z[i, j, k], we have that $h(k) \leq |Z[s, t, k]| \leq \max\{|Z[i, j, k]| : 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq r\}$.

Hence $h(k) = \max\{|Z[i, j, k]| : 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq r\}$ and the proof of Claim 3 is complete.

3. The algorithm

Now we can present our algorithm. We assume that $X = x_1 x_2 \ldots x_m$, $Y = y_1\, y_2 \ldots y_n$, and $P = p_1\, p_2 \ldots p_r$. Let M be a three-dimensional array of size (m + 1)(n + 1)(r + 1). It can be thought as a collection of (r + 1) two-dimensional arrays of size (m + 1)(n + 1). The cells M[i][j][k], where $0 \leq i \leq m$, $0 \leq j \leq n$, and $0 \leq k \leq r$, store the lengths of the longest strings such that each of them is a subsequence of X_i, a suffix of Y_j, and has P_k as a subsequence.

If either $i < k$ or $j < k$, there is not a string which is a subsequence of X_i, a suffix of Y_j, and has P_k as a subsequence. This situation is represented by setting $M[i][j][k] = -\infty$, where ∞ should be a larger number, for example, 100mnr. Our algorithm consists of the following steps. Firstly, we fill in the boundary cells in array M.

Step 1. If i = 0 and k = 0 or j = 0 and k = 0, the length of a string which is a subsequence of X_i, a suffix of Y_j, and has P_k as a subsequence is zero. Thus, M[0][j][0] = 0, where $0 \leq j \leq n$; M[i][0][0] = 0, where $0 \leq i \leq m$.

Step 2. If k = 0 or P_k is an empty string. The CLCSSeqSStr problem for two strings X and Y and a constrained string P becomes the LCSSeqSStr problem for two strings X and Y. The cells of M[i][j][0], where $1 \leq i \leq m$ and $1 \leq j \leq n$, can be filled in by the following rules. If $x_i = y_j$, then M[i][j][0] = M[i - 1][j - 1][0] + 1. If $x_i \neq y_j$, then M[i][j][0] = M[i - 1][j][0]. The detailed proofs for the truth of the rules can be found in [10].

Step 3. If i = 0 and $k \geq 1$, there is no string which is a subsequence of X_i, a suffix of Y_j, and has P_k as a subsequence. Thus, $M[0][j][k] = -\infty$, where $0 \leq j \leq n$ and $1 \leq k \leq r$.

Step 4. If j = 0 and $k \geq 1$, there is no string which is a subsequence of X_i, a suffix of Y_j, and has P_k as a subsequence. Thus, $M[i][0][k] = -\infty$, where $0 \leq i \leq m$ and $1 \leq k \leq r$.

Next, we will fill in the cells M[i][j][k], where $i \geq 1$, $j \geq 1$, and $k \geq 1$.

Step 5. If $i \geq 1$, $j \geq 1$, $k \geq 1$, and $x_i = y_j = p_k$, then M[i][j][k] = M[i - 1][j - 1][k - 1] + 1.

Step 6. If $i \geq 1$, $j \geq 1$, $k \geq 1$, and $x_i = y_j \neq p_k$, then M[i][j][k] = M[i - 1][j - 1][k] + 1.

Step 7. If $i \geq 1$, $j \geq 1$, $k \geq 1$, and $x_i \neq y_j$, $x_i \neq p_k$, and $y_j = p_k$, then M[i][j][k] = M[i - 1][j][k].

Step 8. If $i \geq 1$, $j \geq 1$, $k \geq 1$, and $x_i \neq y_j$, $x_i \neq p_k$, and $y_j \neq p_k$, then M[i][j][k] = M[i - 1][j][k].

Notice that Claim 3 implies that if a longest string which is a subsequence of $X = X_m$, a substring of $Y = Y_n$, and has $P = P_r$ as a subsequence exists then its length is equal to $\max\{|Z[i, j, r]| : 1 \leq i \leq m, 1 \leq j \leq n\} = \max\{M[i][j][r] : 1 \leq i \leq m, 1 \leq j \leq n\}$. Hence a longest string which is a subsequence of X, a substring of Y, and has P as a subsequence can be found in the following steps.

Step 9. Define one variable called *maxLength* which eventually represents the length of a longest string which is a subsequence of X, a substring of Y, and has P as a subsequence and its initial value is 0.

Step 10. Define another variable called *lastIndexOnY* which eventually represents the last index of the desired string which is a substring of Y and its initial value is n.

Step 11. Visit all the cells of M[i][j][r], where $0 \leq i \leq m$ and $0 \leq j \leq n$, in the last two-dimensional array created in the algorithm above by using a loop embedded in another loop. During the visitation, if M[i][j][r] > *maxLength*, then update *maxLength* and *lastIndexOnY* as M[i][j][r] and j, respectively.

Step 12. After finishing the visitation of all the cells of M[i][j][r], where $0 \leq i \leq m$ and $0 \leq j \leq n$, we return the substring of Y between (*lastIndexOnY* - *maxLength*) and *lastIndexOnY*.

From Claim 1, Claim 2, and Claim 3, we have the following theorem.

Theorem 1. The above algorithm is correct and both the time complexity and space complexity of the algorithm are $O((m + 1)(n + 1)(r + 1)) = O(m\,n\,r)$.

4. Conclusion

In this paper, we introduce a new problem called the constrained longest common subsequence and substring problem for two strings X, Y, and a constrained string P. We propose an algorithm with time complexity and space complexity of $O(|X|\,|Y|\,|P|)$ to solve the problem. In future, we will design new algorithms to improve the time and space complexities and find the applications of our algorithm in the real world.

Acknowledgements. The authors would like to thank the referee for his/her suggestions which led to the improvements of 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.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.
2. A.Apostolico, Chapter 13: General pattern matching, in: M. J. Atallah (Ed.), *Handbook of Algorithms and Theory of Computation*, CRC, Boca Raton, FL, 1998.
3. D.Gusfield, *II: Suffix Trees and Their Uses, Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, 1997.
4. L.Bergroth, H. Hakonen, and T. Raita, A survey of longest common subsequence algorithms, in: SPIRE, A Coruna, Spain, 2000.
5. 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.
6. T.Cormen, C. Leiserson, and R. Rivest, Section 16.3: Longest common subsequence, Introduction to Algorithms, MIT Press, Cambridge, MA, 1990.
7. D.Hirschberg, A linear space algorithm for computing maximal common subsequences, *Communications of the ACM*, 18 (1975) 341-343.
8. D.Hirschberg, Serial computations of Levenshtein distances, in: A. Apostolico and Z. Galil (Eds.), *Pattern Matching Algorithms*, Oxford University Press, Oxford, 1997.
9. J.Hunt and T. Szymanski, A fast algorithm for computing longest common subsequences, *Communications of the ACM*, 20 (1977) 350-353.
10. 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.
11. C.Rick, New algorithms for the longest common subsequence problem, Research Report No. 85123-CS, University of Bonn, 1994.
12. Y.T.Tsai, The constrained longest common subsequence problem, *Information Processing Letters*, 88 (2003) 173-176.
13. P.Weiner, Linear pattern matching algorithms. In: 14th Annual Symposium on Switching and Automata Theory, Iowa City, Iowa, USA, October 15–17, 1973, 1–11 (1973).