An Algorithm for the Constrained Longest Common Subsequence and Substring Problem

Rao Li
Dept. of Computer Science, Engineering, and Mathematics
University of South Carolina Aiken
Aiken, SC 29801
USA
Email: raol@usca.edu

Jyotishmoy Deka
Dept. of Electrical Engineering
Tezpur University
Tezpur, Assam 784028
India
Email: jyotishmoydeka62@gmail.com

Kaushik Deka
Dept. of Computer Science and Engineering
National Institute of Technology Silchar
Cachar, Assam 788010
India
Email: jagatdeka20@gmail.com

Dorothy Li
12000 Market Street, Unit 63
Reston, VA 20190
USA
Email: dorothy.li1994@gmail.com

Aug. 1, 2023

Abstract

Let Σ be an alphabet. For two strings X, Y, and a constrained string P over the alphabet Σ, 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: longest common subsequence, longest common substring, longest common subsequence and substring, constrained longest common subsequence

1. Introduction

Let Σ be an alphabet and S a string over Σ. A subsequence of a string S over an alphabet Σ is obatined 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 longest common subsequence problem (LCSSeq) 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 last several decades. More details on the studies for the first problem can be found in [1], [2], [4], [6], [7], [8], [9], and [11] and the second problem can be found in [3] and [13].

Tsai [12] extended the longest common subsequence problem for two strings to the constrained longest common subsequence (CLCSSeq) probelm 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 string Z such that Z is a longest common subsequence for 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, where $|X|$, $|Y|$, and $|P|$ denote the lengths of the strings X, Y, and P, respectively. 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 and 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, 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 to CLCSSeq for two strings, we introduce 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 string Z such that Z is a longest common subsequence of X, a substring of Y, and has P as a subsequence. Clearly, the CLCSSeq problem is a special CLCSSeqSStr problem with an empty constrained string. In this paper, we, using some ideas and techniques developed 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 $S = s_1s_2...s_l$ over an alphabet Σ, the size of S, denoted $|S|$, is defined as the number of letters in S. The i prefix of S is defined as $S_i = s_1s_2...s_i$, where $1 \leq i \leq l$. Conventionally, S_0 is defined as an empty string. The l suffixes of S are the strings of $s_1s_2...s_l$, $s_2s_3...s_l$, ..., $s_{l-1}s_l$, and s_l. Let $X = x_1x_2...x_m$ and $Y = y_1y_2...y_n$ be two strings and $P = p_1p_2...p_r$ 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. Let $U^k = u_1^k u_2^k ... u_{h_k}^k$ 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 1. 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

$\max\{\,|Z[i,j,k]| : 1 \le i \le m, 1 \le j \le n, 1 \le k \le r\,\} \le h_k$.

Since $U^k = u_1^k u_2^k ... u_{h_k}^k$ is a longest 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 $U^k = u_1^k u_2^k ... u_{h_k}^k$ 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 $h_k \le |Z[s,t,k]| \le \max\{\,|Z[i,j,k]| : 1 \le i \le m, 1 \le j \le n, 1 \le k \le r\,\}$.

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

Claim 2. Suppose that $X_i = x_1 x_2 ... x_i$, $Y_j = y_1 y_2 ... y_j$, and $P = p_1 p_2 ... p_k$, where $1 \le i \le m$ and $1 \le j \le n$, $1 \le k \le r$. If $Z[i,j,k] = z_1 z_2 ... z_a$ 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. $x_i = y_j \ne p_k$. We have $|Z[i,j,k]| = |Z[i-1,j-1,k]| + 1$ in this case.

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

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

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

Proof of Claim 2**.** 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 $Z[i, j, k] = z_1 z_2 ... z_a$ is a suffix of Y_j, we have that $z_a = y_j = x_i = p_k$. Let $W = w_1 w_2 ... w_b = Z[i-1, j-1, k-1]$ be 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-1} as a subsequence,
- under (1), (2) and (3), its length is as large as possible.

Note that $z_1 z_2 ... z_{a-1}$ 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 $W = w_1 w_2 ... w_b$, we have that $a-1 \leq b$. Namely, $a \leq b+1$.

Note that $w_1 w_2 ... w_b z_a$ is a string satisfying 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 $Z[i, j, k] = z_1 z_2 ... z_a$, 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 $Z[i, j, k] = z_1 z_2 ... z_a$ is a suffix of Y_j, we have that $z_a = y_j = x_i \neq p_k$. Let $U = u_1 u_2 ... u_c = Z[i-1, j-1, k]$ be 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 subsequenc,
- under (1), (2) and (3), its length is as large as possible.

Note that $z_1z_2...z_{a-1}$ 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 $U = u_1u_2...u_c = Z[i-1, j-1, k]$, we have that $a-1 \leq c$. Namely, $a \leq c+1$.

Note that $u_1u_2...u_c$ 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 $u_1u_2...u_cy_j$ 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 $Z[i,j,k] = z_1z_2...z_a$, we have that $c+1 \leq a$. Thus $a = c+1$ and $|Z[i,j,k]| = |Z[i-1, j-1, k]| + 1$.

Case 3. Since $Z[i,j,k] = z_1z_2...z_a$ is a suffix of Y_j, we have that $z_a = y_j = p_k \neq x_i$. Let $V = v_1v_2...v_d = Z[i-1, j, k]$ be 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,
- under (1), (2) and (3), its length is as large as possible.

Note that $z_1z_2...z_a$ 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 $V = v_1v_2...v_d = Z[i-1, j, k]$, we have that $a \leq d$.

Note that $v_1v_2...v_d$ 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 $v_1v_2...v_d$ 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 $Z[i,j,k] = z_1z_2...z_a$, we have that $d \leq a$. Thus $a = d$ and $|Z[i,j,k]| = |Z[i-1,j,k]|$.

Case 4. Since $Z[i,j,k] = z_1z_2...z_a$ 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 $Q = q_1q_2...q_e = Z[i-1,j,k]$ be 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,
- under (1), (2) and (3), its length is as large as possible.

Note that $z_1z_2...z_a$ 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 $Q = q_1q_2...q_e = Z[i-1,j,k]$, we have that $a \leq e$.

Note that $q_1q_2...q_e$ 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 $q_1q_2...q_e$ 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 $Z[i,j,k] = z_1z_2...z_a$, we have that $e \leq a$. Thus $a = e$ and $|Z[i,j,k]| = |Z[i-1,j,k]|$.

Case 5. Since $Z[i,j,k] = z_1z_2...z_a$ is a suffix of Y_j, we have that $z_a = y_j \neq x_i = p_k$. Since $z_1z_2...z_a$ 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, $p_1p_2...p_k$ cannot be a subsequence of $z_1z_2...z_a$, a conradiction. Note that since this case does not happen, we will not deal with this case in our algorithm.

Therefore the proof of Claim 2 is complete.

The following Claim 3 which will used in our algorithm demonstrates the implications of the condition that there is not a string which is a subsequence of $X_i = x_1x_2...x_i$, a suffix of $Y_j = y_1y_2...y_j$, and has $P_k = p_1p_2...p_k$ as a subsequence.

Claim 3. Suppose there is not a string which is a subsequence of $X_i = x_1x_2...x_i$, a suffix of $Y_j = y_1y_2...y_j$, and has $P_k = p_1p_2...p_k$ as a subsequence.

[1]. If $x_i = y_j = p_k$, then there is not a string which is a subsequence of $X_{i-1} = x_1x_2...x_{i-1}$, a suffix of $Y_{j-1} = y_1y_2...y_{j-1}$, and has $P_{k-1} = p_1p_2...p_{k-1}$ as a subsequence.

[2]. If $x_i = y_j \neq p_k$, then there is not a string which is a subsequence of $X_{i-1} = x_1x_2...x_{i-1}$, a suffix of $Y_{j-1} = y_1y_2...y_{j-1}$, and has $P_k = p_1p_2...p_k$ 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 $X_{i-1} = x_1x_2...x_{i-1}$, a suffix of $Y_j = y_1y_2...y_j$, and has $P_k = p_1p_2...p_k$ 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 $X_{i-1} = x_1x_2...x_{i-1}$, a suffix of $Y_j = y_1y_2...y_j$, and has $P_k = p_1p_2...p_k$ as a subsequence.

Proof of Claim 3. We next will prove the statements in the four cases.

[1]. Now we have that $x_i = y_j = p_k$. Suppose, to the con-

trary, that there is a string W_1 which is a subsequence of $X_{i-1} = x_1x_2...x_{i-1}$, a suffix of $Y_{j-1} = y_1y_2...y_{j-1}$, and has $P = p_1p_2...p_{k-1}$ as a subsequence. Then W_1x_i is a string which is a subsequence of $X_i = x_1x_2...x_i$, a suffix of $Y_j = y_1y_2...y_j$, and has $P_k = p_1p_2...p_k$ 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 $X_{i-1} = x_1x_2...x_{i-1}$, a suffix of $Y = y_1y_2...y_{j-1}$, and has $P_k = p_1p_2...p_k$ as a subsequence. Then W_2x_i is a string which is a subsequence of $X_i = x_1x_2...x_i$, a suffix of $Y_j = y_1y_2...y_j$, and has $P_k = p_1p_2...p_k$ 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 $X_{i-1} = x_1x_2...x_{i-1}$, a suffix of $Y_j = y_1y_2...y_j$, and has $P_k = p_1p_2...p_k$ as a subsequence. Then W_3 is a string which is a subsequence of $X_i = x_1x_2...x_i$, a suffix of $Y_j = y_1y_2...y_j$, and has $P_k = p_1p_2...p_k$ 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 $X_{i-1} = x_1x_2...x_{i-1}$, a suffix of $Y_j = y_1y_2...y_j$, and has $P_k = p_1p_2...p_k$ as a subsequence. Then W_4 is a string which is a subsequence of $X_i = x_1x_2...x_i$, a suffix of $Y = y_1y_2...y_j$, and has $P_k = p_1p_2...p_k$ as a subsequence, a contradiction.

Therefore the proof of Claim 3 is complete.

3. The Algorithm

Now we can present our algorithm. We assume that $X = x_1x_2...x_m$, $Y = y_1y_2...y_n$, and $P = p_1p_2...p_r$. 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 $0 \le i \le m$, $0 \le j \le n$, and $0 \le k \le r$, store the lengths of 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 $i < r$ or $j < r$, 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 ∞ can be any number which is greater than the larger one between m and n. Now we can fill in some boundary cells in array M.

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 $0 \le j \le n$ and $M[i][0][0] = 0$, where $0 \le i \le m$.

If $k = 0$ or P 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 $1 \le i \le m$ and $1 \le j \le n$, can be filled in by the following rules. If $x_i = y_j$, then $M[i][j] = M[i-1][j-1]+1$. If $x_i \ne y_j$, then $M[i][j] = M[i-1][j]$. The detailed proofs for the truth of the rules can be found in [10].

If $i = 0$ and $k \ge 1$, there is not a 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 $0 \le j \le n$ and $1 \le k \le r$.

If $j = 0$ and $k \ge 1$, there is not a string which is a subsequence of X_i, a suffix of Y_j, and has P as a subsequence. Thus $M[i][0][k] = -\infty$, where $0 \le i \le m$ and $1 \le k \le r$.

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

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

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

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 $M[i][j][k] = M[i-1][j][k]$.

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 1 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 way. 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. 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. 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 another loop. During the visitation, if $M[i][j][r] > maxLength$, then update $maxLength$ and $lastIndexOnY$ as $M[i][j][r]$ and j, respectively. 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 $lastIndexOnY$.

The correctness of the above algorithm is ensured by Claim 1, Claim 2, and Claim 3. It is clear that both time complexity and space complexity of the above algorithm are $O((m+1)(n+1)(r+1)) = O(mnr)$.

We implemented our algorithm in Java and the program can be found at “https://sciences.usca.edu/math/˜mathdept/rli/CLCSSeqSStr/CLCSubseqSubstr.pdf”.

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 Coruña, 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) 341343.

[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) 350353.

[10] R. Li, J. Deka, and K. Deka, An algorithm for the longest common subsequence and substring problem, manuscript, July 2023. The implementation of the algorithm in Java can be found at "https://sciences.usca.edu/math/˜mathdept /rli/LCSSeqSStr/LCSS.pdf".

[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 1517, 1973, pp. 111 (1973).