Journal of Mathematics and Informatics
Vol. 25, 2023, 77-81
ISSN: 2349-0632 (P), 2349-0640 (online)
Published 3 September 2023
www.researchmathsci.org
DOI: http://dx.doi.org/10.22457/jmi.v25a08231

Journal of
Mathematics and Informatics

An Algorithm for the Longest Common Subsequence and Substring Problem

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

[1]Department of Computer Science, Engineering, and Mathematics
University of South Carolina Aiken, Aiken, SC 29801, USA
E-mail: raol@usca.edu
[2]Departtment 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

Received 16 July 2023; accepted 2 September 2023

Abstract. In this note, we first introduce a new problem called the longest common subsequence and substring problem. Let X and Y be two strings over an alphabet ∑. The longest common subsequence and substring problem for X and Y is to find the longest string, which is a subsequence of X and a substring of Y. We propose an algorithm to solve the problem.

Keywords: Algorithm, the longest common subsequence, the longest common 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 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. Let X and Y be two strings over an alphabet. The longest common subsequence problem for X and Y is to find the longest string, which is a subsequence of both X and Y. The longest common substring problem for X and Y is to find the longest string, which is a substring of both X and Y. Both the longest common subsequence problem and the longest common substring problem have been well-studied in the last several decades. They have applications in different fields, for example, in molecular biology, the lengths of the longest common subsequence and the longest common substring are the suitable measurements for the similarity between two biological sequences. More details on the algorithms for the first problem can be found in [1], [2], [4], [5], [7], and [8] and the second problem can be found in [3] and [9]. Motivated by the two problems above, we introduce a new problem called the longest common subsequence and substring problem. The longest common subsequence and substring problem for X and Y is to find the longest string, which is a subsequence of X and a substring of Y. In this note, we propose an algorithm to solve this problem.

2. The foundations of the algorithm

In order to present our algorithm, we need to prove some facts which are the foundations for our algorithm. Before proving the facts, we need some notations as follows. For a given string $S = s_1 s_2 \ldots s_l$ over an alphabet $\sum$, the size of S, denoted $|S|$, is defined as the number of letters in S. 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. We define Z[i, j] as a string satisfying the following conditions, where $1 \leq i \leq m$ and $1 \leq j \leq n$.

(1) It is a subsequence of X_i.
(2) It is a suffix of Y_j.
(3) Under (1) and (2), its length is as large as possible.

Fact 1. Let $U = u_1 u_2 \ldots u_r$ be a longest string which is a subsequence of X and substring of Y. Then $r = \max\{|Z[i, j]| : 1 \leq i \leq m, 1 \leq j \leq n\}$.

Proof of Fact 1. For each i with $1 \leq i \leq m$ and each j with $1 \leq j \leq n$, we, from the definition of Z[i, j], have that Z[i, j] is a subsequence of X and substring of Y. By the definition of U, we have that $|Z[i, j]| \leq |U| = r$. Thus $\max\{|Z[i, j]| : 1 \leq i \leq m, 1 \leq j \leq n\} \leq r$.

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

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

Fact 2. Suppose that $X_i = x_1 x_2 \ldots x_i$ and $Y_j = y_1 y_2 \ldots y_j$, where $1 \leq i \leq m$ and $1 \leq j \leq n$. If $Z[i, j] = z_1 z_2 \ldots z_a$ is a string satisfying conditions (1), (2), and (3) above. Then we have

[1]. If $x_i = y_j$, then a = 1 + the length of a longest string which is a subsequence of X_{i-1} and a suffix of Y_{j-1}.

[2]. If $x_i \neq y_j$, then a = the length of the longest string which is a subsequence of X_{i-1} and a suffix of Y_j.

Proof of [1] in Fact 2. Suppose $W = w_1 w_2 \ldots w_b$ is a string satisfying the following conditions.

(i) It is a subsequence of X_{i-1}.
(ii) It is a suffix of Y_{j-1}.
(iii) Under (i) and (ii), its length is as large as possible.

Since $W = w_1 w_2 \ldots w_b$ is a subsequence of X_{i-1}, a suffix of Y_{j-1}, and $x_i = y_j$, $W = w_1 w_2 \ldots w_b x_i$ is a subsequence of X_i and a suffix of Y_j. From the definition of Z[i, j], we have $|W| + 1 = b + 1 \leq |Z[i, j]| = a$.

Since $Z[i, j] = z_1 z_2 \ldots z_a$ is a string satisfying conditions (i), (ii), and (iii) above, we have that $z_a = y_j = x_i$. We further have that $z_1 z_2 \ldots z_{a-1}$ is a string which is a subsequence of X_{i-1} and a suffix of Y_{j-1}. From the definition of $W = w_1 w_2 \ldots w_b$, we have that $a - 1 \leq b$. Thus $a = 1 + b$ and $a = 1 +$ the length of the longest string, which is a subsequence of X_{i-1} and a suffix of Y_{j-1}.

Proof of [2] in Fact 2. Suppose $U = u_1 u_2 \ldots u_c$ is a string satisfying the following conditions.

(α) It is a subsequence of X_{i-1}.
(β) It is a suffix of Y_j.
(γ) Under (α) and (β), its length is as large as possible.

Since $U = u_1 u_2 \ldots u_c$ is a subsequence of X_{i-1} and a suffix of Y_j, $U = u_1 u_2 \ldots u_c$ is a subsequence of X_i and a suffix of Y_j. By the definition of Z[i, j], we have $|U| = c \leq |Z[i, j]| = a$.

Since $Z[i, j] = z_1 z_2 \ldots z_a$ is a string satisfying conditions (1), (2), and (3) above, we have that $z_a = y_j \neq x_i$. Thus $z_1 z_2 \ldots z_a$ is a string that is a subsequence of X_{i-1} and a suffix of Y_j. From the definition of $U = u_1 u_2 \ldots u_c$, we have that $a \leq c$. Thus $a = c$ and $a =$ the length of the longest string, which is a subsequence of X_{i-1} and a suffix of Y_j. Hence, the proof of Fact 2 is complete.

3. An algorithm for the longest common subsequence and substring problem

Based on Fact 1 and Fact 2 in Section 2, we can design an algorithm for the longest common subsequence and substring problem. Once again, we assume that $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ are two strings over an alphabet ∑. In the following Algorithm A, W is a two-dimensional array of size $(m + 1) \times (n + 1)$ and the cells W(i, j), where $1 \leq i \leq m$ and $1 \leq j \leq n$, store the lengths of strings such that each of them satisfies the following conditions.

(1) It is a subsequence of X_i.
(2) It is a suffix of Y_j.
(3) Under (1) and (2), its length is as large as possible.

ALG A (X, Y, m, n, W)
1. Initialization: W(i, 0) $\Longleftarrow$ 0, where i = 0, 1, ..., m
   W(0, j) $\Longleftarrow$ 0, where j = 0, 1, ..., n
   maxLength = 0
   lastIndexOnY = n
2. **for** i $\Longleftarrow$ 1 to m
3. **for** j $\Longleftarrow$ 1 to n
   if $x_i = y_j$ W(i, j) $\Longleftarrow$ W(i - 1, j - 1) + 1
   else W(i, j) $\Longleftarrow$ W(i - 1, j)
   if W(i, j) > *maxLength*
   maxLength = W(i, j)
   lastIndexOnY = j
4. **return** A substring of Y from (*lastIndexOnY* – *maxLength* + 1) to *lastIndexOnY*

Because of Fact 1 and Fact 2 in Section 2, Algorithm A is correct. Obviously, the time complexity of Algorithm A is O(mn) and the space complexity of Algorithm A is also O(mn). We implemented Algorithm A in Java and the program can be found at "https://sciences.usca.edu/math/˜mathdept/rli/LCSSeqSStr/LCSS.pdf".

Below is an example that illustrates Algorithm A above. Suppose X = abuvbc and Y = dabca. Then the two-dimensional array W in Algorithm A is computed as follows.

	Y	d	a	b	c	a
X	0	0	0	0	0	0
a	0	0	1	0	0	1
b	0	0	1	2	0	1
u	0	0	1	2	0	1
v	0	0	1	2	0	1
b	0	0	1	2	0	1
c	0	0	1	2	3	1

Fig. 1. The two-dimensional array W computed in Algorithm A

Also, Algorithm A yields *maxLengt*h = 3, *lastIndexOnY* = 4, and outputs a string of abc, the longest string that is a subsequence of X = abuvbc and a substring of Y = dabca.

4. Conclusion

In this note, we introduce a new problem called the longest common subsequence and substring problem for two strings X and Y. Even though we can design an algorithm with time and space complexities of O(|X||Y|) to solve the problem, we plan to 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. There are no conflicts of interest among the authors.

Authors' contributions. All the authors contributed equally.

REFERENCES

1. A.Apostolico, String editing and longest common subsequences, in: G. Rozenberg and A. Salomaa (Eds.), Handbook of Formal Languages, Vol. 2, Linear Modeling: Background and Application, Springer-Verlag, Berlin, 1997, 361-398.
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 Coruňa, Spain, 2000.
5. T.Cormen, C.Leiserson, R.Rivest and C.Stein, Section 15.4: Longest common subsequence, Introduction to Algorithms (second edition), MIT Press, Cambridge, MA, 2001.

6. D.Hirschberg, A linear space algorithm for computing maximal common subsequences, *Comm. ACM*, 18 (1975) 341-343.
7. D.Hirschberg, Serial computations of Levenshtein distances, in: A. Apostolico and
8. Z.Galil (Eds.), Pattern Matching Algorithms, Oxford University Press, Oxford, 1997.
9. C.Rick, New algorithms for the longest common subsequence problem, Research Report No. 85123-CS, University of Bonn, 1994.
10. 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.