Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT

Information Processing Letters 88 (2003) 173–176

Information Processing Letters

www.elsevier.com/locate/ipl

The constrained longest common subsequence problem

Yin-Te Tsai

Department of Computer Science and Information Management, Providence University, 200 Chung Chi Road, Shalu, Taichung Hsien 433, Taiwan, R.O.C.

Received 23 October 2002; received in revised form 10 July 2003

Communicated by F.Y.L. Chin

Abstract

This paper considers a constrained version of longest common subsequence problem for two strings. Given strings S_1, S_2 and P, the constrained longest common subsequence problem for S_1 and S_2 with respect to P is to find a longest common subsequence *lcs* of S_1 and S_2 such that P is a subsequence of this *lcs*. An $O(rn^2m^2)$ time algorithm based upon the dynamic programming technique is proposed for this new problem, where n, m and r are lengths of S_1, S_2 and P, respectively.

Keywords: Constrained longest common subsequence problems; Algorithms; Dynamic programming

1. Introduction

A string is a sequence of symbols over an alphabet set Σ. A subsequence of a string s is obtained by deleting zero or more symbols from s. The longest common subsequence (LCS) problem for strings is to find a common subsequence having maximum length. For example, if $S_1 =$ `abcacba` and $S_2 =$ `aabbccbbaa`, `abccba` is a LCS for these two strings. This problem has many important applications in data compression, file comparison, and pattern recognition. In molecular biology, LCS is an appropriate measure of the similarity of biological sequences. When we want to know how homologous those DNA or protein sequences are, we can calculate the maximum number of identical symbols among them. That is exactly an LCS of them.

The LCS problem on multiple strings was shown to be NP-hard [6] (even on a binary alphabet). However, the LCS problem on two strings is polynomial-time solvable and has received much attention. Many authors have designed algorithms using the dynamic programming technique on this problem [8,4,7]. You may get several surveys for this problem from [5,1–3].

Suppose we want to compute an LCS for the similarity of S_1 and S_2 as shown in Fig. 1. We may say that the similarity of them is 15, because an LCS of S_1 and S_2 is `abcabcabcabcabc` of length 15. However, this LCS is not satisfactory if we know that subsequence `def` appears in both strings and this subsequence should be considered for the similarity measurement. In this case, both `dhejifabcabc` and `gdejifabcabc` of length

E-mail address: yttsai@pu.edu.tw (Y.-T. Tsai).

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2003.07.001

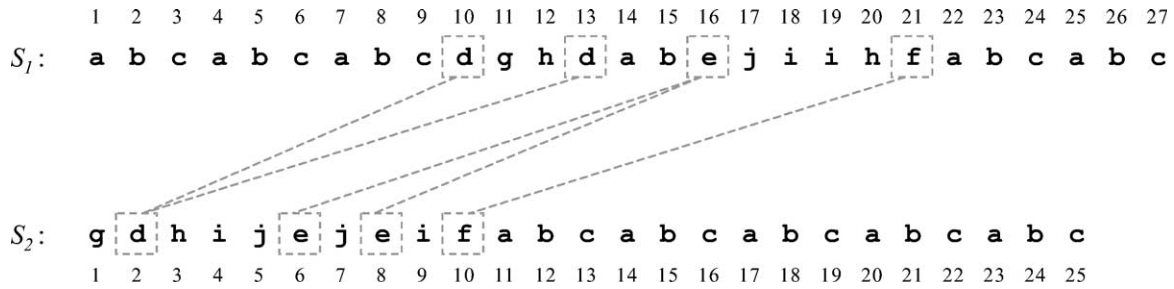

Fig. 1. An example for S_1 and S_2.

12 are LCSs under this constraint, and the similarity you concern becomes 12. Such a problem could arise in computing the homology of two biological sequences which have a specific or putative structure in common.

This paper considers a new problem of finding an LCS with a requested pattern for two strings. Given strings S_1, S_2 and P, the constrained LCS problem for S_1 and S_2 with respect to P is to find a longest common subsequence *lcs* of S_1 and S_2 such that P is a subsequence of this *lcs*. For example, if S_1 and S_2 are as shown in Fig. 1, both `dhejifabcabc` and `gdejifabcabc` are constrained LCSs for S_1 and S_2 with respect to $P = \texttt{def}$. The LCS problem for strings S_1 and S_2 on Σ can be reduced to a constrained LCS problem for strings $S_1' = \$ + S_1$ and $S_2' = \$ + S_2$ with respect to $P = \$$, where $\$ \notin \Sigma$. For example, the LCS problem for $S_1 = \texttt{abcacba}$ and $S_2 = \texttt{aabbccbbaa}$ could be reduced to the constrained LCS problem for strings $S_1' = \$\texttt{abcacba}$ and $S_2' = \$\texttt{aabbccbbaa}$ with respect to $P = \$$. The lower bound of time complexity of the LCS problem is also a lower bound of the constrained LCS problem. In this paper, we propose an algorithm based upon the dynamic programming technique with $O(rn^2m^2)$ time for this new problem, where n, m and r are lengths of S_1, S_2 and P, respectively.

The rest of this paper is organized as follows: Section 2 describes our algorithm and give the time complexity analysis for the algorithm. Finally some future research directions are provided in Section 3.

2. The algorithm

Let $S[x..y]$ denote the substring of string S from positions x to y if $x \leqslant y$, and an empty string otherwise. Let $S[x]$ be the character at position x in string S. Let $L(x, y, x', y')$ be the length of LCS of strings $S_1[x..x']$ and $S_2[y..y']$ if $1 \leqslant x \leqslant x' \leqslant n$ and $1 \leqslant y \leqslant y' \leqslant m$, and 0 otherwise. For $1 \leqslant k \leqslant r$, $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$, let $L_k(i, j)$ be the length of constrained LCS of strings $S_1[1..i]$ and $S_2[1..j]$ with respect to $P[1..k]$ if $S_1[i] = S_2[j] = P[k]$, and $-\infty$ otherwise. For $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$, it is easy to know that $L_1(i, j) = L(1, 1, i-1, j-1) + 1$ if $S_1[i] = S_2[j] = P[1]$, and $-\infty$ otherwise. Then we have the following result.

Lemma 1. *For* $2 \leqslant k \leqslant r$, $1 \leqslant i \leqslant n$ *and* $1 \leqslant j \leqslant m$,

$$L_k(i, j) = \begin{cases} \max_{1 \leqslant x < i, 1 \leqslant y < j} \{L_{k-1}(x, y) + L(x+1, y+1, i-1, j-1) + 1\} & \text{if } S_1[i] = S_2[j] = P[k]; \\ -\infty & \text{otherwise}. \end{cases}$$

Proof. Suppose that $S_1[i] = S_2[j] = P[k]$. Assume that $k > 1$. Let x and y be such that $S_1[x] = S_2[y] = P[k-1]$ where $1 \leqslant x < i$ and $1 \leqslant y < j$. Obviously, there is a constrained common subsequence of $S_1[1..i]$ and $S_2[1..j]$ with length $L_{k-1}(x, y) + L(x+1, y+1, i-1, j-1) + 1$. Since $L_k(i, j)$ is the longest length of constrained common subsequence for $S_1[1..i]$ and $S_2[1..j]$ with respect to $P[1..k]$, we have

$$L_k(i, j) \geqslant \max_{1 \leqslant x < i, 1 \leqslant y < j} \{L_{k-1}(x, y) + L(x+1, y+1, i-1, j-1) + 1\}.$$

Assume $L_k(i, j) > \max_{1 \leqslant x < i, 1 \leqslant y < j}\{L_{k-1}(x, y) + L(x + 1, y + 1, i - 1, j - 1) + 1\}$. Let x' and y' be such that $S_1[x']$, $S_2[y']$ and $P[k - 1]$ are identical and aligned together in an optimal solution $\mathcal{L}$ with length $L_k(i, j)$, where $1 \leqslant x' < i$ and $1 \leqslant y' < j$. Then $L_k(i, j) = L' + L'' + 1$, where L' denote the length of constrained LCS of $S_1[1..x']$ and $S_2[1..y']$ in $\mathcal{L}$ and L'' denote the length of LCS of $S_1[x' + 1..i - 1]$ and $S_2[y' + 1..j - 1]$ in $\mathcal{L}$. So we have the following inequality by the assumption

$$L' + L'' + 1 > \max_{1 \leqslant x < i, 1 \leqslant y < j}\bigl\{L_{k-1}(x, y) + L(x + 1, y + 1, i - 1, j - 1) + 1\bigr\}. \tag{1}$$

By the definitions of functions L_{k-1} and L, we have $L_{k-1}(x', y') \geqslant L'$ and $L(x' + 1, y' + 1, i - 1, j - 1) \geqslant L''$. Then the following inequality can be derived from inequality (1):

$$\begin{aligned} &L_{k-1}(x', y') + L(x' + 1, y' + 1, i - 1, j - 1) + 1 \\ &\quad > \max_{1 \leqslant x < i, 1 \leqslant y < j}\bigl\{L_{k-1}(x, y) + L(x + 1, y + 1, i - 1, j - 1) + 1\bigr\}. \end{aligned} \tag{2}$$

Inequality (2) implies that $L_{k-1}(x', y') + L(x' + 1, y' + 1, i - 1, j - 1) + 1$ is larger than the maximum value of $L_{k-1}(x, y) + L(x + 1, y + 1, i - 1, j - 1) + 1$, which is a contradiction. Therefore, the following inequality holds:

$$L_k(i, j) \leqslant \max_{1 \leqslant x < i, 1 \leqslant y < j}\bigl\{L_{k-1}(x, y) + L(x + 1, y + 1, i - 1, j - 1) + 1\bigr\}. \qquad \square$$

Lemma 2. *The length of constrained LCS lcs for strings S_1 and S_2 with respect to string P is $|lcs| = \max_{1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m}\{L_r(i, j) + L(i + 1, j + 1, n, m)\}$, where n, m and r are lengths of S_1, S_2 and P, respectively.*

Proof. Assume S_1, S_2 and P are strings over an alphabet set Σ. Let $S_1' = S_1 + \$$, $S_2' = S_2 + \$$ and $P' = P + \$$, where $\$ \notin \Sigma$. Let $L'(x, y, x', y')$ be the length of LCS of strings $S_1'[x..x']$ and $S_2'[y..y']$ if $1 \leqslant x \leqslant x' \leqslant n + 1$ and $1 \leqslant y \leqslant y' \leqslant m + 1$, and 0 otherwise. For $1 \leqslant k \leqslant r + 1$, $1 \leqslant i \leqslant n + 1$ and $1 \leqslant j \leqslant m + 1$, let $L_k'(i, j)$ be the length of constrained LCS of strings $S_1'[1..i]$ and $S_2'[1..j]$ with respect to $P'[1..k]$ if $S_1'[i] = S_2'[j] = P'[k]$, and $-\infty$ otherwise. By the result of Lemma 1, we easily have the following equation for $2 \leqslant k \leqslant r + 1$, $1 \leqslant i \leqslant n + 1$ and $1 \leqslant j \leqslant m + 1$:

$$L_k'(i, j) = \begin{cases} \max_{1 \leqslant x < i, 1 \leqslant y < j}\bigl\{L_{k-1}'(x, y) + L'(x + 1, y + 1, i - 1, j - 1) + 1\bigr\} & \text{if } S_1'[i] = S_2'[j] = P'[k]; \\ -\infty & \text{otherwise.} \end{cases}$$

Let lcs' denote a constrained LCS for S_1' and S_2' with respect to P'. It is easy to know $|lcs| = |lcs'| - 1$. Since $S_1'[n + 1] = S_2'[m + 1] = P'[r + 1] = \$$, we have

$$|lcs'| = L_{r+1}'(n + 1, m + 1) = \max_{1 \leqslant x < n+1, 1 \leqslant y < m+1}\bigl\{L_r'(x, y) + L'(x + 1, y + 1, n, m) + 1\bigr\}.$$

Since $S_1[1..n] = S_1'[1..n]$, $S_2[1..m] = S_2'[1..m]$ and $P[1..r] = P'[1..r]$, we have $L_r'(x, y) = L_r(x, y)$ and $L'(x + 1, y + 1, n, m) = L(x + 1, y + 1, n, m)$ for $1 \leqslant x \leqslant n$ and $1 \leqslant y \leqslant m$. Then the above equation can be rewritten as

$$|lcs'| = \max_{1 \leqslant x \leqslant n, 1 \leqslant y \leqslant m}\bigl\{L_r(x, y) + L(x + 1, y + 1, i - 1, j - 1) + 1\bigr\}.$$

Finally, we conclude that

$$|lcs| = |lcs'| - 1 = \max_{1 \leqslant x \leqslant n, 1 \leqslant y \leqslant m}\bigl\{L_r(x, y) + L(x + 1, y + 1, i - 1, j - 1)\bigr\}. \qquad \square$$

In the following, we show the values of functions L_1, L_2 and L_3 to compute the length of constrained LCS *lcs* for two strings given in Fig. 1 with respect to $P =$ `def`.

(1) $L_1(i, j) = -\infty$ for $1 \leqslant i \leqslant 27$ and $1 \leqslant j \leqslant 25$, except that $L_1(10, 2) = L(1, 1, 9, 1) + 1 = 1$ and $L_1(13, 2) = L(1, 1, 12, 1) + 1 = 2$.

(2) $L_2(i, j) = -\infty$ for $1 \leqslant i \leqslant 27$ and $1 \leqslant j \leqslant 25$, except that $L_2(16, 6) = \max\{L_1(10, 2) + L(11, 3, 15, 5), L_1(13, 2) + L(14, 3, 15, 5)\} + 1 = 3$, and $L_2(16, 8) = \max\{L_1(10, 2) + L(11, 3, 15, 7), L_1(13, 2) + L(14, 3, 15, 7)\} + 1 = 3$.
(3) $L_3(i, j) = -\infty$ for $1 \leqslant i \leqslant 27$ and $1 \leqslant j \leqslant 25$, except that $L_3(21, 10) = \max\{L_2(16, 6) + L(17, 7, 20, 9), L_2(16, 8) + L(17, 9, 20, 9)\} + 1 = 6$.
(4) $|lcs| = \max\{L_3(21, 10) + L(22, 11, 27, 25)\} = 12$.

Theorem 1. *The constrained LCS problem for strings S_1 and S_2 with respect to string P can be solved in* $O(rn^2m^2)$ *time, where n, m and r are lengths of S_1, S_2 and P, respectively.*

Proof. First of all, we describe the preprocessing steps of $O(n^2m^2)$ time for determining $L(x, y, x', y')$ in $O(1)$ time. For $1 \leqslant a \leqslant n$ and $1 \leqslant b \leqslant m$, let $M_{a,b}$ be a 2D matrix of size $(n - a + 1) \times (m - b + 1)$ such that the value of $M_{a,b}[u, v]$ is the length of longest common subsequence of $S_1[a..a + u - 1]$ and $S_2[b..b + v - 1]$, where $1 \leqslant u \leqslant n - a + 1$ and $1 \leqslant v \leqslant m - b + 1$. Matrix $M_{a,b}$ can be computed in $O((n - a + 1) \cdot (m - b + 1))$ time by the algorithm in [8] for computing the length of LCS of $S_1[a..n]$ and $S_2[b..m]$. Hence, all matrices $M_{a,b}$ can be found in $\sum_{1 \leqslant a \leqslant n} \sum_{1 \leqslant b \leqslant m} O((n - a + 1) \cdot (m - b + 1)) = O(n^2m^2)$ time. After those preprocessing steps, the value of $L(x, y, x', y')$ can be determined in $O(1)$ time by table lookup for $M_{x,y}[x' - x + 1, y' - y + 1]$.

According to the formulation in Lemma 1, each $L_k(i, j)$ can be found in $O(ij)$ time if function L_{k-1} is known. Then function L_k can be obtained in $O(\sum_{1 \leqslant i < n, 1 \leqslant j < m} ij) = O(n^2m^2)$ time. Thus all L_k's need $O(rn^2m^2)$ time in total. Moreover, it takes $O(nm)$ time to compute $|lcs|$ in Lemma 2 when function L_r is known. Therefore, we conclude that the constrained LCS problem can be solved in $O(n^2m^2 + rn^2m^2 + nm) = O(rn^2m^2)$ time in total. □

3. Concluding remarks

This paper considers a new problem for finding a longest common subsequence lcs for two strings S_1 and S_2 such that string P is a subsequence of the solution lcs. An $O(rn^2m^2)$ time algorithm based upon dynamic programming has been proposed for this new problem, where n, m and r are lengths of S_1, S_2 and P, respectively. To reduce the time and space requirement of this problem would be the next important work.

The LCS problem on multiple strings was shown to be NP-hard [6]. It is easy to show that the constrained LCS problem for multiple strings is also NP-hard. To exploit exact and approximate algorithms for this problem is a new research direction.

References

[1] A. Apostolico, String editing and longest common subsequences, in: G. Rozenberg, A. Salomaa (Eds.), Linear Modeling: Background and Application, in: Handbook of Formal Languages, Vol. 2, Springer-Verlag, Berlin, 1997, pp. 361–398.
[2] A. Apostolico, General pattern matchings, in: M.J. Atallah (Ed.), Handbook of Algorithms and Theory of Computation, CRC, Boca Raton, FL, 1998, Chapter 13.
[3] L. Bergroth, H. Hakonen, T. Raita, A survey of longest common subsequence algorithms, in: SPIRE, A Coruña, Spain, 2000, pp. 39–48.
[4] D.S. Hirschberg, Algorithms for the longest common subsequence problem, J. ACM 24 (1977) 664–675.
[5] D.S. Hirschberg, Serial computations of Levenshtein distances, in: A. Apostolico, Z. Galil (Eds.), Pattern Matching Algorithms, Oxford University Press, Oxford, 1997, pp. 123–141.
[6] D. Maier, The complexity of some problems on subsequences and supersequences, J. ACM 25 (1978) 322–336.
[7] W.J. Masek, M.S. Paterson, A faster algorithm computing string edit distances, J. Comput. System Sci. 20 (1980) 18–31.
[8] R.A. Wagner, M.J. Fischer, The string-to-string correction problem, J. ACM 21 (1974) 168–173.