

An Algorithm for Longest Common Subsequence (LCS) Problem in Multiple Sequences

Md. Alamgeer

Bioinformatician & Software Engineer, Head & Asst. Professor, Deptt. of Bioinformatics, Singhania University, Pachari Bari, Distt. Jhunjhunu (Rajasthan), India

ABSTRACT

The Longest Common Subsequence (LCS) problem is a long studied prototype of sequence comparison. I present exact algorithm for NP complete LCS problem in multiple sequences, which is very useful operation for analyzing of the bioinformatics data. I propose a recursive program to compute the longest subsequence (if exists) whose pattern matched in all sequences.

Keywords: Longest Common Subsequence, Sequence Comparison, LCS, Pattern Matching.

INTRODUCTION

Finding differences (edit distance) between sequences is often equivalent to finding similarities between these sequences. For example, if edit operations are limited to insertions and deletions (no substitutions), the edit distance problem is equivalent to the Longest Common Subsequence (LCS) Problem. Mathematicians became interested in the LCS Problem long before the dynamic programming algorithm for sequence comparison was discovered. Studies of the symmetric group revealed surprising connections between representation theory and the problem of finding the LCS between two permutations. The first algorithm for this problem was described by Robinson, 1938 [1]. Robinson's work was forgotten until the 1960s, when Schensted, 1961 [2] and Knuth, 1970 [3] re-discovered the relationships between the LCS and tableaux.

LONGEST COMMON SUBSEQUENCE (LCS)

In the early papers on sequence comparison, scientists attempted to find the similarity between entire two strings (V and W), called - global alignment. In many biological applications, the score of alignment between substring of V and W may be larger than the score of alignment between the entire strings. This problem is known as the *local alignment* problem.

A simple dynamic programming algorithm to compute common subsequence between two strings V and W - $s(V, W)$, has been discovered independently by many authors. But there was no solution to compute LCS in multiple sequences.

LCS in Multiple Sequences

There is a recursive program to find the LCS in multiple

sequences. This algorithm compute the longest common subsequence - $s(S_1, S_2, \dots, S_n)$ which exists in list of sequences (S_1, S_2, \dots, S_n) as substring of maximum length.

Let $S_1 \dots S_n$ be n sequences that we want align to find out the LCS. To compute the LCS, we must first pick one sequence of the shortest length. Just assume that the shortest sequence (S_c) has been selected, whose index is a number c between 1 and n and its length is l . Now we find all the possible subsequences of sequence S_c of length l to 3. If the subsequence is of length l , then only one subsequence is possible. On decreasing length by one, number of subsequences increases by same number recursively. Now we match each subsequence in sequences $(S_1 \dots S_n, \text{excluding } S_c)$.

We aggregate this recursive program using two techniques. First is "once pattern of a subsequence is match in sub-parts of all sequences, assign this subsequence as LCS and terminate the iteration". The second technique is "once pattern of a subsequence not matches with sub-parts of any sequence, escape this subsequence and select next possible subsequence".

Algorithm Longest Common Subsequence

Input: list of sequences (S_1, \dots, S_n)

Output: $LCS(S_1, \dots, S_n)$

$n \leftarrow$ number of sequences

$S_c \leftarrow$ shortest sequence of length l

$new_seq_list \leftarrow$ list of sequences after eliminating S_c

$j \leftarrow 0$

// Compute all possible subsequences.

for $i \leftarrow l$ to 3 **do**

$j = j + 1$

```

for s ← 0 to j-1 do
  sub_seq ← sub-sequence of Sc of length
             i from s
  match ← search_lcs (sub_seq)
  if match = 1 then
    s = j, i = 0
    LCS = sub_seq
// Compute pattern matching in all sequences.
search_lcs (sub_seq)
for k ← 0 to n - 2 do
  while sub_seq in new_seq_list[k] do
    flag ← 1
  if flag = 1 then
    return 1
  else
    return 0

```

Figure 1: An Algorithm to Find Longest Common Subsequence in Multiple Sequences

Time Complexity

The time complexity of this programming operation depends on the number of sequences and the length of the shortest sequence used to compute LCS. To see this, first count the number of possible subsequences in shortest sequence are possible. This is the dominant term in the time complexity. Now pattern of each subsequence matched with all sequences excluding shortest sequence until LCS not found.

Let we are entering the n number of sequences. Out of these sequences, one shortest sequence is of length l . Now we need to perform number of all possible subsequences of shortest sequence from length l to 3 decreasing by 1. The sum of all subsequences can be computed as -

$$1 + 2 + 3 + \dots + ((l-3) + 1)$$

The above sum can be computed in closed form as follows -

$$(((l-3) + 1) \times ((l-3) + 2)) / 2$$

Getting a common subsequence, we need to match the pattern of each subsequence in all sequence excluding one shortest sequence. The number of sequences required to match each subsequence is equal to $n - 1$.

If the LCS in sequences is not found, then the total number of iterations (worst score) required to execute the program is -

$$\begin{aligned}
& \{(((l-3) + 1) \times ((l-3) + 2)) / 2\} \times (n-1) \\
& = \sum_{i=l, j=0}^{i=2} (j+1)(n-1)
\end{aligned}$$

Let the LCS of length p ($p \geq l$ and $p \leq 3$) is found. And the starting position of the LCS in shortest sequence (S_c) is s . Then the total iterations (best score) required to computes the program is -

$$\begin{aligned}
& \{(((l-p)) \times ((l-p) + 1)) / 2\} \times (n-1) + s(n-1) \\
& = \sum_{i=l, j=0}^{i=p-1} (j+1)(n-1) + s(n-1)
\end{aligned}$$

RESULTS AND PROOF

This algorithm used to compute the longest common subsequence LCS) in multiple sequences. This accepts n number of sequences ($n \geq 2$). After computing, it returns a subsequence commonly present (locally matched) in all the sequences.

To compute the LCS, let, we inputs five sequences $\{S_1, S_2, \dots, S_n\}$ where $n = 5$.

S_1 . ACGGCGGATATCAGTCATGCGGTTTAGCATGAC

S_2 . ACATCCGATGCGGTTTATATGCGGTTGCCCGCA

S_3 . ATCCGATGCGGTTTA

S_4 . TGACGCGACCCATATCATGCGGTT

S_5 . CCATTCATGCGGTTGCCCATCCGATGCGGTTIACA

First it finds the sequence of shortest length (S_c). In present example, the sequence S_3 is of the shortest length (l).

$$S_c = S_3 = ATCCGATGCGGTTTA; l = 15$$

Now compute all the possible subsequences of S_c from length l to 3 characters recursively, and its pattern matched in sequences $\{S_1, S_2, S_4, \text{ and } S_5\}$. If the pattern of subsequence matched in all the sequences, then present subsequence is initialized as LCS otherwise ignore it by the next iteration. The present example performs as -

In first iteration, only one subsequence of length l $\{ATCCGATGCGGTTTA\}$ is possible. The pattern of this subsequence matched only in S_2 & S_5 but not in S_1 & S_4 . So this subsequence is not the LCS, ignore it and go to the next iteration.

In second iteration, two subsequences of the length $l-1$ $\{ATCCGATGCGGTTT, TCCGATGCGGTTTA\}$ are possible. The pattern of first subsequence of present iteration matched in sequence S_2 & S_5 only, so ignore this. In similar way, the second subsequence also ignored due to mismatch in S_1 & S_4 .

Similarly at sixth iteration, six subsequences of the length $l - 5$ {ATCCGATGCG, TCCGATGCGG, CCGATGCGGT, CGATGCGGTT, GATGCGGTTT, ATGCGGTTTA} are possible. First five subsequences of the present iteration are ignored because pattern of these all subsequences are only matched in sequence S_2 & S_5 . The pattern of last sixth subsequence matched in sequence S_1 , S_2 & S_5 , but not in the sequence S_4 . This subsequence also ignored by the algorithm because this is not common in all the sequences.

By recursion process, the number of subsequences increases and length of subsequence decreases simultaneously. Until subsequence not matched in all sequences and subsequence is of length ≥ 3 , repeat the same iteration process.

To execute the present example, in eighth iteration, total eight subsequences of the length $l - 7$ {ATCCGATG, TCCGATGC, CCGATGCG, CGATGCGG, GATGCGGT, ATGCGGTT, TGCGGTTT, GCGGTTTA} are possible. All the subsequences are ignored after pattern matching except sixth. The pattern of sixth subsequence {ATGCGGTT} matched in all the sequences (S_1 , S_2 , S_4 , & S_5) and this is the subpart of the sequence S_3 . That means, this subsequence is the longest sub part, common in all the sequences. So this subsequence is the LCS.

Time Complexity (best case)

To compute the above example, total required iterations (best score) are calculated as -

$$\{[(l-p) \times ((l-p) + 1)] / 2\} \times (n-1) + s(n-1)$$

where $l = 15, p = 8, n = 5, s = 6$

After putting all the variables value, total iteration

$$= \{[(15-8) \times ((15-8) + 1)] / 2\} \times (5-1) + 6(5-1)$$

$$= 136$$

Time Complexity (worst case)

Suppose LCS is not found in the above example, then total required iterations (worst score) are calculated as -

$$\{[(l-3) + 1] \times ((l-3) + 2) / 2\} \times (n-1)$$

where $l = 15, p = 8, n = 5$

After putting all the variables value, total iteration

$$= \{[(15-3) + 1] \times ((15-3) + 2) / 2\} \times (5-1)$$

$$= 364$$

AVAILABILITY

Through bioinformatics approach, GenSolution is a website, accessible at the URL [http:// gensolution.org](http://gensolution.org). This website provides the site to solve some biological problems by computational method. The author also implements the algorithm of Longest Common Subsequence problem in multiple sequences here. The LCS program is accessible through drop down menu option of SeqComparison present at website home page.

ACKNOWLEDGEMENTS

The author thanks Dr. S.I. Ahson and Dr. Khurshid Haider for discussions and encouragements. He gratefully acknowledges the partial support of Dr. Abdul Ilah and Dr. Kamal Raval. The author also thanks Dr. Kulvinder Singh Saini and Dr. V.C. Kalia for giving me opportunity to realize such types of biological problem during the project training at Ranbaxy and IGIB research center respectively.

REFERENCES

- [1] G.de E. Robinson, "On Representations of the Symmetric Group", *American Journal of Mathematics*, 60:745-760, 1938.
- [2] C. Schensted, "Longest Increasing and Decreasing Subsequences", *Canadian Journal of Mathematics*, 13:179-191, 1961.
- [3] D.E. Knuth, "Permutations, Matrices and Generalized Young Tableaux", *Pacific Journal of Mathematics*, 34:709-727, 1970.