Problem statement: The longest filled common sequence problem

Marko Djukanović

April 15, 2025

1 The longest filled common sequence problem

1.1 Introduction

The Longest Common Subsequence (LCS) problem is an important optimization problem that measures the structural similarity between two or more biological sequences. It has applications in computational biology, data compression, text editing, etc. Notable examples include the `diff` tool in Unix systems or integration in various plagiarism detection systems.

In computational biology, the reconstruction of genomic data plays an important role in analyzing genomes, thus yielding to development of various LCS variants. Among them, recently introduced *the longest filled common sequence* (LFCS) problem is one of such kind.

1.2 Task

In the LFCS problem, given is an arbitrary set of genes $\mathcal{M}$ provided by biologists that are found important or sensible after gene sequencing, and two biological sequences. The task is to find an appropriate set of gene imputations in one of these sequences so that the structural similarity of the affected sequence and the second sequence is the highest possible. In that way, more is told about how likely is that these two molecules share the same origin and thus more likely to possess similar functional behavior. The LFCS is inspired by the particulars of the known *Scaffold Filling* problem in genome reconstruction.

1.3 Problem statement

Given are two strings s_1 and s_2 over a finite alphabet Σ and a multiset $\mathcal{M}$ of symbols from Σ. We say that s_2^* is *filled* by s_2 if it is obtained from s_2 by implanting a subset of allowed symbols from $\mathcal{M}$.
The objective is to find a filled sequence s_2^* so that it maximizes the LCS between s_1 and (filled) s_2^*.

A feasible *solution* s_2^* is any filled sequence obtained by implanting an arbitrary subset of characters from (multiset) $\mathcal{M}$ in s_2.

The objective value of a filled (feasible) solution s_2^* is the length of the LCS between s_1 and s_2^*.

It is formally proven that the LFCS problem is $\mathcal{APX}$-hard, thus also $\mathcal{NP}$-hard.

1.4 Instance data file

Each *problem instance* is characterized by two (input) sequences of length $m = |s_1|$ and $n = |s_2|$ that may be written in the first two lines of the file. The third line is designed to store the characters of the multiset $\mathcal{M}$, where k represents the size of $\mathcal{M}$. Therefore, each instance is represented by a triple $(s_1, s_2, \mathcal{M})$.

The extension of the file can be *.txt*, *.csv*, or some other similar (textual) format.

1.5 Solution file

A solution can be stored in a textual-type file. In the first line is a sequence of characters (a string datatype representation) representing the resulting filled sequence s_2^*. Hereby, the objective value can be stored in the second line.

1.6 Example

For the shake of clarity, given is an example in the subsequent section.

1.6.1 Instance

Input file:

```
abcdbcda
cabbdda
abd
```

According to the notation provide with the problem definition, $s_1 =$ `abcdbcda`, $s_2 =$ `cabbdda`, and $\mathcal{M} = \{\texttt{a}, \texttt{b}, \texttt{d}\}$, where $\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$

1.6.2 Solution

An optimal solution to the problem problem instance given below is: $s_2^* =$ `abcdabbdda` with the resulting optimal value of $7 = |\text{LCS}(s_1, s_2^*)|$.

1.6.3 Explanation

There are possibly $3 \cdot 8 + 3 \cdot 8 \cdot 7 + 8 \cdot 7 \cdot 6 + 1 = 24 + 168 + 336 + 1 = 529$ different feasible solutions for the above problem instance, which actually correspond to any possibility that each subset of three letters to be implanted at any of

8 possible places (at index 0 – before the leading `c`, at position 1 – between the leading `a` and `b`, ..., at position 8 – after the last `a`). For each of these 529 solutions, one should determine the LCS between them and s_1 (e.g. by a dynamic programming), yielding the conclusion that `abcdabbdda` is an optimal solution along with two others, `abcadbbdda`, `abcabdbdda`.

References

[1] Castelli, M., Dondi, R., Mauri, G., & Zoppis, I. (2017). The longest filled common subsequence problem. In 28th Annual Symposium on Combinatorial Pattern Matching (CPM 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[2] Mincu, R. S., & Popa, A. (2018, September). Heuristic algorithms for the longest filled common subsequence problem. In 2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC) (pp. 449-453). IEEE.