

Quadratic Conditional Lower Bounds for String Problems and Dynamic Time Warping

Karl Bringmann* Marvin Künnemann†

April 6, 2015

Abstract

Classic similarity measures of strings are longest common subsequence and Levenshtein distance (i.e., the classic edit distance). A classic similarity measure of curves is dynamic time warping. These measures can be computed by simple $\mathcal{O}(n^2)$ dynamic programming algorithms, and despite much effort no algorithms with significantly better running time are known.

We prove that, even restricted to binary strings or one-dimensional curves, respectively, these measures do not have strongly subquadratic time algorithms, i.e., no algorithms with running time $\mathcal{O}(n^{2-\varepsilon})$ for any $\varepsilon > 0$, unless the Strong Exponential Time Hypothesis fails. We generalize the result to edit distance for arbitrary fixed costs of the four operations (deletion in one of the two strings, matching, substitution), by identifying trivial cases that can be solved in constant time, and proving quadratic-time hardness on binary strings for all other cost choices. This improves and generalizes the known hardness result for Levenshtein distance [Backurs, Indyk STOC'15] by the restriction to binary strings and the generalization to arbitrary costs, and adds important problems to a recent line of research showing conditional lower bounds for a growing number of quadratic time problems.

As our main technical contribution, we introduce a framework for proving quadratic-time hardness of similarity measures. To apply the framework it suffices to construct a single gadget, which encapsulates all the expressive power necessary to emulate a reduction from satisfiability.

Finally, we prove quadratic-time hardness for longest palindromic subsequence and longest tandem subsequence via reductions from longest common subsequence, showing that conditional lower bounds based on the Strong Exponential Time Hypothesis also apply to string problems that are not necessarily similarity measures.

1 Introduction

For many classic polynomial time problems the worst-case running time is stagnant for decades, e.g., a classic algorithm solves the problem in time $\tilde{\mathcal{O}}(n^2)$, up to logarithmic factors, but it is unknown whether any faster algorithms exist. For these problems we would like to explain why it is hard to find faster algorithms. One type of explanation is a *conditional lower bound*. Here we assume that some problem P has no algorithms faster than a long-standing time barrier and prove resulting lower bounds for other problems, via reductions from P . The most prominent such approach is 3SUM-hardness, which dates back to 1995 [11]: Assuming that 3SUM has no

*Institute of Theoretical Computer Science, ETH Zurich, Switzerland, karlb@inf.ethz.ch

†Max Planck Institute for Informatics, Saarbrücken, Germany, marvin@mpi-inf.mpg.de

(strongly) subquadratic algorithms, many lower bounds have been shown, especially for problems in computational geometry. However, for many other problems it seems to be impossible to find a reduction from 3SUM.

In the last years, new assumptions emerged that allow to prove conditional lower bounds for problems where 3SUM-hardness does not seem to apply. The prime example is the Strong Exponential Time Hypothesis (SETH), which was introduced by Impagliazzo and Paturi [13] and asserts that satisfiability has no algorithms that are much faster than exhaustive search.

Hypothesis SETH: *For no $\varepsilon > 0$, k -SAT can be solved in time $\mathcal{O}(2^{(1-\varepsilon)N})$ for all $k \geq 3$.*

Note that exhaustive search takes time $\mathcal{O}(2^N)$ and the best-known algorithms for k -SAT have a running time of the form $\mathcal{O}(2^{(1-c/k)N})$ for some constant $c > 0$ [18]. Thus, SETH is a reasonable hypothesis and, due to lack of progress in the last decades, can be considered unlikely to fail.

The idea to use SETH to prove conditional lower bounds for polynomial time problems dates back to 2005 [23], but only in recent years more and more such conditional lower bounds have been proven, see, e.g., [1, 2, 3, 6, 8, 17, 19]. Two recent examples, that motivated this paper, are the conditional lower bounds for Fréchet distance [8] and Levenshtein distance [6]. Both problems are natural *similarity measures* between two sequences (curves or strings, respectively). In this paper we study additional classic similarity measures between strings and curves. We propose a framework for proving lower bounds for such similarity measures. This allows us to prove quadratic-time hardness of the following problems.

Edit Distance Given two strings x, y of length n, m ($n \geq m$), we start in their first symbols at positions $(1, 1)$ and traverse them up to their last symbols at positions (n, m) using the following operations: If we are at positions (i, j) we may (1) delete a symbol in x (this costs $c_{\text{del-x}}$ and we advance to $(i + 1, j)$), (2) delete a symbol in y (this costs $c_{\text{del-y}}$ and we advance to $(i, j + 1)$), (3) match the current symbols, which is only possible if $x[i] = y[j]$ (this costs c_{match} and we advance to $(i + 1, j + 1)$), or (4) substitute the current symbols, which is only possible if $x[i] \neq y[j]$ (this costs c_{subst} and we advance to $(i + 1, j + 1)$). The minimum total cost of such a sequence of operations is called the edit distance of x and y , and we denote the problem of computing the edit distance by $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$. The Levenshtein distance (i.e., the classic edit distance) is $\text{Edit}(1, 1, 0, 1)$. An important special case is the longest common subsequence (LCS) of two strings, which can be seen to be equivalent to $\text{Edit}(1, 1, 0, 2)$. One obtains more variants for other cost choices, e.g., for aligning DNA sequences a classic choice is $\text{Edit}(2, 2, -1, 1)$ [22].

Edit distance has a natural dynamic programming algorithm with running time $\mathcal{O}(nm)$, which is taught in many undergraduate algorithms courses. Since such string distance measures have many applications in bioinformatics and data comparison, Levenshtein distance and LCS are well-studied with a rich literature focussing on approximation algorithms (see, e.g., [5]) and algorithms that perform well on special cases (see, e.g., [12] and see [7] for a survey). However, the best-known worst-case running time (of an exact algorithm) is $\mathcal{O}(nm/\log n + n)$ [16], i.e., algorithms are stuck slightly below quadratic time. Even if we restrict the input to strings over a binary alphabet $\{0, 1\}$ no better worst-case running time is known. In this paper we present a possible explanation for this situation by proving conditional lower bounds for edit distance on binary strings, thus improving and generalizing the known quadratic-time hardness for the Levenshtein distance on alphabet size 4 [6].

Dynamic Time Warping (DTW) Fix a metric space (M, d) . A sequence of points in M is called a *curve*. Consider two curves x, y of length n, m ($n \geq m$). We may *traverse* x and y by starting in their first entries, in any time step advancing to the next entry in x or y or both, and ending in their last entries (see Section 2 for details). The cost of such a traversal is the sum over all points in time of the distance between the current entries. The dynamic time warping distance of x and y is the minimal cost of any traversal. This similarity measure can, e.g., readily detect whether two given signals are equal up to time accelerations or decelerations. This property, among others, makes it a very useful measure in practice, with many applications in comparing temporal data such as video and audio, e.g., for speech recognition or music processing (see, e.g., [20]). The best-known worst-case running time is achieved by a simple dynamic programming algorithm that computes the DTW distance of x and y in time $\mathcal{O}(nm)$. To break this apparent barrier in practice, many heuristics have been designed for this problem (see, e.g., [21]).

An important special case that frequently arises in practice is *dynamic time warping on one-dimensional curves*. Here the metric space is $M = \mathbb{R}$ and the distance measure is $d(a, b) := |a - b|$ for any $a, b \in \mathbb{R}$. Even for this important special case the best-known algorithm takes time $\mathcal{O}(nm)$. We provide a possible explanation for this situation by proving a conditional lower bound for DTW on one-dimensional curves.

1.1 Our Results

Dynamic Time Warping As our first main result, we prove a conditional lower bound for DTW. This shows that strongly subquadratic algorithms for DTW can be considered unlikely to exist. Specifically, obtaining such algorithms is at least as hard as a breakthrough for satisfiability.

Theorem 1.1. *DTW on one-dimensional curves taking values in $\{0, 1, 2, 4, 8\} \subseteq \mathbb{R}$ has no $\mathcal{O}(n^{2-\varepsilon})$ algorithm for any $\varepsilon > 0$, unless SETH fails.*

Edit Distance Our second main result is a classification of $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ for all operation costs $c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}}$: We identify trivial variants where the edit distance is independent of the input x, y , and only depends on n, m . In this case, it can be computed in constant time. For all remaining choices of the operation costs we prove quadratic-time hardness, even restricted to binary strings. This includes quadratic-time hardness of LCS and Levenshtein distance on binary strings. Compared to the known lower bound for Levenshtein distance [6], our result decreases the alphabet size from 4 to 2 and adds hardness of a large class of problems including LCS.

Theorem 1.2. *$\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ can be solved in constant time if $c_{\text{subst}} = c_{\text{match}}$ or $c_{\text{del-x}} + c_{\text{del-y}} \leq \min\{c_{\text{match}}, c_{\text{subst}}\}$. Otherwise, $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ on binary strings has no $\mathcal{O}(n^{2-\varepsilon})$ algorithm for any $\varepsilon > 0$, unless SETH fails.*

As first step of the hardness part of this theorem, for some $0 < c'_{\text{subst}} \leq 2$ depending on $c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}}$ we reduce $\text{Edit}(1, 1, 0, c'_{\text{subst}})$ to $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$. This reduction is what fails for the trivial cases. Then we prove hardness of $\text{Edit}(1, 1, 0, c'_{\text{subst}})$ using a construction that is parameterized by c'_{subst} .

Unbalanced Inputs Our main results are most meaningful for inputs with $n \approx m$. It is conceivable that for unbalanced inputs, i.e., $m \ll n$, faster algorithms exist, say the running time of

$\mathcal{O}(nm)$ could be reduced to $\tilde{\mathcal{O}}(n + m^2)$. For DTW we show that such an improvement is unlikely, by proving that “for any m ” no algorithm with running time $\mathcal{O}((nm)^{1-\varepsilon})$ exists, assuming SETH. This is analogous to the situation for Fréchet distance [8].

Theorem 1.3. *Unless SETH fails, DTW on one-dimensional curves taking values in $\{0, 1, 2, 4, 8\}$ has no $\mathcal{O}((nm)^{1-\varepsilon})$ algorithm for any $\varepsilon > 0$, and this even holds restricted to instances with $n^{\alpha-o(1)} \leq m \leq n^{\alpha+o(1)}$ for any $0 < \alpha < 1$.*

For edit distance, Theorem 1.2 implies that there is no $\mathcal{O}(m^{2-\varepsilon})$ algorithm for any $\varepsilon > 0$ (in the worst case over all strings x, y with $|x| \leq n$ and $|y| \leq m$ for any $n \geq m$). Our reduction from SETH cannot result in unbalanced strings, and thus we are not able to prove better lower bounds than $\mathcal{O}(m^{2-\varepsilon})$. This behaviour hints at the possibility of an $\tilde{\mathcal{O}}(n + m^2)$ algorithm for edit distance - and indeed there is an algorithm for LCS from '77 due to Hirschberg [12] matching this time complexity. For completeness, we show that this algorithm can be generalized to edit distance.

Theorem 1.4. *Edit($c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}}$) has an $\tilde{\mathcal{O}}(n + m^2)$ algorithm.*

Thus, for unbalanced inputs DTW and edit distance differ in their behaviour, but using SETH we can readily explain this difference.

Reductions from Longest Common Subsequence Note that any near-linear time reduction from LCS to another problem P transfers the quadratic-time lower bound of LCS to P . We think that this notion of *LCS-hardness* could be used to prove lower bounds for many string problems (not only distance measures). To support this claim, we present two easy results in this direction.

A *palindromic subsequence* (also called symmetric subsequence) of a string x of length n is a subsequence z that is the same as its reverse $\text{rev}(z)$. Computing a longest palindromic subsequence is a popular exercise in undergraduate text books (e.g., [9, Exercise 15-2]), since it can be easily solved by a reduction to LCS or adapting the dynamic programming solution of LCS, both resulting in an $\mathcal{O}(n^2)$ algorithm. A *tandem subsequence* of a string x is a subsequence z that can be written as the concatenation $z = yy$ of a string y with itself. In contrast to longest palindromic subsequence, it is non-trivial to compute a longest tandem subsequence in time $\mathcal{O}(n^2)$ [15]. We present reductions from LCS to both of these problems, which yields the following lower bounds.

Theorem 1.5. *On binary strings, longest palindromic subsequence and longest tandem subsequence have no $\mathcal{O}(n^{2-\varepsilon})$ algorithms for any $\varepsilon > 0$, unless SETH fails.*

These results show that SETH-based lower bounds via LCS are applicable to string problems that are not necessarily similarity measures.

1.2 Technical Contribution

We introduce a framework for proving SETH-based lower bounds for similarity measures. It is based on a construction that we call *alignment gadget*. Given instances x_1, \dots, x_n and y_1, \dots, y_m , $m \leq n$, an alignment gadget consists of two instances x, y whose similarity $\delta(x, y)$ is closely related to $\sum_{(i,j) \in A} \delta(x_i, y_j)$, where $A = \{(i_1, 1), \dots, (i_m, m)\}$ is the best-possible ordered alignment of the numbers in $[m]$ to $[n]$ (for details see Section 3). We prove a quadratic lower bound for any similarity measure admitting an alignment gadget. This proof is a simplified version of a construction in the

known lower bound for Levenshtein distance [6], which is also closely related to the lower bound for Fréchet distance [8].

Working with our framework has two advantages: First, it unifies three constructions that are separate proof steps in other SETH-based lower bounds [6, 8], thus reducing the amount of work necessary to prove SETH-based lower bounds. Second, it hides the reduction from satisfiability, providing a level of abstraction that allows to ignore the details of the satisfiability problem and instead focus on the details of the problem we reduce to. This makes it possible to tackle general problems such as $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$, where the reduction depends on parameters of the problem, without resulting in an overly complex proof.

We present alignment gadgets for edit distance and dynamic time warping. This part needs careful problem-specific constructions. In particular, we have to construct instances where the optimal sequence of edit distance operations has some exploitable structure, which is made difficult by the fact that we work over binary alphabet, so that in principle any two zeroes and any two ones can be matched.

1.3 Related Work

Independently of our work, similar lower bounds for LCS and DTW have been shown by Abboud et al. [1]. Let us briefly compare our approaches. Our main technical contribution is the alignment-framework, which allows us to give shorter hardness proofs. The proofs of Abboud et al. are longer, in particular since they are using the lower bound for Levenshtein distance [6], while our proofs are self-contained. The main technical contribution of Abboud et al., apart from careful reductions, seems to be that they reduce from a novel problem that they call Most-Orthogonal Vectors. Regarding the problem LCS, our hardness result is stronger, since we show hardness on binary strings, while Abboud et al. need alphabet size 7. Regarding DTW, we prove hardness of different special cases, as we consider DTW on one-dimensional curves over alphabets of size 5 (where the distance of two numbers is their absolute difference), while Abboud et al. consider DTW on strings over alphabets of size 5 (where the distance of two symbols is 1 or 0, depending on whether they are equal or not). On top of these core results, Abboud et al. generalize their result for LCS to k -LCS, the longest common subsequence of k strings. We classify the complexity of edit distance for arbitrary operation costs and prove hardness of additional string problems via reductions from LCS.

1.4 Organization

In Section 2 we fix notation and discuss alternative assumptions to SETH that can be used to prove our results. We present our framework for obtaining quadratic lower bounds in Section 3. We then first prove a conditional lower bound for LCS in Section 4; this proof is superseded by the conditional lower bound for edit distance in Section 5, but it is shorter and might be more accessible. Quadratic-time hardness of dynamic time warping follows in Section 6. Finally, in Section 7 we prove hardness of longest palindromic subsequence and longest tandem subsequence.

2 Preliminaries

For a sequence x , we write $|x|$ for its length, $x[k]$ for its k -th entry, $x[k..\ell]$ for the substring from $x[k]$ to $x[\ell]$, and $\text{rev}(x)$ for the reversed sequence. For sequences x, y we denote their concate-

nation by xy . A *traversal* of two sequences x, y of length n, m , respectively, is a sequence of pairs $((a_1, b_1), \dots, (a_t, b_t))$ with $t \in \mathbb{N}$ satisfying (1) $(a_1, b_1) = (1, 1)$, (2) $(a_t, b_t) = (n, m)$, and (3) (a_{i+1}, b_{i+1}) is either of $(a_i + 1, b_i)$, $(a_i, b_i + 1)$, or $(a_i + 1, b_i + 1)$ for all $1 \leq i < t$.

Edit Distance Let x, y be strings over an alphabet Σ of length n, m ($n \geq m$), respectively. For a traversal $T = ((a_1, b_1), \dots, (a_t, b_t))$ of x, y we say that its i -th operation, $1 \leq i < t$, is (1) a *deletion in x* if $(a_{i+1}, b_{i+1}) = (a_i + 1, b_i)$, (2) a *deletion in y* if $(a_{i+1}, b_{i+1}) = (a_i, b_i + 1)$, (3) a *matching* if $(a_{i+1}, b_{i+1}) = (a_i + 1, b_i + 1)$ and $x[a_i] = y[b_i]$, or (4) a *substitution* if $(a_{i+1}, b_{i+1}) = (a_i + 1, b_i + 1)$ and $x[a_i] \neq y[b_i]$. These four operations incur costs of $c_{\text{del-}x}$, $c_{\text{del-}y}$, c_{match} , and c_{subst} , respectively. We will always assume that these costs are rational constants, so that we can ignore representation issues. The cost $\delta_{\text{Edit}}(T)$ of a traversal T is the total cost of all its operations. The edit distance $\delta_{\text{Edit}}(x, y)$ is the minimal cost of any traversal of x, y . We write $\text{Edit}(c_{\text{del-}x}, c_{\text{del-}y}, c_{\text{match}}, c_{\text{subst}})$ for the problem of computing the edit distance of two given strings with costs $c_{\text{del-}x}, c_{\text{del-}y}, c_{\text{match}}$, and c_{subst} . We write $\text{Edit}(c_{\text{subst}})$ as a shorthand for $\text{Edit}(1, 1, 0, c_{\text{subst}})$. Note that for these problems the costs of all four operations are constant, i.e., they stay fixed with growing n, m . We will mostly consider edit distance over binary strings, i.e., we set $\Sigma = \{0, 1\}$.

Dynamic Time Warping (DTW) Let (M, d) be any metric space. Let x, y be curves, i.e., sequences over M of length n, m ($n \geq m$), respectively. The cost $\delta_{\text{DTW}}(T)$ of a traversal $T = ((a_1, b_1), \dots, (a_t, b_t))$ is $\sum_{i=1}^t d(x[a_i], y[b_i])$. The dynamic time warping distance $\delta_{\text{DTW}}(x, y)$ is the minimal cost of any traversal of x and y . We obtain the special case of *dynamic time warping on one-dimensional curves* by setting $M = \mathbb{R}$ and $d(a, b) := |a - b|$ for any $a, b \in \mathbb{R}$.

2.1 Hardness Assumptions

Consider the *Orthogonal Vectors problem (OV)*: Given sets A, B of vectors in $\{0, 1\}^d$, $|A| = n, |B| = m$, decide whether there is a pair of vectors $a \in A, b \in B$ such that $a[k] \cdot b[k] = 0$ for all k (which we denote by $\langle a, b \rangle = 0$). Clearly, this problem can be solved in time $\mathcal{O}(n^2d)$. The best-known algorithm runs in time $n^{2-1/\mathcal{O}(\log(d/\log n))}$ [4], which is only slightly subquadratic for $d \gg \log n$. Thus, the following hypotheses are reasonable.

Orthogonal Vectors Hypothesis (OVH): *For no $\varepsilon > 0$ there is an algorithm for OV, restricted to $n = m$, that runs in time $\mathcal{O}(n^{2-\varepsilon} \text{poly}(d))$.*

Unbalanced Orthogonal Vectors Hypothesis (UOVH): *Let $0 < \alpha \leq 1$. For no $\varepsilon > 0$ there is an algorithm for OV, restricted to $m = \Theta(n^\alpha)$ and $d \leq n^{o(1)}$, that runs in time $\mathcal{O}((nm)^{1-\varepsilon})$.*

It is well-known that SETH implies OVH [23]. A slight generalization shows that SETH also implies UOVH. Hence, these hypotheses are weaker assumptions than SETH.

Lemma 2.1. *SETH implies OVH and UOVH.*

Proof. For OVH the statement follows from [23]. Let $0 < \varepsilon < 1/2$ and $0 < \alpha \leq 1$. Assume that Orthogonal Vectors, restricted to $m = \Theta(n^\alpha)$ and $d \leq n^{o(1)}$, has an $\mathcal{O}((nm)^{1-\varepsilon})$ algorithm. We show that this contradicts SETH. To this end, let φ be an instance of k -SAT with N variables and M clauses. We use the sparsification lemma [14], which yields $t := 2^{\varepsilon N/2}$ k -SAT instances

$\varphi_1, \dots, \varphi_t$ with N variables and $f(k, \varepsilon) \cdot N$ clauses such that φ is satisfiable if and only if some φ_i is satisfiable. If $N \leq f(k, \varepsilon)$ then we decide each φ_i in time $\mathcal{O}_{k, \varepsilon}(1)$. Otherwise, φ_i has at most N^2 clauses, and we can assume equality by duplicating clauses. In this case, we construct an instance of Orthogonal Vectors as follows. Let x_1, \dots, x_N be the variables and C_1, \dots, C_{N^2} be the clauses of φ_i . We set $d := N^2$ and split the variables into the left half $x_1, \dots, x_{N/(1+\alpha)}$ and the right half $x_{N/(1+\alpha)+1}, \dots, x_N$. The set A consists of one vector $a_z \in \mathbb{R}$ for every assignment z of true and false to the left half of the variables. If z causes clause C_i to be true, i.e., some unnegated variable of C_i is set to true in z or some negated variable of C_i is set to false in z , then we set $a_z[i] := 0$. Otherwise, we set $a_z[i] := 1$. Similarly, set B has a vector $b_{z'}$ for any assignment z' of true or false to the right half of the variables and $b_{z'}[i] = 0$ or 1 , depending on whether z' causes clause C_i to be true. Then $\langle a_z, b_{z'} \rangle = 0$ if and only if (z, z') forms a satisfying assignment of φ_i . Thus, we can decide φ_i by solving the constructed instance of Orthogonal Vectors. Note that $n = |A| = 2^{N/(1+\alpha)}$ and $m = |B| = 2^{N\alpha/(1+\alpha)}$, so that indeed $m = \Theta(n^\alpha)$. Moreover, $d = N^2 \leq 2^{o(N)} = n^{o(1)}$. Thus, we can apply the algorithm for Orthogonal Vectors, that we assumed to exist, running in time $\mathcal{O}((nm)^{1-\varepsilon}) = \mathcal{O}(2^{(1-\varepsilon)N})$. Running this procedure for all φ_i decides φ in time $\mathcal{O}(t \cdot 2^{(1-\varepsilon)N}) = \mathcal{O}(2^{(1-\varepsilon/2)N})$, contradicting SETH. \square

Thus, any lower bound conditional on OVH or UOVH also holds conditional on SETH. In fact, we prove all of our results by reductions from Orthogonal Vectors, so that in our results we may replace the assumption SETH by OVH or UOVH. Specifically, in Theorems 1.1, 1.2, and 1.5 we can replace SETH by OVH, and in Theorem 1.3 we can replace SETH by UOVH. We remark that a version of OVH has also been used in [1] and is implicit in many other SETH-based lower bounds.

3 Framework

We consider a similarity (or distance) measure $\delta : \mathcal{I} \times \mathcal{I} \rightarrow \mathbb{N}_0$, where \mathcal{I} denotes the set of inputs, e.g., all binary strings or all one-dimensional curves. By a reduction from Orthogonal Vectors, we prove that computing this similarity measure cannot be done in strongly subquadratic time unless SETH fails if δ admits a gadget that allows us to exactly realize alignments of inputs $x_1, \dots, x_n \in \mathcal{I}$ and $y_1, \dots, y_m \in \mathcal{I}$. To formally state the requirement, we start by introducing the following notions.

Types In this paper, we define the *type* of a sequence $x \in \mathcal{I}$ to be its length and the sum of its entries, i.e., $\text{type}(x) := (|x|, \sum_i x[i])$ (where for binary strings $\sum_k x[k]$ is to be interpreted as the number of ones in x). The definition of types can be customized to the similarity measure under consideration and is chosen to work for the problems considered in this paper. We define $\mathcal{I}_t := \{x \in \mathcal{I} \mid \text{type}(x) = t\}$ as the set of inputs of type t .

Alignments Let $n \geq m$. A (*partial*) *alignment* is a set $A = \{(i_1, j_1), \dots, (i_k, j_k)\}$ with $0 \leq k \leq m$ such that $1 \leq i_1 < \dots < i_k \leq n$ and $1 \leq j_1 < \dots < j_k \leq m$. We say that $(i, j) \in A$ are *aligned*. Any $i \in [n]$ or $j \in [m]$ that is not contained in any pair in A is called *unaligned*. We denote the set of all partial alignments (with respect to n, m) by $\mathcal{A}_{n,m}$.

We call the partial alignment $\{(\Delta + 1, 1), \dots, (\Delta + m, m)\}$, with $0 \leq \Delta \leq n - m$, a *structured alignment*. We denote the set of all structured alignments by $\mathcal{S}_{n,m}$.

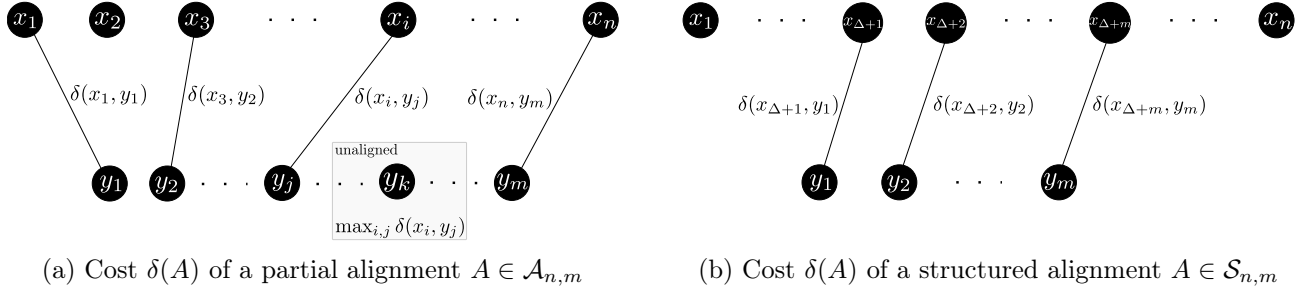


Figure 1: Costs of Alignments

For any $x_1, \dots, x_n \in \mathcal{I}$ and $y_1, \dots, y_m \in \mathcal{I}$ we define the *cost* of alignment $A \in \mathcal{A}_{n,m}$ by

$$\delta(A) = \delta_{y_1, \dots, y_m}^{x_1, \dots, x_n}(A) := \sum_{(i,j) \in A} \delta(x_i, y_j) + (m - |A|) \max_{i,j} \delta(x_i, y_j).$$

In other words, for any $j \in [m]$ which is aligned to some i we pay the distance $\delta(x_i, y_j)$, while for any unaligned j we pay the maximal distance of any $(x_{i'}, y_{j'})$ (note that there are $m - |A|$ unaligned $j \in [m]$, see Figure 1). This means that we get punished for any unaligned j .

Alignment Gadget We start with some intuition. Consider the problem of computing the value $\min_{A \in \mathcal{S}_{n,m}} \delta(A)$. This can be solved in time $\mathcal{O}(nm)$ if each $\delta(x_i, y_j)$ can be evaluated in constant time, since $|\mathcal{S}_{n,m}| = \mathcal{O}(n)$ and evaluating $\delta(A)$ amounts to computing m values $\delta(x_i, y_j)$. Moreover, intuitively it should not be possible to compute this value in strongly subquadratic time. We will show that in some sense it is even hard to compute, in strongly subquadratic time, *any value* v with

$$\min_{A \in \mathcal{A}_{n,m}} \delta(A) \leq v \leq \min_{A \in \mathcal{S}_{n,m}} \delta(A). \quad (1)$$

Now, an alignment gadget is simply a pair of instances (x, y) such that from $\delta(x, y)$ we can infer¹ a value v as above. The main reason to relax our goal from computing $\min_{A \in \mathcal{S}_{n,m}} \delta(A)$ to satisfying (1) is that this makes constructing alignment gadgets much easier. Note that for the alignment gadget (x, y) computing $\delta(x, y)$ is as hard as computing $\min_{A \in \mathcal{S}_{n,m}} \delta(A)$ (in an approximate sense as given by (1)), which we argued above should take quadratic time. This informal discussion motivates the following definition.

Definition 3.1. The similarity measure δ admits an *alignment gadget*, if the following conditions hold: Given instances $x_1, \dots, x_n \in \mathcal{I}_{t_x}$, $y_1, \dots, y_m \in \mathcal{I}_{t_y}$ with $m \leq n$ and types $t_x = (\ell_x, s_x)$, $t_y = (\ell_y, s_y)$, we can construct new instances $x = \text{GA}_x^{m, t_y}(x_1, \dots, x_n)$ and $y = \text{GA}_y^{n, t_x}(y_1, \dots, y_m)$ and $C \in \mathbb{Z}$ such that

$$\min_{A \in \mathcal{A}_{n,m}} \delta(A) \leq \delta(x, y) - C \leq \min_{A \in \mathcal{S}_{n,m}} \delta(A). \quad (2)$$

Moreover, $\text{type}(x)$ and $\text{type}(y)$ only depend on n, m, t_x , and t_y . Finally, this construction runs in time $\mathcal{O}((n+m)(\ell_x + \ell_y))$.

If the construction additionally fulfills $|x| = \mathcal{O}(n(\ell_x + \ell_y))$ and $|y| = \mathcal{O}(m(\ell_x + \ell_y))$, then we say that δ admits an *unbalanced alignment gadget*.

¹For us “infer” will simply mean that $v = \delta(x, y) - C$ for an appropriate C .

Note that the types serve the purpose of simplifying the algorithmic problem in the above definition by restricting the inputs to same-type objects. If we can construct suitable x and y for arbitrary inputs x_1, \dots, x_n and y_1, \dots, y_m then we may completely disregard types.

Definition 3.2. The similarity measure δ admits *coordinate values*, if there exist $\mathbf{0}_x, \mathbf{0}_y, \mathbf{1}_x, \mathbf{1}_y \in \mathcal{I}$ satisfying

$$\delta(\mathbf{1}_x, \mathbf{1}_y) > \delta(\mathbf{0}_x, \mathbf{1}_y) = \delta(\mathbf{0}_x, \mathbf{0}_y) = \delta(\mathbf{1}_x, \mathbf{0}_y),$$

and moreover, $\text{type}(\mathbf{0}_x) = \text{type}(\mathbf{1}_x)$ and $\text{type}(\mathbf{0}_y) = \text{type}(\mathbf{1}_y)$.

Theorem 3.3. Let δ be a similarity measure admitting an alignment gadget and coordinate values and consider the problem of computing $\delta(x, y)$ with $|x| \leq n$, $|y| \leq m$, and $m \leq n$. For no $\varepsilon > 0$ this problem can be solved in time $\mathcal{O}(m^{2-\varepsilon})$ unless OVH fails. If δ even admits an unbalanced alignment gadget, then for no $\varepsilon > 0$ this problem can be solved in time $\mathcal{O}((nm)^{1-\varepsilon})$, unless UOVH fails. Both statements hold restricted to $n^{\alpha-o(1)} \leq m \leq n^{\alpha+o(1)}$ for any $0 < \alpha \leq 1$.

3.1 Proof of Theorem 3.3

We present a reduction from OV to the problem of computing δ . This uses constructions and arguments similar to [8, 6]. Consider an instance $a_1, \dots, a_n \in \{0, 1\}^d$ and $b_1, \dots, b_m \in \{0, 1\}^d$ of OV, $n \geq m$. We construct $x, y \in \mathcal{I}$ and $\rho \in \mathbb{N}_0$ such that $\delta(x, y) \leq \rho$ if and only if there are $i \in [n]$ and $j \in [m]$ with $\langle a_i, b_j \rangle = 0$. To this end, let $a_i[k]$ denote the k -th component of a_i . For all $i \in [n]$ and $j \in [m]$, we construct *coordinate gadgets* as follows

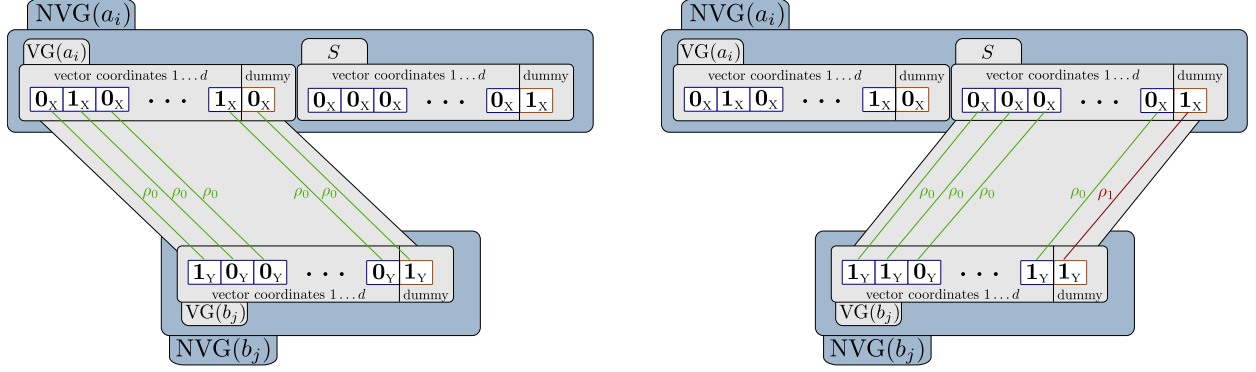
$$\begin{aligned} \text{CG}(a_i, k) &:= \begin{cases} \mathbf{0}_x & \text{if } a_i[k] = 0 \\ \mathbf{1}_x & \text{if } a_i[k] = 1 \end{cases} & 1 \leq k \leq d, & \text{CG}(a_i, d+1) &:= \mathbf{0}_x, \\ \text{CG}(b_j, k) &:= \begin{cases} \mathbf{0}_y & \text{if } b_j[k] = 0 \\ \mathbf{1}_y & \text{if } b_j[k] = 1 \end{cases} & 1 \leq k \leq d, & \text{CG}(b_j, d+1) &:= \mathbf{1}_y. \end{aligned}$$

Note that we have $\text{type}(\text{CG}(a_i, 1)) = \dots = \text{type}(\text{CG}(a_i, d+1)) =: t_x$ and $\text{type}(\text{CG}(b_j, 1)) = \dots = \text{type}(\text{CG}(b_j, d+1)) =: t_y$ by definition of coordinate values. This allows us to use the alignment gadget to obtain the following *vector gadgets*

$$\begin{aligned} \text{VG}(a_i) &:= \text{GA}_x^{d+1, t_y}(\text{CG}(a_i, 1), \dots, \text{CG}(a_i, d+1)), \\ \text{VG}(b_j) &:= \text{GA}_y^{d+1, t_x}(\text{CG}(b_j, 1), \dots, \text{CG}(b_j, d+1)), \\ S &:= \text{GA}_x^{d+1, t_y}(\underbrace{\mathbf{0}_x, \dots, \mathbf{0}_x, \mathbf{1}_x}_{d+1}), \end{aligned}$$

Note that $\text{type}(\text{VG}(a_1)) = \dots = \text{type}(\text{VG}(a_n)) = \text{type}(S) =: t'_x$ and $\text{type}(\text{VG}(b_1)) = \dots = \text{type}(\text{VG}(b_m)) =: t'_y$, because the type of the output of the alignment gadget only depends on the number of input elements and their type, which are all t_x or all t_y , respectively. We introduce *normalized vector gadgets* as follows

$$\begin{aligned} \text{NVG}(a_i) &:= \text{GA}_x^{1, t'_y}(S, \text{VG}(a_i)), \\ \text{NVG}(b_j) &:= \text{GA}_y^{2, t'_x}(\text{VG}(b_j)). \end{aligned}$$



(a) Case $\langle a_i, b_j \rangle = 0$. Aligning $\text{VG}(b_j)$ with $\text{VG}(a_i)$ achieves an alignment cost of $(d + 1)\rho_0$.

(b) Case $\langle a_i, b_j \rangle > 0$. Aligning $\text{VG}(b_j)$ with S achieves an alignment cost of $d\rho_0 + \rho_1$.

Figure 2: Schematic illustration of the coordinate, vector, and normalized vector gadgets.

Note that we have $\text{type}(\text{NVG}(a_1)) = \dots = \text{type}(\text{NVG}(a_n)) =: t_x''$ and $\text{type}(\text{NVG}(b_1)) = \dots = \text{type}(\text{NVG}(b_m)) =: t_y''$. We finally obtain x and y by setting

$$\begin{aligned} x &:= \text{GA}_x^{m, t_y''}(\text{NVG}(a_1), \dots, \text{NVG}(a_n), \text{NVG}(a_1), \dots, \text{NVG}(a_n)), \\ y &:= \text{GA}_y^{2n, t_x''}(\text{NVG}(b_1), \dots, \text{NVG}(b_m)). \end{aligned}$$

We denote by C, C', C'' the value C in the three invocations of Property (2) of the alignment gadget.

Observe that x and y have length $\mathcal{O}((n + m)d)$ and can be constructed in time $\mathcal{O}((n + m)d)$ by applying the algorithm implicit in Definition 3.1 three times. Moreover, if δ admits an *unbalanced* alignment gadget, then we have $|x| = \mathcal{O}(nd)$ and $|y| = \mathcal{O}(md)$. It remains to show that if we know $\delta(x, y)$ then we can decide the given OV instance in constant time, i.e., correctness of our construction, which we do below. This finishes our reduction from OV to the problem of computing δ . To obtain Theorem 3.3, let $0 < \alpha \leq 1$ and assume that $\delta(x', y')$ can be computed in time $\mathcal{O}(M^{2-\varepsilon})$ whenever $|x'| \leq N$, $|y'| \leq M$, and $N^{\alpha-o(1)} \leq M \leq N^{\alpha+o(1)}$. Then in particular for $n = m$ we can compute $\delta(x, y)$ in time $\mathcal{O}(\min\{|x|, |y|\}^{2-\varepsilon} + |x| + |y|) = \mathcal{O}(((n + m)d)^{2-\varepsilon}) = \mathcal{O}((nd)^{2-\varepsilon})$, contradicting OVH. In case of an unbalanced alignment gadget, assume that $\delta(x', y')$ can be computed in time $\mathcal{O}((NM)^{1-\varepsilon})$ whenever $|x'| \leq N$, $|y'| \leq M$, and $N^{\alpha-o(1)} \leq M \leq N^{\alpha+o(1)}$. Then for $m = \Theta(n^\alpha)$ and $d \leq n^{o(1)}$ we can compute $\delta(x, y)$ in time $\mathcal{O}((|x||y|)^{1-\varepsilon} + |x| + |y|) = \mathcal{O}(((nd)(md))^{1-\varepsilon} + (n + m)d) = \mathcal{O}((nm)^{1-\varepsilon/2})$, contradicting UOVH. This proves Theorem 3.3.

Correctness We now prove correctness of our construction and refer to Figure 2 for an intuition for coordinate, vector, and normalized vector gadgets. Let $\rho_0 := \delta(\mathbf{0}_x, \mathbf{0}_y) = \delta(\mathbf{0}_x, \mathbf{1}_y) = \delta(\mathbf{1}_x, \mathbf{0}_y)$ and $\rho_1 := \delta(\mathbf{1}_x, \mathbf{1}_y)$. Recall that $\rho_0 < \rho_1$.

Claim 3.4. For any $i \in [n]$, $j \in [m]$, if $\langle a_i, b_j \rangle = 0$, then $\delta(\text{VG}(a_i), \text{VG}(b_j)) = C + (d + 1)\rho_0$. Otherwise, $\delta(\text{VG}(a_i), \text{VG}(b_j)) \geq C + d\rho_0 + \rho_1$. Moreover, $\delta(S, \text{VG}(b_j)) = C + d\rho_0 + \rho_1$.

Proof. If $\langle a_i, b_j \rangle = 0$, then the structured alignment $\{(1, 1), \dots, (d + 1, d + 1)\}$ has cost $\delta(A) = \sum_{k=1}^{d+1} \delta(\text{CG}(a_i, k), \text{CG}(b_j, k)) = (d + 1)\rho_0$, since in each component k at least one value is $\mathbf{0}_x$ or $\mathbf{0}_y$, incurring a cost of ρ_0 (indeed even in position $k = d + 1$ we have $\text{CG}(a_i, d + 1) = \mathbf{0}_x$). By definition of alignment gadgets, we obtain $\delta(\text{VG}(a_i), \text{VG}(b_j)) - C \leq (d + 1)\rho_0$. Moreover, since the

cost $\delta(A)$ of any alignment $A \in \mathcal{A}_{d+1,d+1}$ consists of $d+1$ summands of the form $\delta(u_x, u_y)$ with $u_x \in \{\mathbf{0}_x, \mathbf{1}_x\}, u_y \in \{\mathbf{0}_y, \mathbf{1}_y\}$, we also have $\delta(\text{VG}(a_i), \text{VG}(b_j)) - C \geq (d+1)\rho_0$.

If $\langle a_i, b_j \rangle > 0$, then consider any $A \in \mathcal{A}_{n,m}$. If $|A| = d+1$ then $A = \{(1, 1), \dots, (d+1, d+1)\}$, and this alignment incurs a cost of at least $d\rho_0 + \rho_1$, since in at least one position k we have $\text{CG}(a_i, k) = \mathbf{1}_x$ and $\text{CG}(b_j, k) = \mathbf{1}_y$. Otherwise, if $|A| < d+1$, then $\delta(A)$ consists of $d+1$ summands of the form $\delta(u_x, u_y)$ with $u_x \in \{\mathbf{0}_x, \mathbf{1}_x\}, u_y \in \{\mathbf{0}_y, \mathbf{1}_y\}$, and at least one of these summands is the punishment term $\max_{k,\ell} \delta(\text{CG}(a_i, k), \text{CG}(b_j, \ell))$ because $|A| < d+1$. Since $\langle a_i, b_j \rangle = 1$, the punishment term is ρ_1 and we obtain $\delta(A) \geq d\rho_0 + \rho_1$. By definition of alignment gadgets, we have $\delta(\text{VG}(a_i), \text{VG}(b_j)) - C \geq d\rho_0 + \rho_1$.

We argue similarly for $\delta(S, \text{VG}(b_j))$: The alignment $\{(1, 1), \dots, (d+1, d+1)\}$ incurs a cost of $d\rho_0 + \rho_1$, since the $(d+1)$ -th component of S is $\mathbf{1}_x$ and of $\text{VG}(b_j)$ is $\text{VG}(b_j, d+1) = \mathbf{1}_y$, while all other components of S are $\mathbf{0}_x$. Moreover, any alignment with $|A| < d+1$ incurs a punishment term, so that it incurs cost of at least $d\rho_0 + \rho_1$. \square

Claim 3.5. For any $i \in [n], j \in [m]$, if $\langle a_i, b_j \rangle = 0$ then $\delta(\text{NVG}(a_i), \text{NVG}(b_j)) = C + C' + (d+1)\rho_0 =: \rho'_0$. Otherwise, $\delta(\text{NVG}(a_i), \text{NVG}(b_j)) = C + C' + d\rho_0 + \rho_1 =: \rho'_1$.

Proof. Note that $\{(1, 1)\}, \{(2, 1)\}$, and \emptyset are the only alignments in $\mathcal{A}_{2,1}$, which corresponds to aligning $(S, \text{VG}(b_j))$ or $(\text{VG}(a_i), \text{VG}(b_j))$ or nothing. Moreover, the structured alignments are $\{(1, 1)\}$ and $\{(2, 1)\}$. Observe that the cost of the alignment \emptyset is simply the maximum of the other two alignments. By Claim 3.4, if $\langle a_i, b_j \rangle = 0$ then the minimal cost is $C + (d+1)\rho_0$, attained by alignment $\{(2, 1)\}$. Otherwise, the minimal cost is $C + d\rho_0 + \rho_1$, attained by alignment $\{(1, 1)\}$. By definition of alignment gadgets, this yields that $\delta(\text{NVG}(a_i), \text{NVG}(b_j)) - C'$ is equal to $C + (d+1)\rho_0$ or $C + d\rho_0 + \rho_1$, respectively. \square

The claim shows that $\delta(\text{NVG}(a_i), \text{NVG}(b_j))$ attains one of only two values, depending on whether $\langle a_i, b_j \rangle = 0$.

Claim 3.6. If there is no $i \in [n], j \in [m]$ with $\langle a_i, b_j \rangle = 0$, then $\delta(x, y) \geq C'' + m\rho'_1$. Otherwise, $\delta(x, y) \leq C'' + (m-1)\rho'_1 + \rho'_0$.

Proof. If $\langle a_i, b_j \rangle > 0$ for all i, j , then by the previous claim we have $\delta(\text{NVG}(a_i), \text{NVG}(b_j)) \geq \rho'_1$ for all i, j . Since the cost of any alignment consists of m summands of the form $\delta(\text{NVG}(a_i), \text{NVG}(b_j))$ for some i, j , the cost of any alignment is at least $m\rho'_1$. By definition of alignment gadgets, we obtain $\delta(x, y) - C'' \geq m\rho'_1$.

If $\langle a_i, b_j \rangle = 0$ for some i, j , then consider the structured alignment $A = \{(\Delta+1, 1), \dots, (\Delta+m, m)\}$ with $\Delta := i-j$ if $i \geq j$, or $\Delta := n+i-j$ if $i < j$. Its cost consists of m summands, of which one is $\delta(\text{NVG}(a_i), \text{NVG}(b_j)) = \rho'_0$ and all others are at most ρ'_1 . Hence, the cost of A is at most $(m-1)\rho'_1 + \rho'_0$ and by definition of alignment gadgets, we obtain $\delta(x, y) - C'' \leq (m-1)\rho'_1 + \rho'_0$. \square

By setting $\rho := C'' + (m-1)\rho'_1 + \rho'_0$ we have found a threshold such that $\delta(x, y) \leq \rho$ if and only if there is a pair (i, j) with $\langle a_i, b_j \rangle = 0$. Thus, computing $\delta(x, y)$ allows to decide the given OV instance. This finishes the proof of Theorem 3.3.

4 Longest Common Subsequence

In this section, we present an alternative hardness proof for longest common subsequence (LCS), which is shorter than for the more general problem $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ in Section 5.

Given two strings x, y over an alphabet Σ , a longest common subsequence is a binary string z that appears in x and in y as a subsequence and has maximal length. We denote by $\text{LCS}(x, y)$ some longest common subsequence of x and y , and by $|\text{LCS}(x, y)|$ the length of any longest common subsequence of x and y .

We present an alignment gadget and coordinate values for LCS over binary strings, i.e., we consider the set of inputs $\mathcal{I} := \bigcup_{k \geq 0} \{0, 1\}^k$. Note that LCS is a maximization problem, but Definition 3.1 implicitly assumes a minimization problem, so we instead consider the number of unmatched symbols $\delta_{\text{LCS}}(x, y) := |x| + |y| - 2|\text{LCS}(x, y)|$ for binary strings x, y . Note that this is equivalent to $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ for $c_{\text{del-x}} = c_{\text{del-y}} = 1$, $c_{\text{match}} = 0$, and $c_{\text{subst}} = 2$.

Lemma 4.1. *LCS admits coordinate values by setting*

$$\mathbf{1}_x := 11100, \mathbf{0}_x := 10011, \mathbf{1}_y := 00111, \mathbf{0}_y := 11001.$$

Proof. All four values have the same length and the same number of 1s, so they have equal type. Short calculations show that $\text{LCS}(\mathbf{1}_x, \mathbf{1}_y) = 111$, $\text{LCS}(\mathbf{1}_x, \mathbf{0}_y) = 1100$, $\text{LCS}(\mathbf{0}_x, \mathbf{1}_y) = 0011$, and $\text{LCS}(\mathbf{0}_x, \mathbf{0}_y) = 1001$. Thus, $4 = \delta_{\text{LCS}}(\mathbf{1}_x, \mathbf{1}_y) > \delta_{\text{LCS}}(\mathbf{1}_x, \mathbf{0}_y) = \delta_{\text{LCS}}(\mathbf{0}_x, \mathbf{1}_y) = \delta_{\text{LCS}}(\mathbf{0}_x, \mathbf{0}_y) = 2$. \square

Definition 4.2. Consider instances $x_1, \dots, x_n \in \mathcal{I}_{t_x}$ and $y_1, \dots, y_m \in \mathcal{I}_{t_y}$ with $n \geq m$ and types $t_x = (\ell_x, s_x), t_y = (\ell_y, s_y)$. Set $\gamma_1 := \ell_x + \ell_y, \gamma_2 := 6(\ell_x + \ell_y), \gamma_3 := 10(\ell_x + \ell_y) + 2s_x - \ell_x, \gamma_4 := 13(\ell_x + \ell_y)$. We *guard* the input strings by blocks of zeroes and ones, setting $G(z) := 1^{\gamma_2} 0^{\gamma_1} z 0^{\gamma_1} 1^{\gamma_2}$. We define the alignment gadget as

$$\begin{aligned} x &:= G(x_1) 0^{\gamma_3} G(x_2) 0^{\gamma_3} \dots G(x_{n-1}) 0^{\gamma_3} G(x_n), \\ y &:= 0^{n\gamma_4} G(y_1) 0^{\gamma_3} G(y_2) 0^{\gamma_3} \dots G(y_{m-1}) 0^{\gamma_3} G(y_m) 0^{n\gamma_4}. \end{aligned}$$

Lemma 4.3. *Definition 4.2 realizes an alignment gadget for LCS.*

Thus, Theorem 3.3 is applicable, implying a lower bound of $\mathcal{O}(m^{2-\varepsilon})$ for LCS. We remark that our construction is no *unbalanced* alignment gadget, as the length of y grows linearly in n , not necessarily in $m \leq n$. Thus, we do not obtain a conditional lower bound of $\mathcal{O}((nm)^{1-\varepsilon})$ (for $m \approx n^\alpha$ for any $0 < \alpha < 1$).

Proof of Lemma 4.3. Observe that indeed x only depends on m, t_y , and x_1, \dots, x_n , and $\text{type}(x)$ only depends on n, m, t_x , and t_y , and similarly for y . Moreover, x and y can clearly be constructed in time $\mathcal{O}((n+m)(\ell_x + \ell_y))$, where $\ell_x = |x_1| = \dots = |x_n|$ and $\ell_y = |y_1| = \dots = |y_m|$.

It remains to prove that by setting $C := 2n\gamma_4$ we have

$$\min_{A \in \mathcal{A}_{n,m}} \delta(A) \leq \delta(x, y) - C \leq \min_{A \in \mathcal{S}_{n,m}} \delta(A). \quad (3)$$

We first prove the following three useful observations. Here for a string z and indices $a \leq b$ we denote the substring from $z[a]$ to $z[b]$ by $z[a..b]$.

Claim 4.4. Let x and z_1, \dots, z_k be binary strings. Set $z = z_1 \dots z_n$. Then we have

$$\delta_{\text{LCS}}(x, z) = \min_{x(z_1), \dots, x(z_k)} \sum_{j=1}^k \delta_{\text{LCS}}(x(z_j), z_j),$$

where $x(z_1), \dots, x(z_k)$ range over all *ordered partitions* of x into k substrings, i.e., $x(z_1) = x[i_0 + 1..i_1], x(z_2) = x[i_1 + 1..i_2], \dots, x(z_k) = x[i_{k-1} + 1..i_k]$ for any $0 = i_0 \leq i_1 \leq \dots \leq i_k = |x|$.

Proof. For any ordered partition, the substrings $x(z_j)$ are disjoint and ordered along x , so we can take the longest common subsequences of $(x(z_j), z_j)$, $j \in [k]$, and concatenate them to form a common subsequence of (x, z) . This shows $|\text{LCS}(x, z)| \geq \sum_{j=1}^k |\text{LCS}(x(z_j), z_j)|$. Since furthermore $|x| = \sum_{j=1}^k |x(z_j)|$ and $|z| = \sum_{j=1}^k |z_j|$ we obtain $\delta_{\text{LCS}}(x, z) \leq \sum_{j=1}^k \delta_{\text{LCS}}(x(z_j), z_j)$.

Now consider a longest common subsequence s of (x, y) , which can be seen as a matching of symbols in x and y . Let J_j be the indices in x that are matched to symbols in z_j by s . Note that $\sum_{j=1}^k |J_j| = |\text{LCS}(x, y)|$, as any matched symbol in x is matched to some z_j . Also note that the matching is ordered, meaning that for any $i \in J_j$ and $i' \in J_{j'}$ with $j < j'$ we have $i < i'$. This allows to find an ordered partition $x(z_1), \dots, x(z_k)$ of x such that $x(z_j)$ contains the indices J_j for any j . Finally, for this partition we have $|\text{LCS}(x(z_j), z_j)| \geq |J_j|$ so that $\delta_{\text{LCS}}(x(z_j), z_j) \leq |x(z_j)| + |z_j| - 2|J_j|$. Summing up over j , we obtain $\sum_{j=1}^k \delta_{\text{LCS}}(x(z_j), z_j) \leq |x| + |z| - 2|\text{LCS}(x, z)| = \delta_{\text{LCS}}(x, z)$. Together both halves of the proof imply the desired statement. \square

Claim 4.5. Let z, w be binary strings and $\ell, k \in \mathbb{N}_0$. Then we have (1) $\delta_{\text{LCS}}(1^k z, 1^k w) = \delta_{\text{LCS}}(z, w)$ and (2) $\delta_{\text{LCS}}(0^\ell z, 1^k w) \geq \min\{k, \delta_{\text{LCS}}(z, 1^k w)\}$. Symmetrically, we have (2') $\delta_{\text{LCS}}(0^k z, 1^\ell w) \geq \min\{k, \delta_{\text{LCS}}(0^k z, w)\}$, and we obtain more symmetric statements by reversing all involved strings.

Proof. (1) It suffices to show the claim for $k = 1$, then the general statement follows by induction. Consider a LCS s of $(1z, 1w)$. At least one '1' is matched in s , as otherwise we can extend s by matching both '1's. If exactly one '1' is matched in s , then the other '1' is free, so we may instead match the two '1's. Thus, without loss of generality a LCS of $(1z, 1w)$ matches the two '1's. This yields $|\text{LCS}(1z, 1w)| = 1 + |\text{LCS}(z, w)|$. Hence, $\delta_{\text{LCS}}(1z, 1w) = |1z| + |1w| - 2|\text{LCS}(1z, 1w)| = |z| + |w| - 2|\text{LCS}(z, w)| = \delta_{\text{LCS}}(z, w)$.

(2) Fix a LCS s of $(0^\ell z, 1^k w)$. If s starts with a 0, then it does not contain the leading 1^k of the second argument, leaving at least k symbols unmatched, so that $\delta_{\text{LCS}}(0^\ell z, 1^k w) \geq k$. Otherwise, if s starts with a 1, then it does not contain the leading 0^ℓ of the first argument, so that $|\text{LCS}(0^\ell z, 1^k w)| = |\text{LCS}(z, 1^k w)|$. Then we have $\delta_{\text{LCS}}(0^\ell z, 1^k w) = |0^\ell z| + |1^k w| - 2|\text{LCS}(0^\ell z, 1^k w)| \geq |z| + |1^k w| - 2|\text{LCS}(z, 1^k w)| = \delta_{\text{LCS}}(z, 1^k w)$. \square

Claim 4.6. Let $\ell \geq 0$. For any prefix x' of x we have $\delta_{\text{LCS}}(x', 0^\ell) \geq \ell$. Moreover, if x' is of the form $G(x_1)0^{\gamma_3} \dots G(x_i)0^{\gamma_3}$ for some $0 \leq i < n$ and $\ell \geq i \cdot (2\gamma_2 + s_x)$, then $\delta_{\text{LCS}}(x', 0^\ell) = \ell$. Symmetric statements hold for any suffix of x .

Proof. We first show that for any $i \in [n]$ the string $G(x_i)0^{\gamma_3}$ contains as many ones as zeroes, and any prefix of $G(x_i)0^{\gamma_3}$ contains at least as many ones as zeroes. To this end, note that each x_i has length ℓ_x and contains s_x ones, so that the number of ones of $G(x_i)0^{\gamma_3}$ is $2\gamma_2 + s_x$, while the number of zeroes is $\ell_x - s_x + 2\gamma_1 + \gamma_3$, and we chose γ_3 such that both values are equal. For a prefix, note that $G(x_i)$ starts with γ_2 ones. Since each $G(x_i)$ contains $2\gamma_1 + \ell_x - s_x \leq \gamma_2$ zeroes, any prefix of $G(x_i)$ has as most as many zeroes as ones. Thus, we would have to advance to 0^{γ_3} to see more zeroes than ones, however, even $G(x_i)0^{\gamma_3}$ does not contain more zeroes than ones.

Hence, any prefix x' of x contains at least as many ones as zeroes, implying $|\text{LCS}(x', 0^\ell)| \leq |x'|/2$. This yields $\delta_{\text{LCS}}(x', 0^\ell) = |x'| + |0^\ell| - 2|\text{LCS}(x', 0^\ell)| \geq \ell$. If x' is of the form $G(x_1)0^{\gamma_3} \dots G(x_i)0^{\gamma_3}$ and sufficiently many zeroes are available in 0^ℓ then we have equality. \square

Let us give names to the substrings consisting only of zeroes in x and y . In x , we denote the 0^{γ_3} -block after $G(x_i)$ by Z_i^X , $i \in [n-1]$. In y , we denote the 0^{γ_3} -block after $G(y_j)$ by Z_j^Y , $j \in [m-1]$. Moreover, we denote the prefix $0^{n\gamma_4}$ by L^Y and the suffix $0^{n\gamma_4}$ by R^Y .

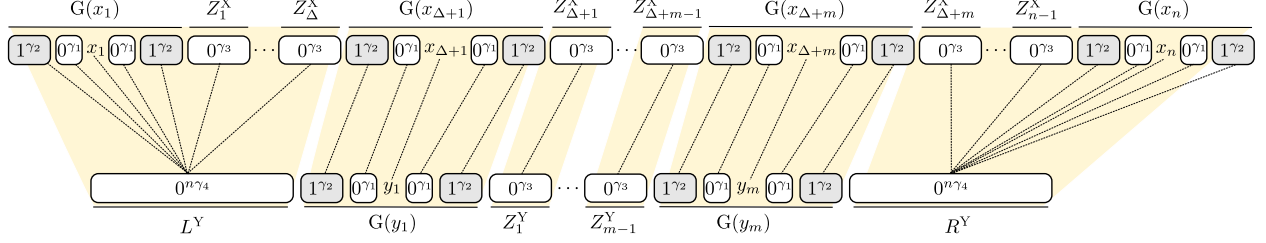


Figure 3: Optimal traversal corresponding to structured alignment $A = \{(\Delta+j, j) \mid j \in [m]\} \in \mathcal{S}_{n,m}$.

We now show the upper bound of (3), i.e., $\delta_{\text{LCS}}(x, y) \leq 2n\gamma_4 + \min_{A \in \mathcal{S}_{n,m}} \delta(A)$. Consider a structured alignment $A = \{(\Delta+1, 1), \dots, (\Delta+m, m)\} \in \mathcal{S}_{n,m}$. We construct an ordered partition of x as in Claim 4.4 by setting (see Figure 3)

$$\begin{aligned} x(\text{G}(y_j)) &:= \text{G}(x_{\Delta+j}) \quad \text{for } j \in [m], \\ x(Z_j^Y) &:= Z_{\Delta+j}^X \quad \text{for } j \in [m-1], \\ x(L^Y) &:= \text{G}(x_1)Z_1^X \dots \text{G}(x_{\Delta})Z_{\Delta}^X, \\ x(R^Y) &:= Z_{\Delta+m}^X \text{G}(x_{\Delta+m+1}) \dots Z_{n-1}^X \text{G}(x_n). \end{aligned}$$

Note that indeed these strings partition x and y , respectively. Thus, Claim 4.4 yields

$$\delta_{\text{LCS}}(x, y) \leq \delta_{\text{LCS}}(x(L^Y), L^Y) + \delta_{\text{LCS}}(x(R^Y), R^Y) + \sum_{j=1}^m \delta_{\text{LCS}}(\text{G}(x_{\Delta+j}), \text{G}(y_j)) + \sum_{j=1}^{m-1} \delta_{\text{LCS}}(Z_{\Delta+j}^X, Z_j^Y).$$

Since $L^Y = 0^{n\gamma_4}$ and $x(L^Y)$ is a prefix of x of the correct form, by Claim 4.6 we have $\delta_{\text{LCS}}(x(L^Y), L^Y) = n\gamma_4$ (note that γ_4 is chosen sufficiently large to make Claim 4.6 applicable). Similarly we obtain $\delta_{\text{LCS}}(x(R^Y), R^Y) = n\gamma_4$. Since $Z_i^X = Z_j^Y = 0^{\gamma_3}$ we have $\delta_{\text{LCS}}(Z_{\Delta+j}^X, Z_j^Y) = 0$. Finally, by matching the guarding ones and zeroes of $\text{G}(x_{\Delta+j}) = 1^{\gamma_2}0^{\gamma_1}x_{\Delta+j}0^{\gamma_1}1^{\gamma_2}$ and $\text{G}(y_j) = 1^{\gamma_2}0^{\gamma_1}y_j0^{\gamma_1}1^{\gamma_2}$ we obtain $\delta_{\text{LCS}}(\text{G}(x_{\Delta+j}), \text{G}(y_j)) \leq \delta_{\text{LCS}}(x_{\Delta+j}, y_j)$. Hence, we have

$$\delta_{\text{LCS}}(x, y) \leq 2n\gamma_4 + \sum_{(i,j) \in A} \delta_{\text{LCS}}(x_i, y_j).$$

As $A \in \mathcal{S}_{n,m}$ was arbitrary, we proved $\delta_{\text{LCS}}(x, y) \leq 2n\gamma_4 + \min_{A \in \mathcal{S}_{n,m}} \delta(A)$, as desired.

It remains to prove the lower bound of (3), i.e., $\delta_{\text{LCS}}(x, y) \geq 2n\gamma_4 + \min_{A \in \mathcal{A}_{n,m}} \delta(A)$. As in Claim 4.4, let $x(L^Y)$, $x(\text{G}(y_j))$ for $j \in [m]$, $x(Z_j^Y)$ for $j \in [m-1]$, $x(R^Y)$ be an ordered partition of x such that

$$\delta_{\text{LCS}}(x, y) = \delta_{\text{LCS}}(x(L^Y), L^Y) + \delta_{\text{LCS}}(x(R^Y), R^Y) + \sum_{j=1}^m \delta_{\text{LCS}}(x(\text{G}(y_j)), \text{G}(y_j)) + \sum_{j=1}^{m-1} \delta_{\text{LCS}}(x(Z_j^Y), Z_j^Y).$$

Clearly, we can bound $\delta_{\text{LCS}}(x(Z_j^Y), Z_j^Y) \geq 0$. Since $L^Y = 0^{n\gamma_4}$ and $x(L^Y)$ is a prefix of x , by Claim 4.6 we have $\delta_{\text{LCS}}(x(L^Y), L^Y) \geq n\gamma_4$, and similarly we get $\delta_{\text{LCS}}(x(R^Y), R^Y) \geq n\gamma_4$. It remains to construct an alignment $A \in \mathcal{A}_{n,m}$ satisfying

$$\delta(A) \leq \sum_{j=1}^m \delta_{\text{LCS}}(x(\text{G}(y_j)), \text{G}(y_j)), \quad (4)$$

then together we have shown the desired inequality $\delta_{\text{LCS}}(x, y) \geq 2n\gamma_4 + \min_{A \in \mathcal{A}_{n,m}} \delta(A)$.

Let us construct such an alignment A . For any $j \in [m]$, if $x(\text{G}(y_j))$ contains more than half of some $x_{i'}$ (which is part of $\text{G}(x_{i'})$), then let i be the leftmost such index and align i and j . Note that the set A of all these aligned pairs (i, j) is a valid alignment in $\mathcal{A}_{n,m}$, since no x_i or y_j can be aligned more than once.

Since by definition we have $\delta(A) = \sum_{(i,j) \in A} \delta_{\text{LCS}}(x_i, y_j) + (m - |A|) \max_{i,j} \delta_{\text{LCS}}(x_i, y_j)$ and since $\max_{i,j} \delta_{\text{LCS}}(x_i, y_j) \leq \max_{i,j} (|x_i| + |y_j|) = \ell_x + \ell_y$, in order to show (4) it suffices to prove the following two claims.

Claim 4.7. For any aligned pair $(i, j) \in A$ we have $\delta_{\text{LCS}}(x(\text{G}(y_j)), \text{G}(y_j)) \geq \delta_{\text{LCS}}(x_i, y_j)$.

Proof. Recall that $x(\text{G}(y_j))$ contains more than half of x_i . First consider the case that $x(\text{G}(y_j))$ touches not only $\text{G}(x_i)$ but also $\text{G}(x_{i'})$ for some $i' \neq i$. As between x_i and $\text{G}(x_{i'})$ there is at least one block of zeroes 0^{γ_3} and half of the guarding of $\text{G}(x_i)$ (i.e., $1^{\gamma_2}0^{\gamma_1}$ or $0^{\gamma_1}1^{\gamma_2}$), we obtain $|x(\text{G}(y_j))| \geq |0^{\gamma_3}| + |1^{\gamma_2}0^{\gamma_1}| = \gamma_3 + \gamma_2 + \gamma_1$. Thus, any matching of $x(\text{G}(y_j))$ and $\text{G}(y_j)$ leaves at least $|x(\text{G}(y_j))| - |\text{G}(y_j)| \geq (\gamma_3 + \gamma_2 + \gamma_1) - (2\gamma_2 + 2\gamma_1 + \ell_y) = \gamma_3 - \gamma_2 - \gamma_1 - \ell_y \geq \ell_x + \ell_y$ unmatched symbols, implying $\delta_{\text{LCS}}(x(\text{G}(y_j)), \text{G}(y_j)) \geq \ell_x + \ell_y \geq \delta_{\text{LCS}}(x_i, y_j)$.

Now consider the remaining case, where $x(\text{G}(y_j))$ touches no other $\text{G}(x_{i'})$. In this case, $x(\text{G}(y_j))$ is a substring of $0^{\gamma_3}\text{G}(x_i)0^{\gamma_3}$, i.e., we can write $x(\text{G}(y_j))$ as $0^{h_L}z0^{h_R}$, where z is a substring of $\text{G}(x_i)$. Since $\text{G}(y_j)$ starts with γ_2 ones, by Claim 4.5.(2) we have $\delta_{\text{LCS}}(x(\text{G}(y_j)), \text{G}(y_j)) \geq \min\{\gamma_2, \delta_{\text{LCS}}(z0^{h_R}, \text{G}(y_j))\}$. Since $\gamma_2 \geq \ell_x + \ell_y \geq \delta_{\text{LCS}}(x_i, y_j)$, it suffices to bound $\delta_{\text{LCS}}(z0^{h_R}, \text{G}(y_j))$ from below. By a symmetric argument, we eliminate the block 0^{h_R} and only have to bound $\delta_{\text{LCS}}(z, \text{G}(y_j))$ from below. We can assume that $|z| > |\text{G}(y_j)| - \gamma_2$, since otherwise $\delta_{\text{LCS}}(z, \text{G}(y_j)) \geq \gamma_2 \geq \ell_x + \ell_y \geq \delta_{\text{LCS}}(x_i, y_j)$. Thus, we have $\text{G}(y_j) = 1^{\gamma_2}0^{\gamma_1}y_j0^{\gamma_1}1^{\gamma_2}$ and can write z as $1^{r_L}0^{\gamma_1}x_i0^{\gamma_1}1^{r_R}$ with $r_L, r_R > 0$. By Claim 4.5.(1) we have $\delta_{\text{LCS}}(z, \text{G}(y_j)) = \delta_{\text{LCS}}(0^{\gamma_1}x_i0^{\gamma_1}1^{r_R}, 1^{\gamma_2-r_L}0^{\gamma_1}y_j0^{\gamma_1}1^{\gamma_2})$. By Claim 4.5.(2'), this yields $\delta_{\text{LCS}}(z, \text{G}(y_j)) \geq \min\{\gamma_1, \delta_{\text{LCS}}(0^{\gamma_1}x_i0^{\gamma_1}1^{r_R}, 0^{\gamma_1}y_j0^{\gamma_1}1^{\gamma_2})\}$, and since $\gamma_1 \geq \ell_x + \ell_y \geq \delta_{\text{LCS}}(x_i, y_j)$ it suffices to bound the latter term. By a symmetric argument we eliminate the ones on the right side, and it suffices to bound $\delta_{\text{LCS}}(0^{\gamma_1}x_i0^{\gamma_1}, 0^{\gamma_1}y_j0^{\gamma_1})$. Using Claim 4.5.(1) twice, this is equal to $\delta_{\text{LCS}}(x_i, y_j)$. Hence, we have shown the desired inequality $\delta_{\text{LCS}}(x(\text{G}(y_j)), \text{G}(y_j)) \geq \delta_{\text{LCS}}(x_i, y_j)$. \square

Claim 4.8. If j is unaligned in A , then $\delta_{\text{LCS}}(x(\text{G}(y_j)), \text{G}(y_j)) \geq \ell_x + \ell_y$.

Proof. Since $x(\text{G}(y_j))$ contains less than half of any x_i , examining the structure of x we see that $x(\text{G}(y_j))$ is a substring² of $P := x_i0^{\gamma_1}1^{\gamma_2}0^{\gamma_3}1^{\gamma_2}0^{\gamma_1}x_{i+1}$ for some $1 \leq i < n$, where at most half of x_i and x_{i+1} can be part of $x(\text{G}(y_j))$. If $x(\text{G}(y_j))$ contains ones to the left and to the right of 0^{γ_3} in P , then $x(\text{G}(y_j))$ contains at least γ_3 zeroes. Since $\text{G}(y_j)$ contains $2\gamma_1 + \ell_y - s_y \leq 2\gamma_1 + \ell_y$ zeroes, at most $2\gamma_1 + \ell_y$ zeroes of $x(\text{G}(y_j))$ can be matched, leaving at least $\gamma_3 - 2\gamma_1 - \ell_y$ unmatched zeroes. Thus, $\delta_{\text{LCS}}(x(\text{G}(y_j)), \text{G}(y_j)) \geq \gamma_3 - 2\gamma_1 - \ell_y \geq \ell_x + \ell_y$. Otherwise, if $x(\text{G}(y_j))$ contains only ones to the left of 0^{γ_3} in P (or only to the right), then $x(\text{G}(y_j))$ contains at most $\gamma_2 + \ell_x$ ones. Thus, among the $2\gamma_2 + s_y \geq 2\gamma_2$ ones of $\text{G}(y_j)$ at least $\gamma_2 - \ell_x$ ones remain unmatched, implying $\delta_{\text{LCS}}(x(\text{G}(y_j)), \text{G}(y_j)) \geq \gamma_2 - \ell_x \geq \ell_x + \ell_y$. \square

This finishes the proof of Lemma 4.3. \square

²Actually $x(\text{G}(y_j))$ could also be a substring of $1^{\gamma_2}0^{\gamma_1}x_1$ or of $x_n0^{\gamma_1}1^{\gamma_2}$. We treat these border cases by setting $x_0 := x_1$ and $x_{n+1} := x_n$ and letting from now on $0 \leq i \leq n$.

5 Edit Distance

We first show that the trivial cases of $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ can be solved in constant time. For all other cases, on binary strings we present a reduction from $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ to $\text{Edit}(c'_{\text{subst}})$ and vice versa, see Section 5.1. Then in Section 5.2 we prove a conditional lower bound of $\mathcal{O}(m^{2-\varepsilon})$ for $\text{Edit}(c_{\text{subst}})$ by applying our alignment-framework. Finally, in Section 5.3 we show that $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ can be solved in time $\tilde{\mathcal{O}}(n+m^2)$, which matches our lower bound.

5.1 Easy Reductions

All of our reductions are of the following form. Let $E_1 = \text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ and $E_2 = \text{Edit}(c'_{\text{del-x}}, c'_{\text{del-y}}, c'_{\text{match}}, c'_{\text{subst}})$ be two variants of the edit distance and denote the cost of any traversal T with respect to E_i by $\delta_{E_i}(T)$. We say that E_1 and E_2 are *equivalent*, if there are constants α, β such that for any traversal T we have $\delta_{E_1}(T) = \alpha \cdot \delta_{E_2}(T) + \beta$. Then the complexity of computing E_1 and E_2 is asymptotically equal.

Lemma 5.1. (1) $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ can be solved in constant time if $c_{\text{subst}} = c_{\text{match}}$ or $c_{\text{del-x}} + c_{\text{del-y}} \leq \min\{c_{\text{match}}, c_{\text{subst}}\}$. Otherwise, $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ on binary strings is equivalent to $\text{Edit}(c'_{\text{subst}})$ on binary strings for some $0 < c'_{\text{subst}} \leq 2$.

(2) $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ is equivalent to $\text{Edit}(c'_{\text{del-x}}, c'_{\text{del-y}}, c'_{\text{match}}, c'_{\text{subst}})$ for some positive integers $c'_{\text{del-x}}, c'_{\text{del-y}}, c'_{\text{match}}, c'_{\text{subst}}$.

Note that by the first statement, hardness for general rational cost parameters follows by proving hardness of $\text{Edit}(c'_{\text{subst}})$ for $0 < c'_{\text{subst}} \leq 2$. The second statement allows us to assume positive integer costs when giving an algorithm for $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ in Section 5.3.

Proof of Lemma 5.1. Let x, y be strings of length n, m . By symmetry, we may assume $n \geq m$. Observe that we can write the cost of any traversal T with respect to $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ as

$$\delta_{\text{Edit}}(T) = A \cdot c_{\text{match}} + B \cdot c_{\text{subst}} + C \cdot (c_{\text{del-x}} + c_{\text{del-y}}) + (n - m) \cdot c_{\text{del-x}},$$

for some $A, B, C \geq 0$ with $A + B + C = m$, since matchings and substitutions touch as many symbols in x as in y , so that we need exactly $n - m$ more deletions in x than deletions in y .

(1) If $c_{\text{del-x}} + c_{\text{del-y}} \leq \min\{c_{\text{match}}, c_{\text{subst}}\}$, then we can replace any matching or substitution by a deletion in x and a deletion in y without increasing the cost. Thus, an optimal traversal has $C = m$ and minimal cost $n \cdot c_{\text{del-x}} + m \cdot c_{\text{del-y}}$, which can be computed in constant time. Similarly, if $c_{\text{match}} = c_{\text{subst}}$, then the minimal cost is independent of the symbols in x and y . We may arbitrarily set $A + B$ and C subject to $A + B + C = m$ and $A + B, C \geq 0$, and the minimal cost is $m \cdot \min\{c_{\text{match}}, c_{\text{del-x}} + c_{\text{del-y}}\} + (n - m)c_{\text{del-x}}$, which can be computed in constant time.

Now assume that $c_{\text{match}} \neq c_{\text{subst}}$ and $c_{\text{del-x}} + c_{\text{del-y}} > \min\{c_{\text{match}}, c_{\text{subst}}\}$. Restricting our attention to binary strings, by flipping all symbols in y but not in x we can swap the costs of matching and substitution. Thus, we may assume that $c_{\text{subst}} > c_{\text{match}}$ (and $c_{\text{del-x}} + c_{\text{del-y}} > c_{\text{match}}$). We set

$$c'_{\text{subst}} := \alpha(c_{\text{subst}} - c_{\text{match}}) \quad \text{where} \quad \alpha := \frac{2}{c_{\text{del-x}} + c_{\text{del-y}} - c_{\text{match}}}.$$

One can easily verify that for any traversal T with cost $\delta_{\text{Edit}}(T) = A \cdot c_{\text{match}} + B \cdot c_{\text{subst}} + C \cdot (c_{\text{del-x}} + c_{\text{del-y}}) + (n - m) \cdot c_{\text{del-x}}$ (with respect to $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$) we have

$$\alpha \delta_{\text{Edit}}(T) - \alpha m \cdot c_{\text{match}} + (n - m)(1 - \alpha c_{\text{del-x}}) = B \cdot c'_{\text{subst}} + C \cdot 2 + (n - m).$$

As the latter is the cost of T with respect to $\text{Edit}(c'_{\text{subst}})$, this proves that $\text{Edit}(c'_{\text{subst}})$ is equivalent to $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$. Finally, note that $c'_{\text{subst}} > 0$. If $c'_{\text{subst}} > 2$, then we can replace it by 2 without changing the cost of the optimal traversal, since we can replace any substitution (of cost 2) by a deletion and an insertion (both of cost 1). This yields $0 < c'_{\text{subst}} \leq 2$.

(2) Since we always assume all operation costs to be rationals, without loss of generality $c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}}$ have a common denominator D . We obtain positive integral operation costs by setting $c'_{\text{del-x}} := Dc_{\text{del-x}} + M$, $c'_{\text{del-y}} := Dc_{\text{del-y}} + M$, $c'_{\text{match}} := Dc_{\text{match}} + 2M$, $c'_{\text{subst}} := Dc_{\text{subst}} + 2M$ for a sufficiently large integer M . Both variants are equivalent, since $\delta_{\text{Edit}}(T)$ is changed to

$$D\delta_{\text{Edit}}(T) + m \cdot 2M + (n - m) \cdot M. \quad \square$$

5.2 Hardness Proof

In this section we study the edit distance with matching cost 0, deletion and insertion cost 1, and substitution cost $0 < c_{\text{subst}} \leq 2$. We abbreviate $\delta_{\text{Edit}} = \delta_{\text{Edit}(c_{\text{subst}})}$.

Lemma 5.2. *Edit(c_{subst}) admits coordinate values by setting*

$$\mathbf{1}_X := 11100, \mathbf{0}_X := 10011, \mathbf{1}_Y := 00111, \mathbf{0}_Y := 11001.$$

Proof. All four values have the same length and the same number of ones, so they have equal type. Using Fact 5.5.(1) (to be proven below), we have $\delta_{\text{Edit}}(\mathbf{0}_X, \mathbf{0}_Y) = \delta_{\text{Edit}}(10011, 11001) = \delta_{\text{Edit}}(0011, 1001) = \delta_{\text{Edit}}(001, 100)$. Depending on c_{subst} , the optimal traversal of $(001, 100)$ is either to delete both ones or to substitute the first and last symbols. This yields $\delta_{\text{Edit}}(001, 100) = \min\{2, 2c_{\text{subst}}\}$. Similarly, we obtain $\delta_{\text{Edit}}(\mathbf{1}_X, \mathbf{0}_Y) = \delta_{\text{Edit}}(\mathbf{0}_X, \mathbf{1}_Y) = \delta_{\text{Edit}}(\mathbf{0}_X, \mathbf{0}_Y) = \min\{2, 2c_{\text{subst}}\}$ and $\delta_{\text{Edit}}(\mathbf{1}_X, \mathbf{1}_Y) = \delta_{\text{Edit}}(11100, 00111) = \min\{4, 4c_{\text{subst}}\}$. Hence, $\delta_{\text{Edit}}(\mathbf{1}_X, \mathbf{1}_Y) > \delta_{\text{Edit}}(\mathbf{1}_X, \mathbf{0}_Y) = \delta_{\text{Edit}}(\mathbf{0}_X, \mathbf{1}_Y) = \delta_{\text{Edit}}(\mathbf{0}_X, \mathbf{0}_Y)$. □

Definition 5.3. Consider instances $x_1, \dots, x_n \in \mathcal{I}_{t_X}$ and $y_1, \dots, y_m \in \mathcal{I}_{t_Y}$ with $n \geq m$ and types $t_X = (\ell_X, s_X), t_Y = (\ell_Y, s_Y)$. We define the parameters $\rho := 2\lceil 1/c_{\text{subst}} \rceil$, $\gamma_1 := 10\rho(\ell_X + \ell_Y)$, $\gamma_2 := 6\rho\gamma_1 + 5s_X - \ell_X$, and $\gamma_3 := 2\gamma_2$ (since c_{subst} is constant, these parameters are $\Theta(\ell_X + \ell_Y)$).

To guard a string by blocks of zeroes and ones, we set $G(z) := (1^{\gamma_1}0^{\gamma_1})^\rho z(0^{\gamma_1}1^{\gamma_1})^\rho$. Now the alignment gadget is

$$\begin{aligned} x &:= G(x_1) 0^{\gamma_2} G(x_2) 0^{\gamma_2} \dots G(x_{n-1}) 0^{\gamma_2} G(x_n), \\ y &:= 0^{n\gamma_3} G(y_1) 0^{\gamma_2} G(y_2) 0^{\gamma_2} \dots G(y_{m-1}) 0^{\gamma_2} G(y_m) 0^{n\gamma_3}. \end{aligned}$$

Let us provide some intuition on the complex guarding $G(z)$, which contains more parts compared to the construction for LCS. Consider a block $B = (1^\gamma 0^\gamma)^\rho$. Clearly, B can be completely matched to B , resulting in a cost of 0. Consider a slight perturbation B' of B by prepending Δ ones and deleting the last Δ zeroes. Then the edit distance of B and B' is at most 2Δ , since we may delete the prepended ones in B' and the additional zeroes at the end of B . Another upper bound for the edit distance of B and B' is $2\rho \cdot \Delta c_{\text{subst}}$, since we may match the first γ ones, then substitute the next Δ symbols, then match the next $\gamma - \Delta$ zeroes, and so on. By choosing $\rho := 2\lceil 1/c_{\text{subst}} \rceil$, the traversal using substitutions is more expensive, and indeed we prove that then the edit distance is at least 2Δ . This provides a building block where we got rid of substitutions and where slight perturbations are severely punished. Thus, our guarding $G(z) = (1^{\gamma_1}0^{\gamma_1})^\rho z(0^{\gamma_1}1^{\gamma_1})^\rho$ ensures that an optimal traversal of $G(x)$ and $G(y)$ aligns x and y , and this also holds after small perturbations.

Lemma 5.4. *For any $0 < c_{\text{subst}} \leq 2$, Definition 5.3 realizes an alignment gadget for $\text{Edit}(c_{\text{subst}})$.*

Thus, Theorem 3.3 is applicable, implying a lower bound of $\mathcal{O}(m^{2-\varepsilon})$ for $\text{Edit}(c_{\text{subst}})$. Combining this with Lemma 5.1 proves Theorem 1.2. We remark that our construction is no *unbalanced* alignment gadget, as the length of y grows linearly in n , not necessarily in m . Thus, we do not obtain a conditional lower bound of $\mathcal{O}((nm)^{1-\varepsilon})$ (i.e., not for $m \approx n^\alpha$ for all $0 < \alpha < 1$), which in fact is ruled out by the algorithmic result of Theorem 1.4, see Section 5.3.

In the proof of Lemma 5.4 we make use of the following basic observations.

Fact 5.5. *Let x, y, z be binary strings and $\ell, k \in \mathbb{N}_0$. Then we have (1) $\delta_{\text{Edit}}(1^k x, 1^k y) = \delta_{\text{Edit}}(x, y)$, (2) $\delta_{\text{Edit}}(x, y) \geq ||x| - |y||$ and (3) $|\delta_{\text{Edit}}(xz, y) - \delta_{\text{Edit}}(x, y)| \leq |z|$. We obtain symmetric statements by replacing all 1's by 0's and by reversing all involved strings.*

Proof. We show (1) for $k = 1$, then the general statement follows by induction. Consider an optimal traversal T of $1x, 1y$. If both '1's are deleted in T , then we can instead match them and improve T , contradicting optimality. If exactly one '1' is matched or substituted, then the other '1' is deleted, so we may instead match the two '1's without increasing cost. Thus, without loss of generality an optimal traversal of $(1x, 1y)$ matches the two '1's.

For (2), note that matchings and substitutions touch as many symbols in x as in y . Hence, there have to be at least $|x| - |y|$ deletions in x and at least $|y| - |x|$ deletions in y .

For (3), taking an optimal traversal of (x, y) and appending $|z|$ deletions of the symbols in z shows that $\delta_{\text{Edit}}(xz, y) \leq \delta_{\text{Edit}}(x, y) + |z|$. For the other direction, consider an optimal traversal T of (xz, y) . Replace any matching or substitution of a symbol in z with a symbol $y[j]$ in y by a deletion of $y[j]$. Also remove every deletion of a symbol in z . This results in a traversal T' of (x, y) with cost at most $\delta_{\text{Edit}}(xz, y) + |z|$, as we introduced at most $|z|$ deletions in y . This proves the desired inequality $\delta_{\text{Edit}}(x, y) \leq \delta_{\text{Edit}}(xz, y) + |z|$. \square

Fact 5.6. *Let $\ell, m, r \geq 0$. Then for any $x \in \{0^\ell 1^m 0^r, 1^{m-\ell-r}, 1^{m-\ell} 0^r, 0^\ell 1^{m-r}\}$ we have $\delta_{\text{Edit}}(x, 1^m) \geq |\ell - r| + c_{\text{subst}} \cdot \min\{\ell, r\}$.*

Proof. Fact 5.5.(2) yields $\delta_{\text{Edit}}(0^\ell 1^m 0^r, 1^m), \delta_{\text{Edit}}(1^{m-\ell-r}, 1^m) \geq \ell + r \geq |\ell - r| + c_{\text{subst}} \cdot \min\{\ell, r\}$, since $c_{\text{subst}} \leq 2$. For $x = 0^\ell 1^{m-r}$, consider any optimal traversal T . If T substitutes s zeroes and deletes the remaining $\ell - s$ zeroes, then $\delta_{\text{Edit}}(0^\ell 1^{m-r}, 1^m) = c_{\text{subst}} \cdot s + (\ell - s) + \delta_{\text{Edit}}(1^{m-r}, 1^{m-s})$. By Fact 5.5.(1), $\delta_{\text{Edit}}(1^{m-r}, 1^{m-s}) = \delta_{\text{Edit}}(\epsilon, 1^{|r-s|}) = |r - s|$, where ϵ is the empty string. Hence, $\delta_{\text{Edit}}(0^\ell 1^{m-r}, 1^m) = \min_{0 \leq s \leq \ell} \{c_{\text{subst}} \cdot s + \ell - s + |r - s|\}$. A short case analysis shows that this term is minimized for $s = \min\{\ell, r\}$, where it evaluates to $c_{\text{subst}} \cdot \min\{\ell, r\} + \ell + r - 2 \min\{\ell, r\} = c_{\text{subst}} \cdot \min\{\ell, r\} + |\ell - r|$. The case $x = 1^{m-\ell} 0^r$ is symmetric. \square

For a string y and indices $a \leq b$ we denote the substring from $y[a]$ to $y[b]$ by $y[a..b]$.

Fact 5.7. *Let x and y_1, \dots, y_k be binary strings. Set $y = y_1 \dots y_n$. Then we have*

$$\delta_{\text{Edit}}(x, y) = \min_{x(y_1), \dots, x(y_k)} \sum_{j=1}^k \delta_{\text{Edit}}(x(y_j), y_j),$$

where $x(y_1), \dots, x(y_k)$ ranges over all ordered partitions of x into k substrings, i.e., $x(y_1) = x[i_0 + 1..i_1], x(y_2) = x[i_1 + 1..i_2], \dots, x(y_k) = x[i_{k-1} + 1..i_k]$ for any $0 = i_0 \leq i_1 \leq \dots \leq i_k = |x|$.

Proof. For any ordered partition, the substrings $x(y_j)$ are disjoint and ordered along x , so we can concatenate (optimal) traversals of $(x(y_j), y_j)$, $j \in [k]$, to form a traversal of (x, y) . This shows $\delta_{\text{Edit}}(x, y) \leq \sum_{j=1}^k \delta_{\text{Edit}}(x(y_j), y_j)$.

Now let T be an optimal traversal of (x, y) . Let J_j be the indices in x that appear in a matching or substitution operation with symbols in y_j . Note that these sets are ordered, in the sense that for any $i \in J_j$ and $i' \in J_{j'}$ with $j < j'$ we have $i < i'$. This allows to find an ordered partition $x(y_1), \dots, x(y_k)$ of x such that $x(y_j)$ contains the indices J_j for any j . Let us denote the total cost of the substitutions involving y_j by s_j . Since traversal T deletes $|y_j| - |J_j|$ symbols in y_j and $|x(y_j)| - |J_j|$ symbols in $x(y_j)$, we have $\delta(T) = \sum_{j=1}^k |y_j| + |x(y_j)| - 2|J_j| + s_j$. Clearly, we can construct a traversal of $(x(y_j), y_j)$ that follows the matchings and substitutions in J_j and deletes all other symbols, showing $\delta_{\text{Edit}}(x(y_j), y_j) \leq |y_j| + |x(y_j)| - 2|J_j| + s_j$. By optimality of T , we obtain $\delta_{\text{Edit}}(x, y) \geq \sum_{i=1}^k \delta_{\text{Edit}}(x(y_j), y_j)$. \square

Proof of Lemma 5.4. From now on let x, y be as in Definition 5.3. Observe that indeed x only depends on m, t_Y , and x_1, \dots, x_n , and $\text{type}(x)$ only depends on n, m, t_X , and t_Y , and similarly for y . Moreover, x and y can clearly be constructed in time $\mathcal{O}((n+m)(\ell_X + \ell_Y))$, where $\ell_X = |x_1| = \dots = |x_n|$ and $\ell_Y = |y_1| = \dots = |y_m|$.

It remains to prove that for some C , we have

$$\min_{A \in \mathcal{A}_{n,m}} \delta(A) \leq \delta(x, y) - C \leq \min_{A \in \mathcal{S}_{n,m}} \delta(A). \quad (5)$$

We will set

$$C := 2n\gamma_3 - \beta(n-m)(\gamma_4 + \gamma_2),$$

where

$$\beta := 1 - c_{\text{subst}}/5 \quad \text{and} \quad \gamma_4 := 4\rho\gamma_1 + \ell_X.$$

Note that γ_4 is the length of $G(x_i)$.

Let us give names to the substrings consisting only of zeroes in x and y . In x , we denote the 0^{γ_2} -block after $G(x_i)$ by Z_i^X , $i \in [n-1]$. In y , we denote the 0^{γ_2} -block after $G(y_j)$ by Z_j^Y , $j \in [m-1]$. Moreover, we denote the prefix $0^{n\gamma_3}$ by L^Y and the suffix $0^{n\gamma_3}$ by R^Y .

We first prove the crucial property that for any prefix x' of x the distance $\delta_{\text{Edit}}(x', L^Y)$ is essentially $|L^Y| - \beta|x'| = n\gamma_3 - \beta|x'|$. This is due to a careful choice of the parameters γ_1, γ_2, ρ .

Claim 5.8. For any prefix x' of x we have $\delta_{\text{Edit}}(x', L^Y) \geq n\gamma_3 - \beta|x'|$, with equality if x' is of the form $G(x_1)0^{\gamma_2} \dots G(x_i)0^{\gamma_2}$ for any $0 \leq i < n$. Symmetric statements hold for $\delta_{\text{Edit}}(x'', R^Y)$ where x'' is any suffix of x .

Proof. The parameter γ_3 is chosen such that $|x'| \leq |x| \leq |L^Y|$: Indeed, $|x| \leq n(4\rho\gamma_1 + \ell_X + \gamma_2) \leq n \cdot 2\gamma_2 \leq n\gamma_3 = |L^Y|$. Observe that all zeroes of x' can be matched to zeroes of L^Y , while all ones of x' have to be substituted. The remaining zeroes of L^Y have to be deleted. Denoting the number of ones in x' by ℓ , we obtain $\delta_{\text{Edit}}(x', L^Y) = \ell \cdot c_{\text{subst}} + (|L^Y| - |x'|)$. We will show $\ell \geq |x'|/5$, with equality if x' has the special form as in the statement. In other words, the *relative number of ones* $\ell/|x'|$ is at least $1/5$, with equality if x' has the special form. This implies $\delta_{\text{Edit}}(x', L^Y) \geq n\gamma_3 - \beta|x'|$, with equality if x' has the special form.

Note that each x_i has length ℓ_X and contains s_X ones, so that $G(x_i)Z_i^X = (1^{\gamma_1}0^{\gamma_1})^\rho x_i (0^{\gamma_1}1^{\gamma_1})^\rho 0^{\gamma_2}$ contains $2\rho\gamma_1 + (\ell_X - s_X) + \gamma_2$ zeroes and $2\rho\gamma_1 + s_X$ ones. The parameter γ_2 is chosen so that the number of zeroes is four times the number of ones, implying that the relative number of ones is

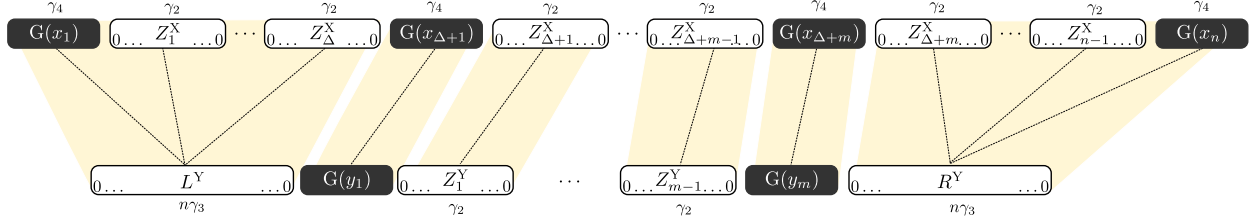


Figure 4: Optimal traversal corresponding to structured alignment $A = \{(\Delta+j, j) \mid j \in [m]\} \in \mathcal{S}_{n,m}$.

1/5. Note that any prefix of $(1^{\gamma_1}0^{\gamma_1})^\rho$ has relative number of ones at least 1/2. Since $x_i0^{\gamma_1}$ has less than $2\gamma_1$ zeroes and $|(1^{\gamma_1}0^{\gamma_1})^\rho| \geq 2\gamma_1$, any prefix of $(1^{\gamma_1}0^{\gamma_1})^\rho x_i0^{\gamma_1}$ has relative number of ones at least 1/4. Since any prefix of $1^{\gamma_1}(0^{\gamma_1}1^{\gamma_1})^{\rho-1}$ has relative number of ones at least 1/2, any prefix of $(1^{\gamma_1}0^{\gamma_1})^\rho x_i(0^{\gamma_1}1^{\gamma_1})^\rho$ has relative number of ones at least 1/4. The relative number of ones decreases by adding any prefix of 0^{γ_2} , however, for the final string $(1^{\gamma_1}0^{\gamma_1})^\rho x_i(0^{\gamma_1}1^{\gamma_1})^\rho 0^{\gamma_2}$, we already argued that the relative number of ones is 1/5. This shows that the relative number of ones of any prefix of x is at least 1/5. \square

We now show the upper bound of (5), i.e., $\delta_{\text{Edit}}(x, y) \leq C + \min_{A \in \mathcal{S}_{n,m}} \sum_{(i,j) \in A} \delta_{\text{Edit}}(x_i, y_j)$. Consider a structured alignment $A = \{(\Delta+1, 1), \dots, (\Delta+m, m)\} \in \mathcal{S}_{n,m}$. We construct an ordered partition of x as in Fact 5.7 by setting (see Figure 4)

$$\begin{aligned} x(G(y_j)) &:= G(x_{\Delta+j}) \quad \text{for } j \in [m], \\ x(Z_j^Y) &:= Z_{\Delta+j}^X \quad \text{for } j \in [m-1], \\ x(L^Y) &:= G(x_1)Z_1^X \dots G(x_\Delta)Z_\Delta^X, \\ x(R^Y) &:= Z_{\Delta+m}^X G(x_{\Delta+m+1}) \dots Z_{n-1}^X G(x_n). \end{aligned}$$

Note that indeed these strings partition x and y , respectively. Thus, Fact 5.7 yields

$$\delta_{\text{Edit}}(x, y) \leq \delta_{\text{Edit}}(x(L^Y), L^Y) + \delta_{\text{Edit}}(x(R^Y), R^Y) + \sum_{j=1}^m \delta_{\text{Edit}}(G(x_{\Delta+j}), G(y_j)) + \sum_{j=1}^{m-1} \delta_{\text{Edit}}(Z_{\Delta+j}^X, Z_j^Y).$$

Since $x(L^Y)$ is a prefix of x of the correct form, by Claim 5.8 we have $\delta_{\text{Edit}}(x(L^Y), L^Y) = n\gamma_3 - \beta|x(L^Y)|$. Symmetrically, we obtain $\delta_{\text{Edit}}(x(R^Y), R^Y) = n\gamma_3 - \beta|x(R^Y)|$. Note that $|G(x_i)Z_i^X| = \gamma_4 + \gamma_2$, so that $|x(L^Y)| + |x(R^Y)| = (n-m)(\gamma_4 + \gamma_2)$. Moreover, as $Z_i^X = Z_j^Y = 0^{\gamma_2}$ we have $\delta_{\text{Edit}}(Z_{\Delta+j}^X, Z_j^Y) = 0$. Finally, by matching all guarding zeroes and ones of $G(x_{\Delta+j}) = (1^{\gamma_1}0^{\gamma_1})^\rho x_{\Delta+j}(0^{\gamma_1}1^{\gamma_1})^\rho$ and $G(y_j) = (1^{\gamma_1}0^{\gamma_1})^\rho y_j(0^{\gamma_1}1^{\gamma_1})^\rho$ we conclude $\delta_{\text{Edit}}(G(x_{\Delta+j}), G(y_j)) \leq \delta_{\text{Edit}}(x_{\Delta+j}, y_j)$. This yields

$$\delta_{\text{Edit}}(x, y) \leq 2n\gamma_3 - \beta(n-m)(\gamma_4 + \gamma_2) + \sum_{j=1}^m \delta_{\text{Edit}}(x_{\Delta+j}, y_j) = C + \sum_{(i,j) \in A} \delta_{\text{Edit}}(x_i, y_j).$$

As $A \in \mathcal{S}_{n,m}$ was arbitrary, the desired inequality follows.

It remains to prove the lower bound of (5), i.e., $\delta_{\text{Edit}}(x, y) \geq C + \min_{A \in \mathcal{A}_{n,m}} \delta(A)$. As in Fact 5.7, let $x(L^Y)$, $x(G(y_j))$ for $j \in [m]$, $x(Z_j^Y)$ for $j \in [m-1]$, $x(R^Y)$ be an ordered partition of x such that

$$\delta_{\text{Edit}}(x, y) = \delta_{\text{Edit}}(x(L^Y), L^Y) + \delta_{\text{Edit}}(x(R^Y), R^Y) + \sum_{j=1}^m \delta_{\text{Edit}}(x(G(y_j)), G(y_j)) + \sum_{j=1}^{m-1} \delta_{\text{Edit}}(x(Z_j^Y), Z_j^Y).$$

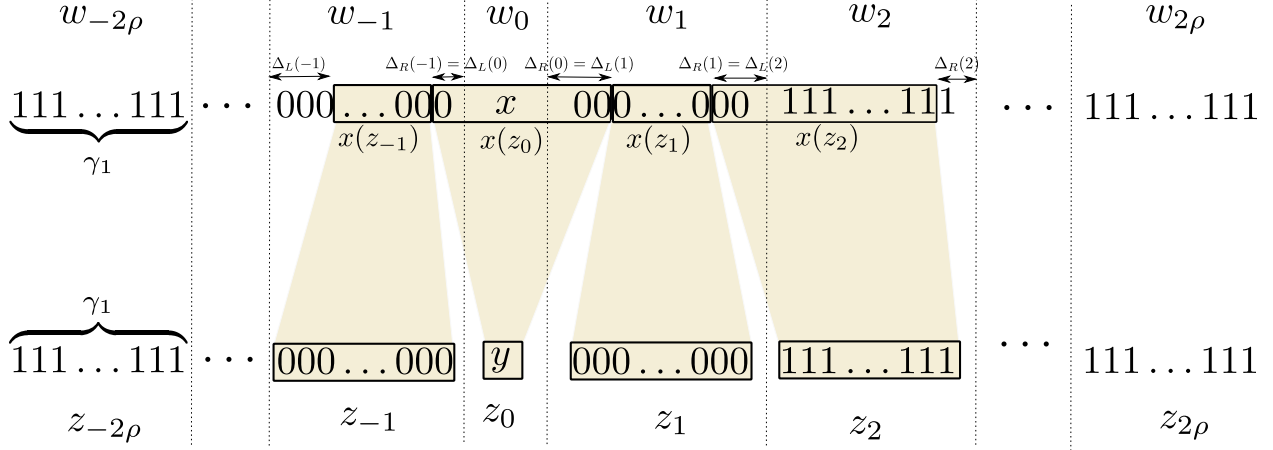


Figure 5: Illustration for the proof of Claim 5.9.

We define an alignment A as follows. If there is some i such that x_i is contained in $x(G(y_j))$, then align j with any such i . Otherwise leave j unaligned.

Claim 5.9. We have

$$\delta_{\text{Edit}}(x(G(y_j)), G(y_j)) \geq \beta(\gamma_4 - |x(G(y_j))|) + \begin{cases} \delta_{\text{Edit}}(x_i, y_j) & \text{if } j \text{ is aligned to } i, \\ \max_{i,j'} \delta_{\text{Edit}}(x_i, y_{j'}) & \text{if } j \text{ is unaligned.} \end{cases}$$

Proof. If $|x(G(y_j))| \geq \gamma_2$, then $|x(G(y_j))| \geq \gamma_2 \geq \gamma_4 + 2(\ell_x + \ell_y) \geq \gamma_4 + 2 \max_{i,j'} \delta_{\text{Edit}}(x_i, y_{j'})$ and by $\beta > 1/2$ the right hand side of the claim is at most 0, so the claim holds trivially. Otherwise $x(G(y_j))$ is shorter than any $Z_i^x = 0^{\gamma_2}$, implying that $x(G(y_j))$ is a substring of $0^{\gamma_2}G(x_i)0^{\gamma_2}$ for some $i \in [n]$.

We write $G(y_j)$ as $z_{-2\rho}z_{-2\rho+1} \dots z_{2\rho-1}z_{2\rho}$, where $z_{-2k} = z_{2k} = 1^{\gamma_1}$, $z_{-2k+1} = z_{2k-1} = 0^{\gamma_1}$, and $z_0 = y_j$ (for all $1 \leq k \leq \rho$). As in Fact 5.7, we split up $x(G(y_j))$ into $x(z_k)$, $-2\rho \leq k \leq 2\rho$, such that $\delta_{\text{Edit}}(x(G(y_j)), G(y_j)) = \sum_{k=-2\rho}^{2\rho} \delta_{\text{Edit}}(x(z_k), z_k)$. Similarly, we write $G(x_i)$ as $w_{-2\rho}w_{-2\rho+1} \dots w_{2\rho-1}w_{2\rho}$. We denote the distance of the start of $x(z_k)$ to the start of w_k by $\Delta_L(k)$, i.e., if $x(z_k) = x[a..b]$ and $w_k = x[c..d]$ we set $\Delta_L(k) := |a - c|$. Similarly, we set $\Delta_R(k) := |b - d|$. For an illustration, see Figure 5. Note that $\Delta_R(k) = \Delta_L(k + 1)$ holds for any k .

First assume (*): for some $k \neq 0$ the string $x(z_k)$ is longer than $\frac{5}{4}\gamma_1$ or $x(z_k)$ has less than $\frac{3}{4}\gamma_1$ common symbols with z_k . Then clearly $\delta_{\text{Edit}}(x(G(y_j)), G(y_j)) \geq \delta_{\text{Edit}}(x(z_k), z_k) \geq \frac{1}{4}\gamma_1$. By Fact 5.5.(2), we also have $\delta_{\text{Edit}}(x(G(y_j)), G(y_j)) \geq |G(y_i)| - |x(G(y_j))| = \gamma_4 - |x(G(y_j))|$. As a linear combination of these two lower bounds, we obtain $\delta_{\text{Edit}}(x(G(y_j)), G(y_j)) \geq \beta(\gamma_4 - |x(G(y_j))|) + (1 - \beta)\frac{1}{4}\gamma_1$. Since $(1 - \beta)\frac{1}{4}\gamma_1 = \frac{c_{\text{subst}}}{20}\gamma_1 \geq \ell_x + \ell_y \geq \max_{i,j'} \delta_{\text{Edit}}(x_i, y_{j'})$, we have proven the statement in this case.

If (*) does not hold, then we have $\Delta_L(k), \Delta_R(k) \leq \frac{1}{2}\gamma_1$ for any $|k| > 1$: It suffices to show the claim for any even $k \neq 0$, since $\Delta_R(k) = \Delta_L(k + 1)$. For any even $k \neq 0$, the string $x(z_k)$ has to contain the majority of a block w_ℓ with even $\ell \neq 0$. Since the numbers of blocks are identical in $G(y_j)$ and $G(x_i)$, $x(z_k)$ has to contain the majority of w_k for any even $k \neq 0$. Specifically, $x(z_k)$ contains at least $\frac{3}{4}\gamma_1$ symbols of w_k and has length at most $\frac{5}{4}\gamma_1$, implying the desired inequalities for $\Delta_L(k), \Delta_R(k)$. Note that in this case i and j are aligned.

Note that for even $k \neq 0$ we obtain $x(z_k)$ from $w_k = z_k = 1^{\gamma_1}$ by either deleting a prefix of $\Delta_L(k)$ ones or prepending $\Delta_L(k)$ zeroes, and by either deleting a suffix of $\Delta_R(k)$ ones or by appending $\Delta_R(k)$ zeroes. Hence, Fact 5.6 shows that

$$\delta_{\text{Edit}}(x(z_k), z_k) \geq |\Delta_L(k) - \Delta_R(k)| + c_{\text{subst}} \cdot \min\{\Delta_L(k), \Delta_R(k)\}. \quad (6)$$

The same argument works for any k with $|k| > 1$. For $k \in \{-1, 1\}$ the argument does not work, since $z_{-1} = z_1 = 0^{\gamma_1}$ is not surrounded by blocks of 1^{γ_1} . However, for $k \in \{-1, 1\}$ we have the weaker $\delta_{\text{Edit}}(x(z_k), z_k) \geq |\Delta_L(k) - \Delta_R(k)|$ by Fact 5.5.(2). Moreover, by Fact 5.5.(3) we have

$$\delta_{\text{Edit}}(x(z_0), z_0) \geq \delta_{\text{Edit}}(x_i, y_j) - \Delta_L(0) - \Delta_R(0). \quad (7)$$

Combining these inequalities yields $\delta_{\text{Edit}}(x(\text{G}(y_j)), \text{G}(y_j)) \geq \delta_{\text{Edit}}(x_i, y_j) + \Delta_L(-2\rho) + \Delta_R(2\rho)$ as we show in the following. This implies the desired statement, since $\Delta_L(-2\rho) + \Delta_R(2\rho) \geq ||\text{G}(y_j)| - |x(\text{G}(y_j))|| \geq \gamma_4 - |x(\text{G}(y_j))| \geq \beta(\gamma_4 - |x(\text{G}(y_j))|)$. To show the claim, we set $s_L := \min\{\Delta_L(k) \mid -2\rho \leq k \leq 0\}$ and $s_R := \min\{\Delta_R(k) \mid 0 \leq k \leq 2\rho\}$. Note that $\Delta_L(k)$ has a total variation of at least $\Delta_L(-2\rho) - s_L + \Delta_L(0) - s_L$ over $k = -2\rho, \dots, 0$, since it starts in $\Delta_L(-2\rho)$, changes to s_L , and then changes to $\Delta_L(0)$. Thus, summing $|\Delta_L(k) - \Delta_R(k)| = |\Delta_L(k) - \Delta_L(k+1)|$ over all $-2\rho \leq k \leq -1$ yields at least $\Delta_L(-2\rho) - s_L + \Delta_L(0) - s_L$. Moreover, for every $-2\rho \leq k < -1$ inequality (6) applies and the summand $c_{\text{subst}} \cdot \min\{\Delta_L(k), \Delta_R(k)\}$ is at least $c_{\text{subst}} \cdot s_L$. As the number of such k 's is $2\rho - 1 \geq 2/c_{\text{subst}}$, the total contribution of the summand $c_{\text{subst}} \cdot \min\{\Delta_L(k), \Delta_R(k)\}$ over all $k < 0$ is at least $2s_L$. Thus, we have

$$\begin{aligned} \sum_{k=-2\rho}^{-1} \delta_{\text{Edit}}(x(z_k), z_k) &\geq \sum_{k=-2\rho}^{-1} |\Delta_L(k) - \Delta_R(k)| + \sum_{k=-2\rho}^{-2} c_{\text{subst}} \cdot \min\{\Delta_L(k), \Delta_R(k)\} \\ &\geq (\Delta_L(-2\rho) - s_L + \Delta_L(0) - s_L) + (2s_L) \geq \Delta_L(-2\rho) + \Delta_L(0). \end{aligned}$$

Using a symmetric statement for the sum over all $k > 0$ as well as equation (7), we obtain the desired inequality $\delta_{\text{Edit}}(x(\text{G}(y_j)), \text{G}(y_j)) = \sum_{k=-2\rho}^{2\rho} \delta_{\text{Edit}}(x(z_k), z_k) \geq \delta_{\text{Edit}}(x_i, y_j) + \Delta_L(-2\rho) + \Delta_R(2\rho)$. \square

Since $L^Y = 0^{n\gamma_3}$ and $x(L^Y)$ is a prefix of x , by Claim 5.8 we have $\delta_{\text{Edit}}(x(L^Y), L^Y) \geq n\gamma_3 - \beta|x(L^Y)|$, and symmetrically we get $\delta_{\text{Edit}}(x(R^Y), R^Y) \geq n\gamma_3 - \beta|x(L^Y)|$. By Fact 5.5.(2), we have $\delta_{\text{Edit}}(x(Z_j^Y), Z_j^Y) \geq ||Z_j^Y| - |x(Z_j^Y)|| \geq \beta(\gamma_2 - |x(Z_j^Y)|)$. Putting all of this together, we obtain

$$\delta_{\text{Edit}}(x, y) \geq 2n\gamma_3 + \beta \left[\sum_{j=1}^m (\gamma_4 - |x(\text{G}(y_j))|) + \sum_{j=1}^{m-1} (\gamma_2 - |x(Z_j^Y)|) - |x(L^Y)| - |x(R^Y)| \right] + \delta(A),$$

where we used $\delta(A) = \sum_{(i,j) \in A} \delta_{\text{Edit}}(x_i, y_j) + (m - |A|) \max_{i,j} \delta_{\text{Edit}}(x_i, y_j)$. Note that by definition of x and since the strings $x(\text{G}(y_j)), x(Z_j^Y), x(L^Y), x(R^Y)$ partition x we have

$$n\gamma_4 + (n-1)\gamma_2 = |x| = \sum_{j=1}^m |x(\text{G}(y_j))| + \sum_{j=1}^{m-1} |x(Z_j^Y)| + |x(L^Y)| + |x(R^Y)|.$$

Together, this yields the desired bound from below

$$\delta_{\text{Edit}}(x, y) \geq 2n\gamma_3 - \beta(n-m)(\gamma_4 + \gamma_2) + \delta(A). \quad \square$$

5.3 Algorithm

For completeness, we prove a generalization of the algorithm of Hirschberg [12] from LCS to edit distance. Recall that the trivial dynamic programming algorithm computes a table storing all distances $\delta_{\text{Edit}}(x[1..i], y[1..j])$. In contrast, we build a dynamic programming table storing for any index j and any cost k the minimal index i with $\delta_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i - j) = k$. For some intuition, note that for $i \geq j$ at least $i - j$ symbols in $x[1..i]$ have to be deleted so that the cost $\delta_{\text{Edit}}(x[1..i], y[1..j])$ is at least $c_{\text{del-x}}(i - j)$. Thus, it makes sense to “normalize” the cost by subtracting $c_{\text{del-x}}(i - j)$. As we will see, the normalized cost is bounded by $\mathcal{O}(m)$ (the length of the smaller of the two strings), which reduces the table size to $\mathcal{O}(m^2)$.

Theorem 5.10. *Let $c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}}$ be positive integers. $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ can be solved in time $\mathcal{O}((n + m^2) \log |\Sigma|)$ on strings of length n, m with $n \geq m$ over alphabet Σ .*

Note that it is easy to ensure $\Sigma \subseteq [n + m]$ after $\mathcal{O}(n \log(\min\{|\Sigma|, n\}))$ preprocessing.³ Thus, the running time is at most $\mathcal{O}((n + m^2) \log n) = \tilde{\mathcal{O}}(n + m^2)$, and Theorem 1.4 follows from the above theorem and the second part of Lemma 5.1. Our algorithm is designed for the pointer machine model; on the Word RAM the log-factor can be improved.

Consider strings x, y over alphabet Σ of length n, m , respectively, $n \geq m$. For convenience, we set $\min \emptyset := \infty$. For any index $i \in \{0, \dots, n\}$ and symbol $\sigma \in \Sigma$ we set

$$\begin{aligned} \text{Next}_{=\sigma}^x(i) &:= \min\{i' \mid i < i' \leq n \text{ and } x[i'] = \sigma\}, \\ \text{Next}_{\neq\sigma}^x(i) &:= \min\{i' \mid i < i' \leq n \text{ and } x[i'] \neq \sigma\}. \end{aligned}$$

We argue that a data structure can be built in $\mathcal{O}(n \log |\Sigma|)$ preprocessing time supporting $\text{Next}_{=\sigma}^x(i)$ and $\text{Next}_{\neq\sigma}^x(i)$ queries in time $\mathcal{O}(\log |\Sigma|)$. A simple solution with worse running time is to precompute all answers to all possible queries $\text{Next}_{=\sigma}^x(i)$ and $\text{Next}_{\neq\sigma}^x(i)$, with $i \in \{0, \dots, n\}$, $\sigma \in \Sigma$, in time $\mathcal{O}(|\Sigma|n)$ by one scan from $x[n]$ to $x[1]$. To improve the preprocessing time for $\text{Next}_{\neq\sigma}^x(i)$, note that $\text{Next}_{\neq\sigma}^x(i) = i + 1$ for all $\sigma \neq x[i + 1]$. Thus, we only have to precompute $\text{Next}_{\neq x[i+1]}^x(i)$ (which can be done in time $\mathcal{O}(n)$ by one scan from $x[n]$ to $x[1]$), then $\text{Next}_{\neq\sigma}^x(i)$ can be queried in time $\mathcal{O}(1)$. For $\text{Next}_{=\sigma}^x(i)$, for any $i \in \{0, \dots, n\}$ we build a dictionary D_i storing $\text{Next}_{=\sigma}^x(i)$ for each $\sigma \in \Sigma$. Note that D_{i-1} and D_i differ only for the symbol $x[i + 1]$. Thus, we can use persistent search trees [10] as dictionary data structures, resulting in a preprocessing time of $\mathcal{O}(n \log |\Sigma|)$ for building D_0, \dots, D_n and a lookup time of $\mathcal{O}(\log |\Sigma|)$ for querying $\text{Next}_{=\sigma}^x(i)$. Using such a Next data structure, we can formulate our dynamic programming algorithm, see Algorithm 1.

Since $\text{Next}_{=\sigma}^x$ and $\text{Next}_{\neq\sigma}^x$ can be queried in time $\mathcal{O}(\log |\Sigma|)$ and $M = (c_{\text{del-x}} + c_{\text{del-y}})m = \mathcal{O}(m)$, Algorithm 1 runs in time $\mathcal{O}(m^2 \log |\Sigma|)$. Together with the preprocessing time for the Next data structure, we obtain a total running time of $\mathcal{O}((n + m^2) \log |\Sigma|)$. It remains to argue correctness.

Correctness We prove that the dynamic programming table $I[j, k]$ has the following meaning.

Lemma 5.11. *Algorithm 1 computes for any $j \in [m]$, $k \in \mathbb{Z}$*

$$I[j, k] = \min\{0 \leq i \leq n \mid \delta_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i - j) = k\}.$$

³To compress the alphabet we build a balanced binary search tree T whose nodes correspond to Σ (by simply adding all symbols of x and y to T). Then we replace each symbol by its index in some fixed ordering of the nodes of T .

Algorithm 1 Algorithm for solving $\text{Edit}(c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}})$ in time $\mathcal{O}((n + m^2) \log |\Sigma|)$.

Assumption: $c_{\text{del-x}}, c_{\text{del-y}}, c_{\text{match}}, c_{\text{subst}}$ are positive integers

Input: strings x, y of length n, m , $n \geq m$

Output: $\delta_{\text{Edit}}(x, y)$

```

 $M \leftarrow (c_{\text{del-x}} + c_{\text{del-y}})m$ 
Implicitly set  $I[j, k] \leftarrow \infty$  for all  $j$  and all  $k < 0$  or  $k > M$ 
 $I[0, 0] \leftarrow 0$ 
 $I[0, k] \leftarrow \infty$  for  $0 < k \leq M$ 
for  $j = 1, \dots, m$  do
  for  $k = 0, \dots, M$  do
     $I[j, k] \leftarrow \min\{I[j - 1, k - c_{\text{del-x}} - c_{\text{del-y}}],$ 
       $\text{Next}_{=y[j]}^x(I[j - 1, k - c_{\text{match}}]),$ 
       $\text{Next}_{\neq y[j]}^x(I[j - 1, k - c_{\text{subst}}])\}$ 
  end for
end for
return  $c_{\text{del-x}}(n - m) + \min\{0 \leq k \leq M \mid I[m, k] < \infty\}$ .
```

Proof. Let $R[j, k] := \min\{0 \leq i \leq n \mid \delta_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i - j) = k\}$ be the right hand side of the statement.

The statement is true for $j = 0$, since for the empty string ϵ we have $\delta_{\text{Edit}}(x[1..i], \epsilon) = c_{\text{del-x}} \cdot i$, so that $R[0, k] = 0$ for $k = 0$, and ∞ otherwise, which is exactly how we initialize $I[0, k]$.

We show that $R[j, k] = \infty$ for $k < 0$ or $k > M$, which is also implicitly assumed for $I[j, k]$ in Algorithm 1. Note that for $i \geq j$ we have to delete at least $i - j$ symbols in $x[1..i]$ when traversing it with $y[1..j]$, which implies $\delta_{\text{Edit}}(x[1..i], y[1..j]) \geq c_{\text{del-x}}(i - j)$. Since additionally for $i < j$ the term $-c_{\text{del-x}}(i - j)$ is positive, we have $\delta_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i - j) \geq 0$ for all i, j . Thus, for no $k < 0$ we can have $\delta_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i - j) = k$, implying $R[j, k] = \infty$ in this case. Moreover, $\delta_{\text{Edit}}(x[1..i], y[1..j]) \leq c_{\text{del-x}} \cdot i + c_{\text{del-y}} \cdot j$, which implies $\delta_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i - j) \leq (c_{\text{del-y}} + c_{\text{del-x}})j \leq M$. Thus, we also have $R[j, k] = \infty$ for $k > M$.

It remains to show the statement for $j > 1$ and $0 \leq k \leq M$. Inductively, we can assume that the statement holds for $j - 1$. We show that $R[j, k]$ satisfies the same recursive equation as $I[j, k]$ in Algorithm 1. Let $i := R[j, k]$ and consider an optimal traversal T of $(x[1..i], y[1..j])$. We obtain a traversal T' by removing the last operation in T .

If the last operation in T is a deletion in x , then T' is an optimal traversal of $(x[1..i - 1], y[1..j])$ with cost $\delta_{\text{Edit}}(x[1..i - 1], y[1..j]) = \delta_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}$. Thus, we can decrease i to $i - 1$ while keeping $k = \delta_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i - j) = \delta_{\text{Edit}}(x[1..i - 1], y[1..j]) - c_{\text{del-x}}(i - 1 - j)$. This contradicts minimality of $i = R[j, k]$, so the last operation in T cannot be a deletion in x .

If the last operation in T is a deletion in y , then T' is an optimal traversal of $(x[1..i], y[1..j - 1])$ with cost $\delta_{\text{Edit}}(x[1..i], y[1..j - 1]) = \delta_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-y}}$. Thus, we have

$$\begin{aligned}
R[j, k] &= \min\{0 \leq i \leq n \mid \delta_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{del-x}}(i - j) = k\} \\
&= \min\{0 \leq i \leq n \mid \delta_{\text{Edit}}(x[1..i], y[1..j - 1]) - c_{\text{del-x}}(i - (j - 1)) = k - c_{\text{del-y}} - c_{\text{del-x}}\} \\
&= R[j - 1, k - c_{\text{del-x}} - c_{\text{del-y}}].
\end{aligned}$$

If the last operation in T is a matching of $x[i]$ and $y[j]$, then T' is an optimal traversal of

$(x[1..i-1], y[1..j-1])$ with cost $\delta_{\text{Edit}}(x[1..i-1], y[1..j-1]) = \delta_{\text{Edit}}(x[1..i], y[1..j]) - c_{\text{match}}$. This implies $\delta_{\text{Edit}}(x[1..i-1], y[1..j-1]) - c_{\text{del-x}}((i-1) - (j-1)) = k - c_{\text{match}}$, so that $i-1$ is a candidate for $R[j-1, k - c_{\text{match}}]$. Let $i' := R[j-1, k - c_{\text{match}}]$ and note that $i' \leq i-1$. As $x[i] = y[j]$, we obtain $i \geq \text{Next}_{=y[j]}^x(i') =: i^*$. In the following we show $i = i^*$. By definition of i' we have $\delta_{\text{Edit}}(x[1..i'], y[1..j-1]) - c_{\text{del-x}}(i' - j + 1) = k - c_{\text{match}}$. Hence, $\delta_{\text{Edit}}(x[1..i^*], y[1..j]) \leq c_{\text{match}} + c_{\text{del-x}}(i^* - i' - 1) + \delta_{\text{Edit}}(x[1..i'], y[1..j-1]) = k + c_{\text{del-x}}(i^* - j)$. We even have equality, since otherwise $\delta_{\text{Edit}}(x[1..i], y[1..j]) \leq \delta_{\text{Edit}}(x[1..i^*], y[1..j]) + c_{\text{del-x}}(i - i^*) < k + c_{\text{del-x}}(i - j)$, contradicting the definition of i . Thus, i^* is a candidate for $R[j, k]$, implying that we also have $i \leq i^*$. Hence, we have $R[j, k] = i^* = \text{Next}_{=y[j]}^x(i') = \text{Next}_{=y[j]}^x(R[j-1, k - c_{\text{match}}])$.

We argue analogously if the last operation in T is a substitution of $x[i]$ and $y[j]$. This yields

$$R[j, k] = \min\{R[j-1, k - c_{\text{del-x}} - c_{\text{del-y}}], \text{Next}_{=y[j]}^x(R[j-1, k - c_{\text{match}}]), \text{Next}_{\neq y[j]}^x(R[j-1, k - c_{\text{subst}}])\}.$$

Hence, $R[j, k]$ satisfies the same recursion as $I[j, k]$, and we proved $R[j, k] = I[j, k]$ for all j, k . \square

Lemma 5.12. *Algorithm 1 correctly computes $\delta_{\text{Edit}}(x, y)$.*

Proof. Among all optimal traversals of (x, y) , pick a traversal T that ends with the maximal number d of deletions in x , and set $i := n - d$. Observe that i is minimal with $\delta_{\text{Edit}}(x[1..i], y[1..m]) + c_{\text{del-x}}(n - i) = \delta_{\text{Edit}}(x, y)$, which is equivalent to $\delta_{\text{Edit}}(x[1..i], y[1..m]) - c_{\text{del-x}}(i - m) = \delta_{\text{Edit}}(x, y) - c_{\text{del-x}}(n - m) =: k$. Thus, $i = I[m, k] < \infty$, which implies that the return value of Algorithm 1 is at most $c_{\text{del-x}}(n - m) + k = \delta_{\text{Edit}}(x, y)$.

Moreover, for any k with $I[m, k] < \infty$ there is a $0 \leq i \leq n$ with $\delta_{\text{Edit}}(x[1..i], y[1..m]) - c_{\text{del-x}}(i - m) = k$. By appending $n - i$ deletions in x to any optimal traversal of $(x[1..i], y[1..m])$, we obtain $\delta_{\text{Edit}}(x, y) \leq \delta_{\text{Edit}}(x[1..i], y[1..m]) + c_{\text{del-x}}(n - i) = k + c_{\text{del-x}}(n - m)$. Hence, the return value of Algorithm 1 is also at least $\delta_{\text{Edit}}(x, y)$. \square

6 Dynamic Time Warping

We present coordinate values and an *unbalanced* alignment gadget for DTW on one-dimensional curves taking values in \mathbb{N}_0 , i.e., we consider the set of inputs $\mathcal{I} := \bigcup_{k \geq 0} \mathbb{N}_0^k$.

Lemma 6.1. *DTW admits coordinate values by setting*

$$\mathbf{1}_x := 1100, \mathbf{0}_x := 0110, \mathbf{1}_y := 0011, \mathbf{0}_y := 1010.$$

Proof. All four values have the same length and sum of all entries, so they have equal type. Short calculations show that $4 = \delta_{\text{DTW}}(\mathbf{1}_x, \mathbf{1}_y) > \delta_{\text{DTW}}(\mathbf{0}_x, \mathbf{1}_y) = \delta_{\text{DTW}}(\mathbf{0}_x, \mathbf{0}_y) = \delta_{\text{DTW}}(\mathbf{1}_x, \mathbf{0}_y) = 1$. \square

Definition 6.2. Consider instances $x_1, \dots, x_n \in \mathcal{I}_{t_x}$ and $y_1, \dots, y_m \in \mathcal{I}_{t_y}$ with $n \geq m$ and types $t_x = (\ell_x, s_x), t_y = (\ell_y, s_y)$. We define $M := 2z$, where z is the largest value contained in any of the one-dimensional curves $x_1, \dots, x_n, y_1, \dots, y_m$, and we set $\kappa := 3(\ell_x + \ell_y)$. We construct

$$\begin{aligned} \text{GA}_x^{m, t_y}(x_1, \dots, x_n) &:= M^\kappa x_1 M^\kappa x_2 M^\kappa \dots M^\kappa x_n M^\kappa, \\ \text{GA}_y^{n, t_x}(y_1, \dots, y_m) &:= M^\kappa y_1 M^\kappa y_2 M^\kappa \dots M^\kappa y_m M^\kappa, \end{aligned}$$

where M^κ is to be understood as a sequence with κ times the entry M .

Lemma 6.3. *Definition 6.2 realizes an unbalanced alignment gadget for dynamic time warping.*

Thus, Theorem 3.3 is applicable, implying a lower bound of $\mathcal{O}((nm)^{1-\varepsilon})$ for DTW on one-dimensional curves over \mathbb{N}_0 . To restrict the alphabet further, note that our basic values use the alphabet $\{0, 1\} \subseteq \mathbb{N}_0$ and each invocation of the alignment gadget introduces a new symbol which is twice as large as the largest value seen so far. Since in the proof of Theorem 3.3 we use alignment gadgets three times, we introduce the symbols 2, 4, and 8. In total, we prove quadratic-time hardness of DTW on one-dimensional curves taking values in $\{0, 1, 2, 4, 8\} \subseteq \mathbb{N}_0$. This proves Theorems 1.1 and 1.3.

Proof of Lemma 6.3. Observe that $x := \text{GA}_x^{m, t_y}(x_1, \dots, x_n)$ and $y := \text{GA}_y^{n, t_x}(y_1, \dots, y_m)$ can be computed in time $\mathcal{O}((n+m)(\ell_x + \ell_y))$ yielding strings of length $\mathcal{O}(n(\ell_x + \ell_y))$ and $\mathcal{O}(m(\ell_x + \ell_y))$, respectively. Moreover, $\text{type}(x)$ and $\text{type}(y)$ only depend on t_x, t_y, n, m . It remains to show the inequalities (2) of Definition 3.1, for which we set $C := (n-m)(\ell_x M - s_x)$.

We start with the following useful observations.

Claim 6.4. Let $\ell \geq 1$ and $a, a', b, b' \in \mathbb{N}_0$. For any $i \in [n], j \in [m]$, we have

- (1) $\delta_{\text{DTW}}(x_i, M^\ell) \geq \delta_{\text{DTW}}(x_i, M) = \ell_x M - s_x \geq \ell_x M/2$ and $\delta_{\text{DTW}}(M^\ell, y_j) \geq \delta_{\text{DTW}}(M, y_j) = \ell_y M - s_y \geq \ell_y M/2$,
- (2) $\delta_{\text{DTW}}(x_i, y_j) < (\ell_x + \ell_y)M/2$.
- (3) $\delta_{\text{DTW}}(x', M^\kappa) \geq \kappa M/2$ and $\delta_{\text{DTW}}(M^\kappa, y') \geq \kappa M/2$ for any substrings x' of x_i and y' of y_j ,
- (4) $\delta_{\text{DTW}}(M^a x_i M^{a'}, M^b y_j M^{b'}) \geq \delta_{\text{DTW}}(x_i, y_j)$.

Proof. For (1), observe that each symbol of x_i can only be traversed together with the symbol M and hence,

$$\delta_{\text{DTW}}(x_i, M^\ell) \geq \delta_{\text{DTW}}(x_i, M) = \sum_{k=1}^{\ell_x} |M - x_i[k]| = \ell_x M - \sum_{k=1}^{\ell_x} x_i[k] = \ell_x M - s_x.$$

Since $x_i[k] \leq z = M/2$, we have $s_x \leq \ell_x M/2$. The statement for y_j is symmetric.

For (2) and (3), note that all symbols in x' are in $[0, z]$. Hence, we obtain $\delta_{\text{DTW}}(x_i, y_j) \leq \max\{|x_i|, |y_j|\} \cdot z < (\ell_x + \ell_y)M/2$. Likewise, $\delta_{\text{DTW}}(x', M^\kappa) \geq \kappa(M - z) = \kappa M/2$. The inequality for y_j follows symmetrically.

To prove (4), consider an optimal traversal T of $M^a x_i M^{a'}$ and $M^b y_j M^{b'}$. We construct a traversal T' of x_i and y_j that has no larger cost. If T does not already traverse $x_i[1]$ together with $y_j[1]$, then at some step in T either a symbol in x_i is traversed together with a symbol of the prefix M^b or a symbol in y_j is traversed together with a symbol of the prefix M^a . Let us assume the first case, since the second is symmetric. A contiguous part T^H of T consists of traversing a prefix x' of x_i together with all symbols in M^b , incurring a cost of at least $|x'|M/2$. Let T^R be the remaining part of T after T^H . We construct a traversal T'' of $x_i M^{a'}$ and $y_j M^{b'}$ as follows. We first traverse x' together with $y_j[1]$ and then follow T^R , which is possible since T^R starts at $y_j[1]$. Since traversing x' together with $y_j[1]$ incurs a cost of at most $|x'|z = |x'|M/2$, which is smaller than the cost of T^H , the cost of our constructed traversal T'' is no larger than the cost of T . Symmetrically, we eliminate the suffixes $M^{a'}$ and $M^{b'}$ and construct a traversal T' of x_i and y_j of cost no larger than T . \square

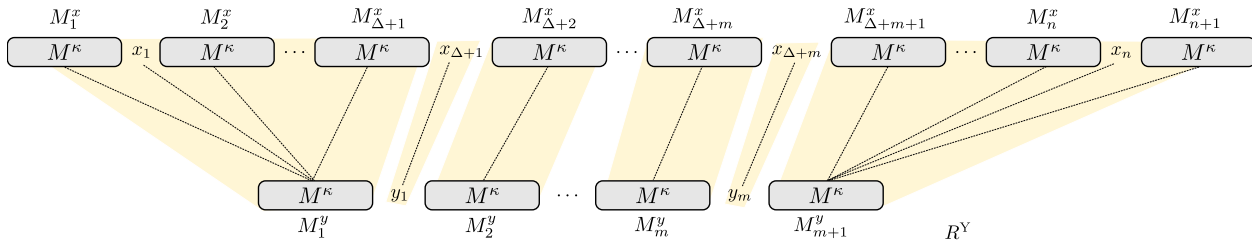


Figure 6: Optimal traversal corresponding to structured alignment $A = \{(\Delta+j, j) \mid j \in [m]\} \in \mathcal{S}_{n,m}$.

We first verify that

$$\delta_{\text{DTW}}(x, y) \leq (n - m)(\ell_x M - s_x) + \min_{A \in \mathcal{S}_{n,m}} \sum_{(i,j) \in A} \delta_{\text{DTW}}(x_i, y_j),$$

by designing a traversal (illustrated in Figure 6) that achieves this bound. Let $A \in \mathcal{S}_{n,m}$ be the alignment minimizing the expression, and note that $A = \{(\Delta + 1, 1), \dots, (\Delta + m, m)\}$ for some $0 \leq \Delta \leq n - m$. We first traverse $M^\kappa x_1 M^\kappa \dots M^\kappa x_\Delta$ together with the first symbol of y , M , which contributes a cost of $\sum_{i=1}^{\Delta} \delta_{\text{DTW}}(x_i, M) = \Delta(\ell_x M - s_x)$. For $i = 1, \dots, m$ we repeat the following: We traverse $M^\kappa x_{\Delta+i}$ together with $M^\kappa y_i$ by traversing M^κ and M^κ simultaneously, and x_i and y_i in a locally optimal manner; this incurs a cost of $\delta_{\text{DTW}}(x_{\Delta+i}, y_i)$ for each i . Finally, we traverse the last block M^κ in y with the current block M^κ in x , and then traverse the remainder $x_{\Delta+m+1} M^\kappa \dots M^\kappa x_n M^\kappa$ of x together with the last symbol of y , M . The total cost amounts to $\Delta(\ell_x M - s_x) + \sum_{i=1}^m \delta_{\text{DTW}}(x_{\Delta+i}, y_i) + (n - \Delta - m)(\ell_x M - s_x) = (n - m)(\ell_x M - s_x) + \sum_{(i,j) \in A} \delta_{\text{DTW}}(x_i, y_j)$.

In the remainder of the proof, we verify that

$$\delta_{\text{DTW}}(x, y) \geq (n - m)(\ell_x M - s_x) + \min_{A \in \mathcal{A}_{n,m}} \left[\sum_{(i,j) \in A} \delta_{\text{DTW}}(x_i, y_j) + (m - |A|) \max_{i,j} \delta_{\text{DTW}}(x_i, y_j) \right].$$

Let $T^* = ((a_1^*, b_1^*), \dots, (a_t^*, b_t^*))$ be an optimal traversal of (x, y) (see Section 2 for the definition of traversals). Substrings x' of x and y' of y are *paired* if for some index i in x' and some index j in y' we have $(i, j) = (a_{t'}^*, b_{t'}^*)$ for some $1 \leq t' \leq t$.

We call the i -th occurrence of M^κ in x the i -th M -block M_i^x of x , and similarly for y . Let $X := \{M_i^x \mid i \in [n + 1]\}$, $Y := \{M_j^y \mid j \in [m + 1]\}$ be the sets of all M -blocks of x and y , respectively. We define a bipartite graph G_M with vertex set $X \cup Y$, where M -blocks M_i^x and M_j^y are connected by an edge if and only if they are paired. We show the following properties of G_M .

Claim 6.5 (Planarity). For any paired M_i^x, M_j^y and paired $M_{i'}^x, M_{j'}^y$ we have $i \leq i'$ and $j \leq j'$ (or $i \geq i'$ and $j \geq j'$).

Proof. By monotonicity of traversals, for $k \leq k'$ we have $a_k^* \leq a_{k'}^*$ and $b_k^* \leq b_{k'}^*$. Thus, if $x[a_k^*]$ is in M_i^x and $x[a_{k'}^*]$ is in $M_{i'}^x$, then $i \leq i'$. Similarly, if $y[b_k^*]$ is in M_j^y and $y[b_{k'}^*]$ is in $M_{j'}^y$, then $j \leq j'$. Hence, for any paired M_i^x, M_j^y and $M_{i'}^x, M_{j'}^y$ we have $i \leq i', j \leq j'$ or $i \geq i', j \geq j'$. \square

Claim 6.6. G_M has no isolated vertices.

Proof. Assume that some M -block M_i^x is not paired with any M -block of y , and let i be maximal with this property. Note that $i < n + 1$, as the last M -block of x is always paired with the last M -block of y . Then there is some $j \in [m]$ such that M_i^x is paired with y_j , but M_i^x is not paired with any part of y outside y_j . By maximality of i and planarity, M_{j+1}^y is paired with x_i or M_{i+1}^x , as otherwise M_{i+1}^x is not paired with any $M_{j'}^y$.

We can find a cheaper traversal as follows. Consider the first time t_1 at which the traversal T^* is simultaneously at the first symbol of M_i^x and any symbol of y_j (this exists since M_i^x is paired to y_j , but to no part of y outside y_j), and any time t_2 at which T^* is at M_{j+1}^y and x_i or at M_{j+1}^y and M_{i+1}^x . Between t_1 and t_2 , T^* has a cost of at least $\delta_{\text{DTW}}(y', M^\kappa)$, where y' is any substring of y_j . By Claim 6.4.(3), this is at least $\kappa M/2$. We replace this part of T^* by traversing (i) the remainder of y_j with the first symbol of M_i^x , (ii) M_i^x with the necessary part of M_{j+1}^y , and (iii) the necessary part of x_i and M_{i+1}^x with the current symbol in y , M . By Claim 6.4.(1), this incurs a cost of at most $\delta_{\text{DTW}}(x_i, M) + \delta_{\text{DTW}}(y_j, M) = \ell_x M - s_x + \ell_y M - s_y \leq (\ell_x + \ell_y)M$. By our choice of $\kappa = 3(\ell_x + \ell_y)$, we improve the cost of the traversal, contradicting optimality of T^* . This shows that no vertex in X is isolated, we argue similarly for vertices in Y . \square

Claim 6.7. G_M contains no path of length 3.

Proof. Assume that G_M contains a path $M_i^x - M_j^y - M_{j'}^x - M_{j''}^y$. Without loss of generality we assume $i < i'$, the case $i > i'$ is symmetric. By planarity, we have $j < j'$. Since G_M has no isolated vertices and by planarity, every $M_{j''}^y$ with $i \leq i'' \leq i'$ is paired with M_j^y , so we can assume that $i' = i + 1$ (after replacing i with $i' - 1$). Similarly, we can assume $j' = j + 1$, and the path is $M_i^x - M_j^y - M_{i+1}^x - M_{j+1}^y$.

We can find a cheaper traversal as follows. Consider any time t_1 at which the traversal T^* is simultaneously at M_i^x and M_j^y (this exists since M_i^x and M_j^y are paired), and consider any time t_2 at which T^* is simultaneously at M_{i+1}^x and M_{j+1}^y . Between t_1 and t_2 , T^* traverses x_i with (parts of) M_j^y , and y_j with (parts of) M_{i+1}^x , which by Claim 6.4.(1) incurs a cost of at least $\delta_{\text{DTW}}(x_i, M) + \delta_{\text{DTW}}(M, y_j) \geq (\ell_x + \ell_y)M/2$. We replace this part of T^* by traversing (i) the remaining parts of M_i^x and M_j^y , (ii) x_i and y_j (in a locally optimal way), and (iii) the necessary parts of M_{i+1}^x and M_{j+1}^y . This incurs a cost of $\delta_{\text{DTW}}(x_i, y_j) < (\ell_x + \ell_y)M/2$ (by Claim 6.4.(4)), which contradicts optimality of T^* . \square

By the above two claims, G_M is a disjoint union of stars. By planarity and since G_M has no isolated vertices, the leaves of any star in G_M have to be consecutive M -blocks. Hence, we can write the components of G_M as C_1, \dots, C_s with $C_k = \{M_{i_k}^x\} \cup \{M_{j_k}^y, M_{j_k+1}^y, \dots, M_{j_k+d_k-1}^y\}$, and $C'_1, \dots, C'_{s'}$ with $C'_k = \{M_{j'_k}^y\} \cup \{M_{i'_k}^x, M_{i'_k+1}^x, \dots, M_{i'_k+d'_k-1}^x\}$.

Claim 6.8. We have $\sum_{k=1}^s d_k = m - s' + 1$ and $\sum_{k=1}^{s'} d'_k = n - s + 1$.

Proof. Since the components C_1, \dots, C_s and $C'_1, \dots, C'_{s'}$ partition G_M , restricted to Y we have $s' + \sum_{k=1}^s d_k = \sum_{k=1}^{s'} |C'_k \cap Y| + \sum_{k=1}^s |C_k \cap Y| = |Y| = m + 1$. The second claim follows analogously. \square

We construct an alignment by aligning the x_i, y_j that lie between two consecutive components of G_M . More formally, we define an alignment A by aligning $(i_k - 1, j_k - 1)$ (for all $k \in [s]$ with $i_k, j_k > 1$) and aligning $(i'_k - 1, j'_k - 1)$ (for all $k \in [s']$ with $i'_k, j'_k > 1$). Since G_M has no isolated vertices, A is a valid alignment. We have $|A| = s + s' - 1$, since only the leftmost component of G_M has $i_k = 1, j_k = 1, i'_k = 1$, or $j'_k = 1$, and all other components give rise to exactly one aligned pair.

Let us calculate the cost of T^* . Each y_j that lies between the leafs of a star C_k in G_M (i.e., $j_k \leq j < j_k + d_k$) has to be traversed together with (parts of) $M_{i_k}^x$. By Claim 6.4.(1), this incurs a cost of at least $\delta_{\text{DTW}}(M, y_j) = \ell_Y M - s_Y$. Likewise, each x_i that lies between the leafs of a star C'_k incurs a cost of at least $\ell_X M - s_X$. For any $(i, j) \in A$, x_i is traversed together with a substring of $M^\kappa y_j M^\kappa$, and y_j is traversed together with a substring of $M^\kappa x_i M^\kappa$. Hence, there are $a, a', b, b' \geq 0$ such that we traverse $M^a x_i M^{a'}$ together with $M^b y_j M^{b'}$. By Claim 6.4.(4), this incurs a cost of at least $\delta_{\text{DTW}}(x_i, y_j)$. In total, the cost of the optimal traversal T^* is

$$\delta_{\text{DTW}}(x, y) \geq \sum_{k=1}^s (d_k - 1)(\ell_Y M - s_Y) + \sum_{k=1}^{s'} (d'_k - 1)(\ell_X M - s_X) + \sum_{(i,j) \in A} \delta_{\text{DTW}}(x_i, y_j).$$

By Claim 6.8, we have $\sum_{k=1}^s (d_k - 1) = m - (s + s' - 1) = m - |A|$. Similarly, $\sum_{k=1}^{s'} (d'_k - 1) = n - |A| = (n - m) + (m - |A|)$. Additionally bounding $\ell_Y M - s_Y + \ell_X M - s_X \geq (\ell_X + \ell_Y)M/2 > \max_{i,j} \delta_{\text{DTW}}(x_i, y_j)$, we obtain the desired inequality

$$\delta_{\text{DTW}}(x, y) \geq (m - |A|) \max_{i,j} \delta_{\text{DTW}}(x_i, y_j) + (n - m)(\ell_X M - s_X) + \sum_{(i,j) \in A} \delta_{\text{DTW}}(x_i, y_j). \quad \square$$

7 Palindromic and Tandem Subsequences

In this section, we prove quadratic-time hardness of longest palindromic subsequence (LPS) and longest tandem subsequence (LTS) by presenting reductions from LCS. This proves Theorem 1.5. We will use the following simple facts about LCS, where we regard LCS as a minimization problem by defining $\delta_{\text{LCS}}(x, y) := |x| + |y| - 2|\text{LCS}(x, y)|$. In the whole section we let Σ be any alphabet with $0, 1 \in \Sigma$.

Fact 7.1. *Let z, w be binary strings and $\ell, k \in \mathbb{N}_0$. Then we have (1) $\delta_{\text{LCS}}(1^k z, 1^k w) = \delta_{\text{LCS}}(z, w)$, (2) $\delta_{\text{LCS}}(1^k z, w) \geq \delta_{\text{LCS}}(z, w) - k$ and (3) $\delta_{\text{LCS}}(0^\ell z, 1^k w) \geq \min\{k, \delta_{\text{LCS}}(z, 1^k w) + \ell\}$. We obtain symmetric statements by flipping all bits and by reversing all involved strings.*

Proof. (1) is a restatement of Claim 4.5.(1). (2) follows from Fact 5.5.(2). For (3), fix a LCS s of $(0^\ell z, 1^k w)$. If s starts with a 0, then it does not contain the leading 1^k of the second argument, leaving at least k symbols unmatched, so that $\delta_{\text{LCS}}(0^\ell z, 1^k w) \geq k$. Otherwise, if s starts with a 1, then it does not contain the leading 0^ℓ of the first argument, so that $|\text{LCS}(0^\ell z, 1^k w)| = |\text{LCS}(z, 1^k w)|$. Then we have $\delta_{\text{LCS}}(0^\ell z, 1^k w) = |0^\ell z| + |1^k w| - 2|\text{LCS}(0^\ell z, 1^k w)| = \ell + |z| + |1^k w| - 2|\text{LCS}(z, 1^k w)| = \ell + \delta_{\text{LCS}}(z, 1^k w)$. \square

7.1 Longest Palindromic Subsequence

We show that computing the length of the longest palindromic subsequence is essentially computationally equivalent to computing the length of the longest common subsequence of two strings. For completeness, we provide the following well known result which shows that LPS can be reduced to LCS in linear time. Recall that for a string x we denote the reversed string by $\text{rev}(x)$.

Fact 7.2 (Folklore). *For any input $x \in \Sigma^*$, we have $|\text{LPS}(x)| = |\text{LCS}(x, \text{rev}(x))|$.*

Proof. Let p be a palindromic subsequence of x . Then $p = \text{rev}(p)$ is a common subsequence of x and $\text{rev}(x)$, yielding $|\text{LCS}(x, \text{rev}(x))| \geq |\text{LPS}(x)|$.

For the other direction, let c be any LCS of x and $\text{rev}(x)$ of length ℓ . It remains to show that we can find a palindromic subsequence p of x with $|p| \geq \ell$ (observe that c itself is not necessarily a palindrome). Note that c gives rise to a sequence of pairs $(a_1, b_1), \dots, (a_\ell, b_\ell)$ such that $a_1 < \dots < a_\ell$, $b_1 > \dots > b_\ell$, and $c = (x[a_1], \dots, x[a_\ell]) = (x[b_1], \dots, x[b_\ell])$. Define $m := \lfloor \frac{\ell}{2} \rfloor + 1$. If $a_m \leq b_m$, then $a_1 < \dots < a_m \leq b_m < \dots < b_1$ and hence $(x[a_1], \dots, x[a_{m-1}], x[a_m], x[b_{m-1}], \dots, x[b_1])$ is a palindromic subsequence of x of length $2m-1 = 2\lfloor \frac{\ell}{2} \rfloor + 1 \geq \ell$. Otherwise, i.e., if $a_m > b_m$, then $b_\ell < \dots < b_m < a_m < \dots < a_\ell$ gives rise to the palindromic subsequence $(x[b_\ell], \dots, x[b_m], x[a_m], \dots, x[a_\ell])$ of x with length $2(\ell - m + 1) = 2\ell - 2\lfloor \frac{\ell}{2} \rfloor \geq \ell$. \square

To prove our lower bound for computing a longest palindromic subsequence of a string x , we present a simple reduction from LCS to LPS, and then appeal to our lower bound for LCS, which is equivalent to $\text{Edit}(1, 1, 0, 2)$, see Theorem 1.2.

Theorem 7.3. *On input $x, y \in \Sigma^*$, we can compute, in time $\mathcal{O}(|x| + |y|)$, a string $z \in \Sigma^*$ and $\kappa \in \mathbb{N}$ such that $|\text{LPS}(z)| = 3\kappa + 2|\text{LCS}(x, y)|$.*

Proof. Let $\kappa := 2(\ell_x + \ell_y + 1)$, where $\ell_x := |x|$, $\ell_y := |y|$. We define

$$z := x 0^\kappa 1^\kappa 0^\kappa \text{rev}(y).$$

Clearly, z and κ can be computed in time $\mathcal{O}(\ell_x + \ell_y)$. Let s be a LCS of x and y . Then $s 0^\kappa 1^\kappa 0^\kappa \text{rev}(s)$ is a palindromic subsequence of z , which proves $|\text{LPS}(z)| \geq 3\kappa + 2|\text{LCS}(x, y)|$.

To show $|\text{LPS}(z)| \leq 3\kappa + 2|\text{LCS}(x, y)|$, fix a LPS p of z and let ℓ be its length. We define $m := \lfloor \frac{\ell}{2} \rfloor$ and denote by $p_1 = (p[1], \dots, p[m])$ the first ‘‘half’’ of p . Let $z_1 = (z[1], \dots, z[i])$ be the shortest prefix of z that contains p_1 as a subsequence and let $z_2 := (z[i+1], \dots, z[\ell])$ be the remainder of z . Then p_1 , which by definition equals $(p[\ell], \dots, p[\ell - m + 1])$, is a subsequence of $\text{rev}(z_2)$. This shows that if ℓ is even, then $\ell \leq 2|\text{LCS}(z_1, \text{rev}(z_2))|$. If ℓ is odd, we may without loss of generality assume that $p[m+1] = z_2[1]$. Hence $\text{rev}(p_1)$ is a subsequence of $z'_2 := (z_2[2], \dots, z_2[\ell])$, so that $\ell \leq 2|\text{LCS}(z_1, \text{rev}(z'_2))| + 1$. It remains to show that (i) $|\text{LCS}(z_1, \text{rev}(z_2))| \leq \frac{3}{2}\kappa + |\text{LCS}(x, y)|$ and (ii) $|\text{LCS}(z_1, \text{rev}(z'_2))| \leq \frac{3}{2}\kappa + |\text{LCS}(x, y)| - \frac{1}{2}$.

Assume that $|z_1| \leq \ell_x + \kappa$ or $|z_2| \leq (\ell_y + 1) + \kappa$, then by $|\text{LCS}(x, y)| \leq \min\{|x|, |y|\}$ we obtain that $|\text{LCS}(z_1, \text{rev}(z'_2))| \leq |\text{LCS}(z_1, \text{rev}(z_2))| \leq \max\{\ell_x, \ell_y + 1\} + \kappa < \frac{3}{2}\kappa + |\text{LCS}(x, y)|$. Hence without loss of generality, $z_1 = x 0^\kappa 1^a$ and $z_2 = 1^a 0^\kappa \text{rev}(y)$ with $a' \geq 1$, where we assume that $a' \geq a$ since the other case is symmetric. Note that (i) and (ii) are equivalent to $\delta_{\text{LCS}}(z_1, \text{rev}(z_2)) \geq \delta_{\text{LCS}}(x, y)$ and $\delta_{\text{LCS}}(z_1, \text{rev}(z'_2)) \geq \delta_{\text{LCS}}(x, y)$, respectively. We compute

$$\begin{aligned} \delta_{\text{LCS}}(z_1, \text{rev}(z_2)) &= \delta_{\text{LCS}}(x 0^\kappa 1^a, y 0^\kappa 1^{a'}) \\ &= \delta_{\text{LCS}}(x 0^\kappa, y 0^\kappa 1^{a'-a}) && \text{(by Fact 7.1.(1))} \\ &\geq \min\{\kappa, \delta_{\text{LCS}}(x 0^\kappa, y 0^\kappa)\} && \text{(by Fact 7.1.(3))} \\ &= \min\{\kappa, \delta_{\text{LCS}}(x, y)\} = \delta_{\text{LCS}}(x, y). && \text{(by Fact 7.1.(1)).} \end{aligned}$$

By replacing a' by $a' - 1 \geq 0$, we obtain $\delta_{\text{LCS}}(z_1, \text{rev}(z'_2)) \geq \delta_{\text{LCS}}(x, y)$ by the same calculation. This yields $|\text{LPS}(z)| = \ell \leq 3\kappa + 2|\text{LCS}(x, y)|$, as desired. \square

7.2 Longest Tandem Subsequence

As for LPS, our lower bound for LTS follows from a simple reduction from LCS and appealing to our lower bound for LCS of Theorem 1.2.

Theorem 7.4. *On input $x, y \in \Sigma^*$, we can compute, in time $\mathcal{O}(|x| + |y|)$, a string $z \in \Sigma^*$ and $\kappa \in \mathbb{N}$ such that $|\text{LTS}(z)| = 4\kappa + 2|\text{LCS}(x, y)|$.*

Proof. Let $\kappa := \ell_x + \ell_y$, where $\ell_x := |x|$ and $\ell_y := |y|$. We define

$$z := 0^\kappa x 1^\kappa 0^\kappa y 1^\kappa.$$

Clearly, z can be computed in time $\mathcal{O}(\ell_x + \ell_y)$. Let s be a LCS of x and y . Then $t := t' t'$ with $t' := 0^\kappa s 1^\kappa$ is a tandem subsequence of z . Hence, we have $|\text{LTS}(z)| \geq |t| = 4\kappa + 2|\text{LCS}(x, y)|$.

To show $|\text{LTS}(z)| \leq 4\kappa + 2|\text{LCS}(x, y)|$, fix a LTS $t = t' t'$ of z . Let i be the smallest index such that t' is a subsequence of $z_1 := (z[1], \dots, z[i])$ and let $z_2 := (z[i+1], \dots, z[|z|])$. By choice of t , t' is also a subsequence of z_2 , so that $|\text{LTS}(z)| = 2|t'| \leq 2|\text{LCS}(z_1, z_2)|$. Thus, it remains to prove that $2|\text{LCS}(z_1, z_2)| \leq 4\kappa + 2|\text{LCS}(x, y)|$.

Assume that $|z_1| \leq \kappa + \ell_x$ or $|z_2| \leq \kappa + \ell_y$. Then, using $|\text{LCS}(x, y)| \leq \min\{|x|, |y|\}$, we conclude that $2|\text{LCS}(z_1, z_2)| \leq 2\kappa + 2(\ell_x + \ell_y) \leq 4\kappa + 2|\text{LCS}(x, y)|$.

Hence, without loss of generality, we have (i) $z_1 = 0^\kappa x 1^\ell$ and $z_2 = 1^{\ell'} 0^\kappa y 1^\kappa$ or (ii) $z_1 = 0^\kappa x 1^\kappa 0^\ell$ and $z_2 = 0^{\ell'} y 1^\kappa$, for some ℓ, ℓ' with $\ell + \ell' = \kappa$. We only consider case (i), since case (ii) is symmetric. Note that $2|\text{LCS}(z_1, z_2)| \leq 4\kappa + 2|\text{LCS}(x, y)|$ is equivalent to $\delta_{\text{LCS}}(z_1, z_2) \geq \delta_{\text{LCS}}(x, y)$. We obtain

$$\begin{aligned} \delta_{\text{LCS}}(z_1, z_2) &= \delta_{\text{LCS}}(0^\kappa x 1^\ell, 1^{\ell'} 0^\kappa y 1^\kappa) \\ &\geq \min\{\kappa, \delta_{\text{LCS}}(0^\kappa x 1^\ell, 0^\kappa y 1^\kappa) + \ell'\} && \text{(by Fact 7.1.(3))} \\ &= \min\{\kappa, \delta_{\text{LCS}}(x 1^\ell, y 1^\kappa) + \ell'\} && \text{(by Fact 7.1.(1))} \\ &= \min\{\kappa, \delta_{\text{LCS}}(x, y 1^{\kappa-\ell}) + \ell'\} && \text{(by Fact 7.1.(1))} \\ &\geq \min\{\kappa, \delta_{\text{LCS}}(x, y) - (\kappa - \ell) + \ell'\} && \text{(by Fact 7.1.(2))} \\ &= \min\{\kappa, \delta_{\text{LCS}}(x, y)\} = \delta_{\text{LCS}}(x, y), \end{aligned}$$

which proves the desired inequality $2|\text{LCS}(z_1, z_2)| \leq 4\kappa + 2|\text{LCS}(x, y)|$. \square

8 Conclusion

We prove conditional lower bounds for natural polynomial-time problems: Edit distance for general operation costs, including its special case longest common subsequence, dynamic time warping, longest palindromic subsequence, and longest tandem subsequence. Our results give strong evidence that the known algorithms for these problems are optimal up to lower order factors, even restricted to binary strings and one-dimensional curves, respectively. We hope that the underlying framework will find application in hardness proofs for further similarity measures, and that the studied problems serve as starting points for further reductions.

It remains an open question whether constant-factor approximations running in strongly subquadratic time can be ruled out for the above problems assuming SETH. Furthermore, most polynomial-time lower bounds show quadratic-time barriers, and it is challenging to prove matching SETH-based lower bounds for problems with, say, cubic or $\mathcal{O}(n^{3/2})$ -time algorithms (only few results are known in this direction [1, 17]).

References

- [1] Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Quadratic-time hardness of LCS and other sequence similarity measures, 2015. arXiv 1501.07053.
- [2] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS'14)*, 2014. To appear.
- [3] Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Proc. 41st International Colloquium on Automata, Languages, and Programming (ICALP'14)*, pages 39–51, 2014.
- [4] Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'15)*, pages 218–230.
- [5] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. *54th Annual Symposium on Foundations of Computer Science (FOCS'10)*, pages 377–386, 2010.
- [6] Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false), 2014. arXiv 1412.0348.
- [7] Lasse Bergroth, Harri Hakonen, and Timo Raita. A survey of longest common subsequence algorithms. In *Proc. 7th Int. Symp. on String Processing and Information Retrieval (SPIRE'00)*, pages 39–48, 2000.
- [8] Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS'14)*, 2014. To appear.
- [9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.
- [10] James R. Driscoll, Neil Sarnak, Daniel Dominic Sleator, and Robert E. Tarjan. Making data structures persistent. In *Proc. 18th Annual ACM Symposium on Theory of Computing (STOC'86)*, pages 109–121, 1986.
- [11] Anka Gajentaan and Mark H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry: Theory and Applications*, 5(3):165–185, 1995.
- [12] Daniel S. Hirschberg. Algorithms for the longest common subsequence problem. *Journal of the ACM*, 24(4):664–675, 1977.
- [13] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.
- [14] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

- [15] Adrian Kosowski. An efficient algorithm for the longest tandem scattered subsequence problem. In *String Processing and Information Retrieval*, volume 3246 of *LNCS*, pages 93–100. Springer, 2004.
- [16] William J. Masek and Michael S. Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System sciences*, 20(1):18–31, 1980.
- [17] Mihai Pătraşcu and Ryan Williams. On the possibility of faster SAT algorithms. In *Proc. 21st ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*, pages 1065–1075, 2010.
- [18] Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. An improved exponential-time algorithm for k-sat. *Journal of the ACM*, 52(3):337–364, 2005.
- [19] Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proc. 45th Annual ACM Symposium on Symposium on Theory of Computing (STOC'13)*, pages 515–524, 2013.
- [20] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978.
- [21] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.
- [22] João Carlos Setubal and João Meidanis. *Introduction to Computational Molecular Biology*. Computer Science Series. PWS Pub., 1997.
- [23] Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoretical Computer Science*, 348(2):357–365, 2005.