



Hairpin Completion Distance Lower Bound

Itai Boneh  

Reichman University and University of Haifa, Israel

Dvir Fried  

Bar Ilan University, Ramat Gan, Israel

Shay Golan  

Reichman University and University of Haifa, Israel

Matan Kraus  

Bar Ilan University, Ramat Gan, Israel

Abstract

Hairpin completion, derived from the hairpin formation observed in DNA biochemistry, is an operation applied to strings, particularly useful in DNA computing. Conceptually, a right hairpin completion operation transforms a string S into $S \cdot S'$ where S' is the reverse complement of a prefix of S . Similarly, a left hairpin completion operation transforms a string S into $S' \cdot S$ where S' is the reverse complement of a suffix of S . The hairpin completion distance from S to T is the minimum number of hairpin completion operations needed to transform S into T . Recently Boneh et al. [3] showed an $O(n^2)$ time algorithm for finding the hairpin completion distance between two strings of length at most n . In this paper we show that for any $\varepsilon > 0$ there is no $O(n^{2-\varepsilon})$ -time algorithm for the hairpin completion distance problem unless the Strong Exponential Time Hypothesis (SETH) is false. Thus, under SETH, the time complexity of the hairpin completion distance problem is quadratic, up to sub-polynomial factors.

2012 ACM Subject Classification Theory of computation \rightarrow Design and analysis of algorithms

Keywords and phrases Fine-grained complexity, Hairpin completion, LCS

Digital Object Identifier 10.4230/LIPIcs.CPM.2024.11

Related Version *Full Version:* <https://arxiv.org/abs/2404.11673> [2]

Funding *Itai Boneh:* supported by Israel Science Foundation grant 810/21.

Dvir Fried: supported by ISF grant no. 1926/19, by a BSF grant 2018364, and by an ERC grant MPM under the EU's Horizon 2020 Research and Innovation Programme (grant no. 683064).

Shay Golan: supported by Israel Science Foundation grant 810/21.

Matan Kraus: supported by the ISF grant no. 1926/19, by the BSF grant 2018364, and by the ERC grant MPM under the EU's Horizon 2020 Research and Innovation Programme (grant no. 683064).

1 Introduction

Hairpin completion [6], derived from the hairpin formation observed in DNA biochemistry, is an operation applied to strings, particularly useful in DNA computing [10, 9, 8, 7]. Consider a sequences over an alphabet Σ with involution $\text{Inv} : \Sigma \rightarrow \Sigma$ assigning for every $\sigma \in \Sigma$ an inverse symbol $\bar{\sigma}$. For a string $S \in \Sigma^*$, a left hairpin completion transforms S into $\overleftarrow{S'} \cdot S$, where S' is a suffix of S , and for any $X \in \Sigma^*$ we define $\overleftarrow{X} = \overleftarrow{X[|X|]} \cdot \overleftarrow{X[|X| - 1]} \cdot \dots \cdot \overleftarrow{X[1]}$. This operation can only be applied under the restriction that the suffix S' is preceded by the symbol $\overleftarrow{S[1]}$. Similarly, a right hairpin completion transforms S into $S \cdot \overleftarrow{S'}$ where S' is a prefix of S followed by $\overleftarrow{S[|S|]}$.

Several problems regarding hairpin completion were studied [11, 12, 13, 14, 3]. In this paper, we consider the hairpin completion distance problem. In this problem, we are given two strings x and y and our goal is to compute the minimum number of hairpin completion



© Itai Boneh, Dvir Fried, Shay Golan, and Matan Kraus;
licensed under Creative Commons License CC-BY 4.0

35th Annual Symposium on Combinatorial Pattern Matching (CPM 2024).

Editors: Shunsuke Inenaga and Simon J. Puglisi; Article No. 11; pp. 11:1–11:16



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

11:2 Hairpin Completion Distance Lower Bound

operations one has to apply on y to transform y into x , or to report that there is no sequence of hairpin completion operation can turn y into x . In 2009, Manea, Martín-Vide and Mitrana [12] proposed the problem and introduced a cubic time $O(n^3)$ algorithm (where $n = |x|$). Later Manea [11] introduced a faster algorithm that runs in $O(n^2 \log n)$ time. Recently, Boneh et al. [3] showed that the time complexity of the problem is $O(n^2)$. Moreover, Boneh et al. posed the following open problem.

► **Problem 1.** *Can one prove a lower bound for hairpin completion distance computation that matches the $O(n^2)$ upper bound?*

In this paper, we show that for every $\varepsilon > 0$, there is no $O(n^{2-\varepsilon})$ time algorithm for computing the hairpin completion distance from y to x , unless the Strong Exponential Time Hypothesis (SETH) [5] is false. Thus, we provide a conditional lower bound matching the upper bound of [3] up to sub-polynomial factors.

► **Theorem 2.** *Let $\varepsilon > 0$. If there is an algorithm that computes the hairpin completion distance from y to x in $O(|x|^{2-\varepsilon})$ time, then SETH is false. This holds even if the input strings are over an alphabet of size 4.*

We note that due to the relationship between hairpin operations and DNA biochemistry, a typical output for a hairpin-related problem is over the alphabet $\{A, C, G, T\}$ of size 4. Hence, our lower bound applies to a natural set of practical inputs.

Theorem 2 is proven by reducing Longest Common Subsequence (LCS) problem to the hairpin completion distance problem. Namely, for two ternary strings S and T , we show a linear time construction of a pair of strings x and y such that $\text{LCS}(S, T)$ can be computed in linear time from the hairpin completion distance from y to x . The hardness of hairpin completion computation follows from the conditional lower bound on the LCS problem [4, 1]. We refer the reader to Section 3 where we introduce the reduction and to Section 3.1 where we provide a high-level discussion regarding the correctness of our construction.

2 Preliminaries

For $i, j \in \mathbb{N}$ let $[i..j] = \{k \in \mathbb{N} \mid i \leq k \leq j\}$. We denote $[i] = [1..i]$.

A string S over an alphabet Σ is a sequence of characters $S = S[1]S[2] \dots S[|S|]$. For $i, j \in [|S|]$, we call $S[i..j] = S[i]S[i+1] \dots S[j]$ a *substring* of S . If $i = 1$, $S[i..j]$ is a prefix of S , and if $j = |S|$, $S[i..j]$ is a suffix of S . Let x and y be two strings over an alphabet Σ . $x \cdot y$ is the concatenation of x and y . For strings x_1, x_2, \dots, x_m , we denote as $\bigodot_{i=1}^m = x_1 \cdot x_2 \cdot \dots \cdot x_m$. For a string x and $k \in \mathbb{N}$ we write the concatenation of x to itself k times as x^k . For a symbol $\sigma \in \Sigma$, we denote as $\#_\sigma(x) = |\{i \in [|x|] \mid x[i] = \sigma\}|$ the number of occurrences of σ in x . We say that a string y occurs in x (or that x contains an occurrence of y) if there is an index $i \in [|x| - |y| + 1]$ such that $x[i..i + |y| - 1] = y$.

For two sets of strings \mathcal{S} and \mathcal{T} , we define the set of strings $\mathcal{S} * \mathcal{T} = \{s \cdot x \cdot t \mid s \in \mathcal{S}, x \in \Sigma^*, t \in \mathcal{T}\}$. We use the notations $\mathcal{S}^* = \mathcal{S} * \Sigma^*$ and $*\mathcal{S} = \Sigma^* * \mathcal{S}$. When using $*$ notation, we sometimes write $s \in \Sigma^*$ to denote the set $\{s\}$ (for example, 0^* is the set of all strings starting with 0).

Hairpin Operations. Let $\text{Inv} : \Sigma \rightarrow \Sigma$ be a permutation on Σ . We say that Inv is an *inverse function* on Σ if $\text{Inv} = \text{Inv}^{-1}$ and $\text{Inv}(\sigma) \neq \sigma$ for every $\sigma \in \Sigma$. Throughout this paper, we discuss strings over alphabet $\Sigma = \{0, 1\}$ with $\text{Inv}(\sigma) = 1 - \sigma$. For every symbol in σ , we denote $\bar{\sigma} = \text{Inv}(\sigma)$. We further extend this notation to strings by denoting $\bar{x} = \overline{x[1]} \cdot \overline{x[2]} \cdot \dots \cdot \overline{x[|x|]}$. We denote $\overleftarrow{x} = \bar{x}[|x|] \cdot \bar{x}[|x| - 1] \cdot \dots \cdot \bar{x}[2] \cdot \bar{x}[1]$.

We define several types of hairpin operations that can be applied to a string over Σ with an inverse function on Σ . In [6], hairpin operations are defined as follows.

► **Definition 3 (Hairpin Operations).** *Let $S \in \Sigma^*$. A right hairpin completion of length $\ell \in [|S|]$ transforms S into $S \cdot \overleftarrow{S[1..\ell]}$. A right hairpin completion operation of length ℓ can be applied on S only if $S[\ell + 1] = \overleftarrow{S[|S|]}$. Similarly, a left hairpin completion of length ℓ transforms S into $\overleftarrow{S[|S| - \ell + 1..|S|]} \cdot S$. A left hairpin completion of length ℓ can be applied to S only if $S[|S| - \ell] = \overleftarrow{S[1]}$. A right (resp. left) hairpin deletion operation of length $\ell \in [\lfloor \frac{|S|}{2} \rfloor]$ transforms a string S into a prefix (resp. suffix) S' of S such that S can be obtained from S' by a valid right (resp. left) hairpin completion of length ℓ .*

Throughout this paper, we use the following modified definition of hairpin operation, which removes the constraints regarding $S[1]$ and $S[|S|]$.

► **Definition 4 (Hairpin Operations, Modified definition).** *Let $S \in \Sigma^*$. A right hairpin completion of length $\ell \in [|S|]$ transforms S into $S \cdot \overleftarrow{S[1..\ell]}$. Similarly, a left hairpin completion of length ℓ transforms S into $\overleftarrow{S[|S| - \ell + 1..|S|]} \cdot S$. A right (resp. left) hairpin deletion of length $\ell \in [\lfloor \frac{|S|}{2} \rfloor]$ operation transforms a string S into a prefix (resp. suffix) S' of S such that S can be obtained from S' by a valid right (resp. left) hairpin completion of length ℓ .*

We highlight that the modified definition is *not* equivalent to the definition of [6]. Even though the paper is phrased in terms of the modified definition, we emphasize that Theorem 2 is correct with respect to both definitions. In the full version of this paper, we discuss the machinery required to make our hardness result applicable to Definition 3. The complete details for bridging this gap are developed in the full version of this paper [2].

Let x and y be two strings. We denote by $\text{HDD}(x, y)$ (resp. $\text{HCD}(x, y)$) the minimum number of hairpin deletion (resp. completion) operations required to transform x into y , counting both left and right operations. Note that $\text{HDD}(x, y) = \text{HCD}(y, x)$.

For the sake of analysis, we define the following graph.

► **Definition 5 (Hairpin Deletion Graph).** *For a string x the Hairpin Deletion Graph $G_x = (V, E)$ is defined as follows. V is the set of all substrings of x , and $(u, v) \in E$ if v can be obtained from u in a single hairpin deletion operation.*

We define the distance between two vertices s and t in a graph G (denoted as $\text{dist}_G(s, t)$) to be the minimal length (number of edges) of a path from s to t in G (of ∞ if there is no such path). Note that for a source string x and a destination string y , it holds that $\text{HDD}(x, y) = \text{dist}_{G_x}(x, y)$. We distinguish between two types of edges outgoing from $x[i..j]$. An edge of the form $x[i..j] \rightarrow x[i+\ell..j]$ for some $\ell \in \mathbb{N}$ is called a *left edge* and it corresponds to a left hairpin deletion operation of length ℓ . Similarly, an edge of the form $x[i..j] \rightarrow x[i..j-\ell]$ for some $\ell \in \mathbb{N}$ is called a *right edge* and it corresponds to a right hairpin deletion operation of length ℓ . When a path p in G_x traverses a left (resp. right) edge outgoing from v , we say that p applies a left (resp. right) hairpin deletion to v . For a path p we denote by $\text{cost}(p)$ the length of p .

Hairpin Deletion. Since the paper makes intensive use of hairpin deletion notations, we introduce an alternative, more intuitive definition for hairpin deletion, equivalent to Definition 4. For a string S , if for some $\ell \in [\lfloor \frac{|S|}{2} \rfloor]$ we have $S[1..\ell] = \overleftarrow{S[|S| - \ell + 1..|S|]}$ then a left (resp. right) hairpin deletion operation transforms S into $S[\ell + 1..|S|]$ (resp. $S[1..|S| - \ell]$). In particular, if $S[1] \neq \overleftarrow{S[|S|]}$ then there is no valid hairpin deletion operation on S .

11:4 Hairpin Completion Distance Lower Bound

Longest Common Subsequence. A subsequence of a string S of length n is a string X of length ℓ such that there is an increasing sequence $1 \leq i_1 < i_2 < \dots < i_\ell \leq n$ satisfying $X[k] = S[i_k]$ for every $k \in [\ell]$. For two strings S and T , a string X is a common subsequence of S and T if X is a subsequence of both S and T . The LCS problem is, given two strings S and T of length at most n , compute the maximum *length* of a common subsequence of S and T , denoted as $\text{LCS}(S, T)$.

Bringmann and Künnemann [4] have shown the following.

► **Fact 6 (Hardness of LCS).** *For every $\varepsilon > 0$, there is no $O(n^{2-\varepsilon})$ -time algorithm that solves the LCS problem for ternary input strings unless SETH is false.*

Fibonacci sequence. The Fibonacci sequence is defined as follows. $\text{Fib}(0) = 1$, $\text{Fib}(1) = 1$ and for all integer $i > 1$ we have $\text{Fib}(i) = \text{Fib}(i-1) + \text{Fib}(i-2)$. The inverse function $\text{Fib}^{-1} : \mathbb{R} \rightarrow \mathbb{N}$ is defined as $\text{Fib}^{-1}(x) = \min\{y \in \mathbb{N} \mid \text{Fib}(y) \geq x\}$.

3 The Reduction

Here we introduce a reduction from the LCS problem on ternary strings. We also provide in Section 3.1 a high-level discussion of why the reduction should work.

We present a linear time algorithm such that given two strings $S, T \in \{0, 1, 2\}^*$, constructs two (binary) strings x and y with $|x| = O(|S| + |T|)$ and $|y| = O(1)$. The strings x and y have the property that $\text{HDD}(x, y)$ can be used to infer $\text{LCS}(S, T)$ in linear time. Thus, by Fact 6, we deduce that any algorithm computing $\text{HDD}(x, y)$ cannot have running time $O(|x|^{2-\varepsilon})$ for any $\varepsilon > 0$ (assuming SETH).

We use several types of gadgets. Let:

- $I_L(0) = (010^3)^{i_0}$
- $I_L(1) = (010^5)^{i_1}$
- $I_L(2) = (010^7)^{i_2}$
- $P_L = (010^9)^p$
- $\text{Sync}_L = 01$
- $I_R(0) = (\overline{0^3 1 0})^{i_0} = \overleftarrow{I_L(0)}$
- $I_R(1) = (\overline{0^5 1 0})^{i_1} = \overleftarrow{I_L(1)}$
- $I_R(2) = (\overline{0^7 1 0})^{i_2} = \overleftarrow{I_L(2)}$
- $P_R = (\overline{0^9 1 0})^p = \overleftarrow{P_L}$
- $\text{Sync}_R = \overline{010}$

with $i_0 = 55$, $i_1 = 54$, $i_2 = 53$ and $p = 144$. We call $I_L(0)$, $I_L(1)$ and $I_L(2)$ left *information* gadgets and $I_R(0)$, $I_R(1)$ and $I_R(2)$ right information gadgets. We say that $I_L(\alpha)$ and $I_R(\beta)$ *match* if $\alpha = \beta$ or *mismatch* otherwise. P_L and P_R are called left and right *protector* gadgets, respectively. Sync_L and Sync_R are called left and right *synchronizer* gadgets, respectively. We say that two gadgets g_1, g_2 are *symmetric* if $g_2 = \overleftarrow{g_1}$. Specifically, $(I_L(0), I_R(0))$, $(I_L(1), I_R(1))$, $(I_L(2), I_R(2))$ and (P_L, P_R) are the pairs of symmetric gadgets. We emphasize that Sync_L and Sync_R are *not* symmetric.

Using the gadgets above, we define 6 mega gadgets encoding characters from S and T . For $\alpha \in \{0, 1, 2\}$ we define

$$E_L(\alpha) = P_L \cdot \text{Sync}_L \cdot I_L(\alpha) \cdot \text{Sync}_L \quad \text{and} \quad E_R(\alpha) = \text{Sync}_R \cdot I_R(\alpha) \cdot \text{Sync}_R \cdot P_R.$$

Finally, we define $y = \boxed{P_L \cdot \text{Sync}_L \cdot 01 \cdot 11\bar{1}\bar{1} \cdot \bar{1}\bar{0} \cdot \text{Sync}_R \cdot P_R}$ and $x = \left(\bigodot_{i=1}^{|S|} E_L(S[i]) \right) \cdot y \cdot \left(\bigodot_{i=|T|}^1 E_R(T[i]) \right)$. Note that on the suffix of x we concatenate the elements of T in *reverse* order.

In the remainder of this paper, we only use ' x ' and ' y ' to refer to the strings defined above. We define notations for indices in x which are endpoints of protector and information gadgets as follows. For $\ell \in [|S| + 1]$, let $\text{left}_\ell^P = 1 + \sum_{j=1}^{\ell-1} |E_L(S[j])|$ be the leftmost index of the ℓ th P_L gadget (from the left) in x . Notice that $\text{left}_{|S|+1}^P$ corresponds to the left P_L gadget contained in y . For $r \in [|T| + 1]$, let $\text{right}_r^P = |x| - \sum_{j=1}^{r-1} |E_R(T[j])|$ be the rightmost index of the r th P_R gadget (from the right) in x . Notice that $\text{right}_{|T|+1}^P$ corresponds to the right P_R gadget contained in y . For $\ell \in [|S|]$, let $\text{left}_\ell^I = \text{left}_\ell^P + |P_L| + |\text{Sync}_L|$ be the leftmost index of the ℓ th information gadget (from the left) in x . For $r \in [|T|]$, let $\text{right}_r^I = \text{right}_r^P - |P_R| - |\text{Sync}_R|$ be the rightmost index of the r th information gadget (from the right) in x .

The rest of the paper is dedicated for proving the following property of x and y .

► **Lemma 7** (Reduction Correctness). *For some constants $D(0), D(1), D(2)$ and B we have: $\text{HDD}(x, y) = \sum_{\alpha \in \{0,1,2\}} D(\alpha)(\#_\alpha(S) + \#_\alpha(T)) - \text{LCS}(S, T) \cdot B$.*

Note that $\#_\alpha(S)$ and $\#_\alpha(T)$ can be easily computed for all values of α in linear time. Therefore, if $\text{HDD}(x, y)$ can be computed in $O(n^{2-\varepsilon})$ time for some $\varepsilon > 0$, LCS can be computed in $O(n + n^{2-\varepsilon})$ (the values of the constants are fixed in the proof). Since $\text{HDD}(x, y) = \text{HCD}(y, x)$, hairpin deletion and hairpin completion distance are computationally equivalent. Recall that $\text{HDD}(x, y)$ refers to the *modified* hairpin deletion distance (Definition 4). In order to bridge the gap to the original definition of hairpin deletion distance, we provide a linear time construction of strings x' and y' such that $\text{HDD}'(x', y') = \text{HDD}(x, y)$ in the full version of this paper [2]. Here, $\text{HDD}'(x', y')$ denotes the hairpin deletion distance from x' to y' as defined in Definition 3. It clearly follows from this construction and the above discussion that $\text{HDD}'(x', y')$ can not be computed in $O(n^{2-\varepsilon})$, unless SETH is false. Thus, proving Theorem 2.

3.1 Intuition for the Reduction Correctness

We provide some high-level discussion regarding the correctness of the construction. First, notice that y has a single occurrence in x . Therefore, a sequence of hairpin deletion operations transforming x into y has to delete all mega gadgets. Consider an intermediate step in a deletion sequence in which the substring $x[i..j]$ is obtained such that i is the leftmost index of some left gadget g_i and j is the rightmost index of some right gadget g_j .

If g_i and g_j are not symmetric, the next hairpin deletion would not be able to make much progress. This is due to the 1 symbols in g_i and the $\bar{1}$ symbols in g_j being separated by a different number of 0's and $\bar{0}$'s. For the goal of minimizing the number of deletions for removing all mega gadgets, this is a significant set-back, as either g_i or g_j would have to be removed using roughly $\#_1(g_i)$ (or $\#_{\bar{1}}(g_j)$) deletions.

Now consider the case in which g_i and g_j are symmetric to each other. In this case, either one of them can be deleted using a single hairpin deletion. However, note that greedily removing g_i will put us in the asymmetric scenario. Notice that there is another possible approach for deleting symmetric gadgets - a synchronized deletion. In this process, g_i and g_j are both deleted gradually. One can easily figure out a way to apply such synchronized deletion using roughly $\log(\#_1(g_i))$ steps.

Think of a scenario in which $i = \text{left}_\ell^P$ and $j = \text{right}_r^P$ for some ℓ and r , i.e. i and j are a leftmost index and a rightmost index of left and right mega gadgets m_ℓ and m_r , respectively. Initially, both i and j are in the beginning of protector gadgets p_ℓ and p_r . If m_ℓ and m_r are

11:6 Hairpin Completion Distance Lower Bound

mega gadgets corresponding to the same symbol α , the protector gadgets and the information gadgets of m_ℓ and m_r are symmetric to each other. It is therefore very beneficial to remove the mega gadgets in a synchronized manner. The event in which the ℓ 'th left mega gadget and the r 'th right mega gadgets are deleted in a synchronized manner corresponds to $S[\ell]$ and $T[r]$ being matched by the longest common subsequence.

Now, consider the case in which m_ℓ and m_r do not match i.e. $S[\ell] \neq T[r]$. Deleting the protectors in a synchronized manner would not yield much benefit in this scenario, since the information gadgets are not symmetric, and therefore would have to be removed slowly. In this case, since an inefficient deletion of an information gadget is inevitable, it is more efficient to delete one of the protector gadgets using a single deletion operation, and proceed to delete the following information gadget inefficiently. This would result in either the left mega gadget being deleted, or the right one. The event in which the ℓ 'th left (resp. r 'th right) mega gadget is deleted in a non-synchronized manner corresponds to $S[\ell]$ (resp. $T[r]$) not being in the longest common subsequence. The gadgets are designed in a way such that deleting mega gadgets in a synchronized way is faster than deleting each mega gadget in a non-synchronized way. Furthermore, the cost reduction of a synchronized deletion over a non-synchronized deletion is a constant number B . Therefore, by selecting $D(\alpha)$ as the cost of deleting a mega gadget corresponding to the symbol α in a non-synchronized way, one obtains Lemma 7.

The above discussion makes an implicit assumption that the sequence of deletion is applied in *phases*. Each phase starts with $x[i..j]$ such that i and j are edge endpoints of mega gadgets and proceeds to either delete both in a synchronized manner or one in a non-synchronized manner. In order to show that $\text{HDD}(x, y)$ is at most the term in Lemma 7, this is sufficient since we can choose a sequence of deletion with this structure as a witness. In order to show that $\text{HDD}(x, y)$ is at least the expression in Lemma 7, one has to show that there is an optimal sequence of deletions with this structure. This is one of the main technical challenges in obtaining Theorem 2.

In a high level, the *sync* gadgets function as “anchors” that force any sequence of deletions to stop in their proximity. Another key property of our construction that enforces the “phases” structure is the large size of a protector relatively to the information. Intuitively, an optimal sequence would always avoid deleting a protector gadget inefficiently, so if a left protector is deleted using a right protector, left deletions would continue to occur until the next left protector is reached.

In Section 4, we provide the formal definition for a well-structured sequence and prove that there is an optimal sequence of deletions with this structure. In Section 5 we provide a precise analysis of every phase in a well-structured sequence. In Section 6 we put everything together and prove Lemma 7.

4 Well-Behaved Paths

We start by defining a well-behaved path.

► **Definition 8** (Well-Behaved Path). *A path p from x to y in G_x is well-behaved if for every $\ell \in [|S| + 1]$ and $r \in [|T| + 1]$, if p visits $x[\text{left}_\ell^P \dots \text{right}_r^P]$, one of the following vertices is also visited by p : $x[\text{left}_{\ell+1}^P \dots \text{right}_r^P]$, $x[\text{left}_\ell^P \dots \text{right}_{r+1}^P]$, or $x[\text{left}_{\ell+1}^P \dots \text{right}_{r+1}^P]$. If one of $\ell + 1$ and $r + 1$ is undefined, the condition is on the subset of defined vertices. If both are undefined, the condition is considered satisfied.*

This section is dedicated to proving the following lemma.

► **Lemma 9** (Optimal Well-Behaved Path). *There is a shortest path from x to y in G_x which is well-behaved.*

We start by proving properties regarding paths and shortest paths from x to y in G_x . Due to space constraints, the proofs for the lemmata in this section appear in the full version of this paper [2].

4.1 Properties of Paths in G_x

We start by observing that an x to y path in G_x never deletes symbols from y .

► **Observation 10** (Never Delete y). *The substring $x[\text{left}_{|S|+1}^P \dots \text{right}_{|T|+1}^P] = y$ is the unique occurrence of y in x . Let p be a path from x to y in G_x . For every vertex $x[i..j]$ in p , $[\text{left}_{|S|+1}^P \dots \text{right}_{|T|+1}^P] \subseteq [i..j]$.*

Since each hairpin deletion operation deletes a prefix (or a suffix) of a substring of x , we have the following observation and immediate corollary.

► **Observation 11**. *Let $x[i..j] \in 010^a 1 * \overline{10^b 10}$ for $a \neq b$. A single left hairpin deletion operation removes at most a single 1 character. Symmetrically, a right hairpin deletion operation removes at most a single $\bar{1}$ character.*

► **Corollary 12**. *A single left (resp. right) hairpin deletion operation on $x[i..j]$ can remove more than a single 1 (resp. $\bar{1}$) character only if i and j are in symmetric gadgets.*

The next lemma assures a restriction over the vertices along a path from x to y in G_x .

► **Lemma 13** (Always $01*$ or $*\bar{10}$). *Let p be a path from s to t in G_x such that $s, t \in 01 * \bar{10}$. For every vertex $x[i..j]$ visited by p , we have $x[i] = 0$ and $x[j] = \bar{0}$. Furthermore, $x[i+1] = 1$ or $x[j-1] = \bar{1}$.*

Due to the equivalence between a path in G_x and a sequence of hairpin deletions and due to the symmetry between hairpin deletion and hairpin completion, we obtain the following.

► **Corollary 14**. *Let $s, t \in 01 * \bar{10}$ and let \mathcal{H} be a sequence of h hairpin deletion operations (or a sequence of hairpin completion operations) that transforms s into t . For $i \in [h]$, let S_i be the string obtained by applying the first i operations of \mathcal{H} on s . For every $i \in [h]$, we have $S_i[1] = 0$ and $S_i[|S_i|] = \bar{0}$. Furthermore, either $S_i[2] = 1$ or $S_i[|S_i| - 1] = \bar{1}$.*

The following lemma discusses the situation in which p visits a vertex not in $01 * \bar{10}$. Essentially, the lemma claims that when p visits a substring $x[i..j]$ with a prefix 00 , the next step would be $x[i+1..j]$, i.e., deleting a single zero from the left.

► **Lemma 15** (Return to $01 * \bar{10}$). *Let p be a path from x to y in G_x . If p visits a vertex $x[i..j]$ such that $x[i..j] = 01 * \overline{10^k}$ for some integer $k \geq 1$, then for every $k' \in [k-1]$ it must be that p visits $x[i..j-k']$ as well. Symmetrically, if p visits a vertex $x[i..j]$ such that $x[i..j] = 0^k 1 * \bar{10}$ for some integer $k \geq 1$, then for every $k' \in [k-1]$ it must be that p visits $x[i+k'..j]$ as well.*

The following is a direct implication of Lemmata 13 and 15.

► **Observation 16**. *Let p be a path from x to y in G_x . If p applies a right hairpin deletion operation on v then $v \in 01*$. Symmetrically, if p applies a left hairpin deletion operation on v then $v \in *\bar{10}$.*

The following lemma establishes the importance of the synchronizer gadgets.

► **Lemma 17** (Synchronizer Lemma). *Let p be a path from x to y in G_x and let $s = x[i_s..j_s] = \text{Sync}_L$ be a left synchronizer which is not contained in y , p must visit a vertex $x[j_s + 1..k]$ for some integer k . Symmetrically, if $x[i_s..j_s] = \text{Sync}_R$ is a right synchronizer which is not contained in y , p must visit a vertex $x[k..i_s - 1]$.*

Notice that the leftmost index of every left information and protector gadget is $j_s + 1$ for some left synchronizer $x[i_s..j_s]$ (excluding the leftmost protector gadget). A similar structure occurs with right gadgets. The following directly follows from Lemma 17.

► **Corollary 18.** *Let p be a path from x to y in G_x . Then, for each $\ell \in [|S|]$ the path p visits vertices $u = x[i..j]$ with $i = \text{left}_\ell^P$ and $v = x[i..j]$ with $i = \text{left}_\ell^I$. Symmetrically, for each $r \in [|T|]$ the path p visits vertices $u = x[i..j]$ with $j = \text{right}_r^P$ and $v = x[i..j]$ with $j = \text{right}_r^I$.*

4.2 Transitions Between Gadgets

In this section, we address the way shortest paths apply to vertices that transit from a gadget to the gadget afterward.

► **Lemma 19.** *Let p be a shortest path from x to y in G_x . For some $\ell \in |S|$, let $v = x[i_1..j_1]$ be the first vertex visited by p with $i_1 = \text{left}_\ell^I$. Let $u = x[i_2..j_2]$ be the first vertex visited by p with $i_2 = \text{left}_{\ell+1}^P$. Then, there is no occurrence of P_R in $x[j_2..j_1]$.*

Symmetrically, for some $r \in |T|$ let $v = x[i_1..j_1]$ be the first vertex visited by p with $j_1 = \text{right}_r^I$. Let $u = x[i_2..j_2]$ be the first vertex visited by p with $j_2 = \text{right}_{r+1}^P$. Then, there is no occurrence of P_L in $x[i_1..i_2]$.

Proof Sketch, Complete Proof in the full version of this paper [2]. The proof is by contradiction. If there is only a *single* P_R gadget that is contained in $x[j_2..j_1]$, by Corollary 12 the number of hairpin deletions that must happen just to remove this gadget is at least \mathfrak{p} . We introduce an alternative, shorter path: At the moment p reaches this P_R gadget, it first removes all the $I_L(S[\ell])$ gadget, and then removes all the remaining characters on the right side greedily, until reaching $x[i_2..j_2]$. The reason why this alternative path is indeed shorter is since in this way the removal of the P_R gadget takes 1 operation, instead of \mathfrak{p} , and we may pay at most $i_\alpha + i_\beta \leq 2i_0$ for some $\alpha, \beta \in \{0, 1, 2\}$, for removing one information gadget in the left side and the information gadget following the right protector gadget. Since \mathfrak{p} is much larger than i_0 , the alternative path is shorter, contradicting the assumption that p is a shortest path. Notice that if there are more than one P_R gadgets in $x[j_2..j_1]$, the benefit from deleting $I_L(S[\ell])$ first is even larger. ◀

The following lemma states that every right deletion on $x[i..j]$ with i being within a non Sync_L gadget can also be applied if i is the leftmost index of the gadget. A symmetric argument is stated as well.

► **Lemma 20.** *Let p be a path from x to y in G_x . Let $v = x[i..j]$ be a vertex visited by p such that $i \in [\text{left}_\ell^P.. \text{left}_\ell^I - 1]$ for some $\ell \in [|S|]$. Let $v' = x[\text{left}_\ell^P..j]$, let $u = x[i..j - k]$ and $u' = x[\text{left}_\ell^P..j - k]$ for some k . If (v, u) is an edge in p , then (v', u') is an edge in G_x .*

The above statement considers the case in which v interacts with a P_L gadget, the following similar statements, regarding different gadgets hold as well:

■ **P_R :** *Let $v = x[i..j]$ be a vertex visited by p such that $j \in [\text{right}_r^I + 1.. \text{right}_r^P]$ for some $r \in [|T|]$. Let $v' = x[i.. \text{right}_r^P]$, let $u = x[i + k..j]$ for some k , and let $u' = x[i + k.. \text{right}_r^P]$. If (v, u) is an edge in p , then (v', u') is an edge in G_x .*

- $I_L(\alpha)$ for some $\alpha \in \{0, 1, 2\}$: Let $v = x[i..j]$ be a vertex visited by p such that $i \in [\text{left}_\ell^I .. \text{left}_{\ell+1}^P - 1]$ for some $\ell \in [|S|]$. Let $v' = x[\text{left}_\ell^I .. j]$, let $u = x[i..j - k]$ and $u' = x[\text{left}_\ell^I .. j - k]$ for some k . If (v, u) is an edge in p , then (v', u') is an edge in G_x .
- $I_R(\alpha)$ for some $\alpha \in \{0, 1, 2\}$: Let $v = x[i..j]$ be a vertex visited by p such that $j \in [\text{right}_{r+1}^P + 1 .. \text{right}_r^I]$ for some $r \in [|T|]$. Let $v' = x[i.. \text{right}_r^I]$, let $u = x[i + k..j]$ for some k , and let $u' = x[i + k.. \text{right}_r^I]$. If (v, u) is an edge in p , then (v', u') is an edge in G_x .

Proof Sketch, Complete Proof in the full version of this paper [2]. We distinguish between two cases. If $x[i..i + 1] \neq 01$, by Lemma 15 the hairpin deletion removes exactly one $\bar{0}$ character, and by $x[\text{left}_\ell^P] = 0$ the edge (u', v') is in G_x . If $x[i..i + 1] = 01$, we first prove (using Corollary 18) that $k \leq \text{left}_\ell^I - i$. Moreover, since $i \in [\text{left}_\ell^P .. \text{left}_\ell^I - 1]$ it must be the case that $i = \text{left}_\ell^P + q \cdot 11$ for some q . Since $x[\text{left}_\ell^P .. \text{left}_\ell^I - 1]$ is periodic with period 11. Thus, $x[\text{left}_\ell^P .. \text{left}_\ell^P + k] = x[\text{left}_\ell^P + q \cdot 11 .. \text{left}_\ell^P + q \cdot 11 + k] = x[i..i + k]$ and the claim follows. ◀

We are now ready to prove Lemma 9.

► **Lemma 9 (Optimal Well-Behaved Path).** *There is a shortest path from x to y in G_x which is well-behaved.*

Proof. We describe a method that converts a shortest path p from x to y that visits $u = x[\text{left}_\ell^P .. \text{right}_r^P]$ into a shortest path p' from x to y that visits one of the following vertices: $x[\text{left}_{\ell+1}^P .. \text{right}_r^P]$, $x[\text{left}_\ell^P .. \text{right}_{r+1}^P]$, or $x[\text{left}_{\ell+1}^P .. \text{right}_{r+1}^P]$. Moreover, the prefixes of p and p' from x to u are identical. Using this technique, it is straightforward to convert a shortest path from x to y in G_x into a well-behaved path of the same length.

Let $v_L = x[i_L..j_L]$ be the first vertex in p with $i_L = \text{left}_{\ell+1}^P$ and let $v_R = x[i_R..j_R]$ be the first vertex in p with $j_R = \text{right}_{r+1}^P$. By Corollary 18, v_L and v_R are well defined (unless $\ell = |S| + 1$ or $r = |T| + 1$, in such a case just one of the vertices is well defined and the claim follows trivially from Observation 10). We consider the case where v_L appears before v_R in p and show how to convert p . The other case is symmetric.

We distinguish between two cases:

Case 1: $j_L \in [\text{right}_r^I + 1 .. \text{right}_r^P]$. Let q be the sub-path of p from u to v_L . We present a path q^* from u to v_L that is not longer than q and visits $x[\text{left}_{\ell+1}^P .. \text{right}_r^P]$. Recall that an edge of the form $x[i..j] \rightarrow x[i + k..j]$ is called a *left* edge, and an edge of the form $x[i..j] \rightarrow x[i..j - k]$ is called a *right* edge. Let cost_L be the number of left edges in q and cost_R be the number of right edges in q . We first show a path from $u = x[\text{left}_\ell^P .. \text{right}_r^P]$ to $x[\text{left}_{\ell+1}^P .. \text{right}_r^P]$ of length cost_L . Let $e = x[i_1..j] \rightarrow x[i_2..j]$ be a left edge in q . It must be that $j \in [j_L .. \text{right}_r^P] \subseteq [\text{right}_r^I + 1 .. \text{right}_r^P]$. Hence, by Lemma 20, there exists an edge $e' = x[i_1.. \text{right}_r^P] \rightarrow x[i_2.. \text{right}_r^P]$. Let $e_1, e_2, \dots, e_{\text{cost}_L}$ be the subsequence of all left edges in q . The path $q_1^* = e'_1, e'_2, \dots, e'_{\text{cost}_L}$ is a valid path of length cost_L from $u = x[\text{left}_\ell^P .. \text{right}_r^P]$ to $x[\text{left}_{\ell+1}^P .. \text{right}_r^P]$.

If $j_L = \text{right}_r^P$ then $q^* = q_1^*$ is a path that satisfies all the requirements. Otherwise, $\text{cost}_R \geq 1$. We claim that there is an edge e_R from $x[\text{left}_{\ell+1}^P .. \text{right}_r^P]$ to $v_L = x[\text{left}_{\ell+1}^P .. j_L]$. This is true since $x[\text{left}_{\ell+1}^P .. \text{left}_{\ell+1}^I - 1] = P_L \cdot \text{Sync}_L = \overleftarrow{\text{Sync}_R[2..|\text{Sync}_R|]} \cdot P_R = x[\text{right}_r^I + 2 .. \text{right}_r^P]$ and $j_L \in [\text{right}_r^I + 1 .. \text{right}_r^P]$. We conclude q^* by appending e_R to the end of q_1^* . Finally, $\text{cost}(q^*) = \text{cost}_L + 1 \leq \text{cost}_L + \text{cost}_R = \text{cost}(q)$, and q^* visits $x[\text{left}_{\ell+1}^P .. \text{right}_r^P]$.

11:10 Hairpin Completion Distance Lower Bound

Case 2: $j_L \in [\mathbf{right}_{r+1}^P - 1 .. \mathbf{right}_r^I]$. We first prove the following claim.

▷ **Claim.** $i_R \in [\mathbf{left}_{\ell+1}^P .. \mathbf{left}_{\ell+1}^I - 1]$.

Proof. Since v_R appears after v_L , we have $i_R \geq i_L = \mathbf{left}_{\ell+1}^P$. Assume to the contrary that $i_R \geq \mathbf{left}_{\ell+1}^I$. Let $v_f = x[i_f .. j_f]$ be the first vertex in p with $j_f = \mathbf{right}_r^I$ (v_f exists according to Corollary 18). Note that v_f does not appear after v_L in p since $j_L = \leq \mathbf{right}_r^I = j_f$. Therefore, $i_f \leq i_L = \mathbf{left}_{\ell+1}^P$ and $[\mathbf{left}_{\ell+1}^P .. \mathbf{left}_{\ell+1}^I] \subseteq [i_f .. i_R]$. Therefore, the occurrence of P_L starting in $\mathbf{left}_{\ell+1}^P$ is contained in $x[i_f .. i_R]$. Since p is a shortest path, this is a contradiction to Lemma 19. ◀

Let q be the sub-path of p from v_L to v_R . Let \mathbf{cost}_L be the number of left edges in q and \mathbf{cost}_R be the number of right edges in q . We present a path q^* from v_L to v_R that is not longer than q and visits $x[\mathbf{left}_{\ell+1}^P .. \mathbf{right}_{r+1}^P]$. We first show a path q_1^* from $v_L = x[\mathbf{left}_{\ell+1}^P .. j_L]$ to $x[\mathbf{left}_{\ell+1}^P .. \mathbf{right}_{r+1}^P]$ of length \mathbf{cost}_R . Let $e = x[i .. j_1] \rightarrow x[i .. j_2]$ be a right edge in q . It must be that $i \in [\mathbf{left}_{\ell+1}^P .. i_R] \subseteq [\mathbf{left}_{\ell+1}^P .. \mathbf{left}_{\ell+1}^I - 1]$ due to the claim. Hence, by Lemma 20, there exists an edge $e' = x[\mathbf{left}_{\ell+1}^P .. j_1] \rightarrow x[\mathbf{left}_{\ell+1}^P .. j_2]$. Let $e_1, e_2, \dots, e_{\mathbf{cost}_L}$ be the subsequence of all right edges in q . The path $q_1^* = e'_1, e'_2, \dots, e'_{\mathbf{cost}_L}$ is a valid path of length \mathbf{cost}_R from $v_L = x[\mathbf{left}_{\ell+1}^P .. j_L]$ to $x[\mathbf{left}_{\ell+1}^P .. \mathbf{right}_{r+1}^P]$.

If $i_R = \mathbf{left}_{\ell+1}^P$ then $q^* = q_1^*$ is a path that satisfies all the requirements. Otherwise, $\mathbf{cost}_L \geq 1$. We claim that there is an edge e_L from $x[\mathbf{left}_{\ell+1}^P .. \mathbf{right}_{r+1}^P]$ to $v_R = x[i_R .. \mathbf{right}_{r+1}^P]$. This is true since $x[\mathbf{left}_{\ell+1}^P .. \mathbf{left}_{\ell+1}^I] = P_L \cdot \mathbf{Sync}_L \cdot 0 = \overleftarrow{\mathbf{Sync}_R \cdot P_R} = x[\mathbf{right}_{r+1}^I + 1 .. \mathbf{right}_{r+1}^P]$ and $i_R \in [\mathbf{left}_{\ell+1}^P .. \mathbf{left}_{\ell+1}^I - 1]$. We conclude q^* by appending e_L to the end of q_1^* . Finally, $\mathbf{cost}(q^*) = \mathbf{cost}_L + 1 \leq \mathbf{cost}_L + \mathbf{cost}_R = \mathbf{cost}(q)$, and q^* visits $x[\mathbf{left}_{\ell+1}^P .. \mathbf{right}_r^P]$. ◀

5 Cost of Well-Behaved Steps

In this section, we analyze the cost of each of the possible phases of a well-behaved path (Definition 8). We first consider the cost of deletion of a single mega-gadget.

► **Lemma 21** (Non Synchronized Deletion). *Let $v = x[\mathbf{left}_\ell^P .. \mathbf{right}_r^P]$, $u_1 = [\mathbf{left}_{\ell+1}^P .. \mathbf{right}_r^P]$ and $u_2 = [\mathbf{left}_\ell^P .. \mathbf{right}_{r+1}^P]$ for $\ell \in [|S|]$ and $r \in [|T|]$. Let $S[\ell] = \alpha$ and $T[r] = \beta$. It holds that $\mathbf{dist}_{G_x}(v, u_1) = i_\alpha + 2$ and $\mathbf{dist}_{G_x}(v, u_2) = i_\beta + 2$.*

Proof. We prove $\mathbf{dist}_{G_x}(v, u_1) = i_\alpha + 2$. The proof for $\mathbf{dist}_{G_x}(v, u_2) = i_\beta + 2$ is symmetrical. We prove the lemma by showing $\mathbf{dist}_{G_x}(v, u_1) \geq i_\alpha + 2$ and $\mathbf{dist}_{G_x}(v, u_1) \leq i_\alpha + 2$.

$\mathbf{dist}_{G_x}(v, u_1) \geq i_\alpha + 2$. Let p be a v to u_1 path in G_x . Note that vertex $x[i .. j]$ in p has $j = \mathbf{right}_r^P$. According to Corollary 18, p must visit $z = x[\mathbf{left}_\ell^I .. \mathbf{right}_r^P]$. Since $z \neq v$, the sub-path of p from v to z induces a cost of at least 1 to p . Consider the sub-path q of p from z to u_1 . According to Corollary 12, every left hairpin deletion step in q deletes at most a single '1' symbol. Due to $x[\mathbf{left}_\ell^I .. \mathbf{left}_{\ell+1}^P - 1] = I_L(\alpha) \cdot \mathbf{Sync}_L$, the sub-path q consists of at least $\#_1(I_L(\alpha)) + 1 = i_\alpha + 1$ additional left hairpin deletions.

$\mathbf{dist}_{G_x}(v, u_1) \leq i_\alpha + 2$. We present a path p with cost exactly $i_\alpha + 2$ from v to u_1 . Initially, p deletes a prefix of length $|P_L| + |\mathbf{Sync}_L|$ from v in one step. This is possible since v has a suffix $\overline{10} \cdot P_R$. Then, p proceeds to delete $x[\mathbf{left}_\ell^I .. \mathbf{left}_{\ell+1}^P - 1] = I_L(\alpha) \cdot \mathbf{Sync}_L$ a single '1' character at a time. Note that this is possible regardless of the value of α due to $x[\mathbf{right}_r^P - 8 .. \mathbf{right}_r^P] = \overline{0^710}$. The total cost of this path is $i_\alpha + 2$ as required. ◀

In the following lemma, we show that the cost of deleting two disagreeing mega-gadgets is the same as deleting each one of them separately.

► **Lemma 22** (Synchronized Deletion of Disagreeing Mega Gadgets). *Let $v = x[\text{left}_\ell^P \dots \text{right}_r^P]$, $u = [\text{left}_{\ell+1}^P \dots \text{right}_{r+1}^P]$ for $\ell \in [|S|]$ and $r \in [|T|]$ with $S[\ell] \neq T[r]$. It holds that $\text{dist}_{G_x}(v, u) = i_\alpha + i_\beta + 4$ with $S[\ell] = \alpha$ and $T[r] = \beta$.*

Proof. We prove the claim by showing $\text{dist}_{G_x}(v, u) \geq i_\alpha + i_\beta + 4$ and $\text{dist}_{G_x}(v, u) \leq i_\alpha + i_\beta + 4$.

$\text{dist}_{G_x}(v, u) \geq i_\alpha + i_\beta + 4$. Let p be a path from v to u in G_x . According to Corollary 18, p visits vertices $z_1 = x[\text{left}_\ell^I \dots j]$ and $z_2 = x[i \dots \text{right}_r^I]$ for some i, j . The last left hairpin deletion in p before z_1 and the last right hairpin deletion in p before z_2 induce a cost of 2 to p . Consider a left hairpin deletion that is applied to a vertex $x[i' \dots j']$ after z_1 in p . Note that i' is either within an $I_L(\alpha)$ gadget or within a Sync_L gadget, and j' is either within an $I_R(\beta)$ gadget, a Sync_R gadget or a P_R gadget. In any of the above cases, Corollary 12 suggests that the deletion operation deletes at most a single '1' character. Therefore, there are at least $i_\alpha + 1$ left deletions after z_1 in p . Due to similar reasoning, there are at least $i_\beta + 1$ right hairpin deletions after z_2 in p . It follows that the total cost of p is at least $i_\alpha + i_\beta + 4$.

$\text{dist}_{G_x}(v, u) \leq i_\alpha + i_\beta + 4$. Consider the path p that is composed of two sub-paths, the prefix p_1 is a shortest path from v to $w = x[\text{left}_{\ell+1}^P \dots \text{right}_r^P]$ and the suffix p_2 is a shortest path from w to u . By Lemma 21 we have $\text{cost}(p_1) = i_\alpha + 2$ and $\text{cost}(p_2) = i_\beta + 2$. Therefore $\text{cost}(p) = \text{cost}(p_1) + \text{cost}(p_2) = i_\alpha + 2 + i_\beta + 2$. ◀

The last case we have to analyze is a synchronized deletion of agreeing mega gadgets. We first present the concept of *Fibonacci-regular numbers*.

► **Definition 23** (Fibonacci-regular number). *We say that $a \in \mathbb{N}$ is a Fibonacci-regular number if for all $2 \leq k \leq a$ it holds that $\text{Fib}^{-1}(a) \leq \text{Fib}^{-1}(a/k) + k - 1$.*

► **Fact 24.** $i_2 = 53, i_1 = 54, i_0 = 55$ and $p = 144$ are Fibonacci-regular numbers.

The following lemma, which proof is in the full version of this paper, provides the required machinery to analyze the cost of a synchronized deletion [2].

► **Lemma 25.** *Let $\text{per} = 010^{\text{ext}}$ and let $q \in 010^{\text{int}}01 * 11\bar{1}\bar{1} * \overline{100^{\text{int}}10}$ with $\text{int} \neq \text{ext}$ and $\min\{\text{int}, \text{ext}\} \geq 3$. For every Fibonacci-regular number a , we have $\text{HDD}(\text{per}^a \cdot \text{Sync}_L \cdot q \cdot \text{Sync}_R \cdot \overleftarrow{\text{per}}^a, q) = \text{Fib}^{-1}(a) + \max(\text{ext} - \text{int} - 1, 0) + 3$.*

Finally, we are ready to analyze the cost of synchronized deletion of agreeing mega gadgets.

► **Lemma 26** (Synchronized Deletion of Agreeing Mega Gadgets). *Let $v = x[\text{left}_\ell^P \dots \text{right}_r^P]$, $u = [\text{left}_{\ell+1}^P \dots \text{right}_{r+1}^P]$ for $\ell \in [|S|]$ and $r \in [|T|]$ with $S[\ell] = T[r] = \alpha$. Then $\text{dist}_{G_x}(v, u) = \text{Fib}^{-1}(p) + \text{Fib}^{-1}(i_\alpha) + 11 - 2\alpha$.*

Proof. Let $w = x[\text{left}_\ell^I \dots \text{right}_r^I]$. Consider the following path p' from u to v in G_x . The path p' consists of a prefix p'_1 which is a shortest path from u to w and a suffix p'_2 which is a shortest path from w to v . Since i_0, i_1, i_2 and p are Fibonacci-regular numbers (Fact 24), according to Lemma 25 (with $\text{ext} = 9$ and $\text{int} = 3 + 2\alpha$) we have $\text{cost}(p'_1) = \text{Fib}^{-1}(p) + \max(9 - (3 + 2\alpha) - 1, 0) + 3 = \text{Fib}^{-1}(p) + 8 - 2\alpha$. Similarly, according to Lemma 25 (with $\text{ext} = 3 + 2\alpha$

11:12 Hairpin Completion Distance Lower Bound

and $\text{int} = 9$) we have $\text{cost}(p'_2) = \text{Fib}^{-1}(i_\alpha) + \max(3 + 2\alpha - 9 - 1, 0) + 3 = \text{Fib}^{-1}(i_\alpha) + 3$. In total we have $\text{cost}(p') = \text{cost}(p'_1) + \text{cost}(p'_2) = \text{Fib}^{-1}(p) + \text{Fib}^{-1}(i_\alpha) + 11 - 2\alpha$. Therefore $\text{dist}_{G_x}(v, u) \leq \text{Fib}^{-1}(p) + \text{Fib}^{-1}(i_\alpha) + 11 - 2\alpha = 31 - 2\alpha$.

We prove the following claim.

▷ **Claim.** There is a shortest path p from v to u that visits w .

Proof. Let $v_L = x[i_L..j_L]$ and $v_R = x[i_R..j_R]$ be the first vertices visited by p with $i_L = \text{left}_\ell^I$ and $j_R = \text{right}_r^I$. Assume without loss of generality that v_R occurs before v_L in p . We consider two cases regarding j_L .

Case 1: $j_L = \text{right}_{r+1}^P$. Consider the suffix p_s of p from v_L to u . Let $v' = x[i'..j']$ be a vertex in p_s that is immediately followed by a left hairpin deletion operation in p_s . Since $i' \in [\text{left}_\ell^I.. \text{left}_{\ell+1}^P - 1]$ is either within an $I_L(\alpha)$ gadget or within a Sync_L gadget, and $j' = \text{right}_{r+1}^P$ is in a P_R gadget, Corollary 12 suggests that the left hairpin deletion applied to v' deletes at most a single '1' character. It follows from the above analysis that the number of left hairpin deletions in p_s is at least $\#_1(I_L(\alpha)) + 1 \geq i_2 + 1 = 54$. Therefore, the cost of p is at least $55 > 31 \geq \text{cost}(p')$, which contradicts the minimality of p .

Case 2: $j_L > \text{right}_{r+1}^P$. Let q be the sub-path of p from v_R to v_L . Let cost_L be the number of left edges in q and cost_R be the number of right edges in q . We first show a path from v_R to $x[\text{left}_\ell^I.. \text{right}_r^I]$ of length cost_L . Let $e = x[i_1..j] \rightarrow x[i_2..j]$ be a left edge in q . It must be that $j \in [j_L.. \text{right}_r^I] \subseteq [\text{right}_{r+1}^P + 1.. \text{right}_r^I]$. Hence, by Lemma 20, there exists an edge $e' = x[i_1.. \text{right}_r^I] \rightarrow x[i_2.. \text{right}_r^I]$. Let $e_1, e_2, \dots, e_{\text{cost}_L}$ be the subsequence of all left edges in q . The path $q_1^* = e'_1, e'_2, \dots, e'_{\text{cost}_L}$ is a valid path of length cost_L from v_R to $x[\text{left}_\ell^I.. \text{right}_r^I]$.

If $j_L = \text{right}_r^I$ then $q^* = q_1^*$ is a path that satisfies all the requirements. Otherwise, $\text{cost}_R \geq 1$. We claim that there is an edge e_R from $x[\text{left}_\ell^I.. \text{right}_r^I]$ to $v_L = x[\text{left}_\ell^I.. j_L]$. This is true since $x[\text{left}_\ell^I.. \text{left}_{\ell+1}^P - 1] = I_L(S[\ell]) \cdot \text{Sync}_L = I_L(T[r]) \cdot \text{Sync}_L = \overleftarrow{\text{Sync}_R[2..|\text{Sync}_R|]} \cdot I_R(T[r]) = x[\text{right}_{r+1}^P + 2.. \text{right}_r^I]$ and $j_L \in [\text{right}_{r+1}^P + 1.. \text{right}_r^I]$. We conclude q^* by appending e_R to the end of q_1^* . Finally, $\text{cost}(q^*) = \text{cost}_L + 1 \leq \text{cost}_L + \text{cost}_R = \text{cost}(q)$, and q^* visits $x[\text{left}_\ell^I.. \text{right}_r^I]$. ◁

Let p be a shortest path from u to v in G_x . According to the claim, we can indeed assume that p consists of a shortest path p_1 from v to w and a shortest path p_2 from w to v . Therefore we have $\text{cost}(p) = \text{cost}(p') = \text{Fib}^{-1}(p) + \text{Fib}^{-1}(i_\alpha) + 11 - 2\alpha$ as required. ◀

6 Correctness

Let $D(0) = 57$, $D(1) = 56$, $D(2) = 55$, $D_{\text{sync}}(0) = 31$, $D_{\text{sync}}(1) = 29$, $D_{\text{sync}}(2) = 27$, and $B = 83$. The following lemma summarize Lemmata 21, 22, and 26.

► **Lemma 27.** Let $\ell \in [|S|]$ and let $r \in [|T|]$ be two integers. Denote $S[\ell] = \alpha$ and $T[r] = \beta$. The following is satisfied.

1. $\text{dist}_{G_x}(x[\text{left}_\ell^P.. \text{right}_r^P], x[\text{left}_{\ell+1}^P, \text{right}_r^P]) = D(\alpha)$
2. $\text{dist}_{G_x}(x[\text{left}_\ell^P.. \text{right}_r^P], x[\text{left}_\ell^P, \text{right}_{r+1}^P]) = D(\beta)$
3. $\text{dist}_{G_x}(x[\text{left}_\ell^P.. \text{right}_r^P], x[\text{left}_{\ell+1}^P, \text{right}_{r+1}^P]) = D(\alpha) + D(\beta)$ if $\alpha \neq \beta$
4. $\text{dist}_{G_x}(x[\text{left}_\ell^P.. \text{right}_r^P], x[\text{left}_{\ell+1}^P, \text{right}_{r+1}^P]) = D_{\text{sync}}(\alpha)$ if $\alpha = \beta$
5. $2D(0) - D_{\text{sync}}(0) = 2D(1) - D_{\text{sync}}(1) = 2D(2) - D_{\text{sync}}(2) = B$

Proof. According to Lemma 21, we have $D(\gamma) = i_\gamma + 2$ for every $\gamma \in \{0, 1, 2\}$. It follows from Lemma 22 that if $\alpha \neq \beta$ we have $\text{dist}_{G_x}(x[\text{left}_\ell^P \dots \text{right}_r^P], x[\text{left}_{\ell+1}^P, \text{right}_{r+1}^P]) = i_\alpha + i_\beta + 4 = i_\alpha + 2 + i_\beta + 2 = D(\alpha) + D(\beta)$. It follows from Lemma 26 that $D_{\text{sync}}(\gamma) = \text{Fib}^{-1}(p) + \text{Fib}^{-1}(i_\alpha) + 11 - 2\gamma = 11 + 9 + 11 - 2\gamma = 31 - 2\gamma$ for every $\gamma \in \{0, 1, 2\}$.

Indeed, we have $2 \cdot D(0) - D_{\text{sync}}(0) = 2 \cdot 57 - 31 = 83$, $2 \cdot D(1) - D_{\text{sync}}(1) = 56 \cdot 2 - 29 = 83$ and $2 \cdot D(2) - D_{\text{sync}}(2) = 55 \cdot 2 - 27 = 83$ as required. \blacktriangleleft

We are now ready to prove Lemma 7 which concludes the correctness of the reduction.

► **Lemma 7 (Reduction Correctness).** *For some constants $D(0), D(1), D(2)$ and B we have: $\text{HDD}(x, y) = \sum_{\alpha \in \{0,1,2\}} D(\alpha)(\#_\alpha(S) + \#_\alpha(T)) - \text{LCS}(S, T) \cdot B$.*

Proof. We prove the equality claimed, by showing two sides of inequality.

$\text{HDD}(x, y) \leq \sum_{\alpha \in \{0,1,2\}} D(\alpha)(\#_\alpha(S) + \#_\alpha(T)) - \text{LCS}(S, T) \cdot B$. Denote $c = \text{LCS}(S, T)$. Let $\mathcal{I} = i_1 < i_2 < i_3 \dots, \dots, \dots < i_c \subseteq [|S|]$ and $\mathcal{J} = j_1 < j_2 < j_3 \dots < j_c \subseteq [|T|]$ be two sequences of indices such that $S[i_k] = T[j_k]$ for every $k \in [c]$. Thus, \mathcal{I} and \mathcal{J} represent a maximal common subsequence of S and T .

We present a path p in G_x from x to y . The path p starts in $x = x[\text{left}_1^P \dots \text{right}_1^P]$, and consists of 3 types of subpaths.

1. Left deletion subpath: a shortest path from $x[\text{left}_\ell^P \dots \text{right}_r^P]$ to $x[\text{left}_{\ell+1}^P \dots \text{right}_r^P]$ for some $\ell \in [|S|]$ and $r \in [|T| + 1]$.
2. Right deletion subpath: a shortest path from $x[\text{left}_\ell^P \dots \text{right}_r^P]$ to $x[\text{left}_\ell^P \dots \text{right}_{r+1}^P]$ for some $\ell \in [|S| + 1]$ and $r \in [|T|]$.
3. Match subpath: a shortest path from $x[\text{left}_\ell^P \dots \text{right}_r^P]$ to $x[\text{left}_{\ell+1}^P \dots \text{right}_{r+1}^P]$ for some $\ell \in [|S|]$ and $r \in [|T|]$.

Specifically, if p visits $x[\text{left}_\ell^P \dots \text{right}_r^P]$, then p proceeds in a left deletion subpath if $\ell \in [|S|] \setminus \mathcal{I}$. Otherwise, p proceeds in a right deletion subpath if $r \in [|T|] \setminus \mathcal{J}$. If both $\ell \in \mathcal{I}$ and $r \in \mathcal{J}$, the path p proceeds in a match subpath. Note that it is guaranteed that as long as $\ell \neq |S| + 1$ or $r \neq |T| + 1$, p continues to make progress until finally reaching $x[\text{left}_{|S|+1}^P \dots \text{right}_{|T|+1}^P] = y$.

We proceed to analyze the cost of p . For $\alpha \in \{0, 1, 2\}$ we introduce the following notation regarding \mathcal{I} and \mathcal{J} . Let $u_L(\alpha) = |\{i \mid S[i] = \alpha \text{ and } i \notin \mathcal{I}\}|$, $u_R(\alpha) = |\{j \mid T[j] = \alpha \text{ and } j \notin \mathcal{J}\}|$. In addition, let $c(\alpha) = |\{k \mid k \in [c] \text{ and } S[i_k] = \alpha\}|$.

Clearly, by Lemma 27 every $k \in [c]$ induces a cost of $D_{\text{sync}}(S[i_k])$ to p . Moreover, every $i \in [|S|] \setminus \mathcal{I}$, induces a cost of $D(S[i])$ to p , and every $j \in [|T|] \setminus \mathcal{J}$ induces a cost of $D(T[j])$ to p . Thus, we have

$$\begin{aligned} \text{cost}(p) &= \sum_{\alpha \in \{0,1,2\}} D(\alpha)(u_L(\alpha) + u_R(\alpha)) + \sum_{\alpha \in \{0,1,2\}} D_{\text{sync}}(\alpha) \cdot c(\alpha) \\ &= \sum_{\alpha \in \{0,1,2\}} D(\alpha)(u_L(\alpha) + u_R(\alpha)) + \sum_{\alpha \in \{0,1,2\}} (2D(\alpha) - B) \cdot c(\alpha) \\ &= \sum_{\alpha \in \{0,1,2\}} D(\alpha)(u_L(\alpha) + u_R(\alpha) + 2c(\alpha)) - B \cdot \sum_{\alpha \in \{0,1,2\}} c(\alpha) \\ &= \sum_{\alpha \in \{0,1,2\}} D(\alpha)(\#_\alpha(S) + \#_\alpha(T)) - c \cdot B. \end{aligned}$$

Where the first equality follows from $B = 2 \cdot D(\alpha) - D_{\text{sync}}$ for every $\alpha \in \{0, 1, 2\}$, and the last equality is since for every $\alpha \in \{0, 1, 2\}$ we have $\#_\alpha(S) = u_L(\alpha) + c_\alpha, \#_\alpha(T) = u_R(\alpha) + c_\alpha$ and $c = c(0) + c(1) + c(2)$.

11:14 Hairpin Completion Distance Lower Bound

$\text{HDD}(x, y) \geq \sum_{\alpha \in \{0,1,2\}} D(\alpha)(\#_{\alpha}(S) + \#_{\alpha}(T)) - \text{LCS}(S, T) \cdot B$. Let p be a well-behaved shortest path from x to y in G_x . According to Lemma 9, such a path p exists.

Let $\mathcal{X} = \{v = x[\text{left}_{\ell}^P \dots \text{right}_r^P] \mid p \text{ visits } v\}$. Notice that the vertices of \mathcal{X} are naturally ordered by the order of their occurrences in p , so we denote the i th vertex in \mathcal{X} by x_i . For $i \in [|\mathcal{X}| - 1]$, we classify the vertex $x_i = x[\text{left}_{\ell}^P \dots \text{right}_r^P]$ for some $\ell \in [|S| + 1]$ and $r \in [|T| + 1]$ into one of the following four disjoint types.

1. **Match vertex** : if $x_{i+1} = [\text{left}_{\ell+1}^P \dots \text{right}_{r+1}^P]$ and $S[\ell] = T[r]$.
2. **Mismatch vertex** : if $x_{i+1} = [\text{left}_{\ell+1}^P \dots \text{right}_{r+1}^P]$ and $S[\ell] \neq T[r]$.
3. **Left deletion vertex** : if $x_{i+1} = [\text{left}_{\ell+1}^P \dots \text{right}_r^P]$.
4. **Right deletion vertex** : if $x_{i+1} = [\text{left}_{\ell}^P \dots \text{right}_{r+1}^P]$.

Notice that since p is well-behaved, x_i is classified into one of the four types.

We proceed to analyze the cost of the subpath of p from $x_i = x[\text{left}_{\ell}^P \dots \text{right}_r^P]$ to x_{i+1} using Lemma 27. If x_i is a match vertex, it induces a cost of $D_{\text{sync}}(S[\ell])$. If x_i is a mismatch vertex, it induces a cost of $D(S[\ell]) + D(T[r])$. If x_i is a right (resp. left) deletion vertex, it induces a cost of $D(S[\ell])$ (resp. $D(T[r])$). For $\alpha, \beta \in \{0, 1, 2\}$ we present the following notations.

- $c_{\text{match}}(\alpha) = |\{x \in \mathcal{X} \mid x \text{ is a match vertex with } S[\ell] = \alpha\}|$
- $c_{\text{mis}}(\{\alpha, \beta\}) = |\{x \in \mathcal{X} \mid x \text{ is a mismatch vertex with } \{S[\ell], T[r]\} = \{\alpha, \beta\}\}|$
- $c_{\text{mis}}(\alpha) = |\{x \in \mathcal{X} \mid x \text{ is a mismatch vertex with } S[\ell] = \alpha \text{ or } T[r] = \alpha\}|$
- $c_{\text{left}}(\alpha) = |\{x \in \mathcal{X} \mid x \text{ is a left deletion vertex with } S[\ell] = \alpha\}|$
- $c_{\text{right}}(\alpha) = |\{x \in \mathcal{X} \mid x \text{ is a right deletion vertex with } T[r] = \alpha\}|$

Note that since every super-gadget is deleted exactly once as a part of an x_i to x_{i+1} subpath. It follows that for every $\alpha \in \{0, 1, 2\}$ we have $\#_{\alpha}(S) + \#_{\alpha}(T) = c_{\text{left}}(\alpha) + c_{\text{right}}(\alpha) + c_{\text{mis}}(\alpha) + 2c_{\text{match}}(\alpha)$. Note that for $\alpha \in \{0, 1, 2\}$ we have $c_{\text{mis}}(\alpha) = \sum_{\beta \neq \alpha} c_{\text{mis}}(\{\alpha, \beta\})$. We denote $c_{\text{match}} = c_{\text{match}}(0) + c_{\text{match}}(1) + c_{\text{match}}(2)$. We make the following claim:

▷ **Claim.** $c_{\text{match}} \leq \text{LCS}(S, T)$.

Proof. We show that there is a common subsequence of S and T with length c_{match} . Let $\text{Pairs} = \{(\ell, r) \mid x[\text{left}_{\ell}^P \dots \text{right}_r^P] \text{ is a match vertex}\}$. Note that Pairs is naturally ordered by the order of occurrences of the corresponding vertices in p . We denote by (ℓ_i, r_i) the i th pair in Pairs according to this order. Note that for every $i \in [|\text{Pairs}| - 1]$, we have $\ell_i < \ell_{i+1}$ and $r_i < r_{i+1}$ due to the definition of a match vertex. Furthermore, we have $S[\ell_i] = T[r_i]$ for every $i \in [|\text{Pairs}|]$. It follows that the subsequence $S[\ell_1], S[\ell_2], \dots, S[\ell_{|\text{Pairs}|}]$ equals to the subsequence $T[r_1], T[r_2], \dots, T[r_{|\text{Pairs}|}]$. Therefore, S and T have a common subsequence of length $|\text{Pairs}| = c_{\text{match}}$. ◁

It follows from the above analysis that

$$\begin{aligned}
\text{cost}(p) &= \sum_{\alpha \in \{0,1,2\}} D_{\text{sync}}(\alpha) c_{\text{match}}(\alpha) + \sum_{\alpha \neq \beta \in \{0,1,2\}} (D(\alpha) + D(\beta)) \cdot c_{\text{mis}}(\{\alpha, \beta\}) \\
&\quad + \sum_{\alpha \in \{0,1,2\}} D(\alpha) (c_{\text{left}}(\alpha) + c_{\text{right}}(\alpha)) \\
&= \sum_{\alpha \in \{0,1,2\}} D_{\text{sync}}(\alpha) c_{\text{match}}(\alpha) + \sum_{\alpha \in \{0,1,2\}} D(\alpha) \sum_{\beta \neq \alpha} c_{\text{mis}}(\{\alpha, \beta\}) \\
&\quad + \sum_{\alpha \in \{0,1,2\}} D(\alpha) (c_{\text{left}}(\alpha) + c_{\text{right}}(\alpha)) \\
&= \sum_{\alpha \in \{0,1,2\}} D_{\text{sync}}(\alpha) c_{\text{match}}(\alpha) + \sum_{\alpha \in \{0,1,2\}} D(\alpha) \cdot c_{\text{mis}}(\alpha) \\
&\quad + \sum_{\alpha \in \{0,1,2\}} D(\alpha) (c_{\text{left}}(\alpha) + c_{\text{right}}(\alpha)) \\
&= \sum_{\alpha \in \{0,1,2\}} (2D(\alpha) - B) c_{\text{match}}(\alpha) + \sum_{\alpha \in \{0,1,2\}} D(\alpha) (c_{\text{left}}(\alpha) + c_{\text{right}}(\alpha) + c_{\text{mis}}(\alpha)) \\
&= \sum_{\alpha \in \{0,1,2\}} D(\alpha) (c_{\text{left}}(\alpha) + c_{\text{right}}(\alpha) + c_{\text{mis}}(\alpha) + 2c_{\text{match}}(\alpha)) - B \sum_{\alpha \in \{0,1,2\}} c_{\text{match}}(\alpha) \\
&= \sum_{\alpha \in \{0,1,2\}} D(\alpha) \cdot (\#_{\alpha}(S) + \#_{\alpha}(T)) - B \cdot c_{\text{match}} \\
&\geq \sum_{\alpha \in \{0,1,2\}} D(\alpha) \cdot (\#_{\alpha}(S) + \#_{\alpha}(T)) - B \cdot \text{LCS}(S, T).
\end{aligned}$$

Where the last inequality follows from the claim. ◀

References

- 1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 59–78. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.14.
- 2 Itai Boneh, Dvir Fried, Shay Golan, and Matan Kraus. Hairpin completion distance lower bound, 2024. arXiv:2404.11673.
- 3 Itai Boneh, Dvir Fried, Adrian Miclaus, and Alexandru Popa. Faster algorithms for computing the hairpin completion distance and minimum ancestor. In Laurent Bulteau and Zsuzsanna Lipták, editors, *34th Annual Symposium on Combinatorial Pattern Matching, CPM 2023, June 26-28, 2023, Marne-la-Vallée, France*, volume 259 of *LIPICs*, pages 5:1–5:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICs.CPM.2023.5.
- 4 Karl Bringman and Marvin Künnemann. Multivariate fine-grained complexity of longest common subsequence. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1216–1235. SIAM, 2018.
- 5 Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In Jianer Chen and Fedor V. Fomin, editors, *Parameterized and Exact Computation*, pages 75–85, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- 6 Daniela Cheptea, Carlos Martin-Vide, and Victor Mitrană. A new operation on words suggested by DNA biochemistry: Hairpin completion. *Transgressive Computing*, January 2006.
- 7 Lila Kari, Stavros Konstantinidis, Elena Losseva, Petr Sosík, and Gabriel Thierrin. Hairpin structures in DNA words. In Alessandra Carbone and Niles A. Pierce, editors, *DNA Computing*, pages 158–170, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

11:16 Hairpin Completion Distance Lower Bound

- 8 Lila Kari, Stavros Konstantinidis, Petr Sosík, and Gabriel Thierrin. On hairpin-free words and languages. In Clelia De Felice and Antonio Restivo, editors, *Developments in Language Theory*, pages 296–307, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- 9 Lila Kari, Elena Losseva, Stavros Konstantinidis, Petr Sosík, and Gabriel Thierrin. A formal language analysis of DNA hairpin structures. *Fundamenta Informaticae*, 71(4):453–475, 2006.
- 10 Lila Kari, Kalpana Mahalingam, and Gabriel Thierrin. The syntactic monoid of hairpin-free languages. *Acta Informatica*, 44(3):153–166, June 2007.
- 11 Florin Manea. A series of algorithmic results related to the iterated hairpin completion. *Theoretical Computer Science*, 411(48):4162–4178, 2010.
- 12 Florin Manea, Carlos Martín-Vide, and Victor Mitrana. On some algorithmic problems regarding the hairpin completion. *Discret. Appl. Math.*, 157(9):2143–2152, 2009. doi:10.1016/J.DAM.2007.09.022.
- 13 Florin Manea, Carlos Martín-Vide, and Victor Mitrana. Hairpin lengthening. In Fernando Ferreira, Benedikt Löwe, Elvira Mayordomo, and Luís Mendes Gomes, editors, *Programs, Proofs, Processes*, pages 296–306, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
- 14 Florin Manea, Carlos Martín-Vide, and Victor Mitrana. On some algorithmic problems regarding the hairpin completion. *Discrete Applied Mathematics*, 157(9):2143–2152, 2009. Optimal Discrete Structures and Algorithms.