On the Communication Complexity of Approximate Pattern Matching

Tomasz Kociumaka
Max Planck Institute for Informatics, SIC,
Saarbrücken, Germany

Jakob Nogler
ETH Zürich,
Zurich, Switzerland

Philip Wellnitz
Max Planck Institute for Informatics, SIC,
Saarbrücken, Germany

Abstract

The decades-old *Pattern Matching with Edits* problem, given a length-n string T (the text), a length-m string P (the pattern), and a positive integer k (the threshold), asks to list all fragments of T that are at edit distance at most k from P. The one-way communication complexity of this problem is the minimum amount of space needed to encode the answer so that it can be retrieved without accessing the input strings P and T.

The closely related Pattern Matching with Mismatches problem (defined in terms of the Hamming distance instead of the edit distance) is already well understood from the communication complexity perspective: Clifford, Kociumaka, and Porat [SODA 2019] proved that $\Omega(n/m \cdot k \log(m/k))$ bits are necessary and $O(n/m \cdot k \log(m|\Sigma|/k))$ bits are sufficient; the upper bound allows encoding not only the occurrences of P in T with at most k mismatches but also the substitutions needed to make each k-mismatch occurrence exact.

Despite recent improvements in the running time [Charalampopoulos, Kociumaka, and Wellnitz; FOCS 2020 and 2022], the communication complexity of Pattern Matching with Edits remained unexplored, with a lower bound of $\Omega(n/m \cdot k \log(m/k))$ bits and an upper bound of $O(n/m \cdot k^3 \log m)$ bits stemming from previous research. In this work, we prove an upper bound of $O(n/m \cdot k \log^2 m)$ bits, thus establishing the optimal communication complexity up to logarithmic factors. We also show that $O(n/m \cdot k \log m \log(m|\Sigma|))$ bits allow encoding, for each k-error occurrence of P in T, the shortest sequence of edits needed to make the occurrence exact. Our result further emphasizes the close relationship between Pattern Matching with Mismatches and Pattern Matching with Edits.

We leverage the techniques behind our new result on the communication complexity to obtain quantum algorithms for Pattern Matching with Edits: we demonstrate a quantum algorithm that uses $O(n^{1+o(1)}/m \cdot \sqrt{km})$ queries and $O(n^{1+o(1)}/m \cdot (\sqrt{k}m + k^{3.5}))$ quantum time. Moreover, when determining the existence of at least one occurrence, the algorithm uses $O(\sqrt{n^{1+o(1)}/m} \cdot \sqrt{km})$ queries and $O(\sqrt{n^{1+o(1)}/m} \cdot (\sqrt{k}m + k^{3.5}))$ time. For both cases, we establish corresponding lower bounds to demonstrate that the query complexity is optimal up to sub-polynomial factors.

Acknowledgements The work of Jakob Nogler has been carried out mostly during a summer internship at the Max Planck Institute for Informatics.

Contents

1 Introduction

While a *string* is perhaps the most basic way to represent data, this fact makes *algorithms* working on strings more applicable and powerful. Arguably, the very first thing to do with any kind of data is to find *patterns* in it. The *Pattern Matching* problem for strings and its variations are thus perhaps among the most fundamental problems that Theoretical Computer Science has to offer.

In this paper, we study the practically relevant *Pattern Matching with Edits* variation [Sel80]. Given a text string T of length n, a pattern string P of length m, and a threshold k, the aim is to calculate the set $\mathrm{Occ}^E_k(P,T)$ consisting of (the starting positions of) all the fragments of T that are at most k edits away from the pattern P. In other words, we compute the set of k-error occurrences of P in T, more formally defined as

$$\mathrm{Occ}^E_k(P,T) \coloneqq \{i \in [\,0\,..\,n\,] : \exists_{j\in[\,i\,..\,n\,]} \delta_E(P, T[\,i\,..\,j\,) \le k)\},$$

where we utilize the classical edit distance δ_E (also referred to as the Levenshtein distance) [Lev65] as the distance measure. Here, an edit is either an insertion, a deletion, or a substitution of a single character.

Pattern Matching with Edits
Input: a pattern P of length m, a text T of length n, and an integer threshold $k > 0$.
Output: the set $\mathrm{Occ}^E_k(P,T)$.

Even though the Pattern Matching with Edits problem is almost as classical as it can get, with key algorithmic advances (from $O(mn)$ time down to $O(kn)$ time) dating back to the early and late 1980s [Sel80, LV88, LV89], major progress has been made very recently, when Charalampopoulos, Kociumaka, and Wellnitz [CKW22] obtained an $\tilde{O}(n + k^{3.5}n/m)$-time[1] solution and thereby broke through the 20-years-old barrier of the $O(n + k^4 n/m)$-time algorithm by Cole and Hariharan [CH02]. And the journey is far from over yet: the celebrated Orthogonal-Vectors-based lower bound for edit distance [BI18] rules out only $O(n + k^{2-\Omega(1)}n/m)$-time algorithms (also consult [CKW22] for details), leaving open a wide area of uncharted algorithmic territory. In this paper, we provide tools and structural insights that—we believe—will aid the exploration of the said territory.

We add to the picture a powerful new finding that sheds new light on the solution structure of the Pattern Matching with Edits problem—similar structural results [BKW19, CKW20] form the backbone of the aforementioned breakthrough [CKW22]. Specifically, we investigate how much space is needed to store all k-error occurrences of P in T. We know from [CKW20] that $O(n/m \cdot k^3 \log m)$ bits suffice since one may report the occurrences as $O(k^3)$ arithmetic progressions if $n = O(m)$. However, such complexity is likely incompatible with algorithms running faster than $\tilde{O}(n + k^3 n/m)$. In this paper, we show that, indeed, $O(n/m \cdot k \log^2 m)$ bits suffice to represent the set $\mathrm{Occ}^E_k(P,T)$.

Formally, the communication complexity of Pattern Matching with Edits measures the space needed to encode the output so that it can be retrieved without accessing the input. We may interpret this setting as a two-party game: Alice is given an instance of the problem and constructs a message for Bob, who must be able to produce the output of the problem given Alice's message. Since Bob does not have any input, it suffices to consider one-way single-round communication protocols.

Main Theorem 1. *The Pattern Matching with Edits problem admits a one-way deterministic communication protocol that sends $O(n/m \cdot k \log^2 m)$ bits. Within the same communication complexity, one can also encode the family of all fragments of $T[\,i\,..\,j\,)$ that satisfy $\delta_E(P, T[\,i\,..\,j\,)) \le k$, as well as all optimal alignments $P \rightsquigarrow T[\,i\,..\,j\,)$*

[1] The $\tilde{O}(\cdot)$ and $\hat{O}(\cdot)$ notations suppress factors poly-logarithmic and sub-polynomial in the input size $n + m$, respectively.

for each of these fragments. Further, increasing the communication complexity to $O(n/m \cdot k \log m \log(m|\Sigma|))$, where Σ denotes the input alphabet, one can also retrieve the edit information for each optimal alignment.

Observe that our encoding scheme suffices to retrieve not only the set $\mathrm{Occ}_k^E(P, T)$ (which contains only starting positions of the k-error occurrences) but also the fragments of T with edit distance at most k from P. In other words, it allows retrieving all pairs $0 \le i \le j \le n$ such that $\delta_E(P, T[i \mathinner{.\,.} j)) \le k$.

We complement Main Theorem 1 with a simple lower bound that shows that our result is tight (essentially up to one logarithmic factor).

Main Theorem 2. *Fix integers n, m, k such that $n/2 \ge m > k > 0$. Every communication protocol for the Pattern Matching with Edits problem uses $\Omega(n/m \cdot k \log(m/k))$ bits for $P = 0^m$ and some $T \in \{0, 1\}^n$.*

Observe that our lower bound holds for the very simple case that the pattern is the all-zeros string and only the text contains nonzero characters. In this case, the edit distance of the pattern and another string depends only on the length and the number of nonzero characters in the other string, and we can thus easily compute the edit distance in linear time.

From Structural Insights to Better Algorithms: A Success Story

Let us take a step back and review how structural results aided the development of approximate-pattern-matching algorithms in the recent past.

First, let us review the key insight of [CKW20] that led to the breakthrough of [CKW22]. Crucially, the authors use that, for any pair of strings P and T with $|T| \le 3/2 \cdot |P|$ and threshold $k \ge 1$, either (a) P has at most $O(k^2)$ occurrences with at most k edits in T, or (b) P and the relevant part of T are at edit distance $O(k)$ to periodic strings with the same period. This insight helps as follows: First, one may derive that, indeed, all k-error occurrences of P in T form $O(k^3)$ arithmetic progressions. Second, it gives a blueprint for an algorithm: one has to tackle just two important cases: an easy *nonperiodic* case, where P and T are highly unstructured and k-error occurrences are rare, and a not-so-easy periodic case, where P and T are highly repetitive and occurrences are frequent but appear in a structured manner.

The structural insights of [CKW20] have found widespread other applications. For example, they readily yielded algorithms for differentially private approximate pattern matching [Ste24], approximate circular pattern matching problems [CKP+21, CKP+22, CPR+24], and they even played a key role in obtaining small-space algorithms for (online) language distance problems [BKS23], among others.

Interestingly, an insight similar to the one of [CKW20] was first obtained in [BKW19] for the much easier problem of Pattern Matching with Mismatches (where we allow neither insertions nor deletions) before being tightened and ported to Pattern Matching with Edits in [CKW20]. Similarly, in this paper, we port a known communication complexity bound from Pattern Matching with Mismatches to Pattern Matching with Edits; albeit with a much more involved proof. As proved in [CKP19], Pattern Matching with Mismatches problem admits a one-way deterministic $O(k \log(m|\Sigma|/k))$-bit communication protocol. While we discuss later (in the Technical Overview) the result of [CKP19] as well as the challenges in porting it to Pattern Matching with Edits, let us highlight here that their result was crucial for obtaining an essentially optimal *streaming* algorithm for Pattern Matching with Mismatches.

Finally, let us discuss the future potential of our new structural results. First, as a natural generalization of [CKP19], $\hat{O}(k)$-space algorithms for Pattern Matching with Edits should be plausible in the semi-streaming and (more ambitiously) streaming models, because $\hat{O}(k)$-size edit distance sketches have been developed in parallel to this work [KS24]. Nevertheless, such results would also require $\hat{O}(k)$-space algorithms constructing sketches and recovering the edit distance from the two sketches, and [KS24] does not provide such space-efficient algorithms. Second, our result sheds more light on the structure of the non-periodic case of [CKW20]: as it turns out, when relaxing the notion of periodicity even further, we obtain a periodic structure also for patterns with just a (sufficiently large) constant number of k-error occurrences. This opens up a perspective for classical Pattern Matching with Edits algorithms that are even faster than $\tilde{O}(n/m + k^3)$.

Application of our Main Result: Quantum Pattern Matching with Edits

As a fundamental problem, Pattern Matching with Edits has been studied in a plethora of settings, including the compressed setting [GS13, Tis14, BLR+15, CKW20], the dynamic setting [CKW20], and the streaming setting [Sta17, KPS21, BK23], among others. However, so far, the *quantum setting* remains vastly unexplored. While quantum algorithms have been developed for Exact Pattern Matching [HV03], Pattern Matching with Mismatches [JN23], Longest Common Factor (Substring) [GS23, AJ23, JN23], Lempel–Ziv factorization [GJKT24], as well as other fundamental string problems [AGS19, WY24, ABI+20, BEG+21, CKK+22], no quantum algorithm for Pattern Matching with Edits has been known so far. The challenge posed by Pattern Matching with Edits, in comparison to Pattern Matching with Mismatches, arises already from the fact that, while the computation of Hamming distance between two strings can be easily accelerated in the quantum setting, the same is not straightforward for the edit distance case. Only very recently, Gibney, Jin, Kociumaka, and Thankachan [GJKT24] demonstrated a quantum edit-distance algorithm with the optimal query complexity of $\tilde{O}(\sqrt{kn})$ and the time complexity of $\tilde{O}(\sqrt{kn} + k^2)$.

We follow the long line of research on quantum algorithms on strings and employ our new structural results (combined with the structural results from [CKW20]) to obtain the following quantum algorithms for the Pattern Matching with Edits problem.

Main Theorem 3. *Let P denote a pattern of length m, let T denote a text of length n, and let $k > 0$ denote an integer threshold.*

(1) *There is a quantum algorithm that solves the Pattern Matching with Edits problem using $\hat{O}(n/m \cdot \sqrt{km})$ queries and $\hat{O}(n/m \cdot (\sqrt{k}m + k^{3.5}))$ time.*

(2) *There is a quantum algorithm deciding whether $\mathrm{Occ}^E_k(P, T) \neq \varnothing$ using $\hat{O}(\sqrt{n/m} \cdot \sqrt{km})$ queries and $\hat{O}(\sqrt{n/m} \cdot (\sqrt{k}m + k^{3.5}))$ time.*

Surprisingly, for $n = O(m)$, we achieve the same query complexity as quantum algorithms for computing the (bounded) edit distance [GJKT24] and even the bounded Hamming distance of strings (a simple application of Grover search yields an $\tilde{O}(\sqrt{kn})$ upper bound; a matching $\Omega(\sqrt{kn})$ lower bound is also known [BBC+01]). While we did not optimize the time complexity of our algorithms (reasonably, one could expect a time complexity of $\tilde{O}(n/m \cdot (\sqrt{km} + k^{3.5}))$ based on our structural insights and [CKW22]), we show that our query complexity is essentially optimal by proving a matching lower bound.

Main Theorem 4. *Let us fix integers $n \geq m > k > 0$.*

(1) *Every quantum algorithm that solves the Pattern Matching with Edits problem uses $\Omega(n/m \cdot \sqrt{k(m-k)})$ queries for $P = 0^m$ and some $T \in \{0,1\}^n$.*

(2) *Every quantum algorithm that decides whether $\mathrm{Occ}^E_k(P, T) \neq \varnothing$ uses $\Omega(\sqrt{n/m} \cdot \sqrt{k(m-k)})$ queries for $P = 0^m$ and some $T \in \{0,1\}^n$.*

Again, our lower bounds hold already for the case when the pattern is the all-zeroes string and just the text contains nonzero entries.

2 Technical Overview

In this section, we describe the technical contributions behind our positive results: Main Theorems 1 and 3. We assume that $n \le {}^3/_2\, m$ (if the text is longer, one may split the text into $O(n/m)$ overlapping pieces of length $O(m)$ each) and that $k = o(m)$ (for $k = \Theta(m)$, our results trivialize).

2.1 Communication Complexity of Pattern Matching with Mismatches

Before we tackle Main Theorem 1, it is instructive to learn how to prove an analogous result for Pattern Matching with Mismatches. Compared to the original approach of Clifford, Kociumaka, and Porat [CKP19], we neither optimize logarithmic factors nor provide an efficient decoding algorithm; this enables significant simplifications. Recall that our goal is to encode the set $\mathrm{Occ}^H_k(P,T)$, which is the Hamming-distance analog of the set $\mathrm{Occ}^E_k(P,T)$. Formally, we set

$$\mathrm{Occ}^H_k(P,T) := \{i \in [\,0\,..\,n-m\,] : \delta_H(P, T[\,i\,..\,i+m\,)) \le k\}.$$

Without loss of generality, we assume that $\{0, n-m\} \subseteq \mathrm{Occ}^H_k(P,T)$, that is, P has k-mismatch occurrences both as a prefix and as a suffix of T. Otherwise, either we have $\mathrm{Occ}^H_k(P,T) = \varnothing$ (which can be encoded trivially), or we can crop T by removing the characters to the left of the leftmost k-mismatch occurrence and to the right of the rightmost k-mismatch occurrence.

Encoding All k-Mismatch Occurrences. First, if $k = 0$, as a famous consequence of the Periodicity Lemma [FW65], the set $\mathrm{Occ}^H_0(P,T) = \mathrm{Occ}(P,T)$ is guaranteed to form a single arithmetic progression (recall that $n \le {}^3/_2\, m$ and see Lemma 3.2), and thus it can be encoded using $O(\log m)$ bits. Consult Figure 1 for a visualization of an example.

If $k > 0$, the set $\mathrm{Occ}^H_k(P,T)$ does not necessarily form an arithmetic progression. Still, we may consider the smallest arithmetic progression that contains $\mathrm{Occ}^H_k(P,T)$ as a subset. Since $0 \in \mathrm{Occ}^H_k(P,T)$, the difference of this progression can be expressed as $g := \gcd(\mathrm{Occ}^H_k(P,T))$.

A crucial property of the $\gcd(\cdot)$ function is that, as we add elements to a set maintaining its greatest common divisor g, each insertion either does not change g (if the inserted element is already a multiple of g) or results in the value g decreasing by a factor of at least 2 (otherwise). Consequently, there is a set $\{0, n-m\} \subseteq S \subseteq \mathrm{Occ}^H_k(P,T)$ of size $|S| = O(\log m)$ such that $\gcd(S) = \gcd(\mathrm{Occ}^H_k(P,T)) = g$.

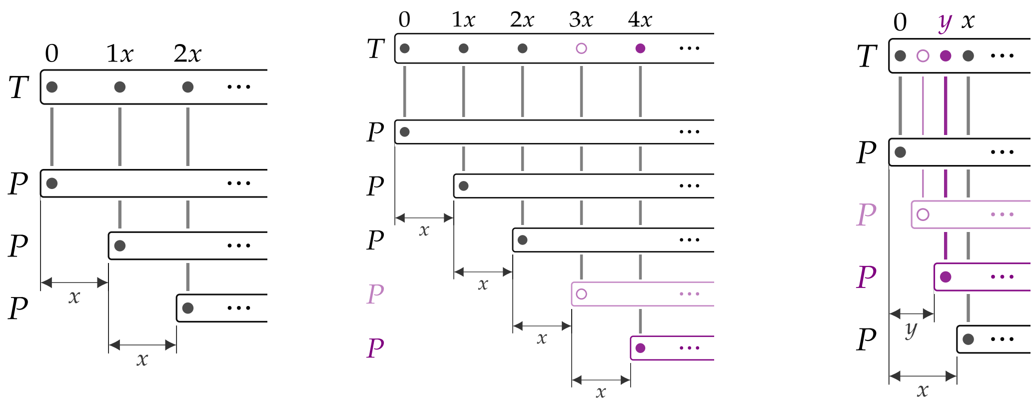
The encoding that Alice produces consists of the set S with each k-mismatch occurrences $i \in S$ augmented with the *mismatch information* for P and $T[\,i\,..\,i+m\,)$, that is, a set

$$\{(j, P[j], T[i+j]) : j \in [\,0\,..\,m\,) \text{ such that } P[j] \ne T[i+j]\}.$$

For a single k-mismatch occurrence, the mismatch information can be encoded in $O(k\log(m|\Sigma|))$ bits, where Σ is the alphabet of P and T. Due to $|S| = O(\log m)$, the overall encoding size is $O(k \log m \log(m|\Sigma|))$.

Recovering the k-Mismatch Occurrences. It remains to argue that the encoding is sufficient for Bob to recover $\mathrm{Occ}^H_k(P,T)$. To that end, consider a graph $\mathbf{G}_S$ whose vertices correspond to characters in P and T. For every $i \in S$ and $j \in [\,0\,..\,m\,)$, the graph $\mathbf{G}_S$ contains an edge between $P[j]$ and $T[i+j]$. If $P[j] = T[i+j]$, then the edge is *black*; otherwise, the edge is *red* and annotated with the values $P[j] \ne T[i+j]$. Observe that Bob can reconstruct $\mathbf{G}_S$ using the set S and the mismatch information for the k-mismatch occurrences at positions $i \in S$.

Next, we focus on the connected components of the graph $\mathbf{G}_S$. We say that a component is black if all of its edges are black and red if it contains at least one red edge. Observe that Bob can reconstruct the values of all characters in red components: the annotations already provide this information for vertices incident to red edges, and since black edges connect matching characters, the values can be propagated



(a) The pattern P occurs in T starting at the positions 0, x, and $2x$; these starting positions form the arithmetic progression $(ix)_{0\le i\le 2}$.

(b) Suppose that we were to identify an additional occurrence of P in T starting at position $4x$. Now, since occurrences start at $0, 2x$, and $4x$ (which in particular implies that $T[0\,..\,2x+|P|\,) = T[\,2x\,..\,4x+|P|\,)$), as well as at position x, we directly obtain that there is also an occurrence that starts at position $3x$ in T; which means that the arithmetic progression from Figure 1a is extended to $(ix)_{0\le i\le 4}$. More generally, one may prove that any additional occurrence at a position ix extends the existing arithmetic progression in a similar fashion.

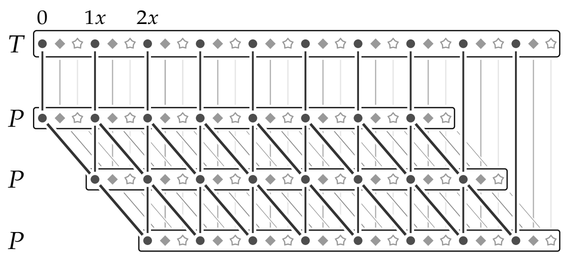
(c) Suppose that we were to identify an additional occurrence of P in T starting at position $0 < y < x$. Now, similarly to Figure 1b, we can argue that there is also an occurrence that starts at every position of the form $i\gcd(x, y)$ (this is a consequence of the famous Periodicity Lemma due to [FW65]; see Lemma 3.1)—again an arithmetic progression.
Crucially, the difference of the arithmetic progression obtained in this fashion decreased by a factor of at least two compared to the initial arithmetic progression.

Figure 1 The structure of occurrences of exact pattern matching is easy: either all exact occurrences of P in T form an arithmetic progression or there is just one such occurrence (which we may also view as a degenerate arithmetic progression).
Depicted is a text T and exact occurrences starting at the positions denoted above the text; we may assume that there is an occurrence that starts at position 0 and that there is an occurrence that ends at position $|T| - 1$.

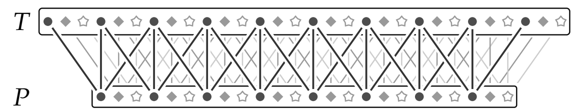
along black edges, ultimately covering all vertices in red components. The values of characters in black components remain unknown, but each black component is guaranteed to be *uniform*, meaning that every two characters in a single black component match.

The last crucial observation is that the connected components of $\mathbf{G}_S$ are very structured: for every remainder $c \in [\,0\,..\,g\,)$ modulo g, there is a connected component consisting of all vertices $P[i]$ and $T[i]$ with $i \equiv_g c$. This can be seen as a consequence of the Periodicity Lemma [FW65] applied to strings obtained from P and T by replacing each character with a unique identifier of its connected component. Consult Figure 2 for an illustration of an example for the special case if there are no mismatches and consult Figure 3 for a visualization of an example with mismatches.

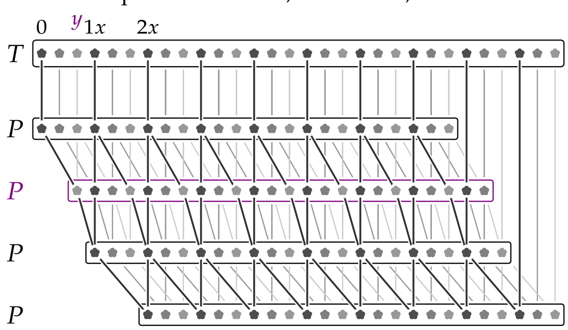
Testing if an Occurrences Starts at a Given Position. With these ingredients, we are now ready to explain how Bob tests whether a given position $i \in [\,0\,..\,n-m\,]$ belongs to $\mathrm{Occ}^H_k(P, T)$. If i is not divisible by g, then for sure $i \notin \mathrm{Occ}^H_k(P, T)$. Otherwise, for every $j \in [\,0\,..\,m\,)$, the characters $P[j]$ and $T[i+j]$ belong to the same connected component. If this component is red, then Bob knows the values of $P[j]$ and $T[i+j]$, so he can simply check if the characters match. Otherwise, the component is black, meaning that $P[j]$ and $T[i+j]$ are guaranteed to match. As a result, Bob can compute the Hamming distance



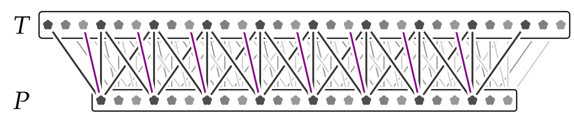
(a) Compare Figure 1a. So far, we identified three occurrences of P in T; each occurrence is an exact occurrence. Correspondingly, we have $S = \{(0, \varnothing), (x, \varnothing), (2x, \varnothing)\}$. With this set S, we obtain three different black components, which we depict with a circle, a diamond, or a star.



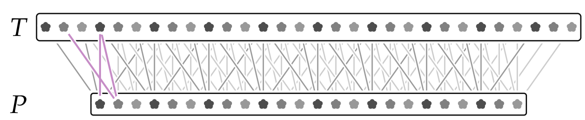
(b) The graph $\mathbf{G}_S$ that corresponds to Figure 2a: observe how we collapsed the different patterns from Figure 2a into a single pattern P.
In the example, we have three black components, that is, $\mathrm{bc}(\mathbf{G}_S) = 3$.



(c) Suppose that we were to identify an additional occurrence of P in T starting at position $0 < y < x$ (highlighted in purple). From Figure 1c, we know how the set of all occurrences changes, but—and this is the crucial point—we do not add all of these implicitly found occurrences to S, but just y.
In our example, we observe that the black components collapse into a single black component, which we depict with a cloud.

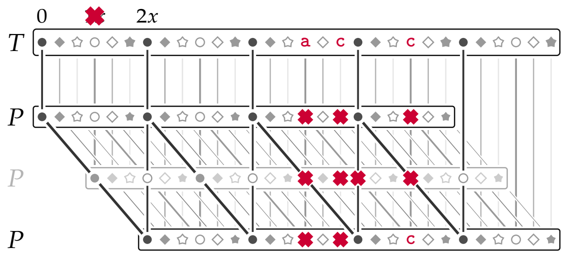


(d) The graph $\mathbf{G}_S$ that corresponds to Figure 2c: observe how we collapsed the different patterns from Figure 2c into a single pattern P. Highlighted in purple are some of the edges that we added due to the new occurrence that we added to S.
In the example, we have one black components, that is, $\mathrm{bc}(\mathbf{G}_S) = 1$.

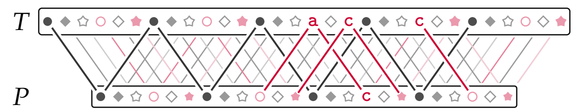


(e) Recovering an occurrence in $\mathbf{G}_S$ from Figure 2d that starts at position $\gcd(x, y)$, illustrated for the first character of the pattern.

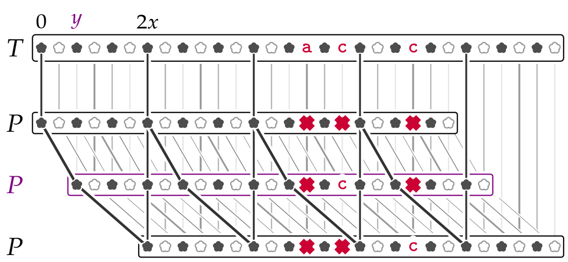
Figure 2 Compare Figure 1: we fully understand the easy structure of exact pattern matching. In this figure, we reinterpret our knowledge in terms of the encoding scheme of Alice for Pattern Matching with Mismatches (in particular we show just the occurrences included in the set S) and showcase how the corresponding graph $\mathbf{G}_S$ and its black components evolve.
We connect the same positions in P, as well as pairs of positions that are aligned by an occurrence of P in T. As there are no mismatches, every such line implies that the connected characters are equal.
For each connected component of the resulting graph (a black component), we know that all involved positions in P and T must have the same symbol. For illustrative purposes, we assume that $x = 3$ and we replace each character of a black component with a sentinel character (unique to that component), that is, we depict the strings $P^{\#}$ and $T^{\#}$.



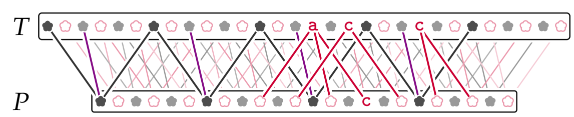
(a) Compare Figure 2a. We depict mismatched characters in an alignment of P to T by placing a cross over the corresponding character in P.
If we allow at most 3 mismatches, we now do not have an occurrence starting at position x anymore; hence we obtain six black components.



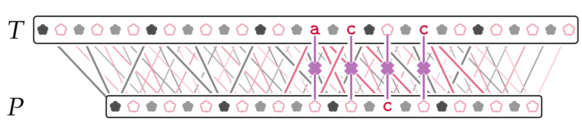
(b) The graph $\mathbf{G}_S$ that correspond to Figure 3a. We make explicit characters that are different from the "default" character of a component; the corresponding red edges (that are highlighted) are exactly the mismatch information that is stored in S. For the remaining edges, the color depicts the color of the connected component that they belong to.
In the example, we have four black components, that is, $\mathrm{bc}(\mathbf{G}_S) = 4$.
(Observe that contrary to what the image might make you believe, not every "non-default" character needs to end in a highlighted red edge.)



(c) Compare Figure 2c. We are still able to identify an additional occurrence of P in T starting at position $0 < y < x$ (highlighted in purple). Now, as before, connected components of $\mathbf{G}_S$ merge; this time, this also means that some characters that were previously part of a black component now become part of a red component (but crucially never vice-versa).
In the example, this means that we now have just a single black component, that is, $\mathrm{bc}(\mathbf{G}_S) = 1$.



(d) The graph $\mathbf{G}_S$ for the situation in Figure 3c. Again, we make explicit characters that are different from the "default" character of a component; the corresponding red edges (that are highlighted) are exactly the mismatch information that is stored in S. For the remaining edges, the color depicts the color of the connected component that they belong to (where purple highlights some of the black edges added due to the new occurrence).



(e) Checking for an occurrence at position $2\gcd(x, y)$ (which would be an occurrence were it not for mismatched characters). We check two things, first that the black component aligns; and second, for the red component where we know all characters, we compute exactly the Hamming distance (which is 4 in the example, meaning that there is no occurrence at the position in question).

Figure 3 Compared to Figure 2, we now have characters in P and T that mismatch. Again, we showcase how the corresponding graph $\mathbf{G}_S$ and its black components evolve; in the example, we allow for up to $k = 3$ mismatches. Again, for illustrative purposes, we assume that $x = 3$ and we replace each character of a black component with a sentinel character (unique to that component), that is, we depict the strings $P^{\#}$ and $T^{\#}$.

$\delta_H(P, T[\,i\,..\,i+m\,))$ and check if it does not exceed k. In either case (as long as i is divisible by g), he can even retrieve the underlying mismatch information.

A convenient way of capturing Bob's knowledge about P and T is to construct auxiliary strings $P^\#$ and $T^\#$ obtained from P and T, respectively, by replacing all characters in each black component with a sentinel character (unique for the component). Then, $\mathrm{Occ}^H_k(P,T) = \mathrm{Occ}^H_k(P^\#, T^\#)$ and the mismatch information is preserved for the k-mismatch occurrences.

2.2 Communication Complexity of Pattern Matching with Edits

On a very high level, our encoding for Pattern Matching with Edits builds upon the approach for Pattern Matching with Mismatches presented above:

- Alice still constructs an appropriate size-$\mathcal{O}(\log m)$ set S of k-error occurrences of P in T, including a prefix and a suffix of T.
- Bob uses the edit information for the occurrences in S to construct a graph $\mathbf{G}_S$ and strings $P^\#$ and $T^\#$, obtained from P and T by replacing characters in some components with sentinel characters so that $\mathrm{Occ}^E_k(P,T) = \mathrm{Occ}^E_k(P^\#, T^\#)$.

At the same time, the edit distance brings new challenges, so we also deviate from the original strategy:

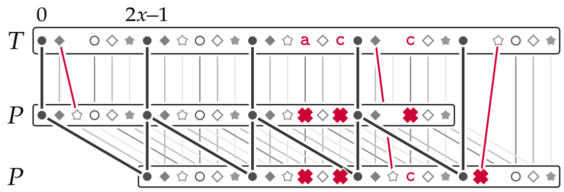
- Connected components of $\mathbf{G}_S$ do not have a simple periodic structure, so $g = \gcd(S)$ loses its meaning. Nevertheless, we prove that black components still behave in a structured way, and thus the number of black components, denoted $\mathrm{bc}(\mathbf{G}_S)$, can be used instead.
- The value $\mathrm{bc}(\mathbf{G}_S)$ is not as easy to compute as $\gcd(S)$, so we grow the set $S \subseteq \mathrm{Occ}^E_k(P,T)$ iteratively. In each step, either we add a single k-error occurrence so that $\mathrm{bc}(\mathbf{G}_S)$ decreases by a factor of at least 2, or we realize that the information related to the alignments already included in S suffices to retrieve all k-error occurrences of P in T.
- Once this process terminates, there may unfortunately remain k-error occurrences whose addition to S would decrease $\mathrm{bc}(\mathbf{G}_S)$—yet, only very slightly. In other words, such k-error occurrences generally obey the structure of black components, but may occasionally violate it. We need to understand where the latter may happen and learn the characters behind the black components involved so that they are not masked out in $P^\#$ and $T^\#$. This is the most involved part of our construction, where we use recent insights relating edit distance to compressibility [CKW23, GJKT24] and store compressed representations of certain fragments of T.

2.2.1 General Setup

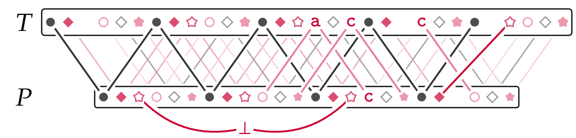
Technically, the set S that Alice constructs contains, instead of k-error occurrences $T[\,t\,..\,t'\,)$, specific alignments $P \rightsquigarrow T[\,t\,..\,t'\,)$ of cost at most k. Every such alignment describes a sequence of (at most k) edits that transform P onto $T[\,t\,..\,t'\,)$; see Definition 3.3. In the message that Alice constructs, each alignment is augmented with *edit information*, which specifies the positions and values of the edited characters; see Definition 3.5. For a single alignment of cost k, this information takes $\mathcal{O}(k\log(m|\Sigma|))$ bits, where Σ is the alphabet of P and T.

Just like for Pattern Matching with Mismatches, we can assume without loss of generality that P has k-error occurrences both as a prefix or as a suffix of T. Consequently, we always assume that S contains an alignment $\mathcal{X}_{\mathsf{pref}}$ that aligns P with a prefix of T and an alignment $\mathcal{X}_{\mathsf{suf}}$ that aligns P with a suffix of T.

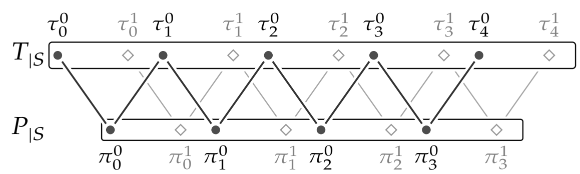
The graph $\mathbf{G}_S$ is constructed similarly as for mismatches: the vertices are characters of P and T, whereas the edges correspond to pairs of characters aligned by any alignment in S. Matched pairs of characters correspond to black edges, whereas substitutions correspond to red edges, annotated with



(a) Compare Figure 3a. In addition to mismatched characters, we now also have missing characters in P and T (depicted by a white space). Further, as alignments for occurrences are no longer unique, we have to choose an alignment for each occurrence in the set S (which can fortunately be stored efficiently).



(b) The graph $\mathbf{G}_S$ that corresponds to the situation in Figure 3a. Observe that now, we also have a sentinel vertex $\perp$ to represent that an insertion or deletion happened. Observe further that due to insertions and deletions, the last empty star character of T now belongs to the component of filled stars.
In the example, we have two black components, that is, $\mathrm{bc}(\mathbf{G}_S) = 2$.



(c) An illustration of the additional notation that we use to analyze $\mathbf{G}_S$. Removing every character involved in a red component, we obtain the strings $T_{|S}$ and $P_{|S}$. For each black component, we number the corresponding characters in P and T from left to right.

Figure 4 Compare Figures 2 and 3. In addition to mismatches, we now also allow character insertions or deletions. In the example, we depict occurrence with at most $k = 4$ edits.

the values of the mismatching characters. Insertions and deletions are also captured by red edges; see Definition 4.1 for details.

Again, we classify connected components of $\mathbf{G}_S$ into black (with black edges only) and red (with at least one red edge). Observe that Bob can reconstruct the graph $\mathbf{G}_S$ and the values of all characters in red components and that black components remain *uniform*, that is, every two characters in a single black component match. Consult Figure 4 for a visualization of an example.

Finally, we define $\mathrm{bc}(\mathbf{G}_S)$ to be the number of black components in $\mathbf{G}_S$. If $\mathrm{bc}(\mathbf{G}_S) = 0$, then Bob can reconstruct the whole strings P and T, so we henceforth assume $\mathrm{bc}(\mathbf{G}_S) > 0$.

First Insights into $\mathbf{G}_S$. Our first notable insight is that black components exhibit periodic structure. To that end, write $P_{|S}$ for the subsequence of P that contains all characters of P that are contained in a black component in $\mathbf{G}_S$ and write $T_{|S}$ for the subsequence of T that contains all characters of T that are contained in a black component in $\mathbf{G}_S$. Then, for every $c \in [0\mathinner{.\,.} \mathrm{bc}(\mathbf{G}_S))$, there is a component consisting of all characters $P_{|S}[i]$ and $T_{|S}[i]$ such that $i \equiv_{\mathrm{bc}(\mathbf{G}_S)} c$; for a formal statement and proof, consult Lemma 4.4. Also consult Figure 4c for an illustration of an example.

Next, we denote the positions in P and T of the subsequent characters of $P_{|S}$ and $T_{|S}$ belonging to a specific component $c \in [0\mathinner{.\,.} \mathrm{bc}(\mathbf{G}_S))$ as $\pi_0^c, \pi_1^c, \ldots$ and $\tau_0^c, \tau_1^c, \ldots$, respectively; see Definition 4.7. The characterization of the black components presented above implies that $\pi_j^c < \pi_{j'}^{c'}$ if and only if either $j < j'$ or $j = j'$ and $c < c'$ (analogously for $\tau_j^c < \tau_{j'}^{c'}$). We assume that the cth black component contains m_c characters of P and n_c characters in T; note that $m_c \in \{m_0, m_0 - 1\}$ and $n_c \in \{n_0, n_0 - 1\}$.

2.2.2 Extra Information to Capture Close Alignments

By definition of the graph $\mathbf{G}_S$, the alignments in S obey the structure of the black components. Specifically, for every $\mathcal{X} \in S$, there is a shift $i \in [\,0\,..\,n_0 - m_0\,]$ such that $\mathcal{X}$ matches $P[\pi_j^c]$ with $T[\tau_{i+j}^c]$ for every $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ and $j \in [\,0\,..\,m_c\,)$. The quasi-periodic structure of P and T suggests that we should expect further shifts $i \in [\,0\,..\,n_0 - m_0\,]$ with low-cost alignments matching $P[\pi_j^c]$ with $T[\tau_{i+j}^c]$ for every $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ and $j \in [\,0\,..\,m_c\,)$. Unfortunately, even if an optimum alignment $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$ matches $P[\pi_j^c]$ with $T[\tau_{i+j}^c]$, there is no guarantee that it also matches $P[\pi_{j'}^{c'}]$ with $T[\tau_{i+j'}^{c'}]$ for other values $c' \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ and $j' \in [\,0\,..\,m_{c'}\,)$. Even worse, it is possible that no optimal alignment $P[\,\pi_j^{c-1}\,..\,\pi_j^{c+1}\,) \rightsquigarrow T[\,\tau_{i+j}^{c-1}\,..\,\tau_{i+j}^{c+1}\,)$ matches $P[\pi_j^c]$ with $T[\tau_{i+j}^c]$. The reason behind this phenomenon is that the composition of optimal edit-distance alignments is not necessarily optimal (more generally, the edit information of optimal alignments $X \rightsquigarrow Y$ and $Y \rightsquigarrow Z$ is insufficient to recover $\delta_E(X, Z)$).

In these circumstances, our workaround is to identify a set $C_S \subseteq [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ such that the underlying characters can be encoded in $\tilde{O}(k|S|)$ space and every alignment $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$ that we need to capture matches $P[\pi_j^c]$ with $T[\tau_{i+j}^c]$ for every $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,) \setminus C_S$ and $j \in [\,0\,..\,m_c\,)$. For this, we investigate how an optimal alignment $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$ may differ from a canonical alignment $\mathcal{A} : P \rightsquigarrow T[\,t\,..\,t'\,)$ that matches $P[\pi_j^c]$ with $T[\tau_{i+j}^c]$ for all $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ and $j \in [\,0\,..\,m_c\,)$. Following recent insights from [CKW23, GJKT24] (see Lemma 3.9), we observe that the fragments of P on which $\mathcal{A}$ and $\mathcal{X}$ are disjoint can be compressed into $O(\delta_E^{\mathcal{A}}(P, T[\,t\,..\,t'\,)))$ space (using Lempel–Ziv factorization [ZL77], for example). Moreover, the compressed size of each of these fragments is at most proportional to the cost of $\mathcal{A}$ on the fragment. Consequently, our goal is to understand where $\mathcal{A}$ makes edits and learn all the fragments of P (and T) with a sufficiently high density of edits compared to the compressed size. Due to the quasi-periodic nature of P and T, for each $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$, all characters in the cth black component are equal to $T[\tau_0^c]$, so we can focus on learning fragments of $T[\,\tau_0^0\,..\,\tau_0^{\mathrm{bc}(\mathbf{G}_S)-1}\,]$.

The bulk of the alignment $\mathcal{A}$ can be decomposed into pieces that align $P[\,\pi_j^c\,..\,\pi_j^{c+1}\,)$ onto $T[\,\tau_{i+j}^c\,..\,\tau_{i+j}^{c+1}\,)$. In Lemma 4.10, we prove that $\delta_E(P[\,\pi_j^c\,..\,\pi_j^{c+1}\,), T[\,\tau_{i+j}^c\,..\,\tau_{i+j}^{c+1}\,)) \le \mathrm{w}_S(c)$, where $\mathrm{w}_S(c)$ is the total cost incurred by alignments in S on all fragments $P[\,\pi_{j'}^c\,..\,\pi_{j'}^{c+1}\,)$ for $j' \in [\,0\,..\,m_c\,)$. Intuitively, this is because the path from $P[\pi_j^c]$ to $T[\tau_{i+j}^c]$ in $\mathbf{G}_S$ allows obtaining an alignment $P[\,\pi_j^c\,..\,\pi_j^{c+1}\,) \rightsquigarrow T[\,\tau_{i+j}^c\,..\,\tau_{i+j}^{c+1}\,)$ as a composition of pieces of alignments in S and their inverses. Every component $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ uses distinct pieces, so the total weight $w := \sum_c \mathrm{w}_S(c)$ does not exceed $k \cdot |S|$.

The weight function $\mathrm{w}_S(c)$ governs which characters of $T[\,\tau_0^0\,..\,\tau_0^{\mathrm{bc}(\mathbf{G}_S)-1}\,]$ need to be learned. We formalize this with a notion of a *period cover* $C_S \subseteq [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$; see Definition 4.14. Most importantly, we require that $[\,a\,..\,b\,] \subseteq C_S$ holds whenever the compressed size of $T[\,\tau_0^a\,..\,\tau_0^b\,]$ is smaller than the total weight $\sum_{c=a-1}^{b} \mathrm{w}_S(c)$ (scaled up by an appropriate constant factor). Additionally, to handle corner cases, we also learn the longest prefix and the longest suffix of $T[\,\tau_0^0\,..\,\tau_0^{\mathrm{bc}(\mathbf{G}_S)-1}\,]$ of compressed size $O(w + k)$. As proved in Lemma 4.15, the set $\{(c, T[\tau_0^c]) : c \in C_S\}$ can be encoded in $O((w + k)\log(m|\Sigma|)) = O(k|S|\log(m|\Sigma|))$ bits on top of the graph $\mathbf{G}_S$ (which can be recovered from the edit information for alignments in S).

Following the aforementioned strategy of comparing the regions where $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$ is disjoint with the canonical alignment $\mathcal{A} : P \rightsquigarrow T[\,t\,..\,t'\,)$, we prove the following result. Due to corner cases arising at the endpoints of $T[\,t\,..\,t'\,)$ and between subsequent fragments $T[\,\tau_{i+j}^0\,..\,\tau_{i+j}^{\mathrm{bc}(\mathbf{G}_S)-1}\,]$ and $T[\,\tau_{i+j+1}^0\,..\,\tau_{i+j+1}^{\mathrm{bc}(\mathbf{G}_S)-1}\,]$, the proof is rather complicated.

Proposition 4.24. *Let $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$ be an optimal alignment of P onto a fragment $T[\,t\,..\,t'\,)$ such that $\delta_E(P, T[\,t\,..\,t'\,)) \le k$. If there exists $i \in [\,0\,..\,n_0 - m_0\,]$ such that $|\tau_i^0 - t - \pi_0^0| \le w + 3k$, then the following holds for every $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,) \setminus C_S$:*

(1) *$\mathcal{X}$ aligns $P[\pi_j^c]$ to $T[\tau_{i+j}^c]$ for every $j \in [\,0\,..\,m_c\,)$, and*

(2) *$\tau_{i'}^c \notin [\,t\,..\,t'\,)$ for every $i' \in [\,0\,..\,n_c\,) \setminus [\,i\,..\,i + m_c\,)$.*

2.2.3 Extending S with Uncaptured Alignments

Proposition 4.24 indicates that S *captures* all k-error occurrences $T[\,t\,..\,t'\,)$ such that $|\tau_i^0 - t - \pi_0^0| \le w + 3k$ holds for some $i \in [\,0\,..\,n_0 - m_0\,]$. As long as S does not capture some k-error occurrence $T[\,t\,..\,t'\,)$, we add an underlying optimal alignment $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$ to the set S. In Lemma 4.25, we prove that $\mathrm{bc}(\mathbf{G}_{S\cup\{\mathcal{X}\}}) \le \mathrm{bc}(\mathbf{G}_S)/2$ holds for such an alignment $\mathcal{X}$. For this, we first eliminate the possibility of $t + \pi_0^0 \gg \tau_{n_0-m_0}^0$ (using $\mathcal{X}_{\mathsf{suf}} \in S$, which matches $P[\pi_0^0]$ with $T[\tau_{n_0-m_0}^0]$). If $|\tau_i^0 - t - \pi_0^0| > w + 3k$ holds for every $i \in [\,0\,..\,n_0\,)$, on the other hand, then there is no $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ such that $P[\pi_0^c]$ can be matched with any character in the cth connected component. Consequently, each black component becomes red or gets merged with another black component, resulting in the claimed inequality $\mathrm{bc}(\mathbf{G}_{S\cup\{\mathcal{X}\}}) \le \mathrm{bc}(\mathbf{G}_S)/2$.

By Lemma 4.25 and since $\mathrm{bc}(\mathbf{G}_S) \le m$ holds when we begin with $|S| = 2$, the total size $|S|$ does not exceed $O(\log m)$ before we either arrive at $\mathrm{bc}(\mathbf{G}_S) = 0$, in which case the whole input can be encoded in $O(k|S|\log(m|\Sigma|))$ bits, or S captures all k-error occurrences. In the latter case, the encoding consists of the edit information for all alignments in S, as well as the set $\{(c, T[\tau_0^c]) : c \in C_S\}$ encoded using Lemma 4.15. Based on this encoding, we can construct strings $P^\#$ and $T^\#$ obtained from P and T, respectively, by replacing with $\#_c$ every character in the cth connected component for every $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,) \setminus C_S$. As a relatively straightforward consequence of Proposition 4.24, in Theorem 4.27 we prove that $\mathrm{Occ}_k^E(P, T) = \mathrm{Occ}_k^E(P^\#, T^\#)$ and that the edit information is preserved for every optimal alignment $P \rightsquigarrow T[\,t\,..\,t'\,)$ of cost at most k.

2.3 Quantum Query Complexity of Pattern Matching with Edits

As an illustration of the applicability of the combinatorial insights behind our communication complexity result (Main Theorem 1), we study quantum algorithms for Pattern Matching with Edits. As indicated in Main Theorems 3 and 4, the query complexity we achieve is only a sub-polynomial factor away from the unconditional lower bounds, both for the decision version of the problem (where we only need to decide whether $\mathrm{Occ}_k^E(P, T)$ is empty or not) and for the standard version asking to report $\mathrm{Occ}_k^E(P, T)$.

Our lower bounds (in Main Theorem 4) are relatively direct applications of the adversary method of Ambainins [Amb02], so this overview is solely dedicated to the much more challenging upper bounds. Just like for the communication complexity above, we assume that $n \le {}^3\!/_2\, m$ and $k = o(m)$. In this case, our goal is to achieve the query complexity of $\hat{O}(\sqrt{km})$.

Our solution incorporates four main tools:

- the universal approximate pattern matching algorithm of [CKW20],
- the recent quantum algorithm for computing (bounded) edit distance [GJKT24],
- the novel combinatorial insights behind Main Theorem 1,
- a new quantum $n^{o(1)}$-factor approximation algorithm for edit distance that uses $\hat{O}(\sqrt{n})$ queries and is an adaptation of a classic sublinear-time algorithm of [GKKS22].

2.3.1 Baseline Algorithm

We set the stage by describing a relatively simple algorithm that relies only on the first two of the aforementioned four tools. This algorithm makes $\tilde{O}(\sqrt{k^3 m})$ quantum queries to decide whether $\mathrm{Occ}_k^E(P,T) = \varnothing$.

The findings of [CKW20] outline two distinct scenarios: either there are *few k-error occurrences of P in T* or the pattern is *approximately periodic*. In the former case, the set $\mathrm{Occ}_k^E(P,T)$ is of size $O(k^2)$, and it is contained in a union of $O(k)$ intervals of length $O(k)$ each. In the latter case, a primitive *approximate period* Q of small length $|Q| = O(m/k)$ exists such that P and the relevant portion of T (excluding the characters to the left of the leftmost k-error occurrence and to the right of the rightmost k-error occurrence) are at edit distance $O(k)$ to substrings of Q^∞. It is solely the pattern that determines which of these two cases holds: the initial two options in the following lemma correspond to the *non-periodic* case, where there are few k-error occurrences of P in T, whereas the third option indicates the (approximately) *periodic* case, where the pattern admits a short approximate period Q. Here, $\delta_E(S, {}^*Q^*)$ denotes the minimum edit distance between S and any substring of Q^∞.

Lemma 2.1 ([CKW20, Lemma 5.19])**.** *Let P denote a string of length m and let $k \le m$ denote a positive integer. Then, at least one of the following holds:*

(a) *The string P contains $2k$ disjoint fragments $B_1, \ldots, B_{2k}$ (called* breaks*) each having period* $\mathrm{per}(B_i) > m/128k$ *and length $|B_i| = \lfloor m/8k \rfloor$.*

(b) *The string P contains disjoint* repetitive regions *$R_1, \ldots, R_r$ of total length $\sum_{i=1}^{r} |R_i| \ge 3/8 \cdot m$ such that each region R_i satisfies $|R_i| \ge m/8k$ and has a primitive* approximate period *Q_i with $|Q_i| \le m/128k$ and $\delta_E(R_i, {}^*Q_i^*) = \lceil 8k/m \cdot |R_i| \rceil$.*

(c) *The string P has a primitive* approximate period *Q with $|Q| \le m/128k$ and $\delta_E(P, {}^*Q^*) < 8k$.*

The proof of Lemma 2.1 is constructive, providing a classical algorithm that performs the necessary decomposition and identifies the specific case. The analogous procedure for Pattern Matching with Mismatches also admits an efficient quantum implementation [JN23] using $\tilde{O}(\sqrt{km})$ queries and time. As our first technical contribution (Lemma 6.5), we adapt the decomposition algorithm for the edit case to the quantum setting so that it uses $\tilde{O}(\sqrt{km})$ queries and $\tilde{O}(\sqrt{km} + k^2)$ time.

Compared to the classic implementation in [CKW20] and the mismatch version in [JN23], it is not so easy to efficiently construct repetitive regions. In this context, we are given a length-$\lfloor m/8k \rfloor$ fragment with exact period Q_i and the task is to extend it to R_i so that $k_i := \delta_E(R_i, {}^*Q_i^*)$ reaches $\lceil 8k/m \cdot |R_i| \rceil$. Previous algorithms use Longest Common Extension queries and gradually grow R_i, increasing k_i by one unit each time; this can be seen as an online implementation of the Landau–Vishkin algorithm for the bounded edit distance problem [LV88]. Unfortunately, the near-optimal quantum algorithm for bounded edit distance [GJKT24] is much more involved and does not seem amenable to an online implementation. To circumvent this issue, we apply exponential search (just like in Newton's root-finding method, this is possible even though the sign of $\lceil 8k/m \cdot |R_i| \rceil - \delta_E(R_i, {}^*Q_i^*)$ may change many times). At each step, we apply a slightly extended version of the algorithm of [GJKT24] that allows simultaneously computing the edit distance between R_i and multiple substrings of Q_i^∞; see Lemma 6.2.

Once the decomposition has been computed, the next step is to apply the structure of the pattern in various cases to find the k-error occurrences. The fundamental building block needed here is a subroutine that *verifies* an interval I of $O(k)$ positive integers, that is, computes $\mathrm{Occ}_k^E(P,T) \cap I$. The aforementioned extension of the bounded edit distance algorithm of [GJKT24] (Lemma 6.2) allows implementing this operation using $\tilde{O}(\sqrt{km})$ quantum queries and $\tilde{O}(\sqrt{km} + k^2)$ time.

By directly following the approach of [CKW20], computing $\mathrm{Occ}_k^E(P,T)$ can be reduced to verification of $O(k^2)$ intervals (the periodic case constitutes the bottleneck for the number of intervals), which yields total a query complexity of $\tilde{O}(\sqrt{k^5 m})$. If we only aim to decide whether $\mathrm{Occ}_k^E(P,T)$, we can apply Grover's search on top of the verification algorithm, reducing the query complexity to $\tilde{O}(\sqrt{k^3 m})$. One can also hope for further speed-ups based on the more recent results of [CKW22], where the number of intervals is effectively reduced to $\tilde{O}(k^{1.5})$. Nevertheless, already in the non-periodic case, where the number of intervals is $O(k)$, this approach does not provide any hope of reaching query complexity beyond $\tilde{O}(\sqrt{k^2 m})$ for the decision version and $\tilde{O}(\sqrt{k^3 m})$ for the reporting version of Pattern Matching with Edits.

2.3.2 How to Efficiently Verify $O(k)$ Candidate Intervals?

As indicated above, the main bottleneck that we need to overcome to achieve the near-optimal query complexity is to verify $O(k)$ intervals using $\hat{O}(\sqrt{km})$ queries. Notably, an unconditional lower bound for bounded edit distance indicates that $\Omega(\sqrt{km})$ queries are already needed to verify a length-1 interval.

A ray of hope stemming from our insights behind Main Theorem 1 is that, as described in Section 2.2, already a careful selection of just $O(\log m)$ among the k-error occurrences reveals a lot of structure that can be ultimately used to recover the whole set $\mathrm{Occ}_k^E(P,T)$. To illustrate how to use this observation, let us initially make an unrealistic assumption that every candidate interval I contains a K-error occurrence for some $K = \hat{O}(k)$. Such occurrences can be detected using the existing verification procedure using $\tilde{O}(\sqrt{Km}) = \hat{O}(\sqrt{km})$ queries.

First, we verify the leftmost and the rightmost intervals. This allows finding the leftmost and the rightmost K-error occurrences of P in T. We henceforth assume that text T is cropped so that these two K-error occurrences constitute a prefix and a suffix of T, respectively. The underlying alignments are the initial elements of the set S that we maintain using the insights of Section 2.2. Even though these two alignments have cost at most K, for technical reasons, we subsequently allow adding to S alignments of cost up to $K' = K + O(k)$. Using the edit information for alignments $\mathcal{X} \in S$, we build the graph $\mathbf{G}_S$, calculate its connected components, and classify them as red and black components.

If there are no black components, that is, $\mathrm{bc}(\mathbf{G}_S) = 0$, then the edit information for the alignments $\mathcal{X} \in S$ allows recovering the whole input strings P and T. Thus, no further quantum queries are needed, and we complete the computation using a classical verification algorithm in $O(m + k^3)$ time.

If there are black components, we retrieve the positions $\pi_0^0, \ldots, \pi_{m_0-1}^0$ and $\tau_0^0, \ldots, \tau_{n_0-1}^0$ contained in the 0-th black component. Based on these positions, we can classify K'-error occurrences $T[\,t \mathinner{.\,.} t'\,)$ into those that are *captured* by S (for which $|\tau_i^0 - \pi_0^0 - t|$ is small for some $i \in [\,0 \mathinner{.\,.} n_0 - m_0\,]$) and those which are not captured by S. Although we do not know K'-error occurrences other than those contained in S, the test of comparing $|\tau_i^0 - \pi_0^0 - t|$ against a given threshold (which is $O(K'|S|)$) can be performed for any position t, and thus we can classify arbitrary positions $t \in [\,0 \mathinner{.\,.} |T|\,]$ into those that are captured by S and those that are not.

If any of the candidate intervals I contains a position $t \in I$ that is not captured by S, we verify that interval and, based on our assumption, obtain a K-error occurrence of P in T that starts somewhere within I. Furthermore, we can derive an optimal alignment $\mathcal{X} : P \twoheadrightarrow T[\,t \mathinner{.\,.} t'\,)$ whose cost does not exceed $K + |I| \le K'$ because $|I| = O(k)$. This K'-error occurrence is not captured by S, so we can add $\mathcal{X}$ to S and, as a result, the number of black components decreases at least twofold by Lemma 4.25.

The remaining possibility is that S captures all positions t contained in the candidate intervals I. In this case, our goal is to construct strings $P^\#$ and $T^\#$ of Theorem 4.27, which are guaranteed to satisfy $\mathrm{Occ}_k^E(P,T) \cap I = \mathrm{Occ}_k^E(P^\#,T^\#) \cap I$ for each candidate interval I because $k \le K'$. For this, we need to build a period cover C_S satisfying Definition 4.14, which requires retrieving certain compressible substrings of T. The minimum period cover C_S utilized in our encoding (Lemma 4.15) does not seem to admit an efficient quantum construction procedure, so we build a slightly larger period cover whose encoding incurs a logarithmic-factor overhead; see Lemma 4.16. The key subroutine that we repeatedly use while constructing this period cover asks to compute the longest fragment of T (or of the reverse text $\overline{T}$) that starts at a given position and admits a Lempel–Ziv factorization [ZL77] of size bounded by a given threshold. For this, we use exponential search combined with the recent quantum LZ factorization algorithm [GJKT24]; see Theorem 5.8 and Lemma 6.7. Based on the computed period cover, we can construct the strings $P^\#$ and $T^\#$ and resort to a classic verification algorithm (that performs no quantum queries) to process all $O(k)$ intervals I in time $O(m + k^3)$.

The next step is to drop the unrealistic assumption that every candidate interval I contains a K-error occurrence of P. The natural approach is to test each of the candidate intervals using an approximation algorithm that either reports that $\mathrm{Occ}_k(P, T) \cap I = \varnothing$ (in which case we can drop the interval since we are ultimately looking for k-error occurrences) or that $\mathrm{Occ}_K(P, T) \cap I \neq \varnothing$ (in which case the interval satisfies our assumption). Given that $|I|$ is much smaller than K, it is enough to approximate $\delta_E(P, T[\,t \mathinner{.\,.} t + m\,))$ for an arbitrary single position $t \in I$ (distinguishing between distances at most $O(k)$ and at least $K - O(k)$). Although the quantum complexity of approximating edit distance has not been studied yet, we observe that the recent sublinear-time algorithm of Goldenberg, Kociumaka, Krauthgamer, and Saha [GKKS22] is easy to adapt to the quantum setting, resulting in a query complexity of $\hat{O}(\sqrt{n})$ and an approximation ratio of $n^{o(1)} = \hat{O}(1)$; see Section 5.3 for details.

Unfortunately, we cannot afford to run this approximation algorithm for every candidate interval: that would require $\hat{O}(k\sqrt{m})$ queries. Our final trick is to use Grover's search on top: given a subset of the $O(k)$ candidate intervals, using just $\hat{O}(\sqrt{km})$ queries, we can either learn that none of them contains any k-error occurrence (in this case, we can discard all of them) or identify one that contains a K-error occurrence. Combined with binary search, this approach allows discarding some candidate intervals so that the leftmost and the rightmost among the remaining ones contain K-error occurrences. The underlying alignments (constructed using the exact quantum bounded edit distance algorithm of [GJKT24]) are used to initialize the set S. At each step of growing S, on the other hand, we apply our approximation algorithm to the set of all candidate intervals that are not yet (fully) captured by S. Either none of these intervals contain k-error occurrences (and the construction of S may stop), or we get one that is guaranteed to contain a K-error occurrence. In this case, we construct an appropriate low-cost alignment $\mathcal{X}$ using the exact algorithm and extend the set S with $\mathcal{X}$. Thus, the unrealistic assumption is not needed to construct the set S and the strings $P^\#$ and $T^\#$ using $\hat{O}(\sqrt{km})$ queries.

2.3.3 Handling the Approximately Periodic Case

Verifying $O(k)$ candidate intervals was the only bottleneck of the non-periodic case of Pattern Matching with Edits. In the approximately periodic case, on the other hand, we may have $O(k^2)$ candidate intervals, so a direct application of the approach presented above only yields an $\hat{O}(\sqrt{k^2 m})$-query algorithm.

Fortunately, a closer inspection of the candidate intervals constructed in [CKW20] reveals that they satisfy the unrealistic assumption that we made above: each of them contains an $O(k)$-error occurrence of P. This is because both P and the relevant part of T are at edit distance $O(k)$ from substrings of Q^∞ and each of the intervals contains a position that allows aligning P into T via the substrings of Q^∞ (so that perfect copies of Q are matched with no edits). Consequently, the set S of $O(\log m)$ alignments covering all candidate intervals can be constructed using $\tilde{O}(\sqrt{km})$ queries. Moreover, once we construct the strings $P^\#$ and $T^\#$, instead of verifying all $O(k^2)$ candidate intervals, which takes $O(m + k^4)$ time, we can use the classic $\tilde{O}(m + k^{3.5})$-time algorithm of [CKW22] to construct the entire set $\mathrm{Occ}^E_k(P^\#, T^\#) = \mathrm{Occ}^E_k(P, T)$.

3 Preliminaries

Sets. For integers $i, j \in \mathbb{Z}$, we write $[\,i \mathinner{.\,.} j\,]$ to denote the set $\{i, \ldots, j\}$ and $[\,i \mathinner{.\,.} j\,)$ to denote the set $\{i, \ldots, j-1\}$; we define the sets $(\,i \mathinner{.\,.} j\,]$ and $(\,i \mathinner{.\,.} j\,)$ similarly.

For a set S, we write kS to denote the set obtained from S by multiplying every element with k, that is, $kS := \{k \cdot s : s \in S\}$. Similarly, we define $\lfloor S/k \rfloor := \{\lfloor s/k \rfloor : s \in S\}$ and $k\lfloor S/k \rfloor := \{k \cdot \lfloor s/k \rfloor : s \in S\}$.

Strings. An *alphabet* Σ is a set of characters. We write $X = X[0]\,X[1] \cdots X[n-1] \in \Sigma^n$ to denote a *string* of length $|X| = n$ over Σ. For a *position* $i \in [\,0 \mathinner{.\,.} n\,)$, we say that $X[i]$ is the i-th character of X. For integer indices $0 \le i \le j \le |X|$, we say that $X[\,i \mathinner{.\,.} j\,) := X[i] \cdots X[j-1]$ is a *fragment* of X. We may also write $X[\,i \mathinner{.\,.} j-1\,]$, $X(\,i-1 \mathinner{.\,.} j-1\,]$, or $X(\,i-1 \mathinner{.\,.} j\,)$ for the fragment $X[\,i \mathinner{.\,.} j\,)$. A *prefix* of a string X is a fragment that starts at position 0, and a *suffix* of a string X is a fragment that ends at position $|X|-1$.

A string Y of length $m \in [\,0 \mathinner{.\,.} n\,]$ is a *substring* of another string X if there is a fragment $X[\,i \mathinner{.\,.} i+m\,)$ that is equal to Y. In this case, we say that there is an *exact occurrence* of Y at position i in X. Further, we write $\mathrm{Occ}(Y, X) := \{i \in [\,0 \mathinner{.\,.} n-m\,] : Y = X[\,i \mathinner{.\,.} i+m\,)\}$ for the set of starting positions of the (exact) occurrences of Y in X.

For two strings A and B, we write AB for their concatenation. We write A^k for the concatenation of k copies of the string A. We write A^∞ for an infinite string (indexed with non-negative integers) formed as the concatenation of an infinite number of copies of the string A. A *primitive* string is a string that cannot be expressed as A^k for any string A and any integer $k > 1$.

An integer $p \in [\,1 \mathinner{.\,.} n\,]$ is *a period* of a string $X \in \Sigma^n$ if we have $X[i] = X[i+p]$ for all $i \in [\,0 \mathinner{.\,.} n-p\,)$. *The period* of a string X, denoted $\mathrm{per}(X)$, is the smallest period of X. A string X is *periodic* if $\mathrm{per}(X) \le |X|/2$.

An important tool when dealing with periodicity is Fine and Wilf's Periodicity Lemma [FW65].

Lemma 3.1 (Periodicity Lemma [FW65]). *If p, q are periods of a string X of length $|X| \ge p + q - \gcd(p, q)$, then $\gcd(p, q)$ is a period of X.*

This allows us to derive the following relationship between exact occurrences and periodicity.

Lemma 3.2. *Consider a non-empty pattern P and a text T with $|T| \le 2|P| + 1$. If $\{0, |T| - |P|\} \subseteq \mathrm{Occ}(P, T)$, that is, P occurs both as a prefix and as a suffix of T, then $\gcd(\mathrm{Occ}(P, T))$ is a period of T.*

Proof. Write $\mathrm{Occ}(P, T) = \{p_0, \ldots, p_{\ell-1}\}$, where $0 = p_0 < \cdots < p_{\ell-1} = |T| - |P|$, and set $g := \gcd(\mathrm{Occ}(P, T))$. From $T[\,p_i \mathinner{.\,.} p_i + |P|\,) = P = T[\,p_{i-1} \mathinner{.\,.} p_{i-1} + |P|\,)$, we obtain that, for each $i \in (\,0 \mathinner{.\,.} \ell\,)$, the pattern P has period $p_i - p_{i-1}$. Since

$$\sum_{i \in (\,0 \mathinner{.\,.} \ell\,)} (p_i - p_{i-1}) = p_{\ell-1} = |T| - |P| \le |P| + 1 \le |P| + g,$$

Lemma 3.1 implies that $\gcd\{p_i - p_{i-1} : i \in (\,0 \mathinner{.\,.} \ell\,)\}$ is a period of P. By repeatedly applying the property $\gcd(a + b, b) = \gcd(a, b)$, we obtain $\gcd\{p_i - p_{i-1} : i \in (\,0 \mathinner{.\,.} \ell\,)\} = g$.

It remains to prove that g is also a period of T, that is, we need to show that $T[j] = T[j \bmod g]$ holds for each $j \in [\,0 \mathinner{.\,.} |T|\,)$. To that end, fix a position $j \in [\,0 \mathinner{.\,.} |T|\,)$. Unless $j = |P|$, $|T| = 2|P| + 1$, and $\mathrm{Occ}(P, T) = \{0, |T| - |P|\}$, we have $j \in [\,p_i \mathinner{.\,.} p_i + |P|\,)$ for some $i \in [\,0 \mathinner{.\,.} \ell\,)$. Now, as g is a period of P and a divisor of p_i, and because $\{0, p_i\} \subseteq \mathrm{Occ}(P, T)$, we obtain

$$T[j] = P[j - p_i] = P[(j - p_i) \bmod g] = P[j \bmod g] = T[j \bmod g].$$

Finally, if $j = |P|$, $|T| = 2|P| + 1$, and $\mathrm{Occ}(P, T) = \{0, |T| - |P|\}$, then we observe that $g = |P| + 1$, so $T[j] = T[j \bmod g]$ is trivially satisfied. ◀

Edit Distance and Alignments. The *edit distance* (the *Levenshtein distance* [Lev65]) between two strings X and Y, denoted by $\delta_E(X,Y)$, is the minimum number of character insertions, deletions, and substitutions required to transform X into Y. Formally, we first define an *alignment* between string fragments.

Definition 3.3 ([CKW22, Definition 2.1]). *A sequence $\mathcal{A} = (x_i, y_i)_{i=0}^{m}$ is an* alignment *of $X[x\mathinner{.\,.} x')$ onto $Y[y\mathinner{.\,.} y')$, denoted by $\mathcal{A} : X[x\mathinner{.\,.} x') \rightsquigarrow Y[y\mathinner{.\,.} y')$, if it satisfies $(x_0,y_0) = (x,y)$, $(x_{i+1},y_{i+1}) \in \{(x_i+1, y_i+1),(x_i+1,y_i),(x_i,y_i+1)\}$ for $i \in [0\mathinner{.\,.} m)$, and $(x_m,y_m) = (x',y')$. Moreover, for $i \in [0\mathinner{.\,.} m)$:*
- *If $(x_{i+1},y_{i+1}) = (x_i+1,y_i)$, we say that $\mathcal{A}$* deletes *$X[x_i]$.*
- *If $(x_{i+1},y_{i+1}) = (x_i,y_i+1)$, we say that $\mathcal{A}$* inserts *$Y[y_i]$.*
- *If $(x_{i+1},y_{i+1}) = (x_i+1,y_i+1)$, we say that $\mathcal{A}$* aligns *$X[x_i]$ to $Y[y_i]$. If additionally $X[x_i] = Y[y_i]$, we say that $\mathcal{A}$* matches *$X[x_i]$ and $Y[y_i]$; otherwise, $\mathcal{A}$* substitutes *$X[x_i]$ with $Y[y_i]$.*

Recall Figure 4 for a visualization of an example for an alignment.

The *cost* of an alignment $\mathcal{A}$ of $X[x\mathinner{.\,.} x')$ onto $Y[y\mathinner{.\,.} y')$, denoted by $\delta_E^{\mathcal{A}}(X[x\mathinner{.\,.} x'), Y[y\mathinner{.\,.} y'))$, is the total number of characters that $\mathcal{A}$ inserts, deletes, or substitutes. The edit distance $\delta_E(X,Y)$ is the minimum cost of an alignment of $X[0\mathinner{.\,.} |X|)$ onto $Y[0\mathinner{.\,.} |Y|)$. An alignment of X onto Y is *optimal* if its cost is equal to $\delta_E(X,Y)$.

An alignment $\mathcal{A}'' : X[x\mathinner{.\,.} x') \rightsquigarrow Z[z\mathinner{.\,.} z')$ is a *product* of alignments $\mathcal{A} : X[x\mathinner{.\,.} x') \rightsquigarrow Y[y\mathinner{.\,.} y')$ and $\mathcal{A}' : Y[y\mathinner{.\,.} y') \rightsquigarrow Z[z\mathinner{.\,.} z')$ if, for every $(\bar{x},\bar{z}) \in \mathcal{A}''$, there is $\bar{y} \in [y\mathinner{.\,.} y']$ such that $(\bar{x},\bar{y}) \in \mathcal{A}$ and $(\bar{y},\bar{z}) \in \mathcal{A}'$. A product alignment always exists, and every product alignment satisfies

$$\delta_E^{\mathcal{A}''}(X[x\mathinner{.\,.} x'), Z[z\mathinner{.\,.} z')) \le \delta_E^{\mathcal{A}}(X[x\mathinner{.\,.} x'), Y[y\mathinner{.\,.} y')) + \delta_E^{\mathcal{A}'}(Y[y\mathinner{.\,.} y'), Z[z\mathinner{.\,.} z')).$$

For an alignment $\mathcal{A} : X[x\mathinner{.\,.} x') \rightsquigarrow Y[y\mathinner{.\,.} y')$ with $\mathcal{A} = (x_i,y_i)_{i=0}^{m}$, we define the *inverse alignment* $\mathcal{A}^{-1} : Y[y\mathinner{.\,.} y') \rightsquigarrow X[x\mathinner{.\,.} x')$ as $\mathcal{A}^{-1} := (y_i,x_i)_{i=0}^{m}$. The inverse alignment satisfies

$$\delta_E^{\mathcal{A}^{-1}}(Y[y\mathinner{.\,.} y'), X[x\mathinner{.\,.} x')) = \delta_E^{\mathcal{A}}(X[x\mathinner{.\,.} x'), Y[y\mathinner{.\,.} y')).$$

Given an alignment $\mathcal{A} : X[x\mathinner{.\,.} x') \rightsquigarrow Y[y\mathinner{.\,.} y')$ and a fragment $X[\bar{x}\mathinner{.\,.} \bar{x}')$ of $X[x\mathinner{.\,.} x')$, we write $\mathcal{A}(X[\bar{x}\mathinner{.\,.} \bar{x}'))$ for the fragment $Y[\bar{y}\mathinner{.\,.} \bar{y}')$ of $Y[y\mathinner{.\,.} y')$ that $\mathcal{A}$ aligns against $X[\bar{x}\mathinner{.\,.} \bar{x}')$. As insertions and deletions may render this definition ambiguous, we formally set

$$\bar{y} := \min\{\hat{y} : (\bar{x},\hat{y}) \in \mathcal{A}\} \quad \text{and} \quad \bar{y}' := \begin{cases} y' & \text{if } \bar{x}' = x', \\ \min\{\hat{y}' : (\bar{x}',\hat{y}') \in \mathcal{A}\} & \text{otherwise.} \end{cases}$$

This particular choice satisfies the following decomposition property.

Fact 3.4 ([CKW22, Fact 2.2]). *For any alignment $\mathcal{A}$ of X onto Y and a decomposition $X = X_1 \cdots X_t$ into t fragments, $Y = \mathcal{A}(X_1)\cdots\mathcal{A}(X_t)$ is a decomposition into t fragments with $\delta_E^{\mathcal{A}}(X,Y) = \sum_{i=1}^{t} \delta_E^{\mathcal{A}}(X_i, \mathcal{A}(X_i))$. Further, if $\mathcal{A}$ is an optimal alignment, then $\delta_E(X,Y) = \sum_{i=1}^{t} \delta_E(X_i,\mathcal{A}(X_i))$.*

We use the following *edit information* notion to encode alignments in space proportional to their costs.

Definition 3.5 (Edit information). *For an alignment $\mathcal{A} = (x_i,y_i)_{i=0}^{m}$ of $X[x\mathinner{.\,.} x')$ onto $Y[y\mathinner{.\,.} y')$, the* edit information *is defined as the set of 4-tuples $\mathsf{E}_{X,Y}(\mathcal{A}) = \{(x_i, \mathsf{cx}_i : y_i, \mathsf{cy}_i) : i \in [0\mathinner{.\,.} m) \text{ and } \mathsf{cx}_i \ne \mathsf{cy}_i\}$, where*

$$\mathsf{cx}_i = \begin{cases} X[x_i] & \textit{if } x_{i+1} = x_i + 1, \\ \varepsilon & \textit{otherwise;} \end{cases} \qquad \textit{and} \qquad \mathsf{cy}_i = \begin{cases} Y[y_i] & \textit{if } y_{i+1} = y_i + 1, \\ \varepsilon & \textit{otherwise.} \end{cases}$$

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
X	a	b	a	c	a	b	c	a	b	c	a	a	a	a	b

Figure 5 The LZ77 factorization of a string $X = \texttt{abacabcabcaaaab}$ of length $n = 15$. The resulting encoding has $z = 8$ elements: $(\texttt{a}, 0)$, $(\texttt{b}, 0)$, $(0, 1)$, $(\texttt{c}, 0)$, $(0, 2)$, $(3, 5)$, $(10, 3)$, $(8, 1)$.

Observe that given two strings X and Y, along with the edit information $\mathsf{E}_{X,Y}(\mathcal{A})$ for an alignment $\mathcal{A} : X \rightsquigarrow Y[\,y \mathinner{.\,.} y'\,)$ of non-zero cost, we are able to fully reconstruct $\mathcal{A}$. This is because elements in $\mathcal{A}$ without corresponding entries in $\mathsf{E}_{X,Y}(\mathcal{A})$ represent matches. Therefore, we can deduce the missing pairs of $\mathcal{A}$ between two consecutive elements in $\mathsf{E}_{X,Y}(\mathcal{A})$, as well as before (or after) the first (or last) element of $\mathsf{E}_{X,Y}(\mathcal{A})$. This inference requires at least one 4-tuple to be contained in $\mathsf{E}_{X,Y}(\mathcal{A})$.

Pattern Matching with Edits. We denote the minimum edit distance between a string S and any prefix of a string T^∞ by $\delta_E(S, T^*) := \min\{\delta_E(S, T^\infty[\,0 \mathinner{.\,.} j\,)) : j \in \mathbb{Z}_{\ge 0}\}$. We denote the minimum edit distance between a string S and any substring of T^∞ by $\delta_E(S, {}^*T^*) := \min\{\delta_E(S, T^\infty[\,i \mathinner{.\,.} j\,)) : i, j \in \mathbb{Z}_{\ge 0} \text{ and } i \le j\}$.

In the context of two strings P (referred to as the pattern) and T (referred to as the text), along with a positive integer k (referred to as the threshold), we say that there is a k-error or k-edits occurrence of P in T at position $i \in [\,0 \mathinner{.\,.} |T|\,]$ if $\delta_E(P, T[\,i \mathinner{.\,.} j\,)) \le k$ holds for some position $j \in [\,i \mathinner{.\,.} |T|\,]$. The set of all starting positions of k-error occurrences of P in T is denoted by $\mathrm{Occ}^E_k(P, T)$; formally, we set

$$\mathrm{Occ}^E_k(P, T) := \{i \in [\,0 \mathinner{.\,.} |T|\,] : \exists_{j \in [\,i \mathinner{.\,.} |T|\,]}, \delta_E(P, T[\,i \mathinner{.\,.} j\,) \le k)\}.$$

We often need to compute $\mathrm{Occ}^E_k(P, T)$. To that end, we use the recent algorithm of Charalampopoulos, Kociumaka, and Wellnitz [CKW22].

Lemma 3.6 ([CKW22, Main Theorem 1])**.** *Let P denote a pattern of length m, let T denote a text of length n, and let $k \in \mathbb{Z}_{\ge 0}$ denote a threshold. Then, there is a (classical) algorithm that computes $\mathrm{Occ}^E_k(P, T)$ in time $\tilde{O}(n + n/m \cdot k^{3.5})$.*

Compression and Lempel–Ziv Factorizations. We say that a fragment $X[\,i \mathinner{.\,.} i + \ell\,)$ is a *previous factor* if it has an earlier occurrence in X, that is, $X[\,i \mathinner{.\,.} i + \ell\,) = X[\,i' \mathinner{.\,.} i' + \ell\,)$ holds for some $i' \in [\,0 \mathinner{.\,.} i\,)$. An *LZ77-like factorization* of X is a factorization $X = F_1 \cdots F_f$ into non-empty *phrases* such that each phrase F_j with $|F_j| > 1$ is a previous factor. In the underlying *LZ77-like representation*, every phrase is encoded as follows (consult Figure 5 for a visualization of an example).

- A previous factor phrase $F_j = X[\,i \mathinner{.\,.} i+\ell\,)$ is encoded as (i', ℓ), where $i' \in [\,0 \mathinner{.\,.} i\,)$ satisfies $X[\,i \mathinner{.\,.} i+\ell\,) = X[\,i' \mathinner{.\,.} i' + \ell\,)$. The position i' is chosen arbitrarily in case of ambiguities.
- Any other phrase $F_j = X[i]$ is encoded as $(X[i], 0)$.

The LZ77 factorization [ZL77] (or the LZ77 parsing) of a string X, denoted by $\mathsf{LZ}(X)$ is an LZ77-like factorization that is constructed by greedily parsing X from left to right into the longest possible phrases. More precisely, the j-th phrase $F_j = X[\,i \mathinner{.\,.} i + \ell\,)$ is the longest previous factor starting at position i; if no previous factor starts at position i, then F_j is the single character $X[i]$. It is known that the aforementioned greedy approach produces the shortest possible LZ77-like factorization.

Self-Edit Distance of a String. Another measure of compressibility that will be instrumental in this paper is the *self-edit distance* of a string, recently introduced by Cassis, Kociumaka, and Wellnitz [CKW23]. An alignment $\mathcal{A} : X \rightsquigarrow X$ is a *self-alignment* if $\mathcal{A}$ does not align any character $X[x]$ to itself. The *self-edit distance* of X, denoted by $\mathsf{self}\text{-}\delta_E(X)$, is the minimum cost of a self-alignment; formally, we set $\mathsf{self}\text{-}\delta_E(X) := \min_{\mathcal{A}} \delta^{\mathcal{A}}_E(X, X)$, where the minimization ranges over all self-alignments $\mathcal{A} : X \rightsquigarrow X$. A small self-edit distance implies that we can efficiently encode the string.

Lemma 3.7. *For any string X, we have $|\mathsf{LZ}(X)| \le 2 \cdot \mathsf{self}\text{-}\delta_E(X)$.*

Proof. Consider an optimal self-alignment $\mathcal{A} : X \rightsquigarrow X$. Without loss of generality, we may assume that each point $(x, y) \in \mathcal{A}$ satisfies $x \ge y$.[2] Let us partition X into individual characters that $\mathcal{A}$ deletes or substitutes, as well as maximal fragments that $\mathcal{A}$ matches perfectly.

Observe that each fragment $X[\,x \mathinner{..} x'\,)$ that $\mathcal{A}$ matches perfectly is a previous factor: indeed, $X[\,x \mathinner{..} x'\,)$ matches $X[\,y \mathinner{..} y'\,)$, and $x > y$ holds because $(x, y) \in \mathcal{A}$, $x \ge y$, and $\mathcal{A}$ does not match $X[x]$ with itself. Consequently, $X[\,x \mathinner{..} x'\,)$ is a previous factor and the partition defined above forms a valid LZ77-like representation.

Since $\mathcal{A}$ makes at most $\mathsf{self}\text{-}\delta_E(X)$ edits and (if $|X| > 0$) this includes a deletion of $X[0]$, the total number of phrases in the partition does not exceed $2 \cdot \mathsf{self}\text{-}\delta_E(X)$. ∎

Note that, since $\mathsf{self}\text{-}\delta_E(X)$ is insensitive to string reversal, we also have $|\mathsf{LZ}(\overline{X})| \le 2 \cdot \mathsf{self}\text{-}\delta_E(X)$, where $\overline{X} = X[|X| - 1] \cdots X[1]X[0]$ is the reversal of X. We use the following known properties of $\mathsf{self}\text{-}\delta_E$.

Lemma 3.8 (Properties of $\mathsf{self}\text{-}\delta_E$, [CKW23, Lemma 4.2])**.** *For any string X, all of the following hold:*
Monotonicity. *For any $\ell' \le \ell \le r \le r' \in [\,0 \mathinner{..} |X|\,]$, we have $\mathsf{self}\text{-}\delta_E(X[\,\ell \mathinner{..} r\,)) \le \mathsf{self}\text{-}\delta_E(X[\,\ell' \mathinner{..} r'\,))$.*
Sub-additivity. *For any $m \in [\,0 \mathinner{..} |X|\,]$, we have $\mathsf{self}\text{-}\delta_E(X) \le \mathsf{self}\text{-}\delta_E(X[\,0 \mathinner{..} m\,)) + \mathsf{self}\text{-}\delta_E(X[\,m \mathinner{..} |X|\,))$.*
Triangle inequality. *For any string Y, we have $\mathsf{self}\text{-}\delta_E(Y) \le \mathsf{self}\text{-}\delta_E(X) + 2\delta_E(X, Y)$.* ∎

Cassis, Kociumaka, and Wellnitz [CKW23] also proved the following lemma that bounds the self-edit distance of Y in the presence of two disjoint alignments mapping Y to nearby fragments of another string X.

Lemma 3.9 ([CKW23, Lemma 4.5])**.** *Consider strings X, Y and alignments $\mathcal{A} : Y \rightsquigarrow X[\,i \mathinner{..} j\,)$ and $\mathcal{A}' : Y \rightsquigarrow X[\,i' \mathinner{..} j'\,)$. If there is no $(y, x) \in \mathcal{A} \cap \mathcal{A}'$ such that both $\mathcal{A}$ and $\mathcal{A}'$ match $Y[y]$ with $X[x]$, then*

$$\mathsf{self}\text{-}\delta_E(Y) \le |i - i'| + \delta_E^{\mathcal{A}}(Y, X[\,i \mathinner{..} j\,)) + \delta_E^{\mathcal{A}'}(Y, X[\,i' \mathinner{..} j'\,)) + |j - j'|.$$ ∎

4 New Combinatorial Insights for Pattern Matching with Edits

In this section, we fix a threshold k and two strings P and T over a common input alphabet Σ. Furthermore, we let S be a set of alignments of P onto fragments of T such that all alignments in S have cost at most k.

4.1 The Periodic Structure induced by S

We now formally define the concepts introduced in the Technical Overview. Again, revisit Figure 4 for a visualization of an example.

Definition 4.1. *We define the undirected graph $\mathbf{G}_S = (V, E)$ as follows.*
The vertex set V *contains:*
- *$|P|$ vertices representing characters of P;*
- *$|T|$ vertices representing characters of T; and*
- *one special vertex $\perp$.*

The edge set E *contains the following edges for each alignment $\mathcal{X} \in S$:*

---

[2] If this is not the case, we write $\mathcal{A} = (x_t, y_t)_{t=0}^{m}$ and observe that $\mathcal{A}' = (\max(x_t, y_t), \min(x_t, y_t))_{t=0}^{m}$ is a self-alignment of the same cost. Geometrically, this means that we replace parts of $\mathcal{A}$ below the main diagonal with their mirror image.

(i) $\{P[x], \bot\}$ *for every character* $P[x]$ *that* $\mathcal{X}$ *deletes;*
(ii) $\{\bot, T[y]\}$ *for every character* $T[y]$ *that* $\mathcal{X}$ *inserts;*
(iii) $\{P[x], T[y]\}$ *for every pair of characters* $P[x]$ *and* $T[y]$ *that* $\mathcal{X}$ *aligns.*

We say that an edge $\{P[x], T[y]\}$ *is* black *if* $\mathcal{X}$ *matches* $P[x]$ *and* $T[y]$*; all the remaining edges are* red.

We say that a connected component of $\mathbf{G}_S$ *is* red *if it contains at least one red edge; otherwise, we say that the connected component is* black. *We denote with* $\mathrm{bc}(\mathbf{G}_S)$ *the number of black components in* $\mathbf{G}_S$.

Note that all vertices contained in black components correspond to characters of P and T. Moreover, all characters of a single black component are the same—a remarkable property. This is because the presence of a black edge indicates that some alignment in S matches the two corresponding characters. Hence, the two characters must be equal. Since a black component contains only black edges, all characters contained in one such component are equal.

Suppose we know the edit information $\mathsf{E}_{P,T}(\mathcal{X})$ for each alignment $\mathcal{X} : P \rightsquigarrow T[t\mathinner{.\,.} t')$ in S. Then, we can reconstruct the complete edge set of $\mathbf{G}_S$, without needing any other information. Moreover, we can also distinguish between red and black edges, and consequently between red and black components.

Next, we introduce the property that we assume about S. This property induces the periodic structure in P and T at the core of our combinatorial results.

Definition 4.2. *We say* S encloses T *if* $|T| \le 2\cdot|P| - 2k$ *and there exist two distinct alignments* $\mathcal{X}_{\mathsf{pref}}, \mathcal{X}_{\mathsf{suf}} \in S$ *such that* $\mathcal{X}_{\mathsf{pref}}$ *aligns* P *with a prefix of* T *and* $\mathcal{X}_{\mathsf{suf}}$ *aligns* P *with a suffix of* T*, or equivalently* $(0,0) \in \mathcal{X}_{\mathsf{pref}}$ *and* $(|P|,|T|) \in \mathcal{X}_{\mathsf{suf}}$.

Suppose that $\mathrm{bc}(\mathbf{G}_S) = 0$. Then, we want to argue that the information $\{\mathsf{E}_{P,T}(\mathcal{X}) : S \ni \mathcal{X} : P \rightsquigarrow T[t\mathinner{.\,.} t')\}$ mentioned earlier is sufficient to fully retrieve P and T. As already argued before, the information is sufficient to fully reconstruct $\mathbf{G}_S$. From $\mathrm{bc}(\mathbf{G}_S) = 0$ follows that every character of P and T lies in a red component. As a consequence, for every character of P and T, there exists a path (possibly of length zero) starting at that character, only containing black edges, and ending in a character incident to a red edge. Since the path only contains black edges, all characters contained in the path must be equal. Since we store all characters incident to red edges, we can retrieve the character at the starting point of the path. We conclude that we can fully retrieve P and T. Note that encoding this information occupies $O(k|S|)$ space, since, for each alignment in S, we can store the corresponding edit information using $O(k)$ space.

We dedicate the remaining part of Section 4 until Section 4.6 to the case $\mathrm{bc}(\mathbf{G}_S) > 0$. We will prove which information, other than the edit information, we must store to encode all k-edit occurrences. By storing the edit information for each $\mathcal{X} \in S$, we will be always able to fully reconstruct $\mathbf{G}_S$ and to infer which characters are contained in red components. Hence, the only additional information left to store in this case, are the characters contained in black components.

Throughout the remaining part of Section 4 until Section 4.6, we will always assume that S encloses T and that $\mathrm{bc}(\mathbf{G}_S) > 0$. This is where we will need to exploit the periodic structure induced by S. Before describing more formally the periodic structure induced by S, we need to introduce two additional strings $T_{|S}$ and $P_{|S}$, constructed by retaining characters contained in black connected components.

Definition 4.3. *We let* $T_{|S}$ *denote the subsequence of* T *consisting of characters contained in black components of* $\mathbf{G}_S$*. Similarly,* $P_{|S}$ *denotes the subsequence of* P *consisting of characters contained in black components of* $\mathbf{G}_S$.

Now, we prove the lemma that characterizes the periodic structure induced by S.

Lemma 4.4. *For every $c \in [0\,..\,\mathrm{bc}(\mathbf{G}_S))$, there exists a black connected component with node set*

$$\{P_{|S}[i] : i \equiv_{\mathrm{bc}(\mathbf{G}_S)} c\} \cup \{T_{|S}[i] : i \equiv_{\mathrm{bc}(\mathbf{G}_S)} c\},$$

i.e., there exists a black connected component containing all characters of $P_{|S}$ and $T_{|S}$ appearing at positions congruent to c modulo $\mathrm{bc}(\mathbf{G}_S)$. Moreover, the last characters of $P_{|S}$ and $T_{|S}$ are contained in the same black connected component, that is, $|T_{|S}| \equiv_{\mathrm{bc}(\mathbf{G}_S)} |P_{|S}|$.

Proof. The following claim characterizes the edges of $\mathbf{G}_S$ induced by a single alignment $\mathcal{X} \in S$. The crucial observation is that these edges correspond to an *exact occurrence of $P_{|S}$ in $T_{|S}$*.

Claim 4.5. *For every $\mathcal{X} \in S$, there exists $\Delta_{\mathcal{X}} \in [0\,..\,|T_{|S}| - |P_{|S}|]$ such that $\mathcal{X}$ induces edges between $P_{|S}[p]$ and $T_{|S}[p + \Delta_{\mathcal{X}}]$ for all $p \in [0\,..\,|P_{|S}|)$ and no other edges incident to characters of $P_{|S}$ or $T_{|S}$.*

Proof. By Definition 4.1, black edges induced by $\mathcal{X}$ form a matching between $P_{|S}$ and the characters of $T_{|S}$ contained in the image $\mathcal{X}(P)$. The characters contained in $\mathcal{X}(P)$ appear consecutively in $T_{|S}$, so $\mathcal{X}$ induces a matching between $P_{|S}$ and a length-$|P_{|S}|$ fragment of $T_{|S}$. Every such fragment is of the form $T_{|S}[\Delta_{\mathcal{X}}\,..\,\Delta_{\mathcal{X}} + |P_{|S}|)$ for some $\Delta_{\mathcal{X}} \in [0\,..\,|T_{|S}| - |P_{|S}|]$. Moreover, $\mathcal{X}$ is non-crossing and every edge induced by $\mathcal{X}$ corresponds to an element of $\mathcal{X}$,so the matching induced by $\mathcal{X}$ consists of edges between $P_{|S}[p]$ and $T_{|S}[p + \Delta_{\mathcal{X}}]$ for every $p \in [0\,..\,|P_{|S}|)$.

Next, we explore the consequences of the assumption that S encloses T.

Claim 4.6. *There exist alignments $\mathcal{X}_{\mathsf{pref}}, \mathcal{X}_{\mathsf{suf}} \in S$ such that $\Delta_{\mathcal{X}_{\mathsf{pref}}} = 0$ and $\Delta_{\mathcal{X}_{\mathsf{suf}}} = |T_{|S}| - |P_{|S}|$. Moreover, $|T_{|S}| \leq 2|P_{|S}|$.*

Proof. Since S encloses T, there are alignments $\mathcal{X}_{\mathsf{pref}}, \mathcal{X}_{\mathsf{suf}} \in S$ such that $(0,0) \in \mathcal{X}_{\mathsf{pref}}$ and $(|P|,|T|) \in \mathcal{X}_{\mathsf{suf}}$, that is, $\mathcal{X}_{\mathsf{pref}}(P)$ is a prefix of T whereas $\mathcal{X}_{\mathsf{suf}}(P)$ is a suffix of T. Every alignment $\mathcal{X} \in S$ induces a matching between $P_{|S}$ and the characters of $T_{|S}$ contained in the image $\mathcal{X}(P)$, so we must have $\Delta_{\mathcal{X}_{\mathsf{pref}}} = 0$ and $\Delta_{\mathcal{X}_{\mathsf{suf}}} = |T_{|S}| - |P_{|S}|$. The costs of $\mathcal{X}_{\mathsf{pref}}(P)$ and $\mathcal{X}_{\mathsf{suf}}(P)$ are at most k, so $|\mathcal{X}_{\mathsf{pref}}(P)| \geq |P| - k$ and $|\mathcal{X}_{\mathsf{suf}}(P)| \geq |P| - k$. Due to the assumption $|T| \leq 2|P| - 2k$, this means that every character of T is contained in $\mathcal{X}_{\mathsf{pref}}(P)$ or $\mathcal{X}_{\mathsf{suf}}(P)$. Consequently, the two matchings induced by these alignments jointly cover $T_{|S}$, and thus $|T_{|S}| \leq 2|P_{|S}|$.

Let us assign a unique label $\$_C$ to each black component C of $\mathbf{G}_S$ and define strings P_{bc} and T_{bc} of length $|P_{|S}|$ and $|T_{|S}|$, respectively, as follows: For $i \in [0\,..\,|P_{|S}|)$, set $P_{\mathrm{bc}}[i] = \$_C$, where C is black component containing $P_{|S}[i]$. Similarly, for $i \in [0\,..\,|T_{|S}|)$, set $T_{\mathrm{bc}}[i] = \$_C$, where C is the black connect component containing $T_{|S}[i]$.

By Claims 4.5 and 4.6, $\{0, |T_{\mathrm{bc}}| - |P_{\mathrm{bc}}|\} \subseteq \{\Delta_{\mathcal{X}} : \mathcal{X} \in S\} \subseteq \mathrm{Occ}(P_{\mathrm{bc}}, T_{\mathrm{bc}})$. Due to $|T_{\mathrm{bc}}| \leq 2|P_{\mathrm{bc}}|$, Lemma 3.2 implies that $\gcd(\mathrm{Occ}(P_{\mathrm{bc}}, T_{\mathrm{bc}}))$ is a period of T_{bc}, and thus also $g := \gcd\{\Delta_{\mathcal{X}} : \mathcal{X} \in S\}$ is a period of both T_{bc} and of its prefix P_{bc}. Consequently, for each $c \in [0\,..\,g)$, the set $C_c := \{P_{|S}[i] : i \equiv_g c\} \cup \{T_{|S}[i] : i \equiv_g c\}$ belongs to a single connected component of $\mathbf{G}_S$. It remains to prove $g = \mathrm{bc}(\mathbf{G}_S)$, and we do this by demonstrating that no edge of $\mathbf{G}_S$ leaves C_c. Note that Claim 4.5 further implies that every edge incident to $P_{|S}$ or $T_{|S}$ connects $P_{|S}[p]$ with $T_{|S}[t]$ such that $t = p + \Delta_{\mathcal{X}}$ for some $\mathcal{X} \in S$. In particular, $t \equiv_g p$, so $P_{|S}[p] \in C_c$ holds if and only if $T_{|S}[t] \in C_c$.

Lastly, since P_{bc} is a suffix of T_{bc}, the last characters of $P_{|S}$ and $T_{|S}$ are in the same black component.

For the sake of convenience, we index black connected components with integers in $[0\,..\,\mathrm{bc}(\mathbf{G}_S))$ and introduce some notation:

Definition 4.7. *For $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$, we define the c-th* black connected component *as the black connected component containing $P_{|S}[c]$ and set*

$$m_c := \left\lceil \frac{|P_{|S}| - c}{\mathrm{bc}(\mathbf{G}_S)} \right\rceil \quad \text{and} \quad n_c := \left\lceil \frac{|T_{|S}| - c}{\mathrm{bc}(\mathbf{G}_S)} \right\rceil,$$

as the number of characters in P and T, respectively, belonging to the c-th black connected component. Furthermore:

- *We define $c_{\mathrm{last}} \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ as the black component containing the last characters of $P_{|S}$ and $T_{|S}$.*
- *For $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ and $j \in [\,0\,..\,m_c\,)$ we define $\pi^c_j \in [\,0\,..\,|P|\,)$ as the position of $P_{|S}[c + j \cdot \mathrm{bc}(\mathbf{G}_S)]$ in P.*
- *For $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ and $i \in [\,0\,..\,n_c\,)$, we define $\tau^c_i \in [\,0\,..\,|T|\,)$ as the position of $T_{|S}[c + i \cdot \mathrm{bc}(\mathbf{G}_S)]$ in T.*
- *For sake of convenience, we denote $m_{\mathrm{bc}(\mathbf{G}_S)} = m_0 - 1$ and $\pi^{\mathrm{bc}(\mathbf{G}_S)}_j = \pi^0_{j+1}$ for $j \in [\,0\,..\,m_{\mathrm{bc}(\mathbf{G}_S)}\,)$. Similarly, we denote $n_{\mathrm{bc}(\mathbf{G}_S)} = n_0 - 1$ and $\tau^{\mathrm{bc}(\mathbf{G}_S)}_i = \tau^0_{i+1}$ for $i \in [\,0\,..\,n_{\mathrm{bc}(\mathbf{G}_S)}\,)$.*

Remark 4.8. We conclude this (sub)section with some observations.

(1) Consider $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$. Notice that the characters $\{T[\tau^c_i]\}_{i=0}^{n_c-1}$ and $\{P[\pi^c_j]\}_{j=0}^{m_c-1}$ are precisely those within the c-th black connected component. Consequently, they are all identical.

(2) For all $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,]$, we have $n_c \in \{n_0, n_0 - 1\}$ and $m_c \in \{m_0, m_0 - 1\}$.

(3) The indices $\{\tau^c_i\}_{i,c}$ induce a partition of T into (possibly empty) substrings. This partition consists of: the initial fragment $T[\,0\,..\,\tau^0_0\,)$, all the intermediate fragments $T[\,\tau^c_i\,..\,\tau^{c+1}_i\,)$ for $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ and $i \in [\,0\,..\,n_{c+1}\,)$, and the final fragment $T[\,\tau^{c_{\mathrm{last}}}_{n_0-1}\,..\,|T|\,)$. Similarly, the indices $\{\pi^c_j\}_{j,c}$ induce a partition of P.

(4) Recall that $\mathbf{G}_S$ contains an edge between $P[\pi^c_j]$ and $T[\tau^c_i]$ only if there is an alignment $\mathcal{X} \in S$ such that $(\pi^c_j, \tau^c_i) \in \mathcal{X}$. If $j < m_{c+1}$ and $i < n_{c+1}$, we also have $(\pi^{c+1}_j, \tau^{c+1}_i) \in \mathcal{X}$. For such (π^c_j, τ^c_i), we can associate with it at least one value $\delta^{\mathcal{X}}_E(P[\,\pi^c_j\,..\,\pi^{c+1}_j\,), T[\,\tau^c_i\,..\,\tau^{c+1}_i\,))$, noting that there might be multiple values associated with (π^c_j, τ^c_i).

4.2 Covering Weight Functions

Definition 4.9. *We say that a* weight function $\mathrm{w}_S : [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,) \to \mathbb{Z}_{\geq 0}$ covers S *if all of the following conditions hold:*

(1) *for all $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$, $j \in [\,0\,..\,m_{c+1}\,)$, and $i \in [\,0\,..\,n_{c+1}\,)$, we have*

$$\mathrm{w}_S(c) \geq \delta_E(P[\,\pi^c_j\,..\,\pi^{c+1}_j\,), T[\,\tau^c_i\,..\,\tau^{c+1}_i\,)); \tag{1}$$

(2) $\mathrm{w}_S(\mathrm{bc}(\mathbf{G}_S) - 1) \geq \delta_E(P[\,0\,..\,\pi^0_0\,), T[\,0\,..\,\tau^0_0\,))$;

(3) $\mathrm{w}_S(\mathrm{bc}(\mathbf{G}_S) - 1) \geq \delta_E(P[\,0\,..\,\pi^0_0\,), T[\,t\,..\,\tau^0_i\,))$ *holds for every $i \in [\,1\,..\,n_0\,)$ and some $t \in [\,\tau^{\mathrm{bc}(\mathbf{G}_S)-1}_{i-1}\,..\,\tau^0_i\,]$;*

(4) $\mathrm{w}_S(c_{\mathrm{last}}) \geq \delta_E(P[\,\pi^{c_{\mathrm{last}}}_{m_0-1}\,..\,|P|\,), T[\,\tau^{c_{\mathrm{last}}}_{n_0-1}\,..\,|T|\,))$; *and*

(5) $\mathrm{w}_S(c_{\mathrm{last}}) \geq \delta_E(P[\,\pi^{c_{\mathrm{last}}}_{m_0-1}\,..\,|P|\,), T[\,\tau^{c_{\mathrm{last}}}_i\,..\,t'\,))$ *holds for every $i \in [\,0\,..\,n_0 - 1\,)$ and some $t' \in [\,\tau^{c_{\mathrm{last}}}_i\,..\,\tau^{c_{\mathrm{last}}+1}_i\,]$.*

We define $\sum_{c=0}^{\mathrm{bc}(\mathbf{G}_S)-1} \mathrm{w}_S(c)$ to be the total weight *of* w_S.

Lemma 4.10. *There exists a weight function $\mathrm{w}_S : [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,) \to \mathbb{Z}_{\geq 0}$ that covers S of total weight at most $O(k|S|)$.*

Proof. Let $\bar{\mathbf{G}}_S$ be the subgraph of $\mathbf{G}_S$ induced by all vertices $P[\pi^c_j]$ and $T[\tau^c_i]$ for $c \in [0\,..\,\mathrm{bc}(\mathbf{G}_S))$, $j \in [0\,..\,m_{c+1})$, and $i \in [0\,..\,n_{c+1})$. In other words, we remove all red components as well as vertices $P[\pi^{c_{\mathrm{last}}}_{m_0-1}]$ and $T[\tau^{c_{\mathrm{last}}}_{n_0-1}]$ from the c_{last}-th black component. By doing so, the edges of $\bar{\mathbf{G}}_S$ will be exactly those for which the association observed in Remark 4.8(4) is well defined. We prove that $\bar{\mathbf{G}}_S$ preserves the same connectedness structure of black components as $\mathbf{G}_S$.

Claim 4.11. *For each $c \in [0\,..\,\mathrm{bc}(\mathbf{G}_S))$, the graph $\bar{\mathbf{G}}_S$ contains a connected component $\bar{\mathbf{G}}^c_S$ induced by vertices $P[\pi^c_j]$ and $T[\tau^c_i]$ with $j \in [0\,..\,m_{c+1})$ and $i \in [0\,..\,n_{c+1})$.*

Proof. Since $\bar{\mathbf{G}}_S$ is a subgraph of $\mathbf{G}_S$, the characterization of connected components of $\mathbf{G}_S$ in Lemma 4.4 implies that $\bar{\mathbf{G}}_S$ does not contain any edge leaving $\bar{\mathbf{G}}^c_S$. Thus, it suffices to prove that $\bar{\mathbf{G}}^c_S$ is connected.

For this, let us define strings $P_{\bar{\mathrm{bc}}}$ and $T_{\bar{\mathrm{bc}}}$ analogously to P_{bc} and T_{bc} in the proof of Lemma 4.4, but with respect to $\bar{\mathbf{G}}_S$ rather than $\mathbf{G}_S$. Specifically, $|P_{\bar{\mathrm{bc}}}| = |P_{|S}| - 1$ and $|T_{\bar{\mathrm{bc}}}| = |T_{|S}| - 1$, and the characters $P_{\bar{\mathrm{bc}}}[j]$ and $T_{\bar{\mathrm{bc}}}[i]$ are equal if and only if the corresponding characters $P_{|S}[j]$ and $T_{|S}[i]$ belong to the same connected component of $\bar{\mathbf{G}}_S$.

By Claims 4.5 and 4.6, we have $\{0, |T_{\bar{\mathrm{bc}}}| - |P_{\bar{\mathrm{bc}}}|\} \subseteq \{\Delta_{\mathcal{X}} : \mathcal{X} \in S\} \subseteq \mathrm{Occ}(P_{\bar{\mathrm{bc}}}, T_{\bar{\mathrm{bc}}})$ and $|T_{\bar{\mathrm{bc}}}| = |T_{|S}| - 1 \le 2|P_{|S}| - 1 = 2|P_{\bar{\mathrm{bc}}}| + 1$. If $|P_{\bar{\mathrm{bc}}}| = 0$, then $|T_{\bar{\mathrm{bc}}}| \le 1$, and $\bar{\mathbf{G}}_S$ is either empty or consists of a single vertex constituting the connected component $\bar{\mathbf{G}}^0_S$. Otherwise, Lemma 3.2 implies that $\gcd(\mathrm{Occ}(P_{\bar{\mathrm{bc}}}, T_{\bar{\mathrm{bc}}}))$ is a period of T_{bc}, and thus also $\mathrm{bc}(\mathbf{G}_S) = \gcd\{\Delta_{\mathcal{X}} : \mathcal{X} \in S\}$ is a period of both $T_{\bar{\mathrm{bc}}}$ and of its prefix $P_{\bar{\mathrm{bc}}}$. Consequently, $\bar{\mathbf{G}}^c_S$ is a connected subgraph of $\bar{\mathbf{G}}_S$.

Let us also assign weights to edges of $\bar{\mathbf{G}}_S$. If $\bar{\mathbf{G}}_S$ contains an edge between $P[\pi^c_j]$ and $T[\tau^c_i]$, we define its weight to be the smallest value $\delta^{\mathcal{X}}_E(P[\pi^c_j\,..\,\pi^{c+1}_j), T[\tau^c_i\,..\,\tau^{c+1}_i))$ such that $\mathcal{X} \in S$ and $(\pi^c_j, \tau^c_i) \in \mathcal{X}$. By $\mathrm{w}(\bar{\mathbf{G}}^c_S)$, we denote the total weight of edges in $\bar{\mathbf{G}}^c_S$.

Claim 4.12. *If $\mathrm{w}_S(c) \ge \mathrm{w}(\bar{\mathbf{G}}^c_S)$ for all $c \in [0\,..\,\mathrm{bc}(\mathbf{G}_S))$, then property (1) holds.*

Proof. By Claim 4.11, $\bar{\mathbf{G}}^c_S$ is connected for all $c \in [0\,..\,\mathrm{bc}(\mathbf{G}_S))$. Thus, for any $c \in [0\,..\,\mathrm{bc}(\mathbf{G}_S))$, $j \in [0\,..\,m_{c+1})$, $i \in [0\,..\,n_{c+1})$, we can construct an alignment of cost at most $\mathrm{w}(\bar{\mathbf{G}}^c_S)$ aligning $P[\pi^c_j\,..\,\pi^{c+1}_j)$ onto $T[\tau^c_i\,..\,\tau^{c+1}_i)$. To obtain such alignment, we compose the alignments inducing the weights on the edges contained in the path in $\bar{\mathbf{G}}^c_S$ from $P[\pi^c_j]$ to $T[\tau^c_i]$.

Now, we formally define w_S. We set $\mathrm{w}_S(c) = \mathrm{w}(\bar{\mathbf{G}}^c_S)$ for all $c \in [0\,..\,\mathrm{bc}(\mathbf{G}_S)) \setminus \{c_{\mathrm{last}}, \mathrm{bc}(\mathbf{G}_S) - 1\}$. Furthermore, let $\alpha, \alpha' \in \mathbb{Z}_{\ge 0}$ be determined later. If $\mathrm{bc}(\mathbf{G}_S) - 1 \ne c_{\mathrm{last}}$, then we set $\mathrm{w}_S(\mathrm{bc}(\mathbf{G}_S) - 1) = \mathrm{w}(\bar{\mathbf{G}}^{\mathrm{bc}(\mathbf{G}_S)-1}_S) + \alpha$ and $\mathrm{w}_S(c_{\mathrm{last}}) = \mathrm{w}(\bar{\mathbf{G}}^{c_{\mathrm{last}}}_S) + \alpha'$. Otherwise, if $\mathrm{bc}(\mathbf{G}_S) - 1 = c_{\mathrm{last}}$, set $\mathrm{w}_S(\mathrm{bc}(\mathbf{G}_S) - 1) = \mathrm{w}(\bar{\mathbf{G}}^{\mathrm{bc}(\mathbf{G}_S)-1}_S) + \alpha + \alpha'$. From $\alpha, \alpha' \ge 0$ and Claim 4.12, follows that property (1) holds. We will choose α, α' large enough to ensure that also properties (2), (3), (4), and (5) hold. For that purpose, consider alignments $\mathcal{X}_{\mathsf{pref}}, \mathcal{X}_{\mathsf{suf}} \in S$ such that $(0,0) \in \mathcal{X}_{\mathsf{pref}}$ and $(|P|, |T|) \in \mathcal{X}_{\mathsf{suf}}$. First, let us define α as $\alpha := \delta^{\mathcal{X}_{\mathsf{pref}}}_E(P[0\,..\,\pi^0_0), T[0\,..\,\tau^0_0)) + \delta^{\mathcal{X}_{\mathsf{suf}}}_E(P[0\,..\,\pi^0_0), T[y\,..\,\tau^0_{\hat{\imath}}))$, assuming $\mathcal{X}_{\mathsf{suf}}$ aligns $P[0\,..\,\pi^0_0)$ onto $T[y\,..\,\tau^0_{\hat{\imath}})$ for some $\hat{\imath} \in [0\,..\,n_0)$ and $y \in [\tau^{\mathrm{bc}(\mathbf{G}_S)-1}_{\hat{\imath}-1}\,..\,\tau^0_{\hat{\imath}}]$.

Claim 4.13. *Properties (2), (3) hold.*

Proof. Clearly, property (2) holds because

$$\mathrm{w}_S(\mathrm{bc}(\mathbf{G}_S) - 1) \ge \alpha \ge \delta^{\mathcal{X}_{\mathsf{pref}}}_E(P[0\,..\,\pi^0_0), T[0\,..\,\tau^0_0)) \ge \delta_E(P[0\,..\,\pi^0_0), T[0\,..\,\tau^0_0)).$$

Regarding property (3), consider the case where $(\pi^0_0, \tau^0_0) \in \mathcal{X}_{\mathsf{suf}}$, i.e., $\hat{\imath} = 0$. This implies $n_0 = 1$, as all other alignments in S also align $P[\pi^0_0]$ with $T[\tau^0_0]$. In this case, there is nothing to prove, as the quantifier

i in property (3) ranges over an empty set. It remains to cover the case when $\hat{\imath} \in [1 \mathinner{.\,.} n_0)$. Choose an arbitrary $i \in [1 \mathinner{.\,.} n_0)$. Since $\tau_{\hat{\imath}-1}^{\mathrm{bc}(\mathbf{G}_S)-1}$ and $\tau_{i-1}^{\mathrm{bc}(\mathbf{G}_S)-1}$ belong to the same connected component $\bar{\mathbf{G}}_S^{\mathrm{bc}(\mathbf{G}_S)-1}$, we can employ a similar argument as in Claim 4.12. Thus, there exists an alignment $\mathcal{Y}$ with a cost of at most $\mathrm{w}(\bar{\mathbf{G}}_S^{\mathrm{bc}(\mathbf{G}_S)-1})$, aligning $T[\tau_{\hat{\imath}-1}^{\mathrm{bc}(\mathbf{G}_S)-1} \mathinner{.\,.} \tau_{\hat{\imath}}^0)$ to $T[\tau_{i-1}^{\mathrm{bc}(\mathbf{G}_S)-1} \mathinner{.\,.} \tau_i^0)$. Assume $\mathcal{Y}(T[y \mathinner{.\,.} \tau_{\hat{\imath}}^0)) = T[t \mathinner{.\,.} \tau_i^0)$ for some $t \in [\tau_{i-1}^{\mathrm{bc}(\mathbf{G}_S)-1} \mathinner{.\,.} \tau_i^0]$. By composing $\mathcal{X}_{\mathsf{suf}} : P[0 \mathinner{.\,.} \pi_0^0) \rightsquigarrow T[y \mathinner{.\,.} \tau_{\hat{\imath}}^0)$ and $\mathcal{Y} : T[y \mathinner{.\,.} \tau_{\hat{\imath}}^0) \rightsquigarrow T[t \mathinner{.\,.} \tau_i^0)$, we obtain an alignment $P[0 \mathinner{.\,.} \pi_0^0) \rightsquigarrow T[t \mathinner{.\,.} \tau_i^0)$ of a cost of at most

$$\delta_E^{\mathcal{X}_{\mathsf{suf}}}(P[0 \mathinner{.\,.} \pi_0^0), T[t \mathinner{.\,.} \tau_{\hat{\imath}}^0)) + \mathrm{w}(\bar{\mathbf{G}}_S^{\mathrm{bc}(\mathbf{G}_S)-1}) \le \alpha + \mathrm{w}(\bar{\mathbf{G}}_S^{\mathrm{bc}(\mathbf{G}_S)-1}) = \mathrm{w}_S(\mathrm{bc}(\mathbf{G}_S) - 1). \quad \lrcorner$$

Symmetrically, we set $\alpha' := \delta_E^{\mathcal{X}_{\mathsf{suf}}}(P[\pi_{m_0-1}^{c_{\mathrm{last}}} \mathinner{.\,.} |P|), T[\tau_{n_0-1}^{c_{\mathrm{last}}} \mathinner{.\,.} |T|)) + \delta_E^{\mathcal{X}_{\mathsf{pref}}}(P[\pi_{m_0-1}^{c_{\mathrm{last}}} \mathinner{.\,.} |P|), T[\tau_{\hat{\imath}'}^{c_{\mathrm{last}}} \mathinner{.\,.} y'))$, assuming $\mathcal{X}_{\mathsf{pref}}$ aligns $P[\pi_{m_0-1}^{c_{\mathrm{last}}} \mathinner{.\,.} |P|)$ onto $T[\tau_{\hat{\imath}'}^{c_{\mathrm{last}}} \mathinner{.\,.} y')$ for some $\hat{\imath}' \in [0 \mathinner{.\,.} n_0)$ and $y' \in [\tau_{\hat{\imath}'}^{c_{\mathrm{last}}} \mathinner{.\,.} \tau_{\hat{\imath}'+1}^{c_{\mathrm{last}}}]$. Using a symmetric argument to Claim 4.13, we obtain that also properties (4) and (5) hold.

Finally, we prove that the total cost of w_S is at most $k|S|$. The total cost of w_S is the sum of the cost of partial alignments from S. More specifically, the total cost of w_S is the sum of the following terms:

- $\sum_{c=0}^{\mathrm{bc}(\mathbf{G}_S)-1} \mathrm{w}(\bar{\mathbf{G}}_S^c)$, where each of the $\mathrm{w}(\bar{\mathbf{G}}_S^c)$ is the sum of some $\delta_E^{\mathcal{X}}(P[\pi_j^c \mathinner{.\,.} \pi_j^{c+1}), T[\tau_i^c \mathinner{.\,.} \tau_i^{c+1}))$ for some distinct triplets $(\mathcal{X}, i, j)$ such that $\mathcal{X} \in S$, $j \in [0 \mathinner{.\,.} m_{c+1})$ and $i \in [0 \mathinner{.\,.} n_{c+1})$;
- $\delta_E^{\mathcal{X}_{\mathsf{pref}}}(P[0 \mathinner{.\,.} \pi_0^0), T[0 \mathinner{.\,.} \tau_0^0))$;
- $\delta_E^{\mathcal{X}_{\mathsf{suf}}}(P[0 \mathinner{.\,.} \pi_0^0), T[y \mathinner{.\,.} \tau_{\hat{\imath}}^0))$;
- $\delta_E^{\mathcal{X}_{\mathsf{suf}}}(P[\pi_{m_0-1}^{c_{\mathrm{last}}} \mathinner{.\,.} |P|), T[\tau_{n_0-1}^{c_{\mathrm{last}}} \mathinner{.\,.} |T|))$; and
- $\delta_E^{\mathcal{X}_{\mathsf{pref}}}(P[\pi_{m_0-1}^{c_{\mathrm{last}}} \mathinner{.\,.} |P|), T[\tau_{\hat{\imath}'}^{c_{\mathrm{last}}} \mathinner{.\,.} y'))$.

Since all alignments that are (possibly) involved in the sum are disjoint and are partial alignments that belong to S, we conclude that the total cost of w_S is at most $k|S|$. ▪

In the remaining part of Section 4, we let w_S denote a *weight function* that covers S of total weight at most w. We do not restrict ourselves to the case $w = \mathcal{O}(k|S|)$ to make the following results more general.

4.3 Period Covers

In the following subsection, we formalize the characters contained in the black components that we need to learn. We specify this through a subset of indices $C_S \subseteq [0 \mathinner{.\,.} \mathrm{bc}(\mathbf{G}_S))$ of black components, which we will refer to as *period cover*.

Definition 4.14. *A set $C_S \subseteq [0 \mathinner{.\,.} \mathrm{bc}(\mathbf{G}_S))$ is a* period cover *with respect to w_S if $[a \mathinner{.\,.} b] \subseteq C_S$ holds for every interval $[a \mathinner{.\,.} b] \subseteq [0 \mathinner{.\,.} \mathrm{bc}(\mathbf{G}_S))$ such that at least one of the following holds:*

(1) $\mathsf{self}\text{-}\delta_E(T[\tau_0^a \mathinner{.\,.} \tau_0^b]) \le 6w + 11k$ *and* $a = 0$;
(2) $\mathsf{self}\text{-}\delta_E(T[\tau_0^a \mathinner{.\,.} \tau_0^b]) \le 6w + 11k$ *and* $b = \mathrm{bc}(\mathbf{G}_S) - 1$;
(3) $\mathsf{self}\text{-}\delta_E(T[\tau_0^a \mathinner{.\,.} \tau_0^b]) \le 6w + 11k$ *and* $b = c_{\mathrm{last}}$;
(4) $\mathsf{self}\text{-}\delta_E(T[\tau_0^a \mathinner{.\,.} \tau_0^b]) \le 6w + 11k$ *and* $a = c_{\mathrm{last}} + 1$; *or*
(5) $\mathsf{self}\text{-}\delta_E(T[\tau_0^a \mathinner{.\,.} \tau_0^b]) \le 6 \sum_{c=a-1}^{b} \mathrm{w}_S(c)$. ▪

In the remainder of this (sub)section, we illustrate that by constructing a period cover C_S in two different ways, we can effectively (w.r.t. parameters w, k and $|S|$) encode the information $\{(c, T[\tau_0^c]) : c \in C_S\}$.

First, we establish that this is possible if we construct C_S by straightforwardly verifying whether all intervals $[a \mathinner{.\,.} b]$ satisfy any of the conditions (1),(2),(3),(4),(5) of Definition 4.14.

Lemma 4.15. *Let $\{[\, a_i \,..\, b_i \,]\}_{i=0}^{\ell-1}$ denote the set that contains all intervals that satisfy any of the conditions (1),(2),(3),(4),(5) of Definition 4.14. Consider the period cover $C_S = \bigcup_{i=0}^{\ell-1}[\, a_i \,..\, b_i \,]$. Then, we can encode $\{(c, T[\tau_0^c]) : c \in C_S\}$ using $O(w + k|S|)$ space.*

Proof. First, we briefly argue that it is possible to encode $\{(c, \tau_0^c) : c \in C_S\}$ using $O(k|S|)$ space. As already argued before, by storing for every $\mathcal{X} \in S$ the corresponding edit information, we can totally reconstruct $\mathbf{G}_S$. This allows us to check whether a character of T is contained in a black connected component. Moreover, given it is not contained in a red component, we can infer in which black component it is contained. Since the sets of edit information can be stored using $O(k)$ space (through the compressed edit information) for every $\mathcal{X} \in S$, it follows that we can encode the corresponding information using $O(k|S|)$ space.

Therefore, to prove the lemma, it suffices to show that it is possible to select a subset of intervals indexed by $I \subseteq [\, 0 \,..\, \ell \,)$ such that $\bigcup_{i \in I}[\, a_i \,..\, b_i \,] = C_S$ and that $\{T[\, \tau_0^{a_i} \,..\, \tau_0^{b_i} \,]\}_{i \in I}$ can be efficiently encoded.

Among all intervals that satisfy condition (1) of Definition 4.14, i.e., among all intervals $[\, a_i \,..\, b_i \,]$ such that $a_i = 0$, we add to I the index j that maximizes b_j. Clearly, for all other intervals $[\, a_i \,..\, b_i \,]$ such that $a_i = 0$, we have $[\, a_i \,..\, b_i \,] \subseteq [\, a_j \,..\, b_j \,]$, and we can safely leave them out from I. Moreover, from $\mathsf{self}\text{-}\delta_E(T[\, \tau_0^{a_j} \,..\, \tau_0^{b_j} \,]) \le 6w + 11k$ and from Lemma 3.7 follows that $|\mathsf{LZ}(T[\, \tau_0^{a_j} \,..\, \tau_0^{b_j} \,])| \le O(k+w)$. By using a similar approach for the intervals that satisfy any of the conditions (2), (3), (4), henceforth, we may assume that we have taken care of all the intervals that satisfy any of the conditions (1), (2), (3), (4), and that $\{[\, a_i \,..\, b_i \,]\}_{i=0}^{\ell-1}$ only contains intervals that satisfy (5).

Now, we proceed by iteratively adding indices to I. The first index we add to I is $\arg\min_i\{a_i\}$. Next, we continue to add indices to I based on the last index we added, j, as follows:

- if $\{i : a_j < a_i \le b_j < b_i\} \ne \emptyset$, then we add the index $\arg\max_i\{b_i : a_j < a_i \le b_j < b_i\}$ to I;
- otherwise, we add the index $\arg\min_i\{a_i : b_j < a_i\}$ to I.

We terminate if we can not add any index to I anymore, i.e., if $b_j = \max_i\{b_i\}$. Note, this iterative selection process is guaranteed to end, because in every iteration for the i that we add to I, we have $a_j < a_i$.

We want to argue $\{T[\, \tau_0^{a_i} \,..\, \tau_0^{b_i} \,]\}_{i \in I} = C_S$. To do this, consider an interval $[\, x \,..\, y \,]$ in the interval representation of C_S, i.e., the representation of C_S using the smallest number of non-overlapping intervals. We observe that an index i with $a_i = x$ is added either initially or through the first rule. Subsequently, the selection process applies a combination of the two rules until we include an index i with $b_i = y$ into I.

Finally, we want to show that $\sum_{i \in I} \mathsf{self}\text{-}\delta_E(T[\, \tau_0^{a_i} \,..\, \tau_0^{b_i} \,]) \le 24w$. This is sufficient to prove that $\{T[\, \tau_0^{a_i} \,..\, \tau_0^{b_i} \,]\}_{i \in I}$ can be encoded efficiently, because from Lemma 3.7 follows

$$\sum_{i \in I} |\mathsf{LZ}(T[\, \tau_0^{a_i} \,..\, \tau_0^{b_i} \,])| \le 2 \sum_{i \in I} \mathsf{self}\text{-}\delta_E(T[\, \tau_0^{a_i} \,..\, \tau_0^{b_i} \,]) = O(w).$$

Consider three indices i, i', i'' that are added one after the other to I. If for the selection of i' or i'' we need to apply the second rule, then clearly $b_i < a_{i''}$. Otherwise, if always apply the first rule, then from $b_{i'} < b_{i''}$ follows that $b_i < a_{i''}$, because otherwise, we would have selected i'' instead of i' in the arg min. Hence, for every $c \in [\, 0 \,..\, \mathrm{bc}(\mathbf{G}_S) \,)$ it holds $|\{i \in I : c \in [\, a_i \,..\, b_i \,]\}| \le 2$, from which follows $|\{i \in I : c \in [\, a_i - 1 \,..\, b_i \,]\}| = |\{i \in I : c \in [\, a_i \,..\, b_i \,]\}| + |\{i \in I : c + 1 = a_i\}| \le 4$. Note, the previous property also holds when we add less than three indices to I. Using a double counting argument, we obtain

$$\sum_{i \in I} \mathsf{self}\text{-}\delta_E(T[\, \tau_0^{a_i} \,..\, \tau_0^{b_i} \,]) \le 6 \sum_{i \in I} \sum_{c = a_i - 1}^{b_i} \mathrm{w}_S(c) = 6 \sum_{c=0}^{\mathrm{bc}(\mathbf{G}_S)-1} \sum_{\substack{i \in I \\ c \in [\, a_i - 1 \,..\, b_i \,]}} \mathrm{w}_S(c) \le 24w,$$

which concludes the proof. ◀

One drawback of this approach is the necessity to read all characters within the black components prior to encoding. To circumvent this, we offer a second alternative construction of a period cover C_S designed to minimize the number of queries to the string. Such a strategy proves particularly advantageous in the quantum setting. In the remaining part of this paper, we abuse notation and write $\mathrm{w}_S(-1) = \mathrm{w}_S(\mathrm{bc}(\mathbf{G}_S) - 1)$.

Lemma 4.16. *The following (constructive) definition delivers a period cover C_S. Set*

$$C_S := [\,0\,..\,c_{\mathrm{pref}}\,] \cup [\,c_{\mathrm{suff}}\,..\,\mathrm{bc}(\mathbf{G}_S)\,) \cup [\,c_{\mathrm{lsuff}}\,..\,c_{\mathrm{lpref}}\,] \cup C_S^{0,\mathrm{bc}(\mathbf{G}_S)-1},$$

where c_{pref}, c_{suff}, c_{lsuff}, c_{lpref}, and $C_S^{0,\mathrm{bc}(\mathbf{G}_S)-1}$ are defined as follows.

(a) *Let $c_{\mathrm{pref}} \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ denote the largest index such that $|\mathsf{LZ}(\overline{T[\,\tau_0^0\,..\,\tau_0^{c_{\mathrm{pref}}}\,]})| \le 12w + 22k$.*

(b) *Let $c_{\mathrm{suff}} \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ denote the smallest index such that $|\mathsf{LZ}(T[\,\tau_0^{c_{\mathrm{suff}}}\,..\,\tau_0^{\mathrm{bc}(\mathbf{G}_S)-1}\,])| \le 12w + 22k$.*

(c) *Let $c_{\mathrm{lsuff}} \in [\,0\,..\,c_{\mathrm{last}}\,]$ denote the smallest index such that $|\mathsf{LZ}(\overline{T[\,\tau_0^{c_{\mathrm{lsuff}}}\,..\,\tau_0^{c_{\mathrm{last}}}\,]})| \le 12w + 22k$.*

(d) *Let $c_{\mathrm{lpref}} \in [\,c_{\mathrm{last}}+1\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ denote the largest index such that $|\mathsf{LZ}(T[\,\tau_0^{c_{\mathrm{last}}+1}\,..\,\tau_0^{c_{\mathrm{lpref}}}\,])| \le 12w + 22k$.*

(e) *For $[\,i\,..\,j\,] \subseteq [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$, we define $C_S^{i,j} \subseteq [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ as follows.*
- *If $\sum_{c=i-1}^{j} \mathrm{w}_S(c) = 0$, then $C_S^{i,j} := \varnothing$.*
- *If $i = j$, then $C_S^{i,j} := \{i\}$.*
- *Otherwise, let $h := \lfloor (i+j)/2 \rfloor$. Let $i' \in [\,i\,..\,h\,]$ denote the smallest index such that $|\mathsf{LZ}(\overline{T[\,\tau_0^{i'}\,..\,\tau_0^{h}\,]})| \le 12\sum_{c=i-1}^{j} \mathrm{w}_S(c)$. Similarly, let $j' \in [\,h\,..\,j\,]$ denote the largest index such that $|\mathsf{LZ}(T(\,\tau_0^{h}\,..\,\tau_0^{j'}\,])| \le 12\sum_{c=i-1}^{j} \mathrm{w}_S(c)$. In this case, we define recursively $C_S^{i,j} := C_S^{i,h} \cup C_S^{h+1,j} \cup [\,i'\,..\,j'\,]$.*

Proof. Let $[\,a\,..\,b\,] \subseteq [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ be an interval satisfying at least one of the conditions of Definition 4.14. We need to prove that $[\,a\,..\,b\,] \subseteq C_S$. Assume $[\,a\,..\,b\,]$ satisfies Definition 4.14(1), i.e., $a = 0$ and $\mathsf{self}\text{-}\delta_E(T[\,\tau_0^0\,..\,\tau_0^b\,]) \le 6w + 11k$. By Lemma 3.7 we conclude

$$|\mathsf{LZ}(T[\,\tau_0^0\,..\,\tau_0^b\,])| \le 2\mathsf{self}\text{-}\delta_E(T[\,\tau_0^0\,..\,\tau_0^b\,]) \le 12w + 22k.$$

Consequently, $[\,0\,..\,b\,] \subseteq [\,0\,..\,c_{\mathrm{pref}}\,] \subseteq C_S$. Since $\mathsf{self}\text{-}\delta_E$ is insensitive to string reversal, similar arguments apply for Definition 4.14(2)(3)(4).

Now, assume Definition 4.14(5) holds for $[\,a\,..\,b\,]$ of size at least two. Note, for such interval we have $\mathsf{self}\text{-}\delta_E(T[\,\tau_0^a\,..\,\tau_0^b\,]) > 0$. We want to argue that there exist indices $i, j \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ fulfilling all the three following properties:
- $C_S^{i,j}$ is involved in the recursive construction of $C_S^{0,\mathrm{bc}(\mathbf{G}_S)}$;
- $[\,a\,..\,b\,] \subseteq [\,i\,..\,j\,]$; and
- $h \in [\,a\,..\,b\,)$ for $h := \lfloor (i+j)/2 \rfloor$.

For that purpose, note that, in the recursive definition, we start with an interval $[\,i\,..\,j\,] = [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ which clearly contains $[\,a\,..\,b\,]$. At each step, we recurse on $[\,i\,..\,h\,]$ and $[\,h+1\,..\,j\,]$, where $h = \lfloor (i+j)/2 \rfloor$. If $[\,a\,..\,b\,]$ is contained in neither of these intervals, then $i \le a \le h < h+1 \le b \le j$ and, in particular, $h \in [\,a\,..\,b\,)$. Consequently, $\tau_0^a \le \tau_0^h < \tau_0^b$. By Lemma 3.7 we conclude that

$$|\mathsf{LZ}(\overline{T[\,\tau_0^a\,..\,\tau_0^h\,]})| \le 2\mathsf{self}\text{-}\delta_E(T[\,\tau_0^a\,..\,\tau_0^b\,]) \le 12\sum_{c=a-1}^{b} \mathrm{w}_S(c) \le 12\sum_{c=i-1}^{j} \mathrm{w}_S(c).$$

Similarly,

$$|\mathsf{LZ}(T(\,\tau_0^h\,..\,\tau_0^b\,])| \le 2\mathsf{self}\text{-}\delta_E(T[\,\tau_0^a\,..\,\tau_0^b\,]) \le 12\sum_{c=a-1}^{b} \mathrm{w}_S(c) \le 12\sum_{c=i-1}^{j} \mathrm{w}_S(c).$$

Consequently, $[\,a\,..\,b\,]$ is contained in the interval $[\,i'\,..\,j'\,] \subseteq [\,i\,..\,j\,]$ that is added to $C_S^{i,j} \subseteq C_S$.

Finally, if $[\,a\,..\,a\,]$ satisfies Definition 4.14(5), then $\sum_{c=i-1}^{j} \mathrm{w}_S(c) > 0$ holds for every interval $[\,i\,..\,j\,]$ such that $[\,a\,..\,a\,] \subseteq [\,i\,..\,j\,] \subseteq [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$. Consequently, $a \in C_S^{a,a} \subseteq C_S^{0,\mathrm{bc}(\mathbf{G}_S)-1}$.

Lemma 4.17. *Let C_S be a period cover obtained as described in Lemma 4.16. Then, the information $\{(c, T[\tau_0^c]) : c \in C_S\}$ can be encoded in $O(w \log n + k|S|)$ space.*

Proof. As already argued in the proof of Lemma 4.15, we can encode the information $\{(c, \tau_0^c) : c \in C_S\}$ using $O(k|S|)$ space. Therefore, it suffices to show that we can encode in $O(w \log n + k)$ space the intervals $[\,\tau_0^a\,..\,\tau_0^b\,]$ for all $[\,a\,..\,b\,] \subseteq [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ that we add to C_S in any of Lemma 4.16(a)(b)(c)(e)(d).

First, note that $[\,0\,..\,c_{\mathrm{pref}}\,]$ can be encoded in $O(w+k)$ space since $|\mathsf{LZ}(T[\,\tau_0^0\,..\,\tau_0^{c_{\mathrm{pref}}}\,])| \le 12w + 22k$. Using similar arguments, we get that the four intervals that we add in any of Lemma 4.16(a)(b)(c)(d) can be encoded in $O(w+k)$ space. It remains to show that all intervals that we add in Lemma 4.16(e), i.e., in the recursive construction of $C_S^{0,\mathrm{bc}(\mathbf{G}_S)-1}$, can be encoded in $O(w \log n)$ space. Let $[\,i_0\,..\,j_0\,], \ldots, [\,i_{d-1}\,..\,j_{d-1}\,]$ be all the intervals considered by the recursion at a fixed depth. Observe that $\sum_{r=0}^{d-1} \sum_{c=i_r-1}^{j_r} \mathrm{w}_S(c) \le 2w$, because for every $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G})-1\,)$ the term $\mathrm{w}_S(c)$ does not appear more than twice in the summation. Suppose that for $[\,i\,..\,j\,] \in \{[\,i_r\,..\,j_r\,]\}_{r=0}^{d-1}$ we add $[\,i'\,..\,j'\,]$ to $C_S^{i,j}$. Then, we can encode the corresponding $T[\,\tau_0^{i'}\,..\,\tau_0^{j'}\,]$ using $20\sum_{c=i-1}^{j} \mathrm{w}_S(c)$ space. This is because there exists h such that $|\mathsf{LZ}(\overline{T[\,\tau_{i'}^0\,..\,\tau_h^0\,]})| \le 10\sum_{c=i-1}^{j} \mathrm{w}_S(c)$ and $|\mathsf{LZ}(T(\,\tau_h^0\,..\,\tau_{j'}^0\,])| \le 10\sum_{c=i-1}^{j} \mathrm{w}_S(c)$. As a consequence, we obtain that we can encode all intervals that we add to $C_S^{0,\mathrm{bc}(\mathbf{G}_S)-1}$ at a fixed depth using at most $\sum_{r=0}^{d-1} 20 \sum_{c=i_r-1}^{j_r} \mathrm{w}_S(c) = O(w)$ space. Since the recursion has depth at most $O(\log n)$, we conclude that the intervals that we add in Lemma 4.16(e) can be encoded in $O(w \log n)$ space.

4.4 Close Candidate Positions

This (sub)section is devoted to proving the following result:

Proposition 4.24. *Let $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$ be an optimal alignment of P onto a fragment $T[\,t\,..\,t'\,)$ such that $\delta_E(P, T[\,t\,..\,t'\,)) \le k$. If there exists $i \in [\,0\,..\,n_0 - m_0\,]$ such that $|\tau_i^0 - t - \pi_0^0| \le w + 3k$, then the following holds for every $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,) \setminus C_S$:*

(1) *$\mathcal{X}$ aligns $P[\pi_j^c]$ to $T[\tau_{i+j}^c]$ for every $j \in [\,0\,..\,m_c\,)$, and*

(2) *$\tau_{i'}^c \notin [\,t\,..\,t'\,)$ for every $i' \in [\,0\,..\,n_c\,) \setminus [\,i\,..\,i+m_c\,)$.*

4.4.1 Partition of P and T into Blocks

Proposition 4.24 heavily relies on the fact that P and T exhibit a periodic structure. We divide P and T into the blocks $\{P_j\}_{j\in[\,0\,..\,m_0\,)}$ and $\{T_i\}_{i\in[\,0\,..\,n_0\,)}$ having a similar structure.

Definition 4.18. *For $j \in [\,0\,..\,m_0\,)$ we define*

$$P_j := \begin{cases} P[\,\pi_j^0\,..\,\pi_{j+1}^0\,) & \text{if } j \neq m_0 - 1,\\ P[\,\pi_{m_0-1}^0\,..\,\pi_{m_0-1}^{c_{\mathrm{last}}}\,] & \text{otherwise.} \end{cases}$$

Similarly, for $i \in [\,0\,..\,n_0\,)$ we define

$$T_i := \begin{cases} T[\,\tau_i^0\,..\,\tau_{i+1}^0\,) & \text{if } i \neq n_0 - 1,\\ T[\,\tau_{n_0-1}^0\,..\,\tau_{n_0-1}^{c_{\mathrm{last}}}\,] & \text{otherwise.} \end{cases}$$

The notion of similarity between these blocks is also captured by the fact that for any j, i we can construct an alignment of P_j onto T_i whose cost is upper bounded by the total weight of the weight function w_S. More formally, we can prove the following proposition.

Proposition 4.19. *Let $i \in [0\mathinner{.\,.} n_0)$, $j \in [0\mathinner{.\,.} m_0)$ be arbitrary such that if $i = n_0 - 1$ then $j = m_0 - 1$. Then, there is an alignment $\mathcal{A}_S^{j,i}$ with the following properties.*

(i) *$\mathcal{A}_S^{j,i} : P_j \rightsquigarrow T_i$ if $j \neq m_0 - 1$, and $\mathcal{A}_S^{j,i} : P_{m_0-1} \rightsquigarrow T[\tau_i^0 \mathinner{.\,.} \tau_i^{c_{\text{last}}}]$ otherwise.*

(ii) *Let $c \in [0\mathinner{.\,.} \mathrm{bc}(\mathbf{G}_S))$ if $j \neq m_0 - 1$, and $c \in [0\mathinner{.\,.} c_{\text{last}}]$ if $j = m_0 - 1$. Then, $\mathcal{A}_S^{j,i}$ matches $P[\pi_j^c]$ and $T[\tau_i^c]$.*

(iii) *$\mathcal{A}_S^{j,i}$ has cost at most w.*

Proof. Let $c \in [0\mathinner{.\,.} \mathrm{bc}(\mathbf{G}_S))$ if $j \neq m_0 - 1$ and let $c \in [0\mathinner{.\,.} c_{\text{last}})$ if $j = m_0 - 1$. From Definition 4.9(1) follows that there is an alignment $\mathcal{X}_c : P[\pi_j^c \mathinner{.\,.} \pi_j^{c+1}) \rightsquigarrow T[\tau_i^c \mathinner{.\,.} \tau_i^{c+1})$ with cost at most $\mathrm{w}_S(c)$. Note, since $P[\pi_j^c] = T[\tau_i^c]$, we may assume that $\mathcal{X}_c$ matches $P[\pi_j^c]$ and $T[\tau_i^c]$. Now, it suffices to set

$$\mathcal{A}_S^{j,i} := \begin{cases} \bigcup_{c=0}^{\mathrm{bc}(\mathbf{G}_S)-1} \mathcal{X}_c & \text{if } j \neq m_0 - 1, \\ \left(\bigcup_{c=0}^{c_{\text{last}}-1} \mathcal{X}_c\right) \cup \{(\pi_{m_0-1}^{c_{\text{last}}} + 1, \tau_i^{c_{\text{last}}} + 1)\} & \text{if } j = m_0 - 1. \end{cases}$$

Observe that $\mathcal{A}_S^{j,i}$ is a valid alignment because the last two characters aligned by $\mathcal{X}_{c-1}$ are exactly the first two characters aligned by $\mathcal{X}_c$. It is clear that (i) and (ii) hold. Regarding (iii), note that the cost of $\mathcal{A}_S^{j,i}$ equals the sum of the costs of all $\mathcal{X}_c$ whose union defines $\mathcal{A}_S^{j,i}$. This sum is upper bounded by the sum of the corresponding $\mathrm{w}_S(c)$, and is therefore at most w. ◾

Lemma 4.20. *Let $j \in [0\mathinner{.\,.} m_0)$, $i \in [0\mathinner{.\,.} n_0)$ such that if $i = n_0 - 1$ then $j = m_0 - 1$, and let $\mathcal{A} := \mathcal{A}_S^{j,i}$. Let $P[x\mathinner{.\,.} x')$ and $T[y\mathinner{.\,.} y')$ be fragments of P_j and T_i, respectively, such that $\mathcal{A}$ aligns $P[x\mathinner{.\,.} x')$ onto $T[y\mathinner{.\,.} y')$ with cost $\delta^{\mathcal{A}}(P[x\mathinner{.\,.} x'), T[y\mathinner{.\,.} y')) > 0$. Let $\mathcal{X}$ be an optimal alignment of $P[x\mathinner{.\,.} x')$ onto $T[y\mathinner{.\,.} y')$. If there exists no $(\hat{x}, \hat{y}) \in \mathcal{X} \cap \mathcal{A}$ such that both $\mathcal{X}$ and $\mathcal{A}$ match $P[\hat{x}]$ and $T[\hat{y}]$, then $\{c \in [0\mathinner{.\,.} \mathrm{bc}(\mathbf{G})) : \pi_j^c \in [x\mathinner{.\,.} x')\} \subseteq C_S$.*

Proof. Denote $[a\mathinner{.\,.} b] = \{c \in [0\mathinner{.\,.} \mathrm{bc}(\mathbf{G})) : \pi_j^c \in [x\mathinner{.\,.} x')\}$ and assume that this interval is non-empty (otherwise, there is nothing to prove). Moreover, let $[a'\mathinner{.\,.} b] = \{c \in [0\mathinner{.\,.} \mathrm{bc}(\mathbf{G})) : \pi_j^c \in (x\mathinner{.\,.} x')\}$. Now, we can apply the following chain of inequalities

$$\begin{aligned}
\textsf{self-}\delta_E(T[\tau_i^a \mathinner{.\,.} \tau_i^b]) &\leq \textsf{self-}\delta_E(T[y\mathinner{.\,.} y')) && (2)\\
&\leq 2\delta_E^{\mathcal{A}}(P[x\mathinner{.\,.} x'), T[y\mathinner{.\,.} y')) && (3)\\
&\leq 2\delta_E^{\mathcal{A}}(P[\pi_j^{a'-1} \mathinner{.\,.} \pi_j^{b+1}), T[\tau_i^{a'-1} \mathinner{.\,.} \tau_i^{b+1})) && (4)\\
&= 2\sum_{c=a'-1}^{b} \delta_E^{\mathcal{A}}(P[\pi_j^c \mathinner{.\,.} \pi_j^{c+1}), T[\tau_i^c \mathinner{.\,.} \tau_i^{c+1})) && (5)\\
&\leq 2\sum_{c=a'-1}^{b} \mathrm{w}_S(c) && (6)\\
&\leq 2\sum_{c=a-1}^{b} \mathrm{w}_S(c) && (7)
\end{aligned}$$

where we have used:

(2) monotonicity of $\textsf{self-}\delta_E$ (Lemma 3.8);

(3) Lemma 3.9 and $\delta_E^{\mathcal{X}}(P[x\mathinner{.\,.} x'), T[y\mathinner{.\,.} y')) \leq \delta_E^{\mathcal{A}}(P[x\mathinner{.\,.} x'), T[y\mathinner{.\,.} y'))$;

(4) the definition of $[\, a' \mathinner{.\,.} b \,]$ and the fact that $(\pi_j^{a'-1}, \tau_i^{a'-1}), (\pi_j^{b+1}, \tau_i^{b+1}) \in \mathcal{A}$;
(5) the fact that $(\pi_j^c, \tau_i^c) \in \mathcal{A}$ for all $c \in [\, a \mathinner{.\,.} b \,]$; and
(6) Definition 4.9.

Now, notice that $\delta_E(T[\,\tau_i^a \mathinner{.\,.} \tau_i^b\,], T[\,\tau_0^a \mathinner{.\,.} \tau_0^b\,]) \le \sum_{c=a}^{b-1} \delta_E(T[\,\tau_i^c \mathinner{.\,.} \tau_i^{c+1}\,], T[\,\tau_0^c \mathinner{.\,.} \tau_0^{c+1}\,]) \le 2\sum_{c=a}^{b-1} \mathrm{w}_S(c)$ follows from Definition 4.9. From this together with Lemma 3.7 and self-$\delta_E(T[\,\tau_i^a \mathinner{.\,.} \tau_i^b\,]) \le 2\sum_{c=a-1}^{b} \mathrm{w}_S(c)$ follows self-$\delta_E(T[\,\tau_0^a \mathinner{.\,.} \tau_0^b\,]) \le 6\sum_{c=a-1}^{b} \mathrm{w}_S(c)$. Hence, we can use Definition 4.14 to deduce $[\, a \mathinner{.\,.} b \,] \subseteq C_S$. ◀

4.4.2 Recovering the Edit Distance for a Single Candidate Position

▶ **Lemma 4.21.** *Let $j \in [\, 0 \mathinner{.\,.} m_0 \,)$ and $i \in [\, 0 \mathinner{.\,.} n_0 \,)$ be such that $j = m_0 - 1$ if $i = n_0 - 1$. Let $\mathcal{X} : P_j \rightsquigarrow T[\, y \mathinner{.\,.} y' \,)$ be an optimal alignment of P_j onto an arbitrary fragment $T[\, y \mathinner{.\,.} y' \,)$. If $|\tau_i^0 - y| \le w + 4k$ and $\delta_E(P_j, T[\, y \mathinner{.\,.} y' \,)) \le k$, then $\mathcal{X}$ aligns $P[\pi_j^c]$ to $T[\tau_i^c]$ for all $c \in [\, 0 \mathinner{.\,.} \mathrm{bc}(\mathbf{G}_S) \,) \setminus C_S$ such that $j \in [\, 0 \mathinner{.\,.} m_c \,)$.*

Proof. We first assume that $j < m_0 - 1$ and then briefly argue that the case of $j = m_0 - 1$ can be handled similarly. Assume $[\, 0 \mathinner{.\,.} \mathrm{bc}(\mathbf{G}_S) \,) \setminus C_S \ne \emptyset$; otherwise there is nothing to prove. Denote $c_\mathrm{l} = \min([\, 0 \mathinner{.\,.} \mathrm{bc}(\mathbf{G}_S) \,) \setminus C_S)$ and $c_\mathrm{r} = \max([\, 0 \mathinner{.\,.} \mathrm{bc}(\mathbf{G}_S) \,) \setminus C_S)$. Henceforth, we set $\mathcal{A} := \mathcal{A}_S^{j,i}$.

▷ Claim 4.22. *There exist $(x_\mathrm{l}, y_\mathrm{l}), (x_\mathrm{r}, y_\mathrm{r}) \in \mathcal{A} \cap \mathcal{X}$ such that $(x_\mathrm{l}, y_\mathrm{l}) \le (\pi_j^{c_\mathrm{l}}, \tau_i^{c_\mathrm{l}})$ and $(\pi_j^{c_\mathrm{r}}, \tau_i^{c_\mathrm{r}}) \le (x_\mathrm{r}, y_\mathrm{r})$.*

Proof. First, we want to argue that there exists $(x_\mathrm{l}, y_\mathrm{l}) \in \mathcal{A} \cap \mathcal{X}$ such that $(x_\mathrm{l}, y_\mathrm{l}) \le (\pi_j^{c_\mathrm{l}}, \tau_i^{c_\mathrm{l}})$. Note, following Definition 4.14 we must have self-$\delta_E(T[\,\tau_0^0 \mathinner{.\,.} \tau_0^{c_\mathrm{l}}\,]) > 6w + 11k$; otherwise c_l would have been included in C_S. Suppose that $\mathcal{X}$ aligns $P[\,\pi_j^0 \mathinner{.\,.} \pi_j^{c_\mathrm{l}}\,)$ with $T[\, y \mathinner{.\,.} \hat{y} \,)$. For sake of contradiction, suppose that $\mathcal{A} : P[\,\pi_j^0 \mathinner{.\,.} \pi_j^{c_\mathrm{l}}\,) \rightsquigarrow T[\,\tau_i^0 \mathinner{.\,.} \tau_i^{c_\mathrm{l}}\,)$ and $\mathcal{X} : P[\,\pi_j^0 \mathinner{.\,.} \pi_j^{c_\mathrm{l}}\,) \rightsquigarrow T[\, y \mathinner{.\,.} \hat{y} \,)$ are disjoint. By applying Lemma 3.9 to $\mathcal{A} : P[\,\pi_j^0 \mathinner{.\,.} \pi_j^{c_\mathrm{l}}\,) \rightsquigarrow T[\,\tau_i^0 \mathinner{.\,.} \tau_i^{c_\mathrm{l}}\,)$ and $\mathcal{X} : P[\,\pi_j^0 \mathinner{.\,.} \pi_j^{c_\mathrm{l}}\,) \rightsquigarrow T[\, y \mathinner{.\,.} \hat{y} \,)$, we obtain

$$\begin{aligned}
\text{self-}\delta_E(P[\,\pi_j^0 \mathinner{.\,.} \pi_j^{c_\mathrm{l}}\,)) &\le |\tau_i^0 - y| + \delta_E^{\mathcal{A}}(P[\,\pi_j^0 \mathinner{.\,.} \pi_j^{c_\mathrm{l}}\,), T[\,\tau_i^0 \mathinner{.\,.} \tau_i^{c_\mathrm{l}}\,)) + \delta_E^{\mathcal{X}}(P[\,\pi_j^0 \mathinner{.\,.} \pi_j^{c_\mathrm{l}}\,), T[\, y \mathinner{.\,.} \hat{y} \,)) + |\tau_i^{c_\mathrm{l}} - \hat{y}| \\
&\le 2|\tau_i^0 - y| + 2\delta_E^{\mathcal{A}}(P[\,\pi_j^0 \mathinner{.\,.} \pi_j^{c_\mathrm{l}}\,), T[\,\tau_i^0 \mathinner{.\,.} \tau_i^{c_\mathrm{l}}\,)) + 2\delta_E^{\mathcal{X}}(P[\,\pi_j^0 \mathinner{.\,.} \pi_j^{c_\mathrm{l}}\,), T[\, y \mathinner{.\,.} \hat{y} \,)) \\
&\le 2w + 8k + 2w + 2k = 4w + 10k,
\end{aligned}$$

where we have used

$$\begin{aligned}
|\tau_i^{c_\mathrm{l}} - \hat{y}| &\le |\tau_i^0 - y| + |(\tau_i^{c_\mathrm{l}} - \tau_i^0) - (\pi_j^{c_\mathrm{l}} - \pi_j^0)| + |(\hat{y} - y) - (\pi_j^{c_\mathrm{l}} - \pi_j^0)| \\
&\le |\tau_i^0 - y| + \delta_E^{\mathcal{A}}(P[\,\pi_j^0 \mathinner{.\,.} \pi_j^{c_\mathrm{l}}\,), T[\,\tau_i^0 \mathinner{.\,.} \tau_i^{c_\mathrm{l}}\,)) + \delta_E^{\mathcal{X}}(P[\,\pi_j^0 \mathinner{.\,.} \pi_j^{c_\mathrm{l}}\,), T[\, y \mathinner{.\,.} \hat{y} \,)).
\end{aligned}$$

An application of Lemma 3.7 on $P[\,\pi_j^0 \mathinner{.\,.} \pi_j^{c_\mathrm{l}}\,)$ and $\mathcal{A}_S^{j,0}$ yields

$$\text{self-}\delta_E(T[\,\tau_0^0 \mathinner{.\,.} \tau_0^{c_\mathrm{l}}\,]) \le \text{self-}\delta_E(P[\,\pi_j^0 \mathinner{.\,.} \pi_j^{c_\mathrm{l}}\,)) + 2w = 6w + 10k.$$

As a consequence, we have

$$\text{self-}\delta_E(T[\,\tau_0^0 \mathinner{.\,.} \tau_0^{c_\mathrm{l}}\,]) \le \text{self-}\delta_E(T[\,\tau_0^0 \mathinner{.\,.} \tau_0^{c_\mathrm{l}}\,)) + 1 \le 6w + 11k,$$

and we obtain a contradiction.

Since the alignments $\mathcal{A} : P_j \rightsquigarrow T_i$ and $\mathcal{X} : P_j \rightsquigarrow T[\, y \mathinner{.\,.} y' \,)$ intersect and have costs at most w and k, respectively, we conclude that $|\tau_{i+1}^0 - y'| \le w + k \le w + 4k$. Consequently, by an argument symmetric to the above, there exists $(x_\mathrm{r}, y_\mathrm{r}) \in \mathcal{A} \cap \mathcal{X}$ such that $(\pi_j^{c_\mathrm{r}}, \tau_i^{c_\mathrm{r}}) \le (x_\mathrm{r}, y_\mathrm{r})$. ◁

Now, for the sake of contradiction, suppose that there exists $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)\setminus C_S$ such that $\mathcal{X}$ does not align $P[\pi_j^c]$ to $T[\tau_i^c]$. Let $(x_c, y_c) \in \mathcal{A}\cap\mathcal{X}$ be the largest (x_c, y_c) such that $(x_c, y_c) \le (\pi_j^c, \tau_i^c)$, and let $(x'_c, y'_c) \in \mathcal{A}\cap\mathcal{X}$ be the smallest (x'_c, y'_c) such that $(\pi_j^c, \tau_i^c) \le (x'_c, y'_c)$. Note, we know that such $(x_c, y_c), (x'_c, y'_c)$ exist because $(x_\mathrm{l}, y_\mathrm{l}), (x_\mathrm{r}, y_\mathrm{r}) \in \mathcal{A}\cap\mathcal{X}$ are such that $(x_\mathrm{l}, y_\mathrm{l}) \le (\pi_j^{c_\mathrm{l}}, \tau_i^{c_\mathrm{l}}) \le (\pi_j^c, \tau_i^c) \le (\pi_j^{c_\mathrm{r}}, \tau_i^{c_\mathrm{r}}) \le (x_\mathrm{r}, y_\mathrm{r})$.

Claim 4.23. *$\mathcal{X}$ and $\mathcal{A}$ do not both align $P[x_c]$ to $T[y_c]$ at the same time.*

Proof. For the sake of contradiction, assume $\mathcal{X}$ and $\mathcal{A}$ both align $P[x_c]$ to $T[y_c]$. Thus, $(x_c+1, y_c+1) \in \mathcal{X}\cap\mathcal{A}$. Since $(\pi_j^c, \tau_i^c) \notin \mathcal{X}$, we must have $(x_c, y_c) \ne (\pi_j^c, \tau_i^c)$, and (x_c, y_c) appears strictly before $(\pi_j^c, \tau_i^c) \in \mathcal{A}$ in $\mathcal{A}$. Consequently, $(x_c + 1, y_c + 1) \le (\pi_j^c, \tau_i^c)$. However, this is a contradiction with (x_c, y_c) being the largest $(x_c, y_c) \in \mathcal{X}\cap\mathcal{A}$ such that $(x_c, y_c) \le (\pi_j^c, \tau_i^c)$.

As a consequence, there is no $(\hat{x}, \hat{y}) \in \mathcal{A}\cap\mathcal{X}$ such that both $\mathcal{A} : P[\,x_c\,..\,x'_c\,) \rightsquigarrow T[\,y_c\,..\,y'_c\,)$ and $\mathcal{X} : P[\,x_c\,..\,x'_c\,) \rightsquigarrow T[\,y_c\,..\,y'_c\,)$ align $P[\hat{x}]$ to $T[\hat{y}]$. Therefore, we can use Lemma 4.20 obtaining $c \in C_S$. However, this is a contradiction with the definition of c.

The proof for $j = m_0 - 1$ is almost identical to the proof for $j < m_0 - 1$: in the proof, we replace every occurrence of $[\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ with $[\,0\,..\,c_\mathrm{last}\,]$ and every occurrence of τ_{i+1}^0 with $\tau_i^{c_\mathrm{last}} + 1$.

Finally, to conclude this (sub)section, we prove Proposition 4.24.

Proposition 4.24. *Let $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$ be an optimal alignment of P onto a fragment $T[\,t\,..\,t'\,)$ such that $\delta_E(P, T[\,t\,..\,t'\,)) \le k$. If there exists $i \in [\,0\,..\,n_0 - m_0\,]$ such that $|\tau_i^0 - t - \pi_0^0| \le w + 3k$, then the following holds for every $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)\setminus C_S$:*

(1) *$\mathcal{X}$ aligns $P[\pi_j^c]$ to $T[\tau_{i+j}^c]$ for every $j \in [\,0\,..\,m_c\,)$, and*

(2) *$\tau_{i'}^c \notin [\,t\,..\,t'\,)$ for every $i' \in [\,0\,..\,n_c\,)\setminus[\,i\,..\,i+m_c\,)$.*

Proof. We assume that $C_S \subsetneq [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$; otherwise, there is nothing to prove.

We prove (1) by induction on $j \in [\,0\,..\,m_0\,)$. Define $y_0, \ldots, y_{m_0} \in [\,t\,..\,t'\,]$ so that $\mathcal{X}(P_j) = T[\,y_j\,..\,y_{j+1}\,)$ holds for all $j \in [\,0\,..\,m_0\,)$. Let us first prove that $|\tau_{i+j}^0 - y_j| \le w + 4k$. If $j = 0$, we observe that $|\pi_0^0 - (y_0 - t)| \le k$ because $\mathcal{X}$ aligns $P[\,0\,..\,\pi_0^0\,)$ with $T[\,t\,..\,y_0\,)$ at a cost not exceeding k. Combining this inequality with the assumption $|\tau_i^0 - t - \pi_0^0| \le w + 3k$, we conclude that $|\tau_i^0 - y_0| \le w + 4k$. If $j > 0$, on the other hand, consider $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$, for which the inductive assumption yields $(\pi_{j-1}^c, \tau_{j+i-1}^c) \in \mathcal{X}$. Thus, $\mathcal{X} : P_{j-1} \rightsquigarrow T[\,y_{j-1}\,..\,y_j\,)$ and $\mathcal{A}_S^{j-1,i+j-1} : P_{j-1} \rightsquigarrow T[\,\tau_{i+j-1}^0\,..\,\tau_{i+j}^0\,)$ are intersecting alignments of costs at most k and w, respectively. Consequently, $|\tau_{i+j}^0 - y_j| \le w + k \le w + 4k$. In either case, we have proved that $|\tau_{i+j}^0 - y_j| \le w + 4k$. This lets us apply Lemma 4.21 for $\mathcal{X} : P_j \rightsquigarrow T[\,y_j\,..\,y_{j+1}\,)$, which implies that $\mathcal{X}$ aligns $P[\pi_j^c]$ to $T[\tau_{i+j}^c]$ for all $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)\setminus C_S$ such that $j \in [\,0\,..\,m_c\,)$, completing the inductive argument.

We proceed to the proof of (2). First, consider $i' \in [\,0\,..\,i\,)$ and suppose, for a proof by contradiction, that $\tau_{i'}^c \ge t$. (1) implies $(\pi_0^c, \tau_i^c) \in \mathcal{X}$, so $\mathcal{X}$ aligns $P[\,0\,..\,\pi_0^c\,)$ onto $T[\,t\,..\,\tau_i^c\,)$ at cost at most k. By Definition 4.9(3), there exists an alignment $\mathcal{A}$ of cost at most w that aligns $P[\,0\,..\,\pi_0^c\,)$ onto $T[\,\hat{t}\,..\,\tau_i^c\,)$ for some $\hat{t} \in [\,\tau_{i-1}^{\mathrm{bc}(\mathbf{G}_S)-1}\,..\,\tau_i^0\,]$. Consequently, $|\hat{t} - t| \le w + k$. Since $t \le \tau_{i'}^c \le \tau_{i'}^{\mathrm{bc}(\mathbf{G}_S)-1} \le \tau_{i-1}^{\mathrm{bc}(\mathbf{G}_S)-1} \le \hat{t}$, we conclude $|T[\,\tau_{i'}^c\,..\,\tau_{i'}^{\mathrm{bc}(\mathbf{G}_S)-1}\,)| \le \hat{t} - t \le w + k$. Moreover, $|T[\,\tau_0^c\,..\,\tau_0^{\mathrm{bc}(\mathbf{G}_S)-1}\,)| \le |T[\,\tau_{i'}^c\,..\,\tau_{i'}^{\mathrm{bc}(\mathbf{G}_S)-1}\,)| + \delta_E(T[\,\tau_0^c\,..\,\tau_0^{\mathrm{bc}(\mathbf{G}_S)-1}\,), T[\,\tau_{i'}^c\,..\,\tau_{i'}^{\mathrm{bc}(\mathbf{G}_S)-1}\,)) \le w+k+2w = 3w+k$ and $|T[\,\tau_0^c\,..\,\tau_0^{\mathrm{bc}(\mathbf{G}_S)-1}\,]| \le 3w+k+1 \le 3w+2k$. Observe that for any string X, by only applying insertions/deletions, we have $\text{self-}\delta_E(X) \le 2|X|$. Consequently, $\text{self-}\delta_E(T[\,\tau_0^c\,..\,\tau_0^{\mathrm{bc}(\mathbf{G}_S)-1}\,]) \le 6w + 4k$, contradicting $c \notin C_S$.

The argument for $i' \in [\, i + m_c \,..\, n_c\,)$ is fairly similar. For a proof by contradiction, suppose that $\tau_{i'} < t'$. (1) implies $(\pi^c_{m_c-1}, \tau^c_{i+m_c-1}) \in \mathcal{X}$, so $\mathcal{X}$ aligns $P[\,\pi^c_{m_c-1} \,..\, |P|\,)$ onto $T[\,\tau^c_{i+m_c-1} \,..\, t'\,)$ at cost at most k. By Definition 4.9(5), there exists an alignment $\mathcal{A}$ of cost at most w that aligns $P[\,\pi^c_{m_c-1} \,..\, |P|\,)$ onto $T[\,\tau^c_{i+m_c-1} \,..\, \hat{t}\,)$ for some $\hat{t} \in [\,\tau^{c_{\text{last}}}_{i+m_0-1} \,..\, \tau^{c_{\text{last}}+1}_{i+m_0-1}\,]$. Consequently, $|\hat{t} - t'| \le w + k$. Now, we need to consider two cases.

First, suppose that $c \le c_{\text{last}}$ so that $i' \ge i + m_c = i + m_0$. Since $\hat{t} \le \tau^{c_{\text{last}}+1}_{i+m_0-1} \le \tau^0_{i'} \le \tau^c_{i'} < t'$, we conclude $|T[\,\tau^0_{i'} \,..\, \tau^c_{i'}\,)| \le t' - \hat{t} \le w + k$. Moreover, $|T[\,\tau^0_0 \,..\, \tau^c_0\,)| \le |T[\,\tau^0_{i'} \,..\, \tau^c_{i'}\,)| + \delta_E(T[\,\tau^0_0 \,..\, \tau^c_0\,), T[\,\tau^0_{i'} \,..\, \tau^c_{i'}\,)) \le w + k + 2w = 3w + k$ and $|T[\,\tau^0_0 \,..\, \tau^c_0\,]| \le 3w + k + 1 \le 3w + 2k$. Consequently, $\text{self-}\delta_E(T[\,\tau^0_0 \,..\, \tau^c_0\,]) \le 6w + 4k$, contradicting $c \notin C_S$.

Next, suppose that $c > c_{\text{last}}$ so that $i' \ge i + m_c = i + m_0 - 1$. Since $\hat{t} < \tau^{c_{\text{last}}+1}_{i+m_0-1} \le \tau^{c_{\text{last}}+1}_{i'} \le \tau^c_{i'} < t'$, we conclude $|T[\,\tau^{c_{\text{last}}+1}_{i'} \,..\, \tau^c_{i'}\,)| \le t' - \hat{t} \le w + k$. Moreover, $|T[\,\tau^{c_{\text{last}}+1}_0 \,..\, \tau^c_0\,)| \le |T[\,\tau^{c_{\text{last}}+1}_{i'} \,..\, \tau^c_{i'}\,)| + \delta_E(T[\,\tau^{c_{\text{last}}+1}_0 \,..\, \tau^c_0\,), T[\,\tau^{c_{\text{last}}+1}_{i'} \,..\, \tau^c_{i'}\,)) \le w + k + 2w = 3w + k$ and $|T[\,\tau^{c_{\text{last}}+1}_0 \,..\, \tau^c_0\,]| \le 3w + k + 1 \le 3w + 2k$. Consequently, $\text{self-}\delta_E(T[\,\tau^{c_{\text{last}}+1}_0 \,..\, \tau^c_0\,]) \le 6w + 4k$, contradicting $c \notin C_S$. ◀

4.5 Adding Candidate Positions to S

Lemma 4.25. *Let $\mathcal{Y} : P \rightsquigarrow T[\,t \,..\, t'\,)$ be an alignment of cost at most k. If $|\tau^0_i - t - \pi^0_0| > w + 2k$ holds for every $i \in [\,0 \,..\, n_0 - m_0\,]$, then $\mathrm{bc}(\mathbf{G}_{S \cup \{\mathcal{Y}\}}) \le \mathrm{bc}(\mathbf{G}_S)/2$.*

Proof. It suffices to show that there is no $c \in [\,0 \,..\, \mathrm{bc}(\mathbf{G}_S)\,)$ such that $\mathcal{Y}$ aligns $P[\pi^c_0]$ with $T[\tau^c_i]$ for some $i \in [\,0 \,..\, n_c\,)$. In this way, every black component of $\mathbf{G}_S$ is merged with another (black or red) component in $\mathbf{G}_{S \cup \{\mathcal{Y}\}}$, and thus the number of black components at least halves, i.e., $\mathrm{bc}(\mathbf{G}_{S'}) \le \mathrm{bc}(\mathbf{G}_S)/2$.

For a proof by contradiction, suppose that $(\pi^c_0, \tau^c_i) \in \mathcal{Y}$ for some $i \in [\,0 \,..\, n_c\,)$. We consider two cases.

If $i \in [\,0 \,..\, n_0 - m_0\,]$, we note that $\mathcal{Y}$ aligns $P[\,0 \,..\, \pi^c_0\,)$ to $T[\,t \,..\, \tau^c_i\,)$. Since $\mathcal{Y}$ has cost at most k, we have $|\tau^c_i - t - \pi^c_0| \le k$. At the same time, $\delta_E(P[\,\pi^0_0 \,..\, \pi^c_0\,), T[\,\tau^0_i \,..\, \tau^c_i\,)) \le w$ follows from Definition 4.9, which means that $|(\pi^c_0 - \pi^0_0) - (\tau^c_i - \tau^0_i)| \le w$. Combining the two inequalities, we derive $|\tau^0_i - t - \pi^0_0| \le w + k$. This contradicts the assumption that $|\tau^0_i - t - \pi^0_0| > w + 2k$ holds for every $i \in [\,0 \,..\, n_0 - m_0\,]$.

If $i \in (\,n_0 - m_0 \,..\, n_0\,)$, then we recall that S contains an alignment $\mathcal{X}$ mapping P onto a suffix of T, and $\mathcal{X}$ is an alignment of cost at most k that such that $(\pi^0_0, \tau^0_{n_0-m_0}), (\pi^c_0, \tau^c_{n_0-m_0}), (|P|, |T|) \in \mathcal{X}$. As a result,

$$|(\tau^c_{n_0-m_0} - \tau^0_{n_0-m_0}) - (\pi^c_0 - \pi^0_0)| + |(|T| - \tau^c_{n_0-m_0}) - (|P| - \pi^c_0)| \le k.$$

Since the alignment $\mathcal{Y}$ has cost at most k such that $(0, t), (\pi^c_0, \tau^c_i), (|P|, t') \in \mathcal{Y}$, we conclude that

$$|(\tau^c_i - t) - (\pi^c_0)| + |(t' - \tau^c_i) - (|P| - \pi^c_0)| \le k.$$

Combining these two inequalities, we derive

$$|(\tau^c_{n_0-m_0} - \tau^0_{n_0-m_0}) - (\pi^c_0 - \pi^0_0) - (\tau^c_i - t) + (\pi^c_0)| + |(|T| - \tau^c_{n_0-m_0}) - (|P| - \pi^c_0) - (t' - \tau^c_i) + (|P| - \pi^c_0)| \le 2k,$$

which simplifies to

$$|(\tau^c_{n_0-m_0} - \tau^0_{n_0-m_0}) + \pi^0_0 - (\tau^c_i - t)| + |(|T| - \tau^c_{n_0-m_0}) - (t' - \tau^c_i)| \le 2k.$$

Since $t' \le |T|$ and $\tau^c_i \ge \tau^c_{n_0-m_0}$, we have

$$\begin{aligned}
|\tau^0_{n_0-m_0} - t - \pi^0_0| &= |(\tau^c_{n_0-m_0} - \tau^0_{n_0-m_0}) + \pi^0_0 - (\tau^c_i - t) + (\tau^c_i - \tau^c_{n_0-m_0})| \\
&\le |(\tau^c_{n_0-m_0} - \tau^0_{n_0-m_0}) + \pi^0_0 - (\tau^c_i - t)| + |\tau^c_i - \tau^c_{n_0-m_0}| \\
&= |(\tau^c_{n_0-m_0} - \tau^0_{n_0-m_0}) + \pi^0_0 - (\tau^c_i - t)| + \tau^c_i - \tau^c_{n_0-m_0} \\
&\le |(\tau^c_{n_0-m_0} - \tau^0_{n_0-m_0}) + \pi^0_0 - (\tau^c_i - t)| + \tau^c_i - \tau^c_{n_0-m_0} + |T| - t' \\
&= |(\tau^c_{n_0-m_0} - \tau^0_{n_0-m_0}) + \pi^0_0 - (\tau^c_i - t)| + |\tau^c_i - \tau^c_{n_0-m_0} + |T| - t'| \\
&= |(\tau^c_{n_0-m_0} - \tau^0_{n_0-m_0}) + \pi^0_0 - (\tau^c_i - t)| + |(|T| - \tau^c_{n_0-m_0}) - (t' - \tau^c_i)| \\
&\le 2k.
\end{aligned}$$

This contradicts the assumption that $|\tau^0_{n_0-m_0} - t - \pi^0_0| > w + 2k$. ◾

4.6 Recovering all k-edit Occurrences of P in T

In this (sub)section we drop the assumption that S encloses T and that $\mathrm{bc}(\mathbf{G}_S) > 0$. Every time we assume that S encloses T or that $\mathrm{bc}(\mathbf{G}_S) > 0$, we will explicitly mention it.

Definition 4.26. *Let S be a set of k-edit alignments of P onto fragments of T and let $T[\,t \mathinner{..} t'\,)$ be a k-error occurrence of P in T. We say that S* captures *$T[\,t \mathinner{..} t'\,)$ if S encloses T and exactly one of the two following holds:*

- $\mathrm{bc}(\mathbf{G}_S) = 0$; *or*
- $\mathrm{bc}(\mathbf{G}_S) > 0$ *and* $|\tau^0_i - t - \pi^0_0| \le w + 3k$ *holds for some* $i \in [\,0 \mathinner{..} n_0\,)$. ◾

Theorem 4.27. *Let S be a set of k-edit alignments of P onto fragments of T such that S encloses T and $\mathrm{bc}(\mathbf{G}_S) > 0$. Construct $P^\#$ and $T^\#$ by replacing, for every $c \notin C_S$, every character in the c-th black component with a unique character $\#_c$. Then, $\mathcal{X} : P \rightsquigarrow T[\,t \mathinner{..} t'\,)$ is an optimal alignment of cost $k' \le k$ if and only if $\mathcal{X} : P^\# \rightsquigarrow T^\#[\,t \mathinner{..} t'\,)$ is an optimal alignment of cost $k' \le k$. Moreover, for such $\mathcal{X}$ it holds $\mathsf{E}_{P,T}(\mathcal{X}) = \mathsf{E}_{P^\#,T^\#}(\mathcal{X})$.*

Proof. Suppose $\mathcal{X} : P \rightsquigarrow T[\,t \mathinner{..} t'\,)$ is an optimal alignment of cost $k' \le k$. We first prove that also $\delta^{\mathcal{X}}_E(P^\#, T^\#[\,t \mathinner{..} t'\,)) = k'$. Note, since S encloses T and $\mathrm{bc}(\mathbf{G}_S) > 0$, there exists $i \in [\,0 \mathinner{..} n_0\,)$ such that $|\tau^0_i - t - \pi^0_0| \le w + 3k$, and Proposition 4.24 applies to $\mathcal{X}$. For $c \in [\,0 \mathinner{..} \mathrm{bc}(\mathbf{G}_S)\,)$ we have that $\#_c$ appears in $P^\#$ at the positions $\Pi = \{\pi^c_j : j \in [\,0 \mathinner{..} m_c\,)\}$ and in $T^\#[\,t \mathinner{..} t'\,)$ at the positions $\mathcal{T} = \{\tau^c_j : j \in [\,i \mathinner{..} i + m_c\,)\}$ (this follows from Proposition 4.24). Consequently, the difference in cost between $\mathcal{X} : P \rightsquigarrow T[\,t \mathinner{..} t'\,)$ and $\mathcal{X} : P^\# \rightsquigarrow T^\#[\,t \mathinner{..} t'\,)$ is determined by the subset $\{(x, y) : \mathcal{X}$ aligns $P[x]$ to $P[y], x \in \Pi$ or $y \in \mathcal{T}\} \subseteq \mathcal{X}$. Now, from Proposition 4.24 also follows

$$\{(x, y) : \mathcal{X} \text{ aligns } P[x] \text{ to } T[y], x \in \Pi \vee y \in \mathcal{T}\} = \{(\pi^c_j, \tau^c_{i+j}) : c \in [\,0 \mathinner{..} \mathrm{bc}(\mathbf{G}_S)\,), j \in [\,0 \mathinner{..} m_c\,)\}.$$

Since for all $c \in [\,0 \mathinner{..} \mathrm{bc}(\mathbf{G}_S)\,)$ and $j \in [\,0 \mathinner{..} m_c\,)$ we have that $P[\pi^c_j] = T[\tau^c_j]$ and $P^\#[\pi^c_j] = T^\#[\tau^c_j]$, we conclude that $\mathcal{X} : P \rightsquigarrow T[\,t \mathinner{..} t'\,)$ and $\mathcal{X} : P^\# \rightsquigarrow T^\#[\,t \mathinner{..} t'\,)$ share the same cost k'.

Now, notice that also $\mathcal{X} : P^\# \rightsquigarrow T^\#[\,t \mathinner{..} t'\,)$ is optimal because for all $x \in [\,0 \mathinner{..} |P|\,)$ and $y \in [\,0 \mathinner{..} |T|\,)$, $P^\#[x] = T^\#[y]$ implies $P[x] = T[y]$. Hence, for all $y, y' \in [\,0 \mathinner{..} |T|\,)$ we have $\delta_E(P, T[\,y \mathinner{..} y'\,)) \le \delta_E(P^\#, T^\#[\,y \mathinner{..} y'\,))$, and if $\mathcal{X} : P^\# \rightsquigarrow T^\#[\,t \mathinner{..} t'\,)$ were not to be optimal we would have a contradiction with the optimality of $\mathcal{X} : P \rightsquigarrow T[\,t \mathinner{..} t'\,)$.

Conversely, suppose $\mathcal{X} : P^\# \rightsquigarrow T^\#[\,t \mathinner{..} t'\,)$ is optimal of cost k'. Then, for the same reason of above $\delta^{\mathcal{X}}_E(P, T[\,t \mathinner{..} t'\,)) \le \delta^{\mathcal{X}}_E(P^\#, T^\#[\,t \mathinner{..} t'\,))$. We actually must have $\delta^{\mathcal{X}}_E(P, T[\,t \mathinner{..} t'\,)) = \delta^{\mathcal{X}}_E(P^\#, T^\#[\,t \mathinner{..} t'\,))$ and that $\mathcal{X}$ is optimal. Otherwise, there would exist an optimal alignment $\mathcal{Y} : P \rightsquigarrow T[\,t \mathinner{..} t'\,)$ of cost at most $k' - 1$. Following what we have proved above, $\mathcal{Y} : P^\# \rightsquigarrow T^\#[\,x \mathinner{..} x'\,)$ would have cost at most $k - 1$, and we again reach a contradiction with the optimality of $\mathcal{X}$.

For the last part of the statement, it suffices to notice that all characters that are substituted in P, T (or equivalently all the hashes in $P^\#, T^\#$) are always matched by such $\mathcal{X}$. Since the characters that are matched by $\mathcal{X}$ are stored explicitly neither in $\mathsf{E}_{P,T}(\mathcal{X})$ nor in $\mathsf{E}_{P^\#,T^\#}(\mathcal{X})$, we conclude $\mathsf{E}_{P,T}(\mathcal{X}) = \mathsf{E}_{P^\#,T^\#}(\mathcal{X})$. ◀

4.7 Communication Complexity of Edit Distance

Main Theorem 1. *The Pattern Matching with Edits problem admits a one-way deterministic communication protocol that sends $\mathcal{O}(n/m \cdot k \log^2 m)$ bits. Within the same communication complexity, one can also encode the family of all fragments of $T[\,i\,..\,j\,)$ that satisfy $\delta_E(P, T[\,i\,..\,j\,)) \le k$, as well as all optimal alignments $P \rightsquigarrow T[\,i\,..\,j\,)$ for each of these fragments. Further, increasing the communication complexity to $\mathcal{O}(n/m \cdot k \log m \log(m|\Sigma|))$, where Σ denotes the input alphabet, one can also retrieve the edit information for each optimal alignment.*

Proof. We first prove the second part of the theorem, i.e., we describe a protocol allowing Alice to encode using $\mathcal{O}(n/m \cdot k \log m \log(m|\Sigma|))$ bits the edit information for all alignments in

$$S' := \{\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,) : t, t' \in [\,0\,..\,n\,], \delta_E^{\mathcal{X}}(P, T[\,t\,..\,t'\,)) = \delta_E(P, T[\,t\,..\,t'\,)) \le k\}.$$

We may assume $k \le m/4$; otherwise, it suffices to send P and T to Bob. We want to argue that we may further restrict ourselves to the case where S' encloses T.

Claim 4.28. *If such a protocol using $\mathcal{O}(k \log m \log(m|\Sigma|))$ bits exists in the case where S' encloses T, then such a protocol also exists for the general case.*

Proof. Divide T into $\mathcal{O}(n/m)$ contiguous blocks of length $m - 2k \ge m/2$ (with the last block potentially being shorter). For the i-th block starting at position $b_i \in [\,0\,..\,|T|\,)$, define $B_i = [\,b_i\,..\,b_i{+}2m{-}2k\,] \cap [\,0\,..\,|T|\,]$ and $S_i = \{\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,) : t, t' \in B_i, \delta_E^{\mathcal{X}}(P, T[\,t\,..\,t'\,)) = \delta_E(P, T[\,t\,..\,t'\,)) \le k\}$. Since $S' = \bigcup_i S_i$, it suffices to show that there exists such a protocol using $\mathcal{O}(k \log m \log(m|\Sigma|))$ bits for the case where instead of T we consider an arbitrary B_i.

Therefore, let i be arbitrary. If $S_i = \varnothing$, then Alice does not need to send anything. On the other hand, if $|S_i| = 1$, then Alice directly sends to Bob the edit information of the unique alignment contained in S_i using $\mathcal{O}(k \log(m|\Sigma|))$ bits. Lastly, if $|S_i| \ge 2$, Alice uses the protocol from the assumption of the claim on S_i and $T[\,\ell_i\,..\,r_i\,)$, where $\ell_i = \min\{t : \mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,) \in S_i\}$ and $r_i = \max\{t' : \mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,) \in S_i\}$. This is indeed possible, as S_i encloses $T[\,\ell_i\,..\,r_i\,)$ because $|B_i| - 1 \le 2m - 2k$ and $|S_i| \ge 2$. ◿

Henceforth, suppose S' encloses T. We construct a subset $S \subseteq S'$ that captures all k-error occurrences as follows. First, we add to S the alignments $\mathcal{X}_{\mathsf{pref}}, \mathcal{X}_{\mathsf{suf}} \in S'$ such that $(0, 0) \in \mathcal{X}_{\mathsf{pref}}$ and $(|P|, |T|) \in \mathcal{X}_{\mathsf{suf}}$. This ensures that S encloses T. Next, we iteratively add alignments to S according to the following rules:

- if S captures all k-error occurrences, then we stop;
- otherwise, $\mathrm{bc}(\mathbf{G}_S) > 0$, and we add to S an arbitrary alignment $\mathcal{Y} : P \rightsquigarrow T[\,t\,..\,t'\,)$ of cost at most k such that $|\tau_i^0 - t - \pi_0^0| > w + 2k$ holds for every $i \in [\,0\,..\,n_0 - m_0\,]$ ($w = \mathcal{O}(k|S|)$ denotes the total weight of the weight function from Lemma 4.15).

Note that we never apply the second rule more than $\mathcal{O}(\log n)$ times: from Lemma 4.25 follows that every time we add $\mathcal{Y}$ to S, $\mathrm{bc}(\mathbf{G}_{S \cup \{\mathcal{Y}\}}) \le \mathrm{bc}(\mathbf{G}_S)/2$, and if $\mathrm{bc}(\mathbf{G}_S) = 0$, then the first rule applies. Consequently, $|S| = \mathcal{O}(\log n)$.

Now, if $\mathrm{bc}(\mathbf{G}_S) = 0$, then the information that Alice sends to Bob is $\{\mathsf{E}_{P,T}(\mathcal{X}) : \mathcal{X} : P \rightsquigarrow T \in \mathcal{X}\}$. As already argued before, this allows Bob to fully reconstruct P and T.

Otherwise, if $\mathrm{bc}(\mathbf{G}_S) > 0$, then Alice additionally sends the encoding of $\{(c, T[\tau_0^c]) : c \in C_S\}$ to Bob, where C_S is the black cover from Lemma 4.15. This allows Bob to construct the strings $P^\#$ and $T^\#$ from

Theorem 4.27. Theorem 4.27 guarantees that by computing the answer over $P^\#$ and $T^\#$ instead of P and T, Bob still obtains the information required by the protocol.

Note, $\{\mathsf{E}_{P,T}(\mathcal{X}) : \mathcal{X} : P \rightsquigarrow T[\,t \mathinner{.\,.} t'\,) \in S\}$ and $\{(c, T[\tau_0^c]) : c \in C_S\}$ can be stored using $O(w + k|S|) = O(k \log n)$ space. Since a space unit consists of either a position of P/T, or a character from Σ, we conclude that the second part of the theorem holds.

Lastly, we prove the first part of the theorem. If there is no requirement to send the edit information for every $\mathcal{X} \in S$, we encode the characters in P to an alphabet Σ' with a size of at most $m + 1$. This new alphabet includes an additional special character representing all characters in T that are not present in P. By mapping all characters of P to T to the corresponding characters of Σ', S' and all alignments contained in S' remain unchanged. Thereby, we reduce the number of bits needed by the protocol to $O(n/m \cdot k \log^2 m)$. ◾

Main Theorem 2. *Fix integers n, m, k such that $n/2 \geq m > k > 0$. Every communication protocol for the Pattern Matching with Edits problem uses $\Omega(n/m \cdot k \log(m/k))$ bits for $P = \mathtt{0}^m$ and some $T \in \{\mathtt{0}, \mathtt{1}\}^n$.*

Proof. Let $p = \lfloor n/(2m-2) \rfloor$ and $T = S_0 \cdot S_0 \cdot S_1 \cdot S_1 \cdots S_{p-1} \cdot S_{p-1} \cdot \mathtt{0}^{n-p(2m-2)}$, where $S_0, \ldots, S_{p-1} \in \{\mathtt{0}, \mathtt{1}\}^{m-1}$ are strings that contain exactly k copies of $\mathtt{1}$ and $m-1-k$ copies of $\mathtt{0}$. We shall prove that, for every $q \in [\,0 \mathinner{.\,.} p\,)$ and $i \in [\,0 \mathinner{.\,.} m-1\,)$, we have $S_q[i] = \mathtt{0}$ if and only if $q(2m-2) + i \in \mathrm{Occ}_k^E(P, T)$. Consequently, T can be recovered from $\mathrm{Occ}_k^E(P, T)$, and the theorem follows because the number of possibilities for T is $\binom{m-1}{k}^p = 2^{\Omega(n/m \cdot k \log(m/k))}$.

If $S_q[i] = \mathtt{0}$, then $T[\,q(2m-2) + i \mathinner{.\,.} q(2m-2) + i + m\,) = S_q[\,i \mathinner{.\,.} m-1\,) \cdot S_q[\,0 \mathinner{.\,.} i\,]$ contains exactly k copies of $\mathtt{1}$ and $m - k$ copies of $\mathtt{0}$, so $q(2m-2) + i \in \mathrm{Occ}_k^E(P, T)$.

Similarly, if $S_q[i] = \mathtt{1}$, then $T[\,q(2m-2) + i \mathinner{.\,.} q(2m-2) + i + m\,)$ contains exactly $k+1$ copies of $\mathtt{1}$ and $m-k-1$ copies of $\mathtt{0}$. For $s \geq 0$, every alignment $P \rightsquigarrow T[\,q(2m-2)+i \mathinner{.\,.} q(2m-2)+i+m+s\,)$ makes at least $k+1$ insertions or substitutions (to account for $\mathtt{1}$s). Every alignment $P \rightsquigarrow T[\,q(2m-2)+i \mathinner{.\,.} q(2m-2)+i+m-s\,)$, on the other hand, makes at least s deletions (to account for the length difference) and at least $k+1-s$ insertions or substitutions (to account for $\mathtt{1}$s). The overall number of edits is at least $k+1$ in either case, so $q(2m-2) + i \notin \mathrm{Occ}_k^E(P, T)$. ◾

5 Quantum Algorithms on Strings

We assume the input string $S \in \Sigma^n$ can be accessed in a quantum query model [Amb04, BdW02]: there is an input oracle O_S that performs the unitary mapping $O_s \colon |i, b\rangle \mapsto |i, b \oplus S[i]\rangle$ for any index $i \in [\,0 \mathinner{.\,.} n\,)$ and any $b \in \Sigma$.

The *query complexity* of a quantum algorithm (with 2/3 success probability) is the number of queries it makes to the input oracles. The *time complexity* of the quantum algorithm additionally counts the elementary gates [BBC+95] for implementing the unitary operators that are independent of the input. Similar to prior works [GS23, AJ23, Amb04], we assume quantum random access quantum memory.

We say an algorithm succeeds *with high probability (w.h.p)*, if the success probability can be made at least $1 - 1/n^c$ for any desired constant $c > 1$. A bounded-error algorithm can be boosted to succeed w.h.p. by $O(\log n)$ repetitions. In this paper, we do not optimize sub-polynomial factors of the quantum query complexity (and time complexity) of our algorithms.

For our quantum algorithms, we rely on the following primitive quantum operation.

Theorem 5.1 (Grover search (Amplitude amplification) [Gro96, BHMT02]). *Let $f : [0\mathinner{.\,.} n) \to \{0,1\}$ denote a function, where $f(i)$ for each $i \in [0\mathinner{.\,.} n)$ can be evaluated in time T. There is a quantum algorithm that, with high probability and in time $\tilde{O}(\sqrt{n} \cdot T)$, finds an $x \in f^{-1}(1)$ or reports that $f^{-1}(1)$ is empty.*

In designing lower bounds for our quantum algorithm, we rely on the following framework.

Theorem 5.2 (Adversary method [Amb02]). *Let $f(x_0, \ldots, x_{n-1})$ be a function of n binary variables and $X, Y \subseteq \{0,1\}^n$ be two sets of inputs such that $f(x) \neq f(y)$ if $x \in X$ and $y \in Y$. Let $R \subseteq X \times Y$ be such that:*

(1) *For every $x \in X$, there exist at least m different $y \in Y$ such that $(x,y) \in R$.*
(2) *For every $y \in Y$, there exist at least m' different $x \in X$ such that $(x,y) \in R$.*
(3) *For every $x \in X$ and $i \in [0\mathinner{.\,.} n)$, there is at most one $y \in Y$ such that $(x,y) \in R$ and $x_i \neq y_i$.*
(4) *For every $y \in X$ and $i \in [0\mathinner{.\,.} n)$, there is at most one $x \in X$ such that $(x,y) \in R$ and $x_i \neq y_i$.*

Then, any quantum algorithm computing f uses $\Omega\left(\sqrt{mm'}\right)$ queries

Throughout this paper, we build upon several existing quantum algorithms for string processing. We summarize below the key techniques and results that are relevant to our work.

5.1 Essential Quantum Algorithms on Strings

Theorem 5.3 (Quantum Exact Pattern Matching [HV03]). *There is an $\tilde{O}(\sqrt{n})$-time quantum algorithm that, given a pattern P of length m and a text T of length $n \geq m$, finds an exact occurrence of P in T (or reports that P does not occur in T).*

Kociumaka, Radoszewski, Rytter, and Waleń [KRRW15] established that the computation of $\mathrm{per}(S)$ can be reduced to $O(\log |S|)$ instances of exact pattern matching and longest common prefix operations involving substrings of S. Consequently, the following corollary holds.

Corollary 5.4 (Finding Period). *There is an $\tilde{O}(\sqrt{n})$-time quantum algorithm that, given a string S of length n, computes $\mathrm{per}(S)$.*

We modify the algorithm of [HV03] such that it outputs the whole set $\mathrm{Occ}(P,T)$.

Lemma 5.5. *There is an $\tilde{O}(\sqrt{\mathsf{occ} \cdot n})$-time quantum algorithm that, given a pattern P of length m and a text T of length $n \geq m$, outputs $\mathrm{Occ}(P,T)$, where $\mathsf{occ} = |\mathrm{Occ}(P,T)|$.*

Proof. Let $t_1 < t_2 < \cdots < t_{\mathsf{occ}}$ denote the positions that are contained in $\mathrm{Occ}(P,T)$.

Suppose that we have just determined t_i for some $i \in [0\mathinner{.\,.} \mathsf{occ})$ (if $i = 0$, then we set $t_0 = -1$). To find t_{i+1} we use Theorem 5.3 combined with exponential search. More specifically, at the j-th jump of the exponential search, we use Theorem 5.3 to check whether $\mathrm{Occ}(P, T[t_i + 1\mathinner{.\,.} \min(\sigma_i + 2^{j-1}, |T|)]) \neq \emptyset$. Once we have found the first j for which this set is non-empty, we use Theorem 5.3 combined with binary search to find the exact position of t_{i+1}. By doing so, we use the algorithm of Theorem 5.3 as subroutine at most $O(\log(t_{i+1} - t_i)) = O(\log n)$ times, each time requiring at most $\tilde{O}(\sqrt{t_{i+1} - t_i})$ time.

As a consequence, finding all $t_1, \ldots, t_{\mathsf{occ}}$ requires $\tilde{O}(\sum_{i=0}^{\mathsf{occ}-1} \sqrt{t_{i+1} - t_i})$ time. By the Cauchy–Schwarz inequality, $\sum_{i=0}^{\mathsf{occ}-1} \left(1 \cdot \sqrt{t_{i+1} - t_i}\right) \leq \sqrt{\mathsf{occ}} \cdot \sqrt{t_{\mathsf{occ}} - t_0} \leq \sqrt{\mathsf{occ} \cdot n}$, so the total time complexity is $\tilde{O}(\sqrt{\mathsf{occ} \cdot n})$.

Another useful tool is the following quantum algorithm to compute the edit distance of two strings.

Theorem 5.6 (Quantum Bounded Edit Distance [GJKT24, Theorem 1.1])**.** *There is a quantum algorithm that, given quantum oracle access to strings X, Y of length at most n, computes their edit distance $k := \delta_E(X, Y)$, along with a sequence of k edits transforming X into Y. The algorithm has a query complexity of $\tilde{O}(\sqrt{n+kn})$ and a time complexity of $\tilde{O}(\sqrt{n+kn}+k^2)$.*

The algorithm of Theorem 5.6 returns lists edits from left to right, with each edit specifying the positions and the characters involved (one for insertions and deletions, two for substitutions). In the following corollary, we show that this representation can be easily transformed into the edit information $\mathsf{E}_{X,Y}(\mathcal{A})$ of some optimal alignment $\mathcal{A} : X \rightsquigarrow Y$.

Corollary 5.7. *There is a quantum algorithm that, given quantum oracle access to strings X, Y of length at most n, computes their edit distance $k := \delta_E(X, Y)$ and the edit information $\mathsf{E}_{X,Y}(\mathcal{A})$ of some optimal alignment $\mathcal{A} : X \rightsquigarrow Y$. The algorithm has the same query and time complexity as in Theorem 5.6.*

Proof. We use Theorem 5.6 to retrieve the sequence of k edits, and then we construct $\mathsf{E} := \mathsf{E}_{X,Y}(\mathcal{A})$ as follows. We iterate through the returned list of edits in the reverse (right-to-left) order and add elements to E according to the following rules, maintaining a pair (x', y') initialized as $(|X|, |Y|)$.

- If the current edit substitutes $X[x]$ with $Y[y]$, we set $(x', y') := (x, y)$ and add $(x', X[x], y', Y[y])$ to E.
- If the current edit inserts $Y[y]$, we set $(x', y') := (x' + y - y' + 1, y)$ and add $(x', \varepsilon, y', Y[y])$ to E.
- Lastly, if the current edit deletes $X[x]$, we set $(x', y') := (x, y' + x - x' + 1)$ and add $(x', X[x], y', \varepsilon)$ to E.

Observe that we can reconstruct the underlying alignment $\mathcal{A}$ by a similar iterative scheme used to construct E: just before setting (x', y') to (x'', y''), we add to $\mathcal{A}$ all pairs of the form $(x' - i, y' - i)$ with $i \in [0\,..\,\min(x' - x'', y' - y''))$. Moreover, after processing the entire edit sequence, we add to $\mathcal{A}$ all the pairs of the form $(x' - i, y' - i)$ with $i \in [0\,..\,x']$; we then have $x' = y'$. It is easy to verify at this point that the following construction of $\mathcal{A}$ leads to a correctly defined alignment of cost k such that $\mathsf{E} = \mathsf{E}_{X,Y}(\mathcal{A})$.

Lastly, note that the construction of E requires at most $O(k)$ additional time and no queries.

From the work that established Theorem 5.6, we also obtain an algorithm for computing the LZ factorization of a string.

Theorem 5.8 (Quantum LZ77 factorization [GJKT24, Theorem 1.2])**.** *There is a quantum algorithm that, given quantum oracle access to a string X of length n and an integer threshold $z \geq 1$, decides whether $|\mathsf{LZ}(X)| \leq z$ and, if so, computes the LZ factorization $\mathsf{LZ}(X)$. The algorithm uses $\tilde{O}(\sqrt{zn})$ query and time complexity.*

Corollary 5.9. *There is a quantum algorithm that, given quantum oracle access to a string X of length n and an integer threshold $k \geq 1$, decides whether $\mathsf{self}\text{-}\delta_E(X) \leq k$ using $\tilde{O}(\sqrt{kn})$ queries and $\tilde{O}(\sqrt{kn} + k^2)$ time.*

Proof. First, we apply Theorem 5.8 with $z = 2k$. If $|\mathsf{LZ}(X)| > 2k$, we return NO. Otherwise, we check $\mathsf{self}\text{-}\delta_E(X) \leq k$ as follows.

We construct a classical data structure for LCE (Longest Common Extension) queries in X following the method outlined in [I17]. This data structure can be built in $\tilde{O}(k)$ time and supports LCE queries in $O(\log z)$ time, where $z = |\mathsf{LZ}(X)|$.

We then employ an algorithm, described in Lemma 4.5 of [CKW23], which is a simple modification of the classic Landau-Vishkin algorithm [LV88]. This algorithm requires only a blackbox for computing LCE queries. By using this algorithm, we can check whether $\mathsf{self}\text{-}\delta_E(X) \leq k$ using $O(k^2)$ LCE queries to the data structure, thus requiring $\tilde{O}(k^2)$ time.

Regarding the correctness anaylsis, if $|\mathsf{LZ}(X)| > 2k$, then according to Lemma 3.7, $\mathsf{self}\text{-}\delta_E(X) > k$. Otherwise, the correctness follows from the correctness of the algorithm presented in [CKW23].

5.2 Quantum Algorithm for Edit Distance Minimizing Suffix

In our Quantum Pattern Matching with Edits algorithm, there is a recurring need for a procedure that identifies a suffix of a text that minimizes the edit distance to a specific pattern. To this end, we define

$$\delta_E(X, \circ Y) := \min_{y \in [0\,..\,|Y|]} \delta_E(X, Y[\,y\,..\,|Y|\,]).$$

Similarly, we define $\delta_E(X, Y\circ) := \min_{y \in [0\,..\,|Y|]} \delta_E(X, Y[\,0\,..\,y\,))$ for prefixes. In this (sub)section, we discuss a quantum subroutine that calculates $\delta_E(X, \circ Y)$. Note, by reversing the order of its input strings, the same subroutine computes $\delta_E(X, Y\circ)$.

Lemma 5.10. *Let T denote a text of length n, let P denote a pattern of length m, and let k denote a positive threshold. Then, we can check whether $\delta_E(P, \circ T) \le k$ (and, if this is the case, calculate the value of $\delta_E(P, \circ T)$ and the suffix of T minimizing the edit distance) in $\tilde{O}(\sqrt{km} + k^2)$ quantum time using $\tilde{O}(\sqrt{km} + k)$ quantum queries.*

Proof. Our algorithm relies on the following combinatorial claim.

Claim 5.11. *Let \$ denote a character that occurs in neither P nor T. If $n \le m + k$ and $\delta_E(P, \circ T) \le k$, then $\delta_E(\$^{2k}P, T) = \delta_E(P, \circ T) + 2k$. Moreover, if an optimum alignment $\mathcal{A} : \$^{2k}P \rightsquigarrow T$ substitutes exactly s among \$ characters, then $\delta_E(P, \circ T) = \delta_E(P, T[\,s\,..\,n\,))$.*

Proof. Consider a position $i \in [\,0\,..\,n\,]$ such that $\delta_E(P, T[\,i\,..\,n\,)) = \delta_E(P, \circ T) \le k$. Observe that $|m-(n-i)| \le k$, so the assumption $n \le m+k$ implies $i \le k+n-m \le 2k$. Consequently, $\delta_E(\$^{2k}, T[\,0\,..\,i\,)) \le \max(2k, i) = 2k$ and $\delta_E(\$^{2k}P, T) \le \delta_E(\$^{2k}, T[\,0\,..\,i\,)) + \delta_E(P, T[\,i\,..\,n\,)) \le 2k + \delta_E(P, \circ T)$.

Next, consider an optimal alignment $\mathcal{A} : \$^{2k}P \rightsquigarrow T$ and a position $i \in [\,0\,..\,n\,]$ such that $(2k, i) \in \mathcal{A}$. Note that $\delta_E(\$^{2k}P, T) = \delta_E^{\mathcal{A}}(\$^{2k}, T[\,0\,..\,i\,)) + \delta_E^{\mathcal{A}}(P, T[\,i\,..\,n\,)) \ge 2k + \delta_E(P, \circ T)$; this is because \$ does not occur in T (so its every occurrence needs to be involved in an edit) and due to the definition of $\delta_E(P, \circ T)$.

Since $\delta_E(T, \$^{2k}P) \le 2k + \delta_E(P, \circ T)$, we have $\delta_E^{\mathcal{A}}(T[\,0\,..\,i\,), \$^{2k}) = 2k$ and $\delta_E^{\mathcal{A}}(T[\,i\,..\,n\,), P) = \delta_E(P, \circ T)$. In particular, $\mathcal{A}$ substitutes exactly $s = i$ characters \$, and $\delta_E(P, \circ T) = (T[\,s\,..\,n\,), P)$ holds.

First, observe that we can assume that $n \le m + k$. Otherwise, we can truncate T to $T[\,n - m - k\,..\,n\,)$ since the edit distance between all suffixes of T that we left out and the pattern P exceeds k.

Following Claim 5.11, we use Theorem 5.6 to check whether $\delta_E(T, \$^{2k}P) \le 3k$. If that is not the case, we report that $\delta_E(P, \circ T) > k$. Otherwise, we retrieve $\delta_E(T, \$^{2k}P)$ along with an optimal sequence of edits transforming T into $\$^{2k}P$. We report $\delta_E(T, \$^{2k}P) - 2k$ as the distance $\delta_E(P, \circ T)$ and $T[\,s\,..\,n\,)$, where s is the number of characters \$ that $\mathcal{A}$ substitutes, as the minimizing suffix.

The correctness of this approach follows directly from Claim 5.11. The complexity is dominated by the application of Theorem 5.6: $\tilde{O}(\sqrt{km} + k^2)$ time and $\tilde{O}(\sqrt{km} + k)$ queries.

Given a string Q and a threshold k, Lemma 5.10 also allows us to check if $\delta_E(P, {}^*Q) \le k$ holds, and to return $\delta_E(P, {}^*Q)$ together with the substring of Q^∞ that minimizes $\delta_E(P, {}^*Q)$ using $\tilde{O}(\sqrt{kn} + k^2)$ time and $\tilde{O}(\sqrt{kn})$ queries. In fact, it is easy to verify that $\delta_E(P, {}^*Q) = \delta_E(Q^\infty[\,q - |P| - k\,..\,q\,), \circ P)$, where $q = |Q| \cdot \lceil (|P| + k)/|Q| \rceil$. Similar consideration holds for $\delta_E(P, Q^*)$.

5.3 Quantum Algorithm for Gap Edit Distance

In this section we will show how to devise a quantum subroutine for the Gap Edit Distance problem which is defined as follows.

(β,α)-Gap Edit Distance **Input:** strings X, Y of length $\lvert X\rvert = \lvert Y\rvert = n$, and integer thresholds $\alpha \geq \beta \geq 0$. **Output:** yes if $\delta_E(X,Y) \leq \beta$, no if $\delta_E(X,Y) > \alpha$, and an arbitrary answer otherwise.

We formally prove the following.

Lemma 5.12. *For every positive integer n, there exists a positive integer $\ell = O(n^{1+o(1)})$ and a function $f : \{0,1\}^\ell \times \Sigma^n \times \Sigma^n \to \{\text{yes}, \text{no}\}$ such that:*

- *if* seed $\in \{0,1\}^\ell$ *is sampled uniformly at random, then, for every $X, Y \in \Sigma^n$, the value $f(\text{seed}, X, Y)$ is with high probability a correct answer to the $(k, kn^{o(1)})$-Gap Edit Distance instance with input X and Y;*
- *there exists a quantum algorithm that computes $f(\text{seed}, X, Y)$ in $O(n^{1+o(1)})$ quantum time using $O(n^{\frac{1}{2}+o(1)})$ queries providing quantum oracle access to $X, Y \in \Sigma^n$ and* seed $\in \{0,1\}^\ell$.

The quantum algorithm developed for the gap edit distance problem is a direct adaptation of Goldenberg, Kociumaka, Krauthgamer, and Saha's classical algorithm [GKKS22] for the same problem. The paper [GKKS22] presents several algorithms for the gap edit problem, each progressively improving performance. Among these algorithms, we opt to adapt the one from Section 4, stopping short of the faster implementation discussed in Section 5.

Notably, all algorithms presented in [GKKS22] are non-adaptive, meaning they do not require access to the string to determine which recursive calls to make. This characteristic allows us to pre-generate randomness using a sequence of bits from $\{0,1\}^\ell$. Consequently, when provided with a fixed sequence of bits, the algorithm's output becomes deterministic, producing the same output not only for cases where $\delta_E(X,Y) \leq \beta$ and $\delta_E(X,Y) > \alpha$ but also for scenarios where $\beta < \delta_E(X,Y) \leq \alpha$.

In Section 5.3.1, we will first briefly present the algorithm of [GKKS22]. Then, in Section 5.3.2, we argue how the algorithm can be extended to the quantum framework. For a more rigorous proof, an intuition for why the algorithm works, and a more comprehensive introduction to the problem, we recommend referring directly to [GKKS22].

5.3.1 Gap Edit Distance and Shifted Gap Edit Distance

The algorithm presented in [GKKS22] solves an instance of Gap Edit Distance by querying an oracle solving smaller instances of a similar problem: the Shifted Gap Edit Distance problem. Before properly defining this problem, we need to introduce the notion of *shifted edit distance*.

Definition 5.13. *Given two strings X, Y and a threshold β, define the β-*shifted edit distance *$\delta_E^\beta(X,Y)$ as*

$$\delta_E^\beta(X,Y) := \min\left(\bigcup_{\Delta=0}^{\min(|X|,|Y|,\beta)} \{\delta_E(X[\Delta\mathinner{.\,.}|X|\,), Y[\,0\mathinner{.\,.}|Y|-\Delta\,)), \delta_E(X[\,0\mathinner{.\,.}|X|-\Delta\,)), Y[\,\Delta\mathinner{.\,.}|Y|\,)\}\right).$$

The Shifted Gap Edit Distance problem consists in either verifying that the shifted edit distance between two strings X, Y is small, or that the (regular) edit distance between X and Y is large. More formally:

Algorithm 1 Randomized Algorithm reducing Gap Edit Distance to Shifted Gap Edit Distance

Input: An instance (X, Y) of (β, α)-Gap Edit Distance, and a parameter $\phi \in \mathbb{Z}_+$ satisfying

$$\phi \geq \beta \geq \psi := \lfloor \frac{112\beta\phi\lceil \log n \rceil}{\alpha} \rfloor.$$

1 Set $\rho := \frac{84\phi}{\alpha}$;
2 **for** $p \in [\,\lceil \log(3\phi) \rceil \,..\, \lfloor \log(\rho n) \rfloor\,]$ **do**
3   Set $m_p := \lceil \frac{n}{2^p} \rceil$ to be number of blocks of length 2^p we partition X, Y into (last block might be shorter);
4   For $i \in [\,0 \,..\, m_p\,)$ set $X_{p,i} = X[\,i \cdot 2^p \,..\, \min(n, (i+1)2^p)\,)$ and $Y_{p,i} = Y[\,i \cdot 2^p \,..\, \min(n, (i+1)2^p)\,)$;
5   **for** $t \in [\,0 \,..\, \lceil \rho m_p \rceil\,)$ **do**
6   Select $i \in [\,0 \,..\, m_p\,)$ uniformy at random;
7   Solve the instance $(X_{p,i}, Y_{p,i})$ of the β-Shifted $(\psi, 3\alpha)$-Gap Edit Distance problem;
8 **if** *we obtained at most 5 times* NO *at Line 7* **then**
9   **return** YES
10 **else**
11   **return** NO

β-Shifted $(\gamma, 3\alpha)$-Gap Edit Distance
Input: strings X, Y of length $|X| = |Y| = n$, and integer thresholds $\alpha \geq \beta \geq \gamma \geq 0$.
Output: YES if $\delta_E^\beta(X, Y) \leq \gamma$, NO if $\delta_E(X, Y) > 3\alpha$, and an arbitrary answer otherwise.

The insight of [GKKS22] is that an instance of Shifted Gap Edit Distance can be reduced back to smaller instances of Gap Edit Distance. In this way, [GKKS22] obtains a recursive algorithm switching between instances of Shifted Gap Edit Distance and of Gap Edit Distance. The size of the instances shrinks throughout different calls until the algorithm is left with instances having trivial solutions.

From Gap Edit Distance to Shifted Gap Edit Distance The algorithm that reduces Gap Edit Distance instances to Shifted Gap Edit Distance instances partitions the strings X and Y into blocks of different lengths. It samples these blocks and utilizes them as input for instances of the Shifted Gap Edit Distance problem. If a limited number of instances return NO, the routine outputs YES. Otherwise, it returns NO. A description is provided in Algorithm 1.

Goldenberg, Kociumaka, Krauthgamer, and Saha proved in [GKKS22] the following.

Lemma 5.14 (Lemma 4.3 of [GKKS22]). *Suppose all oracle calls at Line 7 of Algorithm 1, return the correct answer. Then, Algorithm 1 solves an instance (X, Y) of (β, α)-Gap Edit Distance with error probability at most e^{-1}.*

From Shifted Gap Edit Distance to Gap Edit Distance In contrast to the previous algorithm, the algorithm reducing Shifted Gap Edit Distance instances to Gap Edit Distance instances is deterministic. For an instance (X, Y) of β-Shifted $(\gamma, 3\alpha)$-Gap Edit Distance satisfying $\alpha \geq 3\gamma$, the algorithm makes several calls to an oracle that solves $(\gamma, 3\alpha)$-Gap Edit Distance instances. The routine returns YES if and only if at least one such oracle call returned YES. A description is provided in Algorithm 2.

Lemma 5.15 (Lemma 4.3 of [GKKS22]). *Suppose all oracle calls at Line 2 of Algorithm 2, return correct answer. Then, Algorithm 2 solves an instance (X, Y) of β-Shifted $(3\gamma, \alpha)$-Gap Edit Distance.*

Algorithm 2 Deterministic Algorithm reducing Shifted Gap Edit Distance to Gap Edit Distance.

Input: An instance (X, Y) of β-Shifted $(\gamma, 3\alpha)$-Gap Edit Distance satisfying $\alpha \geq 3\gamma$.

1 Set $\xi := \lfloor\sqrt{(1+\beta)(1+\gamma)}\rfloor - 1$ and $n' = n - \beta$;

2 For all $x \in [\,0\,..\,\beta\,]$ such that $x \equiv_{1+\xi} \beta$ or $x \equiv_{1+\xi} 0$, all $y \in [\,0\,..\,\xi\,]$ such that $y \equiv_{1+\gamma} 0$, all $y \in [\,\beta-\xi\,..\,\beta\,]$ such that $y \equiv_{1+\gamma} \beta$, solve the instance $(X[\,x\,..\,x+n'\,), Y[\,y\,..\,y+n'\,))$ of the $(3\gamma, \alpha)$-Gap Edit Distance problem;

3 **if** *we obtained at least one time* yes *at Line 2* **then**

4   **return** yes

5 **else**

6   **return** no

The Base Case The algorithm is structured to switch between instances of Algorithm 1 and Algorithm 2 until it reaches an instance of the (β, α)-Gap Edit Distance problem with $\beta = 0$. This instance represents the base case of the algorithm. To handle it, the algorithm simply checks whether there is at least one position $i \in [\,0\,..\,n\,)$ such that $X[i] \neq Y[i]$, i.e., it checks whether the Hamming distance $\delta_H(X, Y)$ is non-zero. If such a position is found, the algorithm returns no; otherwise, it returns yes. This check is sufficient because if $\delta_E(X, Y) = 0$, then no such position exists and $\delta_H(X, Y) = 0$. Conversely, if $\delta_E(X, Y) \geq \alpha$, there must be at least α positions $i \in [\,0\,..\,n\,)$ such that $X[i] \neq Y[i]$, i.e., $\delta_H(X, Y) \geq \delta_E(X, Y) \geq \alpha$. If we sample $\frac{cn}{1+\alpha}$ positions, the probability of sampling at least one of those α positions is at most

$$\left(1 - \frac{\alpha}{n}\right)^{\frac{cn}{1+\alpha}} \leq e^{-\frac{c\alpha}{1+\alpha}}.$$

By choosing $c = O(1)$ appropriately, we may assume any constant lower bound for the success probability.

5.3.2 Adapting the Algorithm to the Quantum Setting

The algorithm is formally constructed through Proposition 4.6 and Proposition 4.7 in [GKKS22]. These propositions prove, respectively, the existence of an algorithm for the Gap Edit Distance problem and for the Shifted Gap Edit Distance problem which, together with the information for the respective instance, take in input a parameter h, describing the depth of the recursive calls we make until we are at the base described in the previous (sub)section.

In this (sub)section we want to argue that we can take Proposition 4.6 and Proposition 4.7 in [GKKS22] and adapt them to the quantum setting, by creating subroutines that replicate their behavior when provided in input the same sequence of bits. The adaptation to the quantum case will be rather straightforward, since the outcome of both Algorithm 1 and Algorithm 2 relies on independent calls to other oracles that simply return yes or no, and such structure can be exploited by Grover's search.

Proposition 5.16. *(Compare with Proposition 4.6 of [GKKS22]) There exists a quantum algorithm with the following properties:*

- *it takes as input $h \in \mathbb{Z}_{\geq 0}$, $\varepsilon \in \mathbb{R}_+$, and an instance of the (β, α)-*Gap Edit Distance *problem, satisfying $\beta < (336\lceil\log n\rceil)^{\frac{-h}{2}} \alpha^{\frac{h}{h+1}}$, and a sequence of bits* seed $\in \{0,1\}^\ell$ *of length*
$$\ell = O\left(\frac{1+\beta}{1+\alpha} \cdot n \cdot \log^{O(h)} n \cdot \log\frac{1}{\varepsilon} \cdot 2^{O(h)}\right);$$
- *it outputs with high probability the same output of the non-adaptive algorithm from Proposition 4.6 of [GKKS22] with input h, ε, the instance of the (β, α)-*Gap Edit Distance *problem, and* seed*; and*

- *it uses*
$$O\left(\frac{1+\beta}{1+\alpha}\cdot n\cdot \log^{O(h)} n\cdot \log\frac{1}{\varepsilon}\cdot 2^{O(h)}\right) \text{ time and } O\left(\left(\frac{1+\beta}{1+\alpha}\right)^{\frac{1}{2}}\cdot n^{\frac{1}{2}}\cdot \log^{O(h)} n\cdot \log\frac{1}{\varepsilon}\cdot 2^{O(h)}\right) \text{ queries.}$$

Proposition 5.17. *(Compare with Proposition 4.7 of [GKKS22]) There exists a quantum algorithm with the following properties:*

- *it takes as input $h \in \mathbb{Z}_{\geq 0}$, $\varepsilon \in \mathbb{R}_+$, and an instance of the β-*Shifted *$(\gamma, 3\alpha)$-*Gap Edit Distance *problem, satisfying $\gamma < \frac{1}{3}(336\lceil \log n\rceil)^{\frac{-h}{2}}\alpha^{\frac{h}{h+1}}$, and a sequence of bits* seed $\in \{0,1\}^{\ell}$ *of length*
$$\ell = O\left(\frac{1+\beta}{1+\alpha}\cdot n\cdot \log^{O(h)} n\cdot \log\frac{n}{\varepsilon}\cdot 2^{O(h)}\right);$$
- *it outputs with high probability the same output of the non-adaptive algorithm from Proposition 4.7 of [GKKS22] with input h, ε, the instance of the β-*Shifted *$(\gamma, 3\alpha)$-*Gap Edit Distance *problem, and* seed*; and*
- *it uses*
$$O\left(\frac{1+\beta}{1+\alpha}\cdot n\cdot \log^{O(h)} n\cdot \log\frac{n}{\varepsilon}\cdot 2^{O(h)}\right) \text{ time, and } O\left(\left(\frac{1+\beta}{1+\alpha}\right)^{\frac{1}{2}}\cdot n^{\frac{1}{2}}\cdot \log^{O(h)} n\cdot \log\frac{n}{\varepsilon}\cdot 2^{O(h)}\right) \text{ queries.}$$

Proposition 4.6 and Proposition 4.7 of [GKKS22] are proved by induction over h, alternating between Proposition 4.6 and Proposition 4.7 of [GKKS22], mirroring the structure of the recursive calls of the algorithm. In order to ensure correctness with probability at least ε, the propositions use the fact that we can boost an algorithm's success probability from any constant to $1-\varepsilon$ by repeating the algorithm $O(\log\frac{1}{\varepsilon})$ times. We prove Proposition 5.16 and Proposition 5.17 using the same structure of induction.

Proof of Proposition 5.16 and Proposition 5.17. As mentioned in the previous (sub)section, the base case is represented by Proposition 5.16 when $\beta = 0$ or $h = 0$. There, it suffices to interpret the seed as sampled positions from $[\,0\,..\,n\,)$, and to execute Grover's search over all sampled positions checking whether there exists a sampled position $i \in [\,0\,..\,n\,)$ such that $X[i] \neq Y[i]$, thereby obtaining the claimed query time and quantum time.

Induction step for Proposition 5.16 If $\beta \neq 0$ and $h \neq 0$, the algorithm from Proposition 4.6 of [GKKS22] uses Algorithm 1 with $\phi = \beta$, employing as an oracle Proposition 4.7 of [GKKS22] with parameters $h-1$ and $\Theta(\frac{1}{n})$. We adapt Algorithm 1 to the quantum setting as follows. First, we extract the randomness required by the routine, i.e., for each level $p \in [\,\lceil\log(3\phi)\rceil\,..\,\lfloor\log(\rho n)\rfloor\,]$ we have to read $\lceil \rho m_p\rceil = \lceil\frac{84\phi}{\alpha}\frac{n}{2^p}\rceil$ new positions from the sequence bits from the input. This allows us to execute Grover's search over all selected instances separately for each of the levels. Our goal is to detect at least five oracle calls returning no. Since we are only looking for constantly many of those, it suffices to pay an additional $\tilde{O}(1)$ factor in Grover's search. The quantum time becomes:
$$\begin{aligned}
&\sum_{p=\lceil\log(3\phi)\rceil}^{\lfloor\log(\rho n)\rfloor} O\left(\lceil\rho m_p\rceil + \lceil\rho m_p\rceil^{\frac{1}{2}}\cdot\frac{\beta}{\phi}\cdot 2^p\cdot\log^{O(h-1)} n\cdot\log n\cdot 2^{O(h-1)}\right)\\
&\leq \sum_{p=\lceil\log(3\phi)\rceil}^{\lfloor\log(\rho n)\rfloor} O\left(\lceil\rho m_p\rceil\cdot\frac{\beta}{\phi}\cdot 2^p\cdot\log^{O(h-1)} n\cdot\log n\cdot 2^{O(h-1)}\right)\\
&\leq O\left(\frac{\beta}{\alpha}\cdot n\cdot\log^{O(h)} n\cdot 2^{O(h)}\right)
\end{aligned}$$

Similarly, the query complexity becomes:

$$\sum_{p=\lceil \log(3\phi)\rceil}^{\lfloor \log(\rho n)\rfloor} O\left(\lceil \rho m_p \rceil^{\frac{1}{2}} \cdot \left(\frac{\beta}{\phi}\right)^{\frac{1}{2}} \cdot 2^{\frac{p}{2}} \cdot \log^{O(h-1)} n \cdot \log n \cdot 2^{O(h-1)}\right) = O\left(\left(\frac{\beta}{\alpha}\right)^{\frac{1}{2}} \cdot n^{\frac{1}{2}} \cdot \log^{O(h)} n \cdot 2^{O(h)}\right).$$

Induction step for Proposition 5.17 The algorithm from Proposition 4.7 of [GKKS22] uses Algorithm 2 employing as an oracle Proposition 4.6 of [GKKS22] with parameters h and $\Theta(\frac{\varepsilon}{n})$. Adapting Algorithm 2 is even more straightforward, since it suffices to use Grover's search over all $4\lceil\frac{1+\beta}{1+\xi}\rceil\lceil\frac{1+\xi}{1+\gamma}\rceil \le 16 \cdot \frac{1+\beta}{1+\gamma}$ independent oracle calls, where $1+\xi = \lfloor\sqrt{(1+\beta)(1+\gamma)}\rfloor$. Note that each of the oracle calls involves a string of length at most n. Therefore, the quantum time becomes:

$$O\left(\left(\frac{1+\beta}{1+\gamma}\right)^{\frac{1}{2}} \cdot \frac{1+3\gamma}{1+\alpha} \cdot n \cdot \log^{O(h)} n \cdot \log n \cdot 2^{O(h)}\right) \le O\left(\frac{1+\beta}{1+\alpha} \cdot n \cdot \log^{O(h)} n \cdot \log n \cdot 2^{O(h)}\right).$$

Similarly, the query complexity becomes:

$$O\left(\left(\frac{1+\beta}{1+\gamma}\right)^{\frac{1}{2}} \cdot \left(\frac{1+3\gamma}{1+\alpha}\right)^{\frac{1}{2}} \cdot n^{\frac{1}{2}} \cdot \log^{O(h)} n \cdot \log n \cdot 2^{O(h)}\right) = O\left(\left(\frac{1+\beta}{1+\alpha}\right)^{\frac{1}{2}} \cdot n^{\frac{1}{2}} \cdot \log^{O(h)} n \cdot \log n \cdot 2^{O(h)}\right).$$

◀

Proof of Lemma 5.12. We apply, Proposition 5.16, setting $\beta = k$, $\alpha = kn^{o(1)}$ and ε polynomially small. Further, we set h to be the smallest integer such that $\beta < (336\lceil\log n\rceil)^{\frac{-h}{2}}\alpha^{\frac{h}{h+1}}$. By doing so, we obtain the desired quantum time and query time. ◀

6 Near-Optimal Quantum Algorithm for Pattern Matching with Edits

In the following section, we will demonstrate how the combinatorial results from Section 4 pave the way for a quantum algorithm with nearly optimal query time up to a sub-polynomial factor. As already mentioned earlier in Section 2, the development of such an algorithm also relies on other components such as the combinatorial results of [CKW20] and the quantum subroutine for the Gap Edit Distance problem from Section 5.3.

This section is organized as follows. In Section 6.1, we establish the existence of an efficient quantum algorithm that can compute a structural decomposition, as described in Lemma 2.1. In Section 6.2, we illustrate how to construct the combinatorial structures used throughout Section 4. This allows to construct in Section 6.3 a framework that enables the verification of candidate positions, and to examine in Section 6.4 two specific scenarios that are addressed using the framework: one scenario involves limited candidate positions for k-error occurrences, while the other scenario deals with the pattern having very few edits to a period extension of a primitive string. Finally, in Section 6.5, we explain how to assemble all the components and provide a quantum algorithm for the Pattern Matching with Edits problem with near-optimal query time.

6.1 Structural Decomposition of the Pattern

This (sub)section aims to prove the following lemma.

▶ **Lemma 6.5.** *Given a pattern P of length m, we can find a structural decomposition as described in Lemma 2.1 in $\tilde{O}(\sqrt{km})$ query time and $\tilde{O}(\sqrt{km}+k^2)$ quantum time.* ◀

Algorithm 3 A constructive proof of Lemma 2.1 [CKW20, Algorithm 9].

$\mathcal{B} \leftarrow \{\}; \mathcal{R} \leftarrow \{\}$;
while true do
  Consider fragment $P' = P[\, j \mathinner{.\,.} j + \lfloor m/8k \rfloor \,)$ of the next $\lfloor m/8k \rfloor$ unprocessed characters of P;
  if $\mathrm{per}(P') > m/128k$ **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{P'\}$;
    if $|\mathcal{B}| = 2k$ **then return** *breaks* $\mathcal{B}$;
  else
    $Q \leftarrow P[\, j \mathinner{.\,.} j + \mathrm{per}(P') \,)$;
    Search for prefix R of $P[\, j \mathinner{.\,.} m \,)$ with $\delta_E(R, {}^*Q^*) = \lceil 8k/m \cdot |R| \rceil$ and $|R| > |P'|$;
    if *such R exists* **then**
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{(R, Q)\}$;
      if $\sum_{(R,Q)\in\mathcal{R}} |R| \geq 3/8 \cdot m$ **then**
        return *repetitive regions (and their corresponding periods)* $\mathcal{R}$;
    else
      Search for suffix R' of P with $\delta_E(R', {}^*Q^*) = \lceil 8k/m \cdot |R'| \rceil$ and $|R'| \geq m - j$;
      if *such R' exists* **then return** *repetitive region* (R', Q);
      else return *approximate period* Q;

Charalampopoulos, Kociumaka, and Wellnitz [CKW20] introduced a high-level algorithm (see Algorithm 3) that computes such a structural decomposition and can be readily adapted to various computational settings. This algorithm is also easily adaptable to the quantum case. It is worth noting that Algorithm 3 provides a constructive proof of Lemma 2.1 on its own. For a more detailed proof, we recommend referring directly to [CKW20].

Algorithm 3 maintains an index j indicating the position of the string that has been processed and returns as soon as one of the structural properties described in Lemma 2.1 is found. At each step, the algorithm considers the fragment $P' = P[\, j \mathinner{.\,.} + \lfloor m/8k \rfloor \,)$. If $\mathrm{per}(P') \geq m/128k$, P' is added to $\mathcal{B}$. If not, the algorithm attempts to extend P' to a repetitive region. Let Q be the string period of P'. The extension involves searching for a prefix R of $P[\, j \mathinner{.\,.} m \,)$ such that $\delta_E(R, {}^*Q^*) = \lceil 8k/m \cdot |R| \rceil$. If such a prefix is found, R is added to $\mathcal{R}$. If not, it indicates that there were not enough mismatches between Q and $P[\, j \mathinner{.\,.} m \,)$. At this point, the algorithm attempts to extend $P[\, j \mathinner{.\,.} m \,)$ backward to a repetitive region. If successful, a sufficiently long repetitive region has been found to return. Otherwise, it suggests that $\delta_E(P, {}^*Q^*)$ is small, implying that P has approximate period Q.

In adapting Algorithm 3 to the quantum setting, the primary focus lies on the procedure for identifying a suitable prefix R of $P[\, j \mathinner{.\,.} m \,)$ such that $\delta_E(R, {}^*Q^*) = \lceil 8k/m \cdot |R| \rceil$. The first step towards accomplishing this consists into, given a prefix R of $P[\, j \mathinner{.\,.} m \,)$, to determine whether $\delta_E(R, {}^*Q^*)$ is less than, equal to, or greater than $\lceil 8k/m \cdot |R| \rceil$. Note that we already know that for any prefix R of $P[\, j \mathinner{.\,.} m \,)$ such that $|R| \leq \lfloor m/8k \rfloor$, we have $\delta_E(R, {}^*Q^*) = 0$. Therefore, we only consider the case where $|R| > \lfloor m/8k \rfloor$. This allows us to later combine an exponential search/binary search to verify the existence of a prefix with the desired property.

For the sake of conciseness, we introduce the following definition.

Definition 6.1. *Let P denote a pattern of length m, and let Q denote a primitive string of length $|Q| \leq m/128k$ for some positive threshold k. For $j' \in (\, j + \lfloor m/8k \rfloor \mathinner{.\,.} m \,]$, define $\Delta(j') = \delta_E(P[\, j \mathinner{.\,.} j' \,), {}^*Q^*) - \lceil 8k/m \cdot |j' - j'| \rceil$.*

Consequently, the first steps consists into devising a procedure that calculates the sign of $\Delta(j')$ for $j' \in (\, j + \lfloor m/8k \rfloor \mathinner{..} m \,]$.

Lemma 6.2. *Let P denote a pattern of length m, and let Q denote a primitive string of length $|Q| \le m/128k$ for some positive threshold k. Moreover, let $j' \in (\, j + \lfloor m/8k \rfloor \mathinner{..} m \,]$.*

Then, we can verify whether $\Delta(j') < 0$, $\Delta(j') = 0$ or $\Delta(j') > 0$ in $\tilde{O}(|j' - j| \cdot \sqrt{k/m})$ query time and $\tilde{O}(|j' - j| \cdot \sqrt{k/m} + |j' - j|^2 \cdot k^2/m^2)$ quantum time.

Proof. Let $R = [\, j \mathinner{..} j' \,)$. Consider the following procedure that either fails or returns an integer:

(i) divide R into segments of length $2|Q|$, obtaining $|R|/2|Q| \ge 64k/m \cdot |R|$ full segments;

(ii) sample uniformly at random one of the segments, and try to find an exact occurrence of Q in the selected segment using Theorem 5.3, if there is none, fail; lastly,

(iii) if the search for successful, let $i \in [\, 0 \mathinner{..} |R| \,)$ be the starting position where we have found an exact occurrence of Q. Using Lemma 5.10, check whether $\delta_E(R[\, 0 \mathinner{..} i \,), {}^*Q) \le \lceil 8k/m \cdot |R| \rceil$ and $\delta_E(R[\, i + |Q| \mathinner{..} |R| \,), Q^*) \le \lceil 8k/m \cdot |R| \rceil$ If either $\delta_E(R[\, 0 \mathinner{..} i \,), {}^*Q) > \lceil 8k/m \cdot |R| \rceil$, $\delta_E(R[\, i + |Q| \mathinner{..} |R| \,), Q^*) > \lceil 8k/m \cdot |R| \rceil$, or $\delta_E(R[\, 0 \mathinner{..} i \,), {}^*Q) + \delta_E(R[\, i + |Q| \mathinner{..} |R| \,), Q^*) > \lceil 8k/m \cdot |R| \rceil$ then fail. Otherwise, return $\delta_E(R[\, 0 \mathinner{..} i \,), {}^*Q) + \delta_E(R[\, i + |Q| \mathinner{..} |R| \,), Q^*)$.

Claim 6.3. *If $\delta_E(R, {}^*Q^*) \le \lceil 8k/m \cdot |R| \rceil$, i.e. $\Delta(j') \le 0$, then with constant probability the procedure does not fail, and returns $\delta_E(R, {}^*Q^*)$. Moreover, if $\delta_E(R, {}^*Q^*) > \lceil 8k/m \cdot |R| \rceil$, i.e., $\Delta(j') > 0$, the algorithm fails.*

Proof. Suppose $\delta_E(R, {}^*Q^*) \le \lceil 8k/m \cdot |R| \rceil$, and consider an edit alignment of R with the optimal periodic extension of Q on both sides. Note, at least $64k/m \cdot |R| - \lceil 8k/m \cdot |R| \rceil \ge 55k/m \cdot |R|$ of the segments we divide R into do not contain any edit in the edit alignment. This means that these segments match exactly with a fragment of length $2|Q|$ of the optimal periodic extension of Q on both sides.

In any of those fragments of length $2|Q|$ a copy of Q is always fully contained. As a consequence, by sampling uniformly at random one of the segments of R, with probability at least $(55k/m \cdot |R|)/(64k/m \cdot |R|) \ge \Omega(1)$ we find a segment where a copy of Q appears without any edits in the optimal alignment.

Moreover, we have

$$\delta_E(R, {}^*Q^*) = \delta_E(R[\, 0 \mathinner{..} i \,), {}^*Q) + \delta_E(R[\, i + |Q| \mathinner{..} |R| \,), Q^*).$$

Since $\delta_E(R, {}^*Q^*) \le \lceil 8k/m \cdot |R| \rceil$, we have

$$\delta_E(R[\, 0 \mathinner{..} i \,), {}^*Q) \le \lceil 8k/m \cdot |R| \rceil \text{ and } \delta_E(R[\, i + |Q| \mathinner{..} |R| \,), Q^*) \le \lceil 8k/m \cdot |R| \rceil.$$

Next, suppose $\delta_E(R, {}^*Q^*) > \lceil 8k/m \cdot |R| \rceil$. If the procedure fails in (ii), then there is nothing to prove. Therefore, assume that the procedure proceeds to (iii). The second part of the claim follows directly from the fact that if at least one of $\delta_E(R[\, 0 \mathinner{..} i \,), {}^*Q) > \lceil 8k/m \cdot |R| \rceil$, $\delta_E(R[\, i + |Q| \mathinner{..} |R| \,), Q^*) > \lceil 8k/m \cdot |R| \rceil$, and $\delta_E(R[\, 0 \mathinner{..} i \,), {}^*Q) + \delta_E(R[\, i + |Q| \mathinner{..} |R| \,), Q^*) > \lceil 8k/m \cdot |R| \rceil$ holds, then we fail.

Hence, by executing the procedure from above $\tilde{O}(1)$ times, then with high probability we succeed in either calculating $\Delta(j')$ exactly if $\Delta(j') \le 0$, or in verifying $\Delta(j') > 0$. By doing so, the quantum and query time are dominated by Lemma 5.10. Overall, we need $\tilde{O}(|R| \cdot \sqrt{k/m} + |R| \cdot k/m) = \tilde{O}(|R| \cdot \sqrt{k/m})$ query time and $\tilde{O}(|R| \cdot \sqrt{k/m} + |R|^2 \cdot k^2/m^2)$ quantum time. This completes the proof.

Next, we show how to find a prefix R of $P[\, j \mathinner{..} m \,)$ such that $\delta_E(R, {}^*Q^*) = \lceil 8k/m \cdot |R| \rceil$, i.e., how to find $j' \in (\, j \mathinner{..} m \,]$ such that $\Delta(j') = 0$.

Lemma 6.4. *Let P denote a pattern of length m, and let Q denote a primitive string of length $|Q| \le m/128k$ for some threshold k.*

Then, there is a quantum procedure that either

(1) *finds a prefix R of $P[\,j\,..\,m\,)$ such that $\delta_E(R, {}^*Q^*) = \lceil 8k/m \cdot |R| \rceil$ in $\tilde{O}(|R| \cdot \sqrt{k/m})$ query time and $\tilde{O}(|R| \cdot \sqrt{k/m} + |R|^2 \cdot k^2/m^2)$ quantum time; or*

(2) *verifies that $\delta_E(P[\,j\,..\,m\,), {}^*Q^*) \le \lceil 8k/m \cdot |R| \rceil$ in $\tilde{O}(\sqrt{km})$ query time and $\tilde{O}(\sqrt{km} + k^2)$ quantum time. Moreover, it marks the search for R as unsuccessful.*

Proof. The algorithm proceeds as follows:

(i) using Lemma 6.2, it finds the smallest position $i \in [\,0\,..\,\lceil \log(m-j) \rceil\,)$ such that $\Delta(\min(2^i, m-j)) \ge 0$;

(ii) if such i does not exist then it marks the search of R as not successful;

(iii) else if $\Delta(\min(2^i, m-j)) = 0$, then it returns the prefix $R = P[\,j\,..\,\min(2^i, m-j)\,)$; lastly

(iv) if neither of the two previous two cases holds, it executes a binary search. The two indices ℓ, h are kept as a lower bound and an upper bound, and they are initialized to $\ell := j + |Q|$ and $h := \min(j + 2^i, m)$. In every iteration, the algorithm sets mid to be the middle between position ℓ and h, and it uses Lemma 6.2 to retrieve the sign of $\Delta(\mathsf{mid})$. If $\Delta(\mathsf{mid}) = 0$, then it returns the prefix $R = P[\,j\,..\,\mathsf{mid}\,)$. If $\Delta(\mathsf{mid}) > 0$ it recurses on the right side. Lastly, if $\Delta(\mathsf{mid}) < 0$, it recurses on the left side.

Clearly, the algorithm is correct when it returns in (ii) or in (iii). It remains to show that if the algorithm proceeds to (iv), then it finds $j' \in (\,j\,..\,m\,]$ such that $\Delta(j') = 0$. It is easy to verify that the algorithm maintains the invariant $\Delta(\ell) < 0$ and $\Delta(h) > 0$. Indeed, from $Q = P[\,j\,..\,j+|Q|\,)$ and from how we perform the exponential search follows that the invariant holds in the beginning. Moreover, from (iv) follows that we preserve the invariant at every step of the algorithm. For the sake of contradiction, assume the binary search has ended without finding such a prefix R, i.e., $h = \ell + 1$ and the invariant still holds. Let $R_\ell = P[\,j\,..\,\ell\,)$ and $R_h = P[\,j\,..\,h\,)$ Observe that for $h = l + 1$, we have

$$\delta_E(R_l, {}^*Q^*) = \delta_E(R_h, {}^*Q^*) + c \text{ and } \lceil 8k/m \cdot |R_l| \rceil = \lceil 8k/m \cdot |R_h| \rceil + c' \text{ for } c, c' \in \{0, 1\}.$$

However, there are no values of c, c' such that

$$\delta_E(R_l, {}^*Q^*) < \lceil 8k/m \cdot |R_l| \rceil \text{ and } \delta_E(R_h, {}^*Q^*) > \lceil 8k/m \cdot |R_h| \rceil.$$

Hence, the binary search terminates and finds a prefix R of $P[\,j\,..\,m\,)$ such that $\delta_E(R, {}^*Q^*) = \lceil 8k/m \cdot |R| \rceil$.

Regarding the complexity analysis, notice that if the algorithm returns in (ii) then the quantum and query time is dominated by Lemma 6.2 with $|j' - j| = O(m)$; otherwise, it is dominated by Lemma 6.2 with $|j' - j| = O(|R|)$. ◼

Finally, we show how to adapt Algorithm 3 to the quantum setting.

Lemma 6.5. *Given a pattern P of length m, we can find a structural decomposition as described in Lemma 2.1 in $\tilde{O}(\sqrt{km})$ query time and $\tilde{O}(\sqrt{km} + k^2)$ quantum time.*

Proof. We show how to implement Algorithm 3 in the claimed query and quantum time. There is no need to modify its pseudocode. We go through line by line and show how we can turn it into a quantum algorithm by using suitable quantum subroutines.

First, observe that the while loop at line 2 requires at most $O(k)$ iterations, as we process at least $\lfloor m/8k \rfloor$ characters in each iteration.

Since we can find the period of P' using Corollary 5.4 in $\tilde{O}(\sqrt{m/k})$ quantum time and query time, we can conclude that line 4 summed over all iterations runs in $\tilde{O}(k \cdot \sqrt{m/k}) = \tilde{O}(\sqrt{km})$ time. To check for the existence of the prefix R at line 9 we use Lemma 6.4. Using $\sum_{(R,Q)} |R| \le m$, we conclude that summed over all $R \in \mathcal{R}$ this results in $\tilde{O}(\sqrt{km})$ query time and

$$\sum_{(R,Q)} \tilde{O}(|R| \cdot \sqrt{k/m} + |R|^2 \cdot k/m) = \tilde{O}(\sqrt{k^2/m^2}) \sum_{(R,Q)} |R| + \tilde{O}(k^2/m^2) \sum_{(R,Q)} |R|^2 \le \tilde{O}(\sqrt{km} + k^2) \text{ quantum time.}$$

If we did not manage to find such a prefix R, we incur the same quantum and query complexity, and we return either at line 15 or at line 17. Before returning, we still have to determine if R' exists. Similarly, by using again Lemma 6.4, we can argue that the search for R' requires no more than the claimed query and quantum time. ◾

6.2 Construction of the Combinatorial Structures of Section 4

In this section, we show how to construct the combinatorial structures used in Section 4. We will continue to use the same notation introduced in Section 4 relative to strings P, T, and a set S of alignments that align P onto substrings of T with a cost at most k, where k represents a positive threshold.

Lemma 6.6 (See Definition 4.1, Lemma 4.4, Definition 4.7, Definition 4.9, and Lemma 4.10)**.** *Let P be a string of length m, let T be a string, and let k be a positive threshold such that $k \le m$. Further, let S be a set of alignments of P onto substrings of T of cost at most k such that S encloses T. Then, there exists a classical algorithm with the following properties.*

- *The algorithm takes as input the edit information $\mathsf{E}_{P,T}(\mathcal{X})$ for all $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,)$ such that $\mathcal{X} \in S$.*
- *The algorithm constructs $\mathbf{G}_S$ and computes $\mathrm{bc}(\mathbf{G}_S)$. If $\mathrm{bc}(\mathbf{G}_S) > 0$, the algorithm computes τ_i^c and π_j^c for all $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$, $i \in [\,0\,..\,n_c\,)$ and $j \in [\,0\,..\,m_c\,)$. Moreover, it computes $\mathrm{w}_S(c)$ for all $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$, where $\mathrm{w}_S(c)$ is the weight function from Lemma 4.10.*
- *The algorithm takes $\tilde{O}(m \cdot |S|)$ time.*

Proof. The algorithm proceeds as follows.

(i) The algorithm reconstructs all $\mathcal{X} : P \rightsquigarrow T[\,t\,..\,t'\,) \in S$ using the edit information $\mathsf{E}_{P,T}(\mathcal{X})$. Using the same edit information it identifies the edits contained in all $\mathcal{X} \in S$.

(ii) The algorithm constructs $\mathbf{G}_S$ as described in Definition 4.1, and marks an edge red if it corresponds to at least one edit in any alignment in S. Otherwise, the edge is marked black and the algorithm stores a reference to all alignments that contain it. Next, it counts connected components containing only black edges, i.e., it computes $\mathrm{bc}(\mathbf{G}_S)$. If $\mathrm{bc}(\mathbf{G}_S) = 0$, the algorithm reports that $\mathrm{bc}(\mathbf{G}_S) = 0$ and returns.

(iii) The algorithm selects for every black connected component the smallest position of a character of P contained in it. Then, it sorts in non-decreasing order these $\mathrm{bc}(\mathbf{G}_S)$ positions and marks the black connected components containing the c-th character (0-indexed) in this sorted sequence as the c-th black connected component.

(iv) For every $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$, the algorithm sorts in non-decreasing order the characters of P and T contained in the c-th black component depending on the position at which they appear in P and T. The i-th character in the first sorted sequence is marked as τ_i^c. Conversely, the j-th character in the second sorted sequence for is marked as π_j^c.

(v) The algorithm computes $\mathrm{w}_S(c)$ for all $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$ in the way described in Lemma 4.10. To this end, it first constructs $\bar{\mathbf{G}}_S$, i.e., the subgraph of $\mathbf{G}_S$ induced by all vertices $P[\pi_j^c]$ and $T[\tau_i^c]$ for $c \in [\,0\,..\,\mathrm{bc}(\mathbf{G}_S)\,)$, $j \in [\,0\,..\,m_{c+1}\,)$, and $i \in [\,0\,..\,n_{c+1}\,)$. Next, the algorithm assigns weights as described in Lemma 4.10: for every edge $\{P[\pi_j^c], T[\tau_i^c]\}$ contained in $\bar{\mathbf{G}}_S$, the algorithm associates to it the smallest value $\delta_E^{\mathcal{X}}(P[\pi_j^c \mathinner{.\,.} \pi_j^{c+1}), T[\tau_i^c \mathinner{.\,.} \tau_i^{c+1}))$ where $\mathcal{X}$ ranges over all alignments referenced by (π_j^c, τ_i^c). Now, for $c \in [0 \mathinner{.\,.} \mathrm{bc}(\mathbf{G}_S)) \setminus \{c_{\mathrm{last}}, \mathrm{bc}(\mathbf{G}_S) - 1\}$, it suffices to set $\mathrm{w}_S(c)$ to be the total weight of edges in $\bar{\mathbf{G}}_S^c$, where $\bar{\mathbf{G}}_S^c$ is defined as in Lemma 4.10. For $c \in \{c_{\mathrm{last}}, \mathrm{bc}(\mathbf{G}_S) - 1\}$, we need to additional calculate α and α', which can be retrieved directly from the edit information of $\mathcal{X}_{\mathrm{pref}}$ and $\mathcal{X}_{\mathrm{suf}}$, where $\mathcal{X}_{\mathrm{pref}}, \mathcal{X}_{\mathrm{suf}} \in S$ are such that $(0,0) \in \mathcal{X}_{\mathrm{pref}}$ and $(|P|, |T|) \in \mathcal{X}_{\mathrm{suf}}$.

The correctness of the algorithm follows from Definition 4.1, Lemma 4.4, Definition 4.7, Definition 4.9, and Lemma 4.10.

Regarding the time complexity, it is worth noting that reconstructing each of the alignments in S results in $|S|$ alignments, each with a size of $O(m+k)$. Consequently, the graph $\mathbf{G}_S$ has at most $O(m)$ vertices and $O(|S| \cdot m)$ edges. With this in mind, steps (i), (ii), (iii), and (iv) can all be implemented in $\tilde{O}(|S| \cdot m)$ time. Additionally, utilizing the same time complexity, one can precompute the values of $\delta_E^{\mathcal{X}}(P[\pi_j^c \mathinner{.\,.} \pi_j^{c+1}), T[\tau_i^c \mathinner{.\,.} \tau_i^{c+1}))$ for all $(\pi_j^c, \tau_i^c) \in \mathcal{X}$ such that $\{P[\pi_j^c], T[\tau_i^c]\}$ forms an edge in $\bar{\mathbf{G}}_S$, allowing the implementation of (v) within the same time constraints. ◾

Lemma 6.7 (See Lemma 4.16)**.** *The construction of a black cover described in Lemma 4.16 can be adapted to the quantum setting. That is, there exists a quantum algorithm that takes in input the output from Lemma 6.6 for a set S of size $|S| = O(\log n)$, computes a black cover C_S, and outputs the information $\{(c, T[\tau_0^c]) : c \in C_S\}$ using $\tilde{O}(\sqrt{km})$ query time and $\tilde{O}(\sqrt{km} + k^2 + m)$ quantum time.*

Proof. The algorithm computes c_{pref}, c_{suff}, c_{lsuff}, and c_{lpref} and retrieves the LZ compression of the relative strings by combining Theorem 5.8 with an exponential/binary search. Furthermore, the algorithm computes $C_S^{0,\mathrm{bc}(\mathbf{G}_S)-1}$ using the recursive construction described in Lemma 4.16. In the recursion, when computing $C_S^{i,j}$ for $[i \mathinner{.\,.} j] \subseteq [0 \mathinner{.\,.} \mathrm{bc}(\mathbf{G}_S))$, the algorithm again uses Theorem 5.8 combined with an exponential/binary search to find the corresponding indices i' and j'. Once the computation of $C_S^{i,j}$ is complete, the recursion returns i', h, j', $\mathsf{LZ}(\overline{T[\tau_0^{i'} \mathinner{.\,.} \tau_0^{h}]})$, and $\mathsf{LZ}(T(\tau_0^h \mathinner{.\,.} \tau_0^{j'}])$, along with the same information from the lower recursion levels. As proven in Lemma 4.17, this information, together with the provided input, suffices to retrieve the information $\{(c, T[\tau_0^c]) : c \in C_S\}$.

For the complexity analysis, note that the computation of c_{pref}, c_{suff}, c_{lsuff}, and c_{lpref} via Theorem 5.8 combined with an exponential/binary search requires $\tilde{O}(\sqrt{(k+w)m} + (k+w)^2) = \tilde{O}(\sqrt{km} + k^2)$ quantum time and $\tilde{O}(\sqrt{(k+w)m}) = \tilde{O}(\sqrt{km})$ query time. Regarding the analysis of the recursion, let $[i_0 \mathinner{.\,.} j_0], \ldots, [i_{d-1} \mathinner{.\,.} j_{d-1}]$ be the intervals considered by it at a fixed depth. Further, define $w_{i,j} := \sum_{c=i-1}^{j} \mathrm{w}_S(c)$ for $i, j \in [0 \mathinner{.\,.} \mathrm{bc}(\mathbf{G}_S))$. As already noted in the proof of Lemma 4.17, $\sum_{r=0}^{d-1} w_{i_r, j_r} = O(w)$. Suppose that for $[i \mathinner{.\,.} j] \in \{[i_r \mathinner{.\,.} j_r]\}_{r=0}^{d-1}$ we add $[i' \mathinner{.\,.} j']$ to $C_S^{i,j}$. Computing i' and j' by combining Theorem 5.8 with an exponential/binary search requires $\tilde{O}(\sqrt{(j-i+1) \cdot w_{i,j}})$ query time and $\tilde{O}(\sqrt{(j-i+1) \cdot w_{i,j}} + w_{i,j}^2)$ quantum time. The expression $\sum_{r=0}^{d-1} \sqrt{(j_r - i_r + 1) \cdot w_{i_r, j_r}}$ is maximized when the lengths $j_r - i_r + 1$ and the weights w_{i_r, j_r} are all roughly the same size. From $\sum_{r=0}^{d-1} w_{i_r, j_r} = O(w)$ and $\sum_{r=0}^{d-1} (j_r - i_r + 1) = O(m)$, we obtain that for the level of recursion we fixed the algorithm requires

$$\sum_{r=0}^{d-1} \tilde{O}\left(\sqrt{(j_r - i_r) \cdot w_{i_r, j_r}} + w_{i_r, j_r}^2\right) = \sum_{r=0}^{d-1} \tilde{O}\left(\sqrt{\frac{m}{d}} \cdot \sqrt{\frac{w}{d}}\right) + \tilde{O}\left(w^2\right) = \tilde{O}(\sqrt{wm} + w^2) = \tilde{O}(\sqrt{km} + k^2)$$

quantum time and $\tilde{O}(\sum_{r=0}^{d-1} \sqrt{(j_r - i_r + 1) \cdot w_{i_r, j_r}}) \le \tilde{O}(\sqrt{km})$ query time. The overall claimed complexities follow by using the fact that the recursion has depth at most $\tilde{O}(1)$. ◾

We remark that, given the information computed by Lemma 6.6, if $\mathrm{bc}(\mathbf{G}_S) = 0$, we can reconstruct P and T in linear time w.r.t. the size of the edge set of $\mathbf{G}_S$. The edit information of $\mathcal{X}$ for all $\mathcal{X} \in S$ allows us to associate to each node in $\mathbf{G}_S$ incident to a red edge the corresponding character from P or T. We can propagate characters to other nodes contained in red components of $\mathbf{G}_S$ by executing a depth search using exclusively black edges. By doing so, we are guaranteed to retrieve the correct characters of P and T because a black edge between two nodes indicates equality between the two corresponding characters. Moreover, from $\mathrm{bc}(\mathbf{G}_S) = 0$ follows that all characters are contained in red components, and therefore this procedure delivers us a full reconstruction of the characters of P and T.

Similarly, if we have $\mathrm{bc}(\mathbf{G}_S) > 0$ and if we are additionally provided with the information $\{(c, T[\tau_0^c]) : c \in C_S\}$, where C_S denotes a black cover computed by Lemma 6.7, we can reconstruct the strings $P^\#$ and $T^\#$ defined as in Theorem 4.27 in linear time. Using the same procedure as before we can retrieve all characters associated with nodes contained in red components. The additional information provided to us corresponds exactly to the characters associated with nodes contained in black components which, together with the characters associated with nodes contained in red components, we do not substitute with hashes when transforming P and T into $P^\#$ and $T^\#$. Regarding the positions substituted with hashes, note that we do not need to know the corresponding characters, but only in which black connected component they are contained. It is possible to get this last piece of information from the output of Lemma 6.6.

6.3 Framework for the Verification of Candidate Positions

Definition 6.8. *Let P be a string of length m, let T be a string of length $|T| \le {}^3\!/_2\, m$, and let k, K be positive thresholds such that $k \le K \le m$. We call a (P, T, k, K)-*edit oracle *a quantum oracle that takes as input a set S of alignments of cost at most K that enclose T and the edit information for each $\mathcal{X} \in S$, and that*

- *either returns $t, t' \in [0\,..\,|T|]$ such that $T[t\,..\,t')$ is a K-error occurrences of P in T and such that S* does not capture *$T[t\,..\,t')$,*
- *or reports that S* captures all *k-error occurrences of P in T.*

Lemma 6.9. *Let P be a string, let T be a string of length $|T| \le {}^3\!/_2\, m$, and let k and K be positive thresholds such that $k \le K \le m$. Suppose there exists a (P, T, k, K)-edit oracle requiring $\tilde{O}(q_t)$ quantum time and $\tilde{O}(q_r)$ query time for all input sets S of size $O(\log n)$.*

Then, there exists a quantum algorithm that takes in input a set S of alignments of cost at most K that enclose T of size $|S| = 2$ and the edit information $\mathsf{E}_{P,T}(\mathcal{X})$ for each $\mathcal{X} \in S$, and that outputs $\mathrm{Occ}_k^E(P, T)$ in $\tilde{O}(\sqrt{Km} + q_r)$ query time and $\tilde{O}(m + k^{3.5} + K^2 + q_t)$ quantum time.

Proof. The quantum algorithm proceeds as follows.

(i) The algorithm calls the (P, T, k, K)-edit oracle with input S. If S captures all k-error occurrences of P in T, then the algorithm proceeds to (ii). Otherwise, the algorithm passes the two returned positions t, t' by the (P, T, k, K)-edit oracle to the subroutine of Corollary 5.7, computes the exact edit distance between P and $T[t\,..\,t')$, and collects the edit information $\mathsf{E}_{P,T}(\mathcal{X})$, where $\mathcal{X} : P \rightsquigarrow T[t\,..\,t')$ denotes an optimal alignment of cost at most K. The algorithm adds $\mathcal{X}$ to S, and repeats (i).

(ii) Once S captures all k-error occurrences of P in T, the algorithm executes the classical procedure from Lemma 6.6. If $\mathrm{bc}(\mathbf{G}_S) = 0$, then the algorithm fully reconstructs P and T as remarked before, computes $\mathrm{Occ}_k^E(P, T)$ using Lemma 3.6, and returns $\mathrm{Occ}_k^E(P, T)$.

(iii) Otherwise, the algorithm computes a black cover C_S w.r.t. the weight function w_S via Lemma 6.7.

(iv) The algorithm constructs the strings $P^\#$ and $T^\#$ from Theorem 4.27 as remarked before. Finally, it computes $\mathrm{Occ}_k^E(P^\#, T^\#)$ using Lemma 3.6 and returns $\mathrm{Occ}_k^E(P^\#, T^\#)$.

Claim 6.10. *All of the following hold:*

(a) *the algorithm repeats (i) at most $O(\log m)$ times, and in particular $|S| = O(\log m)$; and*

(b) *the algorithm outputs the correct answer.*

Proof. Note, (a) follows directly from Lemma 4.25, because every time we add $\mathcal{X}$ to S, then $\mathrm{bc}(\mathbf{G}_{S\cup\{\mathcal{X}\}}) \leq \mathrm{bc}(\mathbf{G}_S)/2$, and if $\mathrm{bc}(\mathbf{G}_S) = 0$, then S captures all K-error occurrences of P in T.

Clearly, the algorithm always returns the correct answer when $\mathrm{bc}(\mathbf{G}_S) = 0$. If $\mathrm{bc}(\mathbf{G}_S) > 0$, then it suffices to apply Theorem 4.27. This concludes the proof for (b). ◿

From Claim 6.10(a) follows that the complexity for (i) is the same as the complexity required by the oracle plus the complexity from Corollary 5.7 up to a multiplicative logarithmic overhead. After (i) further queries are only needed in (ii) when using Lemma 6.7. Lastly, notice that the classical subroutines from Lemma 6.6 and the Lemma 3.6 require $\tilde{O}(m)$ and $\tilde{O}(m + k^{3.5})$ time, respectively. Putting together all these distinct complexities, we obtain the claimed query time and quantum time. ◾

Lemma 6.11. *Let P be a string of length m, let T be a string of length $|T| \leq {}^3\!/_2\, m$, and let k be a positive threshold such that $k \leq m$. Set $T_{\mathrm{pad}} := T\$^{m-1}$ where \$ denotes a unique special character. Then, there exists a quantum algorithm with the following properties.*

- *The algorithm takes in input:*
  - *a set H such that $\lfloor \mathrm{Occ}^E_k(P,T)/k \rfloor \subseteq H \subseteq \lfloor [\,0\,..\,|T|\,]/k \rfloor$;*
  - *a non-negative threshold $4k \leq K$; and*
  - *a quantum oracle evaluating a function $f : \Sigma^m \times \Sigma^m \longrightarrow \{\textsc{yes}, \textsc{no}\}$ such that:*
    - *for any $X, Y \in \Sigma^m$ the oracle outputs $f(X,Y)$ w.h.p.,*
    - *w.h.p. for every $t \in kH$ the value of $f(P, T_{\mathrm{pad}}[\,t\,..\,t+m\,))$ is a correct answer to the $(4k, K+2k)$-Gap Edit Distance instance with input P and $T_{\mathrm{pad}}[\,t\,..\,t+m\,)$, and*
    - *the oracle needs q_t quantum time and q_r query time.*
- *The algorithm outputs $\mathrm{Occ}^E_k(P,T)$.*
- *The algorithm requires $\tilde{O}(\sqrt{Km} + \sqrt{|H|} \cdot q_r)$ query time and $\tilde{O}(m + k^{3.5} + K^2 + \sqrt{|H|} \cdot q_t)$ quantum time.*

Proof. The quantum algorithm proceeds as follows.

(i) If $4K + 8k > m$, the algorithm reads P and T, computes $\mathrm{Occ}^E_k(P,T)$ via Lemma 3.6, and returns.

(ii) The algorithm executes Grover's search over all $t \in kH$ to find the smallest position t_{pref} and the largest position t_{suf} such that the oracle from the input returns yes on P and $T_{\mathrm{pad}}[\,t\,..\,t+m\,)$.

(iii) If no such t exists, the algorithm returns $\varnothing$.

(iv) Otherwise, the algorithm uses Corollary 5.7 to compute $\delta_E(P, T[\,t_{\mathrm{pref}}\,..\,\min(t_{\mathrm{pref}} + m, |T|)\,))$ and $\delta_E(P, T[\,t_{\mathrm{suf}}\,..\,\min(t_{\mathrm{suf}} + 2k + m, |T|)\,))$, collects the edit information, and adds the two corresponding alignments to a set S of alignments of P onto substrings of T of cost at most $K' := K + 2k$.

(v) Lastly, the algorithm uses the quantum subroutine from Lemma 6.9 on P, $T' = T[\,t_{\mathrm{pref}}\,..\,\min(t_{\mathrm{suf}} + 2k + m, |T|)\,)$, S, k, K' and a (P, T', k, K')-edit oracle which proceeds as follows.
  - First, it uses Lemma 6.6 to construct $\mathbf{G}_S$ and the corresponding combinatorial structures.
  - If $\mathrm{bc}(\mathbf{G}_S) = 0$, the (P, T', k, K')-edit oracle returns that S captures all k-error occurrences of P in T.
  - Otherwise, if $\mathrm{bc}(\mathbf{G}_S) > 0$, the (P, T', k, K')-edit oracle computes $H' \subseteq kH$ containing all positions $t \in kH$ such that $|\tau^0_i - t - \pi^0_0| > w + 2K'$ for all $i \in [\,0\,..\,n_0 - m_0\,]$. Finally, the oracle executes Grover's search over all $t \in H'$ to find a position t such the oracle from the input returns yes on P and $T_{\mathrm{pad}}[\,t\,..\,t+m\,)$. If no such t exits, then the oracle returns that S captures all k-error occurrences of P in T; otherwise, it returns t and $\min(t+m, |T|)$.

Let us assume that the input oracle always evaluates f successfully, and that for every $t \in kH$ the value of $f(P, T_{\mathrm{pad}}[\,t \,..\, t+m\,))$ is a correct answer to the $(4k, K+2k)$-Gap Edit Distance instance with input P and $T_{\mathrm{pad}}[\,t \,..\, t+m\,)$. Clearly, the algorithm is correct when $4K+8k > m$. Henceforth, assume $4K' = 4K+8k \le m$.

Claim 6.12. *Suppose $\mathrm{Occ}^E_k(P,T) \neq \emptyset$, and consider an alignment $\mathcal{X} : P \rightsquigarrow T[\,t \,..\, t'\,)$ of cost at most k. Set $t'' := k \cdot \lfloor t/k \rfloor \in kH$. Then, $\delta_E(P, T[\,t'' \,..\, \min(t''+m, |T|)\,)) \le 4k$ and $\delta_E(P, T_{\mathrm{pad}}[\,t'' \,..\, t''+m\,)) \le 4k$.*

Proof. Note, from $t \in \mathrm{Occ}^E_k(P,T)$ and $\lfloor \mathrm{Occ}^E_k(P,T)/k \rfloor \subseteq H$, follows $0 \le t - t'' \le k$. Furthermore, notice that $\delta_E(P, T[\,t \,..\, t'\,)) \le k$ implies $|m-(t'-t)| \le k$. If $t''+m \le |T|$, then we can use the triangle equality, and we obtain $|t''+m-t'| \le |m-(t'-t)| + |t-t''| \le 2k$. By combining $\delta_E(P, T[\,t \,..\, t'\,)) \le k$, $|t-t''| \le k$ and $|\min(t''+m, |T|) - t'| \le |t''+m-t'| \le 2k$, we get $\delta_E(P, T[\,t'' \,..\, \min(t''+m, |T|)\,)) \le 4k$ and $\delta_E(P, T_{\mathrm{pad}}[\,t'' \,..\, \min(t''+m)\,)) \le 4k$. ◿

Claim 6.12 implies that if $\mathrm{Occ}^E_k(P,T) \neq \emptyset$, then there must exist a position in kH such the input oracle returns yes on it in (ii). Hence, if the oracle returns $\emptyset$, then we must have $\mathrm{Occ}^E_k(P,T) = \emptyset$. Conversely, if the algorithm proceeds to (iv), then the correctness follows from Lemma 6.9, and the following Claim 6.13 and Claim 6.14.

Claim 6.13. *Suppose that the algorithm proceeds to (iv). Then, S encloses T' and the oracle defined in (v) is a (P, T', k, K')-edit oracle.*

Proof. Clearly, the two added alignments are such that one aligns with the prefix of T' and one with the suffix of T'. Moreover, from $4K' \le m$ and $|T'| \le |T| \le \frac{3}{2}m$ follows $|T'| \le 2m - 2K'$. We conclude that S encloses T'.

Now, note that the $(P, T', 4k, K')$-edit oracle behaves correctly when $\mathrm{bc}(\mathbf{G}_S) = 0$. It remains to show that the constructed oracle behaves accordingly when $\mathrm{bc}(\mathbf{G}_S) > 0$. We distinguish two cases depending on the outcome of Grover's search.

In the first case, Grover's search finds a position $t \in H'$ such that the input oracle returns yes on P and $T_{\mathrm{pad}}[\,t \,..\, t+m\,)$. Consequently, $\delta_E(P, T_{\mathrm{pad}}[\,t \,..\, t+m\,)) \le K'$, from which directly follows that $\delta_E(P, T[\,t \,..\, \min(t+m, |T|)\,)) \le K'$ and that the oracle behaves according to the definition of a (P, T', k, K')-edit oracle.

In the second case, for all $t \in kH$ we have that either $\delta_E(P, T_{\mathrm{pad}}[\,t \,..\, t+m\,)) > 4k$ or $|\tau^0_i - t - \pi^0_0| \le w + 2K'$ for some $i \in [\,0 \,..\, n_0 - m_0\,]$ holds. We need to prove that for an arbitrary alignment $\mathcal{X} : P \rightsquigarrow T[\,t \,..\, t'\,)$ of cost at most k we have $|\tau^0_i - t - \pi^0_0| \le w + 3K'$, where $i := \arg\min_{i' \in [\,0 \,..\, n_0\,)} |\tau^0_{i'} - t - \pi^0_0|$. Consider $t'' = k \cdot \lfloor t/k \rfloor \in kH$ for which $0 \le t - t'' \le k$ holds. From Claim 6.12 follows that $\delta_E(P, T_{\mathrm{pad}}[\,t \,..\, t+m\,)) \le 4k$. Therefore, we must have $|\tau^0_{i''} - t'' - \pi^0_0| \le w + 2K'$ for some $i'' \in [\,0 \,..\, n_0\,)$. We conclude

$$|\tau^0_i - t - \pi^0_0| \le |\tau^0_{i''} - t - \pi^0_0| \le |\tau^0_{i''} - t'' - \pi^0_0| + |t'' - t| \le w + 2K' + k \le w + 3K'.$$ ◿

Claim 6.14. *Suppose that the algorithm proceeds to (iv). Then, $\mathrm{Occ}^E_k(P,T') = \mathrm{Occ}^E_k(P,T)$.*

Proof. If $\mathrm{Occ}^E_k(P,T) = \emptyset$, then clearly $\mathrm{Occ}^E_k(P,T') = \emptyset$. Otherwise, if $\mathrm{Occ}^E_k(P,T) \neq \emptyset$, consider an alignment $\mathcal{X}$ of cost at most k that aligns P onto $T[\,t \,..\, t'\,)$. As proven in Claim 6.12, for $t'' = k \cdot \lfloor t/k \rfloor \in kH$ we have $0 \le t - t''$ and $|\min(t''+m, |T|) - t'| \le 2k$. Consequently, $t_{\mathsf{pref}} \le t'' \le t$ and $t' \le \min(t''+2k+m, |T|) \le \min(t_{\mathsf{suf}} + 2k + m, |T|)$. Since $T[\,t \,..\, t'\,)$ is a fragment of T' and since the alignment $\mathcal{X}$ we considered was arbitrary, we obtain $\mathrm{Occ}^E_k(P,T') = \mathrm{Occ}^E_k(P,T)$. ◿

Lastly, let us study the query and time complexity. If $4K + 8k > m$, then we need $O(K) = 8O(\sqrt{Km})$ query time and $O(\sqrt{Km})$ for (i). Step (ii) requires $\tilde{O}(\sqrt{|H|} \cdot q_r)$ query time and $\tilde{O}(\sqrt{|H|} \cdot q_r)$ quantum time. In (iv) retrieving all edits costs $\tilde{O}(\sqrt{Km})$ query time and $\tilde{O}(\sqrt{Km} + K^2)$ quantum time. From Lemma 6.9 we know that the remaining complexity analysis of (v) depends on the query and quantum time needed by the $(P, T', 4k, K')$-edit oracle. The oracle first uses Lemma 6.6 incurring no query time but taking $\tilde{O}(m)$ quantum time for all input sets of size at most $O(\log m)$. It is important to realize that it is possible to compute H' in $O(m)$ time by using the following observation. Consider $t, t' \in kH$ such that $t \leq t'$. Moreover, let $i = \arg\min_{\hat{\imath}} |\tau^0_{\hat{\imath}} - t - \pi^0_0|$ and let $i' = \arg\min_{\hat{\imath}} |\tau^0_{\hat{\imath}} - t' - \pi^0_0|$. Then, we must have $\tau^0_i \leq \tau^0_{i'}$. Hence, we can compute efficiently H' by cleverly scanning the elements of H from the left to the right, and by scanning simultaneously $\{\tau^0_{\hat{\imath}}\}_{\hat{\imath}}$ from the left to the right. Since $H' \subseteq H$ we have that the search over all $h \in H'$ requires again at most $\tilde{O}(\sqrt{|H|} \cdot q_r)$ query time and $\tilde{O}(\sqrt{|H|} \cdot q_t)$ quantum time. We conclude that (iv) takes $\tilde{O}(\sqrt{Km} + \sqrt{|H|} \cdot q_r)$ query time and $\tilde{O}(m + k^{3.5} + K^2 + \sqrt{|H|} \cdot q_t)$ quantum time. Putting together all terms, we obtain the claimed quantum time and query time. ◾

6.4 Verification of Few Candidate Position and Approximately Periodic Case

If we are provided with a set H such that $\lfloor \mathrm{Occ}^E_k(P, T)/k \rfloor \subseteq H$, like Lemma 2.1(c), then we can use the framework of Lemma 6.11 directly on H and the adapted gap edit distance algorithm from Section 5.3.

Lemma 6.15. *Let P be a string, and let T be a string of length $|T| \leq {}^3\!/_2\, m$, and let k be a positive threshold such that $k \leq m$. Then, there exists a quantum algorithm that takes as input a set H such that $\lfloor \mathrm{Occ}^E_k(P, T)/k \rfloor \subseteq H$, and outputs $\mathrm{Occ}^E_k(P, T)$ using $\hat{O}(\sqrt{|H|m})$ query time and $\hat{O}(\sqrt{|H|}m + k^{3.5})$ quantum time.*

Proof. Let $K = k^{o(1)}$. We use the quantum subroutine of Lemma 6.11 on H, k, K and the subroutine from Lemma 5.12 on a sequence of bits of length $\hat{O}(m)$ sampled uniformly at random. By doing so, we obtain a quantum algorithm that returns $\mathrm{Occ}^E_k(P, T)$ and uses $\hat{O}(\sqrt{|H|m})$ queries and $\hat{O}(\sqrt{|H|}m + k^{3.5})$ time. ◾

Another case covered by the framework is when a pattern R and a text T have small edit distance to two substrings of Q^∞ for some primitive *approximate period* Q. Let us denote these substrings with Q_R and Q_T. Then, if the edit distance of R and T to Q_R and Q_T is not too large, one can use the framework on a set that roughly corresponds to the occurrences of Q_R in Q_T (up to some small shifts) and an oracle which always outputs YES.

Lemma 6.16. *Let R be a string such that $m/8k \leq |R| \leq m$, let T be a string of length $|T| \leq {}^3\!/_2\, |R|$, and let k and κ be positive thresholds such that $k \leq m$ and $\kappa \leq |R|$. Suppose there exist a primitive string Q such that $|Q| \leq m/128k$, Q is a primitive* approximate period *of R with $\delta_E(R, {}^*Q^*) \leq \lceil 8k/m \cdot |R| \rceil$, and $\delta_E(T, {}^*Q^*) \leq 4 \cdot \lceil 8k/m \cdot |R| \rceil$.*

Then, there exists a quantum algorithm that takes as input a threshold $\kappa \leq \min(\lfloor 4k/m \cdot |R| \rfloor, k)$ and parameters ℓ_T, r_T, ℓ_R and r_R such that $\delta_E(T, Q^\infty[\ell_T \mathinner{.\,.} r_T)) \leq 4 \cdot \lceil 8k/m \cdot |R| \rceil$ and $\delta_E(R, Q^\infty[\ell_R \mathinner{.\,.} r_R)) \leq \lceil 8k/m \cdot |R| \rceil$. The algorithm computes $\mathrm{Occ}^E_\kappa(R, T)$ in $O(\sqrt{k/m} \cdot |R|)$ query time and $\tilde{O}(|R| + (k/m \cdot |R|)^{3.5})$ quantum time.

Proof. The algorithm proceeds as follows.

(i) The algorithm computes

$$H = \{x \in [0 \mathinner{.\,.} |T| - |R| + \kappa] : \text{there exists } x' \in \mathrm{Occ}_\infty \text{ such that } |x - x'| \leq 6K\},$$

where $K = \lceil 8k/m \cdot |R| \rceil$ and $\mathrm{Occ}_\infty = \{\ell_R - \ell_T + j \cdot |Q| : j \in \mathbb{Z}\}$.

(ii) The algorithm uses Lemma 6.11 on R, T, and κ. The input is $\lfloor H/\kappa \rfloor$, $22K$, and an oracle which always outputs YES.

Claim 6.17. *All of the following hold.*

(a) $\lfloor \mathrm{Occ}^E_\kappa(R,T)/\kappa \rfloor \subseteq \lfloor H/\kappa \rfloor$; *and*

(b) *for all* $t \in \kappa\lfloor H/\kappa \rfloor$ *it holds* $\delta_E(R, T[\,t\,..\,\min(t+|R|,|T|)\,)) \le 22K$.

Proof. Let $Q_T = Q^\infty[\,\ell_T\,..\,r_T\,)$, $Q_R = Q^\infty[\,\ell_R\,..\,r_R\,)$, and let $\mathcal{X}_R : R \rightsquigarrow Q_R$, $\mathcal{X}_T : T \rightsquigarrow Q_T$ be optimal alignments.

For (a) we prove $\mathrm{Occ}^E_\kappa(R,T) \subseteq H$. Suppose $\mathcal{X} : R \rightsquigarrow T[\,t\,..\,t'\,)$ is an alignment of cost at most κ. Note, since $\mathcal{X}$ has cost at most κ, we have $t \in [\,0\,..\,|T|-|R|+\kappa\,]$.

We want to prove that there exists $x' \in \mathrm{Occ}_\infty$ such that $|x'-t| \le 6K$. Consider the alignment $\mathcal{X}'_T \subseteq \mathcal{X}_T$ such that $\mathcal{X}'_T : T[\,t\,..\,t'\,) \rightsquigarrow Q_T[\,q\,..\,q'\,)$ for some $q, q' \in [\,0\,..\,|Q_T|\,]$, and the alignment $\mathcal{Y} = \mathcal{X}_R^{-1} \circ \mathcal{X} \circ \mathcal{X}'_T$ such that $\mathcal{Y} : Q_R \rightsquigarrow Q_T[\,q\,..\,q'\,)$. Note, Q_R contains at least $(|Q_R|-2|Q|)/|Q| = |Q_R|/|Q|-2$ full occurrences of Q. Since $\mathcal{Y}$ has cost at most $5K+\kappa$, at least

$$|Q_R|/|Q| - 2 - 5K - \kappa \ge 128k/m \cdot |R| - 47k/m \cdot |R| \ge 81k/m \cdot |R| \ge 1$$

full occurrences of Q are matched without edits in $\mathcal{Y}$. Therefore, we can consider an arbitrary occurrence that is matched without edits by $\mathcal{Y}$. Suppose this occurrence starts at position $q_R \in [\,0\,..\,|Q_R|-|Q|\,]$ in Q_R and is matched to a occurrence of Q starting at position $q_T \in [\,q\,..\,q'-|Q|\,]$ in Q_T. Further, let $\hat{r}$ and $\hat{t}$ be such that $(\hat{r}, q_R) \in \mathcal{X}_R$, $(\hat{r}, \hat{t}) \in \mathcal{X}$, $(\hat{t}, q_T) \in \mathcal{X}'_T$. Note, for $\hat{r}$ and $\hat{t}$ we have that $|(q_T - q_R) - t| \le K$, $|\hat{r} - (\hat{t} - t)| \le \kappa$, $|\hat{t} - q_T| \le 4K$. By using the triangle inequality, we obtain

$$|(q_T - q_R) - t| \le |\hat{r} - q_R| + |\hat{r} - (\hat{t} - t)| + |\hat{t} - q_T| \le 5K + \kappa \le 6K.$$

To conclude the proof of (a) it suffices to argue that $q_T - q_R \in \mathrm{Occ}_\infty$ (therefore we can set $x' = q_T - q_R$). Since Q is a primitive string, q_R and q_T are such that $q_R + \ell_R \equiv_{|Q|} 0$ and $q_T + \ell_T \equiv_{|Q|} 0$, from which directly follows $q_R - q_T \equiv_{|Q|} \ell_T - \ell_R$ and $q_T - q_R \in \mathrm{Occ}_\infty$.

For (b), consider an arbitrary $x \in [\,0\,..\,|T|-|R|+\kappa\,]$ such that there exists $x' \in \mathrm{Occ}_\infty$ with $|x - x'| \le 6K$. For sake of conciseness, let $y = \kappa\lfloor x/\kappa \rfloor$, and let $y' = \min(y+|R|,|T|)$. First, we would like to argue that

$$||R| - (y'-y)| \le K. \tag{8}$$

If $y + |R| \le |T|$, then (8) trivially holds because $y' - y = |R|$. Otherwise, if $y+|R| > |T|$, then from $x \le |T|-|R|+\kappa$ follows $\lfloor x/\kappa \rfloor \le \lfloor (|T|-|R|+\kappa)/\kappa \rfloor \le (|T|-|R|+\kappa)/\kappa$. Consequently, $y+|R| = \kappa\lfloor x/\kappa \rfloor + |R| \le |T| + \kappa \le |T| + K$ and (8) holds.

Next, consider $\mathcal{X}''_T \subseteq \mathcal{X}_T$ such that $\mathcal{X}''_T : T[\,y\,..\,y'\,) \rightsquigarrow Q_T[\,\ell'_T - \ell_T\,..\,r'_T - r_T\,)$ for some ℓ'_T, r'_T for which $Q^\infty[\,\ell'_T\,..\,r'_T\,) = Q_T[\,\ell'_T - \ell_T\,..\,r'_T - r_T\,)$ holds. We want to argue that the optimal alignment $\mathcal{A} : Q_T \rightsquigarrow Q_R$ has cost at most $17K$. This is sufficient in order to prove (b), because then $\mathcal{X}_R \circ \mathcal{A}^{-1} \circ \left(\mathcal{X}''_T\right)^{-1} : R \rightsquigarrow T[\,y\,..\,y'\,)$ has cost at most $22K$. From $(y, \ell'_T - \ell_T) \in \mathcal{X}_T$ together with

$$|y - x'| = |\kappa\lfloor x/\kappa \rfloor - x'| \le |\kappa\lfloor x/\kappa \rfloor - x| + |x - x'| \le \kappa|\lfloor x/\kappa \rfloor - x/\kappa| + 6K \le \kappa + 6K \le 7K$$

follows $|x' - (\ell'_T - \ell_T)| \le |y - (\ell'_T - \ell_T)| + |y - x'| \le 4K + 7K \le 11K$. Now, since $x' \in \mathrm{Occ}_\infty$ there exists $j \in \mathbb{Z}$ such that $x' = \ell_R - \ell_T + j \cdot |Q|$. Consequently, for such j it holds that

$$|\ell_R - \ell'_T + j \cdot |Q|| = |x' - (\ell'_T - \ell_T)| \le 11K. \tag{9}$$

Moreover, from $\delta_E(T[\,y\,..\,y'\,), Q^\infty[\,\ell'_T\,..\,r'_T\,)) \le 4K$, $\delta_E(R, Q^\infty[\,\ell_R\,..\,r_R\,)) \le K$ and (8) follows

$$|(r_R - \ell_R) - (r'_T - \ell'_T)| \le ||R| - (r'_T - \ell'_T)| + ||R| - (y'-y)| + |(y'-y) - (r'_T - \ell'_T)| \le 6K. \tag{10}$$

By combining (9) and (10), we conclude that the cost of $\mathcal{A}$ is at most $17k$. ◁

The correctness of the algorithm follows from Lemma 6.11 and Claim 6.17. Regarding the complexity analysis, notice that (i) can be implemented using $O(|R|)$ time and no queries to the input strings. This is because for every $x \in [\,0\,..\,|T|-|R|+\kappa\,]$ we can calculate in $O(1)$ the parameter $j \in \mathbb{Z}$ minimizing $|x-(\ell_R-\ell_T-j\cdot|Q|)|$. Further, by using Lemma 6.11 in (ii) we incur $O(\sqrt{k/m\cdot|R|})$ queries and $O(|R|+(k/m\cdot|R|)^{3.5})$ time. ◾

6.5 Bringing All Components Together

In this last subsection, we devise quantum subroutines to handle the three cases of Lemma 2.1. Before devising these three subroutines and explaining how to patch the three cases together, we demonstrate how to drop the assumption in Lemma 6.16 that T has a primitive *approximate period* Q.

Lemma 6.18 (Compare with Lemma 6.16)**.** *Let R be a string such that $m/8k \le |R| \le m$, let T be a string of length $|T| \le {}^3\!/\!_2\, m$, and let k be a positive threshold such that $k \le m$. Suppose there exist a primitive string Q such that $|Q| \le m/128k$, and $\delta_E(R, {}^*Q^*) \le \lceil 8k/m\cdot|R|\rceil$.*

Then, there exists a quantum algorithm that takes as input a positive threshold $\kappa \le \min(\lfloor 4k/m\cdot|R|\rfloor, k)$ and parameters ℓ_R and r_R such that $\delta_E(R, Q^\infty[\,\ell_R\,..\,r_R\,]) \le \lceil 8k/m\cdot|R|\rceil$. The algorithm computes $\mathrm{Occ}^E_\kappa(R,T)$ in $\tilde{O}(\sqrt{km})$ query time and $\tilde{O}(m+k^{3.5})$ quantum time.

Proof. First, we want to argue that we may restrict ourselves to the case $k \le m/32$ so that $\kappa \le |R|/8$. Indeed, if $k > m/32$, then we can read the string using $\tilde{O}(m) = \tilde{O}(\sqrt{km})$ queries, and we can use Lemma 3.6 to compute $\mathrm{Occ}^E_\kappa(R,T)$ in $\tilde{O}(m + m/|R|\cdot\kappa^{3.5}) = \tilde{O}(m+k^{3.5})$ time incurring no further queries.

Next, using a standard technique, we would like to argue that it is sufficient to consider the case $|T| \le {}^3\!/\!_2\,|R|$. Assume that we have a quantum subroutine that computes $\mathrm{Occ}^E_\kappa(R,T')$ for any fragment T' of T such that $|T'| \le {}^3\!/\!_2\,|R|$ in $\tilde{O}(\sqrt{k/m\cdot|R|})$ query time and $\tilde{O}(|R|+(k/m\cdot|R|)^{3.5})$ quantum time. Let us divide T into $O(m/|R|)$ contiguous blocks of length $|R|/2-\kappa$ (the last block might be shorter). Next, we iterate over all blocks. When iterating over a block of the form $T[\,i\,..\,\min(i+|R|/2-\kappa, |T|)\,)$, we call the subroutine on R, $T[\,i\,..\,\min(i+\frac{3}{2}\cdot|R|, |T|)\,)$, and κ. This allows us to retrieve $\mathrm{Occ}^E_\kappa(R, T[\,i\,..\,\min(i+{}^3\!/\!_2\,|R|, |T|)\,))$ for which

$$\mathrm{Occ}^E_\kappa(R,T)\cap[\,i\,..\,\min(i+|R|/2-\kappa,|T|)\,) \subseteq \mathrm{Occ}^E_\kappa(R, T[\,i\,..\,\min(i+{}^3\!/\!_2\,|R|,|T|)\,))$$

holds. Consequently, the union of the returned occurrences from all calls to the subroutine is $\mathrm{Occ}^E_\kappa(R,T)$. By iterating over all blocks and calling the subroutine for each of the blocks, we need at most $\tilde{O}(m/|R|\cdot\sqrt{k/m}\cdot|R|) = \tilde{O}(\sqrt{km})$ query time and $\tilde{O}(m/|R|\cdot(|R|+(k/m\cdot|R|)^{3.5})) = \tilde{O}(m+k^{3.5})$ quantum time.

Henceforth, assume $|T| \le {}^3\!/\!_2\,|R|$. We want to prove the existence of a subroutine that computes $\mathrm{Occ}^E_\kappa(R,T)$ in $O(\sqrt{k/m}\cdot|R|)$ query time and $\tilde{O}(|R|+(k/m\cdot|R|)^{3.5})$ quantum time. For that purpose, consider the following algorithm:

(i) Divide $T[\,|R|/2+\kappa\,..\,|R|-\kappa\,)$ into segments of length $2|Q|$. Select uniformly at random one of the $h := \lfloor(|R|/2-2\kappa)/2|Q|\rfloor$ full segments, and try to find an exact occurrence of Q in the selected segment using Theorem 5.3.

(ii) Suppose the search is successful, and we have found in the selected segment an exact match of Q at position τ of T. By using Lemma 5.10 combined with an exponential/binary search, choose i' to be the smallest i' such that $\delta_E(T[\,i'\,..\,\tau\,), {}^*Q) \le 2\cdot\lceil 8k/m\cdot|R|\rceil$, and similarly, choose j' be the largest j' such that $\delta_E(T[\,\tau+|Q|\,..\,j'\,), Q^*) \le 2\cdot\lceil 8k/m\cdot|R|\rceil$. Using the output from the subroutine of Lemma 5.10, and by performing some elementary calculations, we obtain ℓ_T, r_T such that $\delta_E(T[\,i'\,..\,j'\,), Q^\infty[\,\ell_T\,..\,r_T\,)) \le 4\cdot\lceil 8k/m\cdot|R|\rceil$.

(iii) The algorithm uses Lemma 6.16 on $R, T[\,i' \mathinner{.\,.} j'\,)$ and κ with approximate period Q. The threshold is set to be κ which is passed to the algorithm together with the parameters ℓ_T, r_T, ℓ_R and r_R.

Claim 6.19. *Fix i, j such that $\delta_E(R, T[\,i \mathinner{.\,.} j\,)) \le \kappa$. Then, with constant probability all of the following hold:*

(a) *the search at step (i) succeeds;*

(b) *$T[\,i \mathinner{.\,.} j\,)$ is a fragment of $T[\,i' \mathinner{.\,.} j'\,)$; and*

(c) *$\delta_E(T[\,i' \mathinner{.\,.} j'\,), {}^*Q^*) \le 4 \cdot \lceil 8k/m \cdot |R| \rceil$.*

Proof. First, notice that from $|R|/2 - 2\kappa \ge |R|/4 \ge m/64k \ge 2|Q|$ follows $h > 0$, and therefore the selection process in (i) is well defined. From $i \le |T| - |R| + \kappa \le |R|/2 + \kappa$ and $j \ge |R| - \kappa$, follows that all of the h segments are fragments of $T[\,i \mathinner{.\,.} j\,)$. Since $\delta_E(R, T[\,i \mathinner{.\,.} j\,)) \le \kappa$, at least $h - \kappa$ segments contain no edit in an optimal alignment $\mathcal{A} : R \rightsquigarrow T[\,i \mathinner{.\,.} j\,)$. Let $\mathcal{A}'$ be an optimal alignment of $\delta_E(R, {}^*Q^*)$. Since $\delta_E(R, {}^*Q^*) \le \lceil 8k/m \cdot |R| \rceil$, at least $h - \kappa - \lceil 8k/m \cdot |R| \rceil$ of the segments contain no edit in the alignment $\mathcal{A}^{-1} \circ \mathcal{A}'$. Each of these segments corresponds to a cyclic rotation of QQ, and therefore contains at least one copy of Q. As a result, by selecting uniformly at random one of the h segments and by matching Q exactly in it, with a probability at least

$$\frac{h - \kappa - \lceil 8k/m \cdot |R| \rceil}{h} \ge 1 - \frac{\kappa + \lceil 8k/m \cdot |R| \rceil}{h} \ge 1 - 4|Q| \cdot \frac{\lfloor 4k/m \cdot |R| \rfloor + \lceil 8k/m \cdot |R| \rceil}{|R|} \ge \Omega(1)$$

we succeed in the search. This proves (a).

For (b), suppose $\mathcal{A}$ aligns $T[\,\tau \mathinner{.\,.} \tau + |Q|\,)$ onto $R[\,\rho \mathinner{.\,.} \rho + |Q|\,)$. Note, $\mathcal{A}^{-1} \circ \mathcal{A}'$ aligns $T[\,\tau + |Q| \mathinner{.\,.} j\,)$ onto a prefix of Q^∞, and thus

$$\begin{aligned}
\delta_E(T[\,\tau + |Q| \mathinner{.\,.} j\,), Q^*) &\le \delta_E^{\mathcal{A}}(R[\,\rho + |Q| \mathinner{.\,.} m\,), T[\,\tau + |Q| \mathinner{.\,.} j\,)) + \delta_E(R[\,\rho + |Q| \mathinner{.\,.} m\,), Q^*)\\
&\le \delta_E^{\mathcal{A}}(R, T) + \delta_E(R, {}^*Q^*)\\
&\le \kappa + \lceil 8k/m \cdot |R| \rceil\\
&\le 2 \cdot \lceil 8k/m \cdot |R| \rceil.
\end{aligned}$$

Consequently $j \le j'$, and using a symmetric argument, we deduce $i' \le i$.

Lastly, for (c) notice that

$$\delta_E(T[\,i' \mathinner{.\,.} j'\,), {}^*Q^*) \le \delta_E(T[\,i' \mathinner{.\,.} \tau\,), {}^*Q) + \delta_E(T[\,\tau + |Q| \mathinner{.\,.} j'\,), Q^*) \le 4 \cdot \lceil 8k/m \cdot |R| \rceil,$$

which concludes the proof.

Hence, any κ-error occurrence of R in T is contained in T' with constant probability. By repeating $\tilde{O}(1)$ times the selection of the fragment T', we ensure that with high probability all κ-error occurrence of R in T are contained in some of the selected T'. More specifically, we select independently $T'_1, T'_2, \ldots, T'_{\mathsf{rep}}$ for $\mathsf{rep} = \tilde{O}(1)$ as described in the algorithm, and with high probability we have $\mathrm{Occ}^E_\kappa(R, T) = \bigcup_{i=1}^{\mathsf{rep}} \mathrm{Occ}^E_\kappa(R, T'_i)$. We conclude that it suffices to apply Lemma 6.16 on R and on each of the fragments $T'_1, T'_2, \ldots, T'_{\mathsf{rep}}$.

Regarding the complexity analysis, note that (i) requires $\tilde{O}(\sqrt{|Q|})$ query time and $\tilde{O}(\sqrt{|Q|})$ quantum time. The calls to Lemma 5.10 in (ii) intertwined with the exponential/binary search need at most $\tilde{O}(\sqrt{k/m} \cdot |R| + k/m \cdot |R|) = \tilde{O}(\sqrt{k/m} \cdot |R|)$ query time and $\tilde{O}(\sqrt{k/m} \cdot |R| + (k/m \cdot |R|)^2)$ quantum time. Using Lemma 6.16 in (ii) requires $\tilde{O}(\sqrt{km})$ query time and $\tilde{O}(m + k^{3.5})$ quantum time. Finally, boosting the algorithm's success probability yields a multiplicative logarithmic-factor overhead in the query time and quantum time. Putting together all the factors we obtain the desired query and quantum time for the subroutine. ∎

Now, let us tackle each of the three cases outlined in Lemma 2.1 individually, starting with the last case and working our way backwards. Applying Lemma 6.18 with $R = P$ and $\kappa = k$ allows us to resolve case (c) of Lemma 2.1.

Corollary 6.20. *Let P be a string, let T be a string of length $|T| \le {}^3\!/_2\, m$, and let k be a positive threshold such that $k \le m$. Suppose we are in case (c) of Lemma 2.1. That is, P has a primitive* approximate period *Q with $|Q| \le m/128k$ and $\delta_E(P, {}^*Q^*) < 8k$.*

Then, we can compute $\mathrm{Occ}^E_k(P,T)$ in $\tilde{O}(\sqrt{km})$ query time and $\tilde{O}(m + k^{3.5})$ quantum time.

In Lemma 6.22, we discuss how to tackle the case where the pattern contains disjoint repetitive regions, i.e., we are in case (b) of Lemma 2.1. For that purpose, we will need to use the following results from [CKW20].

Lemma 6.21 (Corollary 5.18 of [CKW20])**.** *Let P denote a string of length m, let T denote a string of length n, and let $k \le m$ denote a positive threshold. Suppose that there are a positive integer $d \ge 2k$ and a primitive string Q with $|Q| \le m/8d$ and $\delta_E(P, {}^*Q^*) = d$. Then $|\lfloor \mathrm{Occ}^E_k(P,T)/d \rfloor| \le 1216 \cdot n/m \cdot d$.*

Lemma 6.22. *Let P be a pattern, and let T be a text of length $|T| \le {}^3\!/_2\, m$. Suppose we are in case (b) of Lemma 2.1. That is, the string P contains disjoint* repetitive regions *$R_1, \dots, R_r$ of total length $\sum_{i=1}^r |R_i| \ge 3/8 \cdot m$ such that each region R_i satisfies $|R_i| \ge m/8k$ and has a primitive* approximate period *Q_i with $|Q_i| \le m/128k$ and $\delta_E(R_i, {}^*Q_i^*) = \lceil 8k/m \cdot |R_i| \rceil$.*

Then, we can compute $\mathrm{Occ}^E_k(P,T)$ in $\hat{O}(\sqrt{km})$ query time and $\hat{O}(\sqrt{k}m + k^{3.5})$ quantum time.

Proof. Consider the following procedure:

(i) Select a repetitive region R among $R_1, \dots, R_r$ with probability proportional to their length, i.e.,

$$\Pr[R = R_i] = \frac{|R_i|}{\sum_{i'=1}^r |R_{i'}|}$$

for $i \in [\,1\,..\,r\,]$.

(ii) Compute $\mathrm{Occ}^E_\kappa(R,T)$ using Lemma 6.18, where $\kappa = \lfloor 4k/m \cdot |R| \rfloor$.

(iii) Suppose $R = P[\,\rho\,..\,\rho + |R|\,)$ and let $d = \lceil 8k/m \cdot |R_i| \rceil$. Compute and return the set

$$H = \{x \in [\,0\,..\,|T|\,] : \text{there exists } x' \in d \cdot \lfloor \mathrm{Occ}^E_\kappa(R,T)/d \rfloor \text{ such that } |x' - \rho - x| \le 10k\}.$$

Claim 6.23. *Let $\mathcal{A} : P \rightsquigarrow T[\,t\,..\,t'\,)$ be an alignment of cost at most k. Then with constant probability all of the following hold:*

(a) $\delta_E^{\mathcal{A}}(P[\,\rho\,..\,\rho + |R|\,), \mathcal{A}(P[\,\rho\,..\,\rho + |R|\,))) \le \lfloor 4k/m \cdot |R| \rfloor$*; and*

(b) $t \in H$.

Moreover, $|\lfloor H/k \rfloor| = O(k)$.

Proof. For (a), define ρ_i for $i \in [\,1\,..\,r\,]$ such that $R_i = P[\,\rho_i\,..\,\rho_i + |R_i|\,)$. Further, define the index set $I := \{i \in [\,1\,..\,r\,] : \delta_E^{\mathcal{A}}(P[\,\rho_i\,..\,\rho_i + |R|\,), \mathcal{A}(P[\,\rho_i\,..\,\rho_i + |R|\,))) \le \lfloor 4k/m \cdot |R_i| \rfloor\}$. For sake of contradiction, assume $\sum_{i \in I} |R_i| < m/16$. As a result, $\sum_{i \notin I} |R_i| = \sum_{i=1}^r |R_i| - \sum_{i \in I} |R_i| > 3/8 \cdot m - 1/16 \cdot m \ge 5/16 \cdot m$, and therefore

$$\delta_E^{\mathcal{A}}(P, T[\,t\,..\,t'\,)) \ge \sum_{i \in I} \delta_E^{\mathcal{A}}(P[\,\rho_i\,..\,\rho_i + |R|\,), \mathcal{A}(P[\,\rho_i\,..\,\rho_i + |R|\,))) \ge \sum_{i \notin I} 4k/m \cdot |R_i| \ge 4k/m \sum_{i \notin I} |R_i| > k,$$

which is a contradiction. Hence, we have that $\sum_{i \in I} |R_i| \ge m/16$, from which follows directly $\Pr[R \in I] = \sum_{i \in I} |R_i| / \sum_{i=1}^r |R_i| \ge (m/16)/m \ge \Omega(1)$, concluding the proof for (a).

For (b) let t_R be such that $(\rho, t_R) \in \mathcal{A}$ for which $t_R \in \mathrm{Occ}^E_\kappa(R, T)$ holds. From $|(t_R - t) - \rho| \le k$ together with the triangle inequality follows $|t - (d\lfloor t_R/d \rfloor - \rho)| \le |d\lfloor t_R/d \rfloor - t_R| + |(t_R - t) - \rho| \le d + k \le 10k$. Hence, for $x' = d\lfloor t_R/d \rfloor$ we have $|x' - \rho - t| \le 10k$ and $t \in H$ holds.

For the last part of the claim it suffices to apply Lemma 6.21 on R (in the role of P), T, κ (in the role of k), and d. Note that all assumptions are satisfied because $\lceil 8k/m \cdot |R_i| \rceil = d \ge 2\kappa \ge 2\lfloor 4k/m \cdot |R| \rfloor$, $|Q| \le m/128k \le |R|/8d$, and $\delta_E(R, {}^*Q^*) = d$. We obtain

$$|\lfloor \mathrm{Occ}^E_\kappa(R, T)/d \rfloor| \le 1216 \cdot |T|/|R| \cdot d \le 608m/|R| \cdot d \le 14592k = \mathcal{O}(k).$$

By rewriting $H = \bigcup_{x' \in d \cdot \lfloor \mathrm{Occ}^E_\kappa(R,T)/d \rfloor} H_{x'}$, where $H_{x'} = \{x \in [\,0\,..\,|T|\,] : |x' - \rho - x| \le 10k\}$, we obtain $|\lfloor H_{x'}/k \rfloor| = \mathcal{O}(1)$ for all $x' \in d \cdot \lfloor \mathrm{Occ}^E_\kappa(R, T)/d \rfloor$. Consequently, $|\lfloor H/k \rfloor| \le \sum_{x'} |\lfloor H_{x'}/k \rfloor| = \mathcal{O}(k)$. ◿

Similarly to Lemma 6.18, we repeat $\tilde{\mathcal{O}}(1)$ times the procedure described above. By doing so, we obtain sets $H_1, \ldots H_{\mathsf{rep}}$ where $\mathsf{rep} = \tilde{\mathcal{O}}(1)$, for which with high probability $\mathrm{Occ}^E_k(P, T) \subseteq \bigcup_{i=1}^{\mathsf{rep}} H_i$ holds. Finally, we use the quantum subroutine from Lemma 6.15 on P, T, k and $\bigcup_{i=1}^{\mathsf{rep}} \lfloor H_i/k \rfloor$, and we return $\mathrm{Occ}^E_k(P, T)$.

First, we analyze the complexity of a single run of the procedure. Selecting R in (i) requires $\mathcal{O}(r) = \mathcal{O}(k)$ time and no queries to the input strings. Computing $\mathrm{Occ}^E_\kappa(R, T)$ using Lemma 6.18 in (ii) requires $\tilde{\mathcal{O}}(\sqrt{km})$ query time and $\tilde{\mathcal{O}}(m + k^{3.5})$ quantum time. Lastly, we can construct $\lfloor H/k \rfloor$ in $\mathcal{O}(\mathrm{Occ}^E_\kappa(R, T))$ time incurring no further queries. This concludes the analysis of a single run of the procedure. Now, from Claim 6.23 follows that $|\lfloor H_i/k \rfloor| = \mathcal{O}(k)$ for all $i \in [\,1\,..\,\mathsf{rep}\,]$. Consequently, $|\bigcup_{i=1}^{\mathsf{rep}} \lfloor H_i/k \rfloor| = \tilde{\mathcal{O}}(k)$ and by using the subroutine from Lemma 6.15 we spend another $\hat{\mathcal{O}}(\sqrt{km})$ query time and $\hat{\mathcal{O}}(\sqrt{k}m + k^{3.5})$ quantum time. Putting together all the different factors, we obtain the claimed query and quantum time. ▪

Lastly, we settle case (c) of Lemma 2.1.

Lemma 6.24. *Let P be a string, let T be a string of length $|T| \le {}^3\!/_2\, m$, and let k be a positive threshold such that $k \le m$. Suppose we are in case (a) of Lemma 2.1. That is, P contains $2k$ disjoint* breaks $B_1, \ldots, B_{2k}$ *each having period* $\mathrm{per}(B_i) > m/128k$ *and length* $|B_i| = \lfloor m/8k \rfloor$.

Then, we can compute $\mathrm{Occ}^E_k(P, T)$ in $\hat{\mathcal{O}}(\sqrt{km})$ query time and $\hat{\mathcal{O}}(\sqrt{k}m + k^{3.5})$ quantum time.

Proof. Consider the following procedure:

(1) select uniformly at random a break B among the $2k$ disjoint breaks $B_1, \ldots, B_{2k}$;
(2) compute $\mathrm{Occ}(B, T)$ using Lemma 5.5; lastly
(3) suppose $B = P[\,\beta\,..\,\beta + |B|\,)$. Compute and return the set

$$H = \{x \in [\,0\,..\,|T|\,] : \text{there exists } x' \in \mathrm{Occ}(B, T) \text{ such that } |x' - \beta - x| \le k\}.$$

Claim 6.25. *Let $\mathcal{A} : P \rightsquigarrow T[\,t\,..\,t'\,)$ be an alignment of cost at most k. Then with constant probability all of the following hold:*

(a) $\delta^{\mathcal{A}}_E(P[\,\beta\,..\,\beta + |B|\,), \mathcal{A}(P[\,\beta\,..\,\beta + |B|\,))) = 0$; *and*
(b) $t \in H$.

Moreover, $|\lfloor H/k \rfloor| = \mathcal{O}(k)$.

Proof. For (a), note that the probability that no edit occurs in break B is at least $k/2k = 1/2$.

For (b), consider x' such that $(\beta, x') \in \mathcal{A}$ for which $x' \in \mathrm{Occ}(B, T)$ holds. Since $\mathcal{A}$ has cost at most k, $|(x' - t) - \beta| \le k$ and $t \in H$.

For the last part of the claim, note that $|\mathrm{Occ}(B, T)| \le \lceil |T|/\mathrm{per}(B) \rceil \le 192k = \mathcal{O}(k)$ because $\mathrm{per}(B) > m/128k$ and $|T| \le {}^3\!/_2\, m$. By rewriting $H = \bigcup_{x' \in \mathrm{Occ}(B,T)} H_{x'}$, where $H_{x'} = \{x \in [\,0\,..\,|T|\,] : |x' - \beta - x| \le k\}$, we obtain $|\lfloor H_{x'}/k \rfloor| = \mathcal{O}(1)$ for all $x' \in \mathrm{Occ}(B, T)$. Consequently, $|\lfloor H/k \rfloor| \le \sum_{x'} |\lfloor H_{x'}/k \rfloor| = \mathcal{O}(k)$. ◿

Similarly to Lemma 6.18 and Lemma 6.22, we repeat $\tilde{O}(1)$ times the procedure described above. By doing so, we obtain sets $H_1, \ldots H_{\mathsf{rep}}$, where $\mathsf{rep} = \tilde{O}(1)$, for which with high probability $\mathrm{Occ}^E_k(P,T) \subseteq \bigcup_{i=1}^{\mathsf{rep}} H_i$ holds. Finally, we use the quantum subroutine from Lemma 6.15 on P, T, k and $\bigcup_{i=1}^{\mathsf{rep}} \lfloor H_i/k \rfloor$, and we return $\mathrm{Occ}^E_k(P,T)$.

First, we analyze the complexity of a single run of the procedure. Selecting B in (1) requires $O(k)$ time and no queries to the input strings. Computing $\mathrm{Occ}(B,T)$ using Lemma 5.5 in (2) requires $\tilde{O}(\sqrt{km})$ query time and quantum time. Lastly, we can construct H in $O(|\mathrm{Occ}(B,T)|)$ time incurring no further queries. This concludes the analysis of a single run of the procedure. Now, from Section 6.5 follows that $|\lfloor H_i/k \rfloor| = O(k)$ for all $i \in [1\,..\,\mathsf{rep}]$. Consequently, $|\bigcup_{i=1}^{\mathsf{rep}} \lfloor H_i/k \rfloor| = \tilde{O}(k)$ and by using the subroutine from Lemma 6.15 we spend another $\hat{O}(\sqrt{km})$ query time and $\hat{O}(\sqrt{k}m + k^{3.5})$ quantum time. Putting together all the different factors, we obtain the claimed query and quantum time. ◾

Having addressed all three cases outlined in Lemma 2.1, we are now prepared to present the proof of Main Theorem 3.

Main Theorem 3. *Let P denote a pattern of length m, let T denote a text of length n, and let $k > 0$ denote an integer threshold.*

(1) *There is a quantum algorithm that solves the Pattern Matching with Edits problem using $\hat{O}(n/m \cdot \sqrt{km})$ queries and $\hat{O}(n/m \cdot (\sqrt{k}m + k^{3.5}))$ time.*

(2) *There is a quantum algorithm deciding whether $\mathrm{Occ}^E_k(P,T) \neq \emptyset$ using $\hat{O}(\sqrt{n/m} \cdot \sqrt{km})$ queries and $\hat{O}(\sqrt{n/m} \cdot (\sqrt{k}m + k^{3.5}))$ time.*

Proof. First, we want to argue that we restrict ourselves to the case $k \le m/4$. Indeed, if $k > m/4$, we can read the strings P and T using $O(n+m) = O(\sqrt{kn})$ queries, and use the classical algorithm from Lemma 3.6 which requires $\hat{O}(n + k^{3.5} \cdot n/m)$ time. This classical algorithm effectively provides the answers we seek for both claims.

Next, we observe that there exists a quantum algorithm that solves the Pattern Matching with Edits problem in the case that $n \le {}^3\!/_2\, m$ using $\hat{O}\left(\sqrt{km}\right)$ queries and $\hat{O}(\sqrt{k}m + k^{3.5})$. We use Lemma 6.5 to find which case of Lemma 2.1 holds for the pattern P. Depending on the case, we use the corresponding lemma. More specifically, for (a) we use Lemma 6.24, for (b) we use Lemma 6.22, and for (c) we use Corollary 6.20.

To complete the proof, we employ a standard technique to partition T into $O(n/m)$ contiguous blocks, each of length $m/4$ (with the last block potentially being shorter). For every block that starts at position $i \in [0\,..\,|T|)$, we consider the segment $T[i\,..\,\min(n, i + {}^3\!/_2\, m))$ having length at most ${}^3\!/_2\, m$. Importantly, since $k \le m/4$, every fragment $T[t\,..\,t')$ with $\delta_E(P, T[t\,..\,t')) \le k$ has a corresponding fully contained segment.

For the first claim, we iterate over all such segments, applying the quantum algorithm to P, the current segment, and k. The final set of occurrences is obtained by taking the union of all sets returned by the quantum algorithm.

For the second claim, Grover's search is employed over all segments, utilizing a function that determines whether the set of k-error occurrences of P in the segment is empty or not. ◾

Main Theorem 4. *Let us fix integers $n \ge m > k > 0$.*

(1) *Every quantum algorithm that solves the Pattern Matching with Edits problem uses $\Omega(n/m \cdot \sqrt{k(m-k)})$ queries for $P = \mathtt{0}^m$ and some $T \in \{\mathtt{0},\mathtt{1}\}^n$.*

(2) *Every quantum algorithm that decides whether* $\mathrm{Occ}^E_k(P,T) \neq \emptyset$ *uses* $\Omega(\sqrt{n/m}\cdot\sqrt{k(m-k)})$ *queries for* $P = \mathtt{0}^m$ *and some* $T \in \{\mathtt{0},\mathtt{1}\}^n$.

Proof. Let $p = \lfloor (n+m)/(2m) \rfloor$. For a tuple $\mathbf{A} = (A_0,\ldots,A_{p-1})$ of p subsets of $[\,0\,..\,m\,)$, consider a text $T_{\mathbf{A}} \in \{\mathtt{0},\mathtt{1}\}^n$ such that:

$$T_{\mathbf{A}}[i] = \begin{cases} \mathtt{0} & \text{if } i = 2qm + r \text{ for } q \in [\,0\,..\,p\,) \text{ and } r \in [\,0\,..\,m\,) \setminus A_q, \\ \mathtt{1} & \text{otherwise.} \end{cases}$$

Claim 6.26. *For* $q \in [\,0\,..\,p\,)$, *we have* $\mathrm{Occ}^E_k(P,T_{\mathbf{A}}) \cap [\,2qm-m\,..\,2qm+m\,) \neq \emptyset$ *if and only if* $|A_q| \le k$.

Proof. Observe that $\delta_E(P, T_{\mathbf{A}}[\,2qm\,..\,2qm+m\,)) = |A_q|$ because $T_{\mathbf{A}}[\,2qm\,..\,2qm+m\,)$ consists of $|A_q|$ copies of $\mathtt{1}$ and $m-|A_q|$ copies of $\mathtt{0}$. Thus, $|A_q| \le k$ implies $2qm \in \mathrm{Occ}^E_k(P,T_{\mathbf{A}})$.

For the converse implication, suppose that $\delta_E(P, T[\,i\,..\,j\,)) \le k$ holds for some fragment $T_{\mathbf{A}}[\,i\,..\,j\,)$ with $i \in [\,2qm-m\,..\,2qm+m\,)$. In particular, this means that $T_{\mathbf{A}}[\,i\,..\,j\,)$ contains at least $m-k$ copies of $\mathtt{0}$ and at most k copies of $\mathtt{1}$. The fragment $T_{\mathbf{A}}[\,2qm\,..\,2qm+m\,)$ is separated from other $\mathtt{0}$s in $T_{\mathbf{A}}$ by at least $m > k$ copies of $\mathtt{1}$ in both directions, so all $\mathtt{0}$s contained in $T_{\mathbf{A}}[\,i\,..\,j\,)$ must lie within $T_{\mathbf{A}}[\,2qm\,..\,2qm+m\,)$. The number of $\mathtt{0}$s within $T_{\mathbf{A}}[\,2qm\,..\,2qm+m\,)$ is $m-|A_q|$. Consequently, $m-|A_q| \ge m-k$.

We prove the query complexity lower bounds using Theorem 5.2. Consider a set X consisting of texts $T_{\mathbf{A}}$ such that $|\{q : |A_q| = k+1\}| = \lceil (p+1)/2 \rceil$ and $|\{q : |A_q| = k\}| = \lfloor (p-1)/2 \rfloor$, as well as set a Y consisting of all texts $T_{\mathbf{A}}$ such that $|\{q : |A_q| = k+1\}| = \lceil (p-1)/2 \rceil$ and $|\{q : |A_q| = k\}| = \lfloor (p+1)/2 \rfloor$. Moreover, let $R \subseteq X \times Y$ be a relation consisting of instances that differ by exactly one substitution in $T_{\mathbf{A}}$. Observe that:

- Every instance in X is in relation with exactly $\lfloor (p+1)/2 \rfloor \cdot (k+1)$ instances in Y.
- Every instance in Y is in relation with exactly $\lceil (p+1)/2 \rceil \cdot (m-k)$ instances in X.
- Instances in R differ at exactly one bit.

By Claim 6.26, any (quantum) algorithm computing $\mathrm{Occ}^E_k(P,T)$ must be able to distinguish instances in X from instances in Y. Consequently, the quantum algorithm has query complexity of at least

$$\Omega\left(\sqrt{\lfloor (p+1)/2 \rfloor \cdot (k+1) \cdot \lceil (p+1)/2 \rceil \cdot (m-k)}\right) = \Omega\left(n/m \cdot \sqrt{k(m-k)}\right).$$

Next, consider a set X' consisting of all texts $T_{\mathbf{A}}$ such that $|\{q : |A_q| = k+1\}| = p$, as well as a set Y' consisting of all texts $T_{\mathbf{A}}$ such that $|\{q : |A_q| = k+1\}| = p-1$ and $|\{q : |A_q| = k\}| = 1$. Moreover, let $R' \subset X' \times Y'$ be a relation consisting of instances that differ by exactly one substitution in $T_{\mathbf{A}}$. Observe that:

- Every instance in X is in relation with exactly $p \cdot (k+1)$ instances in Y'.
- Every instance in Y is in relation with exactly $m-k$ instances in X'.
- Instances in R differ at exactly one bit.

Since $\max \mathrm{Occ}^E_k(P,T_{\mathbf{A}}) < 2pm - m$, by Claim 6.26, any (quantum) algorithm deciding $\mathrm{Occ}^E_k(P,T) \neq \emptyset$ must be able to distinguish instances in X' from instances in Y'. Consequently, the quantum algorithm uses at least $\Omega\left(\sqrt{p \cdot (k+1) \cdot (m-k)}\right) = \Omega\left(\sqrt{n/m} \cdot \sqrt{k(m-k)}\right)$ queries. ∎

Bibliography

ABI+20 Andris Ambainis, Kaspars Balodis, Jānis Iraids, Kamil Khadiev, Vladislavs Kļevickis, Krišjānis Prūsis, Yixin Shen, Juris Smotrovs, and Jevgēnijs Vihrovs. Quantum lower and upper bounds for 2D-grid and Dyck language. In *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020*, volume 170 of *LIPIcs*, pages 8:1–8:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.8. 3

AGS19 Scott Aaronson, Daniel Grier, and Luke Schaeffer. A quantum query complexity trichotomy for regular languages. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019*, pages 942–965. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00061. 3

AJ23 Shyan Akmal and Ce Jin. Near-optimal quantum algorithms for string problems. *Algorithmica*, 85(8):2260–2317, 2023. doi:10.1007/S00453-022-01092-X. 3, 33

Amb02 Andris Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002. doi:10.1006/JCSS.2002.1826. 11, 34

Amb04 Andris Ambainis. Quantum query algorithms and lower bounds. In *Classical and New Paradigms of Computation and their Complexity Hierarchies*, volume 23 of *Trends in Logic*, pages 15–32. Springer, 2004. doi:10.1007/978-1-4020-2776-5_2. 33

BBC+95 Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter Shor, Tycho Sleator, John A. Smolin, and Harald Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52:3457–3467, 1995. doi:10.1103/PhysRevA.52.3457. 33

BBC+01 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. doi:10.1145/502090.502097. 3

BdW02 Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X. 33

BEG+21 Mahdi Boroujeni, Soheil Ehsani, Mohammad Ghodsi, MohammadTaghi Hajiaghayi, and Saeed Seddighin. Approximating edit distance in truly subquadratic time: Quantum and MapReduce. *Journal of the ACM*, 68(3):19:1–19:41, 2021. doi:10.1145/3456807. 3

BHMT02 Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. In *Quantum computation and information*, volume 305 of *Contemporary Mathematics*, pages 53–74. American Mathematical Society, 2002. doi:10.1090/conm/305/05215. 34

BI18 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM Journal on Computing*, 47(3):1087–1097, 2018. doi:10.1137/15M1053128. 1

BK23 Sudatta Bhattacharya and Michal Koucký. Streaming k-edit approximate pattern matching via string decomposition. In *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023*, volume 261 of *LIPIcs*, pages 22:1–22:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ICALP.2023.22. 3

BKS23 Gabriel Bathie, Tomasz Kociumaka, and Tatiana Starikovskaya. Small-space algorithms for the online language distance problem for palindromes and squares. In *34th International Symposium on Algorithms and Computation, ISAAC 2023*, volume 283 of *LIPIcs*, pages 10:1–10:17. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ISAAC.2023.10. 2

BKW19 Karl Bringmann, Marvin Künnemann, and Philip Wellnitz. Few matches or almost periodicity: Faster pattern matching with mismatches in compressed texts. In *30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 1126–1145. SIAM, 2019. doi:10.1137/1.9781611975482.69. 1, 2

BLR+15 Philip Bille, Gad M. Landau, Rajeev Raman, Kunihiko Sadakane, Srinivasa Rao Satti, and Oren Weimann. Random access to grammar-compressed strings and trees. *SIAM Journal on Computing*, 44(3):513–539, 2015. doi:10.1137/130936889. 3

CH02 Richard Cole and Ramesh Hariharan. Approximate string matching: A simpler faster algorithm. *SIAM Journal on Computing*, 31(6):1761–1782, 2002. doi:10.1137/S0097539700370527. 1

CKK+22 Andrew M. Childs, Robin Kothari, Matt Kovacs-Deak, Aarthi Sundaram, and Daochen Wang. Quantum divide and conquer, 2022. arXiv:2210.06419. 3

CKP19 Raphaël Clifford, Tomasz Kociumaka, and Ely Porat. The streaming k-mismatch problem. In *30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 1106–1125. SIAM, 2019. doi:10.1137/1.9781611975482.68. 2, 4

CKP+21 Panagiotis Charalampopoulos, Tomasz Kociumaka, Solon P. Pissis, Jakub Radoszewski, Wojciech Rytter, Juliusz Straszyński, Tomasz Waleń, and Wiktor Zuba. Circular pattern matching with k mismatches. *Journal of Computer and System Sciences*, 115:73–85, 2021. doi:10.1016/J.JCSS.2020.07.003. 2

CKP+22 Panagiotis Charalampopoulos, Tomasz Kociumaka, Solon P. Pissis, Jakub Radoszewski, Wojciech Rytter, Tomasz Waleń, and Wiktor Zuba. Approximate circular pattern matching. In *30th Annual European Symposium on Algorithms, ESA 2022*, volume 244 of *LIPIcs*, pages 35:1–35:19. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ESA.2022.35. 2

CKW20 Panagiotis Charalampopoulos, Tomasz Kociumaka, and Philip Wellnitz. Faster approximate pattern matching: A unified approach. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, pages 978–989. IEEE, 2020. doi:10.1109/FOCS46700.2020.00095. 1, 2, 3, 11, 12, 14, 41, 42, 54

CKW22 Panagiotis Charalampopoulos, Tomasz Kociumaka, and Philip Wellnitz. Faster pattern matching under edit distance: A reduction to dynamic puzzle matching and the seaweed monoid of permutation matrices. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, pages 698–707. IEEE, 2022. doi:10.1109/FOCS54457.2022.00072. 1, 2, 3, 13, 14, 16, 17

CKW23 Alejandro Cassis, Tomasz Kociumaka, and Philip Wellnitz. Optimal algorithms for bounded weighted edit distance. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023*, pages 2177–2187. IEEE, 2023. doi:10.1109/FOCS57990.2023.00135. 8, 10, 17, 18, 35

CPR+24 Panagiotis Charalampopoulos, Solon P. Pissis, Jakub Radoszewski, Wojciech Rytter, Tomasz Waleń, and Wiktor Zuba. Approximate circular pattern matching under edit distance. In *41st International Symposium on Theoretical Aspects of Computer Science, STACS 2024*, LIPIcs. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024. arXiv:2402.14550. 2

FW65 Nathan J. Fine and Herbert S. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, 16(1):109–114, 1965. doi:10.1090/S0002-9939-1965-0174934-9. 4, 5, 15

GJKT24 Daniel Gibney, Ce Jin, Tomasz Kociumaka, and Sharma V. Thankachan. Near-optimal quantum algorithms for bounded edit distance and Lempel-Ziv factorization. In *35th ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 3302–3332. SIAM, 2024. doi:10.1137/1.9781611977912.11. 3, 8, 10, 11, 12, 14, 35

GKKS22 Elazar Goldenberg, Tomasz Kociumaka, Robert Krauthgamer, and Barna Saha. Gap edit distance via non-adaptive queries: Simple and optimal. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022*, pages 674–685. IEEE, 2022. doi:10.1109/FOCS54457.2022.00070. 11, 14, 37, 38, 39, 40, 41

Gro96 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th Annual ACM Symposium on the Theory of Computing, STOC 1996*, pages 212–219. ACM, 1996. doi:10.1145/237814.237866. 34

GS13 Paweł Gawrychowski and Damian Straszak. Beating $O(nm)$ in approximate LZW-compressed pattern matching. In *24th International Symposium on Algorithms and Computation, ISAAC 2013*, volume 8283 of *LNCS*, pages 78–88. Springer, 2013. doi:10.1007/978-3-642-45030-3_8. 3

GS23 François Le Gall and Saeed Seddighin. Quantum meets fine-grained complexity: Sublinear time quantum algorithms for string problems. *Algorithmica*, 85(5):1251–1286, 2023. doi:10.1007/S00453-022-01066-Z. 3, 33

HV03 Ramesh Hariharan and V. Vinay. String matching in $\tilde{O}(\sqrt{n}+\sqrt{m})$ quantum time. *Journal of Discrete Algorithms*, 1(1):103–110, 2003. doi:10.1016/S1570-8667(03)00010-8. 3, 34

I17 Tomohiro I. Longest Common Extensions with Recompression. In Juha Kärkkäinen, Jakub Radoszewski, and Wojciech Rytter, editors, *28th Annual Symposium on Combinatorial Pattern Matching (CPM 2017)*, volume 78 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:15, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops-dev.dagstuhl.de/entities/document/10.4230/LIPIcs.CPM.2017.18, doi:10.4230/LIPIcs.CPM.2017.18. 35

JN23 Ce Jin and Jakob Nogler. Quantum speed-ups for string synchronizing sets, longest common substring, and k-mismatch matching. In *34th ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 5090–5121. SIAM, 2023. doi:10.1137/1.9781611977554.CH186. 3, 12

KPS21 Tomasz Kociumaka, Ely Porat, and Tatiana Starikovskaya. Small-space and streaming pattern matching with k edits. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 885–896. IEEE, 2021. doi:10.1109/FOCS52979.2021.00090. 3

KRRW15 Tomasz Kociumaka, Jakub Radoszewski, Wojciech Rytter, and Tomasz Waleń. Internal pattern matching queries in a text and applications. In *26th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 532–551. SIAM, 2015. doi:10.1137/1.9781611973730.36. 34

KS24 Michal Koucký and Michael Saks. Almost linear size edit distance sketch. In *56th Annual ACM Symposium on the Theory of Computing; STOC 2024*, 2024. 2

Lev65 Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965. URL: http://mi.mathnet.ru/eng/dan31411. 1, 16

LV88 Gad M. Landau and Uzi Vishkin. Fast string matching with k differences. *Journal of Computer and System Sciences*, 37(1):63–78, 1988. doi:10.1016/0022-0000(88)90045-1. 1, 12, 35

LV89 Gad M. Landau and Uzi Vishkin. Fast parallel and serial approximate string matching. *Journal of Algorithms*, 10(2):157–169, 1989. doi:10.1016/0196-6774(89)90010-2. 1

Sel80 Peter H. Sellers. The theory and computation of evolutionary distances: Pattern recognition. *Journal of Algorithms*, 1(4):359–373, 1980. doi:10.1016/0196-6774(80)90016-4. 1

Sta17 Tatiana Starikovskaya. Communication and streaming complexity of approximate pattern matching. In *28th Annual Symposium on Combinatorial Pattern Matching, CPM 2017*, volume 78 of *LIPIcs*, pages 13:1–13:11. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.CPM.2017.13. 3

Ste24 Teresa Anna Steiner. Differentially private approximate pattern matching. In *15th Innovations in Theoretical Computer Science Conference, ITCS 2024*, volume 287 of *LIPIcs*, pages 94:1–94:18. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024. doi:doi.org/10.4230/LIPIcs.ITCS.2024.94. 2

Tis14 Alexander Tiskin. Threshold approximate matching in grammar-compressed strings. In Jan Holub and Jan Žďárek, editors, *Prague Stringology Conference, PSC 2014*, pages 124–138, 2014. URL: http://www.stringology.org/event/2014/p12.html. 3

WY24 Qisheng Wang and Mingsheng Ying. Quantum algorithm for lexicographically minimal string rotation. *Theory of Computing Systems*, 68(1):29–74, 2024. doi:10.1007/S00224-023-10146-8. 3

ZL77 Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977. doi:10.1109/TIT.1977.1055714. 10, 14, 17