Accepted Manuscript

A linear algorithm for string reconstruction in the reverse complement equivalence model

Ferdinando Cicalese, Péter L. Erdős, Zsuzsanna Lipták

Please cite this article in press as: F. Cicalese et al., A linear algorithm for string reconstruction in the reverse complement equivalence model, *Journal of Discrete Algorithms* (2011), doi:10.1016/j.jda.2011.12.003

A Linear Algorithm for String Reconstruction in the Reverse Complement Equivalence Model [*]

FERDINANDO CICALESE

Dipartimento di Informatica ed Applicazioni, University of Salerno, Italy

email: cicalese@dia.unisa.it

PÉTER L. ERDŐS

A. Rényi Institute of Mathematics, Hungarian Academy of Sciences,

Budapest, P.O. Box 127, H-1364 Hungary

email: elp@renyi.hu

ZSUZSANNA LIPTÁK[†]

AG Genominformatik, Technische Fakultät, Bielefeld University, Germany

email: zsuzsa@cebitec.uni-bielefeld.de

December 2, 2011

Abstract

In the *reverse complement* equivalence model, it is not possible to distinguish a string from its reverse complement. We show that one can still reconstruct a string of length n, up to reverse complement, using a linear number of subsequence queries of bounded length. We first give the proof for strings over a binary alphabet, and then extend it to arbitrary finite alphabets. A simple information theoretic lower bound proves the number of queries to be asymptotically tight. Furthermore, our result is optimal w.r.t. the bound on the query length given in [Erdős *et al.*, Ann. of Comb. 2006].

Key words: string reconstruction, reverse complement, string algorithms, subsequences, subwords, combinatorics on words

1 Introduction

Reconstructing a string over a finite alphabet Σ from information about its subsequences is a classic string problem, with applications ranging from coding theory to bioinformatics. Due to differences in terminology in the literature, we want to give a precise definition right here: Given two strings $\mathbf{s} = s_1 \ldots s_n$ and $\mathbf{t} = t_1 \ldots t_m$ over Σ, we say that $\mathbf{t}$ is a *subsequence* (often called *subword*) of $\mathbf{s}$ if there exist $1 \leq i_1 < i_2 < \ldots < i_m \leq n$ such that $\mathbf{t} = s_{i_1} s_{i_2} \ldots s_{i_m}$. It was shown by Simon in 1975 [14] that two strings of length n are equal if their subsequences up to length $\lfloor n/2 \rfloor + 1$ coincide. The proof, as given in Chapter 6 of the classic Lothaire book [13] can be easily adapted

---

[*]An extended abstract of the results for the binary case was published in [3].

[†]Current affiliation: Dipartimento di Informatica ed Applicazioni, University of Salerno, Italy.

to yield an algorithm which reconstructs the string $\mathbf{s}$ of length n, using $O(|\Sigma|n)$ queries of the type "Is $\mathbf{u}$ a subsequence of $\mathbf{s}$?" Here, $\mathbf{u}$ is a string of length at most $\lfloor n/2 \rfloor + 1$.

In this paper, we consider this problem in the RC-equivalence model, which is motivated by reverse complementation of DNA. Our alphabet consists of *pairs* of characters $(a, \bar{a})$, called *complement pairs*, and for every string $\mathbf{s} = s_1 \ldots s_n$ over Σ, we define its *reverse complement* as $\tilde{\mathbf{s}} = \bar{s}_n \ldots \bar{s}_1$. Two strings $\mathbf{s}, \mathbf{t}$ are RC-equivalent if $\mathbf{s} = \mathbf{t}$ or $\mathbf{s} = \tilde{\mathbf{t}}$. A string $\mathbf{u}$ is an RC-subsequence of $\mathbf{s}$ if $\mathbf{u}$ or $\tilde{\mathbf{u}}$ is a subsequence of $\mathbf{s}$. For example, consider the string $\mathbf{s} = \bar{a}a\bar{a}$ over the alphabet $\{a, \bar{a}\}$. Then aa is not a subsequence of $\mathbf{s}$, but it is an RC-subsequence, because $\bar{a}\bar{a}$ is a subsequence of $\mathbf{s}$. Erdős *et al.* showed in [6] that two strings $\mathbf{s}$ and $\mathbf{t}$ of length n are RC-equivalent if and only if all their RC-subsequences up to length $\lceil \frac{2}{3}(n+1) \rceil$ coincide. However, no reconstruction algorithm was given.

Here, we present such an algorithm. First, we give an information theoretic lower bound on the number of queries necessary for exact reconstruction, which is $\Omega(n \log |\Sigma|)$. Then we describe a simple algorithm for arbitrary alphabets, which uses an asymptotically optimal number of queries $O(n \log |\Sigma|)$. This algorithm was adapted from a paper by Skiena and Sundaram [15], where the length of the queries is not bounded. The major part of the paper, however, is devoted to the bounded query case: For the case of a binary alphabet, i.e., where the alphabet consists of two complementary characters, our algorithm reconstructs a string $\mathbf{s}$ of length n, using $O(n)$ queries of the type "Is $\mathbf{u}$ an RC-subsequence of $\mathbf{s}$?" where $\mathbf{u}$ is a string of length at most $\lceil \frac{2}{3}(n+1) \rceil$. We note that our algorithm is optimal w.r.t. the length of the queries, and asymptotically optimal w.r.t. the information theoretic lower bound on the number of queries necessary for exact reconstruction. Finally, we provide an algorithm for the general case of arbitrary alphabets and bounded queries, which uses $O(|\Sigma|n)$ queries. In the whole paper we assume that $|\Sigma| = O(n)$.

Related work. Most literature deals with the classical, i.e. non-RC, model. In addition to the papers mentioned above, we want to point to the following.

When the multiset of subsequences is known, then much shorter subsequences suffice to uniquely identify a string: A string of length n can be uniquely identified by the multiset of its subsequences of length $\lfloor \frac{16}{7}\sqrt{n} \rfloor + 5$, as shown by Krasikov and Roditty [8]. Dudík and Schulman [5] give asymptotic lower and upper bounds, in terms of k, on the length of strings which can be uniquely determined by the multiset of their subsequences of length k.

Levenshtein [9] investigates the maximal number of common subsequences of length k that two distinct strings of length n can have. Here, subsequences are regarded as erroneous versions of the original string. The aim is to find how many times a transmission needs to be repeated, over a channel which allows a constant number of deletions, to make unique recovery of the original message possible.

The case where only substrings are considered has also received much attention. Substrings, often called factors, are contiguous subsequences: $\mathbf{t}$ is a substring of $\mathbf{s}$ if there are $1 \leq i \leq j \leq n$ such that $\mathbf{t} = s_i \ldots s_j$. The length of substrings of a string $\mathbf{s}$ of length n which are necessary for uniquely determining $\mathbf{s}$ depends on a parameter of $\mathbf{s}$, namely on the maximal length of a repeated substring, as shown by de Luca and Carpi in a series of papers [2,4]. An algorithm for reconstruction was given by Fici *et al.* in [7], while the uniqueness bound for multisets of substrings was recently shown to be $\lfloor \frac{n}{2} \rfloor + 1$ by Piña and Uzcágetui [11].

The problem of reconstructing a string of length n using substring queries has also been extensively studied in the setting of Sequencing by Hybridization (SBH), first suggested by Pevzner [10]. Here, a large number of strings of a certain length are queried in parallel, using a DNA chip, and

2

the resulting answers are then used to reconstruct all or parts of the DNA string. A number of different SBH techniques have been proposed, leading to different string combinatorial questions. (See, for example, [12, 16] for some more recent results.)

Reverse complementation over paired alphabets was referred to as *video reversal*, denoted $\bar{\mathbf{s}}^R$, by Bercoff in [1]. However, the object of research there are the generation of infinite sequences with certain properties, and reverse complementation is only used as a function, and not as an equivalence relation.

Overview of paper. In Section 2, we give the necessary definitions and prove an information theoretic lower bound on the number of queries. In Section 3, we describe a simple algorithm for arbitrary alphabets for unbounded query length. We then give a reconstruction algorithm for binary alphabets (Section 4). Sections 5 and 6 present the analogous result for general alphabets, where we first show how to reduce the problem to interleaving two subsequences over disjoint sets of complementary character pairs (Section 5), and then show how to execute this interleaving (Section 6). We end with a brief outlook in Section 7.

2 Preliminaries

By a *paired alphabet* we understand a finite set Σ of size 2δ for some integer $\delta \geq 1$, together with a non-identity involution operation $^{-}: \Sigma \mapsto \Sigma$, which we call *complement*. By convention, we write Σ as $\Sigma = \{a_1, \overline{a_1}, \ldots, a_\delta, \overline{a_\delta}\}$. Notice that by definition, $\overline{\overline{a_i}} = a_i$, for each i.

Let $\mathbf{s} = s_1 \ldots s_n$ be a string over Σ, i.e., $\mathbf{s} \in \Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i$, where, following standard notation, $\Sigma^i = \{x_1 \ldots x_i \mid x_k \in \Sigma, \text{ for each } k = 1, \ldots, i\}$, and Σ^0 is the singleton containing only the empty string ϵ. The *reverse complement* of $\mathbf{s}$ is defined as $\tilde{\mathbf{s}} = \overline{s_n}\,\overline{s_{n-1}} \ldots \overline{s_1}$. Two strings $\mathbf{s}, \mathbf{t}$ are RC-*equivalent*, denoted $\mathbf{s} \equiv_{RC} \mathbf{t}$, if either $\mathbf{s} = \mathbf{t}$ or $\mathbf{s} = \tilde{\mathbf{t}}$. A string $\mathbf{s}$ is called an *RC-palindrome* if $\mathbf{s} = \tilde{\mathbf{s}}$.

For a string $\mathbf{s} = s_1 \ldots s_n$ over the alphabet Σ, we denote by $|\mathbf{s}| = n$ the length of $\mathbf{s}$, and by $|\mathbf{s}|_a = |\{i \mid s_i = a\}|$ the number of a's in $\mathbf{s}$, for $a \in \Sigma$. We write a^k for the string $aa \ldots a$ of length k. We denote by $\text{alph}(\mathbf{s})$ the subset of Σ of characters a for which $|\mathbf{s}|_a > 0$. Two strings $\mathbf{s}$ and $\mathbf{t}$ over Σ are *RC-character-disjoint*, if for all $a \in \Sigma$ s.t. $|\mathbf{s}|_a > 0$, neither a nor $\bar{a}$ occurs in $\mathbf{t}$, and vice versa. Note that this is stronger than the condition $\text{alph}(\mathbf{s}) \cap \text{alph}(\mathbf{t}) = \emptyset$.

Given two strings $\mathbf{s} = s_1 \ldots s_n$ and $\mathbf{t} = t_1 \ldots t_m$ over Σ, $\mathbf{t}$ is a *subsequence* of $\mathbf{s}$, denoted by $\mathbf{t} \prec \mathbf{s}$, if there exist $1 \leq i_1 < i_2 < \ldots < i_m \leq n$ such that $\mathbf{t} = s_{i_1} s_{i_2} \ldots s_{i_m}$. If $\mathbf{t}$ is a subsequence of $\mathbf{s}$, then $\mathbf{s}$ is a *supersequence* of $\mathbf{t}$. Further, we define $\mathbf{t}$ to be an *RC-subsequence*, denoted $\mathbf{t} \prec_{RC} \mathbf{s}$ if and only if $\mathbf{t} \prec \mathbf{s}$ or $\mathbf{t} \prec \tilde{\mathbf{s}}$, i.e., if $\mathbf{t}$ is a subsequence of $\mathbf{s}$ or of its reverse complement. Note that the condition $\mathbf{t} \prec \tilde{\mathbf{s}}$ is equivalent to $\tilde{\mathbf{t}} \prec \mathbf{s}$. For a string $\mathbf{s}$ over Σ, and an index $i \in \{1, \ldots, \delta\}$, we denote by $\mathbf{s}_{|i}$ the longest subsequence of $\mathbf{s}$ containing only characters a_i and $\overline{a_i}$. We call $\mathbf{s}_{|i}$ the i'th *projection* of $\mathbf{s}$.

Example 2.1. *Our motivating example is the alphabet of the 4 nucleotides (DNA)* $\Sigma = \{A, T, C, G\}$, *i.e.* (A, T) *and* (C, G) *are complement pairs. Let* $\mathbf{s} = \text{ACCGATTAC}$*. Then* $\tilde{\mathbf{s}} = \text{GTAATCGGT}$, $\text{GTTT} \not\prec \mathbf{s}$ *but* $\text{GTTT} \prec_{RC} \mathbf{s}$*. The two projections of* $\mathbf{s}$ *are* AATTA *and* CCGC.

Let $\mathbf{s} = s_1 \ldots s_n \in \Sigma^*$. A *run* in $\mathbf{s}$ is a maximal interval (i, j), $1 \leq i \leq j \leq n$ consisting of the same character, i.e. for some $a \in \Sigma$, we have $s_k = a$ for all $i \leq k \leq j$. Any string $\mathbf{s}$ over Σ can be written uniquely in its *runlength encoded* form $\mathbf{s} = a_1^{x_1} a_2^{x_2} \ldots a_r^{x_r}$, where $a_i^{x_i}$, with $x_i > 0$, are the runs of $\mathbf{s}$ and $r \leq n$.

We are now ready to state the problem we investigate in the present paper.

The RC-String Identification Problem. Fix a paired alphabet Σ, together with a string $\mathbf{s}$ over Σ, and let $n = |\mathbf{s}|$. For any positive integer $T \leq n$, a T-bounded RC-subsequence query is any $\mathbf{t} \in \bigcup_{i=1}^{T} \Sigma^i$. The answer to such a query is *yes* (or *positive*) if and only if $\mathbf{t} \prec_{\mathrm{RC}} \mathbf{s}$. Otherwise the answer is *no* (or *negative*). Given the alphabet Σ, the size of the string n, and the threshold on the length of the queries $T \leq n$, the RC-String Identification Problem asks for the minimum number of T-bounded RC-subsequence queries which are sufficient to determine the pair $(\mathbf{s}, \tilde{\mathbf{s}})$, for any unknown string $\mathbf{s}$ of size n.

We first present an information theoretic lower bound that holds even in the case of unbounded queries, i.e. if $T = n$. Here and elsewhere, we denote by $\log x$ the logarithm of x to base 2.

Proposition 2.2 (Lower Bound). *Given a string $\mathbf{s}$ of size n over a paired alphabet Σ. Any deterministic algorithm that identifies $\mathbf{s}$ (up to reverse complement) by asking RC-subsequence queries needs at least $n \log |\Sigma| - 1$ queries.*

Proof. Upon identifying a string with its reverse complement, there are at least $|\Sigma^n|/2$ possible *distinct* strings of length n. Any query $\mathbf{t}$ splits the space of candidate solutions into two parts. Therefore, at least $\log |\Sigma^n|/2 = n \log |\Sigma| - 1$ questions are necessary to identify $\mathbf{s}$. □

3 Unbounded query size

If $T = n$, i.e., no constraint is set on the length of a query, then it is easy to reconstruct a string in linear time. We adapt a simple algorithm from [15], developed for the classic case, i.e., without RC-equivalence.

Theorem 3.1. *There exists an algorithm which reconstructs, up to RC-equivalence, a string of length n using $\Theta(n \log |\Sigma|)$ RC-subsequence queries of unbounded length.*

To prove Theorem 3.1, we will use the following easy lemma, which extends Lemma 14 in [15] to our problem.

Lemma 3.2. *Let $\mathbf{s}$ be an unknown string over the paired alphabet $\Sigma = \{a_1, \overline{a_1}, \ldots, a_\delta, \overline{a_\delta}\}$. Let $\mathbf{u}, \mathbf{v} \prec_{\mathrm{RC}} \mathbf{s}$ be two known RC-character-disjoint strings over Σ, i.e., no character of $\mathbf{u}$ or its pair occurs in $\mathbf{v}$, and vice versa. Then it is possible to construct a string $\mathbf{w}$ such that $\mathbf{u}, \mathbf{v} \prec_{\mathrm{RC}} \mathbf{w}$ and $\mathbf{w} \prec_{\mathrm{RC}} \mathbf{s}$ using at most $2(|\mathbf{u}| + |\mathbf{v}| + 1)$ queries.*

Proof. By definition, at least one of the four cases must hold: $\mathbf{u}, \mathbf{v} \prec \mathbf{s}$, $\tilde{\mathbf{u}}, \tilde{\mathbf{v}} \prec \mathbf{s}$, $\mathbf{u}, \tilde{\mathbf{v}} \prec \mathbf{s}$, or $\tilde{\mathbf{u}}, \mathbf{v} \prec \mathbf{s}$. Let us first assume that $\mathbf{u}, \mathbf{v} \prec \mathbf{s}$. Let $\mathbf{u} = u_1 u_2 \ldots u_k$ and $\mathbf{v} = v_1 v_2 \ldots v_m$, w.l.o.g. $k \leq m$. Finding $\mathbf{w}$ consists of interleaving $\mathbf{u}$ and $\mathbf{v}$ in such a way that the resulting string is a subsequence of $\mathbf{s}$. In other words, we must find indices $0 \leq i_1 \leq i_2 \leq \ldots \leq i_{k+1} \leq k+1$ s.t. $\mathbf{w} = v_1 \ldots v_{i_1} u_1 v_{i_1+1} \ldots v_{i_2} u_2 \ldots u_k v_{i_k+1} \ldots v_{i_{k+1}}$ is a subsequence of $\mathbf{s}$. (For technical reasons, we set v_i to be the empty string for $i < 1$ and $i > m$.) This can be done by asking queries $v_1 \mathbf{u}$, $v_1 v_2 \mathbf{u}$, $v_1 v_2 v_3 \mathbf{u}$ etc. until the first *no* to determine i_1; then queries $v_1 \ldots v_{i_1} u_1 v_{i_1+1} u_2 u_3 \ldots u_k$, $v_1 \ldots v_{i_1} u_1 v_{i_1+1} v_{i_1+2} u_2 u_3 \ldots u_k$, etc. to determine i_2. Proceeding in this way, we will find all i_j's,

4

using $i_j - i_{j-1} + 1$ many queries in the j'th step, so altogether $i_1 + 1 + \sum_{j=2}^{k+1}(i_j - i_{j-1} + 1) = m + k + 1 = |\mathbf{v}| + |\mathbf{u}| + 1$ many queries.

Now note that we have assumed that both $\mathbf{u}$ and $\mathbf{v}$ are subsequences of $\mathbf{s}$. In this case, the string $\mathbf{w}$ which is constructed is also a subsequence of $\mathbf{s}$. If, on the other hand, both $\mathbf{u}$ and $\mathbf{v}$ are subsequences of $\tilde{\mathbf{s}}$, then the $\mathbf{w}$ thus constructed will also be a subsequence of $\tilde{\mathbf{s}}$, thus satisfying $\mathbf{w} \prec_{\mathrm{RC}} \mathbf{s}$. Finally, if neither of these cases holds, then the algorithm sketched above will abort without producing a desired $\mathbf{w}$. Then we repeat it with $\mathbf{u}$ and $\tilde{\mathbf{v}}$, which will produce a string $\mathbf{w}$ that is either a subsequence of $\mathbf{s}$ or of $\tilde{\mathbf{s}}$, in either case $\mathbf{w} \prec_{\mathrm{RC}} \mathbf{s}$, as claimed. The total number of queries is thus at most $2(|\mathbf{u}| + |\mathbf{v}| + 1)$. □

Proof of Theorem 3.1. We first give the algorithm for the case where $|\Sigma| = 2$. Let $\Sigma = \{a, b\}$ where $b = \bar{a}$. Let $M = \max\{\chi \geq 0 \mid a^{\chi} \prec_{\mathrm{RC}} \mathbf{s}\}$. Clearly, we have $M = \max\{|\mathbf{s}|_a, |\mathbf{s}|_b\}$. Notice that we can determine M by asking query $\mathbf{t}_{\chi}^{(0)} = a^{\chi}$, for $\chi = 1, 2, 3, \ldots$, until we get the first negative answer, implying that $M = \chi - 1$. In particular, we need exactly $M + 1$ such queries.

Define $y_1 = \max\{\chi = 0, 1, 2, \ldots, \mid b^{\chi} a^M \prec_{\mathrm{RC}} \mathbf{s}\}$, and for $i = 2, \ldots, M + 1$,

$$y_i = \max\{\chi = 0, 1, 2, \ldots \mid b^{y_1} a b^{y_2} a \ldots b^{y_{i-1}} a b^{\chi} a^{M-i+1} \prec_{\mathrm{RC}} \mathbf{s}\}.$$

In perfect analogy with what we did above, we can determine y_i, by asking the query $\mathbf{t}_{\chi}^{(i)} = b^{y_1} a b^{y_2} a \ldots b^{y_{i-1}} a b^{\chi} a^{M-i+1}$, for each $\chi = 1, 2, \ldots$, until we receive a negative answer, implying that $y_i = \chi - 1$. Therefore, we need exactly $y_i + 1$ such queries to determine y_i.

Let us write $\mathbf{t}$ for $\mathbf{t}_{y_{M+1}}^{(M+1)}$. For constructing $\mathbf{t}$ we need exactly $M + 1 + \sum_{i=1}^{M+1}(y_i + 1) = 2\max_{c=a,b} |\mathbf{s}|_c + 2 + \min_{c=a,b} |\mathbf{s}|_c$ many queries. We claim that $\mathbf{t}$ is a maximum size subsequence of $\mathbf{s}$ or $\tilde{\mathbf{s}}$, i.e., there exists no $\mathbf{t}' \neq \mathbf{t}$ such that $\mathbf{t} \prec \mathbf{t}' \prec_{\mathrm{RC}} \mathbf{s}$. This implies that $\mathbf{t} \in \{\mathbf{s}, \tilde{\mathbf{s}}\}$, i.e., the above procedure determines $\mathbf{s}$ up to reverse complement using $O(n)$ queries.

We prove the claim by contradiction. Assume that $\mathbf{t} \notin \{\mathbf{s}, \tilde{\mathbf{s}}\}$. Since by construction $\mathbf{t} \prec_{\mathrm{RC}} \mathbf{s}$, it follows that there exists a sequence $\mathbf{t}' \neq \mathbf{t}$ such that $\mathbf{t} \prec \mathbf{t}' \prec_{\mathrm{RC}} \mathbf{s}$, and $|\mathbf{t}'| = |\mathbf{t}| + 1$.

By the definition of M, it follows that $|\mathbf{t}|_a = |\mathbf{t}'|_a$, for otherwise $a^{M+1} \prec \mathbf{t}' \prec_{\mathrm{RC}} \mathbf{s}$, contradicting the fact that M is maximal. Therefore it must be $|\mathbf{t}'|_b = |\mathbf{t}|_b + 1$. In analogy with what we have done for $\mathbf{t}$, let us write $\mathbf{t}' = b^{y_1'} a b^{y_2'} a \ldots b^{y_M'} a b^{y_{M+1}'}$, where $y_j' \geq 0$, for each $j = 1, \ldots, M + 1$. By the definition of $\mathbf{t}'$, there exists exactly one j such that $y_j' = y_j + 1$. It follows that $b^{y_1} a b^{y_2} a \ldots b^{y_{j-1}} a b^{y_j + 1} a^{M-j} \prec \mathbf{t}' \prec_{\mathrm{RC}} \mathbf{s}$, which contradicts the definition of y_j.

Now let $|\Sigma| = 2\delta$, with $\delta > 1$. Recall that $\mathbf{s}_{|i}$, the *i'th projection of* $\mathbf{s}$, is the longest subsequence of $\mathbf{s}$ only containing characters from $\{a_i, \overline{a_i}\}$.

It is not hard to see that we can use the above procedure to identify a string $\mathbf{u}_i \in \{\mathbf{s}_{|i}, \tilde{\mathbf{s}}_{|i}\}$. In particular, it follows that the number of queries required for determining the i'th projection is thus at most $\frac{3}{2}n_i + 2$, where $n_i = |\mathbf{s}_{|i}|$.

Once we have identified the i'th projection of $\mathbf{s}$ (up to reverse complement), for each $i = 1, \ldots, \delta$, we can use Lemma 3.2 for iteratively interleaving these projections and constructing $\mathbf{s}$: We first interleave $\mathbf{s}_{|1}$ with $\mathbf{s}_{|2}$, which yields $\mathbf{s}_{|1,2}$. Then we interleave $\mathbf{s}_{|1,2}$ with $\mathbf{s}_{|3,4}$ and so on. By Lemma 3.2, each of these can be done using at most twice the total length of the two strings plus 2. So the total number of queries for the interleaving phase is at most $2n \log \delta + 2(\delta - 1)$: the lengths of the subsequences at each level add up to n, there are $\log \delta$ many levels, and for each of the $\delta - 1$ many inner nodes where the interleavings happen, we may have two additional queries. Thus

5

the total number of queries for the complete algorithm is $\sum_{i=1}^{\delta}(\frac{3}{2}n_i + 2) + 2n \log \delta + 2(\delta - 1) = \frac{3}{2}n + 2n \log \delta + 4\delta - 2 = O(n \log |\Sigma|)$, using the fact that $|\Sigma| = 2\delta$ and our assumption that $|\Sigma| = O(n)$. $\qquad\square$

4   Bounded query size over a binary alphabet

We now turn to subsequence queries whose length is bounded by a threshold T. In this section, the alphabet is binary, i.e., $\Sigma = \{a, b\}$, with $b = \bar{a}$. The following result shows that string identification by T-bounded subsequence queries cannot be attained in general if the threshold T on the size of the subsequence queries is not larger than $\left\lceil \frac{2}{3}n \right\rceil$.

Observation 4.1 (Erdős et al., 2006 [6]). *For any $n \geq 4$ there exist two distinct strings of size n with exactly the same set of subsequences of length up to $\left\lceil \frac{2n}{3} \right\rceil - 1$.*

Proof. Let us write n as $n = 3k + r$, with $r \in \{0, 1, 2\}$. Let $\mathbf{s}_1 = a^{2k+r}b^k$ and $\mathbf{s}_2 = a^{2k+r-1}b^{k+1}$. It is not hard to see that we have $a^{2k+r} \prec_{\mathrm{RC}} \mathbf{s}_1$ but $a^{2k+r} \nprec_{\mathrm{RC}} \mathbf{s}_2$ whilst for any string $\mathbf{t}$ such that $|\mathbf{t}| \leq 2k + r - 1$, we have that $\mathbf{t} \prec_{\mathrm{RC}} \mathbf{s}_1$ if and only if $\mathbf{t} \prec_{\mathrm{RC}} \mathbf{s}_2$. $\qquad\square$

This implies that if we are looking for algorithms which are able to reconstruct *any* binary string of size n, we must allow queries of size $\geq \lceil 2n/3 \rceil$.

Let $\mathbf{s} \in \Sigma^*$. Denote by ρ_a the number of a-runs (runs of a's) in $\mathbf{s}$ and by ρ_b the number of b-runs. Note that $|\rho_a - \rho_b| \leq 1$. Then $\mathbf{s}$ can be written as:

$$\mathbf{s} = a^{x_1}b^{y_1}a^{x_2}b^{y_2}\ldots a^{x_{\rho-1}}b^{y_{\rho-1}}a^{x_\rho}b^{y_\rho}, \tag{1}$$

with x_1 and y_ρ possibly 0, all other $x_i, y_i > 0$. Note that ρ either equals $\max(\rho_a, \rho_b)$ or $\max(\rho_a, \rho_b) + 1$, with the latter being the case if and only if $\mathbf{s}$ begins with a b and ends with an a.) We denote by $A = |\mathbf{s}|_a$ the number of a's and by $B = |\mathbf{s}|_b$ the number of b's in $\mathbf{s}$. In the following we assume that $A \geq B$. This is without loss of generality since otherwise, we exchange the roles of $\mathbf{s}$ and $\tilde{\mathbf{s}}$.

In this section we will prove the following theorem:

Theorem 4.2. *There is an algorithm which reconstructs, up to RC-equivalence, a binary string $\mathbf{s}$ of length n using $O(n)$ many RC-subsequence queries of length at most $\left\lceil \frac{2}{3}(n+1) \right\rceil$.*

Notice that this is at most 1 off the lower bound of Observation 4.1: It is tight when $n \equiv 2 \mod 3$, and it is almost tight otherwise (in this case there is a gap of 1).

The proof of the theorem is by examining four cases separately. Recall that $A = |\mathbf{s}|_a, B = |\mathbf{s}|_b$, and $T = \left\lceil \frac{2}{3}(n+1) \right\rceil$. The four cases are: *1.* $A \geq T$, *2.* $T > A > B$, *3.* $A = B$ and $s_1 = s_n$, and *4.* $A = B$ and $s_1 \neq s_n$. The following simple lemma will be used to distinguish these cases.

Lemma 4.3. *Let $\mathbf{s}$ be a string of length at least 8 over $\{a, b\}$, $T = \left\lceil \frac{2}{3}(n+1) \right\rceil$, and $A = |\mathbf{s}|_a \geq |\mathbf{s}|_b$. Then,*

1. *using $O(\log n)$ RC-subsequence queries of length at most T, it is possible to determine the exact value of $A = |\mathbf{s}|_a$ if $A < T$, or to establish the fact that $A \geq T$.*

2. *Moreover, if $A < T$, then it can be determined whether $\mathbf{s}$ starts and ends with the same character; furthermore, unless $A = \frac{n}{2}$ and $s_1 = s_n$, we can determine s_1 and s_n. Altogether we require at most 3 additional RC-subsequence queries of length at most T.*

6

Proof. 1. Binary search for A, using queries of the form a^χ, for $\chi \in [\frac{n}{2}, T]$, will either return the exact value of A (if $A < T$), or will exit with the maximum size query $a^T \prec_{RC} \mathbf{s}$, thus showing that $A \geq T$.

2. Notice that if $A = B = \frac{n}{2}$, then the query $\mathbf{t} = ab^{\frac{n}{2}}a$ will return *yes* if and only if $s_1 = s_n$. If $s_1 = s_n$, then, due to the complete symmetry, we cannot determine the exact nature of s_1 and s_n. Otherwise, either $T > A = B$ and $s_1 \neq s_n$, or $T > A > B$. In either case, the query ba^A has length at most T and will answer positively if and only if $s_1 = b$. Likewise, the query $a^A b$ will answer positively if and only if $s_n = b$. □

Example 4.4. *Let* $\mathbf{s}_1 = aababbba$. *Then* $\tilde{\mathbf{s}}_1 = baaababb$. *The query* ab^4a *will return* yes *and we can only determine that the first and last characters are equal, but not what they are. Instead, for* $\mathbf{s}_2 = aababbab$, *we have* $\tilde{\mathbf{s}}_2 = abaababb$, *the query* ab^4a *will return* no, *and since the query* ab^4 *is positively answered, we know that the first character is a (and thus the last character is b).*

We will now introduce a fundamental notion used in our algorithms.

Definition 4.5 (Fixes the direction). *Given a string* $\mathbf{s}$ *and a subsequence* $\mathbf{t}$ *of* $\mathbf{s}$, *we say that* $\mathbf{t}$ *fixes the direction of* $\mathbf{s}$ *if* $\mathbf{t} \not\prec \tilde{\mathbf{s}}$.

If $\mathbf{t}$ fixes the direction of $\mathbf{s}$ then for any $\mathbf{t}' \prec \mathbf{s}$, such that $\mathbf{t} \prec \mathbf{t}'$ we also have that $\mathbf{t}'$ fixes the direction of $\mathbf{s}$. In general, we shall try to identify $\mathbf{s}$ by first finding some sequence $\mathbf{t}$ which fixes the direction of $\mathbf{s}$ or $\tilde{\mathbf{s}}$ and then extending this $\mathbf{t}$. The importance of "direction-fixing" is that once we have found $\mathbf{t}$ which fixes the direction of $\mathbf{s}$, by asking queries about super-sequences of $\mathbf{t}$ we are sure that the answers to our queries are only about $\mathbf{s}$ and not its reverse complement.

The following two statements formalize two simple facts which will be used repeatedly in the following, thus, for the sake of completeness, we formally state and prove them here. Let $\mathbf{s}$ be fixed for the rest of this section.

Lemma 4.6. *Let* $\mathbf{t} = t_1 \ldots t_m$ *be a sequence which fixes the direction of* $\mathbf{s}$. *Fix a character* $c \in \Sigma$, *so* $c = a$ *or* $c = b$. *For each* $i = 1, \ldots, m+1$, *let* $\gamma_i = \min\{\max\{j \mid t_1 \ldots t_{i-1} c^j t_i \ldots t_m \prec \mathbf{s}\}, T - m\}$. *Then, for each* $i = 1, \ldots, m+1$, *we can determine* γ_i *using* $2\log \gamma_i + 1$ *queries, or alternatively, using* $\gamma_i + 1$ *queries. In particular, we can determine all* γ_i *using at most* $m + 1 + \sum_{i=1}^{m+1} \gamma_i$ *queries.*

Proof. We can determine all the values γ_i either with one-sided binary search, using $2\log \gamma_i + 1$ queries, or with linear search, using $\gamma_i + 1$ queries. □

Example 4.7. *Note that Lemma 4.6 only assumes that* $\mathbf{t}$ *fixes the direction of* $\mathbf{s}$, *but not that the positions in* $\mathbf{s}$ *to which the characters of* $\mathbf{t}$ *are matched are also fixed. Consider the following example. Let* $\mathbf{s} = a^{10}ba^{10}ba^{10}$. *Then* $\mathbf{t} = aaa$ *fixes the direction of* $\mathbf{s}$. *For* $c = b$, *we get* $\gamma_1 = \gamma_2 = \gamma_3 = \gamma_4 = 2$. *For* $c = a$, *we have* $\gamma_i = \min(27, 19) = 19$ *for all* i *(since* $T = 22$ *and* $m = 3$).

The next lemma says that if there are long a-runs or long b-runs, then there cannot be many runs.

Lemma 4.8. *Let* $\mathbf{s} = a^{x_1} b^{y_1} \ldots a^{x_\rho} b^{y_\rho}$. *Assume that there are* $1 \leq i_1 < i_2 < \cdots < i_q \leq \rho$ *and* $k \geq 0$, *such that* $x_{i_j} \geq T - B - k$ *(resp.* $y_{i_j} \geq T - A - k$) *for each* $j = 1, \ldots, q$, *and for at least one value of* j *it holds that* $x_{i_j} > T - B - k$ *(resp.* $y_{i_j} > T - A - k$). *Then*

$$\rho_a \leq n - B - q(T - B - k - 1) - 1 \qquad (resp. \ \rho_b \leq n - A - q(T - A - k - 1) - 1).$$

7

Proof. We limit ourselves to showing the argument for ρ_a, the number of non-empty a-runs. Since each run counted by ρ_a has at least one a, we have the desired inequality:

$$n - B = A \geq \sum_{j=1}^{q} x_{i_j} + \rho_a - q \geq q(T - B - k) + 1 + \rho_a - q.$$

$\square$

4.1 The case where $A \geq T$

Since $A \geq T = \left\lceil \frac{2}{3}(n+1) \right\rceil$, we have that $B \leq \frac{n}{3} - \frac{2}{3}$, so $2B + 1 = 2(n-A) + 1 \leq \frac{2}{3}(n+1) \leq T$. This implies that we can ask queries which include $B + 1$ many a's and up to B many b's. Let $\beta = \left\lceil \frac{n}{3} - \frac{2}{3} \right\rceil$, and $\mathbf{t} = a^{\beta+1}$. We have $B \leq \beta$ and, therefore, $\mathbf{t}$ fixes the direction of $\mathbf{s}$. Notice also that $B + \beta + 1 \leq T$.

By Lemma 4.6, with $\mathbf{t} = a^{\beta+1}$, we can find $L = \max\{j \mid b^j a^{\beta+1} \prec \mathbf{s}\}$ with $O(\log L)$ queries. Likewise, with $\mathbf{t} = b^L a^{\beta+1}$ we can find $R = \max\{j \mid b^L a^{\beta+1} b^j \prec \mathbf{s}\}$, with $O(\log R)$ queries.

Notice that in $\mathbf{s}$, between the left-most L many b's and the right-most R many b's, there may be more than $\beta + 1$ many a's. More precisely, with reference to (1), the previous queries guarantee that there are $1 \leq i \leq j \leq \rho$ such that $\sum_{k=1}^{i-1} y_k = L$ and $\sum_{k=j}^{\rho} y_k = R$. Let $\mathbf{w}$ be the substring of $\mathbf{s}$ between the L left-most and the R right-most b's, i.e., $\mathbf{w} = a^{x_i} b^{y_i} \cdots a^{x_j}$. Moreover, let $\mathbf{s}_{\text{left}}$ and $\mathbf{s}_{\text{right}}$ be such that $\mathbf{s} = \mathbf{s}_{\text{left}} \mathbf{w} \mathbf{s}_{\text{right}}$. We know that $|\mathbf{s}_{\text{left}}|_b = L, |\mathbf{s}_{\text{right}}|_b = R$, and $|\mathbf{w}|_a \geq \beta + 1$. We will first determine all but the first a-run of $\mathbf{w}$ and all of its b-runs, in particular yielding the exact value of B. Then we determine $\mathbf{s}_{\text{left}}$ and $\mathbf{s}_{\text{right}}$. For any a-runs that have length at least $T - B$, their exact value will be determined during the final stage.

We have $a^{\beta+1} \prec \mathbf{w}$, and by the definition of L and R we also have that $\sum_{k=i+1}^{\rho} x_k \leq \beta$ and $x_i > \sum_{k=j+1}^{\rho} x_k$. It follows that, for $\chi = 1, 2, 3 \ldots, \beta$, the query $b^L a^{\chi} b a^{\beta+1-\chi} b^R$ answers negatively as long as $\sum_{k=i+1}^{j} x_k < \beta + 1 - \chi$. Let χ^* be the first value for which the answer to this query is *yes*, and $\chi^* = \beta + 1$ if the answer is *no* for all values of χ. It is easy to see that $\chi^* = \beta + 1 - \sum_{k=i+1}^{j} x_k$. In particular, $\chi^* = \beta + 1$ if and only if $\mathbf{w}$ does not contain any b's. In this case, set $\mathbf{w}' = a^{\beta+1}$.

If $\chi^* \leq \beta$, define $\mathbf{t} = b^L a^{\chi^*} b a^{\beta+1-\chi^*} b^R$. By Lemma 4.6, with $\mathbf{t}$ we can find the value of y_k for each $k = i, \ldots, j-1$. As a side effect, we also determine the value of x_k for $k = i+1, \ldots, j$. Now we know that $b^L \mathbf{w}' b^R \prec \mathbf{s}$, where $\mathbf{w}' = a^{\chi^*} b^{y_i} a^{x_{i+1}} \ldots b^{y_{j-1}} a^{x_j}$. In other words, we know $\mathbf{w}$ except for its first a-run, which may be longer than χ^*. We also know B, the number of b's of $\mathbf{s}$.

Now we turn to $\mathbf{s}_{\text{right}}$. Let us denote by $\mathbf{w}' - a^{\ell}$ an arbitrary sequence obtained by removing exactly ℓ many a's from $\mathbf{w}'$ and leaving the rest as it is. Now we can use queries of the form $b^L (\mathbf{w}' - a^{\ell}) b^r a^{\ell} b^{R-r}$ with $r = 1, \ldots, R$ and $\ell = 1, 2, 3 \ldots$, in order to determine the values of x_k, for each $k = j+1, \ldots, \rho$. To see this, it is enough to notice that each such query contains $\beta + 1$ many a's, therefore it can only be a subsequence of $\mathbf{s}$ and not of $\tilde{\mathbf{s}}$. Moreover, we notice that in order to determine x_k we need to receive a positive answer to the query $b^L (\mathbf{w}' - a^{x_k}) b^{\sum_{\ell=j}^{k-1} y_\ell} a^{x_k} b^{R - \sum_{\ell=j}^{k-1} y_\ell}$ and a negative answer to the query $b^L (\mathbf{w}' - a^{x_k-1}) b^{\sum_{\ell=j}^{k-1} y_\ell} a^{x_k+1} b^{R - \sum_{\ell=j}^{k-1} y_\ell}$. Because of $\sum_{k=j+1}^{\rho} x_k < \beta + 1$, both these queries have length not larger than T. Again, by determining x_k for each $k = j+1, \ldots, \rho$, we also determine y_k for each $k = j+1, \ldots, \rho$.

By an analogous procedure, we can determine $\mathbf{s}_{\text{left}}$ and the first a-run of $\mathbf{w}$, i.e. all the values x_k, for $k = 1, \ldots, i$, where $x_k \leq T - B$. Again, in this process, we also determine the size of the runs of b's, i.e., the y_k, for each $k = 1, \ldots, i-1$.

Finally, we compute the size of the a-runs in $\mathbf{s}$ that are larger than $T - B$. Notice that for at most two indices we can have $x_k \geq T - B$, for otherwise their total sum would be larger than n, the total length of the string. If there is exactly one such x_k, then we can compute it as $x_k = n - B - \sum_{\ell \neq k} x_\ell$. Otherwise, let $1 \leq i_1 < i_2 \leq i$ be such that $x_{i_1}, x_{i_2} \geq T - B$. Then it must hold that $n - B - \sum_{\ell \neq i_1, i_2} x_\ell = 2(T - B)$, and thus, $x_{i_1}, x_{i_2} = T - B$. Otherwise, we would have that $x_1 + x_2 > 2(T - B)$, and using Lemma 4.8, with $k = 0$, we can then conclude that $\rho_a \leq n - 2T + B + 1 \leq -1$, a contradiction.

Notice that we use at most one query per character of $\mathbf{s}$ plus at most one query for each run of $\mathbf{s}$. Therefore, in total we have $O(\sum_i (x_i + 1) + \sum_i (y_i + 1)) = O(n)$.

4.2 The case $T > A > B$

By Lemma 4.6 with $\mathbf{t} = a^A$ and $c = b$, with $O(n)$ queries we can determine exactly y_k for each k such that $y_k < T - A$. In the process, we also find out exactly x_k for each $k = 1, \ldots, \rho_a$. The only problem now is to determine those runs of b's which have length at least $T - A$.

Let $i_1, \ldots, i_q$ be the q distinct indices of the runs of b's such that $y_{i_j} \geq T - A$, so we have not yet been able to determine their exact value. Clearly, if $q = 1$, we can compute $y_{i_1} = B - \sum_{\ell \neq i_1} y_\ell$. Likewise, if $B - \sum_{\ell \neq i_1, \ldots, i_q} y_\ell = q(T - A)$, then we know $y_{i_j} = T - A$ for all i_j. Otherwise, it must hold that $\sum_{j=1}^q y_{i_j} > q(T - A)$. Let $y_{i_1} \geq y_{i_2} \geq \cdots \geq y_{i_q}$ and $\alpha > 0$ such that $y_{i_1} = T - A + \alpha$. We have

$$\rho_b \leq B - (y_{i_1} + y_{i_2}) + 2 = n - A - (T - A + \alpha) - y_{i_2} + 2 \leq \frac{n}{3} + \frac{4}{3} - \alpha - y_{i_2}.$$

Now, consider the sequence $\mathbf{t}_\chi = (ab)^{i_2 - 1} ab^\chi (ab)^{\rho_b - i_2}$. For each $\chi = T - A + 1, T - A + 2, \ldots, y_{i_2} + 1$, such a string has length at most T, since we have

$$|\mathbf{t}_\chi| = 2\rho_b + \chi - 1 \leq 2\rho_b + y_{i_2} \leq \frac{2n}{3} + \frac{8}{3} - 2\alpha - 2y_{i_2} + y_{i_2} = \frac{2n}{3} + \frac{8}{3} - 2\alpha - y_{i_2} \leq T - 1, \quad (2)$$

where the last inequality follows from the fact that $\alpha, y_{i_2} \geq 1$.

We will finish the proof for the case $T > A > B$ by distinguishing four subcases according to whether $s_1 = s_n$ and whether $s_1 = a$ or $s_1 = b$. (Note that due to the assumption $A > B$ we cannot assume w.l.o.g. the identity of the first character.)

Case 1. If $s_1 = s_n = b$, then $\rho_b = \rho$, we can remove the first a from $\mathbf{t}_\chi$, and the new query fixes the direction of $\mathbf{s}$. This query has length at most T, so we can identify y_{i_2}. By the same argument, we can also identify y_{i_j}, for each $j = 3, \ldots, q$, since $y_{i_j} \leq y_{i_2}$, for each such j. Finally we can determine y_{i_1} by subtraction.

Case 2. If $s_1 = s_n = a$, then $\rho_b = \rho - 1$. Now we have to add an a at the end to get a query which fixes the direction of $\mathbf{s}$, and its length is again at most T. The argument is then analogous to Case 1.

Case 3. Let $s_1 \neq s_n$ and $s_1 = b$. This case is analogous to Case 4. below, replacing $\mathbf{t}_\chi$ by $\mathbf{u}_\chi = (ba)^{i_2 - 1} b^\chi a (ba)^{\rho_b - i_2}$ and all following sequences accordingly.

Case 4. As the final case we have $s_1 \neq s_n$ and $s_1 = a$, so $\rho_b = \rho$. We will now look at the value of $X := x_{\rho - i_2 + 1}$. Note that any query $\mathbf{t}_\chi$ with $\chi \leq X$ would answer *yes* because it would be

9

interpreted as $\tilde{\mathbf{t}}_\chi$. Notice that we know the value of X. If $X < T - 2\rho$, then we ask query $\mathbf{t}_\chi$ for $\chi = X + 1$. If the answer is *yes*, we continue with $X + 2, X + 3 \ldots$ until we receive the first *no*, and we are done, since the last χ where $\mathbf{t}_\chi$ answered positively was equal to y_{i_2}. By (2), these queries do not exceed the threshold.

Otherwise, if the query $\mathbf{t}_{X+1}$ answered *no* or if $X > T - 2\rho$, then we know that $y_{i_2} \leq X$. In this case, we use the following queries to determine y_{i_2}.

Let w.l.o.g. $i_2 \leq \rho - i_2 + 1$ (otherwise exchange the roles of i_2 and $\rho - i_2 + 1$ in the formulas below). Define $\mathbf{t}'_\xi = (ab)^{i_2-1}a^\xi(ba)^{\rho-2i_2+1}b^{\xi+1}(ab)^{i_2-1}$. One can verify that for each $\xi = T - A, T - A + 1, \ldots, y_{i_2}$, we have

$$|\mathbf{t}'_\xi| \leq 2\rho_b + 2y_{i_2} - 1 \leq \frac{2n}{3} + \frac{2}{3} - 2\alpha + 1 \leq T + 1 - 2\alpha \leq T - 1, \tag{3}$$

where the last inequality follows from the fact that $\alpha \geq 1$.

We can ask queries $\mathbf{t}'_\xi$ until either we receive a negative answer or we cannot enlarge it further because it would violate the bound T. The largest value of ξ for which we receive a positive answer to query $\mathbf{t}'_\xi$ correctly gives the value of y_{i_2}. Clearly this is true if we also receive a negative answer, for the next larger value. If, instead, we had to stop because of the bound T, we can be sure that $\xi = y_{i_2}$, because if $y_{i_2} > \xi$, then this would contradict the inequality (3).

We ask at most one query per character plus one query per run, except for Case 4, where we might use two queries per character of the y_{i_2}'th run of b's. Altogether, we have that the total number of queries is $O(n)$.

4.3 The case $T > A = B = \frac{n}{2}$, $s_1 = s_n$

We assume w.l.o.g. that the string starts and ends in a. Therefore, with reference to (1), in this section we have $y_\rho = 0$ and our string looks like this:

$$\mathbf{s} = a^{x_1}b^{y_1}a^{x_2}b^{y_2}\ldots a^{x_{\rho-1}}b^{y_{\rho-1}}a^{x_\rho},$$

with all $x_i, y_i > 0$, i.e., it includes $\rho = \rho_a$ runs of a's and $\rho - 1 = \rho_b$ runs of b's.

By Lemma 4.6 with $\mathbf{t} = ab^{\frac{n}{2}}$ we can exactly determine x_k (run of a's) for each k, such that $x_k < T - \frac{n}{2} - 1 \leq \frac{n}{6} - \frac{1}{3}$. In this process, we determine exactly y_k, for each $k = 1, \ldots, \rho_b$.

Let $1 \leq i_1 < i_2 < \cdots < i_q \leq \rho_a$ be all the indices of the runs of a's whose length we have not been able to determine exactly, i.e., such that $x_{i_j} \geq T - \frac{n}{2} - 1$. By $A = \frac{n}{2}$, we have that $q \leq 3$. In fact, the interesting cases are $q = 2$ and $q = 3$, since, for $q = 1$ we can determine the only missing x_{i_1} as the difference between A and the sum of the remaining x_k's.

For $q = 3$, by Lemma 4.8, we have $\rho_a \leq 3$, thus it follows that $\rho_a = 3$. Let $\mathbf{t} = ababa$, and $c = a$. By Lemma 4.6 we can determine each x_k, such that $x_k \leq T - 5$. Suppose that for all $k = 1, 2, 3$, it holds that $x_k \geq T - 5$. Since there must exist one run of a's of length $\leq \frac{n}{6}$, we have that $n \leq 9$, whence $A \leq 4$, implying that the only possible case is to have two runs of a's of length 1 and one run of a's of length 2. Direct inspection shows that in this case we can easily reconstruct the whole string with T-bounded queries.

Finally, if $q = 2$, by Lemma 4.8, we have $\rho_a \leq \frac{n}{6} + \frac{5}{3}$. We can now use query $\mathbf{t}_1 = (ab)^{i_1-1}a^{T-\frac{n}{2}-1+\chi}(ba)^{\rho-i_1}$, for $\chi = 1, 2, 3, \ldots$, until we receive a negative answer, then $x_{i_1} = T - \frac{n}{2} - 1 + \chi - 1$. If we never receive a negative answer and the query becomes of length T, we can resort to the query $\mathbf{t}_2 = (ab)^{i_2-1}a^{T-\frac{n}{2}-1+\chi}(ba)^{\rho-i_2}$, for $\chi = 1, 2, \ldots$, and proceed analogously. It is easy to see that we

cannot have that both $\mathbf{t}_1$ and $\mathbf{t}_2$ exceed the threshold T; the other value can then be determined by difference.

We have used $O(\log n + A) = O(n)$ many queries.

4.4   The case $T > A = B = \frac{n}{2}$, $s_1 \neq s_n$

Recall that by Lemma 4.3, in this case we can exactly determine s_1 and s_n. Let us assume w.l.o.g. that $s_1 = a$ and $s_n = b$. (Otherwise, rename the characters.) Then the string $\mathbf{s}$ has the following form:

$$\mathbf{s} = a^{x_1}b^{y_1}a^{x_2}b^{y_2}\ldots a^{x_{\rho-1}}b^{y_{\rho-1}}a^{x_\rho}b^{y_\rho}.$$

In particular, it starts with a run of a's and ends with a run of b's.

We need some more notation. For each $i = 1, 2, \ldots, 2\rho$, we use r_i to denote the size of the i'th run in $\mathbf{s}$ starting from the left. I.e., we have $x_i = r_{2i-1}$ and $y_i = r_{2i}$ for each $i = 1, \ldots, \rho$. Also we denote by $m_i = \min\{r_i, r_{2\rho-i+1}\}$ and by $M_i = \max\{r_i, r_{2\rho-i+1}\}$. We use the following technical lemma.

Lemma 4.9. *Fix $i < \rho$ and assume that for each $k = 1, \ldots, i-1$, we know r_k and $r_{2\rho-k+1}$ and it holds that $r_k = r_{2\rho-k+1} < T - \frac{n}{2}$. Then we can determine m_i and $\min\{M_i, T - \frac{n}{2}\}$, asking at most $\max\{m_i, \min\{M_i, T - \frac{n}{2}\}\}$ queries.*

Proof. For each odd i (i.e., r_i denotes the length of a run of a's) we have

$$m_i = \min\left\{\chi = 1, 2, 3, \ldots \mid \mathbf{t}_\chi = a^{x_1+\cdots+x_{i-1}+\chi}ba^{\frac{n}{2}-(x_1+\cdots+x_{i-1}+\chi)} \prec_{\mathrm{RC}} \mathbf{s}\right\},$$

$$\min\left\{M_i, T - \frac{n}{2}\right\} = \max\left\{\chi = m_i, m_i + 1, \ldots, T - \frac{n}{2} \mid \right.$$
$$\left. \mathbf{q}_\chi = a^{\frac{n}{2}-(y_1+\cdots+y_{i-1})}b^\chi a^{y_1+\cdots+y_{i-1}} \prec_{\mathrm{RC}} \mathbf{s}\right\}.$$

Using the above equalities, one can determine the value m_i (resp. $\min\{M_i, T - \frac{n}{2}\}$) by asking the query $\mathbf{t}_\chi$ (resp. $\mathbf{q}_\chi$) for increasing values of χ, until the first positive (resp. negative) answer, or until $\chi = T - \frac{n}{2}$. This settles the case of i odd.

It is not hard to see that exactly the same argument holds for even i, using the following:

$$m_i = \min\left\{\chi = 1, 2, \ldots \mid \mathbf{t}_\chi = b^{y_1+\cdots+y_{i-1}+\chi}ab^{\frac{n}{2}-(y_1+\cdots+y_{i-1}+\chi)} \prec_{\mathrm{RC}} \mathbf{s}\right\},$$

$$\min\left\{M_i, T - \frac{n}{2}\right\} = \max\left\{\chi = m_i, m_i + 1, \ldots, T - \frac{n}{2} \mid \right.$$
$$\left. \mathbf{q}_\chi = b^{\frac{n}{2}-(x_1+\cdots+x_{i-1})}a^\chi b^{x_1+\cdots+x_{i-1}} \prec_{\mathrm{RC}} \mathbf{s}\right\}.$$

This completes the proof of the lemma. $\qquad\square$

Now, let us consider the largest $k \geq 1$ such that $r_j = r_{2\rho-j+1} < T - \frac{n}{2}$ for each $j < k$. Note that by repeated application of Lemma 4.9, we can determine all these r_j's. Assume w.l.o.g. that k is odd and let $i = \lceil k/2 \rceil$. Then we can write:

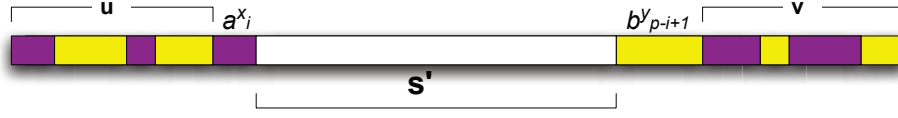$$\mathbf{s} = \mathbf{u}a^{x_i}\mathbf{s}'b^{y_{\rho-i+1}}\tilde{\mathbf{u}}, \tag{4}$$

11

Figure 1: The case where $|\mathbf{s}|_a = |\mathbf{s}|_b$ and $s_1 \neq s_n$. We determine $\mathbf{s}$ by first finding the first assymetry in $\mathbf{s}$ ($x_i \neq y_{\rho-i+1}$), and then extending queries for $\mathbf{s}'$, which has fewer b's than a's. Note that up to index i, string $\mathbf{s}$ is perfectly symmetric, i.e. we have $\mathbf{v} = \tilde{\mathbf{u}}$.

where $\mathbf{u} = a^{x_1}b^{y_1}\ldots a^{x_{i-1}}b^{y_{i-1}}$ is known, and the string $\mathbf{s}'$ is still unknown. Note that also the two values $\min\{x_i, y_{\rho-i+1}\}$ and $\min\{\max\{x_i, y_{\rho-i+1}\}, T - \frac{n}{2}\}$ are known (again by application of Lemma 4.9). Moreover, for determining these two values and string $\mathbf{u}$, we have used a number of queries linear in $2|\mathbf{u}| + \min\{\max\{x_i, y_{\rho-i+1}\}, T - \frac{n}{2}\}$.

According to the magnitude of x_i and $y_{\rho-i+1}$, we will enter one of the following three cases, where we will assume, w.l.o.g., that $x_i \leq y_{\rho-i+1}$. (The case where $y_{\rho-i+1} < x_i$ is symmetric.) We illustrate the situation in Figure 1.

4.4.1 The subcase $A = B = \frac{n}{2}$, $s_1 \neq s_n$, $x_i, y_{\rho-i+1} < T - \frac{n}{2}$.

With reference to (4), we can use a recursive argument to show how to determine $\mathbf{s}'$. Let $n' = |\mathbf{s}'|$. Note that $|\mathbf{s}'|_a > |\mathbf{s}'|_b$ and that $\mathbf{s}'$ starts with a b and ends with an a.

Let $\mathbf{t}'$ be a query for $\mathbf{s}'$: Since $|\mathbf{s}'|_a > |\mathbf{s}'|_b$, such queries were defined by one of the previous cases (Section 4.1 or 4.2). Let $\mathbf{t}'_+$ be the query obtained by adding to $\mathbf{t}'$ an initial b, if $\mathbf{t}'$ does not begin with b, and a final a, if $\mathbf{t}'$ does not end with an a. Define a query $\mathbf{t}$ for $\mathbf{s}$ in the following way:

$$\mathbf{t} = a^{|\mathbf{u}|_a}\mathbf{t}'_+ a^{|\mathbf{u}|_b} \tag{5}$$

Lemma 4.10. *Let $\mathbf{t}$ be defined as in Eq. (5). Then,*

1. $\mathbf{t} \prec_{\mathrm{RC}} \mathbf{s}$ *if and only if* $\mathbf{t}' \prec_{\mathrm{RC}} \mathbf{s}'$.

2. *If* $|\mathbf{t}'| \leq \frac{2(n'+1)}{3}$, *then* $|\mathbf{t}| \leq \frac{2(n+1)}{3}$.

Proof. 1. Let $\mathbf{t} \prec_{\mathrm{RC}} \mathbf{s}$. First assume that $\mathbf{t} \prec \mathbf{s}$. Notice that $\mathbf{t}'_+$ starts with character b and ends with character a, and that $\mathbf{t} = a^{|\mathbf{u}|_a}\mathbf{t}'_+ a^{|\tilde{\mathbf{u}}|_a}$, i.e., the number of a's in $\mathbf{t}$ following $\mathbf{t}'_+$ equals the number of a's in $\tilde{\mathbf{u}}$. Because of the $|\mathbf{u}|_a$ many a's at the beginning of $\mathbf{t}$, the fact that $\mathbf{t}$ is a subsequence of $\mathbf{s}$ implies $\mathbf{t}'_+ a^{|\tilde{\mathbf{u}}|} \prec a^{x_i}\mathbf{s}'b^{y_{\rho-i+1}}\tilde{\mathbf{u}}$, and because $\mathbf{t}'_+$ starts with a b, we also have $\mathbf{t}'_+ a^{|\tilde{\mathbf{u}}|} \prec \mathbf{s}'b^{y_{\rho-i+1}}\tilde{\mathbf{u}}$. This again implies that $\mathbf{t}'_+ \prec \mathbf{s}'b^{y_{\rho-i+1}}$, and because $\mathbf{t}'_+$ ends with an a, also $\mathbf{t}'_+ \prec \mathbf{s}'$, and thus, $\mathbf{t}' \prec \mathbf{s}'$.

Now let $\mathbf{t} \prec \tilde{\mathbf{s}}$, or, equivalently, $\tilde{\mathbf{t}} \prec \mathbf{s}$. We have $\tilde{\mathbf{t}} = b^{|\mathbf{u}|_b}\tilde{\mathbf{t}}'_+ b^{|\mathbf{u}|_a} = b^{|\mathbf{u}|_b}\tilde{\mathbf{t}}'_+ b^{|\tilde{\mathbf{u}}|_b}$, and $\tilde{\mathbf{t}}'_+$ starts with an a and ends with a b. Thus, because of the $|\mathbf{u}|_b$ many b's at the beginning of $\tilde{\mathbf{t}}$ and the fact that $\tilde{\mathbf{t}}'_+$ starts with an a, we have $\tilde{\mathbf{t}}'_+ \prec \mathbf{s}'b^{y_{\rho-i+1}}\tilde{\mathbf{u}}$. Further, because of the $|\tilde{\mathbf{u}}|_b$ many b's at the end and the fact that $\tilde{\mathbf{t}}'_+$ ends with a b, this implies $\tilde{\mathbf{t}}'_+ \prec \mathbf{s}'$. It follows that $\mathbf{t}' \prec \mathbf{s}'$.

Conversely, if $\mathbf{t}'$ (resp. $\tilde{\mathbf{t}}'$) is a subsequence of $\mathbf{s}'$, then clearly, $\mathbf{t}$ (resp. $\tilde{\mathbf{t}}$) is a subsequence of $\mathbf{s}$.

2. The length of $\mathbf{t}$ is $|\mathbf{t}| \leq |\mathbf{u}| + 2 + |\mathbf{t}'|$, where $|\mathbf{t}'| \leq \frac{2}{3}(n'+1)$ and $n' = n - 2|\mathbf{u}| - x_i - y_{\rho-i+1}$, and $y_{\rho-i+1} > x_i \geq 1$. This implies $x_i + y_{\rho-i+1} \geq 3$. Thus,

12

$$|\mathbf{t}| \leq |\mathbf{u}| + 2 + \frac{2}{3}(n - 2|\mathbf{u}| - x_i - y_{\rho-i+1} + 1)$$
$$= \frac{2}{3}(n+1) + 2 - \frac{1}{3}|\mathbf{u}| - \frac{2}{3}(x_i + y_{\rho-i+1})$$
$$\leq \frac{2}{3}(n+1) + 2 - \frac{1}{3}|\mathbf{u}| - 2 \leq \frac{2}{3}(n+1).$$

$\square$

Thus it follows that we can use the analysis of the previous sections to prepare a sequence of queries on $\mathbf{s}$ which is (i) linear in $|\mathbf{s}'|$ and (ii) allows us to determine the substring $\mathbf{s}'$ of $\mathbf{s}$. Once this is accomplished, the whole $\mathbf{s}$ can be fully determined (up to reverse complement).

4.4.2 The subcase $A = B = \frac{n}{2}$, $s_1 \neq s_n$, $x_i, y_{\rho-i+1} \geq T - \frac{n}{2}$.

Notice that, because of the assumption $n \geq 8$ and $x_i, y_{\rho-i+1} \geq T - \frac{n}{2}$, it follows that $x_i + y_{\rho-i+1} \geq 4$. We have $|\mathbf{s}'| = n - 2|\mathbf{u}| - x_i - y_{\rho-i+1}$. This implies

$$|\mathbf{s}'| + |\mathbf{u}| + 2 \leq n - x_i - y_{\rho-i+1} + 2 + |\mathbf{u}| \leq 2n - 2T + 2 - |\mathbf{u}| \leq \frac{2n}{3} + \frac{2}{3} - |\mathbf{u}| \leq T. \tag{6}$$

Thus we can adapt the strategy we described in Section 3 for unbounded RC-reconstruction to determine $\mathbf{s}'$ and then, by subtraction, also x_i and $y_{\rho-i+1}$. We proceed as follows: Suppose that in the strategy for reconstructing $\mathbf{s}'$, in the unbounded-query case, we would ask a query $\mathbf{t}'$, starting with b and ending with a. Then we will ask query $\mathbf{t} = a^{|\mathbf{u}|_a+1}\mathbf{t}'ba^{|\mathbf{u}|_b}$. It is not hard to see that such $\mathbf{t}$ answers positively on $\mathbf{s}$ if and only if $\mathbf{t}'$ answers positively on $\mathbf{s}'$. By (6), $|\mathbf{t}| = |\mathbf{t}'| + 2 + |\mathbf{u}| \leq T$.

The only requirement is that $\mathbf{t}'$ begin with b and end with a. However, the strategy in Section 3 can be easily adapted to this case, under the assumption that the string to be reconstructed begins with b and ends with a, a condition that holds for $\mathbf{s}'$. (Notice, in fact, that because the query size is unbounded, any query in the strategy in Section 3 can be safely extended by an arbitrary prefix and/or suffix of the string we are trying to reconstruct.)

Finally, once we have reconstructed $\mathbf{s}'$ we can determine $\max\{x_i, y_{\rho-i+1}\}$ as $\frac{n}{2} - |\mathbf{s}'|_b - |\mathbf{u}|$. (Recall that we have assumed w.l.o.g. that $x_i \leq y_{\rho-i+1}$; in fact, now that we know $\mathbf{s}'$, we can determine whether this is the case: we have $x_i \leq y_{\rho-i+1}$ if and only if $|\mathbf{s}'|_a \geq |\mathbf{s}'|_b$.)

4.4.3 The subcase $A = B = \frac{n}{2}$, $s_1 \neq s_n$, $x_i < T - \frac{n}{2}, y_{\rho-i+1} \geq T - \frac{n}{2}$.

In order to determine ρ and $x_{i+1}, \ldots, x_{\rho-i+1}$, we can use the query

$$\mathbf{t}_\chi = a^{|\mathbf{u}|_a+x_i}ba^\chi ba^{\frac{n}{2}-|\mathbf{u}|_a-x_i-\chi} \tag{7}$$

as follows. Under the standing hypothesis, we have $x_i < \frac{2(n+1)}{3} - \frac{n}{2} \leq y_{\rho-i+1}$. The above query $\mathbf{t}_\chi$ has size $\frac{n}{2} + 2 \leq T$, for any $n \geq 8$. Moreover, the fact that $x_i < y_{\rho-i+1}$ guarantees that if $\mathbf{t}_\chi \prec \mathbf{s}$ then $\mathbf{t}$ fixes the direction of $\mathbf{s}$. To see this, with reference to (4), it is enough to observe that in this case, in $\mathbf{s}$ there are more a's following the first b of $\mathbf{s}'$ than there are b's preceeding the last a of $\mathbf{s}'$.

We use the query $\mathbf{t}_\chi$ as follows: We ask $\mathbf{t}_\chi$ for each $\chi = 1, 2, 3 \ldots$, until we get the first positive answer. Let χ_1 be the minimum value of χ for which the answer is positive. It is not hard to see

13

that this implies $x_{i+1} = \chi_1$. We now continue asking query $\mathbf{t}_\chi$ for each $\chi = \chi_1 + 1, \chi_1 + 2, \ldots$. Let χ_2 be the minimum value of χ for which we get a new positive answer. Again, this implies that $x_{i+2} = \chi_2 - \chi_1$. More generally, for each $j = 1, \ldots, \rho - i + 1$, let χ_j be the value of χ when we receive the ith positive answer. Then, we have $x_{i+j} = \chi_j - \chi_{j-1}$ (where we set $\chi_0 = 0$ for sake of definiteness).

Note, however, that at this point we do not know ρ. We continue asking $\mathbf{t}_\chi$ as long as $\frac{n}{2} - |\mathbf{u}|_a - x_i - \chi > |\mathbf{u}|_b$, or equivalently, $\chi < \frac{n}{2} - |\mathbf{u}| - x_i$. This way we determine x_j, for $j = i+1, \ldots, \rho - i + 1$ and, in particular, we determine ρ.

Now by Lemma 4.6, with $\mathbf{t} = a^{|\mathbf{u}|_a + x_i} b a^{\frac{n}{2} - x_i - |\mathbf{u}|_a}$, we can determine exactly y_j, for each $j = i, \ldots, \rho - i$ such that $y_j < T - \frac{n}{2}$, or, otherwise, establish the fact that $y_j \geq T - \frac{n}{2}$. As in the previous cases, it now remains to determine the exact values of those runs with length at least $T - \frac{n}{2}$.

Let $i_1, \ldots, i_q$, be such that $y_{i_j} \geq T - \frac{n}{2}$, for each $j = 1, \ldots, q$. We can also assume that for at least one $1 \leq j \leq q$ it holds that $y_{i_j} > T - \frac{n}{2}$, for otherwise we can identify this situation by the fact that $n - \sum_{\ell \notin \{i_1, \ldots, i_q\}} y_\ell = q(T - \frac{n}{2})$, whence we have $y_{i_j} = T - \frac{n}{2}$, for each j.

For each j such that $y_{i_j} \geq T - \frac{n}{2}$ and whose value is not determined yet, we use a query of the form:

$$\mathbf{t}_\chi = (ab)^{i_j - 1} a b^\chi (ab)^{\rho - i - i_j} a b^{x_i + 1} (ab)^{i-1},$$

increasing χ until we get the first positive answer. It remains to show that each of these queries has length smaller or equal to T.

We have that $|\mathbf{t}_\chi| = 2\rho + x_i + \chi - 1$. To see that this is smaller or equal to T for each $\chi \leq y_{i_j}$, notice that $y_{i_j} \geq \frac{2(n+1)}{3} - \frac{n}{2} = \frac{n}{6} + \frac{2}{3}$. Further, by assumption, we have $y_{\rho-i+1}, y_{i_j} \geq \frac{n}{6} + \frac{2}{3}$, implying $\rho \leq \frac{n}{2} - \frac{2n}{6} + \frac{2}{3} = \frac{n}{6} + \frac{2}{3} \leq y_{\rho-i+1}$. Thus, we have $\rho \leq y_{\rho-i+1}$. Moreover, recall that $\frac{n}{2} = B \geq y_{\rho-i+1} + \rho + y_{i_j} - 2$. Putting it all together, we get

$$\mathbf{t}_{y_{i_j}} = 2\rho + x_i + y_{i_j} - 1 \leq \underbrace{y_{\rho-i+1} + \rho + y_{i_j} - 1}_{\leq B+1 = \frac{n}{2}+1} + x_i \leq \frac{n}{2} + \underbrace{x_i + 1}_{\leq \frac{n}{6}+\frac{2}{3}} \leq \frac{2(n+1)}{3} \leq T.$$

As can be readily seen, in all three subcases we use $O(|\mathbf{s}'|)$ queries to determine $\mathbf{s}'$, hence, altogether $O(|\mathbf{s}|)$ queries to complete the reconstruction.

This ends the reconstruction for binary strings. In the next two sections, we turn to the general case of arbitrary constant size paired alphabets.

5 Bounded query size over an arbitrary paired alphabet

In this and the next section we consider the reconstruction of an unknown string $\mathbf{w}$ over a paired alphabet $\Sigma = \{a_1, \overline{a_1}, \ldots, a_\delta, \overline{a_\delta}\}$, with $\delta > 1$, using only bounded queries of size not larger than $T = \left\lceil \frac{2}{3}(n+1) \right\rceil$, where $n = |\mathbf{w}|$. We will show the following result:

Theorem 5.1. *There is an algorithm which reconstructs, up to RC-equivalence, a string $\mathbf{s}$ of length n over a paired alphabet Σ, using $O(|\Sigma|n)$ many RC-subsequence queries of length at most $\left\lceil \frac{2}{3}(n+1) \right\rceil$.*

We will proceed as follows. In this section, we will show that it is possible to determine each projection of $\mathbf{w}$, i.e. the maximal subsequence $\mathbf{w}|_i$ consisting only of characters a_i and $\bar{a}_i$, using the

14

methodology from previous sections. Once determined, we will combine these iteratively. We will also show that it is easy to find a split of the alphabet into two subsets such that all but the last of these merges can be done using unbounded queries. Finally, in Section 6, we examine, case by case, how to execute the final merging.

We want to extend the notion of projection introduced in Section 2 to the case of projection over a set of complement pairs.

Definition 5.2 (*S*-projection)**.** *Let $S \subset [\delta]$, where, using standard notation, $[\delta] = \{1, 2, \ldots, \delta\}$. We denote by $\mathbf{w}_{|S}$ the longest subsequence of $\mathbf{w}$ only containing characters from complement pairs $\{a_i, \overline{a_i}\}$ whose index i is in S. We call $\mathbf{w}_{|S}$ the S-projection of $\mathbf{w}$. (Note that when S is a singleton, $S = \{i\}$, then $\mathbf{w}_{|S} = \mathbf{w}_{|i}$, as defined previously.)*

As a consequence of the results in the previous sections, we have a method for determining the maximal subsequences that consist only of one complement pair, as we will now see.

5.1 Determining the projections

Recall that δ is the number of complement pairs. For each $i = 1, \ldots, \delta$, let $n_i = |\mathbf{w}_{|i}|$ be the length of the projection onto the complement pair $\{a_i, \overline{a_i}\}$. Note that for at most one projection we can have $n_i \geq T$. Also, recall that $A_i = \max\{|\mathbf{w}|_{a_i}, |\mathbf{w}|_{\overline{a_i}}\}$. We will now determine the $\mathbf{w}_{|i}$'s in the following way:

Step 1: First, for each $i = 1, \ldots, \delta$, attempt to determine A_i by asking queries a_i^χ for $\chi = 1, 2, 3, \ldots$. This will be possible for all pairs $\{a_i, \overline{a_i}\}$ except at most one.

Step 2: For all i such that $A_i < T$, attempt to determine the subsequence $\mathbf{w}_{|i}$ using the unbounded query strategy of Section 3. For each i such that $n_i < T$, this will result in full knowledge of $\mathbf{w}_{|i}$ and thus of n_i.

Step 3: If there is an $i \in [\delta]$ for which we have not been able to determine $\mathbf{w}_{|i}$ (we either reached the threshold T in step 1 or in step 2), then this is the only such i. Therefore we can determine n_i by difference, since $n_i = n - \sum_{j \neq i} n_j$. Now we can employ the binary algorithm of Section 4 for finding $\mathbf{w}_{|i}$.

Thus, we have determined all projections onto the individual character pairs. We will now turn to how to combine them into longer strings.

5.2 Iteratively merging the projections

Clearly, there can be at most one projection of size greater than $2n/3$. Moreover, if all projections are smaller than $2n/3$, then we can split the set of projections into two subsets such that the total length of the substrings induced by each of the two subsets is no greater than $2n/3$, in the following way.

We want to find a set $S \subset [\delta]$ such that $|\mathbf{w}_{|S}| \leq 2n/3$ and for the complement set $S^c = [\delta] \setminus S$ it holds that either $|\mathbf{w}_{|S^c}| \leq 2n/3$ or S^c is a singleton. To do this, we order the projections in order of non-decreasing cardinality, i.e., $n_1 \leq n_2 \leq \cdots \leq n_\delta$. Let $k = \min\{i \mid \sum_{j=1}^{i} n_j > 2n/3\}$ and choose $S = [k-1]$. If $k = \delta$, then we are done. Else assume that $|\mathbf{w}_{|S^c}| = \sum_{j=k}^{\delta} n_j > 2n/3$. This implies that $\sum_{j=1}^{k-1} n_j < n/3$. Moreover, note that $\sum_{j=1}^{k} n_j > 2n/3$ implies $\sum_{j=k+1}^{\delta} n_j < n/3$, and

15

thus also $n_k < n/3$, since the cardinalities are non-decreasing. Thus we have $n = (\sum_{j=1}^{k-1} n_j) + n_k + (\sum_{j=k+1}^{\delta} n_j) < n/3 + n/3 + n/3$, a contradiction.

For merging the projections we need $O(n \log |\Sigma|)$ queries. The final merging step, the only one requiring bounded search, is taken care of in the next section.

6 Combining two projections of w over disjoint pairs of complementary characters

The problem we want to solve in this section is, given two strings $\mathbf{s}$ and $\mathbf{t}$, which are RC-character-disjoint (i.e. if $a \in \mathrm{alph}(\mathbf{s})$, then $a, \bar{a} \notin \mathrm{alph}(\mathbf{t})$, and vice versa), find a string $\mathbf{w}$, such that $\mathbf{s} \equiv_{\mathrm{RC}} \mathbf{w}_{|S}$ and $\mathbf{t} \equiv_{\mathrm{RC}} \mathbf{w}_{|S^c}$, where $S \subsetneq [\delta]$. As before, we can only use queries of length at most $T = \lceil \frac{2}{3}(n+1) \rceil$, where $n = |\mathbf{s}| + |\mathbf{t}|$.

The string $\mathbf{w}$, up to RC-equivalence, is an *interleaving* of $\mathbf{s}$ and $\mathbf{t}$, or of $\mathbf{s}$ and $\tilde{\mathbf{t}}$, a concept we will define precisely below.[1] Deciding which is the case means finding the *relative direction* of $\mathbf{s}$ and $\mathbf{t}$. In order to reconstruct $\mathbf{w}$, we need to (a) decide the relative direction of $\mathbf{s}$ and $\mathbf{t}$ and (b) find out how the two strings are interleaved. In the following, we will formalize these notions.

In the whole of this section, we will assume, without loss of generality, that $|\mathbf{s}| \le |\mathbf{t}|$, and that $\mathbf{s} \prec \mathbf{w}$.

6.1 Some preliminary observations

We first introduce a notion which we will use extensively in the following.

Definition 6.1 (Run structure)**.** *Let* $\mathbf{w} \in \Sigma^*$ *and* $\mathbf{s} = \mathbf{w}_{|S}$ *for some* $S \subsetneq [\delta]$*. Then* $\mathbf{w}$ *can be written uniquely as*

$$\mathbf{w} = \mathbf{u}_0 \mathbf{v}_1 \mathbf{u}_1 \mathbf{v}_2 \mathbf{u}_2 \dots \mathbf{v}_{r-1} \mathbf{u}_{r-1} \mathbf{v}_r \mathbf{u}_r,$$

where $\mathbf{v}_1 \dots \mathbf{v}_r = \mathbf{s}$*,* $\mathbf{u}_0$ *and* $\mathbf{u}_r$ *are possibly empty, and* $\mathbf{u}_1, \dots, \mathbf{u}_{r-1}, \mathbf{v}_1, \dots, \mathbf{v}_r \ne \epsilon$*. We define the* run structure *of* $\mathbf{s}$ *(in* $\mathbf{w}$*) as the partition* $\mathbf{v}_1, \dots, \mathbf{v}_r$ *of* $\mathbf{s}$*, together with two bits* $b_L(\mathbf{s})$ *and* $b_R(\mathbf{s})$*, where* $b_L(\mathbf{s}) = 1$ *if and only if* $\mathbf{u}_0 \ne \epsilon$ *and* $b_R(\mathbf{s}) = 1$ *if and only if* $\mathbf{u}_r \ne \epsilon$*. We say that* $\mathbf{s}$*'s run structure is* symmetric *if* $b_L(\mathbf{s}) = b_R(\mathbf{s})$ *and for all* $i = 1, \dots, r$*, it holds that* $|\mathbf{v}_i| = |\mathbf{v}_{r-i+1}|$*; otherwise, it is* asymmetric*.*

Given two projections $\mathbf{s}$ *and* $\mathbf{t}$ *with their respective run structures, we define the* interleaving $\mathbf{s} \odot \mathbf{t}$ *as the unique string* $\mathbf{w}$ *s.t. the given partitions, along with the given bits, are the respective run structures of* $\mathbf{s}$ *and* $\mathbf{t}$ *in* $\mathbf{w}$*.*

Note that from the run structure of $\mathbf{s}$ in $\mathbf{w}$ we can deduce the run structure of $\tilde{\mathbf{s}}$ in $\tilde{\mathbf{w}}$: It is the partition $\tilde{\mathbf{v}}_r, \dots, \tilde{\mathbf{v}}_1$ of $\tilde{\mathbf{s}}$, together with the bits $b_L(\tilde{\mathbf{s}}) = b_R(\mathbf{s})$ and $b_R(\tilde{\mathbf{s}}) = b_L(\mathbf{s})$.

Example 6.2. *Let* $\Sigma = \{a, \bar{a}, b, \bar{b}\}$*, and let* $\mathbf{w} = ab\bar{b}b\bar{a}\bar{a}ba\bar{a}$*. Then the projection onto* $\{a, \bar{a}\}$ *is* $\mathbf{s} = a\bar{a}\bar{a}a\bar{a}$*, and its complement is* $\mathbf{t} = b\bar{b}bb$*, with the following run structures. For* $\mathbf{s}$*:* $a, \bar{a}\bar{a}, a\bar{a}$*,* $b_L(\mathbf{s}) = b_R(\mathbf{s}) = 0$*, and for* $\mathbf{t}$*:* $b\bar{b}b, b$*, and* $b_L(\mathbf{t}) = b_R(\mathbf{t}) = 1$*. Then these can be combined in four ways: (1)* $\mathbf{s} \odot \mathbf{t} = ab\bar{b}b\bar{a}\bar{a}ba\bar{a} = \mathbf{w}$*, (2)* $\mathbf{s} \odot \tilde{\mathbf{t}} = a\bar{b}\bar{a}\bar{a}b\bar{b}\bar{b}a\bar{a}$*, (3)* $\tilde{\mathbf{s}} \odot \mathbf{t} = a\bar{a}b\bar{b}bbaab\bar{a}$*, and (4)*

---

[1] In other words, it is an element of the *shuffle* of $\mathbf{s}$ and $\mathbf{t}$, or of $\mathbf{s}$ and $\tilde{\mathbf{t}}$, see e.g. [13] for a definition of the related concept of shuffle.

$\tilde{\mathbf{s}} \odot \tilde{\mathbf{t}} = a\bar{a}\bar{b}aa\bar{b}\bar{b}\bar{b}\bar{a} = \tilde{\mathbf{w}}$. *There are only two distinct solutions up to RC-equivalnce, namely (1) and (2).*

The next lemma gives some cases in which there is only one possible solution, up to RC-equivalence. Recall that a string $\mathbf{s}$ is an RC-palindrome if $\tilde{\mathbf{s}} = \mathbf{s}$.

Lemma 6.3. *Let $\mathbf{s}, \mathbf{t} \in \Sigma^*$, be the projections of some $\mathbf{w}$ onto $S \subsetneq [\delta]$ and S^c, respectively. Let $b_L(\mathbf{s})$ and $b_R(\mathbf{s})$ be the bits associated to the run structure of $\mathbf{s}$ in $\mathbf{w}$.*

1. *If $b_L(\mathbf{s}) \neq b_R(\mathbf{s})$ then $\mathbf{w}$ is the unique string up to RC-equivalence such that $\mathbf{s} \equiv_{RC} \mathbf{w}_{|S}$ and $\mathbf{t} \equiv_{RC} \mathbf{w}_{|S^c}$.*

2. *If $\mathbf{s}$ is an RC-palindrome and its run structure is symmetric, then $\mathbf{s} \odot \mathbf{t} \equiv_{RC} \mathbf{s} \odot \tilde{\mathbf{t}}$.*

Proof. For *1.*, note that $b_L(\mathbf{s}) \neq b_R(\mathbf{s})$ implies $b_L(\mathbf{t}) \neq b_R(\mathbf{t})$, and since both are projections of the same string, we have $b_L(\mathbf{t}) = 1 - b_L(\mathbf{s})$. Thus the only possibilities of interleaving the two strings are $\mathbf{s} \odot \mathbf{t}$ or $\tilde{\mathbf{s}} \odot \tilde{\mathbf{t}}$, which are RC-equivalent.

For *2.*, let us assume that $b_L(\mathbf{s}) = b_R(\mathbf{s}) = 0$. (The case $b_L(\mathbf{s}) = b_R(\mathbf{s}) = 1$ is analogous.) Let the run structure of $\mathbf{s}$ be $\mathbf{v}_1, \ldots, \mathbf{v}_r$, and of $\mathbf{t}$, $\mathbf{u}_1, \ldots, \mathbf{u}_{r-1}$. We can now write $\mathbf{s} \odot \mathbf{t} = \mathbf{v}_1 \mathbf{u}_1 \mathbf{v}_2 \mathbf{u}_2 \ldots \mathbf{v}_{r-1} \mathbf{u}_{r-1} \mathbf{v}_r$. Then $\mathbf{s} \odot \tilde{\mathbf{t}} = \mathbf{v}_1 \tilde{\mathbf{u}}_{r-1} \mathbf{v}_2 \tilde{\mathbf{u}}_{r-2} \ldots \mathbf{v}_{r-1} \tilde{\mathbf{u}}_1 \mathbf{v}_r$, and $\widetilde{\mathbf{s} \odot \tilde{\mathbf{t}}} = \tilde{\mathbf{v}}_r \mathbf{u}_1 \tilde{\mathbf{v}}_{r-1} \ldots \mathbf{u}_{r-2} \tilde{\mathbf{v}}_2 \mathbf{u}_{r-1} \tilde{\mathbf{v}}_1 = \mathbf{v}_1 \mathbf{u}_1 \mathbf{v}_2 \mathbf{u}_2 \ldots \mathbf{v}_{r-1} \mathbf{u}_{r-1} \mathbf{v}_r = \mathbf{s} \odot \mathbf{t}$. $\square$

The next lemma gives an algorithm for determining the relative direction of $\mathbf{s}$ and $\mathbf{t}$.

Lemma 6.4. *Let $\mathbf{s}, \mathbf{t}, \mathbf{w} \in \Sigma^*$ such that $\mathbf{s} \equiv_{RC} \mathbf{w}_{|S}$ and $\mathbf{t} \equiv_{RC} \mathbf{w}_{|S^c}$. If neither $\mathbf{s}$ nor $\mathbf{t}$ is an RC-palindrome then we can determine their relative direction, i.e., assuming $\mathbf{s} \prec \mathbf{w}$ we can determine whether $\mathbf{t} \prec \mathbf{w}$ or $\tilde{\mathbf{t}} \prec \mathbf{w}$, using $O(n)$ many queries, where $n = |\mathbf{w}|$.*

Proof. Simon's Theorem [14] states that if two strings of length ℓ have the same set of subsequences of length up to $\lfloor \ell/2 \rfloor + 1$, then they are equal. Thus, given two distinct strings $\mathbf{u}$ and $\mathbf{v}$ of length ℓ, there exists a distinguishing subsequence $\mathbf{z}$ of length at most $\lfloor \ell/2 \rfloor + 1$ s.t. either $\mathbf{z} \prec \mathbf{u}$ and $\mathbf{z} \nprec \mathbf{v}$, or $\mathbf{z} \prec \mathbf{v}$ and $\mathbf{z} \nprec \mathbf{u}$. In Appendix A, we give an explicit construction in linear time.

Since $\mathbf{s}$ and $\mathbf{t}$ are not RC-palindromes, $\mathbf{s}$ and $\tilde{\mathbf{s}}$ are two distinct strings, likewise $\mathbf{t}$ and $\tilde{\mathbf{t}}$. Thus, we can find strings $\mathbf{z}_1, \mathbf{z}_2$, such that $|\mathbf{z}_1| \leq |\mathbf{s}|/2 + 1$, $|\mathbf{z}_2| \leq |\mathbf{t}|/2 + 1$, and, up to exchanging the roles of $\mathbf{s}$ and $\tilde{\mathbf{s}}$, and of $\mathbf{t}$ and $\tilde{\mathbf{t}}$,

$$\mathbf{z}_1 \prec \mathbf{s}, \mathbf{z}_1 \nprec \tilde{\mathbf{s}}, \quad \text{and} \quad \mathbf{z}_2 \prec \mathbf{t}, \mathbf{z}_2 \nprec \tilde{\mathbf{t}}.$$

In other words, there exist subsequences which fix the direction of $\mathbf{s}$ and $\mathbf{t}$, and the sum of their lengths is at most $n/2 + 2$. Note that since $\mathbf{s}$ is not an RC-palindrome, either $\mathbf{s} \prec \mathbf{w}$ or $\tilde{\mathbf{s}} \prec \mathbf{w}$ holds, but not both. The same holds for $\mathbf{t}$. We proceed as follows: We try to construct a supersequence $\mathbf{u}$ of both $\mathbf{z}_1$ and $\mathbf{z}_2$ which is a subsequence of $\mathbf{w}$. We use a procedure analogous to the one used in Lemma 4.6, namely, we try to fit as many characters as possible from the beginning of $\mathbf{z}_2$ before the first character of $\mathbf{z}_1$, then we continue trying to fit the next characters of $\mathbf{z}_2$ between the first and the second character of $\mathbf{z}_1$ and so on. If we succeed, then we know that $\mathbf{w}$ is an interleaving of $\mathbf{s}$ and $\mathbf{t}$, otherwise, it is one of $\mathbf{s}$ and $\tilde{\mathbf{t}}$. $\square$

Now let us write $\mathbf{w}$, the string to be reconstructed, as

17

$$\mathbf{w} = \mathbf{u}_0\mathbf{v}_1\mathbf{u}_1\mathbf{v}_2\mathbf{u}_2\ldots\mathbf{v}_{r-1}\mathbf{u}_{r-1}\mathbf{v}_r\mathbf{u}_r, \tag{8}$$

where $\mathbf{v}_1\ldots\mathbf{v}_r = \mathbf{s}$, $\mathbf{u}_0$ and $\mathbf{u}_r$ are possibly empty, and $\mathbf{u}_0\mathbf{u}_1\ldots\mathbf{u}_r = \mathbf{t}$ or $\mathbf{u}_0\mathbf{u}_1\ldots\mathbf{u}_r = \tilde{\mathbf{t}}$. We will now establish some simple facts about the size of the $\mathbf{u}_i$'s. Let us recall once more that $\mathbf{s} \equiv_{\mathrm{RC}} \mathbf{w}_{|S}$ and $\mathbf{t} \equiv_{\mathrm{RC}} \mathbf{w}_{|S^c}$, and $|\mathbf{s}| \leq |\mathbf{t}|$. Further, $T = \lceil\frac{2}{3}(n+1)\rceil$ where $n = |\mathbf{w}|$.

Lemma 6.5. *Let $|\mathbf{s}| \leq |\mathbf{t}|$. Let $I = \{i \mid |\mathbf{s}| + |\mathbf{u}_i| \geq T\}$. Then $|I| \leq 2$. Moreover, if $|I| = 2$, then $|\mathbf{t}| \leq T - 2$.*

Proof. Let $q = |I|$. For all $i \in I$, $|\mathbf{u}_i| \geq T - |\mathbf{s}| \geq \frac{2}{3}n + \frac{2}{3} - |\mathbf{s}|$. This implies

$$n - |\mathbf{s}| \geq |\mathbf{t}| \geq \sum_{i \in I}|\mathbf{u}_i| \geq q(T - |\mathbf{s}|) = \frac{2q}{3}n + \frac{2q}{3} - q|\mathbf{s}|. \tag{9}$$

Thus, for any $q \geq 3$, we have $|\mathbf{s}| \geq \frac{2q-3}{3q-3}n + \frac{2q}{3q-3} > n/2$, a contradiction. Moreover, if $|I| = 2$, then (9) implies $|\mathbf{s}| \geq \frac{n}{3} + \frac{4}{3}$, and thus $|\mathbf{t}| = n - |\mathbf{s}| \leq n - \frac{n}{3} - \frac{4}{3} = \frac{2}{3}n - \frac{4}{3} \leq T - 2$. $\square$

The next lemma is proved analogously:

Lemma 6.6. *Let $|\mathbf{s}| \leq \frac{n}{3} + 1$ and $I = \{i \mid |\mathbf{s}| + |\mathbf{u}_i| \geq T\}$. Then $|I| \leq 1$.*

Proof. Suppose that there exist $i \neq j$ s.t. $|\mathbf{s}| + |\mathbf{u}_i| \geq T$ and $|\mathbf{s}| + |\mathbf{u}_j| \geq T$. This implies

$$n = |\mathbf{t}| + |\mathbf{s}| \geq |\mathbf{u}_i| + |\mathbf{u}_j| + |\mathbf{s}| \geq 2T - |\mathbf{s}| \geq \frac{4}{3}n + \frac{4}{3} - \frac{1}{3}n - 1 = n + \frac{1}{3},$$

a contradiction. $\square$

Thus, in general, for all but at most two i, we can reconstruct the substring $\mathbf{u}_i$ by fitting, one by one, its characters between $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$ until we receive a negative answer. Moreover, if $\mathbf{s}$ is short, i.e. at most $n/3 + 1$, then we can reconstruct all but one $\mathbf{u}_i$ in this way. (We will make this more precise below.)

We are now ready to present our reconstruction procedure starting with an S-projection $\mathbf{s}$ and the complementary S^c-projection $\mathbf{t}$. Recall that we are assuming that $|\mathbf{s}| \leq n/2 \leq |\mathbf{t}|$. We will proceed by distinguishing three cases, according to the length of $\mathbf{s}$: (1) $|\mathbf{s}| > \frac{n}{3} + 1$, (2) $|\mathbf{s}| \leq \frac{n}{3}$, and (3) $|\mathbf{s}| = \lfloor\frac{n}{3} + 1\rfloor$.

6.2   The case $|\mathbf{s}| > \frac{n}{3} + 1$

We will first describe how to determine the run structure of $\mathbf{s}$ and $\mathbf{t}$, and then how to determine their relative direction. If we know these two things, we also know $\mathbf{w}$, which is the unique string $\mathbf{s} \odot \mathbf{t}$ or $\mathbf{s} \odot \tilde{\mathbf{t}}$, where the interleaving happens according to the run structures determined.

Determining the run structure of s and t. Here the main observation is that $|\mathbf{t}| + 2 = n - |\mathbf{s}| + 2 < \frac{2n}{3} + 1 \leq T$. We limit ourselves to explicitly describing how to determine the run structure of $\mathbf{s}$. The analogous procedure can be applied to $\mathbf{t}$.

Let $\mathbf{s} = s_1\ldots s_{|\mathbf{s}|}$ and for technical reasons, we set $s_0 = s_{|\mathbf{s}|+1} = \epsilon$. For each $i = 1,\ldots,|\mathbf{s}| + 1$ and for each $x \in \Sigma$, we define the string

$$\mathbf{q}_i(x) = s_0 s_1 \ldots s_{i-1} x s_i \ldots s_{|\mathbf{s}|+1}. \tag{10}$$

18

Case 1. $\mathbf{s}$ is not an RC-palindrome. Then $\mathbf{s}$ fixes the direction of $\mathbf{w}$, and we ask, for every $i = 1, \ldots, |s| + 1$, and for every $x \in \mathrm{alph}(\mathbf{t})$ (those characters which actually occur in $\mathbf{t}$), the query $\mathbf{q}_i(x)$. If, for fixed i, there is an x such that we get a positive answer, then we know that there is a partition position between s_{i-1} and s_i in the run structure of $\mathbf{s}$ in $\mathbf{w}$. When all queries have been asked, we have determined the run structure of $\mathbf{s}$.

Case 2. $\mathbf{s}$ is an RC-palindrome. Again we ask for each $i = 1, \ldots, |\mathbf{s}| + 1$ and for each $x \in \mathrm{alph}(\mathbf{t})$, the query $\mathbf{q}_i(x)$. However, since $\mathbf{s} = \tilde{\mathbf{s}}$, a positive answer to $q_i(x)$ could indicate either that there is an x between s_{i-1} and s_i, or that there is a $\bar{x}$ between $s_{|\mathbf{s}|-i+1}$ and $s_{|\mathbf{s}|-i+2}$, or both. In order to distinguish these cases, we ask, for any i and x such that $q_i(x)$ answered positively, the query

$$\mathbf{q}_i'(x) = s_0 s_1 \ldots s_{i-1} x s_i \ldots s_{|\mathbf{s}|-i+1} \bar{x} s_{|\mathbf{s}|-i+2} \ldots s_{|\mathbf{s}|+1}. \tag{11}$$

If also $\mathbf{q}_i'(x)$ answers positively, then we know that both (s_{i-1}, s_i) and $(s_{|\mathbf{s}|-i+1}, s_{|\mathbf{s}|-i+2})$ are partition positions in the run structure of $\mathbf{s}$, i.e. they both contain a character from $\mathbf{t}$. So we continue with the next query $\mathbf{q}_{i+1}(x)$. Otherwise, we have found asymmetry in the run structure of $\mathbf{s}$, which we can exploit in the following: Let k be the minimum index for which there is a y s.t. $\mathbf{q}_k(y)$ answered positively and $\mathbf{q}_k'(y)$ answered negatively. Then $\mathbf{q}_k(y)$ fixes the direction of $\mathbf{w}$. Thus, we can finish the construction of the run structure of $\mathbf{s}$ by asking for each $j > k$ and for each x the query

$$\mathbf{q}_j''(x) = s_0 \ldots s_{k-1} y s_k \ldots s_{j-1} x s_j \ldots s_{|\mathbf{s}|+1}. \tag{12}$$

Since $\mathbf{q}_j''(x)$ is a supersequence of $\mathbf{q}_k(y)$, it fixes the direction of $\mathbf{w}$. As before, every time we get a positive answer we record the new partition position and we increment j.

Notice that all queries we have used have size at most $|\mathbf{s}| + 2$. Thus, the analogous procedure is also feasible for determining the run structure of $\mathbf{t}$, since $\mathbf{t}$ has size at most $T - 2$. The number of queries is altogether at most $|\Sigma||\mathbf{s}| + |\Sigma||\mathbf{t}| = O(|\Sigma|n)$.

Determining the relative direction of s and t. Assume now we have already determined the run structure of both $\mathbf{s}$ and $\mathbf{t}$.

Case 1. At least one of $\mathbf{s}$ and $\mathbf{t}$ is both an RC-palindrome and has symmetric run structure. Then, by Lemma 6.3, there is only one possibility of interleaving $\mathbf{s}$ and $\mathbf{t}$, so we are done.

Case 2. Neither $\mathbf{s}$ nor $\mathbf{t}$ are RC-palindromes. Then, by Lemma 6.4, we can determine their relative direction using $O(n)$ many queries of length at most T.

Case 3. $\mathbf{s}$ is either not an RC palindrome, or it is an RC-palindrome and its run structure is asymmetric. Then we can obtain a sequence $\mathbf{s}'$ which fixes the direction of $\mathbf{w}$: Define

$$\mathbf{s}' = \begin{cases} \mathbf{s}, & \text{if } \mathbf{s} \text{ is not RC-palindrome} \\ \mathbf{q}_k(y), & \text{otherwise,} \end{cases} \tag{13}$$

where $\mathbf{q}_k(y)$ is defined as before, i.e. $\mathbf{q}_k(y)$ is the string defined in (10), for the minimum k for which $\mathbf{q}_k'(y)$ from (11) answered negatively.

Case 3a. If $\mathbf{t}$ is not an RC-palindrome, since $\mathbf{s}'$ is also not an RC-palindrome, by Lemma 6.4 we can find their relative direction using $O(n)$ queries.

Case 3b. Finally, assume that $\mathbf{t}$ is an RC-palindrome and has asymmetric run structure. Let $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r$ be the partition of $\mathbf{t}$. Then there exists an index i such that $|\mathbf{u}_i| \neq |\mathbf{u}_{r-i+1}|$.

19

If at least one of $\mathbf{u}_i$ and $\mathbf{u}_{r-i+1}$ is "small", in the sense that $\min\{|\mathbf{u}_i|, |\mathbf{u}_{r+1-i}|\} + |\mathbf{s}'| < T$, then we can determine the direction of $\mathbf{t}$ with respect to $\mathbf{s}'$ as follows. Let $|\mathbf{u}_i| = m$, $|\mathbf{u}_{r+1-i}| = m'$, and w.l.o.g. $m > m'$. Let $\mathbf{u}_i = u_1 \ldots u_m$ and k be the position such that, if $\mathbf{t} \prec \mathbf{w}$ then the run $\mathbf{u}_i$ must appear in $\mathbf{w}$ immediately before the occurrence of the kth character of $\mathbf{s}'$. This k can be determined from the run structures, under the assumption $\mathbf{t} \prec \mathbf{w}$. We can now verify whether $\mathbf{t} \prec \mathbf{w}$, by asking the query

$$s'_1 \ldots s'_{k-1} u_1 \ldots u_{m'+1} s'_k \ldots s'_{|\mathbf{s}'|}, \tag{14}$$

where $\mathbf{s}' = s_1 \ldots s_{|\mathbf{s}'|}$. Clearly, this query answers yes if and only if $\mathbf{t} \prec \mathbf{w}$, otherwise we have $\tilde{\mathbf{t}} \prec \mathbf{w}$.

Else, for all i such that $|\mathbf{u}_i| \neq |\mathbf{u}_{r-i+1}|$, it holds that both runs are "big", i.e., $\min\{|\mathbf{u}_i|, |\mathbf{u}_{r-i+1}|\} + |\mathbf{s}'| \geq T$. Then, by Lemma 6.5, there must be exactly one such i. Moreover, $\mathbf{t}$ must consist of only these two runs, since otherwise, there would be at least another asymmetric pair of runs, which would have to be small. Let $\mathbf{u}_1, \mathbf{u}_2$ be the partition of $\mathbf{t}$, and let $\mathbf{u}_1 = u_1 \ldots u_m$ be the larger of the two runs, $|\mathbf{u}_2| = m' < m$. By Lemma 6.3, if $b_L(\mathbf{t}) \neq b_R(\mathbf{t})$, then the relative direction of $\mathbf{s}$ and $\mathbf{t}$ is fixed by their run structures. We are left with two other possibilites: (i) $b_L(\mathbf{t}) = b_R(\mathbf{t}) = 0$, or (ii) $b_L(\mathbf{t}) = b_R(\mathbf{t}) = 1$.

In case (i) we have that all of $\mathbf{s}$ appears between $\mathbf{u}_1$ and $\mathbf{u}_2$. It follows that $\mathbf{s} \neq \tilde{\mathbf{s}}$, since otherwise, $\mathbf{s}$ would be an RC-palindrome with symmetric run structure, a case we dealt with earlier. Since $\mathbf{s} \neq \tilde{\mathbf{s}}$, we can compute in time linear in $|\mathbf{s}|$ (see appendix) a sequence $\mathbf{z} \prec \mathbf{s}$, such that $\mathbf{z} \not\prec \tilde{\mathbf{s}}$. Consider now the query $u_1 \ldots u_{m'+1}\mathbf{z}$. This query answers positively if and only if $\mathbf{t} \prec \mathbf{w}$, hence allows to determine the direction of $\mathbf{t}$, and to complete the reconstruction. Notice that the size of the query is at most $|\mathbf{t}|/2 + 1 + |\mathbf{s}|/2 + 1 = n/2 + 2 \leq T$, for any $n \geq 8$.

In case (ii) we have $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ as partition of $\mathbf{s}$. It follows that $\mathbf{v}_2 \neq \tilde{\mathbf{v}}_2$ or $\mathbf{v}_1 \neq \tilde{\mathbf{v}}_3$, since otherwise, $\mathbf{s}$ would be RC-palindrome with symmetric run structure. In the former case we ask the query $u_1 \ldots u_{m'+1}\mathbf{z}$, where $\mathbf{z}$ is a subsequence of size $\leq |\mathbf{v}_2|/2 + 1$ distinguishing between $\mathbf{v}_2$ and $\tilde{\mathbf{v}}_2$. This query answers positively if and only if $\mathbf{t} \prec \mathbf{w}$, because $\mathbf{z}$ fixes the direction of $\mathbf{w}$. The query has length $\leq |\mathbf{t}|/2 + 1 + (|\mathbf{s}| - 2)/2 + 1 \leq n/2 + 1 \leq T$ for any $n \geq 2$.

Finally, assume that $\mathbf{v}_2 = \tilde{\mathbf{v}}_2$ and $\mathbf{v}_1 \neq \tilde{\mathbf{v}}_3$. Let $|\mathbf{v}_1| \geq |\mathbf{v}_3|$. Since $\mathbf{v}_1 \neq \tilde{\mathbf{v}}_3$, there exists a sequence $\mathbf{v}'$ of length at most $|\mathbf{v}_3| + 1$ such that $\mathbf{v}' \prec \mathbf{v}_1$ and $\mathbf{v}' \not\prec \tilde{\mathbf{v}}_3$. We ask the query $\mathbf{v}'u_1 \ldots u_{m'+1}xy$ where x is a character from $\mathbf{v}_2$ and y a character form $\mathbf{u}_2$. This query has length at most $|\mathbf{t}|/2 + 1 + (|\mathbf{s}| - 1)/2 + 1 \leq n/2 + 3/2 \leq T$ for any $n \geq 5$. Again, this query answers positively if and only if $\mathbf{t} \prec \mathbf{w}$.

In conclusion, also in Case 3b, a constant number of additional queries allows to determine the relative direction of $\mathbf{s}$ and $\mathbf{t}$ and hence to complete the reconstruction. In total we have use $O(|\Sigma|n)$ queries.

6.3 The case $|\mathbf{s}| \leq \frac{n}{3}$

Recall that we have $\mathbf{w} = \mathbf{u}_0\mathbf{v}_1\mathbf{u}_1 \ldots \mathbf{u}_{r-1}\mathbf{v}_r\mathbf{u}_r$, where $\mathbf{u}_0$ and $\mathbf{u}_r$ are possibly empty strings. By proceeding as in the previous section, using the fact that $|\mathbf{s}| + 2 \leq T$, we can find the run structure of $\mathbf{s}$, and in particular, the partition of $\mathbf{s}$ as $\mathbf{v}_1, \ldots, \mathbf{v}_r$.

Case 1. Assume first that $\mathbf{s}$ is not an RC-palindrome with symmetric run structure. Then, like in Section 6.2, we can define a supersequence $\mathbf{s}'$ of $\mathbf{s}$ which fixes the direction of $\mathbf{w}$: we have $\mathbf{s}' = \mathbf{s}$ if $\mathbf{s}$ is not an RC-palindrome; otherwise, let i be the minimum index such that $\mathbf{v}_i \neq \tilde{\mathbf{v}}_{r+1-i}$, and

x be a character from $\mathbf{u}_i$, which we find when we discover the run structure of $\mathbf{s}$. Then we have $\mathbf{s}' = \mathbf{v}_1 \ldots \mathbf{v}_i x \mathbf{v}_{i+1} \ldots \mathbf{v}_r$.

If $\mathbf{t}$ is not an RC-palindrome, then, using Lemma 6.4, with $O(n)$ additional queries, we can find the relative direction of $\mathbf{t}$ and $\mathbf{s}'$. If $\mathbf{t}$ is RC-palindrome, then it is indifferent, since $\tilde{\mathbf{t}} = \mathbf{t}$. Let us assume, w.l.o.g., that $\mathbf{t} \prec \mathbf{w}$, i.e., it appears in $\mathbf{w}$ in the same direction of $\mathbf{s}'$, hence, $\mathbf{t} = \mathbf{u}_0 \mathbf{u}_1 \ldots \mathbf{u}_r$. Then we are left with the problem of determining the partition of $\mathbf{t}$.

In order to reconstruct the partition, we proceed as follows. First, note that by Lemma 6.6, we know that there is at most one j such that $|\mathbf{u}_j| + |\mathbf{s}'| \geq T$. Suppose we have determined $\mathbf{u}_0, \ldots, \mathbf{u}_{i-1}$ and we know $\mathbf{u}_0 \mathbf{u}_1 \ldots \mathbf{u}_{i-1} = t_1 \ldots t_j$. In order to reconstruct $\mathbf{u}_i$, for $k = 1, 2 \ldots$, we ask the query $\mathbf{q}_k$ defined by inserting in $\mathbf{s}'$ immediately after $\mathbf{v}_i$ the characters $t_{j+1} \ldots t_{j+k}$. We continue until either we receive a negative answer or we have $|\mathbf{s}'| + k = T$. In the former case we can conclude that $\mathbf{u}_i = t_{j+1} \ldots t_{j+k-1}$ and we can repeat the same procedure in order to determine $\mathbf{u}_{i+1}$. In case we stopped because $|\mathbf{s}'| + k = T$, we have discovered that $\mathbf{u}_i$ is the only *long* run in $\mathbf{t}$, i.e., such that $|\mathbf{s}'| + |\mathbf{u}_i| \geq T$. Then we can continue reconstructing $\mathbf{t}$ from the back, i.e., from $\mathbf{u}_r$ using the obvious analogous procedure. Since there is only one long run, we will be able to reconstruct all the remaining runs, $\mathbf{u}_r, \ldots, \mathbf{u}_{i+1}$. Then, once the boundary of $\mathbf{u}_{i+1}$ is known, we have also determined $\mathbf{u}_i$ and the reconstruction of $\mathbf{w}$ is complete.

Since we need a constant number of queries to increase by at least one character our knowledge about the partition of $\mathbf{t}$, it follows that we can complete this case with $O(n)$ additional queries.

Case 2. Assume now that $\mathbf{s}$ is RC-palindrome and its run structure is symmetric, i.e. for each i we have $\mathbf{v}_i = \tilde{\mathbf{v}}_{r+1-i}$. Then we know by Lemma 6.3 that $\mathbf{s} \odot \mathbf{t} \equiv_{\mathrm{RC}} \mathbf{s} \odot \tilde{\mathbf{t}}$. However, we still need to determine the run structure of $\mathbf{t}$ in $\mathbf{w}$. We do this iteratively. By showing that this can be achieved with at most $O(|\Sigma||t|)$ additional queries, we will reach the desired conclusion also in this case, about the linear query complexity of the reconstruction.

Suppose that with $O(|\mathbf{u}_0| + \ldots |\mathbf{u}_{i-1}| + |\mathbf{u}_{r-i+1}| + \ldots |\mathbf{u}_r|)$ queries we have determined $\mathbf{u}_0, \ldots, \mathbf{u}_{i-1}$ and $\mathbf{u}_{r-i+1}, \ldots, \mathbf{u}_r$ and we have $\mathbf{u}_j = \tilde{\mathbf{u}}_{r-j}$, for each $j = 0, \ldots, i-1$. Let us also define $\mathbf{z}_L = \mathbf{u}_0 \mathbf{v}_1 \mathbf{u}_1 \mathbf{v}_2 \ldots \mathbf{v}_{i-1} \mathbf{u}_{i-1} \mathbf{v}_i$ and $\mathbf{z}_R = \mathbf{v}_{r+1-i} \mathbf{u}_{r+1-i} \ldots \mathbf{v}_r \mathbf{u}_r$. Then, under the standing hypotheses, we have $\mathbf{z}_L = \tilde{\mathbf{z}}_R$. Finally, let $\ell = |\mathbf{u}_0 \ldots \mathbf{u}_{i-1}|$.

For reconstructing $\mathbf{u}_i$ and $\mathbf{u}_{r-i}$ we use queries made of the entire sequence $\mathbf{s}$ and some characters in gaps corresponding to the runs $\mathbf{u}_i$ and $\mathbf{u}_{r-i}$: Let

$$
\begin{aligned}
\mathbf{q}_L(k) &= \mathbf{v}_1 \ldots \mathbf{v}_i t_{\ell+1} \ldots t_{\ell+k} \mathbf{v}_{i+1} \ldots \mathbf{v}_r, \\
\mathbf{q}_{R1}(k) &= \mathbf{v}_1 \ldots \mathbf{v}_i t_{\ell+1} \ldots t_{\ell+k} \mathbf{v}_{i+1} \ldots \mathbf{v}_{r-i} t_{|\mathbf{t}|+1-\ell-k} \ldots t_{|\mathbf{t}|-\ell} \mathbf{v}_{r-i+1} \ldots \mathbf{v}_r, \text{ and} \\
\mathbf{q}_{R2}(k) &= \mathbf{v}_1 \ldots \mathbf{v}_i t_{\ell+1} \ldots t_{\ell+k-1} \mathbf{v}_{i+1} \ldots \mathbf{v}_{r-i} t_{|\mathbf{t}|+1-\ell-k} \ldots t_{|\mathbf{t}|-\ell} \mathbf{v}_{r-i+1} \ldots \mathbf{v}_r.
\end{aligned}
$$

We proceed as follows. For each $k = 1, 2, \ldots$, we ask query $\mathbf{q}_L(k)$. If it answers positively, we ask $\mathbf{q}_{R1}(k)$. If it answers negatively, we ask $\mathbf{q}_{R2}(k)$. We continue until we reach a k for which one of the following occurs:

(i) both queries answer negatively;

(ii) exactly one query answers negatively;

(iii) $\mathbf{q}_L(k)$ answers negatively and $\mathbf{q}_{R2}(k)$ exceeds the threshold T;

(iv) $\mathbf{q}_L(k)$ answers positively and $\mathbf{q}_{R1}(k)$ exceeds the threshold T.

21

We shall now analyze the four cases separately.

Case (i). We have determined $\mathbf{u}_i = t_{\ell+1} \dots t_{\ell+k-1}$ and $\mathbf{u}_{r-i} = t_{|\mathbf{t}|+2-\ell-k} \dots t_{|\mathbf{t}|-\ell}$ with $O(|\mathbf{u}_i| + |\mathbf{u}_{r-i}|)$ queries. If $\mathbf{u}_{r-i} = \tilde{\mathbf{u}}_i$, then we can continue with the reconstruction of $\mathbf{u}_{i+1}, \mathbf{u}_{r-i-1}$ as above. Else, we have that $\mathbf{q}_L(k)$ fixes the direction of $\mathbf{w}$, and we proceed as in Case (ii) below.

Case (ii). We can assume w.l.o.g. that $\mathbf{q}_L(k)$ answered positively. (The other case can be handled symmetrically, by switching the role of $\mathbf{u}_i$ and $\mathbf{u}_{r-i}$.) We have now determined $\mathbf{u}_{r-i}$, and we know that the string $\mathbf{u}'_i = t_{\ell+1} \dots t_{\ell+k}$ is a prefix of $\mathbf{u}_i$. Moreover, because of the different answers we are sure that—even if $\mathbf{t}$ is RC-palindrome—the sequence $\mathbf{q}_L(k)$ fixes the direction of $\mathbf{w}$.

We shall now attempt to reconstruct the remaining runs $\mathbf{u}_{r-i-1}, \mathbf{u}_{r-i-2}, \dots$, in this order, i.e. from the back, by using queries which are supersequences of $\mathbf{q}_L(k)$.

For some $j \geq 0$, suppose we have already reconstructed $\mathbf{u}_{r-i-1}, \dots, \mathbf{u}_{r-i-j}$ and ℓ' is such that the sequence $\mathbf{u}_{r-i-j}\mathbf{u}_{r-i-j+1} \dots \mathbf{u}_r = t_{\ell'} \dots t_{|\mathbf{t}|}$. In order to reconstruct $\mathbf{u}_{r-i-j-1}$ we ask queries

$$\mathbf{v}_1 \dots \mathbf{v}_i \mathbf{u}'_i \mathbf{v}_{i+1} \dots \mathbf{v}_{r-i-j-1} t_{\ell'-\lambda} \dots t_{\ell'-1} \mathbf{v}_{r-i-j} \dots \mathbf{v}_r \tag{15}$$

for $\lambda = 1, 2, \dots$, until we either receive a negative answer or the query cannot be enlarged further because the threshold T has been reached. In the former case, we have identified the run and we can continue with $\mathbf{u}_{r-i-j-2}$.

If otherwise, we stop because of the threshold T, we have

$$|\mathbf{s}| + |\mathbf{u}'_i| + |\mathbf{u}_{r-i-j-1}| \geq T. \tag{16}$$

As we will see, this means that any query containing all of $\mathbf{s}$, and $\mathbf{u}'_i$, and any yet unreconstructed gap $\mathbf{u}_{j'}$ is feasible. It even holds that all of the unreconstructed part of $\mathbf{t}$ (excluding $\mathbf{u}_{r-i-k-1}$), together with $\mathbf{s}$ and $\mathbf{u}'_i$, is not larger than T:

$$\begin{aligned}
|\text{ unreconstructed part of } \mathbf{t} | &\leq |\mathbf{t}| - |\mathbf{u}'_i| - |\mathbf{u}_{r-i}| - |\mathbf{u}_{r-i-j-1}| \\
&= n - |\mathbf{s}| - (|\mathbf{u}'_i| + |\mathbf{u}_{r-i-j-1}|) - |\mathbf{u}'_i| + 1 \\
&\leq n - |\mathbf{s}| + |\mathbf{s}| - T - |\mathbf{u}'_i| + 1 \leq n/3 + 1/3 - |\mathbf{u}'_i|,
\end{aligned} \tag{17}$$

where we used that $|\mathbf{u}'_i| = |\mathbf{u}_{r-i}| + 1$ and $-(|\mathbf{u}'_i| + |\mathbf{u}_{r-i-j-1}|) \leq |\mathbf{s}| - T$, by (16). Therefore, for the size of the remaining part of $\mathbf{t}$ we have

$$|\text{ unreconstructed part of } \mathbf{t} | = |\mathbf{u}_i| - |\mathbf{u}'_i| + |\mathbf{u}_{i+1}| + \dots + |\mathbf{u}_{r-i-j-2}| \leq n/3 + 1/3 - |\mathbf{u}'_i|. \tag{18}$$

This implies that we can reconstruct it from left to right using the following queries, which are supersequences of the direction fixing sequence $\mathbf{q}_L(k)$. Recall that $|\mathbf{u}'_i| = k$. Then asking the queries

$$\mathbf{v}_1 \dots \mathbf{v}_i \mathbf{u}'_i t_{\ell+1} \dots t_{\ell+\lambda} \mathbf{v}_{i+1} \dots \mathbf{v}_r \tag{19}$$

for $\lambda = k+1, k+2, \dots$, until we receive a negative answer, will determine $\mathbf{u}_i$. Let λ_1 be the minimum value of λ for which the answer is negative. Then, we ask queries

$$\mathbf{v}_1 \dots \mathbf{v}_i \mathbf{u}'_i \mathbf{v}_{i+1} t_{\ell+\lambda_1+1} \dots t_{\ell+\lambda_1+\lambda} \mathbf{v}_{i+1} \dots \mathbf{v}_r \tag{20}$$

for $\lambda = 1, 2, \dots$, until we receive a negative answer, in order to reconstruct $\mathbf{u}_{i+1}$, and so on. Once we have reconstructed $\mathbf{u}_{r-i-j-2}$, we will be finished since also $\mathbf{u}_{r-i-j-1}$ will be determined.

From the above analysis, it is easy to see that the number of queries asked to complete the reconstruction in this case is $O(|t|)$, as desired.

Case (iii). Ask the query

$$\mathbf{q}_{R3}(k) = \mathbf{v}_1 \ldots \mathbf{v}_{r-i} t_{|\mathbf{t}|+1-\ell-k} \ldots t_{|\mathbf{t}|-\ell} \mathbf{v}_{r-i+1} \ldots \mathbf{v}_r.$$

Since we know that $\mathbf{q}_L(k)$ is not an RC-subsequence of $\mathbf{w}$, we can be sure that $\mathbf{q}_{R3}(k)$ answers positively if and only if $t_{|\mathbf{t}|+1-\ell-k} \ldots t_{|\mathbf{t}|-\ell}$ is a suffix of $\mathbf{u}_{r-i}$. If the answer is negative, we are back in Case (i); if the answer is positive, we are in Case (ii).

Case (iv). This means that (assuming, w.l.o.g. that $|\mathbf{u}_i| \geq |\mathbf{u}_{r-i}|$) we have

$$|\mathbf{u}_i| + |\mathbf{u}_{r-i}| \geq 2|\mathbf{u}_{r-i}| \geq T - |\mathbf{s}| \geq n/3 + 2/3. \tag{21}$$

As a consequence we have that $|\mathbf{t}| + |\mathbf{s}| - (|\mathbf{u}_i| + |\mathbf{u}_{r-i}|) \leq T$.

The last inequality implies that we can reconstruct all the runs $\mathbf{u}_{i+1}, \ldots, \mathbf{u}_{r-i-1}$ by exploiting the unbounded search procedure, even ignoring what we know about $\mathbf{t}$. Precisely, suppose we have reconstructed $\mathbf{u}_{i+1}, \ldots, \mathbf{u}_{i+j}$ for some $j \geq 0$, then we can ask queries

$$\mathbf{q}_{j+1}(\mathbf{p}) = \mathbf{v}_1 \ldots \mathbf{v}_{i+1} \mathbf{u}_{i+1} \mathbf{v}_{i+2} \mathbf{u}_{i+2} \ldots \mathbf{v}_{i+j+1} \mathbf{p} \mathbf{v}_{i+j+2} \ldots \mathbf{v}_r,$$

where $\mathbf{p}$ represents a query we would use in an unbounded query reconstruction of the sole sequence $\mathbf{u}_{i+j+1}$.

Let $\mathbf{z}_C = \mathbf{v}_{i+1} \mathbf{u}_{i+1} \mathbf{v}_{i+2} \ldots \mathbf{u}_{r-i-1} \mathbf{v}_{r-i}$. The above procedure allows us to reconstruct $\mathbf{z}_C$ with $O(|\Sigma||\mathbf{z}_C|)$ queries and to have $\mathbf{z}_L \mathbf{z}_C \mathbf{z}_R \prec_{\mathrm{RC}} \mathbf{w}$. We still need to determine the direction of $\mathbf{z}_C$. In fact we shall show that we can reconstruct the larger of $\mathbf{u}_i$ and $\mathbf{u}_{r-i}$.

Under the assumption that $|\mathbf{u}_i| \geq |\mathbf{u}_{r-i}|$, suppose now that queries based only on $\mathbf{s}$ and subsequences of $\mathbf{u}_i$ (namely the queries $\mathbf{q}_L(k)$ defined above) become too long to be performed because of

$$|\mathbf{u}_i| + |\mathbf{s}| \geq T. \tag{22}$$

Then let us consider the query $\mathbf{q}$ which is made of one character each from the runs of $\mathbf{s}$ and $\mathbf{t}$ and of the whole run $\mathbf{u}_i$. More precisely, we observe that if we have $\mathbf{u}_0 = \mathbf{u}_r = \epsilon$, then we do not use characters from $\mathbf{v}_1$ and $\mathbf{v}_r$. Precisely, if $\mathbf{u}_0 = \mathbf{u}_r = \epsilon$, then we use query

$$\mathbf{q}(k) = b_1 a_2 b_2 a_3 \ldots b_{i-1} a_i t_{\ell+1} \ldots t_{\ell+k} a_{i+1} b_{i+1} \ldots a_{r-1} b_{r-1},$$

where a_i is a character from $\mathbf{v}_i$ and b_i is a character from $\mathbf{u}_i$. Alternatively, if $\mathbf{u}_0 = \mathbf{u}_r \neq \epsilon$, then we use $\mathbf{q}(k) = b_0 a_1 b_1 a_1 \ldots b_{i-1} a_i t_{\ell+1} \ldots t_{\ell+k} a_{i+1} b_{i+1} \ldots a_{r-1} b_{r-1}$. The length of $\mathbf{q}(k)$, for $k = |\mathbf{u}_i|$, is within the threshold T, since

$$\begin{aligned} |\mathbf{q}(|\mathbf{u}_i|)| &\leq 2(|\mathbf{t}| - |\mathbf{u}_i| - |\mathbf{u}_{r-i}| + 2) - 1 + |\mathbf{u}_i| - 1 = 2|\mathbf{t}| - |\mathbf{u}_i| - 2|\mathbf{u}_{r-i}| + 2 \\ &\leq 2n - 2|\mathbf{s}| + |\mathbf{s}| - T + |\mathbf{s}| - T + 2 \leq \frac{2n}{3} + \frac{2}{3}, \end{aligned}$$

where for the first inequality we used the fact that the number of runs considered is maximum when all runs of $\mathbf{t}$ comprise one character except for $\mathbf{u}_i$ and $\mathbf{u}_{r-i}$. For the second inequality we used $|\mathbf{t}| = n - |\mathbf{s}|$ and (22)-(21) for bounding $-|\mathbf{u}_i|$ and $-2|\mathbf{u}_{r-i}|$ with $|\mathbf{s}| - T$.

It follows that we can use queries $\mathbf{q}(k)$ to reconstruct $\mathbf{u}_i$. This, together with $\mathbf{z}_C$, $\mathbf{t}$ and $\mathbf{s}$, allows complete reconstruction of $\mathbf{w}$. The number of additional queries is clearly $O(|\mathbf{u}_i|)$.

23

6.4 The case $|\mathbf{s}| = \lfloor \frac{n}{3} + 1 \rfloor$

We only sketch this case. We can proceed as in Section 6.3, except for the case when $\mathbf{s}$ is RC-palindrome and its run structure is asymmetric. Here, in contrast to Section 6.3, we have that $|\mathbf{s}'| = \lfloor \frac{n}{3} + 2 \rfloor$ and therefore, we cannot use Lemma 6.6 to conclude that there can be at most two big gaps.

However, an analogous argument as in the proof of Lemma 6.6 shows that for $|\mathbf{s}'| = \lfloor \frac{n}{3} + 2 \rfloor$ there can be two big gaps but these must then be the only runs of $\mathbf{t}$, i.e., in this case, $\mathbf{t}$ consists only of the characters which are in these two big gaps. In order to find the size of these two gaps, it is enough to ask queries including the whole $\mathbf{t}$ and one character from $\mathbf{s}$ to mark the separation between the two gaps. Since $|\mathbf{s}| = \lfloor \frac{n}{3} + 1 \rfloor$, we have that $|\mathbf{t}| + 1 = n - |\mathbf{s}| + 1 = \lceil \frac{2n}{3} \rceil \leq T$, whence the queries are feasible. The number of queries used is $O(|\mathbf{t}|)$.

Once we know the size of the gaps, the rest of the argument in Section 6.3 can be applied as before. Thus, we can complete the reconstruction also for $|\mathbf{s}| = \lfloor \frac{n}{3} + 1 \rfloor$, with the desired number of bounded queries.

7 Conclusion

In this paper, we gave reconstruction algorithms from subsequences in the RC-equivalence model, i.e. where strings cannot be distinguished from their reverse complements. We treated the problem separately for strings over binary alphabets, i.e. alphabets consisting only of one complement pair, and for strings over general finite paired alphabets. Our algorithms use queries of the form "Is $\mathbf{u}$ an RC-subsequence of $\mathbf{w}$?", where $\mathbf{w}$ is the string to be reconstructed, with $|\mathbf{w}| = n$, and $\mathbf{u}$ is a string of length at most $\lceil \frac{2n+1}{3} \rceil$. This is optimal w.r.t. the length of the queries, as has been shown in [6]. Our algorithm for the binary case is also asymptotically optimal w.r.t. the number of queries, as we showed in Section 2, while the one for the general case is off only by a factor of $O(|\Sigma|/\log|\Sigma|)$.

We note that our focus in this paper has been to provide algorithms which use $O(n \log |\Sigma|)$ resp. $O(|\Sigma|n)$ many queries, while keeping the length of queries optimal. We have not attempted to minimize the number of queries within the asymptotics.

The RC-equivalence model can be viewed as a special case of erroneous information, where the answers to subsequence queries could be either about the query string or its reverse complement. It is also a special case of a group action on Σ^*, the set of finite strings over Σ. The search in Σ^n is substituted by a search in $\Sigma^n/\sim$, where $\sim$ is the equivalence induced by the group action. String reconstruction in this latter setting is an interesting and more general question, and we plan to revisit it at a later point.

Acknowledgements

References

[1] C. Bercoff. Uniform tag systems for paperfolding sequences. *Discrete Applied Mathematics*, 77(2):119–138, 1997.

[2] A. Carpi and A. de Luca. Words and special factors. *Theoretical Computer Science*, 259(1-2):145–182, 2001.

[3] F. Cicalese, P. L. Erdős, and Zs. Lipták. Efficient reconstruction of RC-equivalent strings. In *Proc. of the 21st International Workshop on Combinatorial Algorithms (IWOCA 2010)*, volume 6460 of *LNCS*, pages 349–362. Springer, 2011.

[4] A. de Luca. On the combinatorics of finite words. *Theoretical Computer Science*, 218(1):13–39, 1999.

[5] M. Dudík and L. J. Schulman. Reconstruction from subsequences. *J. Comb. Theory, Ser. A*, 103(2):337–348, 2003.

[6] P. L. Erdős, P. Ligeti, P. Sziklai, and D. C. Torney. Subwords in reverse-complement order. *Annals of Combinatorics*, 10:415–430, 2006.

[7] G. Fici, F. Mignosi, A. Restivo, and M. Sciortino. Word assembly through minimal forbidden words. *Theoretical Computer Science*, 359(1-3):214–230, 2006.

[8] I. Krasikov and Y. Roditty. On a reconstruction problem for sequences,. *J. Comb. Theory, Ser. A*, 77(2):344–348, 1997.

[9] V. I. Levenshtein. Efficient reconstruction of sequences. *IEEE Transactions on Information Theory*, 47(1):2–22, 2001.

[10] P. Pevzner. *l*-tuple DNA sequencing: Computer analysis. *Journal of Biomolecular Structure and Dynamics*, 7:63–73, 1989.

[11] C. Piña and C. Uzcátegui. Reconstruction of a word from a multiset of its factors. *Theoretical Computer Science*, 400(1-3):70–83, 2008.

[12] F. P. Preparata. Sequencing-by-hybridization revisited: The analog-spectrum proposal. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 1(1):46–52, 2004.

[13] J. Sakarovitch and I. Simon. *Combinatorics on Words, by M. Lothaire*, chapter 6: Subwords. Cambridge University Press, 1983.

[14] I. Simon. Piecewise testable events. In *Automata Theory and Formal Languages*, pages 214–222. Springer, 1975.

[15] S. Skiena and G. Sundaram. Reconstructing strings from substrings. *Journal of Computational Biology*, 2(2):333–353, 1995.

[16] D. Tsur. Tight bounds for string reconstruction using substring queries. In *APPROX-RANDOM*, pages 448–459, 2005.

APPENDIX

A   How to construct distinguishing subsequences

Let $\mathbf{w}$ and $\mathbf{w}'$ be two strings of length n over a finite alphabet $\Sigma = \{a_1, \ldots, a_\sigma\}$, $\mathbf{w} \neq \mathbf{w}'$. We want to construct a sequence $\mathbf{u}$ of length at most $\lfloor n/2 \rfloor + 1$ which is subsequence of $\mathbf{w}$ and not of $\mathbf{w}'$, or vice versa. For a string $\mathbf{s} = s_1 \ldots s_n$ over Σ, let $pr_i(\mathbf{s}) = s_1 \ldots s_i$ be its prefix of length i. In addition, let $p(\mathbf{s}) = (|\mathbf{s}|_{a_1}, \ldots, |\mathbf{s}|_{a_\sigma})$ be the vector whose i'th component counts the number of occurrences in $\mathbf{w}$ of the character a_i.

Case 1. $p(\mathbf{w}) \neq p(\mathbf{w}')$. Then there are two characters a, b s.t. $|\mathbf{w}|_a < |\mathbf{w}'|_a$ and $|\mathbf{w}|_b > |\mathbf{w}'|_b$. W.l.o.g. let $|\mathbf{w}|_a \leq |\mathbf{w}'|_b$. Set $x = |\mathbf{w}|_a$. Then for $\mathbf{u} = a^{x+1}$ we have $\mathbf{u} \not\prec \mathbf{w}$ but $\mathbf{u} \prec \mathbf{w}'$. Moreover, since $|\mathbf{w}|_a \leq |\mathbf{w}'|_b < |\mathbf{w}|_b$, we have that $x = |\mathbf{w}|_a \leq \lfloor n/2 \rfloor$, and thus $|\mathbf{u}| = x + 1 \leq \lfloor n/2 \rfloor + 1$.

Case 2. $p(\mathbf{w}) = p(\mathbf{w}')$. Then there is a smallest position i s.t. $w_i \neq w_i'$, and let $a = w_i, b = w_i'$. Let A be the number of a's and B the number of b's in $\mathbf{w}$ (and in $\mathbf{w}'$), and let k be the number of a's and ℓ be the number of b's in $pr_{i-1}(\mathbf{w})$ (i.e. $k = |pr_{i-1}(\mathbf{w})|_a, \ell = |pr_{i-1}(\mathbf{w})|_b$). Define

$$\mathbf{u} = a^{k+1}b^{B-\ell} \qquad \text{and} \qquad \mathbf{v} = b^{\ell+1}a^{A-k}. \tag{23}$$

Then $\mathbf{u} \prec \mathbf{w}$ and $\mathbf{u} \not\prec \mathbf{w}'$; while $\mathbf{v} \prec \mathbf{w}'$ and $\mathbf{v} \not\prec \mathbf{w}$. Moreover, either $\mathbf{u}$ or $\mathbf{v}$ is not larger than $\lfloor n/2 \rfloor + 1$: Assume that $|\mathbf{u}| > \lfloor n/2 \rfloor + 1$. Then $k + B - \ell > \lfloor n/2 \rfloor$, and thus $n - k - B + \ell \leq \lfloor n/2 \rfloor$. Therefore, $\lfloor n/2 \rfloor + 1 \geq n - k - B + \ell + 1 \geq A - k + \ell + 1 = |\mathbf{v}|$.