

Efficient Reconstruction of RC-Equivalent Strings

Ferdinando Cicalese¹, Péter L. Erdős², and Zsuzsanna Lipták³

¹ Dipartimento di Informatica ed Applicazioni, University of Salerno, Italy
cicalese@dia.unisa.it

² Alfréd Rényi Institute of Mathematics, Budapest, Hungary
elp@renyi.hu

³ AG Genominformatik, Technische Fakultät, Bielefeld University, Germany
zsuzsa@cebitec.uni-bielefeld.de

Abstract. In the reverse complement (RC) equivalence model, it is not possible to distinguish between a string and its reverse complement. We show that one can still reconstruct a binary string of length n , up to reverse complement, using a linear number of subsequence queries of bounded length. A simple information theoretic lower bound proves the number of queries to be tight. Our result is also optimal w.r.t. the bound on the query length given in [Erdős *et al.*, Ann. of Comb. 2006].

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. Because of the confusion in terminology in the literature, we want to give a precise definition right here: Given two strings \mathbf{s}, \mathbf{t} over Σ , $\mathbf{s} = s_1 \dots s_n$ and $\mathbf{t} = t_1 \dots t_m$, we say that \mathbf{t} is a *subsequence* (often called *subword*) of \mathbf{s} if there exist $1 \leq i_1 < i_2 < \dots < i_m \leq n$ such that $\mathbf{t} = s_{i_1} s_{i_2} \dots s_{i_m}$. It was shown by Simon in 1975 [12] 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 [11] can be easily adapted 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} over Σ , $\mathbf{s} = s_1 \dots s_n$, we define its *reverse complement* as $\tilde{\mathbf{s}} = \bar{s}_n \dots \bar{s}_1$. 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} . Erdős *et al.* showed in [4] that two strings \mathbf{s} and \mathbf{t} of length n are equal 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 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 both w.r.t. the length of the queries, and w.r.t. the information theoretic lower bound on the number of queries necessary for exact reconstruction. We also give a simple algorithm for arbitrary alphabets, adapted from a paper by Skiena and Sundaram [13], where the length of the queries is not bounded, using $O(n \log |\Sigma|)$ queries.

It should be noted that the problem differs considerably from the classical model. For example, consider the string $\mathbf{s} = \bar{a}\bar{b}a\bar{a}b$. Then aba is not a subsequence of \mathbf{s} , but it is an RC-subsequence, because $\bar{a}\bar{b}\bar{a}$ is a subsequence of \mathbf{s} .

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 Σ^n / \sim , where \sim is the equivalence induced by the group action.

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 [6]. Dudík and Schulman [3] 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 [7] investigates the maximal number of common subsequences of length k that two distinct subsequences 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 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 $\mathbf{t} = s_i \dots s_j$. 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, 1]. An algorithm for reconstruction was given by Fici *et al.* in [5], while the uniqueness bound for multisets of substrings was recently shown to be $\lfloor \frac{n}{2} \rfloor + 1$ by Piña and Uzcágetui [9].

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 [8]. Here, a large number of strings of a certain length are queried in parallel, using a DNA chip, and 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, [14, 10] for some more recent results.)

Due to space limitations, some proofs are deferred to the Appendix.

2 Preliminaries

By a paired alphabet we understand a finite set $\Sigma = \{a_1, \dots, a_{2\delta}\}$, for some integer $\delta \geq 1$, together with a non-identity involution operation $\bar{\cdot} : \Sigma \mapsto \Sigma$, which we call complement. Thus, for each $i = 1, \dots, 2\delta$, there is a $j \neq i$ such that $a_i = \bar{a}_j$. Notice that by definition, $\overline{\bar{a}_i} = a_i$, for each i .

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

Given two strings \mathbf{s}, \mathbf{t} over Σ , $\mathbf{s} = s_1 \dots s_n$, $\mathbf{t} = t_1 \dots t_m$, we say that \mathbf{t} is a *subsequence*⁴ of \mathbf{s} , denoted by $\mathbf{t} \prec \mathbf{s}$, if there exist $1 \leq i_1 < i_2 < \dots < i_m \leq n$ such that $\mathbf{t} = s_{i_1} s_{i_2} \dots s_{i_m}$. Further, we define \mathbf{t} to be an *RC-subsequence*, denoted $\mathbf{t} \prec_{\text{RC}} \mathbf{s}$ 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 $\tilde{\mathbf{t}} \prec \mathbf{s}$.

Example 1. Our motivating example is the alphabet of the 4 nucleotides (DNA) $\Sigma = \{A, C, G, T\}$ where (A, T) and (G, C) are complement pairs. Let $\mathbf{s} = \text{ACCGATTAC}$. Then $\tilde{\mathbf{s}} = \text{GTAATCGGT}$, $\text{GTTT} \not\prec \mathbf{s}$ but $\text{GTTT} \prec_{\text{RC}} \mathbf{s}$.

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 $\mathbf{t} \prec_{\text{RC}} \mathbf{s}$. 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 questions, i.e. if $T = n$.

Proposition 2 (Lower Bound). *Given a string \mathbf{s} of size n from an alphabet Σ . Any deterministic algorithm that identifies \mathbf{s} (up to reverse complement) by asking RC-subsequence queries needs at least $(n-1) \log |\Sigma|$ queries.*

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

⁴ In the literature, the term 'subword' is also common. However, 'subword' is also used to mean a contiguous subsequence. We avoid the term.

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 [13], originally developed for the classic case (where queries would answer *no* if the subsequence only appears in the reverse complement of the string).

Theorem 3. *There exists an algorithm for reconstructing a string using $\Theta(n \log |\Sigma|)$ RC-subsequence queries of unbounded length.*

Proof. (Sketch.) For the binary case $\Sigma = \{a, b\}$, we first find $A := \max\{|s|_a, |s|_b\}$ by asking queries a^χ for $\chi = 1, 2, 3, \dots$. Clearly, $A = \chi - 1$ for the first χ that gives a *no* answer. Now there are indices $0 \leq i_0, i_1, \dots, i_A$ s.t. $\mathbf{s} = b^{i_0} a b^{i_1} a \dots a b^{i_{A-1}} a b^{i_A}$. We find i_0 by asking $b a^A, b^2 a^A, b^3 a^A$, etc., then find i_1 by asking $b^{i_0} a b a^{A-1}, b^{i_0} a b^2 a^{A-1}$ etc. The total number of queries is at most $\frac{3}{2}n + 2$.

Now let $\Sigma = \{a_1, \bar{a}_1, \dots, a_\delta, \bar{a}_\delta\}$. For each complement pair a_i, \bar{a}_i , we first determine $\mathbf{s}_{|i}$, the longest subsequence of \mathbf{s} which consists only of a_i 's and \bar{a}_i 's. This can be done by using the algorithm for the binary case sketched above. Now we iteratively interleave the projections: first $\mathbf{s}_{|1}$ with $\mathbf{s}_{|2}$, yielding $\mathbf{s}_{|1,2}$, then $\mathbf{s}_{|1,2}$ with $\mathbf{s}_{|3,4}$ etc. Interleaving two strings \mathbf{u}, \mathbf{v} that only contain characters from different complement pairs can be done with $2(|\mathbf{u}| + |\mathbf{v}| + 1)$ queries using the same idea as for the binary case, with the following small alteration: Since either \mathbf{u} and \mathbf{v} , or \mathbf{u} and $\tilde{\mathbf{v}}$ have to be interleaved, we start with \mathbf{u} and \mathbf{v} , and if we get a contradictory answer at some point, then we start over with \mathbf{u} and $\tilde{\mathbf{v}}$ (hence the factor 2). So the total number of queries for interleaving the projections $\mathbf{s}_{|i}$ is at most $2n \log \delta + 2(\delta - 1)$. The number of queries of the first phase is at most $\sum_{i=1}^{\delta} (\frac{3}{2}A_i + 2)$, where $A_i = |s|_{a_i} + |s|_{\bar{a}_i}$, yielding $O(n \log |\Sigma|)$ questions in total, using the (natural) assumption that $|\Sigma| = O(n)$.

4 Bounded query size (binary alphabet)

We now turn to subsequence queries whose length is bounded by a threshold T . In the following, 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 set below $\lceil \frac{2}{3}n \rceil$. The proof can be found in the Appendix.

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

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 $\geq \lceil 2n/3 \rceil$.

Any string \mathbf{s} over Σ can be written uniquely in its runlength encoded form:

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

with x_1 and y_ρ possibly 0, and all other $x_i, y_i > 0$. The number of non-zero x_i, y_i is the number of runs of \mathbf{s} . 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, swap \mathbf{s} and $\tilde{\mathbf{s}}$. We will denote by ρ_a the number of a -runs, and by ρ_b the number of b -runs of \mathbf{s} . Note that both have value either ρ or $\rho - 1$. (We have $\rho_a = \rho_b = \rho - 1$ if and only if the string starts with a b and ends with an a .)

In this section we prove the main result of the paper, which is given in the following theorem:

Theorem 4. *There is an algorithm which reconstructs a binary string \mathbf{s} of length n using $O(n)$ many RC-subsequence queries of length at most $\lceil \frac{2}{3}(n+1) \rceil$.*

Notice that this is tight w.r.t. the lower bound of Fact 1 in all cases except where n is a multiple of 3. Even for these n , a gap of 1 unit is only necessary in the special case $A = \frac{2}{3}n$. In all other cases, our analysis resists the stricter bound of $T = \lceil \frac{2}{3}n \rceil$.

The proof of the theorem is by examining four cases separately. Recall that $A = |\mathbf{s}|_a, B = |\mathbf{s}|_b$, and $T = \frac{2}{3}(n+1)$. 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 5. *Let \mathbf{s} be a string of length at least 8 over $\{a, b\}$, $T = \lceil \frac{2}{3}(n+1) \rceil$, 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 .*

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 $a^T \prec_{\text{RC}} \mathbf{s}$, 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 6. 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).

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 7. *Let $\mathbf{t} = t_1 \dots t_m$ be a sequence which fixes the direction of \mathbf{s} . Fix a character $c \in \Sigma$. For each $i = 1, \dots, m+1$, let $\gamma_i = \min\{\max\{j \mid t_1 \dots t_{i-1} c^j t_i \dots t_m \prec \mathbf{s}\}, T-m\}$. Then, for each $i = 1, \dots, m+1$, 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 8. Note that the lemma 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 $\gamma_1 = \gamma_4 = 0$ and $\gamma_2 = \gamma_3 = 1$. For $c = a$, we have $\gamma_i = \min(27, 22) = 22$ for all i (since $T = 22$).

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

Lemma 9. *Let $\mathbf{s} = a^{x_1}b^{y_1} \dots a^{x_t}b^{y_t}$. Assume that there are $1 \leq i_1 < i_2 < \dots < i_q \leq t$ and $k \geq 0$, such that $x_{i_j} \geq T - B - k$ (resp. $y_{i_j} \geq T - A - k$) for each $j = 1, \dots, q$, 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*

$$\rho_a \leq n - B - q(T - B - k - 1) - 1 \quad (\text{resp. } \rho_b \leq n - A - q(T - A - k - 1) - 1).$$

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

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

4.1 The case where $A \geq T$

Since $A \geq T = \lceil \frac{2}{3}(n+1) \rceil$, 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 $\beta = \lceil \frac{n}{3} - \frac{2}{3} \rceil$, 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 7, 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., $\mathbf{w} = a^{x_i} b^{y_i} \dots a^{x_j}$. Moreover, let \mathbf{s}_{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}_{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 $\chi = 1, 2, 3, \dots, \beta$, 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 7, with \mathbf{t} we can find the value of y_k for each $k = i, \dots, j-1$. As a side effect, we also determine the value of x_k for $k = i+1, \dots, j$. Now we know that $b^L \mathbf{w}' b^R \prec \mathbf{s}$, where $\mathbf{w}' = a^{\chi^*} b^{y_i} a^{x_{i+1}} \dots b^{y_{j-1}} a^{x_j}$. 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 $r = 1, \dots, R$ and $\ell = 1, 2, 3, \dots$, in order to determine the values of x_k , for each $k = j+1, \dots, \rho$. 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 $k = j+1, \dots, \rho$, we also determine y_k for each $k = j+1, \dots, \rho$.

By an analogous procedure, we can determine \mathbf{s}_{left} and the first a -run of \mathbf{w} , i.e. all the values x_k , for $k = 1, \dots, i$, 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 $k = 1, \dots, i-1$.

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 9, 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, it 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 7 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 $k = 1, \dots, \rho_a$. The only problem now is to determine those runs of b 's which have length at least $T - A$.

Let i_1, \dots, 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 $B - \sum_{\ell \neq i_1, \dots, i_q} y_\ell = q(T - A)$, 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 $y_{i_1} \geq y_{i_2} \geq \dots \geq y_{i_q}$ 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 $\chi = T - A + 1, T - A + 2, \dots, y_{i_2} + 1$, 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 cases 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}_χ , 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 $j = 3, \dots, q$, 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}_χ 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}_χ with $\chi \leq X$ would

answer *yes* because it would be 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}_χ for $\chi = X + 1$. If the answer is *yes*, we continue with $X + 2, X + 3 \dots$ until we receive the first *no*, and we are done, since the last χ where \mathbf{t}_χ 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 $\xi = T - A, T - A + 1, \dots, y_{i_2}$, we have

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

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

We can ask queries \mathbf{t}'_ξ 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}'_ξ 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:

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

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 7 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 $k = 1, \dots, \rho_b$.

Let $1 \leq i_1 < i_2 < \dots < i_q \leq \rho_a$ 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 only 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 9, we have $\rho_a \leq 3$, thus it follows that $\rho_a = 3$. Let $\mathbf{t} = ababa$, and $c = a$. By Lemma 7 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 9, 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 $\chi = 1, 2, 3, \dots$, 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 $\chi = 1, 2, \dots$, 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 5, 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 shape

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

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 $i = 1, 2, \dots, 2\rho$, 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 $i = 1, \dots, \rho$. 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, whose proof can be found in the Appendix.

Lemma 10. *Fix $i < \rho$ and assume that for each $k = 1, \dots, i-1$, 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.*

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 10, 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:

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

where $\mathbf{u} = a^{x_1}b^{y_1} \dots a^{x_{i-1}}b^{y_{i-1}}$ 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 10). 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.



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

The case $A = B = \frac{n}{2}$, $s_1 \neq s_n$, $x_i, y_{\rho-i+1} < T - \frac{n}{2}$. With reference to (4), we can use a recursive argument to show how to determine s' . Let $n' = |s'|$. Note that $|s'|_a > |s'|_b$ and that s' starts with a b and ends with an a .

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

$$t = a^{|u|_a} t'_+ a^{|u|_b} \quad (5)$$

Lemma 11. *Let t be defined as in Eq. (5). Then, it holds that*

1. $t \prec_{RC} s$ if and only if $t' \prec_{RC} s'$.
2. If $|t'| \leq \frac{2(n'+1)}{3}$, then $|t| \leq \frac{2(n+1)}{3}$.

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

The case $A = B = \frac{n}{2}$, $s_1 \neq s_n$, $x_i, y_{\rho-i+1} \geq T - \frac{n}{2}$. Notice that, because of the assumption $n \geq 8$ and $T - \frac{n}{2} \leq x_i, y_{\rho-i+1}$, it follows that $x_i + y_{\rho-i+1} \geq 4$. We have $|s'| = n - 2|u| - x_i - y_{\rho-i+1}$. This implies

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

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

The only requirement is that 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 $|\mathbf{s}'|_a \geq |\mathbf{s}'|_b$.)

The case $A = B = \frac{n}{2}$, $s_1 \neq s_n$, $x_i < T - \frac{n}{2}$, $y_{\rho-i+1} \geq T - \frac{n}{2}$. In order to determine ρ and $x_{i+1}, \dots, x_{\rho-i+1}$, we can use the query

$$\mathbf{t}_\chi = a^{|\mathbf{u}|_a + x_i} b a^{\chi} b a^{\frac{n}{2} - |\mathbf{u}|_a - x_i - \chi} \quad (7)$$

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}_χ 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}_χ as follows: We ask \mathbf{t}_χ for each $\chi = 1, 2, 3, \dots$, 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 that this implies $x_{i+1} = \chi_1$. We now continue asking query \mathbf{t}_χ for each $\chi = \chi_1 + 1, \chi_1 + 2, \dots$. 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 $j = 1, \dots, \rho - i + 1$, let χ_j be the value of χ when we receive the i th 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}_χ 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 $j = i + 1, \dots, \rho - i + 1$ and, in particular, we determine ρ .

Now by Lemma 7, 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 $j = i, \dots, \rho - i$ 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, \dots, i_q , be such that $y_{i_j} \geq T - \frac{n}{2}$, for each $j = 1, \dots, q$. 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 $n - \sum_{\ell \notin \{i_1, \dots, i_q\}} y_\ell = q(T - \frac{n}{2})$, 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:

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

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 $t \leq \frac{n}{2} - \frac{2n}{6} + \frac{2}{3} = \frac{n}{6} + \frac{2}{3} \leq y_{\rho-i+1}$. 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

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

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.

References

1. A. Carpi and A. de Luca. Words and special factors. *Theor. Comput. Sci.*, 259(1-2):145–182, 2001.
2. A. de Luca. On the combinatorics of finite words. *Theor. Comput. Sci.*, 218(1):13–39, 1999.
3. M. Dudík and L. J. Schulman. Reconstruction from subsequences. *J. Comb. Theory, Ser. A*, 103(2):337–348, 2003.
4. P. L. Erdős, P. Ligeti, P. Sziklai, and D. C. Torney. Subwords in reverse-complement order. In *Combinatorial and Algorithmic Foundations of Pattern and Association Discovery*, 2006.
5. G. Fici, F. Mignosi, A. Restivo, and M. Sciortino. Word assembly through minimal forbidden words. *Theor. Comput. Sci.*, 359(1-3):214–230, 2006.
6. I. Krasikov and Y. Roditty. On a reconstruction problem for sequences. *Journal of Combinatorial Theory*, 77:344–348, 1997.
7. V. I. Levenshtein. Efficient reconstruction of sequences. *IEEE Transactions on Information Theory*, 47(1):2–22, 2001.
8. P. Pevzner. l -tuple DNA sequencing: Computer analysis. *Journal of Biomolecular Structure and Dynamics*.
9. C. Piña and C. Uzcátegui. Reconstruction of a word from a multiset of its factors. *Theor. Comput. Sci.*, 400(1-3):70–83, 2008.
10. F. P. Preparata. Sequencing-by-hybridization revisited: The analog-spectrum proposal. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 1(1):46–52, 2004.
11. M.-P. Schützenberger and I. Simon. *Combinatorics on Words*, by M. Lothaire, chapter 6: Subwords. 1983.
12. I. Simon. Piecewise testable events. In *Automata Theory and Formal Languages*, pages 214–222. Springer, 1975.
13. S. Skiena and G. Sundaram. Reconstructing strings from substrings. *Journal of Computational Biology*, 2(2):333–353, 1995.
14. D. Tsur. Tight bounds for string reconstruction using substring queries. In *APPROX-RANDOM*, pages 448–459, 2005.

Appendix

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

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 $a^{2k+r} \prec_{\text{RC}} \mathbf{s}_1$ but $a^{2k+r} \not\prec_{\text{RC}} \mathbf{s}_2$ whilst for any string $\mathbf{t} \prec_{\text{RC}} \mathbf{s}_1$ such that $|\mathbf{t}| \leq 2k + r - 1$ it holds that $\mathbf{t} \prec_{\text{RC}} \mathbf{s}_2$.

Unbounded query size (Section 3)

To prove Theorem 3, we first need an easy lemma, which extends Lemma 14 in [13] to our problem.

Lemma 12. *Let \mathbf{s} be an unknown string over the paired alphabet $\Sigma = \{a_1, \bar{a}_1, \dots, a_\delta, \bar{a}_\delta\}$, (where we have explicitly specified the complementing pairs). Let $\mathbf{u}, \mathbf{v} \prec_{\text{RC}} \mathbf{s}$ 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 $\mathbf{u}, \mathbf{v} \prec_{\text{RC}} \mathbf{w}$ and $\mathbf{w} \prec_{\text{RC}} \mathbf{s}$ 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 $\mathbf{u} = u_1 u_2 \dots u_k$ and $\mathbf{v} = v_1 v_2 \dots v_m$, 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 $0 \leq i_1 \leq i_2 \leq \dots \leq i_{k+1} \leq k+1$ s.t. $\mathbf{w} = v_1 \dots v_{i_1} u_1 v_{i_1+1} \dots v_{i_2} u_2 \dots u_k v_{i_k+1} \dots v_{i_{k+1}}$ is a subsequence of \mathbf{s} . (For the sake of conciseness, 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 $v_1 \dots v_{i_1} u_1 v_{i_1+1} u_2 u_3 \dots u_k$, $v_1 \dots v_{i_1} u_1 v_{i_1+1} v_{i_1+2} u_2 u_3 \dots u_k$, etc. to determine i_2 . Proceeding in this way, we will find all i_j 's, 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 $\mathbf{w} \prec_{\text{RC}} \mathbf{s}$. 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 $\mathbf{w} \prec_{\text{RC}} \mathbf{s}$, as claimed. The total number of queries is thus at most $2(|\mathbf{u}| + |\mathbf{v}| + 1)$.

Theorem 3 *There exists an algorithm for reconstructing a string using $\Theta(n \log |\Sigma|)$ RC-subsequence queries of unbounded length.*

Proof. We first give the algorithm for the case where $|\Sigma| = 2$. Let $\Sigma = \{a, b\}$ where $b = \bar{a}$. Let $M = \max\{j \geq 0 \mid a^j \prec_{\text{RC}} \mathbf{s}\}$. Clearly, we have $M = \max_{c=a,b} |\mathbf{s}|_c$. Notice that we can determine M by asking query $\mathbf{t}_\chi^{(0)} = a^\chi$, for $\chi = \lceil \frac{n}{2} \rceil, \lceil \frac{n}{2} \rceil + 1, \dots$, until we get the first negative answer, implying that $M = \chi - 1$. In particular, we need exactly $M + 1$ such queries.

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

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

In perfect analogy with what we did above, we can determine y_i , by asking the query $\mathbf{t}_\chi^{(i)} = b^{y_1} a b^{y_2} a \dots b^{y_{i-1}} a b^\chi a^{M-i+1}$, for each $\chi = 1, 2, \dots$, 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 \mathbf{s}' , i.e., there exists no \mathbf{t}' such that $\mathbf{t} \prec \mathbf{t}' \prec_{\text{RC}} \mathbf{s}$. 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 $\mathbf{t} \prec_{\text{RC}} \mathbf{s}$, it follows that there exists a sequence $\mathbf{t}' \neq \mathbf{t}$ such that $\mathbf{t} \prec \mathbf{t}' \prec_{\text{RC}} \mathbf{s}$, and $|\mathbf{t}'| = |\mathbf{t}| + 1$.

By the definition of M , it follows that $|\mathbf{t}|_a = |\mathbf{t}'|_a$, for otherwise $a^{M+1} \prec \mathbf{t}' \prec_{\text{RC}} \mathbf{s}$, 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 $\mathbf{t}' = b^{y'_1} a b^{y'_2} a \dots b^{y'_M} a b^{y'_{M+1}}$, where $y'_j \geq 0$, for each $j = 1, \dots, M + 1$. By the definition of \mathbf{t}' , there exists exactly one j such that $y'_j = y_j + 1$. It follows that $b^{y_1} a b^{y_2} a \dots b^{y_{j-1}} a b^{y_j+1} a^{M-j} \prec \mathbf{t}' \prec_{\text{RC}} \mathbf{s}$, which contradicts the definition of y_j .

Now let $|\Sigma| = 2\delta$, with $\delta > 1$. For each $i = 1, \dots, \delta$, let us denote by $\mathbf{s}_{|i}$ the longest subsequence of \mathbf{s} only containing characters from $\{a_i, \bar{a}_i\}$. We call $\mathbf{s}_{|i}$ the i 'th projection of \mathbf{s} .

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 $\frac{3}{2}A_i + 2$, where $A_i = |\mathbf{s}_{|i}|$.

Once we have identified the i 'th projection of \mathbf{s} (up to reverse complement), for each $i = 1, \dots, \delta$, we can use Lemma 12 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 12, 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 $2n \log \delta + 2(\delta - 1)$, since 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 the total number of queries for the complete algorithm is $\sum_{i=1}^{\delta} (\frac{3}{2}A_i + 2) + 2n \log \delta + 2(\delta - 1) = \frac{3}{2}n + 2n \log \delta + 4\delta - 2 = O(n \log |\Sigma|)$, using the fact that $|\Sigma| = 2\delta$ and the (natural) assumption that $|\Sigma| = O(n)$.

Missing proofs of Section 4.4

Lemma 10. Fix $i < \rho$ and assume that for each $k = 1, \dots, i-1$, 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

$$\begin{aligned} m_i &= \min \left\{ \chi = 1, 2, 3, \dots \mid \mathbf{t}_\chi = a^{x_1 + \dots + x_{i-1} + \chi} b a^{\frac{n}{2} - (x_1 + \dots + x_{i-1} + \chi)} \prec_{\text{RC}} \mathbf{s} \right\}, \\ \min \left\{ M_i, T - \frac{n}{2} \right\} &= \max \left\{ \chi = m_i, m_i + 1, \dots, T - \frac{n}{2} \mid \right. \\ &\quad \left. \mathbf{q}_\chi = a^{\frac{n}{2} - (y_1 + \dots + y_{i-1})} b^\chi a^{y_1 + \dots + y_{i-1}} \prec_{\text{RC}} \mathbf{s} \right\}. \end{aligned} \quad (8)$$

Using (8), one can determine the value m_i (resp. M_i) by asking the query \mathbf{t}_χ (resp. \mathbf{q}_χ) for increasing values of χ , until the first positive (resp. negative) answer. 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:

$$\begin{aligned} m_i &= \min \left\{ \chi = 1, 2, \dots \mid \mathbf{t}_\chi = b^{y_1 + \dots + y_{i-1} + \chi} a b^{\frac{n}{2} - (y_1 + \dots + y_{i-1} + \chi)} \prec_{\text{RC}} \mathbf{s} \right\}, \\ \min \left\{ M_i, T - \frac{n}{2} \right\} &= \max \left\{ \chi = m_i, m_i + 1, \dots, T - \frac{n}{2} \mid \right. \\ &\quad \left. \mathbf{q}_\chi = b^{\frac{n}{2} - (x_1 + \dots + x_{i-1})} a^\chi b^{x_1 + \dots + x_{i-1}} \prec_{\text{RC}} \mathbf{s} \right\}. \end{aligned} \quad (9)$$

This completes the proof of the lemma.

Lemma 11. Let \mathbf{t} be defined as in Eq. (5). Then, it holds that

1. $\mathbf{t} \prec_{\text{RC}} \mathbf{s}$ if and only if $\mathbf{t}' \prec_{\text{RC}} \mathbf{s}'$.
2. If $|\mathbf{t}'| \leq \frac{2(n'+1)}{3}$, then $|\mathbf{t}| \leq \frac{2(n+1)}{3}$.

Proof. 1. Let $\mathbf{t} \prec_{\text{RC}} \mathbf{s}$. First assume that $\mathbf{t} \prec \mathbf{s}$. Notice that \mathbf{t}'_+ starts with a b and ends with an 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,

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