Quadratic-backtracking Algorithm for String Reconstruction from Substring Compositions

Jayadev Acharya, UCSD, jacharya@ucsd.edu
Hirakendu Das, Yahoo Labs, hdas@yahoo-inc.com
Olgica Milenkovic, UIUC, milenkov@uiuc.edu
Alon Orlitsky, UCSD, alon@ucsd.edu
Shengjun Pan, Google, s1pan@eng.ucsd.edu

***Abstract*—Motivated by the problem of deducing the structure of proteins using mass-spectrometry, we study the reconstruction of a string from the multiset of its substring compositions. We specialize the backtracking algorithm used for the more general *turnpike problem* for string reconstruction. Employing well known results about transience of random walks in ≥ 3 dimensions, we show that the algorithm reconstructs random strings over alphabet size ≥ 4 with high probability in near-optimal quadratic time.**

I. Introduction

Proteins are long sequences made of 20 amino acids and the ordering of these building blocks determines their properties. A common technique for finding the amino-acid sequence is mass-spectrometry [1, 2]. It involves taking a large number of identical proteins, ionizing and randomly breaking them into substrings, and analyzing the resulting mixture to determine the substring weights. These weights are then used to infer the amino-acid sequence.

In [3], the following two assumptions were made to reduce the problem of reconstructing proteins from the mass-spectrometer measurements into a combinatorial string reconstruction problem.

A1 The compositions of a substring can be deduced from its weight. For example, let A, B, and C be three amino acids with respective weights 13, 7, and 4. A string weight of 11 clearly consists of one B and one C. Similarly, weight 18 implies two B's and one C. However, weight 20 could arise from one A and one B or from 5 C's, hence we cannot deduce the composition from the weights. It is assumed that such confusions never arise or can be resolved by other means.

A2 Each protein-sequence bond gets cut independently with the same probability. For example, if the sequence is ABC and the cut probability is p, then the partition $A|B|C$ is obtained with probability p^2, the partitions $A|BC$, and $AB|C$ are obtained with probability $p(1-p)$, and the partition ABC with probability $(1-p)^2$. This assumption ensures that the multiset of weights of all substrings is estimated reliably.

While the assumptions are strong, it gives rise to a simply stated combinatorial problem of string reconstruction from the multiset of its substring compositions.

The *composition*, also known as the *type* or *Parikh vector*, of a string s, denoted $w(s)$ is the multiset of its elements, namely the number of times each element appears, regardless of the order. For example, $w(BABCAA) = \{A, A, A, B, B, C\}$, equivalently denoted by A^3B^2C to indicate that the string consists of three A's, two B's, and one C.

For a string $s = s_1 s_2 \dots s_n \in \Sigma^n$, let $s_i^j \overset{\text{def}}{=} s_i s_{i+1} \dots s_j$. The *composition multiset* of s is

$$\mathcal{S}_s \overset{\text{def}}{=} \{w(s_i^j) : 1 \leq i \leq j \leq n\},$$

the multiset of compositions of all $\binom{n+1}{2}$ contiguous substrings of s. For example,

$$\mathcal{S}_{ACAB} = \{A, A, B, C, AB, AC, AC, A^2C, ABC, A^2BC\}.$$

Note that since these are multisets, there is no apriori information about the ordering among the compositions in $\mathcal{S}_s$. Also note that the reversal $s^* \overset{\text{def}}{=} s_n s_{n-1} \dots s_1$ of s trivially has the same composition multiset. The problem is therefore to reconstruct a string $s \in \Sigma^n$ or its reversal, given only $\mathcal{S}_s$.

In [3], the problem of unique reconstruction of a binary string from its composition multiset is studied. Analyzing small length strings, it is shown that strings s with certain combinatorial properties cannot be uniquely determined from $\mathcal{S}_s$. In [4], string reconstruction is related to the well known turnpike problem, where the locations of n highway exits need to be recovered from the multiset of their $\binom{n}{2}$ inter-exit distances. For example, inter-exit distances $1, 2, 3, 3, 5, 6$ correspond to exit locations $0, 1, 3, 6$. Building on an algebraic characterization for the turnpike problem [5], reconstruction is equivalently formulated as factorization of a bivariate polynomial related to the multiset. Applying results on factorization of cyclotomic polynomials, it is shown that if the length of the sequence is $p-1$ or $2p-1$ for some prime p, or 7, then the composition multiset uniquely determines the string up to reversal. For all other lengths, there exist strings whose reconstruction is not unique.

There are several related works that considered variations of the string reconstruction problem. Reconstruction of a string from a few of its substrings was considered in [6]. Reconstruction from subsequences, not necessarily contiguous, was considered in [7–10]. However, in their settings, the substrings or subsequences themselves, which include the order of their symbols, are given. By contrast, in our problem, for each substring we are given just the composition, neither the order of the symbols within it nor the substring's location in the original string. In another related work [11], sequences are characterized based on the set of compositions, *i.e.,* each repeated composition is assumed to be observed only once.

In Section II we state our results, in Section III we show some useful relations between s and $\mathcal{S}_s$, and use them to develop our reconstruction algorithm in Section IV. Finally, we provide the proofs in Section V.

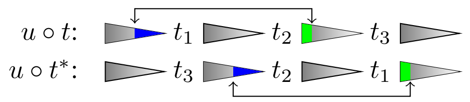
II. Results

We define two strings s and s' to be *equicomposable*, denoted $s \sim s'$, if they have the same composition multiset. Let

$$E_s \overset{\text{def}}{=} \{t : t \sim s\}$$

be the set of all strings equicomposable with s.

As a non-trivial example of equicomposable strings, consider the following construction shown in [3, 4]. The *interleaving* of string u with the string $t = t_1 \dots t_m$ is the string $u \circ t \overset{\text{def}}{=} u\,t_1\,u\,t_2 \cdots t_m\,u$. Let $t^* \overset{\text{def}}{=} t_m t_{m-1} \dots t_1$ be the reversal of t. In [3, 4], it is shown that for any strings u and t, $\mathcal{S}_{u \circ t} = \mathcal{S}_{u \circ t^*}$, *i.e.,* $u \circ t \sim u \circ t^*$. We demonstrate this fact when t has length 3 in the following figure, in which the triangle denotes u. The general result follows by showing a similar bijection from substrings of $u \circ t$ to $u \circ t^*$ that preserves compositions.



From [4, 5], it is known that

$$|E_s| \le \min\{2^{d(n+1)-1}, (n+1)^{1.23}\} \tag{1}$$

where $d(n)$ is the number of divisors of n. This shows that E is of polynomial size. However, for the general turnpike problem no polynomial time reconstruction algorithms are known [12].

Our problem is a special case of the turnpike problem and therefore the polynomial factorization formulation given by [4] can be used along with the LLL algorithm [13] to provide a polynomial time algorithm for reconstructing strings from their composition multiset. Using the current known best algorithms for polynomial factorization, this method can be shown to have complexity $O(n^{12})$ [14]. These algorithms consider factorizing all possible polynomials while our polynomials have a specific structure. Therefore, either the time complexity of such algorithms are high or do not provide guarantees for our problem [15].

Hence, instead of considering algebraic methods, we study an alternative, combinatorial approach of the backtracking algorithm proposed in [5] for the general turnpike problem. In [16], an example for was shown for which this algorithm has exponential complexity. However, we develop a backtracking algorithm for string reconstruction, and show that random strings coming from an alphabet Σ with $|\Sigma| \ge 4$ can be reconstructed in near-optimal quadratic time.

Our algorithm reconstructs strings from both ends, two symmetric positions at a time: (s_1, s_n), followed by (s_2, s_{n-1}) and so on. In *state* i, given the reconstruction s_1^{i-1} and s_{n-i+2}^n up to this state, it tries to extend it and reconstruct s_i and s_{n-i+1} that is consistent with the given composition multiset. However, as shown later, at times there may be more than one choice for (s_i, s_{n-i+1}). In that case, one of them is chosen to proceed to next states. If at some state in future, it turns out that no reconstruction is feasible, then the algorithm backtracks to the last fork and considers other choices, backtracking further if necessary.

Our algorithm relies on compositions from both ends. We therefore define two terms that we use to provide performance guarantees. For a string s, let

$$\ell_s \overset{\text{def}}{=} \left|\{i < n/2 : w(s_1^i) = w(s_{n+1-i}^n) \text{ and } s_{i+1} \neq s_{n-i}\}\right|$$

be the number of substrings from starting at the ends having the same composition, and the next two symbols are distinct. We assume that for $i = 0$, empty strings have the same composition. Let

$$L_s \overset{\text{def}}{=} \max_{t \in E_s} \ell_t$$

be the largest value of ℓ over all strings in E_s.

In Section IV, we describe the backtracking algorithm for string reconstruction, and then modify it slightly to give an algorithm that works with high probability for random strings over alphabet size ≥ 4. The next result bounds the complexity of reconstruction.

Theorem 1. *The backtracking algorithm, given $\mathcal{S}_s$, in time $O(2^{\ell_s} n^2 \log n)$ outputs a subset of E_s that contains s. If run until time $O(2^{L_s} n^2 \log n)$, it outputs E_s.*

The problem then reduces to bounding ℓ for random strings. Using well known results from random walk theory, we show that ℓ_s are bounded by a geometric distribution for alphabet size ≥ 4. This will be used to prove the following result.

Theorem 2. *For a random string s over an alphabet of size $k \ge 4$ the backtracking algorithm with probability $1 - \delta$ outputs a subset of E_s containing s in time $O_{\delta,k}(n^2 \log n)$*[1].

For reconstructing the entire set E_s, we show that L_s is with high probability bounded above by a logarithmic function.

Theorem 3. *For a random string s over an alphabet of size ≥ 4 the backtracking algorithm with probability $1-\delta$ outputs E_s in time $O_\delta\left(n^{\frac{1.23}{\log k}} n^2 \log n\right)$.*

We note that the algorithm also works for alphabet size 3, with weaker guarantees, *e.g.,* we can show using similar tools that,

[1] $g(n, \delta, k) = O_{\delta,k}(f(n))$ if $g(n) < C(\delta, k) f(n)$, where $C(\delta, k)$ is a function independent of n

Theorem 4. *The backtracking algorithm for random strings over alphabet size 3 outputs a subset of E_s containing s in time $O\left(n^{\frac{0.6}{\delta}}\right)$.*

III. Ancillary results

We first present some properties of $\mathcal{S}_s$ that will be used to design an efficient algorithm. The next lemma shows that the composition multiset determines the set $\{s_i, s_{n+1-i}\}$ of symbols at the symmetric positions i and $(n+1-i)$ for $i = 1, 2, \ldots, \lceil \frac{n}{2} \rceil$. If along with the set of symbols, we know of the association, namely instead of $\{s_i, s_{n+1-i}\}$ we know the pair (s_i, s_{n+1-i}), the string is determined trivially.

Lemma 5. *For every s, $\mathcal{S}_s$ determines $\{s_i, s_{n+1-i}\}$ for $i = 1, 2, \ldots, \lceil \frac{n}{2} \rceil$.*

Proof: Let the union of compositions be their union as multisets. For example, $A^2B \cup ABC^2 \cup AC = A^4B^2C^3$. For a string s, let M_i denote the union of the compositions of all substrings of length i. For example, for ABAC, $M_1 = A^2BC$, $M_2 = A^3B^2C$. Note that all M_i's can be easily determined from the string, and it can be shown that for $1 \le i \le \lfloor n/2 \rfloor$, $M_{n+1-i} = M_i$. For a multiset S, let $j \cdot S$ be the j-fold union $S \cup \ldots \cup S$. It is easy to see that

$$M_2 \cup \{s_1, s_n\} = 2 \cdot M_1,$$

hence $\{s_1, s_n\}$, can be deduced from $\mathcal{S}_s$. More generally, for $i = 2, \ldots, \lceil \frac{n}{2} \rceil$,

$$M_i \cup \{s_{i-1}, s_{n-i+2}\} \cup 2 \cdot \{s_{i-2}, s_{n-i+3}\} \cup \ldots \cup (i-1) \cdot \{s_1, s_n\} = i \cdot M_1.$$

Using this equation inductively over $i = 2, \ldots, \lceil \frac{n}{2} \rceil$, yields all multisets $\{s_i, s_{n+1-i}\}$. ∎

The backtracking algorithm reconstructs the string from both ends. We say that the algorithm is at state i, if there is a possible reconstruction of s^i and s^n_{n+1-i}, *i.e.,* we have potential reconstruction of the initial and final i symbols. It next decides on the symbols s_{i+1} and s_{n-i}. We now describe a sufficient condition under which there is a unique reconstruction, *i.e.,* we can determine (s_{i+1}, s_{n-i}) (not just the set $\{s_{i+1}, s_{n-i}\}$).

For $1 \le i < \lceil n/2 \rceil$, let $\mathcal{T}_i$ be the collection of compositions of strings s^k_j where $j, k \le i$, or $j, k \ge n+1-i$, or $j \le i+1 \le n-i \le k$, namely the collection of compositions of all strings that are either on "one side" of s^{n-i}_i or "straddle" s^{n-i}_{i+1}. The next lemma shows that the composition of the whole string, along with the strings s^i and s^n_{n+1-i} determine $\mathcal{T}_i$.

Lemma 6. *$\mathcal{S}_s$, s^i, and s^n_{n+1-i} determine $\mathcal{T}_i$.*

Proof: $\mathcal{T}_i$ consists of compositions of three types of strings, those that are substrings of s^i_1, that are substrings of s^n_{n+1-i}, and substrings that cover all symbols in between, *i.e.,* s^{n-i}_{i+1}. The first and last i symbols determine the compositions of the first two type of strings. The third type of strings are those that contain the symmetric string s^{n-i}_{i+1}, and by Lemma 5 we can determine its composition. Knowing this composition and the first and last i symbols yields the multiset of such strings. ∎

We use the two lemmas to help reconstruct the string. Recall that $\mathcal{S}_s$ determines the multiset $\{s_{i+1}, s_{n-i}\}$.

Lemma 7. *If $w(s^i_1) \ne w(s^n_{n+1-i})$, then $\mathcal{S}_s$, s^i_1, and s^n_{n+1-i} determine the pair (s_{i+1}, s_{n-i}).*

Proof: By Lemma 6, we can determine $\mathcal{T}_i$. Consider the two longest compositions in $\mathcal{S}_s \setminus \mathcal{T}_i$. They correspond to the length-$(n-i-1)$ strings s^{n-i-1}_1 and s^n_{i+2}. The complements of these two compositions are therefore the compositions of s^{i+1} and s^n_{n-i}.

By Lemma 5, we can also derive the multiset $\{s_{i+1}, s_{n-i}\}$. If $s_{i+1} = s_{n-i}$, then we can determine s^{i+1} and s^n_{n-i}. Otherwise, $s_{i+1} \ne s_{n-i}$, and since $w(s^i_1) \ne w(s^n_{n+1-i})$, it is easy to see that $\{w(s^i_1) \cup w(s_{i+1}), w(s_{n-i}) \cup w(s^n_{n+1-i})\} \ne \{w(s^i_1) \cup w(s_{n-i}), w(s_{i+1}) \cup w(s^n_{n+1-i})\}$, hence we can determine the pair (s_{i+1}, s_{n-i}). ∎

We now describe the reconstruction algorithm.

IV. Algorithm

We modify the backtracking algorithm of [5] for the turnpike problem and present an algorithm for reconstructing strings from their composition multiset.

The algorithm reconstructs the string by sequentially deciding on the values of a symmetric pair of symbols s_i and s_{n+1-i}.

The algorithm first determines $s_1 s_2$ and $s_{n-1} s_n$, uniquely up to reversal, as follows By Lemma 5, we know the multiset $\{s_1, s_n\}$ and can arbitrarily decide which is s_1 and which is s_n. It next determines s_2 and s_{n-1}. Again by the lemma we know $\{s_2, s_{n-1}\}$, and if $s_1 = s_n$ we can decide on s_2 and s_{n-1} arbitrarily, while if $s_1 \ne s_n$, by Lemma 7, we can determine s_2 and s_{n-1}.

For $\{s_3, s_{n-2}\}$, the ends s^2_1 and s^n_{n-1} can differ, while their weights could be the same. In such cases Lemma 7 cannot be applied. However if $s_3 = s_{n-2}$, which can be determined from the Lemma 5, we can still determine the bits. In other words, for all points at which $s_i = s_{n+1-i}$, the algorithm sails smoothly. Therefore, from this point on, when the algorithm is in state i, and $w(s^i_1) = w(s^n_{n+1-i})$ but $s_{i+1} \ne s_{n-i}$, it guesses one of the two possibilities for (s_{i+1}, s_{n-i}) and proceeds, while keeping track of the number t and locations $i_1 < i_2 < \ldots < i_t$ of locations where guesses were made. After determining the next two symbols, the algorithm finds $\mathcal{T}_{i+1}$, and update $i \to i+1$, and both can be accomplished in linear time. It then checks whether $\mathcal{T}_i \subseteq \mathcal{S}_s$ as multisets. If at some point $\mathcal{T}_i \subsetneq \mathcal{S}_s$, the algorithm realizes it has done a mistake at some state, and *backtracks*. It changes its guess at location i_t by swapping s_{i_t} and s_{n+1-i_t}, changes t to $t-1$, and restarts reconstruction from location $i_t + 1$. The process continues until the whole string is reconstructed, namely $i = \lceil \frac{n}{2} \rceil$ and $\mathcal{T}_{\lceil n/2 \rceil} = \mathcal{S}_s$. If at that point there are $t \ge 1$ guesses, then as before the algorithm backtracks to guess t and tries to find additional strings in E_s.

We make minor modifications to the algorithm to ensure efficient reconstruction of random strings.

The algorithm induces a tree where nodes represent locations at which there are two possible reconstructions. The procedure described above does a depth-first search. Instead we could also do a breadth first search, where we consider all branches at once, namely at any time, all potential reconstructions correspond to level t or $t+1$. This implies that given $\mathcal{S}_s$, the algorithm is able to find s before depth $\ell_s + 1$.

V. Proofs

A. Proof of Theorem 1

We now analyze the algorithm's complexity. We assume an arbitrary order over the elements of Σ. This introduces a lexicographical ordering over compositions of strings on Σ. We use Red-Black tree [17] to store multisets of compositions. The advantage of this data structure is that insertion, deletion and search all require time $O(\log n)$, where n is the size of the data.

Note that $\mathcal{T}_{i+1}$ is obtained by adding to $\mathcal{T}_i$ compositions of substrings that have an endpoint at s_{i+1} or s_{n-i}. In particular, at most $2n$ compositions are added, requiring $O(n \log n)$ time. For each branch, we keep a copy of $\mathcal{S}_s$ and we prune it to populate $\mathcal{T}$, *i.e.,* while constructing $\mathcal{T}_{i+1}$ we simultaneously remove the new entries/compositions from the copy of $\mathcal{S}$ corresponding to this branch, requiring another $O(n \log n)$. When there are two possibilities for reconstruction at any stage, we make copies of $\mathcal{T}_i$ and $\mathcal{S}$ corresponding to that node and proceed along each. This step takes time $O(n^2 \log n)$.

The algorithm while reconstructing s does not "fork out" more then $\ell_s + 1$ times, therefore the number of branches before reconstructing s is at most 2^{ℓ_s}. Along each path we make $n/2$ iterations requiring a total time of $O(n^2 \log n)$. Combining these, the total complexity of reconstructing s is at most $O(2^{\ell_s} n^2 \log n)$.

To reconstruct E_s, we note that the number of "forks" is $\le L_s$. Therefore, a similar computation shows that the algorithm runs in time $O(2^{L_s} n^2 \log n)$.

B. Proof of Theorem 2

We bound the run time as a function of ℓ_s. We show that the distribution of ℓ_s for random strings over $|\Sigma| \ge 4$ is bounded by a geometric random variable. More precisely,

Lemma 1. *For $k \ge 4$, there exists $p_k < 1$ such that for a random string $s \in \Sigma^n$, and $m \ge 1$*

$$P(\ell_s \ge m) < p_k^m.$$

We first use this result to prove Theorem 2, and return to prove the lemma. The lemma can be restated as $P(\ell_s < m) > 1 - p_k^m$. Suppose the error probability is δ. Therefore, with probability $\ge 1-\delta$, a random string satisfies $\ell < \frac{\log \delta}{\log p_k}$. By Theorem 1, all such strings can be reconstructed in time

$$O(2^{\frac{\log \delta}{\log p_k}} n^2 \log n) = O_{\delta,k}(n^2 \log n).$$

We now prove the lemma using well known results on random walks. These are applications of Stirling's approximation of factorials.

Lemma 8 (Stirling's approximation)**.** *For any $n \ge 1$, there is a $\theta_n \in (\frac{1}{12n+1}, \frac{1}{12n})$ such that*

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\theta_n}.$$

We first apply this and bound the probability that two random strings have the same composition. A stronger version of the result that finds asymptotic equality is proved in [18], but is not required for our purpose.

Lemma 9. *For $|\Sigma| = k$, let s and t be two random length-n strings over Σ, then for $k \ge n$*

$$P\left(w(s) = w(t)\right) < \frac{n!}{k^n},$$

and for $k < n$

$$P\left(w(s) = w(t)\right) < \frac{k^{k/2} e^{1/12n}}{(2\pi n)^{(k-1)/2}}.$$

Proof: The probability that the symbols in Σ appear $i_1, \dots, i_k$ times respectively in a random length-n string is

$$\frac{1}{k^n}\binom{n}{i_1, \dots, i_k}.$$

Therefore, the probability that two random strings have the same composition is

$$\begin{aligned}
&\sum_{i_1+\ldots+i_k=n} \frac{1}{k^{2n}}\binom{n}{i_1,\dots,i_k}^2 \\
\le &\frac{1}{k^n}\max_{i_1,\dots,i_k}\binom{n}{i_1\dots i_k}\sum_{i_1+\ldots+i_k=n}\frac{1}{k^n}\binom{n}{i_1,\dots,i_k} \\
= &\frac{1}{k^n}\max_{i_1,\dots,i_k}\binom{n}{i_1\dots i_k},
\end{aligned}$$

where the last step follows from

$$\sum_{i_1+\ldots+i_k=n}\frac{1}{k^n}\binom{n}{i_1,\dots,i_k} = 1.$$

For $k \ge n$ it is easy to see that

$$\max_{i_1,\dots,i_k}\binom{n}{i_1\dots i_k} = n!.$$

Plugging this above proves the first part.

For $n \ge k$, note that

$$f(x) \overset{\text{def}}{=} \sqrt{2\pi x}\left(\frac{x}{e}\right)^x$$

is convex in $(1, \infty)$. Therefore,

$$\prod_{j=1}^{k} i_j! \overset{(a)}{>} \prod_{j=1}^{k} f(i_j) \overset{(b)}{>} \left(f\left(\frac{n}{k}\right)\right)^k = \left(\sqrt{2\pi\frac{n}{k}}\right)^k \left(\frac{n}{ke}\right)^n.$$

where (a) uses Stirling's approximation, and (b) follows from convexity of f.

$$\max_{i_1,\ldots,i_k} \binom{n}{i_1 \ldots i_k} < \frac{n!}{\left(f\left(\frac{n}{k}\right)\right)^k} < \frac{k^{k/2} e^{1/12n}}{(2\pi n)^{(k-1)/2}},$$

where in the last step we approximate $n!$.

■

Consider two uniformly random independent infinite strings s^∞ and t^∞ over Σ. Let F_m be the event that there are at least m *non-consecutive* integers $i_0 \overset{\text{def}}{=} 0 < i_1 < i_2 < \ldots < i_m$ such that for each $j \le m$, $w(s_1^{i_j}) = w(t_1^{i_j})$. After a location i_j at which $w(s_1^{i_j}) = w(t_1^{i_j})$ by independence the process is equivalent to starting at time 0. Note that $i_1 > 1$ for non-consecutiveness. It follows that $P(F_{j+1}|F_j) = P(F_1)$. Therefore,

$$P(F_m) = P(F_1)^m. \tag{2}$$

Let $M(s^\infty, t^\infty)$ denote the total number of non-consecutive integers for which $w(s_1^i) = w(t_1^i)$. Then by Equation (2),

$$\mathbb{E}[M] = \sum_{m \ge 1} P(M \ge m) = \sum_{m \ge 1} P(F_m) = \frac{P(F_1)}{1 - P(F_1)}.$$

However, if instead of non-consecutiveness, we only restrict $i_1 \ge 2$, Lemma 9 shows that

$$\mathbb{E}[M] \le \sum_{n=2}^{k} \frac{n!}{k^n} + \sum_{n=k+1}^{\infty} \frac{k^{k/2} e^{1/12n}}{(2\pi n)^{(k-1)/2}}.$$

The right hand side of this equation is finite for $k \ge 4$. In fact it decays as $1/k^2$ with k. This implies that $p_k \overset{\text{def}}{=} P(F_1) < 1$ and therefore for a random string s, Equation (2) gives

$$P(\ell_s > m) \le P(F_m) = p_k^m,$$

proving the lemma.

C. Proof sketch of Theorem 3 and Theorem 4

We use the fact that $|E_s| < (n+1)^{1.23}$ and therefore, an union bound shows that

Lemma 10. *With probability $\ge 1 - n^{1.23} p_k^m$, a random string over alphabet size $k \ge 4$ satisfies $L_s < m$.*

Applying this to Theorem 1 proves the result.

For alphabet size 3, a similar computation shows that $P(F_1) = 1$. However, we can show that ℓ_s has expected value $O(\log n)$ and then prove the theorem. Details are omitted due to the lack of space.

References

[1] T. E. Creighton, *Proteins: Structures and Molecular Properties*, 2nd ed. W. H. Freeman, 1992.

[2] D. W. Mount, *Bioinformatics: Sequence and Genome Analysis*, 2nd ed. Cold Spring Harbor Laboratory Press, 2001.

[3] H. Das, O. Milenkovic, and A. Orlitsky, "Order from disorder," Information Theory and Applications Workshop, 2009.

[4] J. Acharya, H. Das, O. Milenkovic, A. Orlitsky, and S. Pan, "On reconstructing a string from its substring compositions," in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, 2010, pp. 1238–1242.

[5] S. S. Skiena, W. D. Smith, and P. Lemke, "Reconstructing sets from interpoint distances (extended abstract)," in *Symposium on Computational Geometry*, 1990, pp. 332–339.

[6] D. Margaritis and S. S. Skiena, "Reconstructing strings from substrings in rounds," in *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, 1995, pp. 613–620.

[7] V. I. Levenshtein, "Efficient reconstruction of sequences from their subsequences or supersequences," *Journal of Combinatorial Theory, Series A*, vol. 93, no. 2, pp. 310–332, 2001.

[8] M. Dudik and L. J. Schulman, "Reconstruction from subsequences," *Journal of Combinatorial Theory, Series A*, vol. 103, no. 2, pp. 337–348, 2003.

[9] T. Batu, S. Kannan, S. Khanna, and A. McGregor, "Reconstructing strings from random traces," in *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, 2004, pp. 910–918.

[10] K. Viswanathan and R. Swaminathan, "Improved string reconstruction over insertion-deletion channels," in *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2008, pp. 399–408.

[11] G. Fici and Z. Liptk, "On prefix normal words," in *Developments in Language Theory*, ser. Lecture Notes in Computer Science, G. Mauri and A. Leporati, Eds. Springer Berlin Heidelberg, 2011, vol. 6795, pp. 228–238.

[12] T. Dakic, "On the turnpike problem," Ph.D. dissertation, Simon Fraser University, 2000.

[13] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Ann.*, vol. 261, pp. 515–534, 1982.

[14] S. Gao, "Factoring multivariate polynomials via partial differential equations." *Math. Comput.*, vol. 72, no. 242, pp. 801–822, 2003.

[15] V. Shoup, "On the deterministic complexity of factoring polynomials over finite fields," *Inform. Process. Lett*, vol. 33, pp. 261–267, 1990.

[16] Z. Zhang, "An exponential example for a partial digest mapping algorithm." *Journal of Computational Biology*, vol. 1, no. 3, pp. 235–239, 1994.

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. The MIT Press, 2001.

[18] L. B. Richmond and J. O. Shallit, "Counting abelian squares," *Electronic Journal of Combinatorics*, vol. 16, no. 1, pp. 317–348, 2009.